

ACSPL+ Commands & Variables

Reference Guide

May 2022

Document Revision: 3.12

ACSPL+ Commands & Variables

Release Date: May 2022

COPYRIGHT

© ACS Motion Control Ltd., 2022. All rights reserved.

Changes are periodically made to the information in this document. Changes are published as release notes and later incorporated into revisions of this document.

No part of this document may be reproduced in any form without prior written permission from ACS Motion Control.

TRADEMARKS

Windows and Intellisense are trademarks of Microsoft Corporation.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Any other companies and product names mentioned herein may be the trademarks of their respective owners.

PATENTS

Israel Patent No. 235022

US Patent Application No. 14/532,023

Europe Patent application No.15187586.1

Japan Patent Application No.: 2015-193179

Chinese Patent Application No.: 201510639732.X

Taiwan(R.O.C.) Patent Application No. 104132118

Korean Patent Application No. 10-2015-0137612

www.acsmotioncontrol.com

support@acsmotioncontrol.com

sales@acsmotioncontrol.com

NOTICE

The information in this document is deemed to be correct at the time of publishing. ACS Motion Control reserves the right to change specifications without notice. ACS Motion Control is not responsible for incidental, consequential, or special damages of any kind in connection with using this document.

Revision History

| Date | Revision | Description |
|----------------|------------|--------------------------------------------------------------------------------------------------------------------|
| May 2022 | 3.12 | New Release, Error mapping, FOE, Modulo, other functions |
| February 2022 | 3.11.01.06 | Corrections to ARC1, ARC2, LINE, switches |
| February 2022 | 3.11.01.05 | Corrections to PRATE |
| January 2022 | 3.11.01.04 | Correction to HOME entry, INTERRUPT correction |
| January 2022 | 3.11.01.03 | Remove SET/GETCONF(29) |
| December 2021 | 3.11.01.02 | Remove AERR peg_engine instead of axis in PEG_I, PEG_R, etc. |
| December 2021 | 3.11.01.01 | Document /q switch for XSEG, PTP, other motion commands Local Coordinates function explanations XSEG example |
| November 2021 | 3.11.01 | PEG & MARK Improvements, MFLAGSX.#SATPROT, AIN/AOUT corrections |
| September 2021 | 3.11 | New and updated functions with ADK Release |
| June 2021 | 3.10.01 | Formatting Corrections, ASSIGNPOUTS for XXMsa |
| April 2021 | 3.10 | SLCRAT, SLVRAT corrections Note M&S State for MTIMEA/B/C Document SET/GETCONF(270) Changes for V3.10 |
| December 2020 | 3.03 | INSHAPEON example if CTIME<1 |
| September 2020 | 3.02 | Remove reference to obsolete SPiiPlus PCI device |
| July 2020 | 3.01 | Fix Error Tables |
| June 2020 | 3.00 | Changes supporting ADK v3.00 |
| September 2019 | 2.70.10 | Corrections to XCURV, XCURI and related variables |






| Date | Revision | Description |
|----------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| July 2019 | 2.70.02 | SLCPA is obsolete, removed from documentation |
| June 2019 | 2.70.01 | Formatting corrections, examples for functions, fixed links for system configuration variables, moved stepper loop variables to servo loop section |
| April 2019 | 2.70 | GCODE Errors Many new functions, commands, and variables for new features. See version 2.70 release notes. |
| October 2018 | 2.60.10 | Updated AST bits for laser and SLEC module Added FOLLOW, UNFOLLOW, EXTFAC, FOLLOWCH Added error code 5042 to the list of motion termination errors |
| July 2018 | 2.60 | Updated for SPiiPlus ADK Suite v2.60 |
| January 2018 | 2.50.01 | Added laser control commands and functions. |
| December 2017 | 2.50 | Updated for SPiiPlus ADK Suite v2.50 |
| September 2017 | 2.40.01 | Updated SETSP function. |
| June 2017 | 2.40 | Updated for SPiiPlus ADK Suite v2.40 |
| December 2016 | 2.30.02 | Removed unsupported ServoBoost variables |
| October 2016 | 2.30.01 | Added support for Absolute Encoders to SLPROUT, SLVROUT, SLCROUT Replaced references to acsc_Write and acsc_Read with acsc_Transaction |
| September 2016 | 2.30.10 | Changed LINE1 to LINE, ARC2 to ARC1, ARC3 to ARC2, and ARC4 to ARC2 Updated XSEG, LINE, ARC1, and ARC2 for 6 axes support |
| August 2016 | 2.30 | Updated for SPiiPlus ADK Suite v2.30 |
| September 2014 | 01 | First Release |

Conventions Used in this Guide

Text Formats

| Format | Description |
|------------------------------------------|----------------------------------------|
| Bold | Names of GUI objects or commands |
| BOLD + UPPERCASE | ACSPL+ variables and commands |
| <code>Monospace + grey background</code> | Code example |
| <i>Italic</i> | Names of other documents |
| Blue | Hyperlink |
| [] | In commands indicates optional item(s) |
| | In commands indicates either/or items |

Flagged Text

| | |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
|  | Note - includes additional information or programming tips. |
|  | Caution - describes a condition that may result in damage to equipment. |
|  | Warning - describes a condition that may result in serious bodily injury or death. |
|  | Model - highlights a specification, procedure, condition, or statement that depends on the product model |
|  | Advanced - indicates a topic for advanced users. |

Related Documents

Documents listed in the following table provide additional information related to this document.

The most updated version of the documents can be downloaded by authorized users from [ACS Motion Control Resources](#) under "Downloads".

Online versions for all ACS software manuals are available to authorized users at [ACS Motion Control Knowledge Center](#).

| Document | Description |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>SPiiPlus C Library Reference</i> | <i>C++ and Visual Basic® libraries for host PC applications. This guide is applicable for all the SPiiPlus motion control products.</i> |
| <i>SPiiPlus COM Library Reference C</i> | <i>COM Methods, Properties, and Events for Communication with the Controller.</i> |
| <i>SPiiPlus .NET Library Reference</i> | <i>.NET Methods, Properties, and Events for Communication with the Controller.</i> |
| <i>SPiiPlusMMI Application Studio User Guide</i> | A complete guide for using the SPiiPlus MMI Application Studio and associated monitoring tools. |
| <i>SPiiPlus Utilities User Guide</i> | A guide for using the SPiiPlus User Mode Driver (UMD) for setting up communication with the SPiiPlus motion controller. |
| <i>SPiiPlus NT/DC Hardware Guide</i> | Technical description of the SPiiPlus NT/DC product line. |
| <i>SPiiPlus PDMnt Hardware Guide</i> | Technical description of the SPiiPlus PDMnt Network Interface. |
| <i>SPiiPlus SDMnt Hardware Guide</i> | Technical description of the SPiiPlus SDMnt Step Motor Drive Module. |
| <i>SPiiPlus UDMnt Hardware Guide</i> | Technical description of the SPiiPlus UDMnt Universal Drive Module. |
| <i>MC4U-CS Control Module Hardware Guide</i> | Technical description of the MC4U Control Module integrated motion control product line. |
| <i>HSSI Expansion Modules Guide</i> | High-Speed Synchronous Serial Interface (HSSI) for expanded I/O, distributed axes, and nonstandard devices. |

| Document | Description |
|----------------------------------------------------------|-----------------------------------------------------------------|
| <i>PEG and MARK Operations Application Notes</i> | Provides details on using the PEG commands in SPiiPlus systems. |

Table of Contents

| | |
|----------------------------------------------------------------------------------|----|
| 1. Introduction | 39 |
| 2. ACSPL+ Commands | 40 |
| 2.1 Axis Management Commands | 44 |
| 2.1.1 BREAK | 45 |
| 2.1.2 COMMUT | 46 |
| 2.1.3 CONNECT | 47 |
| 2.1.4 CSCREATE | 50 |
| 2.1.5 CSDESTROY | 53 |
| 2.1.6 DEPENDS | 54 |
| 2.1.7 DISABLE/DISABLEALL | 55 |
| 2.1.8 ENABLE/ENABLE ALL | 56 |
| 2.1.9 ENCINIT | 57 |
| 2.1.10 ENCREAD | 59 |
| 2.1.11 FCLEAR | 60 |
| 2.1.12 FOLLOW | 61 |
| 2.1.13 GO | 61 |
| 2.1.14 GROUP | 62 |
| 2.1.15 HALT | 63 |
| 2.1.16 HOME | 64 |
| 2.1.17 IMM | 66 |
| 2.1.18 KILL/KILLALL | 67 |
| 2.1.19 SAFETYCONF | 68 |
| 2.1.20 SAFETYGROUP | 70 |
| 2.1.21 SET | 70 |
| 2.1.22 SPLIT | 71 |
| 2.1.23 UNFOLLOW | 72 |
| 2.2 Predefined Homing Methods | 72 |
| 2.2.1 Homing Method 1: Homing on the negative limit switch and index pulse | 72 |
| 2.2.2 Homing Method 2: Homing on positive limit switch and index pulse | 72 |
| 2.2.3 Homing Method 17: Homing on Negative Limit Switch | 73 |
| 2.2.4 Homing Method 18: Homing on Positive Limit Switch | 73 |
| 2.2.5 Homing Method 33 and 34: Homing on the index pulse | 73 |

| | |
|---------------------------------------------------------------------------------|-----|
| 2.2.6 Homing Method 37: Homing on current position | 73 |
| 2.2.7 Homing Method 50: Negative Hard Stop and index pulse (ACS Specific) | 73 |
| 2.2.8 Homing Method 51: Positive Hard Stop and index pulse (ACS Specific) | 74 |
| 2.2.9 Homing Method 52: Negative Hard Stop (ACS Specific) | 74 |
| 2.2.10 Homing Method 53: Positive Hard Stop (ACS Specific) | 74 |
| 2.3 Interactive Commands | 74 |
| 2.3.1 DISP | 74 |
| 2.3.2 INP | 79 |
| 2.3.3 INTERRUPT | 81 |
| 2.3.4 INTERRUPTX | 83 |
| 2.3.5 SEND | 84 |
| 2.3.6 TRIGGER | 86 |
| 2.3.7 OUTP | 87 |
| 2.4 PEG and MARK Commands | 89 |
| 2.4.1 ASSIGNMARK | 89 |
| 2.4.2 ASSIGNPEG | 90 |
| 2.4.3 ASSIGNPOUTS | 92 |
| 2.4.4 GETPEGCOUNT | 94 |
| 2.4.5 PEG_I | 94 |
| 2.4.6 PEG_R | 95 |
| 2.4.7 STARTPEG | 98 |
| 2.4.8 STOPPEG | 99 |
| 2.5 Miscellaneous Commands | 99 |
| 2.5.1 AXISDEF | 100 |
| 2.5.2 DC | 102 |
| 2.5.3 STOPDC | 104 |
| 2.5.4 READ | 105 |
| 2.5.5 SPDC | 106 |
| 2.5.6 STOPSPDC | 109 |
| 2.5.7 WRITE | 110 |
| 2.5.8 SPINJECT | 111 |
| 2.5.9 STOPINJECT | 112 |
| 2.5.10 SPICFG | 112 |
| 2.5.10.1 SPIWRITE | 114 |

| | |
|------------------------------|-----|
| 2.5.11 SPIWRITE | 114 |
| 2.5.12 SPRT | 115 |
| 2.5.13 SPRTSTOP | 118 |
| 2.6 Motion Commands | 118 |
| 2.6.1 ARC1 | 120 |
| 2.6.2 ARC1 | 121 |
| 2.6.3 ARC1 | 124 |
| 2.6.4 ARC2 | 125 |
| 2.6.5 ARC2 | 126 |
| 2.6.6 ARC2 | 129 |
| 2.6.7 BPTP | 129 |
| 2.6.8 BPTPCalc | 132 |
| 2.6.9 BSEG...ENDS | 133 |
| 2.6.10 JOG | 134 |
| 2.6.11 LINE | 135 |
| 2.6.12 LINE | 136 |
| 2.6.13 LINE | 139 |
| 2.6.14 MASTER | 140 |
| 2.6.15 MPOINT | 141 |
| 2.6.16 MPTP...ENDS | 146 |
| 2.6.17 MSEG...ENDS | 149 |
| 2.6.18 PATH...ENDS | 151 |
| 2.6.19 POINT | 153 |
| 2.6.20 PROJECTION | 156 |
| 2.6.21 PTP | 158 |
| 2.6.22 PVSPLINE...ENDS | 160 |
| 2.6.23 SLAVE | 163 |
| 2.6.24 STOPPER | 164 |
| 2.6.25 TRACK | 165 |
| 2.6.26 XSEG...ENDS | 166 |
| 2.6.27 NURBS | 176 |
| 2.6.28 NPOINT | 178 |
| 2.6.29 SPATH | 180 |
| 2.6.30 SEGMENT | 183 |

| | |
|-------------------------------------------------|-----|
| 2.6.31 SMOVE | 183 |
| 2.6.32 Using ARC1, ARC2 and LINE Switches | 184 |
| 2.7 Program Flow Commands | 186 |
| 2.7.1 Assignment Command | 186 |
| 2.7.2 BLOCK...END | 188 |
| 2.7.3 CALL | 189 |
| 2.7.4 GOTO | 189 |
| 2.7.5 IF, ELSEIF, ELSE...END | 190 |
| 2.7.6 INPUT | 192 |
| 2.7.7 LOOP...END | 193 |
| 2.7.8 ON...RET | 194 |
| 2.7.9 TILL | 195 |
| 2.7.10 WAIT | 196 |
| 2.7.11 WHILE...END | 196 |
| 2.8 Program Management Commands | 197 |
| 2.8.1 DISABLEON | 198 |
| 2.8.2 ENABLEON | 198 |
| 2.8.3 PAUSE | 198 |
| 2.8.4 RESUME | 199 |
| 2.8.5 START | 199 |
| 2.8.6 STOP/STOPALL | 201 |
| 2.9 Ethernet/IP ACSPL+ Support Commands | 201 |
| 2.9.1 EIPGETATTR | 201 |
| 2.9.2 EIPGETIND1 | 203 |
| 2.9.3 EIPGETIND2 | 204 |
| 2.9.4 EIPGETTAG | 204 |
| 2.9.5 EIPSETASM | 205 |
| 2.10 Laser Control Commands | 206 |
| 2.10.1 LCENABLE | 206 |
| 2.10.2 LCDISABLE | 206 |
| 2.11 Input Shaping Commands | 206 |
| 2.11.1 INSHAPEON | 207 |
| 2.11.2 INSHAPEOFF | 208 |
| 3. ACSPL+ Variables | 209 |

| | |
|----------------------------------------|-----|
| 3.1 Axis Configuration Variables | 222 |
| 3.1.1 AFLAGS | 224 |
| 3.1.2 ENTIME | 225 |
| 3.1.3 ESTBITS | 225 |
| 3.1.4 E2STBITS | 226 |
| 3.1.5 EMTBITS | 227 |
| 3.1.6 E2MTBITS | 227 |
| 3.1.7 MFF | 228 |
| 3.1.8 MFLAGS | 229 |
| 3.1.9 MFLAGSX | 236 |
| 3.1.10 MODULOMD | 238 |
| 3.1.11 PEGQUE | 242 |
| 3.1.12 SETTLE | 242 |
| 3.1.13 SLPMAX | 243 |
| 3.1.14 SLPMIN | 244 |
| 3.1.15 STEPF | 245 |
| 3.1.16 STEPW | 246 |
| 3.1.17 TARGRAD | 247 |
| 3.2 Brake Variables | 247 |
| 3.2.1 BOFFTIME | 248 |
| 3.2.2 BONTIME | 249 |
| 3.2.3 MBRKROUT | 250 |
| 3.2.4 VELBRK | 251 |
| 3.3 Feedback Variables | 252 |
| 3.3.1 E_AOFFS | 254 |
| 3.3.2 E_FREQ | 255 |
| 3.3.3 E2_AOFFS | 256 |
| 3.3.4 E2_FREQ | 256 |
| 3.3.5 E_FLAGS | 257 |
| 3.3.6 E2_FLAGS | 258 |
| 3.3.7 E_PAR_A | 259 |
| 3.3.8 E2_PAR_A | 260 |
| 3.3.9 E_PAR_B | 261 |
| 3.3.10 E2_PAR_B | 261 |

| | |
|-----------------------|-----|
| 3.3.11 E_PAR_C | 262 |
| 3.3.12 E2_PAR_C | 263 |
| 3.3.13 E_PAR_D | 264 |
| 3.3.14 E2_PAR_D | 264 |
| 3.3.15 E_PAR_E | 265 |
| 3.3.16 E2_PAR_E | 265 |
| 3.3.17 E_SCMUL | 266 |
| 3.3.18 E2_SCMUL | 267 |
| 3.3.19 E_TYPE | 267 |
| 3.3.20 E2_TYPE | 269 |
| 3.3.21 EFAC | 270 |
| 3.3.22 E2FAC | 271 |
| 3.3.23 EOFFS | 272 |
| 3.3.24 E2OFFS | 273 |
| 3.3.25 EPOS | 273 |
| 3.3.26 FVFIL | 274 |
| 3.3.27 F2ACC | 275 |
| 3.3.28 HOMEDEF | 275 |
| 3.3.29 HOMEVELI | 276 |
| 3.3.30 HOMEVELL | 277 |
| 3.3.31 RVFIL | 278 |
| 3.3.32 SCSOFFS | 279 |
| 3.3.33 SCCOFFS | 280 |
| 3.3.34 SC2COFFS | 281 |
| 3.3.35 SC2GAIN | 281 |
| 3.3.36 SC2PHASE | 282 |
| 3.3.37 SC2SOFFS | 282 |
| 3.3.38 SLEBIASA | 283 |
| 3.3.39 SLEBIASB | 284 |
| 3.3.40 SLEBIASC | 285 |
| 3.3.41 SLEBIASD | 286 |
| 3.3.42 SLABITS | 286 |
| 3.3.43 S2LABITS | 287 |
| 3.3.44 SCGAIN | 288 |

| | |
|-----------------------------------|-----|
| 3.3.45 SCPHASE | 288 |
| 3.4 Axis State Variables | 289 |
| 3.4.1 AST | 289 |
| 3.4.2 IND | 292 |
| 3.4.3 IST | 293 |
| 3.4.4 M2ARK | 294 |
| 3.4.5 MARK | 295 |
| 3.4.6 MST | 296 |
| 3.4.7 RMSM | 297 |
| 3.4.8 RMSD | 298 |
| 3.4.9 NST | 298 |
| 3.5 Safety Limits Variables | 300 |
| 3.5.1 CERRA | 301 |
| 3.5.2 CERRI | 302 |
| 3.5.3 CERRV | 303 |
| 3.5.4 DELV | 303 |
| 3.5.5 DELI | 304 |
| 3.5.6 E_ERR | 305 |
| 3.5.7 ERRA | 306 |
| 3.5.8 ERRI | 307 |
| 3.5.9 ERRV | 307 |
| 3.5.10 SLLIMIT | 308 |
| 3.5.11 SLLROUT | 309 |
| 3.5.12 SRLIMIT | 310 |
| 3.5.13 XACC | 311 |
| 3.5.14 XCURCDB | 311 |
| 3.5.15 XCURI | 312 |
| 3.5.16 XCURK | 313 |
| 3.5.17 XCURV | 314 |
| 3.5.18 XRMS | 315 |
| 3.5.19 XRMSD | 315 |
| 3.5.20 XRMSM | 316 |
| 3.5.21 XRMST | 317 |
| 3.5.22 XRMSTD | 318 |

| | |
|--------------------------------------|-----|
| 3.5.23 XRMSTM | 319 |
| 3.5.24 XSACC | 320 |
| 3.5.25 XVEL | 321 |
| 3.6 Data Collection Variables | 321 |
| 3.6.1 DCN | 322 |
| 3.6.2 DCP | 323 |
| 3.6.3 S_DCN | 323 |
| 3.6.4 S_DCP | 324 |
| 3.6.5 S_ST | 325 |
| 3.7 Input and Output Variables | 325 |
| 3.7.1 AIN | 326 |
| 3.7.2 AINOFFS | 327 |
| 3.7.3 AINSCALE | 328 |
| 3.7.4 AOUT | 328 |
| 3.7.5 DOUT | 329 |
| 3.7.6 EXTIN | 330 |
| 3.7.7 EXTOUT | 330 |
| 3.7.8 IN | 331 |
| 3.7.9 OUT | 332 |
| 3.7.10 SPIRXN | 333 |
| 3.7.11 SPIST | 333 |
| 3.8 Monitoring Variables | 334 |
| 3.8.1 BCODECFG | 335 |
| 3.8.2 BCODEUSG | 335 |
| 3.8.3 BGLOBCFG | 336 |
| 3.8.4 BGLOBUSG | 337 |
| 3.8.5 BSRCUSG | 338 |
| 3.8.6 BSRCCFG | 338 |
| 3.8.7 BVARUSG | 339 |
| 3.8.8 BVARCFG | 340 |
| 3.8.9 JITTER | 341 |
| 3.8.10 MSSYNC | 341 |
| 3.8.11 USGBUF | 341 |
| 3.8.12 USGTRACE | 342 |

| | |
|----------------------------|-----|
| 3.8.13 SOFTIME | 342 |
| 3.8.14 TIME | 343 |
| 3.8.15 USAGE | 343 |
| 3.8.16 USAGESIM | 344 |
| 3.9 Motion Variables | 344 |
| 3.9.1 ACC | 347 |
| 3.9.2 APOS | 348 |
| 3.9.3 APOSFILT | 349 |
| 3.9.4 CERRK | 349 |
| 3.9.5 DAPOS | 350 |
| 3.9.6 DEC | 351 |
| 3.9.7 DECOMP | 352 |
| 3.9.8 DELK | 352 |
| 3.9.9 FACC | 353 |
| 3.9.10 FPOS | 353 |
| 3.9.11 F2POS | 354 |
| 3.9.12 FVEL | 355 |
| 3.9.13 F2VEL | 355 |
| 3.9.14 FEEDRF | 355 |
| 3.9.15 GACC | 356 |
| 3.9.16 GJERK | 357 |
| 3.9.17 GMOT | 357 |
| 3.9.18 GMQU | 358 |
| 3.9.19 GMTYPE | 358 |
| 3.9.20 GPATH | 359 |
| 3.9.21 GPHASE | 360 |
| 3.9.22 GRTIME | 361 |
| 3.9.23 GSEG | 362 |
| 3.9.24 GSFREE | 363 |
| 3.9.25 GSNAP | 363 |
| 3.9.26 GVEC | 363 |
| 3.9.27 GVEL | 364 |
| 3.9.28 JERK | 364 |
| 3.9.29 KDEC | 365 |

| | |
|------------------------------------------------|-----|
| 3.9.30 LPOS | 366 |
| 3.9.31 MPOS | 367 |
| 3.9.32 MTIMEA | 367 |
| 3.9.33 MTIMEB | 368 |
| 3.9.34 MTIMEC | 369 |
| 3.9.35 NVEL | 370 |
| 3.9.36 PE | 371 |
| 3.9.37 PPOS | 372 |
| 3.9.38 PPOSCOMP | 372 |
| 3.9.39 PRFLTIME | 373 |
| 3.9.40 RACC | 374 |
| 3.9.41 RJERK | 374 |
| 3.9.42 ROFFS | 374 |
| 3.9.43 RPOS | 375 |
| 3.9.44 RPOSCOMP | 376 |
| 3.9.45 RPOSDEL | 376 |
| 3.9.46 RSNAP | 377 |
| 3.9.47 RVEL | 377 |
| 3.9.48 SETTLEA | 378 |
| 3.9.49 SETTLEB | 378 |
| 3.9.50 SLSFF | 379 |
| 3.9.51 SETTLEC | 379 |
| 3.9.52 SNAP | 380 |
| 3.9.53 STLTIMEA | 381 |
| 3.9.54 STLTIMEB | 382 |
| 3.9.55 STLTIMEC | 383 |
| 3.9.56 TARGRADA | 383 |
| 3.9.57 TARGRADB | 384 |
| 3.9.58 TARGRADC | 385 |
| 3.9.59 TPOS | 386 |
| 3.9.60 VEL | 387 |
| 3.10 Program Execution Control Variables | 388 |
| 3.10.1 ONRATE | 389 |
| 3.10.2 PCHARS | 389 |

| | |
|--------------------------------------|-----|
| 3.10.3 PERL | 390 |
| 3.10.4 PERR | 390 |
| 3.10.5 PEXL | 391 |
| 3.10.6 PFLAGS | 391 |
| 3.10.7 PLINES | 393 |
| 3.10.8 PRATE | 394 |
| 3.10.9 PST | 395 |
| 3.11 Safety Control Variables | 395 |
| 3.11.1 E_ERR | 397 |
| 3.11.2 ECALERR | 397 |
| 3.11.3 ECERR | 398 |
| 3.11.4 ECEXTERR | 398 |
| 3.11.5 ECEXTST | 399 |
| 3.11.6 ECST | 400 |
| 3.11.7 FAULT | 402 |
| 3.11.8 FAULTSIM | 405 |
| 3.11.9 FDEF | 409 |
| 3.11.10 FMASK | 414 |
| 3.11.11 HLLROUT | 417 |
| 3.11.12 HRLROUT | 420 |
| 3.11.13 MERR | 422 |
| 3.11.14 SAFIN | 422 |
| 3.11.15 SAFINI | 424 |
| 3.11.16 S_ERR | 425 |
| 3.11.17 S_FAULT | 425 |
| 3.11.18 S_FDEF | 429 |
| 3.11.19 S_FMASK | 432 |
| 3.11.20 S_SAFIN | 433 |
| 3.11.21 S_SAFINI | 434 |
| 3.11.22 SS11TIME | 435 |
| 3.11.23 SS12TIME | 435 |
| 3.11.24 STODELAY | 436 |
| 3.11.25 SYNC | 437 |
| 3.12 Induction Motor Variables | 437 |

| | | |
|----------|-------------------------------------|-----|
| 3.12.1 | SLCFIELD | 437 |
| 3.12.2 | SLCSLIP | 439 |
| 3.13 | Nanomotion Variables | 440 |
| 3.13.1 | SLDZMIN | 441 |
| 3.13.2 | SLDZMAX | 442 |
| 3.13.3 | SLDZTIME | 443 |
| 3.13.4 | SLZFF | 443 |
| 3.13.5 | SLFRC | 444 |
| 3.13.6 | SLFRCN | 445 |
| 3.13.7 | SLHRS | 446 |
| 3.13.8 | SLVKPDCF | 446 |
| 3.13.9 | SLPKPDCF | 447 |
| 3.13.10 | SLVKIDCF | 448 |
| 3.14 | Servo-Loop Variables | 448 |
| 3.14.1 | DCOM | 449 |
| 3.14.2 | Servo-Loop Current Variables | 450 |
| 3.14.2.1 | SLBIASA | 450 |
| 3.14.2.2 | SLBIASB | 451 |
| 3.14.2.3 | SLBIASC | 452 |
| 3.14.2.4 | SLIKI | 453 |
| 3.14.2.5 | SLIKP | 454 |
| 3.14.2.6 | SLIFILT | 454 |
| 3.14.2.7 | SLIOFFS | 455 |
| 3.14.2.8 | SLUI | 456 |
| 3.14.3 | Servo-Loop Velocity Variables | 456 |
| 3.14.3.1 | SLCRAT | 457 |
| 3.14.3.2 | SLVKI | 458 |
| 3.14.3.3 | SLVKIIF | 458 |
| 3.14.3.4 | SLVKISF | 459 |
| 3.14.3.5 | SLVKITF | 460 |
| 3.14.3.6 | SLVKP | 460 |
| 3.14.3.7 | SLVKPIF | 461 |
| 3.14.3.8 | SLVKPSF | 462 |
| 3.14.3.9 | SLVKPTF | 462 |

| | |
|------------------------------------------------------------|-----|
| 3.14.3.10 SLVLI | 463 |
| 3.14.3.11 SLVRAT | 464 |
| 3.14.4 Servo-Loop Velocity Notch Filter Variables | 465 |
| 3.14.4.1 SLVNFRQ | 465 |
| 3.14.4.2 SLVNWID | 466 |
| 3.14.4.3 SLVNATT | 466 |
| 3.14.5 Servo-Loop Velocity Low Pass Filter Variables | 467 |
| 3.14.5.1 SLVSOF | 467 |
| 3.14.5.2 SLVSOFD | 468 |
| 3.14.6 Servo-Loop Velocity Bi-Quad Filter Variables | 468 |
| 3.14.6.1 SLVBODD | 469 |
| 3.14.6.2 SLVBODF | 470 |
| 3.14.6.3 SLVBOND | 470 |
| 3.14.6.4 SLVBONF | 471 |
| 3.14.7 Servo-Loop Position Variables | 472 |
| 3.14.7.1 SLDRA | 472 |
| 3.14.7.2 SLDRAIF | 473 |
| 3.14.7.3 SLDRX | 474 |
| 3.14.7.4 SLPKI | 475 |
| 3.14.7.5 SLPKIIF | 476 |
| 3.14.7.6 SLPKISF | 476 |
| 3.14.7.7 SLPKITF | 477 |
| 3.14.7.8 SLPLI | 477 |
| 3.14.7.9 SLPKP | 478 |
| 3.14.7.10 SLPKPIF | 479 |
| 3.14.7.11 SLPKPSF | 480 |
| 3.14.7.12 SLPKPTF | 480 |
| 3.14.8 Servo-Loop Compensations Variables | 481 |
| 3.14.8.1 SLAFF | 481 |
| 3.14.8.2 SLFRCD | 482 |
| 3.14.9 Servo Loop Stepper Variables | 483 |
| 3.14.9.1 MFLAGSX | 483 |
| 3.14.9.2 SLSDZ | 485 |
| 3.14.9.3 SLSKI | 486 |

| | |
|--------------------------------------------------|-----|
| 3.14.9.4 SLSKP | 486 |
| 3.14.9.5 SLSMC | 487 |
| 3.14.9.6 SLSOUT | 488 |
| 3.14.9.7 SLSRL | 489 |
| 3.14.10 Servo-Loop Miscellaneous Variables | 489 |
| 3.14.10.1 SLCROUT | 490 |
| 3.14.10.2 SLGCAXN | 491 |
| 3.14.10.3 SLPROUT | 492 |
| 3.14.10.4 SLP2ROUT | 494 |
| 3.14.10.5 SLTFWID | 496 |
| 3.14.10.6 SLVROUT | 496 |
| 3.14.11 Non-Linear Control Variables | 498 |
| 3.14.11.1 SLPAP | 498 |
| 3.14.11.2 SLPDP | 499 |
| 3.14.11.3 SLPAI | 499 |
| 3.14.11.4 SLPDI | 500 |
| 3.14.11.5 SLVAP | 501 |
| 3.14.11.6 SLVDP | 501 |
| 3.14.11.7 SLVAI | 502 |
| 3.14.11.8 SLVDI | 503 |
| 3.15 Commutation Variables | 503 |
| 3.15.1 SLCHALL | 504 |
| 3.15.2 SLCNP | 504 |
| 3.15.3 SLCOFFS | 505 |
| 3.15.4 SLCORG | 506 |
| 3.15.5 SLCPRD | 507 |
| 3.15.6 SLHROUT | 507 |
| 3.15.7 SLSTHALL | 508 |
| 3.16 System Configuration Variables | 509 |
| 3.16.1 CFG | 510 |
| 3.16.2 CTIME | 510 |
| 3.16.3 EXTFAC | 511 |
| 3.16.4 FOLLOWCH | 511 |
| 3.16.5 G_01WCS...G_12WCS | 513 |

| | | |
|---------|-------------------------|-----|
| 3.16.6 | GPEXL | 513 |
| 3.16.7 | GSPEXL | 514 |
| 3.16.8 | GUFAC | 514 |
| 3.16.9 | IENA | 515 |
| 3.16.10 | IMASK | 516 |
| 3.16.11 | ISENA | 517 |
| 3.16.12 | S_FLAGS | 518 |
| 3.16.13 | S_SETUP | 520 |
| 3.16.14 | XSEGAMAX | 522 |
| 3.16.15 | XSEGAMIN | 522 |
| 3.16.16 | XSEGRMAX | 522 |
| 3.16.17 | XSEGRMIN | 523 |
| 3.17 | Communication Variables | 523 |
| 3.17.1 | BAUD | 524 |
| 3.17.2 | COMMCH | 525 |
| 3.17.3 | COMMFL | 526 |
| 3.17.4 | CONID | 527 |
| 3.17.5 | ECHO | 528 |
| 3.17.6 | DISPCH | 529 |
| 3.17.7 | GATEWAY | 531 |
| 3.17.8 | SUBNET | 532 |
| 3.17.9 | TCPIP | 533 |
| 3.17.10 | TCPIP2 | 534 |
| 3.17.11 | TCPPORT | 535 |
| 3.17.12 | UDPPORT | 536 |
| 3.18 | Miscellaneous | 537 |
| 3.18.1 | FK | 537 |
| 3.18.2 | STATIC | 537 |
| 3.18.3 | XARRSIZE | 538 |
| 4. | ACSPL+ Functions | 539 |
| 4.1 | Arithmetical Functions | 544 |
| 4.1.1 | ABS | 545 |
| 4.1.2 | ACOS | 545 |
| 4.1.3 | ASIN | 546 |

| | | |
|---------|-------------------------------------------|-----|
| 4.1.4 | ATAN | 546 |
| 4.1.5 | ATAN2 | 547 |
| 4.1.6 | CEIL | 547 |
| 4.1.7 | COS | 548 |
| 4.1.8 | EXP | 548 |
| 4.1.9 | FLOOR | 549 |
| 4.1.10 | HYPOT | 549 |
| 4.1.11 | LDEXP | 550 |
| 4.1.12 | LOG | 550 |
| 4.1.13 | LOG10 | 551 |
| 4.1.14 | POW | 551 |
| 4.1.15 | SIGN | 552 |
| 4.1.16 | SIN | 552 |
| 4.1.17 | SQRT | 553 |
| 4.1.18 | TAN | 553 |
| 4.1.19 | ROUND | 554 |
| 4.2 | Matrix Functions | 554 |
| 4.2.1 | Matrix Type | 555 |
| 4.2.1.1 | Matrix Initialization in Compilation Time | 555 |
| 4.2.2 | MATRIXADD | 556 |
| 4.2.3 | MATRIXSUB | 556 |
| 4.2.4 | MATRIXMUL | 557 |
| 4.2.5 | MATRIXMULSCA | 558 |
| 4.2.6 | MATRIXMULEW | 559 |
| 4.2.7 | MATRIXDIV | 559 |
| 4.2.8 | MATRIXIDENT | 560 |
| 4.2.9 | MATRIXTRANS | 561 |
| 4.2.10 | MATRIXINVERT | 561 |
| 4.3 | Miscellaneous Functions | 562 |
| 4.3.1 | GETCONF | 562 |
| 4.3.2 | SYSINFO | 569 |
| 4.3.3 | GETVAR | 570 |
| 4.3.4 | SETCONF | 570 |
| 4.3.5 | SETVAR | 583 |

| | |
|--------------------------------------|-----|
| 4.3.6 STR | 583 |
| 4.3.7 STRTONUM | 584 |
| 4.3.8 NUMTOSTR | 585 |
| 4.3.9 BCOPY | 586 |
| 4.3.10 SS1RESET | 588 |
| 4.3.11 MDURATION | 588 |
| 4.4 Array Processing Functions | 591 |
| 4.4.1 AVG | 592 |
| 4.4.2 COPY | 592 |
| 4.4.3 DSHIFT | 594 |
| 4.4.4 FILL | 595 |
| 4.4.5 MAX | 596 |
| 4.4.6 MAXI | 597 |
| 4.4.7 MIN | 597 |
| 4.4.8 MINI | 598 |
| 4.4.9 SIZEOF | 598 |
| 4.5 EtherCAT Functions | 599 |
| 4.5.1 COEGETSIZE | 601 |
| 4.5.2 ECCLOSEPORT | 601 |
| 4.5.3 ECCLRREG | 602 |
| 4.5.4 ECEXTIN | 603 |
| 4.5.5 ECEXTOUT | 604 |
| 4.5.6 ECGETGRPIND | 605 |
| 4.5.7 ECGETPID | 606 |
| 4.5.8 ECGETMAIN | 606 |
| 4.5.9 ECGETOFFSET | 606 |
| 4.5.10 ECGETOPTGRP | 608 |
| 4.5.11 ECGETRED | 608 |
| 4.5.12 ECGETREG | 608 |
| 4.5.13 ECGETSLAVES | 610 |
| 4.5.14 ECGETSTATE | 610 |
| 4.5.15 ECGETVID | 611 |
| 4.5.16 ECGRPINFO | 611 |
| 4.5.17 ECIN | 611 |

| | | |
|--------|---------------------------|-----|
| 4.5.18 | ECOUT | 613 |
| 4.5.19 | ECREPAIR | 615 |
| 4.5.20 | ECRESCAN | 616 |
| 4.5.21 | ECRESCUE | 616 |
| 4.5.22 | ECSAVECFG | 617 |
| 4.5.23 | ECSAVEDCNF | 617 |
| 4.5.24 | ECUNMAP | 618 |
| 4.5.25 | ECUNMAPIN | 618 |
| 4.5.26 | ECUNMAPOUT | 619 |
| 4.5.27 | FOEDOWNLOAD | 619 |
| 4.5.28 | FOEUPLOAD | 620 |
| 4.5.29 | PDOEXT | 621 |
| 4.6 | CoE Functions | 621 |
| 4.6.1 | COERead | 622 |
| 4.6.2 | COEWRIte | 623 |
| 4.7 | Modbus Functions | 625 |
| 4.7.1 | MBOPEN | 626 |
| 4.7.2 | MBGETHANDLE | 627 |
| 4.7.3 | MBCLOSE | 628 |
| 4.7.4 | MBREADHREG | 629 |
| 4.7.5 | MBREADIREG | 632 |
| 4.7.6 | MBWRITEHREG | 636 |
| 4.7.7 | MBREADCOIL | 640 |
| 4.7.8 | MBWRITECOIL | 643 |
| 4.7.9 | MBREADDIN | 646 |
| 4.7.10 | MBUNMAP | 649 |
| 4.7.11 | MBCLEAR | 650 |
| 4.7.12 | MBERR | 651 |
| 4.7.13 | #MBMAPREP | 652 |
| 4.8 | Servo Processor Functions | 653 |
| 4.8.1 | GETSP | 654 |
| 4.8.2 | GETSPA | 655 |
| 4.8.3 | GETSPV | 655 |
| 4.8.4 | SETSP | 655 |

| | |
|---------------------------------------------------|-----|
| 4.8.5 SETSPV | 656 |
| 4.9 Signal Processing Functions | 656 |
| 4.9.1 DEADZONE | 658 |
| 4.9.2 DSIGN | 660 |
| 4.9.3 DSTR | 661 |
| 4.9.4 EDGE | 662 |
| 4.9.5 INTGR | 663 |
| 4.9.6 LAG | 664 |
| 4.9.7 Interpolation Functions | 666 |
| 4.9.7.1 Linear interpolation | 666 |
| 4.9.7.2 Spline interpolation | 666 |
| 4.9.7.3 MAP | 672 |
| 4.9.7.4 MAPB | 674 |
| 4.9.7.5 MAPN | 676 |
| 4.9.7.6 MAPNB | 678 |
| 4.9.7.7 MAPNS | 680 |
| 4.9.7.8 MAPS | 682 |
| 4.9.7.9 MAP2 | 684 |
| 4.9.7.10 MAP2B | 686 |
| 4.9.7.11 MAP2N | 688 |
| 4.9.7.12 MAP2NB | 691 |
| 4.9.7.13 MAP2NS | 693 |
| 4.9.7.14 MAP2S | 696 |
| 4.9.7.15 MATCH | 699 |
| 4.9.7.16 RAND | 700 |
| 4.9.7.17 ROLL | 701 |
| 4.9.7.18 SAT | 702 |
| 4.10 Laser Control Functions | 703 |
| 4.10.1 LCMODULATION | 705 |
| 4.10.1.1 Duty cycle or frequency update | 708 |
| 4.10.1.2 Duty cycle or Frequency monitoring | 708 |
| 4.10.2 LCFixedDist | 708 |
| 4.10.3 LCFixedInt | 712 |
| 4.10.4 LCRandomDist | 714 |

| | | |
|-----------|--------------------------------|-----|
| 4.10.5 | LCTickle | 717 |
| 4.10.6 | LCZone | 718 |
| 4.10.6.1 | LCZoneSet | 720 |
| 4.10.6.2 | LCZoneGet | 720 |
| 4.10.7 | LCStop | 721 |
| 4.10.8 | LCSignalSet | 721 |
| 4.10.9 | LCSignalGet | 724 |
| 4.10.10 | LCS conditioning example | 724 |
| 4.10.11 | Physical outputs configuration | 725 |
| 4.10.11.1 | LCOutputSet | 725 |
| 4.10.11.2 | LCOutputGet | 727 |
| 4.10.12 | LCDelaySet | 728 |
| 4.10.13 | LCDelayGet | 728 |
| 4.10.14 | AxListAsMask | 728 |
| 4.11 | Dynamic Error Compensation | 729 |
| 4.11.1 | ERRORMAP1D | 730 |
| 4.11.2 | ERRORMAPN1D | 731 |
| 4.11.3 | ERRORMAPA1D | 732 |
| 4.11.4 | ERRORMAP2D | 733 |
| 4.11.5 | ERRORMAPN2D | 735 |
| 4.11.6 | ERRORMAPA2D | 737 |
| 4.11.7 | ERRORMAP3DA | 738 |
| 4.11.8 | ERRORMAP3D2 | 741 |
| 4.11.9 | ERRORMAP3D3 | 746 |
| 4.11.10 | ERRORMAPN3D2 | 750 |
| 4.11.11 | ERRORMAPN3D3 | 754 |
| 4.11.12 | ERRORMAP3D5 | 758 |
| 4.11.13 | ERRORMAPN3D5 | 761 |
| 4.11.14 | ERRORMAPN3DA | 764 |
| 4.11.15 | ERRORMAPOFF | 767 |
| 4.11.16 | ERRORMAPON | 767 |
| 4.11.17 | #ERRORMAPREP | 767 |
| 4.11.18 | ERRORUNMAP | 768 |
| 5. | ACSPL+ Standard Structures | 769 |

| | |
|------------------------------------|-----|
| 5.1 LCI Standard Structure | 769 |
| 5.1.1 LCI Functions | 769 |
| 5.1.1.1 PowerPWMOut | 769 |
| 5.1.1.2 PowerAnalogOut | 770 |
| 5.1.1.3 PowerDigitalOut | 771 |
| 5.1.1.4 FixedDistPulse | 772 |
| 5.1.1.5 DistanceArrPulse | 773 |
| 5.1.1.6 CoordinateArrPulse | 774 |
| 5.1.1.7 Tickle | 775 |
| 5.1.1.8 LaserEnable | 776 |
| 5.1.1.9 LaserDisable | 776 |
| 5.1.1.10 DistanceArrGate | 776 |
| 5.1.1.11 CoordinateArrGate | 777 |
| 5.1.1.12 AddZone | 778 |
| 5.1.1.13 SetZone | 778 |
| 5.1.1.14 SetCondition | 779 |
| 5.1.1.15 GetCondition | 781 |
| 5.1.1.16 SegmentGate | 782 |
| 5.1.1.17 SegmentPulse | 782 |
| 5.1.1.18 SetExtClockSync | 782 |
| 5.1.1.19 PowerPWMBurst | 783 |
| 5.1.1.20 SetSafetyMasks | 784 |
| 5.1.1.21 Stop | 784 |
| 5.1.1.22 SetMechPlatformAxes | 784 |
| 5.1.1.23 SetMotionAxes | 785 |
| 5.1.1.24 SetSystemDelay | 786 |
| 5.1.1.25 GetSystemDelay | 786 |
| 5.1.1.26 SetConfigOut | 786 |
| 5.1.1.27 AssignChannels | 788 |
| 5.1.1.28 SetCustomPosCalc | 788 |
| 5.1.1.29 SetCustomVelCalc | 789 |
| 5.1.1.30 SetCustomVelVar | 789 |
| 5.1.2 LCI Structure Fields | 790 |
| 5.1.2.1 MotionAxes | 791 |

| | | |
|----------|-----------------------------------------------------------------------------------------------|-----|
| 5.1.2.2 | PosResolution | 792 |
| 5.1.2.3 | InternalPosResolution | 792 |
| 5.1.2.4 | PWMDutyCycle | 792 |
| 5.1.2.5 | PWMFrequency | 792 |
| 5.1.2.6 | PWMPulseWidth | 792 |
| 5.1.2.7 | TickleFrequency | 792 |
| 5.1.2.8 | TicklePulseWidth | 793 |
| 5.1.2.9 | PWMActive | 793 |
| 5.1.2.10 | TickleActive | 793 |
| 5.1.2.11 | InRange | 793 |
| 5.1.2.12 | LaserEnabled | 793 |
| 5.1.2.13 | OperationMode | 793 |
| 5.1.2.14 | Positions | 793 |
| 5.1.2.15 | UserPos | 794 |
| 5.1.2.16 | MultiAxWinSize | 794 |
| 5.1.2.17 | ExtraPulsesQty | 794 |
| 5.1.2.18 | ExtraPulsesPeriod | 794 |
| 5.1.2.19 | PiercePulsesNum | 795 |
| 5.1.2.20 | PiercePulsesWidth | 795 |
| 5.1.2.21 | GateOnDelay | 795 |
| 5.1.2.22 | GateOffDelay | 795 |
| 5.1.2.23 | PulseDelay | 796 |
| 5.1.2.24 | PowerAOutVal | 796 |
| 5.1.2.25 | Faults | 797 |
| 5.1.2.26 | PWMBurstReady | 797 |
| 5.2 | Diagnostics and Preventive Maintenance (DPM) | 797 |
| 5.2.1 | DPM_Measurement | 798 |
| 5.2.1.1 | DPM_Measurement Fields | 798 |
| 5.2.1.2 | DPM_Measurement Functions | 801 |
| 5.2.2 | DPM_Motion_Status | 803 |
| 5.2.2.1 | DPM_Motion_Status Fields | 803 |
| 5.2.2.2 | DPM_Motion_Status Functions | 804 |
| 5.2.3 | DPM Example - Adding current measurement during acceleration phase to an existing application | 805 |

| | |
|--------------------------------------------------|------------|
| 5.3 Motion Duration | 806 |
| 5.3.1 MotionDuration Struct | 806 |
| 6. Terminal Commands | 811 |
| 6.1 Entering Terminal Commands | 811 |
| 6.2 Query Commands | 811 |
| 6.2.1 Default Query Formats | 813 |
| 6.2.2 Predefined Query Output Formats | 813 |
| 6.2.3 User-Defined Query Output Format | 815 |
| 6.3 Program Management Commands | 815 |
| 6.3.1 Program Management Command Arguments | 815 |
| 6.3.2 Program Buffer Commands | 817 |
| 6.3.2.1 Open/Close Buffer (#) | 817 |
| 6.3.2.2 D | 818 |
| 6.3.2.3 F/IF | 819 |
| 6.3.2.4 L | 820 |
| 6.3.3 RESET | 822 |
| 6.3.4 Listing Program Variables | 822 |
| 6.3.4.1 VGR | 822 |
| 6.3.4.2 VSD | 823 |
| 6.3.4.3 VS/VSG | 823 |
| 6.3.4.4 VSF/VSGF | 824 |
| 6.3.4.5 VG/VGF | 825 |
| 6.3.4.6 VL/VLF | 825 |
| 6.3.4.7 V/VF | 826 |
| 6.3.4.8 VSP | 827 |
| 6.3.4.9 VST/VSGT | 827 |
| 6.3.4.10 VSTF/VSGTF/VSDT | 828 |
| 6.3.4.11 VGV | 828 |
| 6.3.4.12 VGS/VGSF | 828 |
| 6.3.5 Program Handling Commands | 829 |
| 6.3.5.1 C | 830 |
| 6.3.5.2 X | 831 |
| 6.3.5.3 S/SR | 832 |
| 6.3.5.4 P | 832 |

| | |
|--------------------------------------|------------|
| 6.3.6 Debug Commands | 833 |
| 6.3.6.1 XS | 833 |
| 6.3.6.2 XD | 834 |
| 6.3.6.3 BS | 834 |
| 6.3.6.4 BR | 835 |
| 6.4 System Commands | 836 |
| 6.4.1 SI | 836 |
| 6.4.2 SIR | 840 |
| 6.4.3 MEMORY | 852 |
| 6.4.4 IR | 852 |
| 6.4.5 U | 854 |
| 6.4.6 TD | 855 |
| 6.4.7 SC | 855 |
| 6.4.8 ETHERCAT | 857 |
| 6.4.9 ECMAPREP | 866 |
| 6.4.10 CC | 867 |
| 6.4.11 PLC | 868 |
| 6.4.12 LOG | 869 |
| 6.4.13 LOG HOST_TICKS | 870 |
| 6.4.14 LOGP | 871 |
| 7. SPiiPlus Error Codes | 873 |
| 7.1 ACSPL+ Syntax Errors | 873 |
| 7.2 ACSPL+ Compilation Errors | 908 |
| 7.3 ACSPL+ Runtime Errors | 932 |
| 7.4 Errors | 964 |
| 7.5 Encoder Errors | 974 |
| 7.6 System Errors | 976 |
| 7.7 EtherCAT Errors | 979 |
| 7.8 EtherCAT Slave Errors | 981 |
| 7.9 MODBUS Errors | 987 |
| 8. G-Code Error Codes | 988 |
| 8.1 G-Code Syntax Errors | 988 |
| 8.2 G-Code Compilation Errors | 992 |
| 8.3 G-Code Runtime Errors | 992 |

| | |
|-----------------------------------------------|------|
| Appendix A. PEG And MARK Mapping Tables | 996 |
| A.1 ASSIGNPEG Mapping | 996 |
| A.2 ASSIGNPOUTS Mapping | 1009 |
| A.3 ASSIGNMARK Mapping | 1020 |

List Of Figures

| | |
|-----------------------------------------------------------------------------------------------------|-----|
| Figure 2-1. CONNECT Using MAP Function | 50 |
| Figure 2-2. DISABLE and Mechanical Brake Output Process- Positive BONTIME | 56 |
| Figure 2-3. DISABLE and Mechanical Brake Output Process - Negative BONTIME | 56 |
| Figure 2-4. ENABLE and Mechanical Brake Output Process - Positive BOFFTIME | 57 |
| Figure 2-5. ARC1 Coordinate Specification | 120 |
| Figure 2-6. ARC2 Center Point and Rotation Angle Specification | 125 |
| Figure 2-7. SLAVE /pt Illustration | 141 |
| Figure 2-8. Single-Axis Motion Using MPTP | 148 |
| Figure 2-9. Two-Axis Group Motion Using MPTP/v | 149 |
| Figure 2-10. Results of Example MSEG | 151 |
| Figure 2-11. PATH...ENDS Diagram | 153 |
| Figure 2-12. PROJECTION of the XA Plane | 157 |
| Figure 2-13. FPOS - PROJECTION Example | 157 |
| Figure 2-14. PROJECTION Example - Final Result | 158 |
| Figure 2-15. PVSPLINE Motion Diagram | 162 |
| Figure 2-16. Use of STOPPER | 165 |
| Figure 2-17. Corner Processing - Exact Path Option | 172 |
| Figure 2-18. Corner Processing - Permitted Deviation, Permitted Radius and Corner Smoothing Options | 173 |
| Figure 4-1. Illustration of COPY Function | 594 |
| Figure 4-2. Example Mapping | 632 |
| Figure 4-3. Example Mapping | 636 |
| Figure 4-4. Example Mapping | 640 |
| Figure 4-5. Example Mapping | 643 |
| Figure 4-6. Example Mapping | 646 |
| Figure 4-7. Symmetrical Dead Zone Example | 659 |
| Figure 4-8. Asymmetrical Dead Zone Example | 659 |
| Figure 4-9. DSIGN Function Example | 661 |
| Figure 4-10. EDGE Function Example | 663 |
| Figure 4-11. INTGR Function Example | 664 |
| Figure 4-12. LAG Function Example | 666 |
| Figure 4-13. Spline Definition Range | 667 |

| | |
|-------------------------------------------------------------|-----|
| Figure 4-14. Two-Dimensional Spline Definition Range | 668 |
| Figure 4-15. 5-Point Catmull-Rom Spline | 669 |
| Figure 4-16. B-Spline - Approximation of Points | 670 |
| Figure 4-17. Catmull-Ron Spline Beyond the Definition Range | 671 |
| Figure 4-18. B-Spline Map | 672 |
| Figure 4-19. MAP Example on the Scope | 674 |
| Figure 4-20. MAPB Example on the Scope | 676 |
| Figure 4-21. MAPN Example on the Scope | 678 |
| Figure 4-22. MAPNB Example on the Scope | 680 |
| Figure 4-23. MAPNS Example on the Scope | 682 |
| Figure 4-24. MAPS Example on the Scope | 684 |
| Figure 4-25. MAP2 Example on the Scope | 686 |
| Figure 4-26. MAP2B Example on the Scope | 688 |
| Figure 4-27. MAP2N Example on the Scope | 691 |
| Figure 4-28. MAP2NB Example on the Scope | 693 |
| Figure 4-29. MAP2NS Example on the Scope | 696 |
| Figure 4-30. MAP2S Example on the Scope | 699 |
| Figure 4-31. ROLL Example on the Scope | 702 |
| Figure 4-32. SAT Example on the Scope | 703 |
| Figure 4-33. Velocity | 707 |
| Figure 4-34. 2 Pulses with Pulse Width 120 msec. | 795 |
| Figure 4-35. Delays in Gating Mode | 795 |
| Figure 5-1. Communication Terminal Window | 811 |
| Figure 5-2. Interaction of Program Buffer States | 829 |

List of Tables

| | |
|---------------------------------------------------------|-----|
| Table 2-1. The ACSPL+ command set | 40 |
| Table 2-2. Homing Methods | 65 |
| Table 2-3. DISP Command Option Escape Sequences | 75 |
| Table 2-4. Type Characters | 76 |
| Table 2-5. Channel Designation for TRIGGER | 87 |
| Table 2-6. PEG Output Signal Configuration | 96 |
| Table 2-7. Commonly Monitored SPDC Variables | 108 |
| Table 2-8. Matrix Values | 157 |
| Table 2-9. IF Control Structures | 190 |
| Table 3-1. Alphabetical Listing of All ACSPL+ Variables | 210 |
| Table 3-2. AFLAG Bit Description | 224 |
| Table 3-3. MFLAGS Bit Designators | 230 |
| Table 3-4. E_FLAGS Bit Description | 257 |
| Table 3-5. Homing Methods | 276 |
| Table 3-6. AST Bit Descriptions | 290 |
| Table 3-7. MST Bit Descriptions. | 296 |
| Table 3-8. NST Bit Description | 299 |
| Table 3-9. PFLAGS Bit Description 1 | 392 |
| Table 3-10. PST Bit Description | 395 |
| Table 3-11. ECST Bits | 400 |
| Table 3-12. Axis Fault Bits | 402 |
| Table 3-13. FDEF Bit Description | 410 |
| Table 3-14. FMASK Bit Description | 414 |
| Table 3-15. SAFINI Valid Bits | 424 |
| Table 3-16. S_FAULT Fault Bits | 426 |
| Table 3-17. S_FDEF Bit Description | 430 |
| Table 3-18. S_FMASK Bit Description | 432 |
| Table 3-19. SLCROUT Values | 490 |
| Table 3-20. SLPROUT Values | 493 |
| Table 3-21. SLVROUT Values | 497 |
| Table 3-22. IENA Bit Description | 515 |

| | |
|----------------------------------------------|-----|
| Table 3-23. ISENA Bit Description | 518 |
| Table 3-24. S_FLAGS Bit Description | 519 |
| Table 3-25. S_SETUP Bit Designators | 520 |
| Table 3-26. COMMCH Values | 525 |
| Table 3-27. COMMFL Bit Descriptions | 526 |
| Table 3-28. ECHO Channel Numbers | 528 |
| Table 3-29. DISPCH Channel Numbers | 530 |
| Table 4-1. GETCONF Return Values | 563 |
| Table 4-2. SYSINFO Return Values | 569 |
| Table 4-3. 16-bit Binary value Template | 571 |
| Table 4-4. SETCONF Arguments | 572 |
| Table 4-5. Supported Error Counter Registers | 609 |
| Table 4-6. Modbus Error Codes | 651 |
| Table 4-7. MAP Array | 673 |
| Table 4-8. MAPB Array | 675 |
| Table 4-9. MAPN Array | 677 |
| Table 4-10. MAPNB Array | 679 |
| Table 4-11. MAPNS Array | 681 |
| Table 4-12. MAPS Array | 683 |
| Table 4-13. MAP2 | 686 |
| Table 4-14. MAP2B | 688 |
| Table 4-15. MAP2B | 690 |
| Table 4-16. MAP2NB | 693 |
| Table 4-17. MAP2NS | 696 |
| Table 4-18. MAP2S | 698 |
| Table 4-19. Condition Mask for Register 0 | 779 |
| Table 4-20. Condition Mask for Register 1 | 780 |
| Table 4-21. Condition Mask for Register 2 | 780 |
| Table 5-1. Line Designation | 816 |
| Table 6-1. ACSPL+ Syntax Errors | 873 |
| Table 6-2. ACSPL+ Compilation Errors | 909 |
| Table 6-3. ACSPL+ Runtime Errors | 932 |
| Table 6-4. ACSPL+ Motion Termination Errors | 964 |
| Table 6-5. Encoder Errors | 974 |

| | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| Table 6-6. ACSPL+ System Errors | 976 |
| Table 6-7. ACSPL+ EtherCAT Errors | 979 |
| Table 6-8. EtherCAT Slave Errors | 982 |
| Table 6-9. Modbus Errors | 987 |
| Table A-1. Mapping PEG Engines to Encoders (Servo Processor 0) for SPiiPlusNT/DC-LT/HP/LD | 996 |
| Table A-2. Mapping PEG Engines to Encoders (Servo Processor 1) for SPiiPlusNT/DC-LT/HP/LD | 997 |
| Table A-3. Mapping PEG Engines to Encoders (Servo Processor 0) for SPiiPlus CMnt/CMhv/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/UDMhv/UDMnt/UDMpa/UDMpm/UDMpc/UDMcb | 998 |
| Table A-4. Mapping PEG Engines to Encoders (Servo Processor 0) for UDMlc/UDilt/UDlhp/UDMmc/PDIcl | 999 |
| Table A-5. Mapping PEG Engines to Encoders (Servo Processor 0) for NPMpm/NPMpc- | 1000 |
| Table A-6. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) for SPiiPlusNT/DC-LT/HP/LD | 1003 |
| Table A-7. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1) for SPiiPlusNT/DC-LT/HP/LD | 1003 |
| Table A-8. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) for SPiiPlus CMnt/UDMpm/CMhv/UDMhv- | 1004 |
| Table A-9. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) for UDMnt/UDMpa/UDMcb | 1005 |
| Table A-10. Engine to Encoder Assignment for IDMxx, ECMxx, and UDMsm/sa/ma | 1006 |
| Table A-11. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for SPiiPlusNT/DC-LT/HP/LD | 1009 |
| Table A-12. SPiiPlusNT/DC-LT/HP/LD Mapping of Engine Outputs to Physical Outputs (Servo Processor 1) | 1010 |
| Table A-13. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for CMnt/UDMpm/UDMpc/CMhv/UDMhv | 1010 |
| Table A-14. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0, OUT 0-4) for CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa | 1011 |
| Table A-15. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0, OUT_5-9) for CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa | 1012 |
| Table A-16. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for UDMnt/UDMpa/UDMcb | 1013 |
| Table A-17. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for UDMlc/UDMmc/UDilt/UDlhp/PDIcl | 1014 |
| Table A-18. NPMpm/NPMpc Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) | 1015 |

| | |
|--------------------------------------------------------------------------------------------------------------------------------|------|
| Table A-19. IDMxx/ECMxx/UDMsm/UDMsa/UDMma Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) | 1017 |
| Table A-20. Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD | 1020 |
| Table A-21. Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD | 1022 |
| Table A-22. Mark-1 Inputs to Encoders Mapping for with SPiiPlus CMnt/UDMpm-x/UDMpc/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/CMhv/UDMhv | 1024 |
| Table A-23. Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt/UDMpm/UDMpc/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/CMhv/UDMhv | 1025 |
| Table A-24. IDMxx/ECMxx/UDMsm/UDMsa/UDMma Encoder Mapping | 1026 |

1. Introduction

This document details all of the elements making up the SPiiPlus ACSPL+ Programming Language as well as the command set that may be entered through the SPiiPlus MMI Application Studio Communication Terminal for use in a SPiiPlus system.

This document is intended for the use of software engineers.

2. ACSPL+ Commands

ACSPL+ comes with a complete programming command set. The commands are divided into following categories:

- > [Axis Management Commands](#)
- > [Interactive Commands](#)
- > [PEG and MARK Commands](#)
- > [Miscellaneous Commands](#)
- > [Motion Commands](#)
- > [Program Flow Commands](#)
- > [Program Management Commands](#)
- > [Laser Control Commands](#)

Table 2-1. The ACSPL+ command set

| Command | Description |
|-----------------------------|--------------------------------------------------------------------------------------------------------|
| ARC1 | Adds an arc segment to MSEG...ENDS motion. |
| ARC1 | Adds an arc segment to XSEG...ENDS motion |
| ARC2 | Adds an arc segment to MSEG...ENDS motion. |
| ARC2 | Adds an arc segment to XSEG...ENDS motion |
| ASSIGNMARK | Marks inputs-to-encoder assignment |
| ASSIGNPEG | Assigns an encoder to a PEG engine and GP physical output connection. |
| ASSIGNPOUTS | Assignment of physical output pins. |
| AXISDEF | Assigns an alias to an axis |
| BLOCK...END | Performs a group of ACSPL+ commands in one controller cycle. |
| BREAK | Immediately terminates a motion and provides smooth transition to the next motion in the motion queue. |
| CALL | Calls subroutine. |
| COMMUT | Performs auto commutation for DC brushless (AC servo) motors. |
| CONNECT | Defines a formula for calculating reference position (RPOS). |

| Command | Description |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "CSCREATE" on page 50 | Creates a new Local Coordinate System. |
| "CSDESTROY" on page 53 | Cancels the active Local Coordinate System. |
| DC | Activates data collection. |
| DEPENDS | Specifies a logical dependence between a motor and axes. |
| DISABLE/DISABLEALL | Shuts off one or more drives. DISABLE ALL provides DISABLE operation for all axes. |
| DISABLEON | Disables autoroutine activation in a buffer. |
| DISP | Builds a string and sends it to the default communication channel. |
| ECRESCAN | Returns the system back to the operational state if one or more slaves underwent a reset or power cycle. |
| ENABLE/ENABLE ALL | Activates one or more drives |
| ENABLEON | Enables autoroutine activation in a buffer. |
| FCLEAR | Clears faults. |
| GO | Starts a motion that was created using the /w command option. |
| GOTO | Transfers program execution to another point in the program. |
| GROUP | Defines an axis-group for coordinate multi-axis motion. |
| HALT | Terminates one or more motions using a third-order deceleration profile (DEC deceleration). HALTALL provides HALT operation for all axes. |
| IF, ELSEIF, ELSE...END | IF command structure. |
| IMM | Provides on-the-fly change of motion parameters. |
| INPUT | Suspends program execution pending user input |
| INTERRUPT | Causes an interrupt that can be intercepted by the host. |
| INTERRUPTEX | Causes an interrupt similar to that of the INTERRUPT command. |
| JOG | Creates a jog motion. |

| Command | Description |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| KILL/KILLALL | Terminates one or more motions using a second-order deceleration profile and the KDEC deceleration value. KILL ALL provides KILL operation for all axes. |
| LCDISABLE | Stops a pulse generation process, including tickle pulses. |
| LCENABLE | Enables a pulse generation process with current set parameters. |
| LINE | Adds a linear segment to MSEG...ENDS motion. |
| LINE | Adds a linear segment to XSEG...ENDS motion. |
| LOOP...END | Loop command structure. |
| MASTER | Defines a formula for calculating MPOS . |
| MPOINT | Adds a set of points to MPTP...ENDS , PATH...ENDS or PVSPLINE...ENDS motion. |
| MPTP...ENDS | Creates a multipoint motion. |
| MSEG...ENDS | Creates a segmented motion. |
| ON...RET | The autoroutine structure. An the autoroutine is automatically executed when a specific condition is satisfied. The routine interrupts the currently executing program, executes the commands specified in the autoroutine body, and then returns control to the interrupted program. |
| PATH...ENDS | Creates an arbitrary path motion with linear interpolation between the specified points. |
| PAUSE | Suspends program execution in a buffer. |
| | Defines Incremental PEG parameters. activates the PEG engine. |
| | Defines the Random PEG parameters. activates the PEG engine. |
| POINT | Adds a point to MPTP...ENDS , PATH...ENDS , or PVSPLINE...ENDS motion. |
| PROJECTION | An expansion command to the MSEG...ENDS set of commands, that allows the controller to perform a three dimensional segmented motion such as creating arcs and lines on a user-defined plane. |

| Command | Description |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PTP | Creates a point-to-point motion. |
| PVSPLINE...ENDS | Creates an arbitrary path motion with spline interpolation between the specified points. |
| READ | Reads an array from a file in the flash memory. |
| RESUME | Resumes program execution in a buffer. |
| SAFETYCONF | Configures fault processing for one or more axes. |
| SAFETYGROUP | Activates the fault response for all axes in the <i>axis_list</i> when any axis triggers the fault, and manages the axes as a block in response to KILL/KILLALL and DISABLE/DISABLEALL . |
| SEND | Same as DISP , but also specifies the communication channel or channels. |
| SET | Defines the current value of either feedback (FPOS), reference (RPOS), or axis (APOS) position. |
| SLAVE | Creates a master-slave motion. |
| SMOVE | Define segment of movement with transition point smoothing |
| SPDC | Activates data collection from a Servo Processor variable. |
| SPRT | Activates a real-time data transfer process from the MPU to a given Servo Processor. |
| SPRTSTOP | Stops an active real-time data transfer process on the given SP (for cyclic command only). |
| SPINJECT | Initiates the transfer of MPU real-time data to the Servo Processor. |
| SPLIT | SPLIT breaks apart an axis group. SPLITALL breaks apart all axis groups. See GROUP . |
| START | Activates program execution in a buffer. |
| | Restarts PEG at the current position if has been issued and the <i>last_point</i> has not been reached. |
| STOP/STOPALL | Terminates program execution in a buffer. |

| Command | Description |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| STOPDC | Terminates data collection. |
| STOPINJECT | Stops the transfer of MPU real-time data to the Servo Processor. |
| Axis Management Commands | Halts the PEG engine for the specified axis. |
| STOPPER | Adds a segment separator to MSEG...ENDS motion. |
| TILL | Delays program execution until a specified expression produces a non-zero (true) result. |
| TRACK | Creates tracking motion. |
| TRIGGER | Specifies a triggering condition. Once the condition is satisfied, the controller issues an interrupt to the host computer. |
| WAIT | Delays program execution for a specified number of milliseconds. |
| WHILE...END | While command structure. |
| WRITE | Writes an array to a file in the flash memory. |
| XSEG...ENDS | Creates extended segment motion. |

2.1 Axis Management Commands

The Axis Management commands are:

| Command | Description |
|------------------------|--------------------------------------------------------------------------------------------------------|
| BREAK | Immediately terminates a motion and provides smooth transition to the next motion in the motion queue. |
| COMMUT | Performs auto commutation for DC brushless (AC servo) motors. |
| CONNECT | Defines a formula for calculating reference position (RPOS). |
| "CSCREATE" on page 50 | Creates a new Local Coordinate System. |
| "CSDESTROY" on page 53 | Cancel the active Local Coordinate System. |
| DEPENDS | Specifies a logical dependence between a motor and axes. |
| DISABLE/DISABLEALL | Shuts off one or more drives. DISABLE ALL provides DISABLE operation for all axes. |

| Command | Description |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ENABLE/ENABLE ALL | Activates one or more drives |
| ENCINIT | Used for encoder configuration |
| FCLEAR | Clears faults. |
| FOLLOW | Switches an axis into slave mode. |
| GO | Starts a motion that was created using the /w command option. |
| GROUP | Defines an axis-group for coordinate multi-axis motion. |
| HALT | Terminates one or more motions using a third-order deceleration profile (DEC deceleration). HALTALL provides HALT operation for all axes. |
| HOME | Recives parameters for a predefined set of homing methods. |
| IMM | Provides on the fly change of motion parameters. |
| KILL/KILLALL | Terminates one or more motions using a second-order deceleration profile and the KDEC deceleration value. KILLALL provides KILL operation for all axes. |
| SAFETYCONF | Configures fault processing for one or more axes. |
| SAFETYGROUP | Creates a safety axis group. When any axis in the group triggers a fault, the fault affects all axes in the group. |
| SET | Defines the current value of either feedback (FPOS), reference (RPOS), or axis (APOS) position. |
| SPLIT | SPLIT breaks apart an axis group - see GROUP . SPLITALL breaks apart all axis groups. |
| UNFOLLOW | Switches an axis into regular mode. |

2.1.1 BREAK

Description

BREAK immediately terminates the currently executed motion of the specified axis without building a deceleration profile, and initiates the next motion in the axis motion queue, if it exists.

Syntax

BREAK *axis_list*

Arguments

axis_list

Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

Comments

BREAK executes differently in the following cases:

1. When the next motion waits in the motion queue, **BREAK** terminates the current motion and starts the next motion immediately.
2. When there is no next motion in the motion queue **BREAK** has no immediate effect. The current motion continues until the next motion appears in the motion queue. At that moment the controller breaks the current motion and provides a smooth velocity transition profile from motion to motion. If the current motion finishes before the next motion comes to the queue, the command has no effect.

COM Library Methods and .NET Library Methods

Break

C Library Functions

acsc_Break

Example

```

PTP 0, 1E8           !Move to point 1E8
PTP 0, -1E8          !Add another motion to the motion queue
WAIT 10000           !Wait 10 seconds
BREAK 0              !Terminate the first motion (PTP 0, 1E8)
                     !and immediately start the next motion:
                     !(PTP 0, -1E8)
STOP                 !End program

```

2.1.2 COMMUT

Description

COMMUT performs auto commutation and may be used when the following conditions hold true:

- > The motor is DC brushless (AC servo)
- > The motor is enabled
- > The motor is idle
- > The axis is already configured and properly tuned



Versions 2.60 and higher supports **COMMUT** in GANTRY mode. Commutation of the primary axis will automatically trigger commutation of the secondary axis.

Syntax

COMMUT *axis* [*excitation_current*][*settle_time*][*slope*][*gantry_commut_delay*]

Arguments

| | |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis</i> | The affected axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>excitation_current</i> | Optional - Given as a percentage of the full current command. The default value is set to 98% of XRMS . |
| <i>settle_time</i> | Optional - Determines settling time for the auto commutation process initiated by COMMUT . The default value is 500 msec. The entire auto commutation process lasts approximately three times longer, since the command executes the algorithm three times for verification. |
| <i>slope</i> | Optional - The slope parameter is optional, and used only in special cases. If a value is assigned to this parameter then the excitation current command builds up with some slope. The parameter sets the duration of the current build-up process in milliseconds. It is usually recommended to omit this parameter, in which case the excitation current is built instantly. |
| <i>gantry_commut_delay</i> | Optional – can be used only in Gantry mode. It defines the delay time in milliseconds after the commutation of the primary axis is completed and before the commutation of the complimentary axis begins. The default value is 500 msec. |

Comments

COMMUT is generally used in auto commutation-based startup programs.

The excitation current, settle time and slope are optional parameters for the auto commutation process initiated by **COMMUT**.

Refer to the relevant section in the Setup Guide for a complete description of the commutation process.

COM Library Methods and .NET Library Methods

Commut, WaitMotorCommutated

C Library Functions

acsc_Commut, acsc_WaitMotorCommutated

```
COMMUT 0,80,100,30      !Commut 0 axis with an excitation current
                        !of 80%. Settling time is 100msec, and a
                        !current build-up slope of 30msec.
```

2.1.3 CONNECT**Description**

CONNECT defines a formula for calculating reference position (**RPOS**). This formula can include any other axes variables. **DEPENDS** must follow **CONNECT**.

Syntax

CONNECT *axis_RPOS = formula*

Comments

Care needs to be taken when using complex non-default connections. Especially with articulated robots, the non-default connections can involve inverse trigonometric functions, square roots, division, and other mathematical operations that can cause numerical errors when not properly posed. While it is recommended that **CONNECT** command be written to avoid this from occurring, it is not always possible; therefore proper handling of the numerical errors is necessary.

The following are general guidelines concerning the **CONNECT** command:

1. The default relation between an axis position (**APOS**) and its reference (motor) position (**RPOS**) is 1:1.
2. Defining a different relation can be very useful for mechanical error corrections, dynamic error compensation, backlash compensation, inverse kinematics and more.
3. If the **CONNECT** relation is based on another axis position, it creates a strict link (like a mechanical connection) between all defined axes for as long as the function is active.
4. The variable **MFLAGS**<*axis*>.17 (bit 17) disables or enables a customized (non-default) **CONNECT** formula definition. See **MFLAGS**.
5. After **CONNECT** it is recommended to initialize **ROFFS** with the first value in the correction table (as seen in the following example:

```
SET RPOS0=MAP (APOS0 , ARRAY , 100 , 200)
```

This forces **ROFFS** to be zero and prevents the creation of a constant offset to **RPOS**.

6. **ENABLE/ENABLE ALL**, **DISABLE/DISABLEALL** and **KILL/KILLALL** change the value of **ROFFS**. Therefore, if these commands follow **CONNECT**, then redefine the **CONNECT** formula, and **RPOS** should be initialized to nearest value.
7. To stop motion after using **CONNECT**, use **HALT** instead of **KILL**. **HALT** does not affect the **ROFFS** variable.

If a numerical error occurs when evaluating a non-default connection, the output sent to **RPOS** is undefined. As such it is recommended to toggle back to the default connection and then go back to non-default connection.

When going back to the default connection the simplest way is to set **MFLAGS**().17. When this happens **RPOS** does not change, but **APOS** will change and be set to **RPOS**. However, this sudden change of **APOS** may also cause a numerical error if **APOS** is used in a **CONNECT** function. If this happens **MFLAGS**().17 will be set, but the non-default connection will still be active.

A more robust way of handling this change is to first explicitly change the connect function of all applicable axes to **RPOS = APOS**. When this happens neither **RPOS** nor **APOS** will change instantaneously, so no numerical error should occur. Then **MFLAGS**().17 can be set without causing a numerical error.

Related ACSPL+ Commands

DEPENDS

Related ACSPL+ Variables

RPOS, APOS, ROFFS

```

GLOBAL REAL ARRAY(6)           !Define ARRAY.
ARRAY(0)=60;ARRAY(1)=40;ARRAY(2)=90;ARRAY(3)=-40;ARRAY(4)=60; ARRAY(5)=10
                                !Populate Correction point ARRAY for MAP
                                !function.
MFLAGS(0).17=1                 !Set default connection between APOS and
                                !RPOS (RPOS=APOS).
ENABLE 0                        !Enable 0 axis.
MFLAGS(0).17=0                 !Set non-default connection between APOS
                                !and RPOS (RPOS is a function of APOS).
CONNECT RPOS0=APOS0+MAP(APOS0,ARRAY,100,200)
                                !CONNECT formula between RPOS0 and
                                !APOS0 using the MAP function with
                                !correction table ARRAY.
DEPENDS 0,0                     !Assign Axis 0 to Motor 0. See DEPENDS.
SET APOS0=0;SET RPOS0=0        !Initialize APOS and RPOS at 0.
PTP 0, MAP(APOS0,ARRAY,100,200)
                                !Moves axis 0 to the first point in the
                                !correction ARRAY to avoid a constant
                                !offset in ROFFS, as explained in Comment 2.
PTP 0, 1300                    !Move to 1300. Each point during the motion
                                !is modified according to the correction
                                !ARRAY in the MAP function.
STOP                            !End program
    
```

This illustrates the results of the example on the SPiiPlus MMI Application Studio Scope.

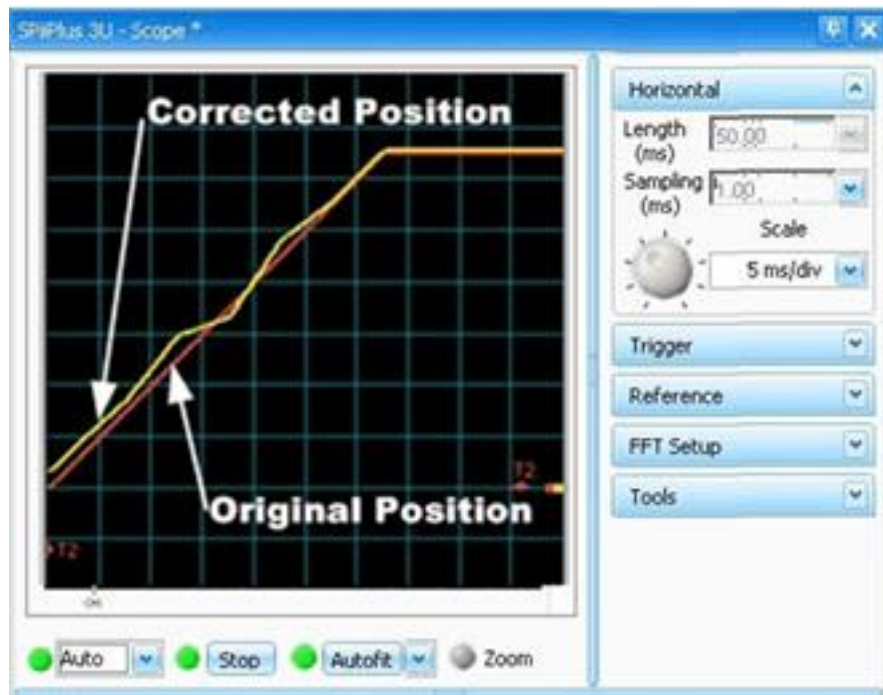


Figure 2-1. CONNECT Using MAP Function

2.1.4 CSCREATE

Description

The **CSCREATE** command creates the new Local Coordinate System (LCS) relative to the Machine Coordinate System or the previous LCS, depending on the applied switches.

Syntax

```
CSCREATE[/r] axis_list, x_trans, y_trans[, rot_axis, rot_angle]
```

or

```
CSCREATE[/r] axis_list, x_trans, y_trans, z_trans[, rot_axis, rot_angle]
```

Arguments

| | |
|------------------|------------------------------------------------------------------------------------------------------------|
| axis_list | The group of 2 or 3 axes. Valid values are: 0, 1, 2 ... up to the number of axes in the system minus 1. |
| x_trans | The new X position of the LCS in user units |
| y_trans | The new Y position of the LCS in user units |
| z_trans | The Z position of the LCS in user units. This parameter is included when axis_list includes 3 axes. |
| rot_axis | (Optional) The rotation axis: 0 – X, 1 – Y, 2 – Z |
| rot_angle | (optional) Rotation angle value: (-3.14159 : +3.14159) in radians |

Switches

| | |
|-----------|------------------------------------------------------------------------------------------------------------------------------|
| /r | The new LCS is relative (additive) to the existing LCS (otherwise the new LCS is relative to the Machine Coordinate System). |
|-----------|------------------------------------------------------------------------------------------------------------------------------|

Comments

- > In function calls which include rotation parameters, the translation parameters are applied to the system first, and then the rotation parameters.
- > The enumeration of axes for the **rot_axis** parameter is a numbered list of axes in the newly created coordinate system. This enumeration relates to the virtual axes, not the physical axes of the system.
- > This command is supported in version 3.10 and higher.

Example 1

This example demonstrates rectangular motion in PTP mode.

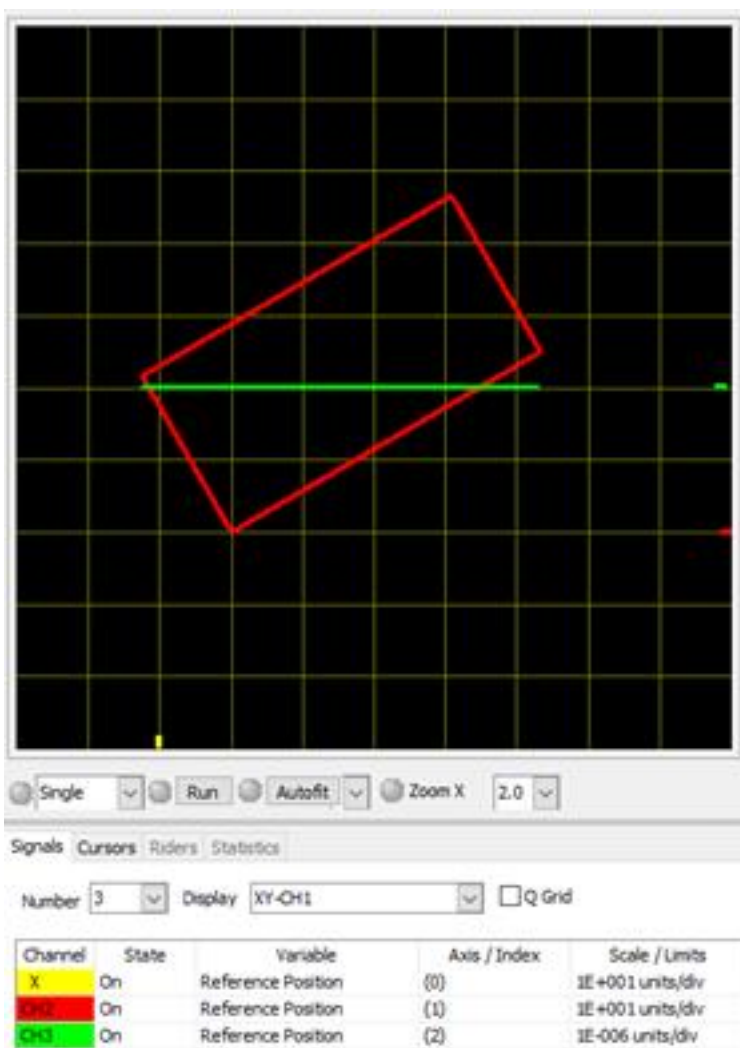
```

!Create LCS with origin (10, 0, 0) and rotated 30 degrees around Z
REAL ang = 30* 3.141592 /180
CSCREATE (X,Y,Z), 10, 0, 0, 2, ang

PTP/ze (X,Y), 0, 0    !go to the beginning position

!Make rectangular motion
PTP/ze (X,Y), 0, 25
PTP/ze (X,Y), 50, 25
PTP/ze (X,Y), 50, 0
PTP/ze (X,Y), 0, 0

CSDESTROY (X,Y,Z)    !restore machine coordinate system
    
```



Example 2

This example demonstrates rectangular motion in MPTP mode.

```

!Create LCS with origin (10, 0, 0) and rotated 30 degrees around Z
REAL ang = 30* 3.141592 /180
CSCREATE (X,Y,Z), 10, 0, 0, 2, ang
PTP/ze (X,Y), 0, 0      !go to the beginning position
MPTP/z (X,Y)
    POINT (X,Y), 0, 25
    POINT (X,Y), 50, 25
    POINT (X,Y), 50, 0
    POINT (X,Y), 0, 0
ENDS (X,Y)
TILL GSEG (X) = -1      !Wait until motion complete
CSDESTROY (X,Y,Z)      !restore machine coordinate system

```

Example 3

This example demonstrates round rectangular motion in XSEG mode.

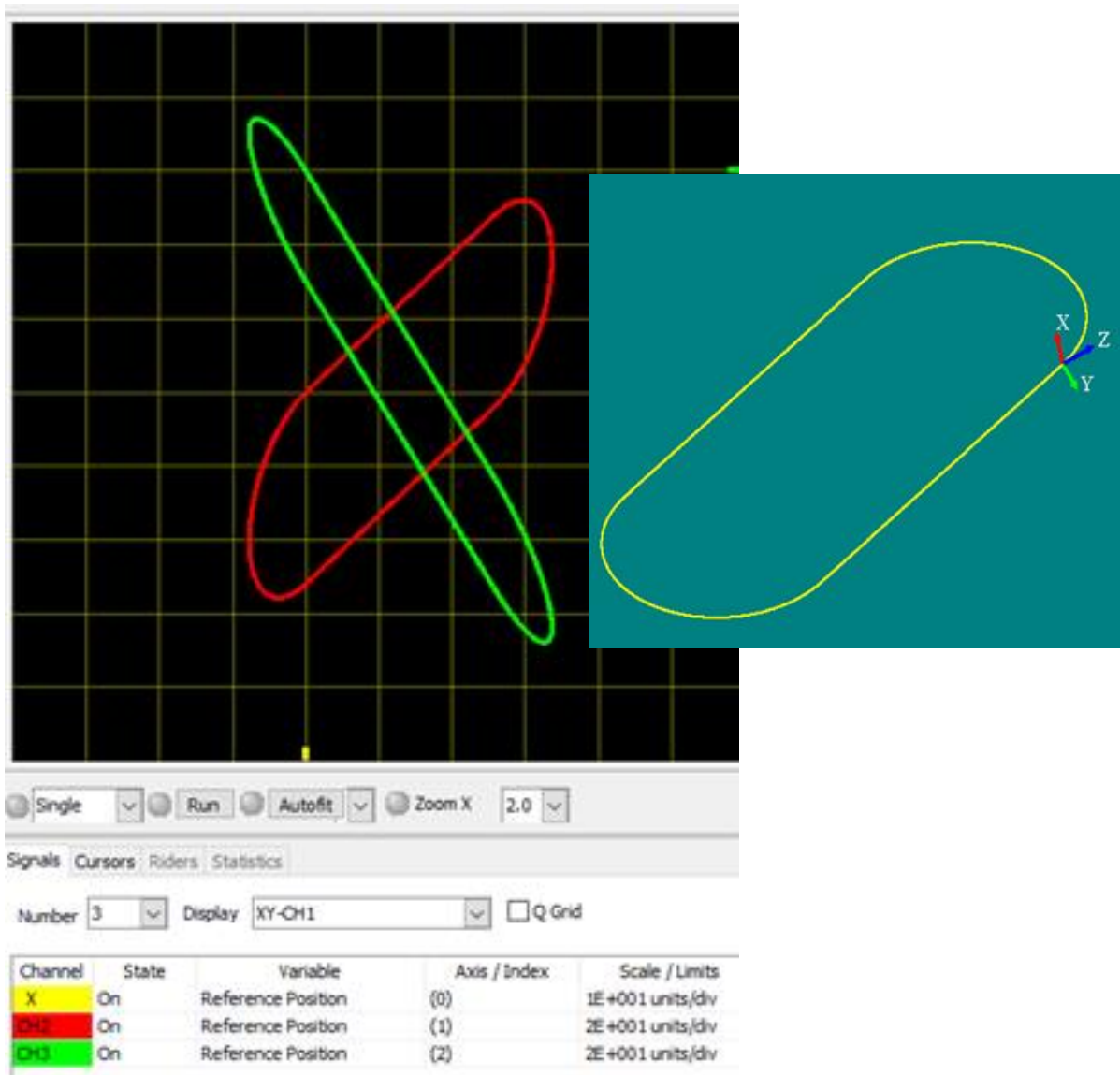
```

!Create LCS rotated 30 degrees around X
REAL ang = 30* 3.141592 /180
CSCREATE (X,Y,Z), 10, 0, 0, 0, ang
ang = 75* 3.141592 /180
CSCREATE/r (X,Y,Z), 0, 0, 0, 1, ang !Additional rotation 75 deg. Around Y

PTP/ze (X,Y), 0, 0      !go to the beginning position

XSEG/z (X,Y), 0, 0
    LINE (X,Y), 100, 0
    ARC2 (X,Y), 100, -30, -3.14159
    LINE (X,Y), 0, -60
    ARC2 (X,Y), 0, -30, -3.14159
ENDS (X,Y)
TILL GSEG (X) = -1      !Wait until motion complete
CSDESTROY (X,Y,Z)      !restore machine coordinate system

```



2.1.5 *CSDESTROY*

Description

The **CSDESTROY** command cancels the active Local Coordinate System and sets the Machine Coordinate System or previous Local Coordinate System.

Syntax

```
CSDESTROY axis_list[, restore_flag]
```

Arguments

| | |
|---------------------|----------------------------------------------------------------------------------------------------|
| axis_list | The group of 3 axes. Valid values are: 0, 1, 2 ... up to the number of axes in the system minus 1. |
| restore_flag | (Optional) Set to 1 to restore the previous LCS; 0 or omitted value restores the MCS. |



Unpredictable and dangerous motion may result if **CSDESTROY** is called before the motion involving the coordinate system created by **CSCREATE** is complete.

Comments

This command is supported in version 3.10 and higher.

Examples

See [CSCREATE](#).

2.1.6 DEPENDS

Description

DEPENDS is used only following [CONNECT](#).

DEPENDS specifies a logical dependence between a physical axis (motor) and the same or other logical axes. By default, the physical axis (motor) is assigned to its axis. **DEPENDS** is necessary because the controller is generally not capable of deriving dependence information from the **CONNECT** formula.

Syntax

DEPENDS *physical_axis*, **axis_list**

Arguments

| | |
|----------------------|-------------------------------------------------------------------------------------------------------|
| physical_axis | Physical axis (motor). |
| axis_list | Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |

Comments

- > Once a **CONNECT** command is executed, the controller resets the motor dependence information to the default-the motor depends only on the corresponding axis.
- > If a connection formula actually causes the motor to be dependent on another axis / axes, the **DEPENDS** command must follow to specify actual dependence.

Related ACSPL+ Commands

[CONNECT](#)

Example

See the examples from [CONNECT](#).

2.1.7 DISABLE/DISABLEALL

Description

DISABLE deactivates one, several, or using **DISABLEALL**, all drives. After **DISABLE**, **RPOS = FPOS** which means that no position error exists, or **PE<axis> = 0**.

Syntax

DISABLE *axis_list* [*reason*]

Arguments

| | |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis, or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>reason</i> | <i>reason</i> is an optional parameter. <i>reason</i> must be an integer constant or expression that equates to an integer and specifies a reason why the motor was disabled. If the parameter is specified, its value is stored in the MERR variable. If the parameter is omitted, MERR stores zero after the disable operation. |

Related ACSPL+ Commands

[ENABLE/ENABLE ALL](#), [FCLEAR](#), [DISABLEALL](#)

Related ACSPL+ Variables

[MERR](#), [FPOS](#), [RPOS](#)

COM Library Methods and .NET Library Methods

Disable, DisableM, DisableAll

C Library Functions

acsc_Disable, acsc_DisableM, acsc_DisableAll

The examples illustrate the **DISABLE** process using positive and negative **BONTIME** values.

The examples are followed by screens illustrating the results of **DISABLE** for both positive and negative **BONTIME**.

Example 1:

```
DISABLE (0,1)                !Disables the 0 and 1 motors. A fault
                             !notification window will be displayed.
```

Example 2:

```
DISABLE (0,1), 6100         !Disable 0 and 1 motors, store 6100 as the
                             !reason. 6100 specifies a user-defined error
                             !code which is stored in MERR.
```

Example 3:

```
DISABLE ALL                 !Disable all motors.
```

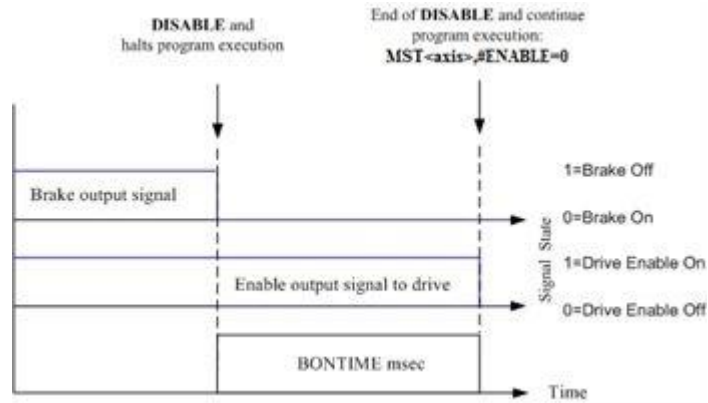


Figure 2-2. DISABLE and Mechanical Brake Output Process- Positive BONTIME

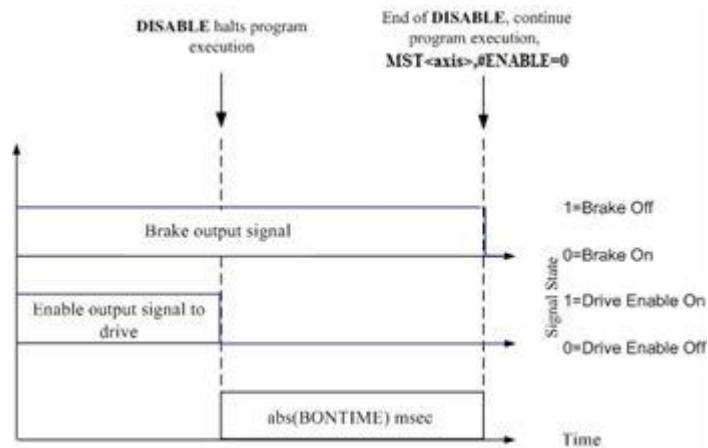


Figure 2-3. DISABLE and Mechanical Brake Output Process - Negative BONTIME

2.1.8 ENABLE/ENABLE ALL

Description

ENABLE activates one or more axes. After **ENABLE**, the motor starts following the reference position (**RPOS**) and axis faults are enabled. **ENABLE ALL** activates all axes.

Syntax

ENABLE|ENABLE ALL *axis_list*

Arguments

| | |
|------------------|-------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------------|-------------------------------------------------------------------------------------------------------|

Comments

Motor specification is a single axis like 0 or 13, a string of axes enclosed in parentheses and separated by commas, for example: (0,1,13), or the keyword: **ALL** for all axes.

Related ACSPL+ Commands

[DISABLE/DISABLEALL](#)

Related ACSPL+ Variables

ENTIME, MFLAGS<axis>.#ENMOD

COM Library Methods and .NET Library Methods

Enable, EnableM, Wait Motor Enabled

C Library Functions

acsc_Enable, acsc_EnableM, acsc_WaitMotorEnabled

Example1:

```
ENABLE 0 !Enable 0 axis.
```

Example 2:

```
ENABLE (0,1) !Enable 0 and 1 axes.
```

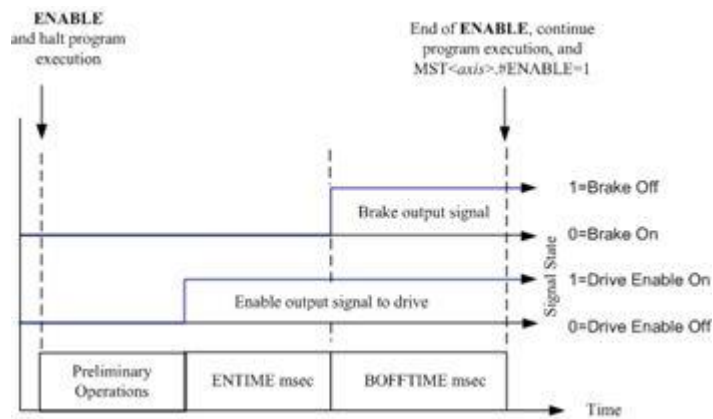


Figure 2-4. ENABLE and Mechanical Brake Output Process - Positive BOFFTIME

2.1.9 ENCINIT

Description

The **ENCINIT** function is used for encoder configuration. The **ENCINIT** function, if executed in a buffer, will wait until initialization is finished.

Syntax

```
ENCINIT (Axis, E_type, [Primary[, Slabits[, E_par_a[, E_par_b[, E_par_c [, e_freq[, e_scmul[, E_aoffs[, ESTBITS[, EMTBITS]]]]]]]]))
```

Arguments

| | |
|--------|----------------------------------------------------------------------------------------------|
| Axis | The affected axis, valid number are: 0,1,2.. up to the number of axis in the system minus 1. |
| E_type | Encoder Type, according to ACSPL+ E_TYPE variable definition. |

| | |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Primary | Optional. Defines the feedback – can be primary or secondary. Possible values: Primary: 1 – default Secondary: 0 |
| Slabits | (Optional, Integer) Used for setting the total number of absolute position bits for an absolute encoder, according to ACSPL+ SLABITS variable definition. |
| E_par_a | (Optional, Double) Used for setting the encoder data transmission actual frequency in MHZ. According to ACSPL+ E_PAR_A definition. |
| E_par_b | (Optional, Integer) Used for setting the encoder data control CRC code. According to ACSPL+ E_PAR_B definition. |
| E_par_c | (Optional, Integer) Used for setting the interval (in microseconds) of encoder position reading. According to ACSPL+ E_PAR_C definition. |
| E_freq | (Optional, Integer) Used for defining the maximum encoder pulse frequency (in MHZ). According to ACSPL+ E_FREQ definition. |
| E_scmul | (Optional, Integer) Used for specifying the Sin-Cos multiplication factor for the encoder. According to ACSPL+ E_SCMUL definition. |
| E_aoffs | (Optional, double) Used for setting user-defined offset for absolute encoder. According to ACSPL+ E_AOFFS definition. |
| ESTBITS | (Optional, integer) Used for setting the single turn resolution (number of bits). |
| EMTBITS | (Optional, integer) Used for setting the multi turn resolution (number of bits). |

Return Value

None

Comments

If an optional parameter is specified, the relevant ACSPL+ variable is modified as well. Otherwise, the initialization of the encoder will use the existing value of the variable.

If the Primary parameter is set to 0, secondary feedback variables are affected or used during initialization, according to the following table.

If one of the parameters is out of range, error 3041 "Assigned value is out of range" given. Not allowed E_TYPE value will trigger error 3194 "Not allowed Encoder Type".

If ESTBITS or EMTBITS are not 0 and SLABITS is not equal to ESTBITS+EMTBITS, an error is triggered.

This function is supported in version 3.00 and higher.

| Primary Feedback ACSPL+ Variable | Secondary Feedback ACSPL+ Variable |
|----------------------------------|------------------------------------|
| SLABITS | S2LABITS |
| E_PAR_A | E2_PAR_A |
| E_PAR_B | E2_PAR_B |
| E_PAR_C | E2_PAR_C |
| E_AOFFS | E2_AOFFS |
| E_FREQ | E2_FREQ |
| E_SCMUL | E2_SCMUL |
| ESTBITS | E2STBITS |
| EMTBITS | E2MTBITS |

Example

```
ENCINIT (0,10)! triggers initialization of Endat 2.2 for axis 0, primary
!feedback
```

```
STOP
```

2.1.10 ENCREAD**Description**

The **ENCREAD** function is used for reading encoder parameters. The function should be executed in a buffer, and it will wait till the execution is completed.

Syntax

```
INT ENCREAD (Axis, E_type, ParamType[, Primary])
```

Arguments

| | |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis | The affected axis, valid number are: 0,1,2.. up to the number of axes in the system minus 1 |
| E_type | Encoder Type, according to ACSPL+ E_TYPE variable definition. |
| ParamType | Parameter Type to read. Supported values are: <ul style="list-style-type: none"> > 0 – Resolution (total number of bits, single turn + multi turn) > 1 – Maximum Frequency (in KHz) |

| | |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Primary | <p>Optional. Defines the feedback – can be primary or secondary.</p> <p>Possible values:</p> <ul style="list-style-type: none"> > Primary: 1 (default) > Secondary: 0 |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Return Value

Parameter value according to the requested ParamType.

Comments

Only Endat encoders are supported. For any other **E_TYPE**, the functions returns error 3196 "Requested Absolute Encoder is not supported".

E_TYPE value is changed to 10 (Endat) after the function is called.

The function can be called only if the axis is disabled.

The function can be called only from a buffer; if called from the terminal, error 2073 is returned.

This variable is supported in version 3.10 and higher.

Example

```
I0=encread(0,10,0) !Axis 0, Endat encoder, Resolution
I1=encread(0,10,1) !Axis 0, Endat encoder, Max Frequency
```



This command is supported in the IDMsM/IDMsA/ECMsM/ECMsA/UDMsM/UDMsA products only, and only for EnDAT encoders.

2.1.11 FCLEAR**Description**

FCLEAR clears the **FAULT** variable and the results of the previous fault stored in **MERR**.

Syntax

FCLEAR *axis_list*

Arguments

| | |
|------------------|-------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------------|-------------------------------------------------------------------------------------------------------|

Comments

- > Motor specification is a single axis like 0 or 13, a string of axes enclosed in parentheses and separated by commas, for example: (0,1,13), or the keyword: ALL for all axes.
- > If the axis designation is omitted the command clears the system faults. If an axis is specified, the command clears the **FAULT** and **MERR** components for the specified axes. However, if the reason for the fault is still active, the controller will immediately set the fault again following **FCLEAR**.

- > If one of the cleared faults is an encoder error, **FCLEAR** also resets the feedback position to zero.

Related ACSPL+ Variables

MERR, FAULT

COM Library Methods and .NET Library Methods

FaultClear, FalutClearM

C Library Functions


acsc_FaultClear, acsc_FaultClearM

Example 1:

```
FCLEAR (0,1)           !Clear FAULT and MERR variables for 0 and 1 axes
```

Example 2:

```
FCLEAR ALL           !Clear FAULT and MERR variables for all axes
```



The FCLEAR.ALL command may cause an increase in USAGE.

2.1.12 FOLLOW

Description

FOLLOW switches an axis into slave mode. The specified axis will follow the profile generated by the RTC6.

Syntax

FOLLOW(axis)

Argument

| | |
|------|---------------------------------------------------------------------------------------|
| axis | Axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------|---------------------------------------------------------------------------------------|

2.1.13 GO

Description

GO starts a motion that has been created using the /w (wait) switch.

Syntax

GO *axis_list*

Arguments

| | |
|------------------|-------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------------|-------------------------------------------------------------------------------------------------------|

Comments

- > Motor specification is a single axis like 0 or 13, a string of axes enclosed in parentheses and separated by commas, for example: (0,1,13), or the keyword: ALL for all axes.
- > Where **GO** specifies a single axis, that axis may not be included in any group. **GO** starts the last created motion for the same axis. If the motion was not created, or has been started before, the command has no effect.
- > Where **GO** specifies a leading axis in a group, **GO** starts the last created motion for the same axis group. If the motion was not created, or has been started before, the command has no effect.

COM Library Methods and .NET Library Methods

Go, GoM

C Library Functions

acsc_Go, acsc_GoM

Related ACSPL+ Commands[HALT](#), [MSEG...ENDS](#), [JOG](#), [MPTP...ENDS](#), [PATH...ENDS](#), [PTP](#), [PVSPLINE...ENDS](#), [SLAVE](#), [TRACK](#)**Example**

```
PTP/w (0,1), 1000, 1000      !Create PTP motion, but do not start
PTP/w 3, 8000              !Create PTP motion, but do not start
GO (0,1)                   !Start both motions at the same time
```

2.1.14 GROUP**Description**

GROUP defines an axis group for coordinate multi-axis motion. The first axis in the axes list is the leading axis. The motion parameters of the leading axis become the default motion parameters for all axes in the group. Motion on all axes in a group will start and conclude at the same time.

Syntax**GROUP** *axis_list***Arguments***axis_list*

Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

Comments

An axis can belong to only one group at a time. If the application requires restructuring the axes, it must split the existing group and only then create the new group.

Related ACSPL+ Variables[VEL](#), [ACC](#), [DEC](#), [JERK](#), [KDEC](#), [GVEC](#), [GVEL](#), [GACC](#), [GJERK](#)**Related ACSPL+ Commands**[SPLIT](#)

COM Library Methods and .NET Library Methods

Group

C Library Functions

acsc_Group

Example

```
GROUP (0,1)           !Creates an axis group that includes axes 0 and 1.
PTP (0,1), 1000, 10000 !PTP axis 0 to 1000, and axis 1 to 10000.
```

2.1.15 HALT

Description

In single axis motion, **HALT** terminates currently executed motion and clears all other motions waiting in the axis motion queue. The deceleration profile is defined by **DEC** (deceleration variable).

In group motion, **HALT** terminates currently executed motion of all group axes, and clears all other motions waiting in the axes motion queues. The deceleration profile is defined by the **DEC** (deceleration) variable of the leading axis.

Syntax

HALT *axis_list*[*reason*]

Arguments

| | |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | List of axes to be halted. |
| <i>reason</i> | An optional argument, which must be an integer constant or expression that equates to an integer, and specifies a cause why the axis was halted. |

Switches

| | |
|----|----------------------------------------------------------------|
| /e | Wait for motion termination before executing the next command. |
|----|----------------------------------------------------------------|

Comments

HALT ALL terminates the motion of all axes.

Related ACSPL+ Commands and .NET Library Methods

[KILL/KILLALL](#)

COM Library Methods

Halt, HaltM

C Library Functions

acsc_Halt, acsc_HaltM

Example 1:

```
HALT 0           !Terminates 0 axis motion, the deceleration is
                 !according to DEC(0)
```

Example 2:

```
HALT (0,2)           !Terminates currently executed motion on axes
                    !0 and 2.
```

Example 3:

```
HALT ALL           !Terminates currently executed motion on all axes.
```

2.1.16 HOME**Description**

The predefined **HOME** command receives the following parameters: Axis, HomingMethod, HomingVel(optional), MaxDistance(optional), HomingOffset(optional), HomeCurrLimit(optional).

Syntax

```
HOME Axis, [opt]HomingMethod, [opt]HomingVel, [opt]MaxDistance,
[opt]HomingOffset, [opt]HomingCurrLimit,
[opt]HardStopThreshold, [opt]SetYawToOpen, [opt]SkewValue, [opt]LookForTwoLS
```

Arguments

| | |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis | axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| HomingMethod | Optional. The number of homing method that should be used for homing. If not specified: the homing method is set according to the value of the HOMEDF variable. |
| HomingVel | Optional. The velocity that will be used for the homing. If not specified: SLCPRD*EFAC/2 |
| MaxDistance | Optional. The Maximum distance that will be used during homing. If not specified, endless motion will be used. |
| HomingOffset | Optional. The machine home position is found during homing. If not specified, 0 is used. If specified, after homing is completed, the zero position is offset from the home position by adding the home offset of the home position. |
| HomingCurrLimit | Optional. Current Limit during the homing process. If not specified: min(XCURV,0.5*XRMSM,0.5*XRMSD) |
| HardStopThreshold | Optional. If specified, Hard Stop will be identified by: Abs(PE)>min(HardStopThreshold,CERRV*0.75) If not specified: Abs(PE)>CERRV*0.75 |

| | |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SetYawToOpen | For Gantry mode only. Optional. 0 – Complementary Axis keeps OPEN LOOP state during homing operation 1- Complementary Axis is set to OPEN LOOP state during homing operation |
| SkewValue | For Gantry mode only. Optional. Used for setting positing for the complementary axis after the homing process is finished. If not specified, the value is 0. |
| LookForTwoLS | For Gantry mode only. Optional. 0 (default) - only one limit is detected 1- both limits are detected |

Homing Methods

Table 2-2. Homing Methods

| Method Number | Explanation |
|---------------|--------------------------------------------------------------------------------------|
| 1 | Homing Method 1: Homing on the negative limit switch and index pulse |
| 2 | Homing Method 2: Homing on positive limit switch and index pulse |
| 17 | Homing Method 17: Homing on Negative Limit Switch |
| 18 | Homing Method 18: Homing on Positive Limit Switch |
| 33/34 | Homing Method 33 and 34: Homing on the index pulse |
| 37 | Homing Method 37: Homing on current position |
| 50 | Homing Method 50: Negative Hard Stop and index pulse (ACS Specific) |
| 51 | Homing Method 51: Positive Hard Stop and index pulse (ACS Specific) |
| 52 | Homing Method 52: Negative Hard Stop (ACS Specific) |
| 53 | Homing Method 53: Positive Hard Stop (ACS Specific) |

Comments

- > The **HOME** command is non-waiting
- > **MFLAGS.#HOME** bit will be set to 1 after the homing is completed.
- > **AST.#INHOMING** bit is 1 during the homing process • **E_TYPE** and other encoder initialization processes will reset the **#HOME** bit
- > The predefined homing methods are defined according to the DS402 standard, except methods 50, 51, 52, 53



- > If the homing method is not supported, the error 3314 "Requested Homing Method is not supported" is given.
- > The axis is required to be enabled, commutated, and not in motion.
- > Disable axis during homing process will cancel the homing process.
- > Other motions cannot be executed while the axis is in home process
- > The following homing methods are supported in Gantry mode: 1, 2, 50, and 51.

Example 1

```
enable 0
commut 0
home 0,34 !homing on Positive Index Pulse
TILL MFLAGS0.#HOME=1

STOP
```

2.1.17 IMM

In single axis motion, **IMM** provides on-the-fly changes of the following motion parameters:

- > **VEL** (Velocity)
- > **ACC** (Acceleration)
- > **DEC** (Deceleration)
- > **JERK** (Jerk)

IMM affects the motion in progress and all motions waiting in the corresponding motion queue.

In group motion, **IMM** provides on-the-fly changes for the motion parameters of the leading axis only.

Syntax

IMM *axis_motion parameter=value or formula*

Arguments

| | |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_motion parameter</i> | The motion parameter with the specified axis (valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1). |
| <i>value or formula</i> | User specified value or formula. |

Related ACSPL+ Variables

VEL, **ACC**, **DEC**, **JERK**

COM Library Methods and .NET Library Methods

Set Acceleration Imm, Set Deceleration Imm, Set Jerk Imm, Set Kill Deceleration Imm, Set Velocity Imm

C Library Functions

acsc_SetAccelerationImm, acsc_SetDecelerationImm, acsc_SetJerkImm, acsc_SetKillDecelerationImm, acsc_SetVelocityImm

Example

```
IMM VEL(0)=5000           !Immediately change the 0 axis velocity to 5000
```

2.1.18 KILL/KILLALL

Description

Use **KILL** after a safety event to decelerate and stop an axis faster than during normal deceleration and stop. **KILLALL** stops all axes.

In single axis motion, **KILL** terminates currently executed motion and clears all other motions waiting in the axis motion queue. The deceleration profile uses a second-order deceleration profile and the **KDEC** (kill deceleration) value.

In group motion, **KILL** terminates currently executed motion only for the specified axes, and clears all other motions waiting in the axis/axes motion queue. The deceleration profile uses a second-order deceleration profile and the **KDEC** (kill deceleration) variable of each axis.

Syntax

KILL *axis_list*[,*reason*]

Arguments

| | |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | List of axes to be killed, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1, or ALL for all axes. |
| <i>reason</i> | An optional argument, which must be an integer constant or expression that evaluates to an integer, and specifies a cause why the axis was killed. If the parameter is specified, its value is stored in the MERR variable. If the parameter is omitted, MERR stores zero after KILL . |

Switches

| | |
|----|----------------------------------------------------------------|
| /e | Wait for motion termination before executing the next command. |
|----|----------------------------------------------------------------|



For systems having more than 15 axes, the use of **KILL ALL** may cause Over Usage or Servo Processor Alarm faults.

Comments

1. **KILL ALL** terminates the motion of all axes. The deceleration profile is defined by the **KDEC** (kill deceleration) variable of each axis.
2. If several sequential **KILL** operations specify different causes for the same motor, only the first **cause** is stored in **MERR** and all subsequent causes are ignored.
3. A **cause** stored in **MERR** is cleared by **FCLEAR** or **ENABLE/ENABLE ALL**.

Related ACSPL+ Commands

[HALT](#), [FCLEAR](#), [ENABLE/ENABLE ALL](#)

Related ACSPL+ Variables

[MERR](#), [KDEC](#)

COM Library Methods and .NET Library Methods

Kill, Kill All

C Library Functions

acsc_Kill, acsc_KillAll

Example 1:

```
KILL 1                !Kill axis 1 deceleration is according to KDEC(1)
```

Example 2:

```
KILL 0, 5011         !Kill 0 axis, store 5011 as the reason.
                    !Code 5011 corresponds to left limit error;
                    !therefore the 0 motor will be reported as
                    !disabled due to fault involving left limit.
```

Example 3:

```
KILL (0,1,2)        !Kill 0, 1 and 2 axes according to the KDEC
                    !of each specified axis.
```

Example 4:

```
KILL ALL, 6100      !Kills all axes, and stores the cause in MERR.
                    !(6100 is the code for a user-defined cause.)
```

2.1.19 SAFETYCONF

Description

SAFETYCONF configures fault processing for one or more axes, by disabling the default response to a defined axis **FAULT**, and performs one of the following responses:

- > Ignore the interrupt
- > Kill the motion
- > Disable the axis
- > Kill the motion and then disable the axis

Syntax

SAFETYCONF *axis_list*, *fault_name*, "*conf_string*"

Arguments

| | |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>fault_name</i> | Any axis fault name like #LL for left limit. |
| <i>conf_string</i> | <p>A string enclosed in double quotation marks with one or more of the following characters determines the action of SAFETYCONF:</p> <ul style="list-style-type: none"> > K (KILL/KILLALL) > D (DISABLE/DISABLEALL) > KD (KILL/KILLALL-DISABLE/DISABLEALL) > + when a fault occurs in any member of the <i>axis_list</i>, the fault response applies to all axes of the controller (each axis fault response can be unique) > - applies FMASK<<i>axis</i>>.#fault-name = 0 for all axes of the controller |

Comments

If an empty string is specified, fault detection is enabled, but the controller has no response to the fault. However, an autoroutine can intercept the fault and provide a response.

In devices implementing STO, the user may not use **SAFETYCONF** to change the default response to an STO fault, which is **KILL**<*axis*> + **DISABLE**<*axis*>.

Related ACSPL+ Commands

The **#SC** Communication Terminal command shows the current fault response configuration for all axes.

Example 1:

```
SAFETYCONF 0,#PE,"K"           !The 0 motion will be killed if the 0
                               !Position Error fault occurs.
```

Example 2:

```
SAFETYCONF (3,5),#LL,"KD"     !Changes the response to the Left Limit fault
of                             of
                               !3 and 5 axes to KILL and then DISABLE.
```

Example 3:

```
SAFETYCONF ALL,#DRIVE,"D+"    !All axes will be disabled if the Drive
                               !Alarm fault occurs in any axis.
```

Example 4:

```
SAFETYCONF ALL,#VL,"-"           !Velocity Limit fault will be masked for
                                !all axes.
```

2.1.20 SAFETYGROUP**Description**

SAFETYGROUP activates the fault response for all axes in the *axis_list* when any axis triggers the fault, and manages the axes as a block in response to [KILL/KILLALL](#) and [DISABLE/DISABLEALL](#).

To cancel the defined **SAFETYGROUP**, send the command again with only the first axis as the *axis_list*.

Syntax

SAFETYGROUP *axis_list*

Arguments

axis_list

List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

COM Library Methods and .NET Library Methods

GetSafetyInputPort

C Library Functions

acsc_GetSafetyInputPort

Example 1:

```
SAFETYGROUP (0,1,5)             !Creates safety group for axes 0, 1 and 5.
```

Example 2:

```
SAFETYGROUP 0                   !Cancels the previously created safety group for
                                !axes 0, 1 and 5.
```

2.1.21 SET**Description**

SET defines the current value of either feedback ([FPOS](#)), reference ([RPOS](#)), or axis ([APOS](#)) position. **SET** can be initiated when the axis is disabled, or on-the-fly. [APOS](#) and [FPOS](#) are updated automatically when **SET** is specified for [RPOS](#),

If a non-default [CONNECT](#) is used, assign different values to [APOS](#) and [RPOS](#).

Related ACSPL+ Variables

[RPOS](#)

Syntax

SET *axis_RPOS*=*value or formula*

Arguments

| | |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_RPOS</i> | The reference position of the specified axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| = | The assignment operator |
| <i>value or formula</i> | User-defined value or formula |

COM Library Methods and .NET Library Methods

SetRPosition, SetFPosition

C Library Functions

acsc_SetRPosition, acsc_SetFPosition

Example

```
SET RPOS(0)=300           !Axis 0 RPOS = 300
```

2.1.22 SPLIT**Description**

SPLIT breaks down a group created using **GROUP** by designating any axis in the axis list. **SPLIT ALL** breaks down all groups.



If the **SPLIT** command specifying an axis that is currently in motion is executed within the buffer, the buffer execution is suspended until the motion is completed. However, if the **SPLIT** command is sent from the host or as a **Communication Terminal** command, it returns error 3087: "Command cannot be executed while the axis is in motion".

Syntax**SPLIT** *axis_list***Arguments**

| | |
|------------------|-------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis or list of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------------|-------------------------------------------------------------------------------------------------------|

Related ACSPL+ Commands**GROUP****COM Library Methods and .NET Library Methods**

Split, SplitAll

C Library Functions

acsc_Split, acsc_SplitAll

Example 1:

```
GROUP (0,1,5)      !Create an axis group including axes 0, 1 and 5.
GROUP (2,3,7)      !Create an axis group including axes 2, 3 and 7.
SPLIT 0            !Breaks down axis group 0, 1 and 5.
```

Example 2:

```
GROUP (0,1,5)      !Create an axis group including axes 0, 1 and 5.
GROUP (2,3,7)      !Create an axis group including axes 2, 3 and 7.
SPLIT ALL          !Breaks down axis group 0, 1 and 5
                  !and group 2, 3 and 7.
```

2.1.23 UNFOLLOW

Description

UNFOLLOW switches an axis into regular mode. The specified axis will follow the profile generated by the ACS controller.

Syntax

UNFOLLOW(axis)

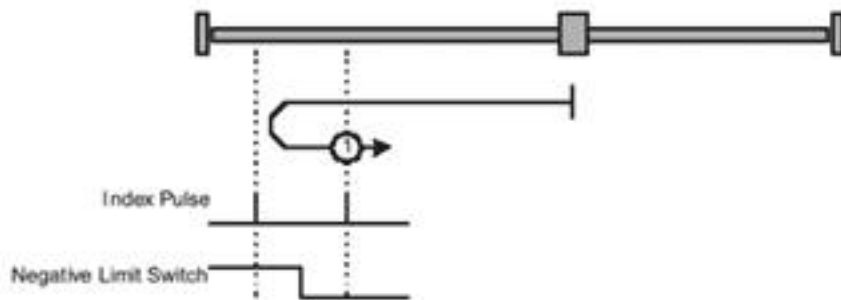
Argument

| | |
|------|---------------------------------------------------------------------------------------|
| axis | Axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------|---------------------------------------------------------------------------------------|

2.2 Predefined Homing Methods

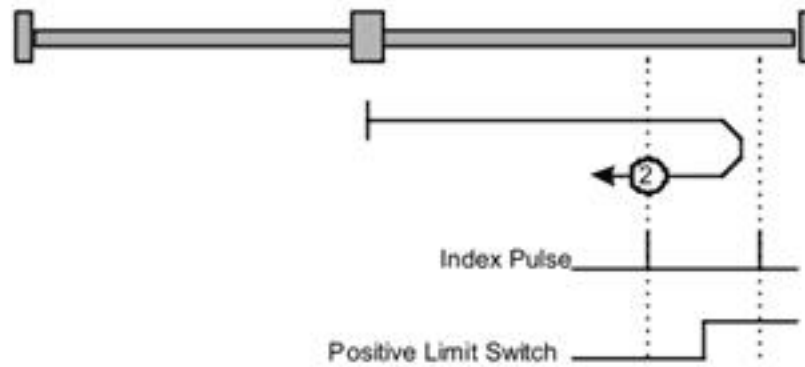
2.2.1 Homing Method 1: Homing on the negative limit switch and index pulse

With this homing method the initial direction movement is leftward if negative limit switch is inactive (shown as low in the figure above). The home position is at the first index pulse right of the position where the negative limit switch becomes active.



2.2.2 Homing Method 2: Homing on positive limit switch and index pulse

With this homing method the initial direction movement is rightward if the positive limit switch is inactive (shown as low in the figure above). The home position is at the first index pulse left of the position where the negative limit switch becomes active.



2.2.3 Homing Method 17: Homing on Negative Limit Switch

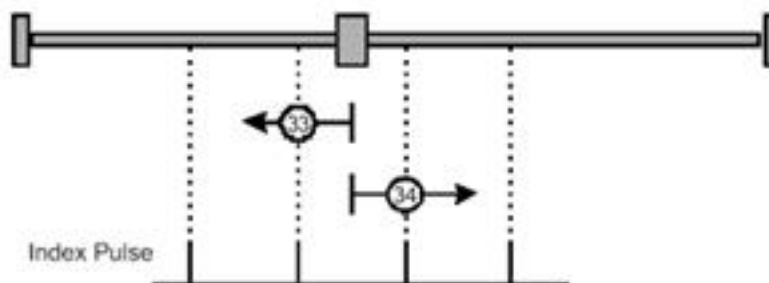
Homing method 17 is similar to method 1, except that it homes on negative limit switch only. If negative limit switch is ON when the function is called, the homing is aborted (no homing attained).

2.2.4 Homing Method 18: Homing on Positive Limit Switch

Homing method 18 is similar to method 2, except that it homes on positive limit switch only. If positive limit switch is ON when the function is called, the homing is aborted (no homing attained).

2.2.5 Homing Method 33 and 34: Homing on the index pulse

Using methods 33 or 34, the direction of homing is negative or positive, respectively. The home position is at the index pulse found in the selected direction.



2.2.6 Homing Method 37: Homing on current position

In this method, the current position shall be taken to be the home position. FW implements the following ACSPL+ code:

```
SET FPOS (<axis>) =HomeOffset.
```

2.2.7 Homing Method 50: Negative Hard Stop and index pulse (ACS Specific)

With this homing method the initial direction movement is negative, till Hard Stop is found (by using the Position Error indication). The home position is at the first index pulse right of the position where the Hard Stop is found.

Position Error is considered as Hard Stop only if the HardStopThreshold condition is met.

2.2.8 Homing Method 51: Positive Hard Stop and index pulse (ACS Specific)

With this homing method the initial direction movement positive, till Hard Stop is found (by using the Position Error indication). The home position is at the first index pulse left of the position where the Hard Stop is found.

2.2.9 Homing Method 52: Negative Hard Stop (ACS Specific)

With this homing method the initial direction movement is negative, till Hard Stop is found (by using the Position Error indication). After the Hard Stop is found, the axis moves back to where the Hard Stop was detected, continues threshold (this is the home position), continues additional threshold multiplied by 2 and halts.

Position Error is considered as Hard Stop only if the HardStopThreshold condition is met.

2.2.10 Homing Method 53: Positive Hard Stop (ACS Specific)

With this homing method the initial direction movement is positive, till Hard Stop is found (by using the Position Error indication). After the Hard Stop is found, the axis moves back to where the Hard Stop was detected, continues threshold (this is the home position), continues additional threshold multiplied by 2 and halts.

2.3 Interactive Commands

The Interactive commands are:

| Command | Description |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>DISP</code> | Builds a string and sends it to the default communication channel. |
| <code>INTERRUPT</code> | Causes an interrupt that can be intercepted by the host. |
| <code>INTERRUPTEX</code> | Causes an interrupt similar to that of the <code>INTERRUPT</code> command. |
| <code>SEND</code> | Same as <code>DISP</code> , but also specifies the communication channel or channels. |
| <code>TRIGGER</code> | Specifies a triggering condition. Once the condition is satisfied, the controller issues an interrupt to the host computer. |

2.3.1 DISP

Description

`DISP` builds an ASCII output string and sends it to a communication channel. The ASCII output can include text segments and variable values defined in various format displays. The output string is sent to the default communication channel defined by the standard variable `DISPCH`.

Syntax

`DISP` *argument* [, *argument* . . .]

Arguments

| | |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>argument</i> | An expression or string where: Expression: ACSPL+ expression as one or more variables String: "[text] [escape-sequence] [format-specifier]" String must be enclosed with double quotation marks. |
| <i>[, argument. . .]</i> | Optional subsequent arguments. |

Command Options

An input string can include one or more of the following:

- > Text
Escape Sequence - appears in the output string as a non-printing character or other specified character.
- > **Formatting Specification** - determines how the results of an expression that follows the input string is formatted in the output string.

Table 2-3. DISP Command Option Escape Sequences

| Escape Sequence | Added Character to Output String |
|-----------------|---------------------------------------------------------------------------------|
| \r | Carriage return - 0x0d |
| \s | Avoid carriage return |
| \n | New line - 0x0a |
| \t | Horizontal tab - 0x09 |
| \xHH | Any ASCII character where HH represents the ASCII code of the character. |



The output format specification syntax adheres to a restricted version of the C language syntax.

The format specification syntax is:

```
% [width] [.precision] type
```

where:

| | |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [width] | Optional specification for the minimum number of characters in the output. If width is smaller than the number of digits of a displayed number, the specified width is ignored, and the displayed number includes all digits. |
| [.precision] | Optional number that specifies the maximum number of characters printed for all or part of the output field, or the minimum number of digits printed for integer values. |
| type | Required character that determines whether the associated argument is interpreted as a character, a string, or a number, as described in Table 2-4 . |

Table 2-4. Type Characters

| Type | Output Format |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| d | Signed decimal integer |
| u | Unsigned decimal integer |
| x | Unsigned hexadecimal integer, using "abcdef" |
| X | Unsigned hexadecimal integer, using "ABCDEF" |
| e | Signed value having the form [-]d.dddd e [sign]ddd where d is a single decimal digit, dddd is one or more decimal digits, ddd is exactly three decimal digits, and sign is + or -. |
| E | Identical to the e format except that E rather than e introduces the exponent. |
| f | Signed value having the form [-]dddd.dddd, where dddd is one or more decimal digits. The number of digits before the decimal point depends on the magnitude of the number, and the number of digits after the decimal point depends on the requested precision. |
| g | Signed value printed in f or e format, whichever is more compact for the given value and precision. The e format is used only when the exponent of the value is less than -4 or greater than or equal to the precision argument. Trailing zeros are truncated, and the decimal point appears only if one or more digits follow it. |
| G | Identical to the g format, except that E , rather than e , introduces the exponent (where appropriate). |

Comments

1. If an input string argument contains **n** format specifiers, the specifiers apply to the **n** subsequent expression arguments.
2. **DISP** processes arguments from left to right, as follows:

- > **Expressions:** The expression is evaluated and the ASCII representation of the result is placed in the output string. The format of the result is determined by the formatting specifications (if any) in the input string.
 - > **Input strings:** Text is sent as-is to the output string. Escape sequences are replaced by the ASCII codes that they represent. Formatting specifications are applied to the results of any expressions that follow the string
3. **DISP** cannot be used from the SPiiPlus MMI Application Studio **Communication Terminal**, only from a program buffer.
 4. **DISP** can only display the value of a single element of an array.

In order to receive unsolicited messages by a host application, perform the following:

1. Set **DISPCH** to -2.
2. Set bit 4 of **COMMFL** to 1.
3. Send **SETCONF**(306,-1,1) from the same communication channel where unsolicited messages are expected to be received.

In order to stop the receipt of unsolicited messages by a host application: send **SETCONF**(306,-1,0) from the same communication channel where there is no need any more to receive unsolicited messages.

Related ACSPL+ Commands

[SEND](#), [SETCONF](#)

Related ACSPL+ Variables

[DISPCH](#), [COMMFL](#)

COM Library Methods and .NET Library Methods

OpenMessageBuffer, GetSingleMessage, CloseMessageBuffer

C Library Functions

acsc_OpenMessageBuffer, acsc_GetMessage, acsc_CloseMessageBuffer

Example 1:

```
DISP "%15.10f",FPOS0      !Display FPOS0 in 15 digits with 10 digits
                          !following the decimal point.
                          !Output: 997.2936183303
STOP
```

Example 2:

```
DISP "0 FVEL=%15.10f", FVEL0      !Output: 0 FVEL= 997.2936183303
STOP
```

Example 3:

```
DISP "%i",IN0.2           !Output: the current state of IN0.2 as
                          !one digit 0 or 1.
STOP
```

Example 4:

```
DISP "IN0 as hex: %04X",IN0      !Output: IN0 as hex: 0A1D
STOP
```

Example 5:

```
DISP "IN0.0-3 as binary: %1u%1u%1u%1u", IN0.0,IN0.1,IN0.2,IN0.3
                          !Output: IN0.0-3 as binary: 0110
STOP
```

Example 6:

```
DISP "Elapsed time is: ", TIME !Output: Elapsed time is: 4.93258E+006
STOP
```

Example 7:

```
DISP "Expression, default formatting:", FPOS0*2+FPOS1+5 , FPOS1
                          !Output: Expression, default formatting: 6.28657
                          !0.286485
STOP
```

Example 8:

```
REAL AXIS_NAME ; AXIS_NAME=0 ;
DISP "Axis",AXIS_NAME," was disabled due to error code", MERR(AXIS_NAME)
      !Display the reason of axis disable due to a fault.
      !Output: Axis 0 was disabled due to error code 0
STOP
```

Example 9:

```
DISP "%5i\r", FPOS0, "%5i", FPOS1
      !Standard format, minimum 5 positions, no decimals,
      !and a carriage return between the two values.
      !Output:
      !711
      !2024
STOP
```

Example 10:

```
DISP "Hexadecimal format: %08X", MFLAGS(0), " and also %08x", MFLAGS(0)
      !Hexadecimal format, minimum 8 positions, capital
      !letters or lower case letters.
      !Output: Hexadecimal format: 002A2300 and also
      !002a2300

STOP
```

Example 11:

```
DISP "0 FPOS: %15.3e", FPOS0," and 1 FPOS: %15.3e", FPOS1
      !Scientific format, minimum 15 positions, 3 decimals
      !of FPOS0 and FPOS1.
      !Output:
      !0 FPOS: 5.000e-001 and
      !1 FPOS: 2.865e-001

STOP
```

Example 12:

```
REAL AAA;           !Standard or scientific format, small letters or
                    !capital letters.
AAA=10;             !Assigning integer value to AAA
DISP "%g", AAA;    !Output: 10
AAA=1e9            !Assigning Hex value to AAA
DISP "%g", AAA     !Output: 1e+009
DISP "%G", AAA     !Output: 1E+009
STOP
```

2.3.2 INP

Description

INP reads data values from a specified channel and stores them to an integer array. This function is useful for creating an interface between the controller and special input devices such as a track-ball, mouse or various sensors. **INP** is also used when the controller acts as a master with the **MODBUS** protocol communication.

Before using **INP**, configure the relevant communication channel as a special communication channel using **SETCONF** function, key 302.

See also **OUTP**.

Syntax

int **INP**(channel, [array,] [start_index,] [number,] [timeout])

Arguments

| | |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>channel</i> | <p>Communication channel index:</p> <ul style="list-style-type: none"> > 1 - serial communication channel COM1 > 2 - serial communication channel COM2 > 6 - Ethernet network (TCP) > 7 - Ethernet network (TCP) > 8 - Ethernet network (TCP) > 9 - Ethernet network (TCP) > 10 - Ethernet Point-to-Point (UDP) > 12 - PCI bus > 16 - communication channel with Modbus slave > 36 - Ethernet network (TCP) > 37 - Ethernet network (TCP) > 38 - Ethernet network (TCP) > 39 - Ethernet network (TCP) |
| <i>array</i> | <p>User-defined integer array to which the data will be stored.</p> <p>If array is omitted, the function purges formerly received characters in the channel.</p> |
| <i>start_index</i> | <p>The first received character is assigned to the array element with the specified index.</p> <p>If start_index is omitted, the assignment starts from the first element of the array.</p> |
| <i>number</i> | <p>The number of characters to be collected to the variable array.</p> <p>If number is omitted, the function continues receiving characters until the last element of the array is assigned, or the carriage return character is received.</p> |
| <i>timeout</i> | <p>The function waits for input not more than the specified number of milliseconds.</p> <p>If timeout is omitted, the waiting time is not limited.</p> |

Return Value

The number of entities that have been stored into the variable.

Error Conditions

None

Example

```

GLOBAL INT MMM(10)      !Defines global user array MMM with ten elements.
SETCONF(302,2,1)      !Assigns COM2 as special input.
INP(2)                 !Purges the input buffer from old values
INP(2,MMM,0,10,1000)  !INP(2)- purge the input buffer from old values
                       !INP(2,MMM,0,10,1000) - store values from COM2 port
                       !to MMM user variable, from array index 0, total of
                       !10 values. The data collection process will end
                       !within 1000 msec or when the 10 values have been
                       !collected.
STOP                   !Ends program

```

2.3.3 INTERRUPT**Description**

INTERRUPT causes an unconditional trigger that is intercepted by the host. Once a program executes **INTERRUPT**, the interrupt signal is sent to the host application. This interrupt is detected by the COM library **EnableEvent** or C Library **acsc_SetCallback** functions which then call interrupt type **ACSC_INTR_ACSPL_PROGRAM**.

Syntax**INTERRUPT****Related ACSPL+ Commands****TRIGGER****COM Library Methods**

EnableEvent, DisableEvent, SetCallbackMask, SetCallbackPriority, GetCallbackMask

C Library Functions

acsc_InstallCallback, acsc_SetCallbackMask, acsc_SetCallbackPriority, acsc_GetCallbackMask

Example 1:

INTERRUPT used in an ACSPL+ program:

```

ENABLE 0                !Enable axis 0
SET FPOS0=0            !Set axis 0 feedback position = 0
PTP 0, 1000           !ACSPL+ executes a PTP motion.
TILL MST(0).#MOVE=0    !The motor reaches the destination point
                       !and stops.
INTERRUPT              !INTERRUPT is sent to the host application.
STOP

```

Example 2:

INTERRUPT used in a Host COM Lib application:

```

SET ch = New Channel    !Initialize ch as a COM library object
CALL ch.EnableEvent(ch.ACSC_INTR_ACSPLPROGRAM)
                       !Enable INTERRUPT as an event. The host application

```

```

        !waits for an interrupt from the controller initiated
        !by INTERRUPT from within the ACSPL+ program.
Private Sub ch_ACSPLPROGRAM(ByVal Param As Long) MsgBox ("Motion is
Stopped")
        !When an interrupt occurs, launch a message box
        !displaying "Motion is Stopped"
EndSub

```

Example 3:

INTERRUPT used in a Host C Lib application:

```

int WINAPI CallbackInput (unsigned __int64 Param,void* UserParameter);
// will be defined late
...
int CardIndex;//Some external variable, which contains card index
// set callback function to monitor digital inputs
if (!acsc_InstallCallback(Handle, // communication handle
    CallbackInput,           // pointer to the user callback function
    &CardIndex,              // pointer to the index
    ACSC_INTR_INPUT         // Type of callback
))
{
printf("callback registration error: %d\n", acsc_GetLastError());
}
...
// If callback was installed successfully, the CallbackInput function
will
// be called each time the any digital input has changed from 0 to 1.
// CallbackInput function checks the digital inputs 0 and 1
int WINAPI CallbackInput (unsigned __int64 Param,void* UserParameter)
{
if (Param & ACSC_MASK_INPUT_0 && *UserParameter == 0)
//Treat input 0 only for card with index 0
{
    // input 0 has changed from 0 to 1
    // doing something here
}
if (Param & ACSC_MASK_INPUT_1 && *UserParameter == 1)
    //Treat input 1 only for card with index 1
{
    // input 1 has changed from 0 to 1
    // doing something here
}
return 0;
}
}

```

2.3.4 INTERRUPTEX

Description

The **INTERRUPTEX** command operates in a manner similar to **INTERRUPT** but has the following differences:

- > It triggers the dedicated callback **ACSC_INTR_ACSPL_PROGRAM_EX** (21)
- > It accepts two mandatory integer parameters and two optional parameters. Their values will be passed along with the interrupt to the host instead of the buffer mask passed in the old **INTERRUPT** function as a 64-bit integer.
- > Adjacent (“glued”) interrupts are processed differently (because parameters are not OR’ed):
 - > There is an internal queue of 256
 - > The next interrupt value will be triggered only after C Library delivers the previous one
 - > The maximum output rate is 1 interrupt per **CTIME**
 - > On queue overflow, the interrupt is lost

Syntax

INTERRUPTEX (*32-bit_high_value,32-bit_low_value[,32-bit_high_value_second,32-bit_low_value_second]*)

Arguments

| | |
|--------------------------|------------------------------------------------------------|
| 32-bit_high_value | Most significant value of a combined 64-bit integer |
| 32-bit_low_value | Least significant value of a combined 64-bit integer |
| 32-bit_high_value_second | Optional. High 32 bits of a combined second 64-bit integer |
| 32-bit_low_value_second | Optional. Low 32 bits of a combined second 64-bit integer |

Comments

INTERRUPTEX is supported by both by SPiiPlusNT and SPiiPlusSC products. For the SPiiPlusSC-HP products, the interrupt is passed via Shared Memory, which makes it very fast ($100+(\text{Cycle-time})/2$ for the round trip on the average). For the SPiiPlusNT and the SPiiPlusSC-LT products, the interrupt is passed via the communication channel (Ethernet/Serial RS-232).



An application that uses C Library must make sure to empty the queue, register the callback and wait enough time until the queue is empty.

The parameters are passed to the host as a single 64-bit integer with the first parameter as the 32-bit most significant value word and the second parameter is the 32-bit least significant word.

COM Library Methods

EnableEvent, DisableEvent, SetCallbackMask, SetCallbackPriority, GetCallbackMask

C Library Functions

acsc_InstallCallback, acsc_SetCallbackMask, acsc_SetCallbackPriority, acsc_GetCallbackMask

Example 1:

```
Enable 0          ! Enable axis 0
SET FPOS0=0      ! Set axis 0 position to 0
PTP/e 0,1000    ! Move to position 1000
INTERRUPTTEX (0x12, 0x34)! interrupt with parameter, host will receive
                  ! 0x0000001200000034

Stop
```

Example 2:

```
Global integer interrupt_description (0x100)
Global integer interrupt_queue
interrupt_queue = 0
interrupt_buff =0          ! No pending interrupts
Enable 0                  ! Enable axis 0
SET FPOS0=0              ! Set axis 0 position to 0
interrupt_description(interrupt_queue)= 0x1    ! Setting description to 1
                                                ! indicating pre-motion
                                                ! state
interrupt_queue = (interrupt_queue+1)&0xff    ! One more pending interrupt
INTERRUPTTEX (0x00, interrupt_queue)         ! Interrupt description in
                                                ! interrupt_description (0)
PTP/e 0,1000                                ! Move to position 1000
interrupt_description(interrupt_queue) = 0x2  ! Setting description to
2
                                                ! indicating post-motion
                                                ! state
interrupt_queue = (interrupt_queue+1)&0xff    ! One more pending interrupt
INTERRUPTTEX (0x00, interrupt_queue)         ! Interrupt description in
                                                ! interrupt_description (1)

Stop
```

Example 3:

```
Enable 0          !Enable axis 0
SET FPOS0=0      !Set axis 0 to position 0
PTP/e 0,1000    !Move to position 1000
INTERRUPTTEX (0x12, 0x34, 0x22, 0x54)
                !interrupttex with parameter, host will receive the following;
                ! 0x0000001200000034, 0x0000002200000054
```

2.3.5 SEND

Description

SEND is the same as **DISP**, but also specifies the communication channel for the output string.

Syntax

SEND *channel_number, argument [, argument...]*

Arguments

| | |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>channel-number</i> | <p>An integer that defines the communication channel to which the message will be sent, as follows:</p> <ul style="list-style-type: none"> -2: All channels -1: Last used channel 1: Serial Port 1 2: Serial Port 2 6: Ethernet network (TCP) 7: Ethernet network (TCP) 8: Ethernet network (TCP) 9: Ethernet network (TCP) 10: Ethernet point-to-point (UDP) 16: MODBUS Slave 36: Ethernet network (TCP) 37: Ethernet network (TCP) 38: Ethernet network (TCP) 39: Ethernet network (TCP) |
| <i>argument</i> | <p>An expression or string where:</p> <p>Expression: ACSPL+ expression as one or more variables</p> <p>String: "[text] [escape-sequence] [format-specifier]" String must be enclosed with double quotation marks.</p> |
| <i>[, argument. . .]</i> | Optional subsequent arguments. |

Command Options

For a list of **Command Options**, relevant **Comments**, and **Examples**, see [DISP](#).

Related ACSPL+ Commands

[DISP](#)

Related ACSPL+ Variables

[DISPCH](#)

COM Library Methods

Send

C Library Functions

acsc_Send

2.3.6 TRIGGER

Description

TRIGGER specifies a triggering condition. Once the condition is satisfied, the controller issues an interrupt to the host computer, as follows:

1. Sets **AST<axis>.#TRIGGER** = 0
2. Examines the triggering condition every MPU cycle

Once the condition is satisfied, the controller performs the following:

1. Sets **AST<axis>.#TRIGGER** = 1
2. Produces an interrupt to the host application (software interrupt 10, enabled by **IENA.26**).

The controller continues calculating the **TRIGGER** expression until another **TRIGGER** command is executed in the same channel. Each time the expression changes its value from zero to non-zero, the controller sets **AST<axis>.#TRIGGER** = 1 and causes an interrupt.



Full application of the **TRIGGER** command to channels greater than 7 is not currently supported.

Syntax

TRIGGER *channel*, *expression* [, *timeout*]

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>channel</i> | An integer number from 0 to 7 that specifies the trigger's sequential number. The number defines the AST element where the triggering bit will be set and defines the bit in the interrupt tag that is automatically sent to the host application as an interrupt. |
| <i>expression</i> | Specifies the triggering condition. After TRIGGER is executed, the controller checks the expression condition each MPU cycle. Triggering occurs when the expression condition is satisfied. If the argument is omitted, triggering in the specified channel is disabled. |
| <i>timeout</i> | Specifies triggering timeout in milliseconds. A positive number specifies the time allowed for the triggering condition to be satisfied. If the timeout has elapsed and the triggering condition has not been satisfied, the controller unconditionally raises the trigger bit. After any triggering, the controller resets timeout counting to zero. If the argument is omitted, triggering works without timeout. |

Table 2-5. Channel Designation for TRIGGER

| Channel | Triggering Bit | Hexadecimal Interrupt Tag (Software Interrupt 10) |
|---------|----------------|------------------------------------------------------|
| 0 | AST0.11 | 0x00000001 |
| 1 | AST1.11 | 0x00000002 |
| 2 | AST2.11 | 0x00000004 |
| 3 | AST3.11 | 0x00000008 |
| 4 | AST4.11 | 0x00000010 |
| 5 | AST5.11 | 0x00000020 |
| 6 | AST6.11 | 0x00000040 |
| 7 | AST7.11 | 0x00000080 |

Related ACSPL+ Commands**INTERRUPT****Related ACSPL+ Variables****IENA, AST****COM Library Methods**

GetCallbackMask, SetCallbackMask

C Library Functions

acsc_GetCallbackMask, acsc_SetCallbackMask

Example

```

TRIGGER 1, MST(0).#MOVE=0, 3000      !1 - once the triggering condition is satisfied,
                                     !the triggering bit AST1.#TRIGGER will be set
                                     !to "1", and the interrupt tag to the host
                                     !application is 0x00000002.
                                     !MST(0).#MOVE=0 - the triggering condition.
                                     !Actuate trigger interrupt when the motor in
                                     !the 0 axis is in position (not moving).
                                     !3000 - check the triggering condition for
                                     !3000 msec. If the triggering condition is not
                                     !satisfied after 3000 msec, then set the
                                     !triggering bit AST(1).TRIGGER to "1".

```

2.3.7 OUTP**Description**

OUTP sends data values from an integer array to a specified channel. This function is useful to create an interface between the controller and special input devices such as a track-ball, mouse and

various sensors. **OUTP** is also used when the controller acts as a master with the **MODBUS** protocol communication.

Before using **OUTP**, configure the relevant communication channel as a special communication channel using **SETCONF** function, key 302.

Each ASCII character is represented by its numerical value and is stored in a separate element of the array.

The user might have to define communication parameters for the special communication channel with **SETCONF** function keys 302, 303, 304, 309.

See also [INP](#).

Syntax

```
int OUTP(channel, variable[, start_index[, number]])
```

Arguments

| | |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| channel | Communication channel index: <ul style="list-style-type: none"> > 1 - serial communication channel COM1 > 2 - serial communication channel COM2 > 6 - Ethernet network (TCP) > 7 - Ethernet network (TCP) > 8 - Ethernet network (TCP) > 9 - Ethernet network (TCP) > 10 - Ethernet Point-to-Point (UDP) > 12 - PCI bus > 16 - communication channel with MODBUS slave > 36 - Ethernet network (TCP) > 37 - Ethernet network (TCP) > 38 - Ethernet network (TCP) > 39 - Ethernet network (TCP) |
| variable | User-defined integer array from which the data will be sent. |
| start_index | The index in the array from which to start. If start_index is omitted, number should also be omitted (in this case, all members of the array are transmitted). |
| number | The number of characters to be transmitted from the variable array. If omitted, all members of the array starting from start_index are transmitted. |

Return Value

The number of entities that have been transmitted.

Error Conditions

If the function fails, an error is generated.

2.4 PEG and MARK Commands

| Command | Description |
|-------------|---------------------------------------------------------------------------------------------------|
| ASSIGNMARK | Assigns MARK inputs pins to encoders. |
| ASSIGNPEG | Assigns PEG engines to encoders. |
| ASSIGNPOUTS | Assigns PEG outputs. |
| GETPEGCOUNT | Returns the pulse counter of the required PEG engine |
| PEG_I | Activates incremental PEG, where pulses are generated at predefined, fixed position intervals. |
| PEG_R | Activates random, table-based PEG where predefined pulses are generated at pre-defined positions. |
| STARTPEG | Starts PEG motion for specified axis. |
| STOPPEG | Terminates PEG motion. |

2.4.1 ASSIGNMARK

Description

The **ASSIGNMARK** function allows assignment of Mark inputs to encoder. It allows a mapping of encoder latching to be triggered by using different physical input pins.

Syntax

ASSIGNMARK[*/i*] *axis, mark_type, inputs_to_encoder_bit_code*



In newer products the *inputs_to_encoder_bit_code* parameter is actually a byte code. See [IDMxx/ECMxx/UDMsm/UDMsa/UDMma Encoder Mapping](#) table.

Arguments

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis | The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. The axis parameter determines which node unit is used. Axis parameter can be any axis number of the same unit. |
| mark_type | 1 for Mark-1 assignment 2 for Mark-2 assignment |

| | |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| inputs_to_encoder_bit_code | <p>Bit code for inputs-to-encoders mapping according to the following tables.</p> <p>The bit code determines which physical input pin leads to each encoder MARK latching.</p> <p>Mark-1 Inputs to Encoder mapping for</p> <p>Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD</p> <p>IDMxx/ECMxx/UDMsm/UDMsa/UDMma Encoder Mapping</p> <p>Mark-1 Inputs to Encoders Mapping for with SPiiPlus CMnt/UDMpm-x/UDMpc/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/CMhv/UDMhv</p> <p>Mark-2 Inputs to Encoder mapping for</p> <p>Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD</p> <p>Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt/UDMpm/UDMpc/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/CMhv/UDMhv</p> |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Comments

Latching of Encoder <index> means **IST**(<index>).#MARK=1, and the **MARK**(<index>) variable value stores the feedback position of encoder <index> (**FPOS**(<index>)).

The Bit Code shown in the tables affects all of the connectors in the row.

Supported products that are not listed in the tables include only the default case.

If the switch: /i is included, the MARK input signal is inverted.

In IDMxx/ECMxx products, the latching for all encoders happens simultaneously.

2.4.2 ASSIGNPEG

Description

The **ASSIGNPEG** function is used for engine-to-encoder assignment as well as for the additional digital outputs assignment for use as PEG pulse outputs. It allows mapping of PEG engines to be triggered by the feedback of a specific encoder.

Syntax

ASSIGNPEG[/f] *axis, engines_to_encoders_code, gp_out_assign_code*

Arguments

| | |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | <p>The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.</p> <p>Axis parameter can be any axis number of the same unit.</p> <p>The axis parameter determines the Servo Processor used.</p> |
| engines_to_encoder_s_ | <p>Bit code for engines-to-encoders mapping according to:</p> <p>SPiiPlusNT/DC-LT/HP/LD Processor 0</p> <p>SPiiPlusNT/DC-LT/HP/LD Processor 1</p> |

| | |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| code | SPiiPlusCMnt/UDMpm/UDMpc/CMba/CMhp/CMxa/UDMba/UDMxa/CMhv/UDMhv/UDMnt/UDMpa/UDMcb UDMlc/UDIlt/UDIhp/UDMmc/PDIcl NPMpm/NPMpc IDMsm, ECMsm, and UDMsm |
| gp_out_assign_code | <p>General purpose outputs assignment to use as PEG pulse outputs according to:</p> SPiiPlusNT/DC-LT/HP/LD SP 0 SPiiPlusNT/DC-LT/HP/LD SP 1 SPiiPlus CMnt/CMhv/UDMpm/UDMhv UDMnt/UDMpa/UDMcb IDMsm, ECMsm, UDMsm |



The **axis** parameter actually serves to determine the Servo Processor used.

Comments

- > **ASSIGNPEG** is a blocking command - the ACSPL+ program moves to the next line or command only after this command has been fully executed or an error is generated.
- > The **axis** parameter can be any of the axes controlled by the same servo processor, the result will be the same.
- > If the "/f" switch is included, fast loading of Random PEG arrays is activated. This feature allows definition of state-arrays with more than 1024-members by using Random PEG. The **PEG_R** command must be called with the "/d" switch.
- > The **Bit Code** shown in the Mapping PEG Engines to Encoders tables affects all of the connectors in the row.



HSSI devices (HSSI-IO16, HSSI-ED2, etc.) cannot be used for the same Servo Processor when fast loading of Random PEG arrays is activated.

SPRT and **SPINJECT** commands cannot be used for the same Servo Processor when fast loading of Random PEG arrays is activated.

Related ACSPL+ Commands

PEG_I, ASSIGNPOUTS, PEG_R, STARTPEG, STOPPEG

COM Library Methods

None

C Library Functions

acsc_AssignPegNT

2.4.3 ASSIGNPOUTS

Description

The **ASSIGNPOUTS** function is used for assigning PEG engine output signals to physical output pins. In addition, the function allows assigning Fast General Purpose output pins and mapping between **FGP_OUT** signals to the bits of the ACSPL+ **OUT(x)** variable, where **x** is the index that has been assigned to the controller in the network during System Configuration.

The assignments can be obtained by running **#SI** in the SPiiPlus MMI Application Studio **Communication Terminal**. For example, the following is a fragment from the response to this command:



```

Axes Assignment: 8,9,10,11
Inputs/Outputs Assignment:
  Digital inputs (IN): 1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7
  Digital outputs (OUT): 1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7

```

OUT is an integer array that can be used for reading or writing the current state of the General Purpose outputs - see *SPiiPlus ACSPL+ Command & Variable Reference Guide*.

Each PEG engine has 1 PEG pulse output and 4 state outputs for a total of 5 outputs per PEG engine and a total of 30 outputs for the whole PEG generator. The controller supports 10 physical output pins that can be assigned to the PEG generator. The user defines which 10 outputs (of the 30) of the PEG generator are assigned to the 10 available physical output pins. Some of the output pins are shared between the PEG and the HSSI.

The tables in Appendix [A.2](#) define how each of the 30 outputs of the 6 PEG engines can be routed to the 10 physical output pins - 4 PEG out signals, 3 PEG state signals, and 3 HSSI signals. Note that some of the signals cannot be routed to physical pins.



Bit Code: 111 is used for switching the physical output pins to Fast General Purpose Outputs, see [ASSIGNPOUTS](#).

Syntax

```
ASSIGNPOUTS axis, peg_output, bit_code
```

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. For controllers with firmware version 2.15 or higher, the axis parameter can be any axis number of the unit. |
| peg_output | The peg output number according to the Mapping of Engine Outputs to Physical Output tables below |

| | |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bit_code | Bit code for engine outputs to physical outputs mapping according to: SPiiPlusNT/DC-LT/HP/LD SP 0 SPiiPlusNT/DC-LT/HP/LD SP 1 CMnt/UDMpm/UDMpc/CMhv/UDMhv CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa (OUT0-4) CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa (OUT5-9) UDMnt/UDMpa/UDMcb UDMlc/UDMmc/UDIlt/UDIhp/PDIcl NPMpm/NPMpc IDMsm/UDMsm/ECMsm |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Comments

ASSIGNPOUTS is a blocking command in the sense that the ACSPL+ program moves to the next line or command only after this command has been fully executed or an error is generated.

A separate **ASSIGNPOUTS** command should be called for every GP output or PEG output.

Examples

The following examples illustrate using the **ASSIGNPOUTS** in order to use PEG outputs as GP outputs

Example 1:

```
ASSIGNPOUTS 0, 2, 0b1111
```

This defines the **Z_PEG** output as **FGP_OUT2** and maps it to the bit 18 of the ACSPL+ **OUT** variable (see [ASSIGNPOUTS](#)).

If you run, for example:

```
OUT (x) .18=0
```

Where **x** is the index assigned to the controller during System Configuration, **FGP_OUT2** output will be activated.

Then if you run:

```
OUT (x) .18=0
```

FGP_OUT2 will be deactivated.

Example 2:

```
ASSIGNPOUTS 4, 7, 0b1111
```

This defines the **X_STATE2** output as **FGP_OUT6** and maps it to the bit 22 of the ACSPL+ **OUT** variable (see [ASSIGNPOUTS](#)).

Related ACSPL+ Commands

[ASSIGNPEG](#), [PEG_I](#), [PEG_R](#), [STARTPEG](#), [STOPPEG](#)

COM Library Methods

None

C Library Functions

acsc_AssignPegOutputsNT

2.4.4 GETPEGCOUNT

Description

The function returns the pulse counter of the required PEG engine.

Syntax

```
GetPEGCount (PEG_engine)
```

2.4.5 PEG_I

Description

The **PEG_I** command is used for setting the parameters for the Incremental PEG mode.

Syntax

PEG_I [/awi] *peg_engine, width, first_point, interval, last_point[, time_based_pulses, time_based_period]*

Arguments

| | |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>peg_engine</i> | The PEG engine number |
| <i>width</i> | Width of the pulse in milliseconds. |
| <i>first_point</i> | A real scalar value in user units indicating the first point for the PEG generation. |
| <i>interval</i> | A real scalar value in user units indicating the distance between PEG events. |
| <i>last_point</i> | A real scalar value in user units indicating the last point for PEG generation. |
| <i>time_based_pulses</i> | Optional parameter - a real scalar value indicating the number of time-based pulses generated after each encoder-based pulse, the range is from 0 to 65,535. |
| <i>time_based_period</i> | Optional parameter, a real scalar in milliseconds - period of time-based pulses, the range is from 0.00005334 to 1.7476. The time-based period must be at least pulse width + 26.6667 nsec (minimum distance between two pulses). |



PEG is generated only after the first pre-defined start point is reached. If the current encoder position exceeds pre-defined start point no PEG pulses are fired. It is recommended to activate the PEG engine before the maximum current position for movement in the positive direction and the minimum current position for movement in the negative direction.

Comments

- > If the switch: /w is included, the execution of the command is delayed until the execution of the [STARTPEG](#) command.
- > If the switch: /i is included, the PEG pulse output signal is inverted.
- > If the switch: /a is included, error accumulation is prevented by taking into account the rounding of the distance between incremental PEG events. You must use this switch if *interval* does not match the whole number of encoder counts. Using this switch is recommended for any application that uses the **PEG_I** command, regardless if *interval* matches the whole number of encoder counts.
- > Valid numbers of the `peg_engine` parameter can be found in the #SI report. In case of multiple network units, the first axis number of each node indicates the first PEG engine of the node.

Example

See the Incremental PEG example in the *PEG and MARK Operations Application Note*.

Related ACSPL+ Commands

[PEG_R](#), [ASSIGNPEG](#), [ASSIGNPOUTS](#), [STARTPEG](#), [STOPPEG](#)

Related ACSPL+ Variables

AST

COM Library Methods

None

C Library Functions

`acsc_PegIncNT`, `acsc_WaitPegReady`

2.4.6 PEG_R

Description

The **PEG_R** command is used for setting the parameters for the Random PEG mode.

Syntax

PEG_R[/wid] *peg_engine*, *width*, *mode*, *first_index*, *last_index*, *POS_ARRAY*[, *STATE_ARRAY*, *time_based_pulses*, *time_based_period*

Arguments

| | |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>peg_engine</i> | The PEG engine number |
| <i>width</i> | Width of the pulse in milliseconds |
| <i>mode</i> | PEG state output signals configuration according to PEG_R . Bits 0-3 relates the PEG State 0 of the specific PEG engine Bits 4-7 relates the PEG State 1 of the specific PEG engine Bits 8-11 relates the PEG State 2 of the specific PEG engine Bits 12-15 relates the PEG State 3 of the specific PEG engine The most commonly used value is 0x4444 - PEG State Outputs 0-3 are configured with the 'State' option (bits 2, 6, 10, 14 are ON). |
| <i>first_index</i> | Index of first entry in the array for PEG generation |
| <i>last_index</i> | Index of last entry in the array for PEG generation |
| <i>POS_ARRAY</i> | The Random Event Positions array, maximum of 256/1024 members. If a longer array is required, use both PEG_R/d and ASSIGNPEG/f switches. |
| <i>STATE_ARRAY</i> | Optional parameter - the Outputs States array defining the four PEG output states, maximum of 256/1024 members. If a longer array is required, use both PEG_R/d and ASSIGNPEG/f switches. |
| <i>time-based-pulses</i> | Optional parameter - number of time-based pulses generated after each encoder-based pulse, the range is from 0 to 65,535. |
| <i>time-based-period</i> | Optional parameter - period of time-based pulses (milliseconds), the range is from 0.00005334 to 1.7476. Time-based period must be at least pulse width + 26.6667 nsec (minimum distance between two pulses). |

Table 2-6. PEG Output Signal Configuration

PEG state output types:

"Three state" - PEG state output is not in use

"State" - PEG state output will be changed according to the STATE_ARRAY values

"Pulse" - PEG state output will be changed according to PEG pulse value

"Pulse & State" - PEG state output will be changed according to the result of AND operation between STATE_ARRAY values AND PEG pulse value

Pulse Polarity:

If positive or negative pulse is used as PEG pulse value for the specific "PEG State Output"

State Polarity:

If positive or negative state is used as PEG pulse value for the specific "PEG State Output"

Comments

- > If the switch **/w** is included, the execution of the command is delayed until the execution of the **STARTPEG** command.
- > If the switch **/i** is specified, the PEG pulse output signal is inverted.
- > If the switch **/d** is specified, dynamic loading of positions is used. Dynamic loading can only be implemented on a PEG engine with fast loading (**ASSIGNPEG /f**). In this case, , it allows definition of a state-array which has more than 1024-members.
- > The parameters that can be set by the command differ from those that could be set for SPiiPlusCM/SPiiPlusSA/SPiiPlus3U controllers with the addition of the new *first_index* and *last_index* parameters.
- > When the PEG pulse is activated, the voltage between the two differential **PEG** outputs (+) and (-) drops to -5V. When the **PEG** pulse is de-activated, the voltage between the two differential **PEG** outputs is 5V.
- > In **PEG_R**, the number of position-based pulses is limited to eight pulses per controller cycle and the **minimum time interval must be >200nsec**.
- > When using a Sin-Cos encoder, **PEG** is triggered at the zero crossing of the sine-cosine waves and not at the precise interpolated position.
- > The last three arguments are optional. If **STATE_ARRAY** is omitted, the controller generates the PEG pulses at each position but does not change the state of any output. If time-based-pulses and time-based-period are omitted, the controller does not generate time based pulses.
- > The dynamic loading feature is limited by the loading frequency. If a high loading frequency is required, the loading capacity may not suffice to keep the FIFO loaded.
- > If the FIFO is emptied before all data arrays have been loaded, a memory overflow fault will be thrown.
- > Valid numbers of the `peg_engine` parameter can be found in the #SI report. In case of multiple network units, the first axis number of each node indicates the first peg engine of the node.

Loading Frequency Table

| CTIME (ms) | Frequency (Hz) |
|------------|----------------|
| 1 | 200 |
| 0.5 | 400 |
| 0.25 | 800 |
| 0.2 | 1000 |

Example

See the Random PEG example in the *PEG and MARK Operations Application Note*.

Related ACSPL+ Commands

[PEG_I](#), [ASSIGNPEG](#), [ASSIGNPOUTS](#), [STARTPEG](#), [STOPPEG](#)

Related ACSPL+ Variables

AST

COM Library Methods

None

C Library Functions

acsc_PegRandomNT, acsc_WaitPegReady

2.4.7 STARTPEG

Description

The **STARTPEG** command initiates the PEG process on the specified engine. The command is used in both the Incremental and Random PEG modes by using /w switch in **PEG_I** or **PEG_R** command. If this switch is included, the execution of the **PEG_I** and **PEG_R** commands is delayed until the execution of the **STARTPEG** command.

Syntax

STARTPEG *peg_engine*

Arguments

| | |
|-------------------|-----------------------|
| <i>peg_engine</i> | The PEG engine number |
|-------------------|-----------------------|

Comments

STARTPEG is a blocking command in the sense that the ACSPL+ program moves to the next line or command only after this command has been fully executed or an error is generated.

If **STOPPEG** has been issued before the last PEG position, you have to use **STARTPEG** to resume PEG engine firings from the current point.

Valid numbers of the *peg_engine* parameter can be found in the #SI report. In case of multiple network units, the first axis number of each node indicates the first peg engine of the node.

Example

```
PEG_I/W 0, 0.003, 1000, 1000, 3000, 2, 0.01
PTP 0, 3000      !The program initiates synchronous PEG, with PTP
!motion on axis 0.
WAIT 2          !Two milliseconds after motion starts, PEG
                !will be initiated by STARTPEG
STARTPEG 0
```

Related ACSPL+ Commands

[ASSIGNPEG](#), [ASSIGNPOUTS](#), [STOPPEG](#)

Related ACSPL+ Variables

AST

COM Library Methods

None

C Library Functions

acsc_StartPegNT

2.4.8 STOPPEG

Description

The **STOPPEG** command terminates the PEG process immediately on the specified engine. The command is used in both the Incremental and Random PEG modes.

Syntax

STOPPEG *peg_engine*

Arguments

| | |
|--------------------------|-----------------------|
| <i>peg_engine</i> | The PEG engine number |
|--------------------------|-----------------------|

Comments

STOPPEG is a blocking command in the sense that the ACSPL+ program moves to the next line or command only after this command has been fully executed or an error is generated.

Valid values of the *peg_engine* parameter can be found in the #SI report. In case of multiple network units, the first axis number of each node indicates the first peg engine of the node.

Example

```
PEG_I 0, 0.003, 1000, 1000, 3000, 2, 0.01
PTP 0, 3000           !The program initiates synchronous PEG, with PTP
                    !motion on axis 0.
WAIT 2              !Two milliseconds after motion starts, PEG
                    !will be terminated by STOPPEG.
STOPPEG 0
```

Related ACSPL+ Commands

[ASSIGNPEG](#), [ASSIGNPOUTS](#), [PEG_I](#), [PEG_R](#), [STARTPEG](#)

COM Library Methods

None

C Library Functions

acsc_StopPegNT

2.5 Miscellaneous Commands

The Miscellaneous commands are:

| Command | Description |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>AXISDEF</code> | Establishes an axis alias. |
| <code>DC</code> | Activates data collection. |
| <code>SPINJECT</code> | Returns the system back to the operational state if one or more slaves underwent a reset or power cycle. |
| <code>READ</code> | Reads an array from a file in the flash memory. |
| <code>SPDC</code> | Activates data collection from a Servo Processor variable. |
| <code>SPINJECT</code> | Initiates the transfer of MPU real-time data to the Servo Processor. |
| <code>STOPINJECT</code> | Stops the transfer of MPU real-time data to the Servo Processor. |
| <code>STOPSPDC</code> | Terminate SPDC data collection |
| <code>SPICFG</code> | Configure SPI Interface |
| <code>SPIWRITE</code> | Issues the SPI transaction with the number of SPI words to be sent and received in a single transaction |
| <code>SPRT</code> | Starts a real-time data transfer process from the MPU to a given Servo Processor. |
| <code>SPRTSTOP</code> | Stops an active real-time data transfer process on the given SP (for cyclic command only). |
| <code>STOPDC</code> | Terminates data collection. |
| <code>WRITE</code> | Writes an array to a file in the flash memory. |

2.5.1 *AXISDEF*

Description

The **AXISDEF** command enables the user to assign an alias to one or more axes. Once assigned, the user can use the alias throughout the program in any command requiring an axis argument.

Syntax

AXISDEF *axis_alias* = *axis*

Arguments

| | |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_alias</i> | <p>Any string with the following restrictions:</p> <ul style="list-style-type: none"> > Only one name can be defined for one axis, that is, different names cannot be used for the same axis. > The names must be unique, i.e., two axes cannot be defined with the same name. > An axis name must not be the same of any other variable name, label, keyword, etc. <p>A compilation error occurs if one of the above restrictions is not satisfied.</p> |
| <i>axis</i> | <p>The axis number, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.</p> <p>Adding (R) to the axis parameter means G-Code run on the axis will ignore the modality entry of G20.</p> |

Comments

The **AXISDEF** command can be repeated many times to define all required aliases.

The axis name must be defined in the D-Buffer. In any case, the axis definition has global scope (the definitions of the same axis in a different program must be identical as applies to all global variables).

Although postfix indexing can be used, it is recommended using explicit indexing and providing names as symbolic constants.

Related ACSPL+ Commands

None

COM Library Methods

None

C Library Functions

None

Example 1

An axis name can be used in expressions as a symbolic constant. For example, given the program includes the declaration:

```
AXISDEF Q=3
```

the following command

```
VEL (Q)=1000;
```

assigns 1000 to the required velocity of axis 3.

Example 2

As user variables, axis-related standard variables accept explicit indexing. However, axis-related standard variables also accept postfix indexing. For example, given a program that includes the declaration:

```
AXISDEF Q=3, X1=1, X2= 2
```

Example 3

Adding (R) to the axis parameter means G-Code run on the axis will ignore the modality entry of G20 in order to support axes driving a rotational motion with G-Code commands.

```
AXISDEF X=0, Y=1, Z=2, A=5 (R) , B = 6 (R) , C=7
```

In this example axis 5 and axis 6 will ignore the modality entry of G20.

| Explicit Indexing | Postfix Indexing |
|-----------------------|------------------|
| VEL(3) or VEL(Q) | VEL3 |
| ACC(1) or ACC(X1) | ACC1 |
| SLVKI(2) or SLVKI(X2) | SLVKI2 |

2.5.2 DC

Description

DC (Data Collection) accumulates data of any specified standard or user-defined variable with a constant sampling rate. **DC** synchronized with motion (see Command Option **/s**) is called Axis Data Collection. **DC** not synchronized with motion is called System Data Collection.

DC terminates due to:

- > STOPDC
- > The defined **DC** array is completed

Syntax (except for DC/s)

DC[/switch] *array_name, number of points, time-interval, variable_1, [variable_2...variable_N]*

Syntax for DC/s

DC/s *axis, global array, number of points, time-interval, variable_1, [variable_2...variable_N]*

Arguments

| | |
|-------------------------|------------------------------------------------|
| global array | The name of a global array that stores samples |
| number of points | Define the number of samples |
| time-interval | Define the time interval between each sample |

| | |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <i>variable_1</i> | Define variable/s to be sampled. The number of rows defined in <i>array_name</i> must match the number of variables to be sampled |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------|

Switches

/switch can be one of the following:

| | |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>/s</i> | Triggers DC with the execution of the next motion command following the call to DC/s. Motions queued before the call to DC/S will not be recorded. DC synchronized with motion is called Axis Data Collection . |
| <i>/w</i> | Create the synchronous data collection, but do not start until GO . Command option <i>/w</i> can only be used with the <i>/s</i> . |
| <i>/c</i> | Cyclic data collection |

Comments

1. **DC** can include up to 24 sampled variables.
2. Only one **DC** (system data collection) process can run at the same time.
3. Up to eight **DC/s** (axis data collection) processes can be simultaneously executed where each process fills a separate array.
4. **DC/c** does not self-terminate. **STOPDC** terminates cyclic data collection.
5. **DC/c** uses the collection array as a cyclic buffer and can continue to collect data indefinitely. When the array is full, each new sample overwrites the oldest sample in the array.
6. After the cyclic data collection concludes, the controller reorganizes the sample array so that the first element represents the oldest sample and the last element represents the most recent sample.
7. Variable **S_ST.#DC** = 1 when non-synchronized **DC** is active.
8. Variable **AST<axis>.#DC** = 1 when synchronized **DC** is active.

Related ACSPL+ Commands

[STOPDC](#), [SPDC](#)

Related ACSPL+ Variables

[S_ST](#), [AST](#), [S_DCN](#), [S_DCP](#), [DCN](#), [DCP](#)

COM Library Methods and .NET Library Methods

DataCollection, WaitCollectEnd, StopCollect

C Library Functions

acsc_DataCollectionExt, acsc_WaitCollectEnd, acsc_StopCollect

Example 1:

Cyclic Data Collection

```
GLOBAL REAL ARRAY (2) (1000)
```

```

!Define a real type array for two variables

!(rows) and 1000 sampling points (columns).
DC/C ARRAY,1000,3,FPOS0,TIME

!Start cyclic data collection and store 1000

!samples in ARRAY.

!The time between each sampling point is 3 msec.

!The FPOS0 standard variable samples are stored

!in the first row of ARRAY, and the TIME

!variable values are stored in the second row.
TILL ^S_ST.#DC
!Wait until S_ST.#DC = 0 (DC collection is

!finished).
STOP

```

Example 2

Motion Synchronized Data Collection

```

GLOBAL REAL SAMPLE_ARRAY(2)(1000)
                                !Define a real type array for two variables
                                !(rows) and 1000 sampling points (columns).
DC/S 0,SAMPLE_ARRAY,1000,3,FPOS0,TIME
                                !Start cyclic data collection when motion
                                !(synchronized on axis 0) begins, and store 1000
                                !samples in SAMPLE_ARRAY. The time
                                !between each sampling point is 3 msec.
                                !The FPOS0 standard variable samples are stored in
                                !the first row of SAMPLE_ARRAY, and the TIME
                                !variable values are stored in the second row.

TILL ^AST(0).#DC
                                !Wait until AST.#DC = 0 (DC collection is
                                !finished).

STOP

```

2.5.3 STOPDC

Description

Immediately terminates **DC** and **SPDC**.

Syntax

STOPDC[/*switch*]

Switch

/switch can be:

| | |
|-----------------|----------------------------------------|
| <code>/s</code> | Terminate synchronized data collection |
|-----------------|----------------------------------------|

Comments

- > **STOPDC** with an argument delays termination of **DC**.
- > **STOPDC/s** terminates synchronous **DC** initiated by **DC/s**.
- > Multiple axis specification is not allowed.

Related ACSPL+ Commands

[DC](#), [SPDC](#)

Related ACSPL+ Variables

[S_ST](#), [AST](#), [S_DCN](#), [S_DCP](#), [DCN](#), [DCP](#)

COM Library Methods and .NET Library Methods

DataCollection, StopCollect, WaitCollectEnd

C Library Functions

acsc_DataCollectionExt, acsc_StopCollect, acsc_WaitCollectEnd

Example 1

```
STOPDC 50      !Collect an additional 50 samples and then
               !terminate DC.
```

Example 2

```
STOPDC/S 1    !Stop synchronous axis data collection for axis 1
```

2.5.4 READ**Description**

Reads a file from the controller's nonvolatile (flash) memory to a user defined array or variable. The file must exist in the nonvolatile memory by previously writing it using the **WRITE** command.

Syntax

READ *array[,filename]*

or

READ/s *user-variable[, filename]*

Switch

| | |
|-----------------|---------------------------------------------------------------|
| <code>/s</code> | Specifies that the user variable is a scalar and not an array |
|-----------------|---------------------------------------------------------------|

Arguments

| | |
|----------------------|-------------------------------------------------------------------------|
| <i>array</i> | User defined array to which the data will be imported |
| <i>user-variable</i> | A scalar variable for use with the /s switch, can be either REAL or INT |
| <i>filename</i> | Optional non-volatile memory file name. |

Comments

1. The **filename** must not include a file name extension.
2. The filename maximum length is 100 chars.
3. The **user-array** name must be declared in the buffer where the command is executed.
4. The **variable** name may be declared in the buffer where the command is executed, or it may be declared in the D-buffer.
5. If **READ** is executed from the **Communication Terminal** as a command, **array** must specify the name of a global array.
6. If **READ/s** is executed from the **Communication Terminal** as a command, **variable** must specify the name of a global variable.
7. If the optional file name is not supplied, the variable name will be used as the file name.

The following error is supported:

- > Error 3333 "File name MAX length is 100 chars"

Related ACSPL+ Commands

WRITE

COM Library Methods

Transaction

C Library Functions

acsc_Transaction

Examples

```
GLOBAL REAL ARRAY(5)      !Define a real global array
READ ARRAY,FILENAME      !Read the defined file to the array
```

```
GLOBAL INT VAR           !Define an integer global variable
READ/s VAR,FILENAME     !Reads from file to the variable(scalar)
STOP
```

2.5.5 SPDC

Description

SPDC (Servo Processor Data Collection) performs fast data collection and accumulates data about the specified Servo Processor variable with a constant maximum sampling rate of 20kHz. A typical

use for **SPDC** is for collecting position error (**PE**) and feedback position (**FPOS**) data at the fast Servo Processor rate.

The Servo Processor value is different from the MPU value. The Servo Processor always uses counts and not units. The Servo Processor position value is not affected by a **SET FPOS** command. An offset is added at the MPU level only. The formula (that you can find in our manuals) is:

$$FPOS = FP * EFAC + EOFFS$$

where **FPOS** is the MPU variable and **FP** is the Servo Processor calculated value.

SPDC terminates due to:

- > **STOPDC**
- > After accumulating the defined number of samples

Syntax

SPDC[/r] *Array, number_of_samples, sampling_period, SP_number, SP_Address1, [SP_Address2]*

Arguments

| | |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Array | Array name, up to XARRSIZE variable value. By default, Array is assumed to be an integer array, if the /r switch is added, it defines the array as real. |
| number_of_samples | The number of samples to collect, the maximum value depends on the size of the array. |
| sampling_period | The time, in milliseconds, that each sample is taken. |
| SP_number | The number of the Servo Processor to be sampled |
| SP_Address1 | The address of the Servo Processor variable in the Servo Processor to sample. |
| SP_Address2 | As an option, you can add another address of an other Servo Processor variable in the Servo Processor to sample. In this case, the array should be defined as (2)(N) |
| SP_Address3 | As an option, you can add another address of an other Servo Processor variable in the Servo Processor to sample, In this case, the array should be defined as (3)(N). |
| SP_Address4 | As an option, you can add another address of an other Servo Processor variable in the Servo Processor to sample. In this case, the array should be defined as (4)(N) |



Since memory addresses may vary between SPiiPlus products and revisions, it is highly recommended to define a variable to represent **SP_Address** as the return value of **GETSPA**. **SPDC** can then use this variable in any SPiiPlus product or revision.

Related ACSPL+ Commands

STOPDC

Comments

Only one **SPDC** command per Servo Processor can run at the same time.

Table 2-7. Commonly Monitored SPDC Variables

| Variable | Axis | Servo Processor Variable |
|-------------------|---------|--------------------------|
| Position Error | 0,1,2,3 | axes[0].PE |
| Feedback Position | 0,1,2,3 | axes[0].fpos |
| Feedback Velocity | 0,1,2,3 | axes[0].fvel |
| Sin Analog Input | 0,1,2,3 | axes[0].sin |
| Cos Analog Input | 0,1,2,3 | axes[0].cos |
| Phase A Current | 0,1,2,3 | axes[0].is |
| Phase B Current | 0,1,2,3 | axes[0].it |
| Current Command | 0,1,2,3 | axes[0].command |

```
! ----- Declare data arrays -----
GLOBAL INT DATA(15000)
!Declare global array of integer of size 15000
REAL PAR_ADDRESS
PAR_ADDRESS=GETSPA(0,"axes[0].PE")
! ----- Define motion parameters-----
VEL(0) = 5000
ACC(0) = 100000
DEC(0) = 100000
JERK(0) =2e6
!-----Run motion and do fast data collection -----
SET FPOS(0) = 0
ENABLE 0
SPDC DATA,15000,0.05,0,PAR_ADDRESS
PTP/e 0, 1000
PTP/e 0, -1000
PTP/e 0, 1000
!Use the following if you need to convert the data to one column to
```

```
!export to Excel (otherwise you can collect 30000 points by SPDC above)
!Convert the array data from one row to one column to fit to export to
Excel.
!INT DATA1(15000) (1)
!INT J; J=0
!LOOP 15000;DATA1(J) (0)=DATA(J);J=J+1;END
STOP
```

Since memory addresses may vary between SPiiPlus products and revisions, it is highly recommended to define a variable to represent **SP_Address** as the return value of **GETSPA**. **SPDC** can then use this variable in any SPiiPlus product or revision.

SP_number may be set to 2 at most for most ACS products. The following products support sampling of up to 4 variables:

- > IMDsm
- > ECMsm
- > UDMsm
- > IDMsA
- > ECMsa
- > UDMsa

Related ACSPL+ Commands

STOPDC

2.5.6 STOPSPDC

Description

The **STOPSPDC** function Immediately terminates the data collection of **SPDC** (Servo processor data collection) for the specified servo processor.

Syntax

```
STOPSPDC SP_number
```

Arguments

SP_Number

The number of the Servo Processor.

Return Value

None

Comments

The following errors are supported:

- > 3034 - "Illegal index value"

NST.#SPDC bit is set to off (for the relevant server processor)

This variable is supported in version 3.10 and higher.

Related ACSPL+ Commands

SPDC

Related ACSPL+ Variables**NST.#SPDC****Example**

```
STOPSPDC 1 !Stop Data Collection for Servo Processor 1
```

2.5.7 WRITE**Description**

Writes an array or scalar (any system or user-defined variable) to a file in the controller's nonvolatile (flash) memory.

Syntax

WRITE *user-array*[, *filename*]

WRITE/s *user-array*[, *filename*]

Switch

| | |
|----|---------------------------------------------------------------|
| /s | Specifies that the user variable is a scalar and not an array |
|----|---------------------------------------------------------------|

Arguments

| | |
|----------------------|---------------------------------------------------------|
| <i>user-array</i> | User defined array from which the data will be imported |
| <i>user-variable</i> | User defined scalar variable, can be either REAL or INT |
| <i>filename</i> | Optional non-volatile memory file name. |

Comments

1. The nonvolatile memory **filename** must not include an extension.
2. The **user-array** or **user-variable** name must be declared in the buffer where the command is executed or it may be declared in a D buffer.
3. If **WRITE** is executed from the **Communication Terminal** as a command, **user-array** must specify the name of a global array.
4. If **WRITE/s** is executed from the **Communication Terminal** as a command, **user-variable** must specify the name of a global scalar variable.
5. If **filename** does not exist, it is created. If the file already exists, it is overwritten.
6. If the optional file name is not supplied, the variable name will be used as the file name.

The following error is supported:

- > Error 3333 "File name MAX length is 100 chars"

Related ACSPL+ Commands**READ****COM Library Methods**

Transaction

C Library Functions

acsc_Transaction

2.5.8 SPINJECT

Description

SPINJECT initiates the transfer of MPU real-time data to the Servo Processor.

Syntax

SPINJECT([/switch] *Array*,*Nsamples*,*Node*,*Addr1*,[*Addr2*])

Arguments

| | |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Array | Data source: 1 or 2 dimensional ACSPL+ array (real or integer). |
| Nsamples | Number of samples from the source Array to inject. If the process is not cyclic, the injection will stop after this number of samples. The last telegram will be padded by the last element if needed. |
| Node | The number of the EtherCAT node as in Servo Processor data collection. |
| SP_Address1 | Address inside the Servo Processor , it must correspond to a floating or integer variable in Servo Processor program. |
| SP_Address2 | Address inside the Servo Processor , it must correspond to a floating or integer variable in Servo Processor program. Used only if Array is 2 dimensional. |
| SP_Address3 | Address inside the Servo Processor , it must correspond to a floating or integer variable in Servo Processor program. Used only if Array is 3 dimensional. |
| SP_Address4 | Address inside the Servo Processor , it must correspond to a floating or integer variable in Servo Processor program. Used only if Array is 4 dimensional. |

Switches

/switch can be one of the following:

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| /c | Cyclic: For each MPU cycle the FW fills CTIME*20 values from the source Array . Once the end is reached, the process continues from the start. |
| /r | Designates a real source. |

Examples

```
!MYDCOM is RT control of DCOM for axis 0 inside Servo Processor 0
REAL MYDCOM(20) ! 20 values are used for CTIME = 1.0
```

```
! Initialize MYDCOM here to the desired values
! ...
SPINJECT/CR MYDCOM,20,0,getspa(0, "axes[0].direct_command")
```

2.5.9 STOPINJECT

Description

STOPINJECT stops active injection process on the given Servo Processor.

Syntax

STOPINJECT *Servo_Processor*

Arguments

*Servo_
Processor*

Identifies the Servo Processor upon which the injection process is operating.

Example

```
STOPINJECT 1
!Stops the injection process on Servo Processor1
```

2.5.10 SPICFG

Description

SPICFG configures and initializes the SPI interface.

Syntax

SPICFG (SlaveIndex, Mode, NumberOfWords, Polarity,Size,Frequency)

Arguments

| | |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SlaveIndex | Index of the EtherCAT slave in the EtherCAT network, or 0 in case of IDMsM/ECMsM/UDMsM |
| Mode | The mode of the SPI interface. The following modes are supported: <ul style="list-style-type: none"> > 0 - Slave > 1 - Master > 2 - SlaveListenOnly > 3 - Disable > 4 - Master Single Transaction (Used by ACSPL+ SPIWRITE) |
| NumberOfWords | Number of SPI Data Words used by the application (FW to SPI). Range: {0,8} Not relevant in case of Master Single Transaction Mode |
| Polarity | Clock Polarity. Four types are available: |

| | |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> > Rising Edge – 0 > Rising Edge with Delay – 1 > Falling Edge – 2 > Falling Edge with Delay - 3 |
| Size | Data size in bits, the range is {1-16}. |
| Frequency | An integer number which defines the frequency. The range is 200KHz-10MHz, limited values are supported (defined in the table below). |

Frequency Values Supported

| Value | Frequency (kHz) |
|-------|-----------------|
| 2 | 800 |
| 3 | 1000 |
| 4 | 1500 |
| 5 | 2000 |
| 6 | 2500 |
| 7 | 3000 |
| 8 | 3500 |
| 9 | 4000 |
| 10 | 5000 |

Return Value

None

Comments

When the SPI interface is not required anymore, SPICFG should be called with Mode=3 (disable) parameter.

Example



2.5.10.1 SPIWRITE

2.5.11 SPIWRITE

Description

SPIWRITE is a function that issues the SPI transaction with the number of SPI words to be sent and received in a single transaction.

The function may be used in two modes: Slave and Single Master Transaction

Syntax

```
int SPIWRITE( SlaveIndex, NumberOfWords,SPIDataWrite,SPIDataRead,TimeOut )
```

Arguments

| | |
|---------------|---------------------------------------------------------------------------------------------------------------------------|
| SlaveIndex | Index of the EtherCAT slave in the EtherCAT network, or 0 in case of IDMsm/ECMsm |
| NumberOfWords | Number of SPI Data Words to be sent/received in the single transaction. Range: {0,8} |
| SPIDataWrite | INT array. Size must be number of words. |
| SPIDataRead | INT array. Size must be number of words. |
| Timeout | (Optional) integer, specifies the timeout in milliseconds. Relevant only in case of slave mode. Default value is 5000. |

Return Value

STATUS value, OK (0) or error.

Comments

Master mode behavior:

The data in the **SPIDataWrite** array written to the SPI interface.

The **SPIDataRead** array contains the reply data.

The function blocks until the reply is ready (when number of received words is equal to the **NumberOfWords** parameter.

Slave mode behavior:

The data in the **SPIDataWrite** array is written to the **EXTOUT** variable (and copied to the SPI interface). The **SPIDataRead** array contains the reply data. The function will wait until one of the following conditions is true:

1. SPIRXN equals to NumberOfWords parameter
2. Timeout is reached
3. Error state is returned The function returns

Example

```
int SPIDataWrite(8)
int SPIDataRead(8)

int i=0

SPICFG(0,4,8,0,16,6) !Master Single Transaction Mode

loop 8
    SPIDataWrite(i)=i+1
    i=i+1
end
SPIWRITE(0,8,SPIDataWrite,SPIDataRead)
STOP
```

2.5.12 SPRT

Description

This function starts a real-time data transfer process from the MPU to a given Servo Processor.

Syntax

SPRT[/c] SP, Value_Array, Addr_Array

Switches

| | |
|----|-------------------------------------------------------------------------------|
| /c | Cyclic. For each MPU cycle, the firmware fills values from the source arrays. |
|----|-------------------------------------------------------------------------------|

Arguments

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SP | Number of the Servo Processor to be used for real-time data transfer. |
| Value_Array | Data source, 1-dimensional ACSPL+ real array <ul style="list-style-type: none"> > up to 20 values for CTIME >= 0.50 > up to 12 values for 0.20 <= CTIME < 0.50 |
| Addr_Array | Array of addresses inside that Servo Processor. It must correspond to the float or integer variable in the Servo Processor's program, as well as to the Value_Array values order and size. |

Comments



The **SPRT** command cannot be used in parallel with the SPINJECT command for the same Servo Processor.

The **SPRT** command can be used for simultaneous and deterministic update of 12-20 Servo Processor variables at the controller cycle rate.

It is superior to the **SETSP** command that can only update one Servo Processor variable in each controller cycle and cannot be used for continuous update (every controller cycle).

For example, assume that the PIV gains (**SLPKP**, **SLVKP**, **SLVKI**) need to be updated simultaneously and frequently for gain scheduling. Even if the variables are set in the same program line, or with a block command, the controller still updates one Servo Processor variable every controller cycle. Each of the parameters **SLVKP**, **SLPKP**, **SLVKI** has three values according to the motion phase (0=motion, 1=idle, 2= settling) and the corresponding idle and settling gains.

If this is not needed, you could simply set the three values equal for each parameter. The update is completed within several controller cycles, that can influence the system performance.

However, using **SPRT**, the internal Servo Processor variables can be updated simultaneously within one cycle.

Note that **SPRT** affects only Servo Processor variables i.e. corresponding ACSPL+ variables don't change.

Example 1

```
GLOBAL REAL Value(9)
INT Address(9)
INT Axis
GLOBAL REAL SLPKP_value, SLVKP_value, SLVKI_value

Axis = 0
! Finding the relevant addresses can be done as one time operation
! (No need to re-use GETSPA prior to each update).

% Address of the Servo Processor SLVKP parameter used during motion
Address(0) = getspa(0,"axes[0].params[0].SLVKP")
% Address of the Servo Processor SLVKP parameter used in idle state
Address(1) = GETSPA(0,"axes[0].params[1].SLVKP")
% Address of the Servo Processor SLVKP parameter used during settling
Address(2) = GETSPA(0,"axes[0].params[2].SLVKP")
Address(3) = GETSPA(0,"axes[0].params[0].SLPKP")
Address(4) = GETSPA(0,"axes[0].params[1].SLPKP")
Address(5) = GETSPA(0,"axes[0].params[2].SLPKP")
Address(6) = GETSPA(0,"axes[0].params[0].SLVKI")
Address(7) = GETSPA(0,"axes[0].params[1].SLVKI")
Address(8) = GETSPA(0,"axes[0].params[2].SLVKI")

BLOCK
! SLPKP=SLPK_value, SLVKP=SLVKP_value, SLVKI=SLVKI_value must be set
! simultaneously.
! The following lines calculate the corresponding dsp variables:

! Desired ACSPL parameters:
! Desired SLVKP value
SLVKP_value = 100
! Desired SLPKP value
SLVKP_value = 50
! Desired SLVKI value
```

```

SLVKP_value = 10

! Translate the ACSPL+ variables to the low level Servo Processor
! variables and store them in an array
Value(0) = SLVKP_value /1024/32766*20000*SAT(POW(2,21)/XVEL(Axis)*EFAC
(Axis),0,1)
Value(1) = Value(0)
Value(2) = Value(0)
Value(3) = SLPKP_value /20000
Value(4)= Value(3)
Value(5) = Value(3)
Value(6) = SLVKI_value /POW(2,16)
Value(7)= Value(6)
Value(8) = Value(6)

! The following command performs the update:
SPRTSTOP 0;
SPRT 0, Value, Address
end
Stop

```

Example 2

```

! Real-Time MPU-Servo Processor Data Transfer (Cyclic)
REAL Value(2)
INT Address(2)
INT SP;
INT i

SP = 0;
i = 0;
Value(0) = 0; Value(1) = 19;
Address(0) = GETSPA(SP, "dummy_double[1]");
Address(1) = GETSPA(SP, "dummy_double[2]");

SPRTSTOP SP ! For making sure there is no previously running SPRT
commands for the same SP
SPRT/C SP, Value, Address

while 1
    BLOCK
        Value(0) = i
        Value(1) = (20 - i)
        i = i + 1
        IF (i = 20)
            i = 0
        END
    END
END
STOP

```

Example 3

```

! Real-Time MPU-Servo Processor Data Transfer
REAL Value(2)
INT Address(2)
INT SP;
INT i
SP = 0;
i = 0;
Value(0) = 0; Value(1) = 19;
Address(0) = GETSPA(SP, "dummy_double[1]");
Address(1) = GETSPA(SP, "dummy_double[2]");
SPRTSTOP SP ! For making sure there is no previously running SPRT
commands for the same SP

while 1
    BLOCK
        Value(0) = i
        Value(1) = (20 - i)
        i = i + 1
        IF (i = 20)
            i = 0
        END
        SPRT SP, Value, Address
    END
END
STOP

```

2.5.13 SPRTSTOP

Description

SPRTSTOP stops an active real-time data transfer process on the given SP (for cyclic command only).

Syntax

SPRTSTOP SP

Arguments

| | |
|-----------|-------------------------------------------------------|
| SP | Number of the EtherCAT node as in SP data collection. |
|-----------|-------------------------------------------------------|

2.6 Motion Commands

The Motion commands are:

| Command | Description |
|----------------------|------------------------------------------------------------|
| ARC1 | Adds an arc segment to MSEG...ENDS motion. |
| ARC1 | Adds an arc segment to XSEG...ENDS motion. |

| Command | Description |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ARC1 | Adds an arc segment to BSEG...ENDS motion |
| ARC2 | Adds an arc segment to MSEG...ENDS motion. |
| ARC2 | Adds an arc segment to BSEG...ENDS motion |
| ARC2 | Adds an arc segment to XSEG...ENDS motion |
| BPTP | Creates a motion boost profile |
| BSEG...ENDS | Creates a blended segmented motion. |
| JOG | Creates a jog motion. |
| LINE | Adds a linear segment to MSEG...ENDS motion. |
| LINE | Adds a linear segment to XSEG...ENDS motion. |
| LINE | Adds an liners segment to BSEG...ENDS motion |
| MASTER | Defines a formula for calculating MPOS . |
| MPOINT | Adds a set of points to MPTP...ENDS , PATH...ENDS or PVSPLINE...ENDS motion. |
| MPTP...ENDS | Creates a multipoint motion. |
| MSEG...ENDS | Creates a segmented motion. |
| PATH...ENDS | Creates an arbitrary path motion with linear interpolation between the specified points. |
| POINT | Adds a point to MPTP...ENDS , PATH...ENDS , or PVSPLINE...ENDS motion. |
| PROJECTION | An expansion command to the MSEG...ENDS set of commands, that allows the controller to perform a three dimensional segmented motion such as creating arcs and lines on a user-defined plane. |
| PTP | Creates a point-to-point motion. |
| PVSPLINE...ENDS | Creates an arbitrary path motion with spline interpolation between the specified points. |
| SLAVE | Creates a master-slave motion. |

| Command | Description |
|-------------|------------------------------------------------------------|
| STOPPER | Adds a segment separator to MSEG...ENDS motion. |
| TRACK | Creates tracking motion. |
| XSEG...ENDS | Creates an extended segment motion. |
| NURBS | Creates NURBS motion |
| NPOINT | Creates NURBS motion segment |
| SPATH | Creates SmoothPath motion |
| SEGMENT | Creates SmoothPath motion segment |
| SMOVE | Define segment of movement with transition point smoothing |



For systems having more than 15 axes, avoid using motion commands to start the motion of all axes simultaneously as this may cause Over Usage or Servo Processor Alarm faults

2.6.1 ARC1

Description

ARC1 must be initialized with **MSEG...ENDS**. Use **ARC1** to specify the center point and final point coordinates of an arc and the direction of rotation. Direction is designated by a plus sign (+) or (-) for clockwise or counterclockwise rotation depending on the encoders' connections.

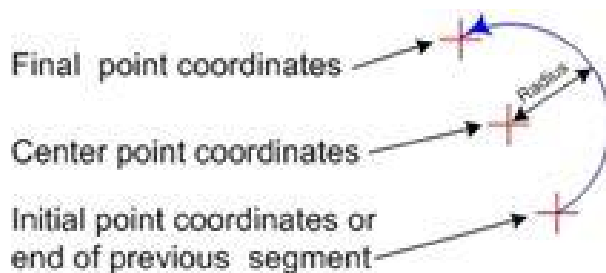



Figure 2-5. ARC1 Coordinate Specification

Syntax

ARC1 *axis_list, center-point, final-point, direction[,user-specified velocity]*

Arguments

| | |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | <p>List of axes involved, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.</p> <p>The ARC1 axis_list can involve two or more axes, see PROJECTION.</p> <div style="border: 1px solid black; border-radius: 10px; padding: 10px; margin-top: 10px;">  <p>A minimum of two axes must be specified.</p> </div> |
| <i>center-point</i> | Center point coordinates. |
| <i>final point</i> | Last point. |
| <i>direction</i> | Use + for motion in the direction of increasing encoder counts, or - for motion in the direction decreasing encoder counts. |
| <i>user-specified velocity</i> | If MSEG command option /v is used, the user-specified velocity must be the last parameter in the ARC1 syntax. |

Comments

- > **ARC1** and **ARC2** differ only by the required arguments. **ARC1** requires the coordinates of the center point, final point, and the direction of rotation. **ARC2** requires the coordinates of the center point and the rotation angle in radians. Each command produces the same result, so selection of either **ARC1** or **ARC2** depends on the available data.
- > A single **ARC1** command can not create a complete circle because the start point and end point of the motion can not be the same. Use two **ARC1** commands, or use **ARC2**.

Related ACSPL+ Commands

[MSEG...ENDS](#), [ARC2](#), [LINE](#), [STOPPER](#), [PROJECTION](#)

COM Library Methods

Arc1, ExtArc1

C Library Functions

acsc_Arc1, acsc_ExtArc1

Example

See [MSEG...ENDS](#).

2.6.2 ARC1

Description

Use **ARC1** to specify the center point and final point coordinates of an arc and the direction of rotation. Direction is designated by a plus sign (+) or (-) for clockwise or counterclockwise rotation depending on the encoders' connections. When **ARC1** is used for Extended Motion, it must be initialized with **XSEG...ENDS**.

Syntax

ARC1 [/switches] (axis_list), center_point_axis1, center_point_axis2, destination_point_axis1, destination_point_axis2, [destination_point_axis3, ... destination_point_axis6,] direction[, velocity] [,end_velocity][,time][,values, variables[,index [,masks]]][, lci_segment_active]

Arguments

| Argument | Description |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_list | Defines one or two axes, specified as axes numbers separated by commas or as axes names separated by commas. The axes should only be those axes specified in the corresponding XSEG command. |
| center_point_axis1 | Center point position for the first axis |
| center_point_axis2 | Center point position for the second axis |
| destination_point_axis1 | Destination position of the first axis |
| destination_point_axis2 | Destination position of the second axis |
| destination_point_axis3 ... destination_point_axis6 | Mandatory only if AXIS_LIST contains more than 2 axes. Destination position of the rest of axes. Number of destination positions must correspond to the number of axes in the AXIS_LIST . |
| direction | Direction is specified as + or -. It defines clockwise or counterclockwise rotation depending on the encoder connection: "+" for motion in the direction of increasing encoder counts, or "-" for motion in the direction decreasing encoder counts. |
| velocity | [Mandatory with /V switch]. Defines required velocity for the current and for all subsequent segments. See Using ARC1, ARC2 and LINE Switches . |
| end_velocity | [Mandatory with /F switch]. Defines required velocity at the end of the current segment. See Using ARC1, ARC2 and LINE Switches . |
| time | [Mandatory with /T switch]. |

| Argument | Description |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Defines segment processing time. See Using ARC1, ARC2 and LINE Switches . |
| values | [Mandatory with /O switch]. Defines the values to be written to variables array at the beginning of the current segment execution. values is a one-dimensional user defined array of integer or real type with maximum size of 10 elements . |
| variables | [Mandatory with /O switch]. Defines the user-defined array, which will be written with values data at the beginning of the current segment execution. variables is a one-dimensional user defined array of the same type and size as the values array. |
| index | [Optional, only used with /O switch] Defines the first element (starting from zero) of the variables array, to which values data will be written. If argument is omitted, values data is written to the variables array starting from the first element (index 0). |
| masks | [Optional, only used if values and variables are integer] Defines the masks that are applied to values before the values are written to variables array at the beginning of the current segment execution. masks is a one-dimensional user-defined array of integer type and the same size as the values array. The masks are only applied for integer values: $\text{variables}(n) = (\text{variables}(n) \text{ AND } (\text{NOT mask}(n))) \text{ OR } (\text{values}(n) \text{ AND } \text{mask}(n))$ If argument is omitted, all values bits are written to variables . If values is a real array, the masks argument should be omitted. |
| lci_segment_active | [Mandatory with /p switch] Integer value. Fire LCI State or Pulse at the beginning of current segment. The function is available if the LCI segment-based mode was previously defined by the SegmentGate or SegmentPulse functions. The value defines the state value in SegmentGate mode (1 or 0). In SegmentPulse mode the value equal 1. |



For information on optional switches for this command, see **Using ARC1, ARC2 and LINE Switches**.

2.6.3 ARC1

This format of ARC1 is used for blended segment motion and in this form must be initialized with BSEG...ENDS. The command adds to the motion path an arc segment that starts in the current point and ends in the destination point with the specified center point.

Syntax

```
ARC1[/switches] (axis_list),
    center_point_axis1,center_point_axis2,
    destination_point_axis1,destination_point_axis2, direction
    [,segment_time [,acceleration_time [,jerk_time [,dwell_time]]]]
```

Arguments

| Argument | Comments |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Defines one or two axes, specified as axes numbers separated by comma or as axes names separated by comma. The axes should only be from those axes specified in the corresponding BSEG command. |
| <i>center_point_axis1</i> | Center point position for the first axis |
| <i>center_point_axis2</i> | Center point position for the second axis |
| <i>destination_point_axis1</i> | Destination position of the first axis |
| <i>destination_point_axis2</i> | Destination position of the second axis |
| <i>direction</i> | Direction is specified as + or -. It defines clockwise or counterclockwise rotation depending on the encoder connection: "+" for motion in the direction of increasing encoder counts, or "-" for motion in the direction decreasing encoder counts. |
| <i>segment_time</i> | Only if switch/ m is specified: Segment time (T_m) in milliseconds. |
| <i>acceleration_time</i> | Only if switch/ a is specified: Acceleration time (T_a) in milliseconds. |
| <i>jerk_time</i> | Only if sufswitchfix /s is specified: Jerk time (T_j) in milliseconds |
| <i>dwell_time</i> | Only if sufswitchfix /d is specified: Dwell time at the final point of the segment in milliseconds. With this switch no blending will be done at the segment final point. |

2.6.4 ARC2

Description

ARC2 must be initialized with **MSEG...ENDS**. Use **ARC2** to specify the center point and rotation angle in radians of an arc segment. Designate direction by positive or negative rotation angle, depending on the encoders' connections.

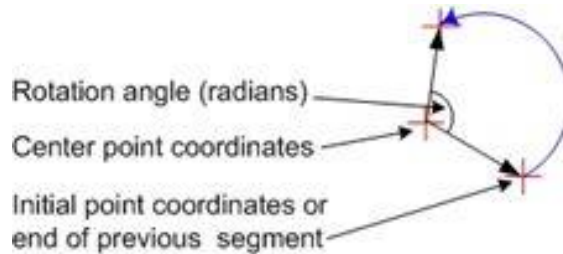



Figure 2-6. ARC2 Center Point and Rotation Angle Specification

Syntax

ARC2 *axis_list*, *center-point*, *rotation-angle and direction* [*user-specified velocity*]

Arguments

| | |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | <p>List of axes involved, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.</p> <p>The ARC2 <i>axis_list</i> can involve two or more axes, see PROJECTION.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>A minimum of two axes must be specified.</p> </div> |
| <i>center-point</i> | center point coordinates |
| <i>final point</i> | last point |
| <i>rotation angle and direction</i> | Rotation is in radians. Use + for motion in the direction of increasing encoder counts, or - for motion in the direction decreasing encoder counts |
| <i>user-specified velocity</i> | If MSEG command option /v is used, the user-specified velocity must be the last parameter in the ARC2 syntax. |

Comments

ARC1 and **ARC2** differ only by the required arguments. **ARC1** requires the coordinates of the center point, final point, and the direction of rotation. **ARC2** requires the coordinates of the center point and the rotation angle. Each command produces the same result, so selection of either **ARC1** or **ARC2** depends on the available data.

See **Using ARC1, ARC2 and LINE Switches** for details about function switches.

Related ACSPL+ Commands

MSEG...ENDS, ARC1, LINE, STOPPER, PROJECTION

COM Library Methods

Arc2, ExtArc2

C Library Functions

acsc_Arc2, acsc_ExtArc2

Example

See MSEG...ENDS.

2.6.5 ARC2


Description

This format of **ARC2** is used for extended segment motion and in this form must be initialized with **XSEG...ENDS**. Use **ARC2** to specify the center point and rotation angle in radians of an arc segment. Designate direction by positive or negative rotation angle, depending on the encoders' connections.

Syntax

ARC2[/switches] (axis_list), center_point_axis1,center_point_axis2, rotation_angle, [,destination_point_axis3, ... destination_point_axis6][,velocity][,end_velocity][,time][,values, variables[,index[,masks]]] [,external_loop_type, external_loop_type, maximum_allowed_deviation][, lci_segment_active]

Arguments

| | |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | <p>List of axes numbers separated by comma or as axes names separated by comma. The axes should only be those axes specified in the corresponding XSEG command.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>A minimum of two axes must be specified.</p> </div> |
| <i>center_point_axis1</i> | Center point position for the first axis |
| <i>center_point_axis2</i> | Center point position for the second axis |
| <i>destination_point_axis3</i> ... <i>destination_point_axis6</i> | <p>Mandatory only if axis_list contains more than 2 axes. Destination position of the rest of axes. Number of destination positions must correspond to the number of axes in the axis_list.</p> |

| | |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>rotation_angle</i> | Defines central angle of the arc, signed according to rotation direction: plus for a counter-clock-wise arc, minus for a clock-wise arc. |
| <i>velocity</i> | [Optional, only used with <i>/v</i> switch] Defines required velocity for the current and for all subsequent segments. See Switches explanation for details. |
| <i>end_velocity</i> | [Optional, only used with <i>/f</i> switch] Defines required velocity at the end of the current segment. See Switches explanation for details. |
| <i>time</i> | [Mandatory with <i>/t</i> switch]. Defines segment processing time. |
| <i>values</i> | [Optional, only used with <i>/o</i> switch] Defines the values to be written to variables array at the beginning of the current segment execution. <i>values</i> is a one-dimensional user defined array of integer or real type with maximum size of 10 elements . |
| <i>variables</i> | [Optional, only used with <i>/o</i> switch] Defines the user-defined array, which will be written with values data at the beginning of the current segment execution. <i>variables</i> is a one-dimensional user defined array of the same type and size as the <i>values</i> array. |
| <i>index</i> | [Optional, only used with <i>/o</i> switch] Defines the first element (starting from zero) of the <i>variables</i> array, to which values data will be written. If argument is omitted, <i>values</i> data is written to the <i>variables</i> array starting from the first element (index 0). |
| <i>masks</i> | [Optional, only used if <i>values</i> and <i>variables</i> are integer] Defines the masks that are applied to <i>values</i> before the values are written to <i>variables</i> array at the beginning of the current segment execution. <i>masks</i> is a one-dimensional user-defined array of integer type and the same size as the <i>values</i> array. The masks are only applied for integer values: $\text{variables}(n) = (\text{variables}(n) \text{ AND } (\text{NOT } \text{mask}(n))) \text{ OR } (\text{values}(n) \text{ AND } \text{mask}(n))$ If argument is omitted, all values bits are written to <i>variables</i> . If <i>values</i> is a real array, the masks argument should be omitted. |

| | |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>external_loop_type</i> | 0 - Cancel external loop 1 - Smooth External loop (line-arc-line) 2 - Triangle External loop (line-line-line) |
| <i>minimum_segment_length</i> | If the lengths of both segments are more than this value, the skywriting algorithm will be applied. |
| <i>maximum_allowed_deviation</i> | The parameter limits the external loop deviation from the defined profile. If the value is negative – no limitation. |
| <i>lci_segment_active</i> | [Mandatory with /p switch] Integer value. Fire LCI State or Pulse at the beginning of current segment. The function is available if the LCI segment-based mode was previously defined by the SegmentGate or SegmentPulse functions. The value defines the state value in SegmentGate mode (1 or 0). In SegmentPulse mode the value equal 1. |

Switches

The following optional ***/switches*** may be used singularly or in combination with the **ARC2** command:

| | |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>/v</i> | Specify required velocity. The switch requires an additional parameter that specifies the required velocity. The switch changes the required velocity for the current segment and for all subsequent segments. If the switch is not specified, the required velocity does not change. |
| <i>/f</i> | Decelerate to the end of segment. The switch requires an additional parameter that specifies the end velocity. The controller decelerates to the specified velocity at the end of segment. The specified value should be less than the required velocity; otherwise the parameter is ignored. The switch affects only one segment. The switch also disables corner detection and processing at the end of segment. If the switch is not specified, deceleration is not required. However, in special cases the deceleration might occur due to corner processing or other velocity control conditions. |
| <i>/o</i> | Synchronize user variables with segment execution. The switch requires additional two or three parameters that specify values , user variable and mask . See details in Arguments for explanation. |
| <i>/b</i> | Use external loops at corners. The switch requires additional parameters that specify the external loop type, the minimum segment length, and the maximum allowed deviation from profile. |
| <i>/p</i> | Specifies that the <i>lci_segment_active</i> parameter is required |

2.6.6 ARC2

This format of ARC2 is used for blended segment motion and in this form must be initialized with BSEG...ENDS. The command adds to the motion path an arc segment that starts in the current point and specified as the center point and rotation angle.

Syntax

```
ARC2[/switches] (axis_list),
    center_point_axis1,center_point_axis2,
    rotation_angle
    [,segment_time [,acceleration_time [,jerk_time [,dwell_time]]]]
```

Arguments

| Arguments | Comments |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Defines one or two axes, specified as axes numbers separated by comma or as axes names separated by comma. The axes should only be from those axes specified in the corresponding BSEG command. |
| <i>center_point_axis1</i> | Center point position for the first axis |
| <i>center_point_axis2</i> | Center point position for the second axis |
| <i>rotation_angle</i> | Defines central angle of the arc, signed according to rotation direction: plus on counter-clock-wise arc, minus on clock-wise arc. |
| <i>segment_time</i> | Only if switch/ m is specified: Segment time (T_m) in milliseconds. |
| <i>acceleration_time</i> | Only if switch/ a is specified: Acceleration time (T_a) in milliseconds. |
| <i>jerk_time</i> | Only if switch/ s is specified: Jerk time (T_j) in milliseconds |
| <i>dwell_time</i> | Only if switch/ d is specified: Dwell time at the final point of the segment in milliseconds. With this switch no blending will be done at the segment final point. |

2.6.7 BPTP

Description

BPTP defines a motion profile using the MotionBoost Feature.

Syntax

```
BPTP[/switch] axis_list, destination_point, [value of Tf, value of Vf, motor_motion_delay]
```

Switches

| Arguments | Comments |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| None | Travel time will be calculated by the MPU to achieve a the minimum possible time |
| /t | Minimum travel time in seconds, The calculated travel time will be at least the specified value. Incompatible with the /d switch. |
| /d | Travel Time – specifies the exact travel time for the motion in seconds. All other considerations are ignored, which could cause a safety fault during motion execution. Incompatible with the /t switch. |
| /f | User will enter final, nonzero velocity |
| /e | Wait for motion termination before executing next command. |
| /r | Relative motion |
| /v | Use velocity parameter instead of default velocity parameters. |
| /w | Create the motion, but to not start until the GO command is issued. |
| /z | Interpret entered coordinates according to the Local Coordinate System. |
| /m | Use the motion profile values of the axis group as a whole, rather than those of the leading axis, without exceeding any of the defined axes motion VEL, ACC, DEC, JERK values. Not compatible with /2 switch. Range is 0-25 ms. |
| /q | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |
| /2 | Use 20 kHz motion mode |



Use of the /d switch to specify minimum travel time overrides all other parameters which might limit velocity and requires careful attention to safety considerations.



BPTP/2 is limited to at most 2 axes in a single function call.



The **BPTP/2** command is limited to at most 2 axes per Servo Processor and at most 4 axes per system.

Arguments

axis_list

| | |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a | Axis a will perform the motion |
| (axes) | Axes will perform a synchronized motion to the destination point along to a straight line connecting the start to the destination point. Axes may be any set of distinct numbers in the range 0 to 63. |

destination-point

| | |
|------------|----------------------------------------------------------------------------------------------------------------------|
| (af,bf,cf) | af is a destination point for axis a bf is a destination point for axis b cf is a destination point for axis c |
|------------|----------------------------------------------------------------------------------------------------------------------|

Value of Tf

| | |
|----|---------------------|
| Tf | Desired travel time |
|----|---------------------|

Value of Vf

| | |
|----|------------------------|
| Vf | Desired final velocity |
|----|------------------------|

Motion Delay

| | |
|-----------------------------|---------------------------------------------------------------------------------------------------|
| motor_movement_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |
|-----------------------------|---------------------------------------------------------------------------------------------------|

GPHASE

Two options are available.

- > Four phases (For motion in positive direction; for motion in negative direction reverse the inequality signs)
 1. Acceleration buildup
 - > RJERK>0, RACC>0
 2. Acceleration finishing
 - > RJERK<0, RACC>0
 3. Deceleration buildup
 - > RJERK<0, RACC<0

4. Deceleration finishing
 - > RJERK>0, RACC<0

Comments

This command is supported in ADK versions 2.70 and higher.

Examples

```
BPTP 0, 100
```

A simple example that moves axis 0 to position 100.

If the axis is moving when the command is issued, the controller creates the motion and inserts it into the axis motion queue. The motion waits in the queue until all motions before it finish, and only then starts.

```
BPTP/ed (0,1), 100, 200, 0.4
```

Move axes 0,1 to position 100,200.

Do not execute the next command in the program until the motion is done.

Execute the motion in exactly 0.4 seconds.

```
// execute a BPTP motion with 20 KHz resolution
// move axes 0,1 to absolute coordinates 1,1

BPTP/2 (0,1),1,1
```

2.6.8 BPTPCalc

Description

The **BPTPCALC** function calculates and allows the user to set the motion variables according to a desired motion time. When the travel time and distance are known in advance, the **BPTPCALC** should be used to generate new **VEL**, **ACC** and **JERK** values.

Syntax

```
BPTPCALC real Motion_duration, real Distance, int index
```

Arguments

| | |
|-----------------|---------------------------------------------------------------------------------------------------------|
| Motion_duration | The desired motion time in seconds. The time will be rounded up to a whole number of controller cycles. |
| Distance | The travel distance in user units |
| Index | 1 - Velocity 2 - Acceleration 3 - Jerk |

Comments

This command is supported in ADK versions 2.70 and higher.

Example

```
! Calculate the motion parameter to execute a 10 unit BPTP motion in 3 ms
VEL(0) = bptpcalc(0.003, abs(10), 1)
ACC(0) = bptpcalc(0.003, abs(10), 2)
JERK(0) = bptpcalc(0.003, abs(10), 3)

enable(0)

BPTP/R(0), 10 ! Move axis 0, 10 units, in 3 ms

STOP
```

2.6.9 BSEG...ENDS

Syntax

BSEG[/switches] (*axis_list*), *initial_position_axis1*, *initial_position_axis2*, *segment_time*, *acceleration_time*, *jerk_time*[, *dwll_time*], [*motor_motion_delay*]

Arguments

| Arguments | Comments |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_list | Axes involved in BSEG motion, specified as axes numbers separated by comma or as axes names separated by comma |
| initial_position_axis1 | Initial position of the first axis |
| initial_position_axis2 | Initial position of the second axis |
| segment_time | Initial segment time (T_m) in milliseconds. The specified segment time will be used for all segments until <i>segment_time</i> argument is specified in segment LINE, ARC1 or ARC2 command. |
| acceleration_time | Initial acceleration time (T_a) in milliseconds. The specified acceleration time will be used for all segments until <i>acceleration_time</i> argument is specified in LINE, ARC1 or ARC2 command. |
| jerk_time | Initial jerk time (T_j) in milliseconds. The specified jerk time will be used for all segments until <i>jerk_time</i> argument is specified in LINE, ARC1 or ARC2 command. |

| | |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dwll_time | Optional] Initial dwell time between segments in milliseconds. If this argument is specified, no blending will be done for all segments of the motion. That means that the motion will be stopped at the end of each segment for the specified <i>dwll_time</i> milliseconds. |
| motor_motion_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

| Switch | Comments |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /w | Do not start until the GO command is executed. If the switch is not specified, the motion starts immediately after the first motion segment is defined. |
| /r | Set the initial axis position as origin. Segment commands positions should be declared relative to the new origin. |
| /z | Interpret entered coordinates according to the Local Coordinate System. |
| /q | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |

2.6.10 JOG

Description

JOG creates a motion with constant velocity and no defined end point.

JOG motion terminates by:

- > **HALT**, **KILL/KILLALL**, **BREAK**, **DISABLE/DISABLEALL**
- > Execution of any other motion command for the same axis
- > Limit switch activation
- > Any fault activation that disables the drive or kills the motion

Syntax

JOG[/switch] axis_list [,user-specified-velocity][,user-specified-direction]

Arguments

| | |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>user-specified-velocity</i> | Optional parameter used if the /v command option is specified |
| <i>user-specified-direction</i> | Optional parameter. Motion direction is designated by a plus sign (+) for increasing feedback counts or (-) or decreasing feedback counts. If no operator is used, motion is in the direction of increasing encoder counts. |

Switches

/switch can be:

| | |
|-----------|----------------------------------------------------------------------------|
| /v | Use the specified velocity instead of the default velocity (VEL). |
| /w | Create the motion, but do not start until GO . |

Related ACSPL+ Commands

[GO](#), [HALT](#), [KILL/KILLALL](#), [BREAK](#), [IMM](#)

Related ACSPL+ Variables

[AST](#), [MST](#), [MERR](#), [VEL](#), [ACC](#), [JERK](#), [FPOS](#), [RPOS](#), [GPHASE](#)

COM Library Methods and .NET Library Methods

Jog, JogM

C Library Functions

acsc_Jog, **acsc_JogM**

Examples

Example 1:

```
JOG/v 0, 500           !Jog 0 axis with VEL 500
```

Example 2:

```
JOG (0,1,2), -++      !Jog axes 0, 1, and 2 where axis 0 jogs in the
                       !negative direction and axes 1 and 2 jog in the
                       !positive direction.
```


2.6.11 LINE**Description**

LINE must be initialized with **MSEG...ENDS**. **LINE** adds a linear segment to a segmented motion.

Syntax

LINE *axis_list, final-position1, final-position2[,user-specified velocity]*

Arguments

| | |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.  A minimum of two axes must be specified. |
| <i>final position1</i> | First coordinate |
| <i>final position2</i> | Second coordinate |
| <i>user specified velocity</i> | Optional user-specified velocity if LINE was initiated by MSEG/v . |

Comments

1. **ENDS** informs the controller that no more segments will be specified. If **ENDS** is omitted, motion stops at the last point of the sequence and waits for the next point.
2. If the **LINE** axis_list involves two or more axes, see [PROJECTION](#).

Related ACSPL+ Commands

[MSEG...ENDS](#), [ARC1](#), [ARC2](#), [STOPPER](#), [PROJECTION](#)

COM Library Methods

See "Points and Segments Manipulation Methods" in *SPiiPlus COM Library Programmer's Guide*.

C Library Functions

See "Points and Segments Manipulation Methods" in *SPiiPlus C Library Reference Programmer's Guide*.

Example

See [MSEG...ENDS](#).


2.6.12 LINE**Description**

This format of **LINE** is used for extended segment motion and in this form must be initialized with [XSEG...ENDS](#). Use **LINE** to add a linear segment that starts at the current point and ends in the destination point to the motion path.

Syntax

LINE [/switches] (axis_list), destination_point_axis1, destination_point_axis2
[,destination_point_axis3 ... ,destination_point_axis6][,velocity][,end_velocity][,time][,values,
variables[,index[,masks]]] [,external_loop_type, external_loop_type, maximum_allowed_deviation][
,,ci_segment_active]

Arguments

| | |
|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | <p>List of axes numbers separated by comma or as axes names separated by comma. The axes should only be those axes specified in the corresponding XSEG command.</p> <div style="border: 1px solid black; border-radius: 10px; padding: 5px; margin-top: 10px;">  <p>A minimum of two axes must be specified.</p> </div> |
| <i>destination_point_axis1</i> | Destination position of the first axis |
| <i>destination_point_axis2</i> | Destination position of the second axis |
| <i>destination_point_axis3</i> ... <i>destination_point_axis6</i> | <p>Mandatory if axis_list contains more than 2 axes. Destination position of the rest of axes. Number of destination positions must correspond to the number of axes in axis_list.</p> |
| <i>velocity</i> | <p>[Optional, only used with /v switch] Defines required velocity for the current and for all subsequent segments. See Switches explanation for details.</p> |
| <i>end_velocity</i> | <p>[Optional, only used with /f switch] Defines required velocity at the end of the current segment. See Switches explanation for details.</p> |
| <i>time</i> | <p>[Mandatory with /t switch]. Defines segment processing time.</p> |
| <i>values</i> | <p>[Optional, only used with /o switch] Defines the values to be written to variables array at the beginning of the current segment execution. values is a one-dimensional user defined array of integer or real type with maximum size of 10 elements .</p> |
| <i>variables</i> | <p>[Optional, only used with /o switch] Defines the user-defined array, which will be written with values data at the beginning of the current segment execution. variables is a one-dimensional user defined array of the same type and size as the values array.</p> |
| <i>index</i> | [Optional, only used with /o switch] |

| | |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Defines the first element (starting from zero) of the variables array, to which values data will be written. If argument is omitted, values data is written to the variables array starting from the first element (index 0). |
| masks | <p>[Optional, only used if values and variables are integer]</p> <p>Defines the masks that are applied to values before the values are written to variables array at the beginning of the current segment execution. masks is a one-dimensional user-defined array of integer type and the same size as the values array. The masks are only applied for integer values:</p> $\text{variables}(n) = (\text{variables}(n) \text{ AND } (\text{NOT } \text{mask}(n))) \text{ OR } (\text{values}(n) \text{ AND } \text{mask}(n))$ <p>If argument is omitted, all values bits are written to variables.</p> <p>If values is a real array, the masks argument should be omitted.</p> |
| external_loop_type | <p>0 - Cancel external loop</p> <p>1 – Smooth External loop (line-arc-line)</p> <p>2 – Triangle External loop (line-line-line)</p> |
| minimum_segment_length | If the lengths of both segments are more than this value, the skywriting algorithm will be applied. |
| maximum_allowed_deviation | The parameter limits the external loop deviation from the defined profile. If the value is negative – no limitation. |
| lci_segment_active | <p>[Mandatory with /p switch]</p> <p>Integer value. Fire LCI State or Pulse at the beginning of current segment. The function is available if the LCI segment-based mode was previously defined by the SegmentGate or SegmentPulse functions. The value defines the state value in SegmentGate mode (1 or 0). In SegmentPulse mode the value equals 1.</p> |

Switches

The following optional **/switches** may be used singularly or in combination with the **LINE** command:

| | |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /v | <p>Specify required velocity.</p> <p>The switch requires an additional parameter that specifies the required velocity. The switch changes the required velocity for the current segment and for all subsequent segments.</p> <p>If the switch is not specified, the required velocity does not change.</p> |
| /f | Decelerate to the end of segment. |

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <p>The switch requires an additional parameter that specifies the end velocity. The controller decelerates to the specified velocity at the end of segment. The specified value should be less than the required velocity; otherwise the parameter is ignored. The switch affects only one segment.</p> <p>The switch also disables corner detection and processing at the end of segment.</p> <p>If the switch is not specified, deceleration is not required. However, in special cases the deceleration might occur due to corner processing or other velocity control conditions.</p> |
| /o | Synchronize user variables with segment execution. The switch requires additional two or three parameters that specify values , user variable and mask . See details in Arguments for explanation. |
| /b | Use external loops at corners. The switch requires additional parameters that specify the external loop type, the minimum segment length, and the maximum allowed deviation from profile. |
| /p | Specifies that the <code>lci_segment_active</code> parameter is required |

2.6.13 LINE

This format of LINE is used for blended segment motion and in this form must be initialized with BSEG...ENDS. The command adds to the motion path a linear segment that starts in the current point and ends in the destination point.

Syntax

```
LINE[/switches] (axis_list),destination_point_axis1,destination_point_axis2
    [,segment_time [,acceleration_time [,jerk_time [,dwell_time]]]]
```

Arguments

| Arguments | Comments |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Defines one or two axes, specified as axes numbers separated by comma or as axes names separated by comma. The axes should only be from those axes specified in the corresponding BSEG command. |
| <i>destination_point_axis1</i> | Destination position of the first axis |
| <i>destination_point_axis2</i> | Destination position of the second axis |
| <i>segment_time</i> | Only if the /m switch is specified: Segment time (Tm) in milliseconds. |
| <i>acceleration_time</i> | Only if /a switch is specified: Acceleration time (Ta) in milliseconds. |
| <i>jerk_time</i> | Only if the /s switch is specified: Jerk time (Tj) in millisecond |

dwel_time

Only if the **/d** switch is specified: Dwell time at the final point of the segment in milliseconds.

With this switch no blending will be done at the segment final point.

2.6.14 MASTER

Description

MASTER defines master-slave motion by creating a dependency between an axis position or velocity to a variable and/or an expression. **MASTER** always follows the (axis) **MPOS** variable. **SLAVE** initiates the motion defined by **MASTER** and must follow **MASTER**.



Velocity Lock is the default state for **MASTER**. Initiate **MASTER** with **SLAVE/p** for position lock.

The following actions terminate the master-slave dependency, however, these actions do not necessarily terminate the motion:

- > **KILL** or **HALT** to the slave axis.
- > **DISABLE** to either axis or both axes.
- > Setting the logical dependence between master and slaves axes to zero. For example, **MASTER MPOS(0) = 0**.

Syntax

MASTER *axis_MPOS=formula*

Arguments

| | |
|------------------|------------------------------|
| <i>axis_MPOS</i> | Define master value for axis |
| = | Assignment operator |
| <i>formula</i> | A variable and/or expression |

Related ACSPL+ Commands

[SLAVE](#), [GO](#), [HALT](#), [KILL/KILLALL](#), [ENABLE/ENABLE ALL](#), [DISABLE/DISABLEALL](#), [MAP](#)

Related ACSPL+ Variables

[AST](#), [XSACC](#), [MFF](#)

COM Library Methods and .NET Library Methods

SetMaster, Slave, SlaveStalled

C Library Functions

acsc_SetMaster, acsc_Slave, acsc_SlaveStalled

Examples

Example 1:

```

MASTER MPOS (0)=5*RVEL (1)
                                     !Creates a master-slave dependency where
                                     !the 0 axis velocity is slaved to five times
                                     !the 1 axis reference velocity (RVEL)
SLAVE 0                               !Initiates the 0 axis motion
  
```

Example 2:

Figure 2-7 illustrates **SLAVE/pt** used in the following syntax example.

```

MASTER MPOS (0)=FPOS (1)+MARK (1)
                                     !Create a master slave dependency, where
                                     !axis 0 is the slave. The master axis
                                     !is 1, and the FPOS (feedback position)
                                     !expression plus the MARK position
                                     !are the references for the slaved axis 0.
SLAVE/pt 0, -500, 500               !SLAVE command with position lock
                                     !and specified boundary switches,
                                     !for axis 0. -500 is the left
                                     !position boundary, and 500 is the
                                     !right position boundary.
  
```

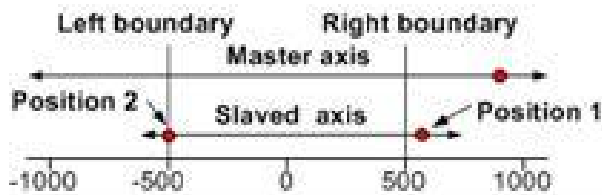


Figure 2-7. SLAVE /pt Illustration

In Figure 2-7, Position 1 is outside of the defined boundary and the master-slave dependency is stalled. Position 2 is within the defined boundary and the master-slave dependency is active.

2.6.15 MPOINT

Description

MPOINT specifies an array of destination points used by **MPTP...ENDS**, **PATH...ENDS** or **PVSPLINE...ENDS** motion commands. An **MPOINT** array must conclude with **ENDS**.

Syntax

MPOINT *axis_list, array-name (number of rows,number of points)*

Arguments

| | |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | List of axes involved. Axes must be in the same order as defined in MPTP , PATH or PVSPLINE . |
| <i>array (number of rows,number of points)</i> | <p>The MPOINT array must be defined as Real.</p> <p>The number of rows is the first element and the number of columns is the second element. A three-axis, five-point array is defined as follows: real array (3)(5).</p> <p>Each row defines the values for the points for each axes, and depending on command preceding MPOINT, the defined velocities, and the defined times. The array must contain, at least, the same number of columns declared in the array as the number of defined points. Extra array columns are ignored. See Array Structures for a description of various array configurations.</p> |

Related ACSPL+ Commands

[MPTP...ENDS](#), [POINT](#), [PATH...ENDS](#), [PVSPLINE...ENDS](#)

COM Library Methods and .NET Library Methods

MultiPoint, MultiPointM, Spline, SplineM, AddPVPoint, AddPVPointM, AddPVTPoint, AddPVTPointM

C Library Functions

acsc_MultiPoint, acsc_MultiPointM, acsc_Spline, acsc_SplineM, acsc_AddPVPoint, acsc_AddPVPointM, acsc_AddPVTPoint, acsc_AddPVTPointM

Array Structures

There are different point array structures depending on the **MPOINT** enabling command and switches. These structures are described below.

1. A five-point, three-axis **MPOINT** array enabled by **MPTP** or **PATH** without switches appears as follows:

| ARRAY (3)(5) | | | | | |
|---------------|---------|---------|---------|---------|---------|
| | Point 1 | Point 2 | Point 3 | Point 4 | Point 5 |
| 0 Axis | 1000 | 2000 | 3000 | 4000 | 5000 |
| 1 Axis | 5000 | 4000 | 3000 | 2000 | 1000 |
| 5 Axis | 0 | 2000 | 4000 | 6000 | 8000 |

See [Example 1](#), illustrating sample code based on the **ARRAY (3)(5)** structure.

2. If **MPOINT** follows **MPTP/v**, the point array must include an additional row to specify the transition velocity from the previous point to the current point and appears as follows:

| ARRAY (4)(5) | | | | | |
|--------------|---------|---------|---------|---------|---------|
| | Point 1 | Point 2 | Point 3 | Point 4 | Point 5 |
| 0 Axis | 1000 | 2000 | 3000 | 4000 | 5000 |
| 1 Axis | 5000 | 4000 | 3000 | 2000 | 1000 |
| 5 Axis | 0 | 2000 | 4000 | 6000 | 8000 |
| Velocity | 6000 | 7000 | 5000 | 4000 | 4000 |

See [Example 2](#), illustrating sample code based on the **ARRAY (4)(5)** structure.

- If **MPOINT** follows **PATH/t**, the point array must include an additional row to specify the time interval between the previous point and the current point and appears as follows:

| ARRAY (4)(5) | | | | | |
|--------------|---------|---------|---------|---------|---------|
| | Point 1 | Point 2 | Point 3 | Point 4 | Point 5 |
| 0 Axis | 1000 | 2000 | 3000 | 4000 | 5000 |
| 1 Axis | 5000 | 4000 | 3000 | 2000 | 1000 |
| 5 Axis | 0 | 2000 | 4000 | 6000 | 8000 |
| Time | 250 | 250 | 250 | 250 | 250 |

Time is specified in milliseconds.

- If **MPOINT** follows **PVSPLINE** without the **/t** switch, the array must include two sets of rows for each axis:
 - > The first set defines the destination points
 - > The second set defines the velocity at the destination points

For example:

| ARRAY (6)(5) | | | | | |
|--------------|---------|---------|---------|---------|---------|
| | Point 1 | Point 2 | Point 3 | Point 4 | Point 5 |
| 0 Axis | 1000 | 2000 | 3000 | 4000 | 5000 |
| 1 Axis | 5000 | 4000 | 3000 | 2000 | 1000 |
| 5 Axis | 0 | 2000 | 4000 | 6000 | 8000 |
| 0 Axis VEL | 5000 | 5000 | 5000 | 5000 | 5000 |

| ARRAY (6)(5) | | | | | |
|--------------|---------|---------|---------|---------|---------|
| | Point 1 | Point 2 | Point 3 | Point 4 | Point 5 |
| 1 Axis VEL | 2500 | 2500 | 2500 | 2500 | 2500 |
| 5 Axis VEL | 3000 | 3000 | 3000 | 3000 | 3000 |

5. If **MPOINT** follows **PVSPLINE/t**, the array must include an additional column that specifies the time interval between the previous point and the current point. **Time** is in milliseconds. For example:

| ARRAY (7)(5) | | | | | |
|--------------|---------|---------|---------|---------|---------|
| | Point 1 | Point 2 | Point 3 | Point 4 | Point 5 |
| 0 Axis | 1000 | 2000 | 3000 | 4000 | 5000 |
| 1 Axis | 5000 | 4000 | 3000 | 2000 | 1000 |
| 5 Axis | 0 | 2000 | 4000 | 6000 | 8000 |
| 0 Axis VEL | 5000 | 5000 | 5000 | 5000 | 5000 |
| 1 Axis VEL | 2500 | 2500 | 2500 | 2500 | 2500 |
| 5 Axis VEL | 3000 | 3000 | 3000 | 3000 | 3000 |
| Time | 500 | 500 | 500 | 500 | 500 |

See [Example 3](#), illustrating sample code based on the **ARRAY (7)(5)** structure.

6. If **MPTP/r** enables **MPOINT**, the array points are relative.

Examples

Example 1

Illustrating sample code based on the [ARRAY \(3\)\(5\)](#) structure.

```
REAL ARRAY(3)(5)           !Defines ARRAY with three rows and
                           !five columns as real.
INT NUMBER_of_POINTS      !Defines NUMBER_of_POINTS as an integer variable.
! ----- Fill the ARRAY array-----
ARRAY(0)(0)=1000;ARRAY(0)(1)=2000;ARRAY(0)(2)=3000;ARRAY(0)(3)=4000;
ARRAY(0)(4)=5000;ARRAY(1)(0)=5000;ARRAY(1)(1)=4000;ARRAY(1)(2)=3000;
ARRAY(1)(3)=2000;ARRAY(1)(4)=1000;ARRAY(2)(0)=0;ARRAY(2)(1)=2000;
ARRAY(2)(2)=4000;ARRAY(2)(3)=6000;ARRAY(2)(4)=8000
NUMBER_of_POINTS=5        !Set NUMBER_of_POINTS
ENABLE(0,1,5)             !Enable axes 0, 1 and 5
PATH(0,1,5), 1            !PATH initiates simultaneous motion for axes 0, 1,
                           !and 5. The time interval is one millisecond between
```



```

                                !each destination point.
MPOINT (0,1,5), ARRAY, NUMBER_of_POINTS
                                !Define an MPOINT array for axes 0, 1 and 5 where
                                !ARRAY and NUMBER_of_POINTS are called
ENDS (0,1,5)                    !Concludes MPOINT
STOP                             !End program

```

Example 2

Illustrating sample code based on the [ARRAY \(4\)\(5\)](#) structure.

```

REAL ARRAY(4)(5)                !Defines ARRAY with four rows and
                                !five columns as real.
INT NUMBER_of_POINTS            !Defines NUMBER_of_POINTS as an integer
                                !variable.
! ----- Fill the ARRAY array-----
ARRAY(0)(0)=1000;ARRAY(0)(1)=2000;ARRAY(0)(2)=3000;ARRAY(0)(3)=4000;
ARRAY (0)(4)=5000;ARRAY(1)(0)=5000;ARRAY(1)(1)=4000;ARRAY(1)(2)=3000;
ARRAY (1)(3)=2000;ARRAY(1)(4)=1000;ARRAY(2)(0)=0;ARRAY(2)(1)=2000;
ARRAY(2)(2)=4000;ARRAY(2)(3)=6000;ARRAY(2)(4)=8000;ARRAY(3)(0)=6000;
ARRAY(3)(1)=7000;ARRAY(3)(2)=5000;ARRAY(3)(3)=4000;ARRAY(3)(4)=4000
NUMBER_of_POINTS=5              !Set NUMBER_of_POINTS
ENABLE (0,1,5)                  !Enable axes 0, 1 and 5
PATH (0,1,5), 1
                                !PATH initiates simultaneous motion for axes
                                !0, 1 and 5. The time interval is one millisecond
                                !between each destination point.
MPOINT (0,1,5), ARRAY, NUMBER_of_POINTS
                                !Define an MPOINT array for axes 0, 1 and 5
                                !where ARRAY and NUMBER_of_POINTS are called.
ENDS (0,1,5)                    !Concludes MPOINT
STOP                             !End program

```

Example 3

Illustrating sample code based on the [ARRAY \(7\)\(5\)](#) structure.

```

REAL ARRAY(7)(5)                !Defines ARRAY with seven rows and
                                !five columns as real.
INT NUMBER_of_POINTS            !Defines NUMBER_of_POINTS as an integer variable.
! ----- Fill the ARRAY array-----
ARRAY(0)(0)=1000;ARRAY(0)(1)=2000;ARRAY(0)(2)=3000;ARRAY(0)(3)=4000;
ARRAY (0)(4)=5000;ARRAY(1)(0)=5000;ARRAY(1)(1)=4000;ARRAY(1)(2)=3000;
ARRAY (1)(3)=2000;ARRAY(1)(4)=1000;ARRAY(2)(0)=0;ARRAY(2)(1)=2000;
ARRAY(2)(2)=4000;ARRAY(2)(3)=6000;ARRAY(2)(4)=8000;ARRAY(3)(0)=5000;
ARRAY(3)(1)=5000;ARRAY(3)(2)=5000;ARRAY(3)(3)=5000;ARRAY(3)(4)=5000;
ARRAY(4)(0)=2500;ARRAY(4)(1)=2500;ARRAY(4)(2)=2500;ARRAY(4)(3)=2500;
ARRAY(4)(4)=2500;ARRAY(5)(0)=3000;ARRAY(5)(1)=3000;ARRAY(5)(2)=3000;
ARRAY(5)(3)=3000;ARRAY(5)(4)=3000;ARRAY(6)(0)=500;ARRAY(6)(1)=500;
ARRAY(6)(2)=500;ARRAY(6)(3)=500;ARRAY(6)(4)=500
NUMBER_of_POINTS=5              !Set NUMBER_of_POINTS

```

```

ENABLE (0,1,5)           !Enable axes 0, 1 and 5
PVSPLINE/t (0,1,5)      !PVSPLINE/t initiates simultaneous motion for axes
                        !0, 1 and 5 with a user-defined time interval
                        !between each point.
MPOINT (0,1,5), ARRAY, NUMBER_of_POINTS
                        !Define an MPOINT array for axes 0, 1 and 5
                        !where ARRAY and NUMBER_of_POINTS are called.
ENDS (0,1,5)            !Concludes MPOINT
STOP                    !End program

```

2.6.16 MPTP..ENDS

Description

MPTP initiates multipoint sequential positioning to a set of points. **MPTP** by itself does not specify any point, however dwell time at each point can be optionally specified. Points are specified by the **POINT** or **MPOINT** commands that follow **MPTP**.

In single axis motion, **MPTP** generates sequential motion between the defined array points, where at the end of each segment, **RVEL** = 0, as if each segment was defined by a separate **PTP** command.

In group motion, where more than one axis is declared, the first axis in the **axis_list** is the leading axis. The motion parameters of the leading axis become the default motion parameters for all axes in the group. Motion on all axes in a group start and conclude at the same time. **MPTP** generates sequential motion between the defined array points, where at the end of each set of points, **RVEL** = 0, as if each motion was defined by a separate **PTP** command.

Transition to the next motion in the motion queue, if it exists, will not occur until **ENDS** executes.

MPTP terminates with:

- > **HALT**, **KILL/KILLALL**, or **BREAK**
- > Any fault activation that disables the drive or kills the motion
- > **DISABLE/DISABLEALL** by the user

Syntax

MPTP[/switch] **axis_list**[,dwell][,motor_motion_delay]

Arguments

| | |
|---------------------------|----------------------------------------------------------------------------------------------------------|
| axis_list | List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| dwell | Dwell is an optional argument, expressed in milliseconds. |
| motor_motion_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

/switch can be:

| | |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>/w</code> | Create the motion, but do not start until the GO command. |
| <code>/v</code> | Use the specified velocity instead of the default velocity (VEL). |
| <code>/c</code> | Use the point sequence as a cyclic array |
| <code>/r</code> | Coordinates of each point are relative to the previous point's coordinates. |
| <code>/z</code> | Interpret entered coordinates according to the Local Coordinate System. |
| <code>/q</code> | <p>Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds.</p> <p>The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms.</p> <p>Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately.</p> |

Comments

1. **MPTP** motion starts only after the first point is specified
2. **MPTP** motion with all of its points is considered as one motion command in the motion queue.
3. An **MPOINT** array declaration used in **MPTP** must be defined as Real.
4. **MPTP/c** does not end automatically. Use **HALT**, **KILL/KILLALL**, or **BREAK** to stop cyclic motion.

Related ACSPL+ Commands

[GO](#), [HALT](#), [KILL/KILLALL](#), [BREAK](#), [IMM](#), [MPOINT](#), [POINT](#)

Related ACSPL+ Variables

[ACC](#), [DEC](#), [JERK](#), [VEL](#)

COM Library Methods and .NET Library Methods

[ToPoint](#), [ToPointM](#), [ExtToPoint](#), [ExtToPointM](#)

C Library Functions

[acsc_ToPoint](#), [acscToPointM](#), [acsc_ExtToPoint](#), [acsc_ExtToPointM](#)

Examples

Example 1:

In the following example, **dwel** is not required, therefore the comma and the second argument following 1000 are omitted.

```
MPTP 0, 1000           !Create a multipoint motion of the 0 axis
                       !with a dwell of 1000 msec at each point.
```

Example 2:

```

REAL ARRAY1 (4)           !Define an array called ARRAY1 with four members
                           !as REAL
! ----- Fill the ARRAY1 array-----
ARRAY1(0)=0;ARRAY1(1)=1000;ARRAY1(2)=0;ARRAY1(3)=1000
MPTP 0, 500              !MPTP motion for 0 axis, dwell 500 msec at each point
POINT 0, 2000            !First point
MPOINT 0, ARRAY1,3      !Use three points from ARRAY1
POINT 0, 3000            !Second point
POINT 0, 5000           !Third point
ENDS 0                   !Ends the point sequence for 0 axis

```

Figure 2-8 illustrates the above code for a single-axis motion initiated by **MPTP**. Note that not all of the members defined in **ARRAY1** necessarily need to be used by **MPOINT**. In this example, only three of the defined members are called.

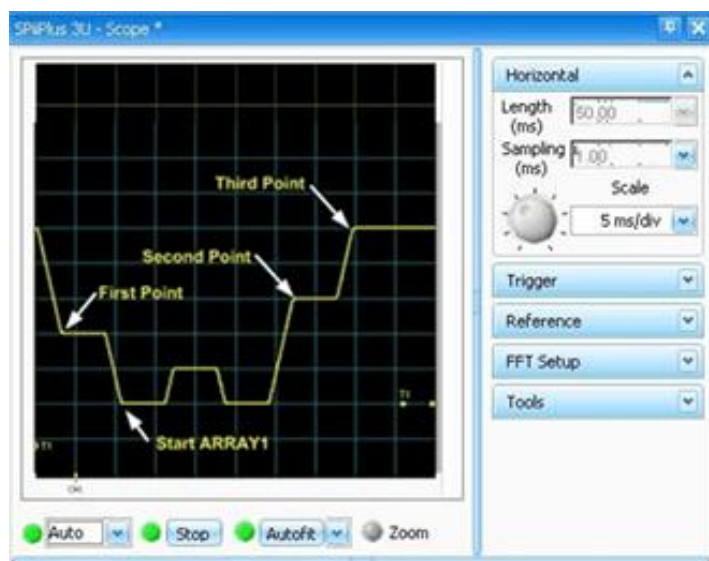


Figure 2-8. Single-Axis Motion Using MPTP

Example 3:

```

MPTP/v XY                !Create multipoint motion in group XY with
                           !user defined velocity and no dwell.
POINT XY,10000,0,5000    !Move to first point at velocity 5000
POINT XY,1000,1000,5000 !Add second point at velocity 5000
POINT XY,2000,1000,5000 !Add third point at velocity 5000
POINT XY,2000,0,100     !Add fourth point at velocity 100
ENDS XY                  !End the point sequence for XY

```

Figure 2-9 illustrates the above code for two-axis group motion initiated by **MPTP/v**.

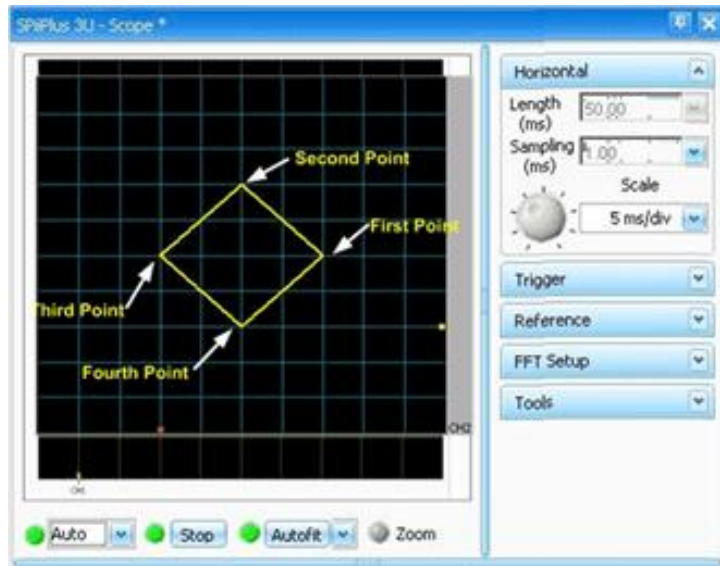


Figure 2-9. Two-Axis Group Motion Using MPTP/v

2.6.17 MSEG...ENDS

Description

MSEG initiates two-axis segmented motion. **MSEG** itself does not specify a line or arc segment. Motion starts only after the first segment is specified with a motion segment command.

Segmented motion moves axes along a continuous path where the path is defined as a sequence of line and arc segments on a plane.

Use the following commands to define segmented motion:

- > **ARC1** - adds an arc segment to a segmented motion and specifies the coordinates of center point, coordinates of the final point, and the direction of rotation
- > **ARC2** - Adds an arc segment to a segmented motion and specifies the coordinates of center point, rotation angle and direction.
- > **ENDS** - terminates the point sequence
- > **LINE** - adds a linear segment to a segmented motion.
- > **PROJECTION** - sets a projection array for a segmented motion.
- > **STOPPER** - provides a smooth transition between two segments of segmented motion.

MSEG motion terminates with:

- > **HALT**, **KILL/KILLALL**, or **BREAK**
- > Any fault activation that disables the drive or kills the motion
- > **DISABLE/DISABLEALL** by the user

Syntax

MSEG[/switch] *axis_list, initial-position-axis1, initial-position-axis2*

Arguments

| | |
|-------------------------------|--------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axes list, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>initial-position axis1</i> | Start coordinate for the first axis |
| <i>initial-position-axis2</i> | Start coordinate for the second axis |

Switches

/switch can be:

| | |
|-----------|----------------------------------------------------------------------------|
| <i>/w</i> | Create the motion, but do not start until GO . |
| <i>/v</i> | Use the specified velocity instead of the default velocity (VEL). |
| <i>/c</i> | Use the MSEG segment sequence as a cyclic array. |

Comments

- > Use command option */c* to create cyclic motion where the final point of the last segment becomes the first point of the next motion cycle. **MSEG/c** does not automatically finish. Use **HALT**, **KILL/KILLALL**, or **BREAK** to end cyclic motion.
- > If command option */v* is used, specify the user-defined velocity in each instance of **LINE**, **ARC1**, or **ARC2**.
- > The MSEG command uses a motion que buffer of up to 50 segments.

Related ACSPL+ Commands

GO, **HALT**, **KILL/KILLALL**, **BREAK**, **IMM**

Related ACSPL+ Variables

ACC, **DEC**, **JERK**, **VEL**

COM Library Methods

For **MSEG**: Segment, Line, ExtLine, Arc1, ExtArc1, Arc2, ExtArc2, Stopper, Projection

For **ENDS**: FinalPoint

C Library Functions

For **MSEG**: acsc_Segment, acsc_Line, acsc_Arc1, acsc_Arc2, acsc_ExtLine, acsc_ExtArc1, acsc_ExtArc2, acsc_Projection, acsc_Stopper

For **ENDS**: acsc_FinalPoint

Example

```

MSEG (0,1), 1000, 1000 !MSEG initiates segmented motion for the 0 and 1
axes
                                !group with initial coordinates of (1000,1000).
ARC1 (0,1), 1000, 0, 1000, -1000, -
                                !Add an arc segment with a center point located at
                                !(1000,0) and the final point located at (1000,-1000)
                                !with negative movement in terms of the encoder.
LINE (0,1), -1000, -1000
                                !Add line segment with final point (-1000,-1000).
ARC2 (0,1), -1000, 0, -3.141529
                                !Add arc segment with center (-1000,0) and a
                                !rotation angle of -p radians.
LINE (0,1), 1000, 1000
                                !Add line segment with center (1000,1000).
ENDS (0,1)
                                !Ends the point sequence for the 0 and 1 axes group.
STOP
                                !Ends program
    
```

Figure 2-10 illustrates the motion created by the example.

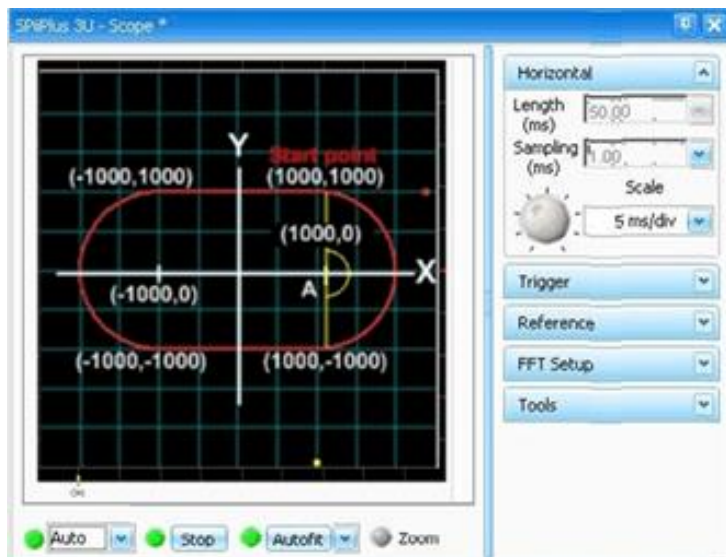


Figure 2-10. Results of Example MSEG

2.6.18 PATH...ENDS

Description

PATH initiates an arbitrary path motion with linear interpolation using **POINT**, or **MPOINT** for an array of points. The arbitrary path sequence must conclude with **ENDS**.

PATH motion terminates due to:

- > Interruption by any new motion command before the current motion concludes terminates the **PATH** motion and causes an error.
- > Any fault activation that disables the drive or kills the motion

- > User termination by [HALT](#), [KILL/KILLALL](#), or [DISABLE/DISABLEALL](#)

Syntax

PATH[/*switches*] *axis_list* [*time-interval*][, *motor_motion_delay*]

Arguments

| | |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axes list, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>time interval</i> | Optional time interval specification in milliseconds. PATH specified without \t must have the time interval specification as the last argument. The argument defines the time interval between all motion points. |
| <i>motor_motion_delay</i> | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

| | |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>/w</i> | Create the motion, but do not start until the GO command. |
| <i>/t</i> | Enables a user-defined time interval in a point array |
| <i>/c</i> | Use the point sequence as a cyclic array. After arriving at the last point, continue from the first point. |
| <i>/r</i> | Coordinates of each point are relative to the previous point's coordinates. |
| <i>/q</i> | <p>Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds.</p> <p>The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms.</p> <p>Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately.</p> |

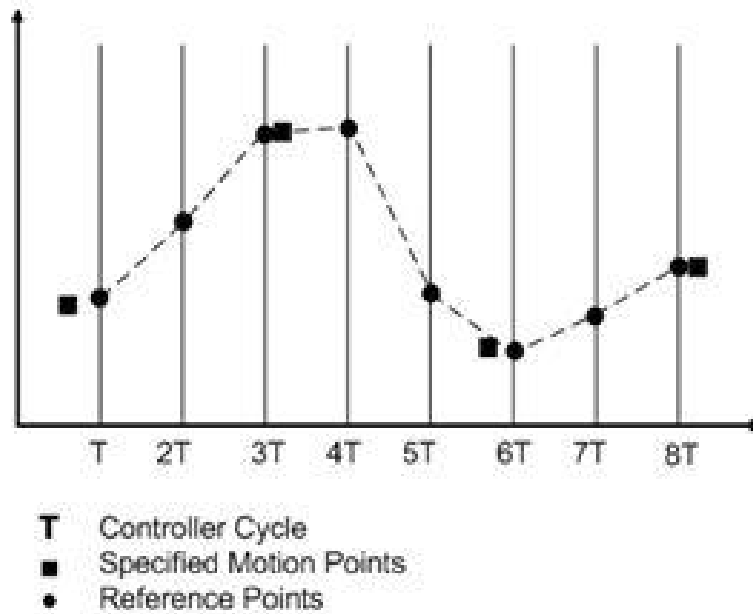


Figure 2-11. PATH...ENDS Diagram

Comments

Since a time interval and the destination point are specified, variables [VEL](#), [ACC](#), [DEC](#), [JERK](#) have no effect on this motion.

Related ACSPL+ Commands

[MPTP...ENDS](#), [POINT](#), [MPOINT](#), [PVSPLINE...ENDS](#)

COM Library Methods and .NET Library Methods

ToPoint, ToPointM, ExtToPoint, ExtToPointM

C Library Functions

acsc_ToPoint, acsc_ToPointM, acsc_ExtToPoint, acsc_ExtToPointM

Example

See [Example 2](#) of [MPOINT](#) for a three axis example of [MPTP...ENDS](#) motion.

2.6.19 POINT

Description

POINT adds a destination point to multi-point or arbitrary motion paths. A sequence of destination points can be specified with a sequence of **POINT** commands. **POINT** must follow [MPTP...ENDS](#), [PATH...ENDS](#), or [PVSPLINE...ENDS](#). Refer to each command for a list of available command options. The sequence of specified points must conclude with **ENDS**.

Syntax

- > **POINT** syntax depends on the command options used in the initializing commands, as follows:
- > **POINT** initiated by **MPTP**:
POINT *axis_list, axis_list destination positions*

- > **POINT** initiated by **MPTP/v**:
POINT *axis_list, axis_list destination positions, vector velocity (GVEL)*
 See [Example 1](#).
- > **POINT** initialized by **PATH**:
PATH[*command options*]*axis_list, time-interval*
POINT *axis_list, axis_list destination positions*
 See [Example 2](#).
- > **POINT** initialized by **PATH/t**:
POINT *axis_list, axis_list destination positions, time interval in milliseconds*
 See [Example 3](#).
- > **POINT** initialized by **PVSPLINE**:
PVSPLINE[*command options*]*axis_list, time-interval*
POINT *axis_list, axis_list destination positions*
 See [Example 4](#).
- > **POINT** initialized by **PVSPLINE/t**:
PVSPLINE[*command options*]*axis_list*
POINT *axis_list, axis_list destination positions, velocity at destination position, time interval in milliseconds*
 See [Example 5](#).

Comments

- > **POINT** axis specifications must follow the same order as the initializing **MPTP**, **PATH** or **PVSPLINE** commands.
- > All arguments following the *axis_list* must appear as a comma separated list, and must correspond to the number of axes.

Related ACSPL+ Commands

[MPOINT](#), [MPTP...ENDS](#), [PATH...ENDS](#), [PVSPLINE...ENDS](#)

COM Library Methods and .NET Library Methods

MultiPoint, MultiPointM, Spline, SplineM, AddPVPoint, AddPVPointM, AddPVTPoint, AddPVTPointM

C Library Functions

acsc_MultiPoint, acsc_MultiPointM, acsc_Spline, acsc_SplineM, acsc_AddPVPoint, acsc_AddPVPointM, acsc_AddPVTPoint, acsc_AddPVTPointM

Examples

Example 1

```
MPTP/v (0,1,5)           !Initiates MPTP motion for axes 0, 1 and 5.
                        !simultaneously with a user-defined velocity(GVEL)
                        !at each point.
POINT (0,1,5), 1000,4000,6000,7000
                        !Defines destination points 1000, 4000, and 6000 for
                        !axes 0, 1, and 5 with a GVEL of 7000 units.
ENDS (0,1,5)           !Ends MPTP/v point sequence for axes 0, 1, and 5.
STOP                   !Ends program
```

Example 2

```

PATH/r (0,1,5),1000      !Initiates PATH motion for axes 0, 1, and 5.
                        !simultaneously where each destination point is
                        !relative to the previous point and the time
                        !interval between each point is one second.

POINT (0,1,5), 1000,2000,3000
                        !Defines destination points 1000, 2000, and 3000 for
                        !axes 0, 1, and 5 respectively.

ENDS (0,1,5)           !Ends PATH point sequence for axes 0, 1, and 5.
STOP                   !Ends program
  
```

Example 3

```

PATH/t (0,1,5)          !Initiates PATH motion for axes 0, 1, and 5.
                        !simultaneously with a user-defined time interval
                        !between points.

POINT (0,1,5), 1000,2000,3000, 500
                        !Defines destination points 1000, 2000, 3000 for axes
                        !0, 1, and 5, respectively and a time interval of
                        !500 msec between points.

ENDS (0,1,5)           !Ends PATH point sequence for axes 0, 1, and 5.
STOP                   !Ends program
  
```

Example 4

```

PVSPLINE (0,1,5), 10   !Initiates PVSPLINE motion for axes 0, 1, and 5.
                        !Points are given at 10 msec intervals.

POINT (0,1,5), 200, 100, 300, 1000, 2000, 1500
                        !Defines points for axes 0, 1, and 5 with respective
                        !values of 200, 100, 300 and velocities at each
                        !point 1000, 2000, 1500, respectively.

ENDS (0,1,5)           !Ends PVSPLINE point sequence.
STOP                   !Ends program
  
```

Example 5

```

PVSPLINE/t (0,1,5)     !Initiates PVSPLINE motion for axes 0, 1, and 5, with
                        !a user-defined time interval between points.

POINT (0,1,5), 200, 100, 300, 1000, 2000, 1500, 250
                        !Defines destination points for axes 0, 1, and 5
                        !with respective values of 200, 100, 300 and
                        !velocities at each respective point of 1000, 2000,
                        !1500. The time interval between points is 250 msec.

ENDS (0,1,5)           !Ends PVSPLINE point sequence for axes 0, 1, and 5.
STOP                   !Ends program
  
```

2.6.20 PROJECTION

Description

PROJECTION is an expansion command to the [MSEG...ENDS](#) set of commands, that allows the controller to perform a three dimensional segmented motion such as creating arcs and lines on a user-defined plane. The method for this 3D segmented motion is to set a transformation matrix that defines a new plane for the segmented motion.

Syntax

PROJECTION *axes_list*, *Transformation Matrix*

Arguments

| | |
|------------------------------|-----------------------------------------------------------------------------------------------|
| <i>axes_list</i> | List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>Transformation Matrix</i> | Defines the new plane |

Comments

1. **MSEG** must precede **PROJECTION**.
2. Prior to using **PROJECTION**, the related axes must be grouped using [GROUP](#).
3. The motion parameters of the segmented motion with **PROJECTION** are calculated based on the leading axis described in the [GROUP](#) command. The other involved axes motion parameters are not relevant. The parameters are calculated to meet a uniform travel time to all grouped axes.

Related ACSPL+ Commands

[MSEG...ENDS](#), [LINE](#), [ARC1](#), [ARC2](#), [GROUP](#)

COM Library Methods and .NET Library Methods

Projection

C Library Functions

acsc_Projection

Example

1. This example program creates a segmented motion on a new plane (in this example - a circle) on a plane rotated around Axis X by 70° relative to plane AX, as illustrated in [Figure 2-12](#).
2. **ARC2** defines the circle's center coordinates on plane AX. **PROJECTION** transforms these coordinates to the new plane, based on the values in the transformation matrix (Matrix M, in this example).
3. Populate the transformation matrix with the values from table given below. The first two rows define the relationship between the X and Y coordinates of the last [MSEG...ENDS](#) motion and their respective axes in the new plane. The third row defines the tangent angle of the new plane. In this example uses a 70° angle in reference to Axis A, where $\tan 70 = 2.74$.

Table 2-8. Matrix Values

Matrix Values

| Axis | X Coordinate | A Coordinate |
|------|--------------|--------------|
| X | 1 | 0 |
| A | 0 | 1 |
| B | 0 | 2.74 |

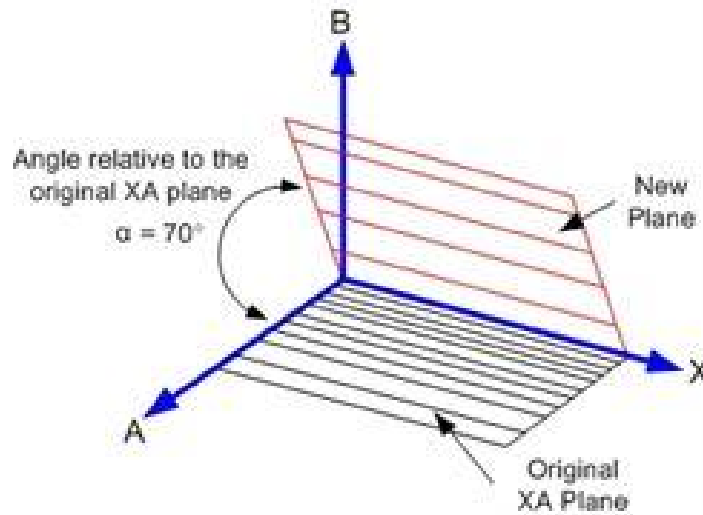


Figure 2-12. PROJECTION of the XA Plane

Figure 2-13 illustrates the reference position of axes 0, 4, and 5 (corresponding to the XAB coordinates illustrated in Figure 2-12).

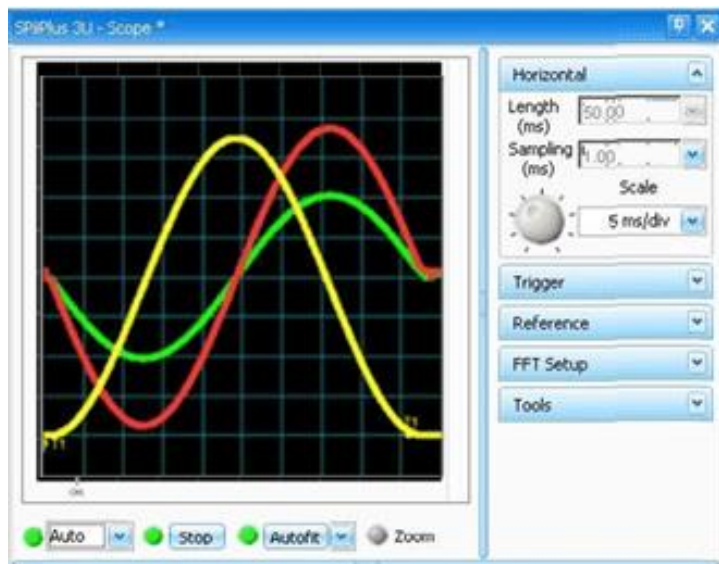


Figure 2-13. FPOS - PROJECTION Example

Figure 2-14 illustrates the circle's trajectory viewed from the XA plane.

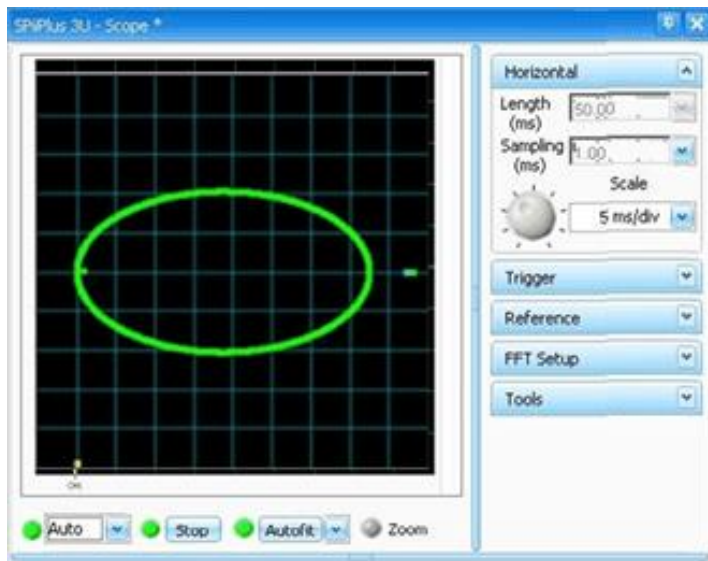


Figure 2-14. PROJECTION Example - Final Result

```

REAL M(3) (2)           !Defines Transformation Matrix M as real array
! ----- Set the transformation matrix values-----
M(0) (0)=1;M(0) (1)=0;M(1) (0)=0;M(1) (1)=1;M(2) (0)=0;M(2) (1)=2.74
VEL(0)=1000;ACC(0)=10000;DEC(0)=10000
                        !Axis motion parameters
ENABLE (0,4,5)         !Enable the 0, 4 and 5 axes (required)
                        !Corresponding to XAB coordinates shown in Figure 15.
GROUP (0,4,5)          !Group the 0, 4 and 5 axes (required)
SET FPOS(0)=0;SET FPOS(4)=0;SET FPOS(5)=0
                        !Sets the FPOS for 0, 4 and 5, respectively, to 0.
MSEG (0,4),0,0         !Define original plane.
PROJECTION (0,4,5),M   !PROJECTION of axes XAB by matrix M.
ARC2 (0,4),750,0,6,24 !ARC2 performed on new plane.
ENDS (0,4)             !Concludes MSEG.
STOP                   !End Program
    
```

2.6.21 PTP

Description

PTP (point-to-point) generates motion for the specified axis or axes to a specified destination point.

When **PTP** specifies a single axis, the motion profile is calculated according to **VEL**, **ACC**, **DEC**, **JERK** values of the axis.

In group motion, when **PTP** specifies multiple axes, the group motion profile is based on the leading axis' **VEL**, **ACC**, **DEC**, **JERK** motion values, unless **PTP/m** is used.

PTP terminates due to:

- > Interruption by any new motion command
- > Any fault activation that disables the drive or kills the motion

- > User termination by **HALT** , **KILL/KILLALL**, or **BREAK**.

Syntax

PTP[switches]*axis_list, destination-point*[value for v, value for f, motor_movement_delay]

Arguments

| | |
|-----------------------------|------------------------------------------------------------------------------------------------------------|
| axis_list | Single axis or axis group, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| destination point | Final destination. |
| value for v | Optional argument for user-defined velocity. |
| value for f | Optional argument for user-defined velocity at a destination point |
| motor_movement_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /e | Wait for motion termination before executing next command. |
| /f | Specify non-zero velocity at each destination point (or points) in a series of PTP motions |
| /m | Use the motion profile values of the axis group as a whole, rather than those of the leading axis, without exceeding any of the defined axes motion VEL , ACC , DEC , JERK values. |
| /r | The destination point is relative to the start point. |
| /v | Use the specified velocity instead of the default velocity (VEL). |
| /w | Create the motion, but do not start until GO. |
| /z | Interpret entered coordinates according to the Local Coordinate System. |
| /q | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |

Comments

- > Axes destination points, and relative velocity in the **PTP** command can also be an expression.

- > **PTP** can be used for executing point-to-point motion for a group of axes. For example, `PTP (0,1,2)` creates motion for axes 0, 1, and 2 as a group.

Related ACSPL+ Commands

[MPTP...ENDS](#), [POINT](#)

COM Library Methods and .NET Library Methods

ToPoint

C Library Functions

acsc_ToPoint

Examples

Example 1:

```
PTP/v 1, 2000, 500      !PTP axis 1 to point 2000 with velocity 500
```

Example 2:

```
PTP/rw (0,1), 1000, 2000 !PTP axes 0 and 1 where the 1 target point is
1000
                                !and the 0 target point is 2000. The target points
                                !are relative to the start point. Motion will not
                                !commence until GO command is issued.
```

Example 3:

```
ENABLE 0                    !Enables axis 0
VEL(0)=10000                !Sets the default axis 0 velocity to 10000
SET RPOS(0)=0               !Sets the current axis 0 position to 0
PTP/rf 0, 2000, 1000       !Initiates a relative axis 0 motion of 2000 with end
                                !velocity of 1000 at the destination point
PTP 0, 4000                 !Initiates an absolute axis 0 motion to 4000
STOP                        !Ends program
```

2.6.22 PVSPLINE...ENDS

Description

PVSPLINE (position-velocity spline) creates an arbitrary motion trajectory where the controller provides cubic spline interpolation between two points. The user specifies the end point and the end velocity for each motion segment. **ENDS** must terminate the point sequence.

PVSPLINE motion terminates due to:

- > Interruption by any new motion command
- > Any fault activation that disables the drive or kills the motion
- > User termination by [HALT](#), [KILL/KILLALL](#), or [DISABLE/DISABLEALL](#)

Syntax

PVSPLINE[*switches*]*axis_list*[*time-interval*] [*motor_motion_delay*]

Arguments

| | |
|---------------------------|------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Single axis or axis group, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>time-interval</i> | Optional argument for user-defined time interval, in milliseconds, between points. |
| motor_motion_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /r | The point coordinates are relative to the previous point |
| /w | Create the motion, but wait to start for GO |
| /t | Non-uniform time interval: time interval is specified for each point along with the point coordinates |
| /c | Perform the point sequence as a cyclic array-after the last point, start the motion again from the first point. |
| /q | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |

Comments

- > Since the time interval and the destination point are defined, variables [VEL](#), [ACC](#), [DEC](#), and [JERK](#) have no effect on **PVSPLINE**.
- > If **PVSPLINE** motion is ended with [HALT](#), the controller does not follow the motion trajectory during deceleration.
- > **PVSPLINE** specified with **/t** must **NOT** have a *time-interval* specification. Instead, specify the time interval for each point as an additional argument for [POINT](#) or as an additional array row in [MPOINT](#), see [Example 5](#).

Related ACSPL+ Commands

[POINT](#), [MPOINT](#)

COM Library Methods and .NET Library Methods

Spline, SplineM, AddPVPoint, AddPVPointM, AddPVTPoint, AddPVTPointM

C Library Functions

acsc_Spline, acsc_SplineM, acsc_AddPVPoint, acsc_AddPVPointM, acsc_AddPVTPoint, acsc_AddPVTPointM

Example

```

REAL ARRAY_NAME(4)(5) !Defines a point array of four rows and five
columns
                                !and a variable that defines the number of array rows
                                !that are part of the motion.

INT NUMBER_of_POINTS !Defines NUMBER_of_POINTS as an integer variable.
NUMBER_of_POINTS=3 !Assigns 3 to the NUMBER_of_POINTS variable.
PVSPLINE/r (0,1),10 !Initiates PVSPLINE motion for axis 0 and 1 with
!relative destination points and a time interval of
!10 milliseconds between each point.

POINT (0,1),200,100,1000,2000 !Defines values for 0 and 1 axes where:
!200, 100=destination point for the 0 and 1 axes,
!respectively; and 1000, 2000=velocity at specified
!points for the 0 and 1 axes, respectively.

MPOINT (0,1), ARRAY_NAME, NUMBER_of_POINTS
!Defines an MPOINT array for the 0 and 1 axes, where
!ARRAY_NAME is called, and where NUMBER_of_POINTS
!defines the number of array columns involved in the
!motion.

ENDS (0,1) !Ends MPOINT for 0 and 1 axes.
STOP !Ends program.
    
```

Figure 2-15 illustrates a typical PVSPLINE motion.

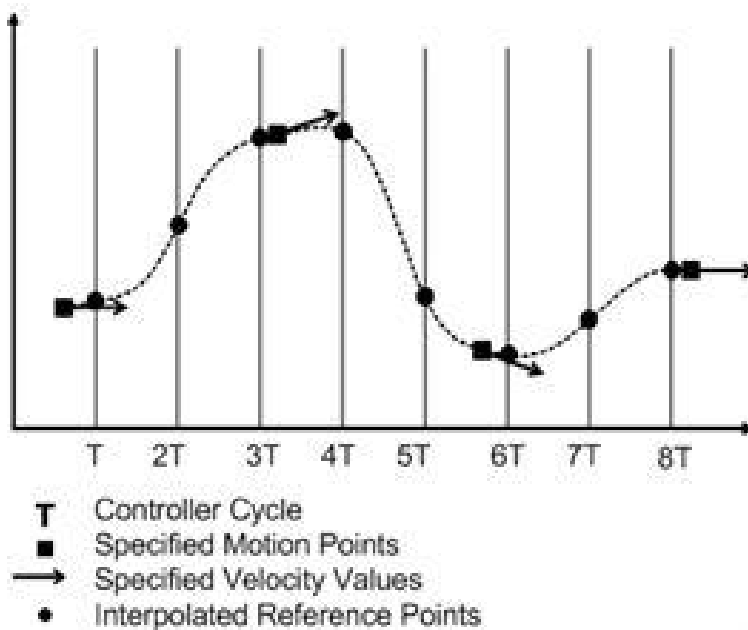


Figure 2-15. PVSPLINE Motion Diagram

2.6.23 SLAVE

Description

SLAVE initiates a motion based on position lock or velocity lock slaved to a master value or expression. Only individual axes are allowed. The initiated motion starts immediately if the axis is idle, otherwise the motion waits in the motion queue until all previously created motions finish.

SLAVE must precede **MASTER**.

MASTER - SLAVE dependency terminates with:

- > Any fault activation that disables the drive or kills the motion
- > **HALT** , **KILL/KILLALL**, **BREAK**
- > **DISABLE/DISABLEALL** to the **SLAVE** axis.
- > Setting the logical dependence between master and slave axes to zero. For example **MASTER MPOS(0)=0**

Syntax

SLAVE*[/switches]* **axis***[,lower boundary, upper boundary]*

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------|
| <i>axis</i> | Axis designation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>lower boundary</i> | Optional argument used for setting the lower boundary of master-dependent SLAVE motion. |
| <i>upper boundary</i> | Optional argument used for setting the upper boundary of master-dependent SLAVE motion. |

Switches

| | |
|-----------|------------------------------------------------------------------|
| <i>/w</i> | Create the motion, but do not start until GO is executed. |
| <i>/p</i> | Master - Slave motion generated by position lock. |
| <i>/t</i> | Create slave motion within specified position boundaries. |

Comments

- > When */p* is used, the controller first initiates **velocity lock**. Only after the achieving **velocity lock** the controller will engage **position lock**.
- > If no command option is specified, the default mode is **velocity lock**.

Related ACSPL+ Commands

MASTER, **GO**, **HALT** , **KILL/KILLALL**, **DISABLE/DISABLEALL**

Related ACSPL+ Variables

XSACC, **MFF**, **JERK**, **ACC**, **VEL APOS**

COM Library Methods and .NET Library Methods

SetMaster, Slave, SlaveStalled

C Library Functions

acsc_SetMaster, acsc_Slave, acsc_SlaveStalled

Example

See [MASTER](#) for an example of **MASTER - SLAVE** syntax.

2.6.24 STOPPER

Description

STOPPER is used in conjunction with [MSEG...ENDS](#) to avoid velocity jumps at segment inflection points. When **STOPPER** is specified between two segments, the controller provides smooth deceleration to zero before **STOPPER** and a smooth acceleration to the default or specified velocity after **STOPPER**.

Syntax

STOPPER *axis_list*

Arguments

| | |
|------------------|------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Single axis or axis group, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------------|------------------------------------------------------------------------------------------------------------|

Related ACSPL+ Commands

[MSEG...ENDS](#), [ARC1](#), [ARC2](#), [LINE](#), [PROJECTION](#)

COM Library Methods and .NET Library Methods

Stopper

C Library Functions

acsc_Stopper

Example

[Figure 2-16](#) illustrates the following example. For this illustration the reference position (**RPOS**) for axes 0 and 1 was set to (0,0).

```
MSEG (0,1), 0, 0      !MSEG initiates segmented motion for the 0 and 1 axis
                      !group with the point on the plane located at (0, 0).
LINE (0,1), 1000, 2500 !Add line segment with final point at (1000,
2500).
STOPPER (0,1)        !Slow down to zero.
ARC1 (0,1), 0,2000, -1000,2500, +
                      !Add arc segment with final point (-1000, 2500) and
                      !center point (0, 2000).
STOPPER (0,1)        !Slow down to zero.
LINE (0,1), 0, 0     !Add line segment with final point (0, 0).
ENDS (0,1)           !End MSEG.
LINE (0,1), 1000, 1000 !Add line segment with final point (1000, 1000).
```

```
ENDS (0,1) !Ends the point sequence for the 0 and 1 axis
!group.
```

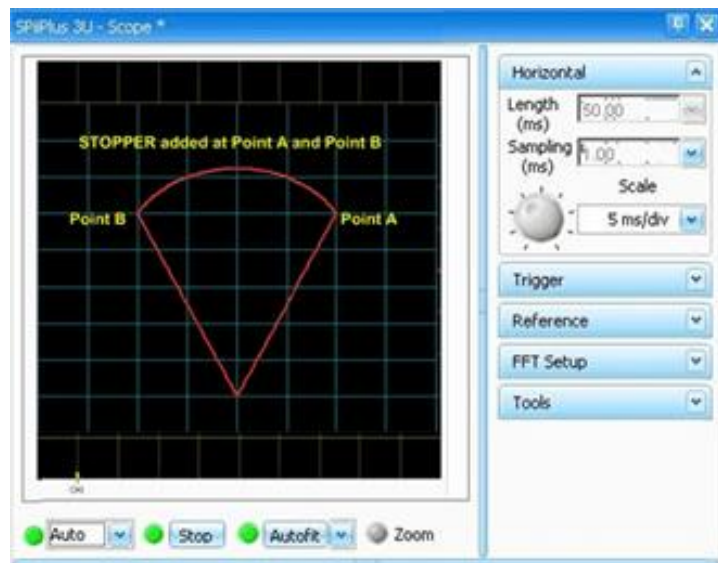


Figure 2-16. Use of STOPPER

2.6.25 TRACK

TRACK initiates track motion. In **TRACK** motion, a new point-to-point move is generated to a new target position whenever the variable **TPOS** (target position) changes. **TRACK** does not terminate automatically. If **TPOS** is not assigned a new value, motion stops at the last **TPOS** and waits. If a new **TPOS** value is assigned, motion continues.

TRACK terminates due to:

- > Any subsequent motion command (except **TRACK**) for the motion axis involved in a track motion, **except the case when the next motion is a group motion**.
- > Any fault activation that disables the drive or kills the motion.
- > User termination by **HALT**, **KILL/KILLALL**, or **DISABLE/DISABLEALL**

Syntax

TRACK [/switch] axis, [motor_motion_delay]

Arguments

| | |
|---------------------------|---------------------------------------------------------------------------------------------------|
| axis | Axis designation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| motor_motion_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switch

/switch can be:

| | |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>/w</code> | Create the motion, but do not start until GO . |
| <code>/q</code> | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |

Comments

- > While **PTP** code appears shorter and simpler, there are applications where **TRACK** is preferable to **PTP**. For example, **TRACK** provides an easy way to change the destination position at any time during the motion by changing the target position (**TPOS**) variable. The controller terminates the current motion and proceeds to the next destination point, on-the-fly.
- > **TRACK** is for single axis motion only.
- > **TPOS** is updated every controller cycle.

Related ACSPL+ Commands

[PTP](#)

COM Library Methods and .NET Library Methods

Track

C Library Functions

acsc_Track

Example

```
TRACK 0                !Initiates TRACK for 0 axis.
TPOS(0) = NewTarget    !The controller generates a PTP motion to the point
                       !designated by NewTarget.
```

2.6.26 XSEG...ENDS

Description

The **XSEG...ENDS** (Extended Segmented Motion) command block provides the following:

- > Corner detection
- > Detection of segments, where required velocity violates axis velocity/acceleration limits
- > Velocity limitation at corners and segments where required velocity violates axis velocity, acceleration and jerk limits
- > Building a velocity profile using multi-segment look-ahead algorithm
- > Corner rounding using different criteria
- > Support of up to 6 axes
- > Support for "Skywriting" - external loops at corners

Use the following commands to define the segmented motion:

- > **ARC1**- adds an arc segment to a segmented motion and specifies the coordinates of center point, coordinates of the final point, and the direction of rotation
- > **ARC2**- Adds an arc segment to a segmented motion and specifies the coordinates of center point, rotation angle and direction.
- > **LINE** - adds a linear segment to a segmented motion.
- > **ENDS**- terminates the point sequence

Syntax


XSEG

```
[/switches] (axis_list), initial_position_axis1,initial_position_axis2[,initial_position_axis3...,initial_position_axis6],
[,velocity][,end_velocity][,junction_velocity][,angle][,curvature_velocity][,deviation][,radius]
[,maximal_length]
[,starvation_margin [,segments_buffer]]
[,external_loop_type, minimal_segment_length, maximum_allowed_deviation]
[output_index, bit_number, polarity]
```

Segment commands (ARC1, ARC2, LINE)

ENDS

Arguments

| | |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_list</i> | Axis group (specified as axes numbers separated by a comma), valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.  A minimum of two axes must be specified. |
| <i>initial_position_axis1</i> | Initial position of the first axis. |
| <i>initial_position_axis2</i> | Initial position of the second axis. |
| <i>initial_position_axis3...</i> <i>initial_position_axis6</i> | Mandatory only if axis_list contains more than 2 axes. Number of initial positions must correspond to the number of axes in axis_list . |
| <i>velocity</i> | [Optional, only used with /v switch] Defines required velocity instead of default velocity (VEL). |
| <i>end_velocity</i> | [Optional, only used with /f switch] Defines required velocity at the end of each segment. |

| | |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>junction_velocity</i> | [Optional, only used with /j switch] Defines required velocity at the junction. |
| <i>angle</i> | [Optional, only used with /a switch] The junction will be treated as a corner if actual junction angle is more than defined. |
| <i>curvature_velocity</i> | [Optional, only used with switch /d] Defines required velocity at curvature discontinuity points. See Switches explanation for details. |
| <i>deviation</i> | [Optional, only used with /g switch] Defines maximal allowed trajectory deviation from the corner point. See Switches explanation for details. |
| <i>radius</i> | [Optional, only used with /u switch] Defines maximal allowed rounding radius of the additional segment. See Switches explanation for details. |
| <i>maximal_length</i> | [Optional, only used with /h switch] Defines the maximal length of the segment for smoothing processing. If the length of a segment that formed a corner exceeds the specified maximal length, the corner will not be smoothed. |
| <i>starvation_margin</i> | [Optional] Starvation margin in milliseconds. The controller sets the AST.#NEWSEGM bit starvation_margin millisecond before the starvation condition occurs. By default, if the argument is not specified, the starvation margin is 500 milliseconds. |
| <i>segments_buffer</i> | [Optional] One-dimensional user defined real array. The controller uses this array to store adding segments. By default, if the argument is not specified, the controller allocates internal buffer for storing 50 segments only. The argument allows the user application to reallocate the buffer for storing larger number of segments. The larger number of segments may be required if the application needs to add many very small segments in advanced. |
| <i>external_loop_type</i> | 0 - Cancel external loop 1 – Smooth External loop (line-arc-line) 2 – Triangle External loop (line-line-line) |
| <i>minimum_segment_length</i> | If the lengths of both segments are more than this value, the skywriting algorithm will be applied. |
| <i>maximum_allowed_deviation</i> | The parameter limits the external loop deviation from the defined profile. If the value is negative – no limitation. |
| <i>output_index</i> | Index of digital output port to assigned to synchronization (read by OUT) |

| | |
|-----------------------------|----------------------------------------------------------------------------------------------------------|
| bit_number | Bit number (0 - 16) assigned to synchronization |
| polarity | 0 or 1 - which value is considered the initial state |
| motor_movement_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

There are three types of optional switches:

- > General
- > Velocity look-ahead
- > Geometry look-ahead

The controller processes the specified switches in the following order:

1. The controller checks and applies geometry look-ahead options.
2. The controller checks and applies velocity look-ahead options.

Switches from different groups can be applied together. For example, it's possible to specify a velocity at curvature discontinuity points (switch **/d**) together with permitted deviation (switch **/g**). In this case, the controller first applies corner rounding for the trajectory and then calculates velocity profile for already processed trajectory.

Optional switches are for use only with the **XSEG** command:

| General | |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /w | Do not start until the GO command. If the switch is not specified, the motion starts immediately after the first motion segment is defined. |
| /v | Specify required velocity. The switch requires additional parameter that specifies required velocity. If the switch is not specified, the required velocity is derived from the leading axis parameters. |
| /m | Use maximum velocity under axis limits. With this switch, no required velocity should be specified. The required velocity is calculated for each segment individually on the base of segment geometry and axis velocities (VEL values) of the involved axes. |

| | |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /l | <p>Velocity limitation under axis limits.</p> <p>As opposed to the switch /m, with this switch the required velocity can be specified either for all motion (switch /v for xseg command) or as segment velocity (switch /v for line, arc1, arc2 commands). In both cases, the controller tries to achieve the required velocity by taking into account also velocity limits of all involved axes. If the required velocity cannot be achieved within the axis velocity limits, the maximal velocity within the axis velocity limits is used.</p> <p>This switch cannot be specified together with switch /m.</p> |
| /z | <p>Interpret entered coordinates according to the Local Coordinate System.</p> <p>With this switch, use 2 axes motion coordinate only (X,Y). Using 3 or more coordinates causes a runtime error.</p> |
| /q | <p>Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds.</p> <p>The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms.</p> <p>Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately.</p> |
| Velocity look-ahead | |
| /f | <p>Decelerate to the end of each segment.</p> <p>The switch requires an additional parameter that specifies end velocity. The controller decelerates to the specified velocity in the end of each segment. The specified value should be less than the required velocity; otherwise the parameter is ignored.</p> <p>If the switch is not specified, deceleration in each segment is not required. However, in specific segments deceleration might occur due to corner processing or other velocity control conditions.</p> |
| /j | <p>Decelerate to corner.</p> <p>The switch requires an additional parameter that specifies corner velocity. The controller detects corner on the path and decelerates to the specified velocity before the corner. The specified value should be less than the required velocity; otherwise the parameter is ignored.</p> <p>If switch j is not specified while switch a is specified, zero value of corner velocity is assumed.</p> <p>If switches j, a, d, and y are not specified, the controller provides automatic calculation of the corner processing.</p> |

| | |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /a | <p>Do not treat junction as a corner, if junction angle is less than or equal to the specified value in radians.</p> <p>The switch requires an additional parameter that specifies negligible angle in radians.</p> <p>If switch a is not specified while switch j is specified, the controller accepts default value of 0.01 radians that is about 0.57 degrees.</p> <p>If switches j, a, d, and y are not specified, the controller provides automatic calculation of the corner processing.</p> |
| /d | <p>Decelerate to curvature discontinuity point. The switch requires an additional parameter that specifies velocity at curvature discontinuity points. Curvature discontinuity occurs in linear-to-arc or arc-to-arc smooth junctions. If the switch is not specified, the controller does not decelerate to smooth junction disregarding curvature discontinuity in the junction. If the switch is specified, the controller detects curvature discontinuity points on the path and provides deceleration to the specified velocity. The specified value should be less than the required velocity; otherwise the parameter is ignored. The switch can be specified together with switches j and/or a.</p> <p>If switches j, a, d, and y are not specified, the controller provides automatic calculation of the corner processing.</p> |
| /y | <p>If the switch is specified the controller provides automatic calculations as described in Enhanced automatic corner and curvature discontinuity points processing (switch /y).</p> |
| /r | <p>Set the initial axis position as origin. Segment commands positions should be declared relative to the new origin.</p> |
| Geometry look-ahead | |
| /g | <p>Use a corner rounding option with the specified permitted deviation The switch requires additional parameter that specifies maximal allowed deviation of motion trajectory from the corner point. The switch cannot be specified together with switches /u and /h</p> |
| /u | <p>Use a corner rounding option with the specified permitted curvature The switch requires additional parameter that specifies maximal allowed rounding radius of the additional segment The switch cannot be specified together with switches /g and /h</p> |
| /h | <p>Use a corner smoothing option. The switch requires additional parameter that specifies the maximal length of the segment for smoothing processing. If a length of one of the segments that built a corner exceeds the specified maximal length, the corner will not be smoothed. Smoothing is only applied to pair of linear segments. If one of the</p> |

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | segments in a pair is arc, the smoothing is not applied for this corner. The switch cannot be specified together with switches /g and /u . |
| /b | Use external loops at corners. The switch requires additional parameters that specify the external loop type, the minimum segment length, and the maximum allowed deviation from profile. |
| /s | Defines output bit to support external loop synchronization. |



XSEG without switches does not require any additional parameters except the initial point coordinates, for example, **XSEG (0,1),0,0** creates segmented motion for axes 0 and 1 with initial point (0,0) with required velocity derived from the axis 0.

Comments

For each two adjacent segments, the controller calculates the tangent vector to each segment in the junction point. If the two vectors are equal, the segments are tangent, and no special processing is required. If not, the two segments build a corner. In a corner, the controller behavior follows the corner processing option selected by the user for **XSEG** motion.

The following options are supported:

Exact path: no deviation from the specified path is permitted. The user specifies two additional parameters: threshold angle and corner velocity. The controller compares the corner angle and the threshold angle. If the corner angle is smaller, the controller ignores the corner and tries to move as if the junction is smooth (the threshold angle cannot be large, otherwise passing the junction at working velocity can produce mechanical jerk). If the corner angle is greater, the controller executes deceleration to achieve the junction point with the specified corner velocity (as shown in [Figure 2-17](#)).

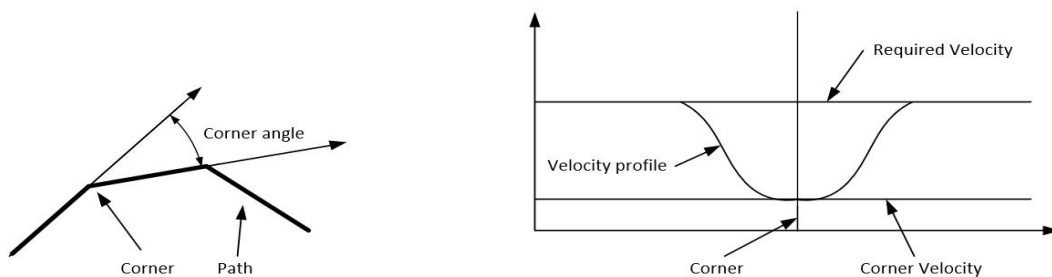


Figure 2-17. Corner Processing - Exact Path Option

Permitted deviation: the user specifies the motion trajectory maximum permitted deviation from the corner point. The controller inserts an additional segment in the corner so that the resulting path is smooth and complies with the maximum deviation.

Permitted radius: the user specifies the additional segment maximum permitted rounding radius. The controller inserts an additional segment in the corner so that the resulting path is smooth and complies with the maximum permitted radius.

Corner smoothing: the user specifies the smoothing maximum segment length. The controller applies smoothing if the length of both segments in the pair is less than the maximum segment length.

Figure 2-18 illustrates the permitted deviation, permitted radius and corner smoothing options.

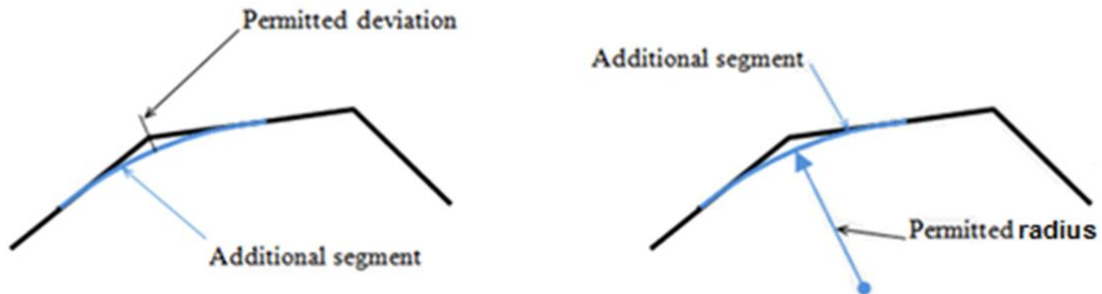


Figure 2-18. Corner Processing - Permitted Deviation, Permitted Radius and Corner Smoothing Options

XSEG updates the following motion related parameters:

- > Motion and Axis Statuses: **AST, MST**
- > Vector velocity, acceleration and jerk: **GVEL, GACC, GJERK**
- > Axis velocity: **GVEC**
- > Motion queue status and motion type: **GMQU, GMTYPE**
- > Trajectory vector distance from the beginning of the first segment: **GPATH**
- > Elapsed motion time: **GETIME**
- > Current executed segment and segments buffer status: **GSEG, GSFREE**

XSEG builds the algorithm upon the following axis motion parameters as axis constraints: **VEL, ACC,** and **JERK**.

In connection with the *segments_buffer* argument, for most applications the internal buffer size is enough and should not be enlarged.



The buffer is for the internal use of the controller only and should not be used by the user application.

The buffer size calculation rule: each segment requires about 750 bytes, so if it is necessary to allocate a buffer for 200 segments, it should be at least $750 * 200 = 150,000$ bytes. The following declaration defines a 150,000 bytes buffer:

```
real buf(18750)
```



See [XARRSIZE](#) for details of how to declare a buffer with more than 100000 elements

Examples

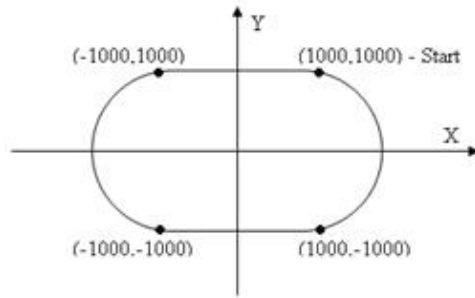
Example 1:

```
XSEG (1,0),0,0 !Segmented motion for axes 1 and 0. Required velocity
!is derived from the axis 1, i.e., VEL(1) value. No
!deviation from the path is permitted. If the path
!contains a corner, and the junction angle is more
!than default value 0.01 radians, the velocity
!decelerates to zero in the corner point.
XSEG/vf !Segmented motion for axes 0 and 1 with initial
(0,1),0,0,100,50 !point (0,0) with required velocity 100 units/sec;
!at the end of each segment, the motion should
!decelerate to 50 units/sec.
XSEG/vja !Segmented motion for axes 1 and 2 with initial
(1,2),1000,1000, !point(1000,1000) and required velocity is 100
!units/sec.
100,20,0.05 !If the path contains a junction, and the junction
!angle is more than 0.05 radians, the velocity
!decelerates to 20 unit/sec in the junction point.
```

Example 2:

```
XSEG (0,1),1000,1000 !Create segment motion in axes XY with initial point
!(1000,1000)
ARC1 (0,1),1000,0,1000,-1000,-!Add arc segment with center (1000,0),
final point(1000,1000), clockwise rotation
LINE (0,1),-1000,-1000!Add line segment with final point ( 1000, 1000)
ARC2 (0,1),-1000,0,-3.141529 !Add arc segment with center ( 1000,0)
!and rotationangle of  $-\pi$ 
LINE (0,1),1000,1000 !Add line segment with final point (1000,1000)
ENDS (0,1) !End the segment sequence
```

The **XSEG** command creates the segment motion. The motion does not start when processing reaches the **XSEG** command. Actual motion starts once the previous motion ends and one or more segments are added. The four segment commands in example 2 specify the following path:



Example 3:

```

GLOBAL INT iXaxis, iYaxis
GLOBAL REAL buf(18750) ! For 200 segments ( ARRAY SIZE = (200[SEG]*750
[BYTES/SEG])/8 )
! Make sure that XARRSIZE value is bigger than ARRAY SIZE (if not, change
XARRSIZE on the terminal)

iXaxis = 0
iYaxis = 1

XSEG (iXaxis,iYaxis),0,0,,buf
LINE (iXaxis,iYaxis),1000,1000
LINE (iXaxis,iYaxis),1001,1001
LINE (iXaxis,iYaxis),1002,1002
!..... Add more segments .....
ENDS (iXaxis,iYaxis)

STOP

```



The **LINE** command may specify one axis that actually moves in this segment. Other axes specified in **XSEG** hold their positions while the linear segment is in progress.

The **ARC1** and **ARC2** commands always specify two axes.

The **ARC1** and **ARC2** commands differ by the required arguments. The **ARC1** command specifies the coordinates of the center point, coordinates of the final point and the direction of rotation (+ for counter-clockwise, – for clockwise rotation). The **ARC2** command specifies the coordinates of the center point and rotation angle (positive for counter clockwise, negative for clockwise rotation). The **ARC1** and **ARC2** commands may produce the same result, so the user may select the one that suits the available data. If the user knows the coordinates of the center point, coordinates of the final point and the direction of rotation, **ARC1** is preferable. If the user knows the coordinates of the center point and rotation angle, **ARC2** is preferable.

The **ENDS** command informs the controller that no more segments will be specified for the motion.

2.6.27 NURBS

Description

Description

Creates **NURBS** motion.

Syntax

NURBS[/switches] (axis_list)[,velocity][,deviation][,exc_angle][,exc_length][,segments_buffer][,motor_motion_delay]

Arguments

| | |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_list | Axes involved in NURBS motion, specified as axes numbers separated by comma or as axis names separated by comma. A minimum of two axes must be specified. By default, the first three axes are the main axes, and all subsequent axes, if any, are dependent axes. |
| velocity | Optional, only used with suffix /v . Defines required feed rate. If not specified, the feed rate is derived from the leading axis parameters. |
| deviation | Optional, only used with suffix /d . Permitted deviation of trajectory from the specified control points. If not specified, the controller exactly follows the spline defined by control points, knots, and weights. If the parameter is specified, but the defined spline deviates more than allowed, the controller tries to modify spline geometry in order to comply with the permitted deviation. |
| exc_angle | Optional, only used with suffix /a . The value defines exceptions from spline interpolation. If for an internal control point directions to the previous and the next control points require direction change more than the specified angle (by modulo), the control point is processed as a corner. Actually, such points divide the spline into two independent splines. |
| exc_length | Optional, only used with suffix /l . The value defines exceptions from spline interpolation. If a distance between two control points appears longer than the specified length, the trajectory between the points is considered straight. Actually, two independent splines are built before and after the segment. |
| segments_buffer | [Optional] One-dimensional user-defined real array. The controller uses this array to store added segments. By default, if the argument is not specified, the controller allocates internal buffer for storing 50 segments only. The argument allows the user application to reallocate the buffer for storing larger number of segments. The larger number of segments may be required if the spline is defined with a large number of closely specified control points; the case is typical when NURBS is used for processing |

| | |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| | minute segments prepared with a CAD system. The buffer is for the controller's internal use only and should not be used by the user application. |
| motor_motion_delay | (Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts. |

Switches

| | |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| w | Do not start until the go command. If the switch is not specified, the motion starts once the first three spline points are specified. |
| v | Specify required velocity. The switch requires an additional parameter that specifies required velocity. If the switch is not specified, the feed rate is derived from the leading axis' parameters. |
| d | Specify permitted deviation. The switch requires an additional parameter that specifies permitted deviation of trajectory from the specified control points. If the switch is not specified, the controller exactly follows the spline defined by control points, knots, and weights. If the parameter is specified but the defined spline deviates more than allowed, the controller tries to modify spline geometry in order to comply with the permitted deviation. |
| a | Specify exception angle. The switch requires an additional parameter that specifies the maximum angle in a control point. The value defines exceptions from spline interpolation. If defined for an internal control point, the directions to the previous and the next control points require direction change more than the specified angle (by modulo) the control point is processed as a corner. Actually, such a point divides the spline into two independent splines. |
| l | Specify exception length. The switch requires an additional parameter that specifies the maximum segment length. The value defines exceptions from spline interpolation. If the distance between two control points appears longer than the specified length, the trajectory between the points is considered straight. Actually, two independent splines are built before and after the segment. |
| q | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |

Return Value

None

Example 1

```

NURBS/v(0,1), 100

NPOINT (0,1), 4.0, -6.0 ! the first point must be the motion
                        ! starting position
NPOINT (0,1), -4.0, 1.0
NPOINT (0,1), -1.5, 5.0
NPOINT (0,1), 0.0, 2.0
NPOINT (0,1), 1.5, 5.0
NPOINT (0,1), 4.0, 1.0
NPOINT (0,1), -4.0, -6.0

ENDS (0,1)

```

Example 2

```

! Start an SPATH motion from current position
! velocity = 1000, go to exact corner if angle > 1RAD
NURBS/AV(0,1), 1000 , 1

NPOINT (0,1), 0, 0 ! the first point must be the motion starting
position
NPOINT (0,1), 50, 40
NPOINT (0,1), 100, 120
NPOINT/f (0,1), 120, 130 , 500 ! velocity for segment = 500
NPOINT/c (0,1), 180, 130 ! come to an exact stop at this point, start
! a new spline after this point

NPOINT/v (0,1), 200, 120, 2000 ! change motion vel to 2000
NPOINT (0,1), 220, 80 ! will be treated as a corner because of
! angle between points

NPOINT (0,1), 200, 60
NPOINT (0,1), 0, 0

ENDS (0,1)
STOP

```

2.6.28 NPOINT**Description**

Add next control point and knot

Syntax

NPOINT[/switches](axis_list),coordinates[,velocity][,knot][,weight][,required_velocity]

Arguments

| | |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_list | The list of axes must be the same as in the corresponding NURBS command. |
| coordinates | The list of coordinate values separated by commas. The list must specify one value for each axis in axis_list. The list defines coordinates of one control point of the spline. |
| velocity | Optional, only used with switch /v or /f. The value changes the required feed rate. The new value is valid for all spline segments after the corresponding control point. |
| knot (Not supported in ALPHA) | Optional, only used with switch /k. The value specifies a delta for the new knot calculation. The delta must be positive or zero. Each npoint command adds a new knot to the knot vector. The new knot is calculated as a previous knot plus delta (positive or zero). If suffix k is not specified, the delta default value is one; a new knot is calculated as the previous knot plus one. A special case occurs in the very first npoint command, if knots command was not specified, so that the previous knot does not exist. In this case, the knot parameter specifies the absolute value of the first knot; if switch /k is omitted, a zero value is assumed as the first knot. If none of the npoint commands specifies a /k switch and knots commands are omitted, the controller builds a uniform spline with evenly spread knots (except for the duplicated knots at the beginning and end). |
| Weight (Not supported in ALPHA) | Optional, only used with /w switch. The value specifies control point weight. Only positive weights are accepted. If a /w switch is not specified, the default value of weight is one. If none of the point commands specifies a /w switch, all control points have the same weight (one); in this case, the spline is actually a non-rational B spline. |
| Required_velocity | Optional, only with /f switch The required velocity for this segment only |

Switches

| | |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /v | Specify new required velocity. The switch is not compatible with /f. The switch requires an additional parameter that specifies the required velocity. The value is used as the required velocity for the current and all subsequent points. |
| /f | Specify required velocity. |

| | |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <p>The switch is not compatible with /v.</p> <p>The switch requires an additional parameter that specifies required velocity.</p> <p>The value is used as the required velocity for the current point, but does not change the required velocity for subsequent points.</p> |
| /k (not supported in Alpha version) | <p>Specify knot delta.</p> <p>The switch requires an additional parameter that specifies the knot delta from the previous knot.</p> <p>The new knot is calculated as the previous knot plus delta.</p> |
| /w (not supported in Alpha version) | <p>Specify control point weight.</p> <p>The switch requires an additional parameter that specifies the weight of the control point.</p> |
| /d | <p>Mark the point specification as dummy.</p> <p>Dummy point specifications can either precede the first control point specification, or follow the last control point specification. Dummy points specification is required in rare cases where default calculation of starting/trailing knots is not appropriate.</p> |
| /c | <p>Mark the current point as a corner.</p> <p>The control point is processed as a corner. Actually, such a point divides the spline into two independent splines.</p> |

Return Value

None

2.6.29 SPATH**Description**

Initiate a path smoothing motion.

Syntax

SPATH [/switches] (axis_list),coordinates[,velocity][,exc_angle][,exc_length][,segments_buffer][,motor_motion_delay]

Arguments

| | |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_list | <p>Axes involved in the path smoothing motion, specified as axes numbers separated by comma or as axes' names separated by comma. A minimum of two axes must be specified. By default, the first three axes are main axes, and all subsequent axes (if any) are dependent axes.</p> |
| velocity | <p>Optional, only used with /v switch</p> <p>Defines required feed rate. If not specified, the feed rate is derived from the leading axis parameters.</p> |

| | |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| exc_angle | <p>Optional, only used with /a switch</p> <p>The value defines exceptions from spline interpolation. If for an internal control point directions to the previous and the next control points require direction change more than the specified angle (by modulo), the control point is processed as a corner. Actually, such point divides the spline into two independent splines.</p> |
| exc_length | <p>Optional, only used with /l switch.</p> <p>The value defines exceptions from spline interpolation. If a distance between two control points appears longer than the specified length, the trajectory between the points is considered straight. Actually, two independent splines are built before and after the segment.</p> |
| segments_buffer | <p>[Optional] One-dimensional user-defined real array.</p> <p>The controller uses this array to store added segments. By default, if the argument is not specified, the controller allocates internal buffer for storing 50 segments only. The argument allows the user application to reallocate the buffer for storing a larger number of segments. The larger number of segments may be required if the spline is defined with a large number of closely specified control points; the case is typical when path smoothing is used to process minute segment prepared with a CAD system.</p> <p>The buffer is for the controller's internal use only and shouldn't be used by the user application</p> |
| motor_motion_delay | <p>(Optional, used only with /q switch) Delay, in milliseconds, before motor motion actually starts.</p> |

Switches

| | |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| w | <p>Do not start until the go command.</p> <p>If the switch is not specified, the motion starts once the first three spline points are specified.</p> |
| v | <p>Specify required velocity.</p> <p>The switch requires an additional parameter that specifies the required velocity. If the switch is not specified, the feed rate is derived from the leading axis parameters.</p> |
| a | <p>Specify exception angle.</p> <p>The switch requires an additional parameter that specifies maximum angle in a control point. The value defines exceptions from spline interpolation. If for an internal control point, directions to the previous and the next control points require direction change more than the specified angle (by modulo), the control point is processed as a corner. Actually, such point divides the spline into two independent splines.</p> |

| | |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| I | Specify exception length. The switch requires an additional parameter that specifies maximum segment length. The value defines exceptions from spline interpolation. If a distance between two control points appears longer than the specified length, the trajectory between the points is considered straight. Actually, two independent splines are built before and after the segment. |
| G | Acceleration consideration. Allow the motion generator to deviate from the specified axes acceleration parameter during velocity profile generation. |
| q | Defines actual motor movement delay in microseconds. The delay resolution is 50 microseconds. The maximum delay is 100 controller cycles: 100ms for CTIME=1ms or 20ms for CTIME=0.2ms. Allows delaying actual motor movement start (RPOS) for the specified delay, while motion profile generation (APOS) starts immediately. |

Return Value

None

Comments

This command is supported in version 3.10 and higher.

Example

```

! Start an SPATH motion from current position (0,0),
! velocity = 1000, wait for go command before starting motion
SPATH/VW(0,1), 0, 0, 1000

SEGMENT (0,1), 50, 0
SEGMENT (0,1), 100, 0
SEGMENT/f (0,1), 100, 100 , 500 ! velocity for segment = 500
SEGMENT/c (0,1), 200, 200 ! come to an exact stop at this point,
! start a new spline after this point

SEGMENT/v (0,1), 300, 200, 2000 ! change motion vel to 2000
SEGMENT (0,1), 300, 300
SEGMENT (0,1), 400, 300
SEGMENT (0,1), 400, 400

GO (0,1) !start motion

ENDS (0,1)
STOP

```

2.6.30 SEGMENT

Description

Add a new control point to the **SPATH** motion generator

Syntax

SEGMENT[/switches](axis_list),coordinates[,velocity][,required_velocity]

Arguments

| | |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_list | The list of axes must be the same as in the corresponding SPATH command. |
| coordinates | The list of coordinate values, separated by commas The list must specify one value for each axis in axis_list . The list defines coordinates of one control point of the spline. |
| velocity | Optional, only used with /v switch. The value changes required feed rate. The new value is valid for all spline segments after the corresponding control point. |
| required_velocity | Optional, only with the /f switch The required velocity for this segment only. |

Switches

| | |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /v | Specify new required velocity The switch is not compatible with /f . The switch requires an additional parameter that specifies required velocity. The value defines the required velocity in the current and all subsequent points. |
| /f | Specify required velocity The switch is not compatible with /v . The switch requires an additional parameter that specifies required velocity. The value defines the required velocity at the current point, but does not change the required velocity for subsequent points. |
| /c | Mark the current point as a corner The control point is processed as a corner. Actually, such point divides the spline into two independent splines. |

Return Value

None

Comments

This command is supported in version 3.10 and higher.

2.6.31 SMOVE

The **SMOVE** command provides for positioning to specific target. **SMOVE** commands work in sequence and the next **SMOVE** command changes the previous target and provide a smooth

transition from one motion direction to another, based on **ACC**, **DEC** and **JERK** values. The motion profile is optimized to pass on a rounded path near the breaking point, minimizing changes in speed and direction that would cause unwanted vibrations in the system.

Syntax

```
SMOVE[/switch] axis_list, target_point[, velocity]
```

Arguments

| | |
|--------------|-------------------------------------------------------------------|
| axis_list | Motion axes. 2 or 3 Cartesian axes can participate in the motion. |
| Target_point | Destination point coordinates |
| velocity | Optional argument for user-defined velocity |

All **SMOVE** commands in the sequence must have the same axis_list parameter.

Switches

| | |
|----|-------------------------------------------------------------------|
| /v | Use the specified velocity instead of the default velocity (VEL). |
| /z | Move relative to Local Coordinate System |

Example

```
SMOVE (X,Y,Z), 100, 100, 300
```

2.6.32 Using ARC1, ARC2 and LINE Switches

The following optional switches may be used singularly or in combination with **ARC1**, **ARC2** and **LINE**:

| | |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /f | <p>Decelerate to the end of segment.</p> <p>The switch requires an additional parameter that specifies the end velocity. The controller decelerates to the specified velocity at the end of segment. The specified value should be less than the required velocity; otherwise the parameter is ignored. The switch affects only one segment.</p> <p>The switch also disables corner detection and processing at the end of segment.</p> <p>If the switch is not specified, deceleration is not required. However, in special cases the deceleration might occur due to corner processing or other velocity control conditions.</p> |
| /v | <p>Specify required velocity.</p> <p>The switch requires an additional parameter that specifies the required velocity. The switch changes the required velocity for the current segment and for all subsequent segments.</p> <p>If the switch is not specified, the required velocity does not change.</p> |

| | |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /o | Synchronize user variables with segment execution. The switch requires additional two or three parameters that specify values , user variable and mask . |
| /t | Specify segment processing time The switch requires an additional parameter that specifies the segment processing time in milliseconds. Unlike the required velocity specification, the segment processing time defines velocity at the current segment only, and has no effect on the subsequent segments. The switch cannot be specified together with /V . |
| /b | Use external loops at corners. The switch requires additional parameters that specify the external loop type, the minimum segment length, and the maximum allowed deviation from profile. The /b switch may be defined with other corner processing options (/u, /g, etc.) . If the Skywriting algorithm is applied, other corner processing options are skipped. If Skywriting is skipped, other defined corner options will be applied. This switched is used for Extended Motion only (initialized with XSEG...ENDS) |
| /p | Specifies that the lci_segment_active parameter is required. This switched is used for Extended Motion only (initialized with XSEG...ENDS) |

For **ARC1**, **ARC2**, and **LINE** some switches require an additional parameter to be specified. If more than one parameter is required, the parameters should be separated by a comma, and the order of parameters is fixed in the following order:

1. Required velocity (used with **/V**)
2. Final velocity (used with **/F**)
3. Segment processing time (used with **/T**)
4. **/O** requires specification of the `values`, `variables`, and `mask` parameters
5. **/B** requires specification of the External Loop Type, Minimum Segment Length, and Maximum Allowed Deviation parameters
6. **/P** requires specification of the `lci_segment_active` parameter

Examples:

| | |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| LINE/v (1,0), 1000, -1000, 500 | Add line segment with end point (1000, -1000) and segment velocity 500. |
| arc1/vf (0,1), 0, 0, 100, 100, +, 500, 100 | Add arc segment with center (0,0), end point (100,100), clockwise direction, segment velocity 500 and end velocity 100 |

| | |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| int Value(1) int Mask(1) Value(0) = 1; Mask (0) = 5 ARC2/o (0,1), 0, 0, 3.141529, Value, OUT, 2, Mask | Add arc segment with center (0,0) and 180 degree (π) angle. At the beginning of the segment execution, sets bit 0 and reset bit 2 of digital outputs OUT(2). |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|

2.7 Program Flow Commands

| Command | Description |
|------------------------|------------------------------------------------------------------------------------------|
| Assignment Command | Assigns values |
| BLOCK...END | Executes a group of commands in one MPU cycle |
| CALL | Calls subroutine. |
| GOTO | Transfers program execution to another point in the program. |
| IF, ELSEIF, ELSE...END | IF command structure. |
| INPUT | Suspends program execution pending user input |
| LOOP...END | Loop command structure. |
| ON...RET | Defines an autoroutine |
| TILL | Delays program execution until a specified expression produces a non-zero (true) result. |
| WAIT | Delays program execution for a specified number of milliseconds. |
| WHILE...END | While command structure. |

2.7.1 Assignment Command

Description

The **Assignment** command (**=**) is used to assign a value to ACSPL+ standard or user-defined variables with read/write options.

Syntax

variable_name = value

Arguments

| | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| variable_name | Can be: <ul style="list-style-type: none"> > Name of ACSPL+ standard or user-defined variable > An element of an ACSPL+ or user array > One bit of integer variable or integer array element (applicable only to those variables having specifically named bits, for example, IMASK) |
| value | Can be of integer or real type. The value argument can either be: <ul style="list-style-type: none"> > A specific value > An expression that during runtime calculates to a value |

Comments

Assigning to an ACSPL+ variable is limited by the following rules:

- > Assignment to read-only variable (for example, **FPOS**) is prohibited
- > Assignment to a protected variable (for example, **ERRI**) is allowed in only in the Configuration mode.

After assignment, the previous value of the variable is replaced by the new value.

User local and global variables must be declared before they can be used in an assignment command.

- > Explicit indexing only is allowed for user array variables.
- > If a user variable is scalar, no indexing is required.
- > If a user variable is one-dimensional array, it requires one index. Two-dimensional arrays require two indexes.

A bit can have only two possible values: 0 (false) or 1 (true), while the **value** result, which defines the bit value, can be any value. Assignments convert the value as follows:

- > If the value is zero, the bit is set to zero
- > If the value is non-zero, the bit is set to one

Although bit assignments are applicable to any integer variable or array element, they are mainly used for changing flag variables and output bits.

The controller executes assignment commands in the following order:

1. Calculate **value**
2. Convert the type of calculated value to the type of **variable_name** (if the types differ)
3. Assign the result to **variable_name**

Examples

```
VEL(0) = 1000           !Assign 1000 to axis 0 default velocity -
                        !explicit indexing.
VEL0 = 1000            !Assign 1000 to axis 0 default velocity -
```

```

Var1 = FPOS(0)           !postfix indexing.
Var2(0)(5) = 200        !Assign value of ACSPL+ variable to user variable.
OUT0.5 = 1              !Assign to element of user array.
                        !Assign to digital output 5

```

2.7.2 BLOCK...END

Description

Commands specified within the **BLOCK...END** structure are executed in one MPU cycle.

Syntax

BLOCK

command-list

END

Arguments

command-list

List of commands, separated by semi-colons.

Comments

- > The structure provides an alternative to specifying *command-list* commands in one line. The commands within the structure can be specified in several lines. However, the controller executes all commands in one controller cycle, as if they were written in one line.
- > Commands and functions that may cause delay ([WAIT](#), [TILL](#), [GETSP](#), [WHILE...END](#), [LOOP...END](#) etc.) provide delay even if they are used within the **BLOCK...END** structure.

Example

```

INT XX,YY,ZZ           !Defines variables as integers
BLOCK                  !BLOCK command
    XX=TIME            !XX is assigned the TIME standard variable
    YY=TIME            !(controller timer from power-up, in mSec.
                        !One execution line later, YY is also
                        !assigned the TIME standard variable.
                        !Because both lines were written in the
                        !BLOCK...END block, they are both executed
                        !during the same controller cycle hence the
                        !difference between the two values =0
                        !(ZZ=0).
                        !If both lines were NOT written within the
                        !BLOCK structure, the difference would be
                        !one controller cycle (ZZ=1).
END                    !End the BLOCK command
ZZ=YY-XX              !The result in this example is ZZ=0.

```

2.7.3 CALL

Description

CALL calls a subroutine according to a specified label. All subroutines must begin with a label and conclude with **RET**.

Syntax

CALL *label*

...

STOP

label:

...

RET

Arguments

| | |
|--------------|-------------------------|
| <i>label</i> | Unique name identifier. |
|--------------|-------------------------|

Related ACSPL+ Commands

[GOTO](#)

Comments

- > A label specified by **CALL** must be defined somewhere in the same program buffer. The subroutine starts after the label and spans all commands up to **RET**.
- > **CALL** transfers program execution to the start of the subroutine and stores the return point in the stack. When **RET** executes, the return point is extracted from the stack and execution continues from the next command after **CALL**.
- > Subroutines can be nested; a subroutine can call another subroutine, and so on.
- > **CALL** can only call a sub-routine located in the same buffer

Example

```
! ----- Start subroutine labeled CHECK_PE -----
CALL CHECK_PE           !Call a subroutine named CHECK_PE.
DISABLE 0               !This line will be executed when the
                        !subroutine is terminated by the RET command.

STOP                   !Ends program.
CHECK_PE:              !Subroutine label name.
WHILE (ABS(PE(0))>20)  !Subroutine command lines.
DISP "PE is :", PE(0)
END                     !End WHILE loop.
RET                    !Terminate subroutine and return to main program.
```

2.7.4 GOTO

Description

GOTO transfers program execution to a particular point in the program specified by a unique label.



Avoid using **GOTO** to enter or exit a subroutine. Failure to do so will result in program termination due to a stack violation error. As an alternative see [CALL](#).

Syntax

GOTO *label*

Arguments

label

A unique label defined in the program buffer.

Comments

- > A label specified by **GOTO** must be defined somewhere in the same program buffer.
- > The next executed command is located after the label.

Related ACSPL+ Commands

[CALL](#)

Example

```
GOTO ARR                !Go to a label named ARR
```

2.7.5 IF, ELSEIF, ELSE...END

Description

IF, **ELSEIF**, **ELSE**, and **END** are building blocks used in the **IF** control structures. These commands specify a condition which must be met before executing a list of commands. **IF** control structures must conclude with **END**.

If the condition result is none-zero (true), the command list in the corresponding clause executes and all subsequent conditions are not validated and all other command lists are skipped. After executing the command list, the program continues from the next line after **END**.

If a condition result is zero (false), the command list is skipped. The program then checks subsequent **ELSEIF** conditions. If no condition result is true, the command list following **ELSE** executes, if it exists. The program then continues from the command following **END**.

[Table 2-9](#) describes the **IF** control structure syntax using these building blocks.

Table 2-9. IF Control Structures

Syntax

| Syntax | IF Structure |
|------------------------------------------------------------------------------|---------------|
| IF <i>expression</i> command list END | IF-END |
| IF <i>expression</i> command list ELSE command list END | IF-ELSE-END |
| IF <i>expression</i> command list ELSEIF sequence END | IF-ELSEIF-END |

| Syntax | IF Structure |
|--------------------------------------------------------------------------------------------------------------------------|--------------------|
| IF <i>expression</i> command list ELSEIF <i>sequence</i> ELSE <i>command list</i> END | IF-ELSEIF-ELSE-END |

Comments

- > The **IF -ELSE -END** control structure has two command lists. One follows **IF** and the second follows **ELSE**. Only one command list executes, depending on the result of the expression following **IF**.
- > The **IF-ELSEIF-END**, and **IF-ELSEIF- ELSE -END** forms provide validation of a sequence of conditions.
- > **IF** control structures may contain any number of **ELSEIF** clauses. Each **ELSEIF** clause specifies its own condition. The conditions are validated in the following order:
 - > Condition after **IF**
 - > Condition after first **ELSEIF**
 - > Condition after second **ELSEIF**

Examples

Example 1:

This example program fragment activates either output 5 or 6 and shuts off the other, depending on the state of input 3.

```

IF IN0.3           !Check if the third bit of IN0 is 1.
    OUT0.5 = 1     !Set the 5th bit of OUT0 to 1.
    OUT0.6 = 0     !Set the 6th bit of OUT0 to 0.
ELSE              !Execute the next commands only if the IF
                  !condition was false
    OUT0.5 = 0     !Set the 5th bit of OUT0 to 0.
    OUT0.6 = 1     !Set the 6th bit of OUT0 to 1.
END               !End of the IF body.
STOP              !Ends program

```

Example 2:

The following program fragment implements a saturation effect by limiting variable **V0** to a range from -256 to +256.

```

IF V0 < -256      !Check if V0 is less than -256
    V0 = -256     !Assign -256 to V0 (low saturation)
ELSEIF V0 > 256   !If V0 is not less than -256, check if it
                  !is greater than 256
    V0 = 256      !Assign 256 to V0 (high saturation)
END              !End of the IF body (no saturation if
                  !V0 is equal to or more than 256)
STOP              !Ends program

```

2.7.6 INPUT

Description

INPUT suspends program execution in a specific buffer until the host or the user (via the **Terminal** option) provides a specific value or values.

Syntax

INPUT(*array*)

Arguments

array

User defined array.

Comments

- > The INPUT command accepts arguments as an array variable. The command delays program execution until the host provides an input. The input from the host is expected to consist of one or more numbers separated by spaces or commas and followed by a carriage return. The INPUT command scans the numbers in the input and stores them in the argument variable.
- > If the argument variable is scalar, INPUT takes the first number in the input and stores it in the argument variable.
- > If the argument is an array, INPUT fills the array argument with the values from the input.
- > The function execution is completed when the scalar is stored or the array is filled or a carriage return is encountered.

COM Library Methods and .NET Library Methods

SuspendBuffer

C Library Functions

acsc_SuspendBuffer

Related ACSPL+ Commands

None

Examples

Example 1:

```
INT ARR(10)      !Declares a ten-element array.
...             !Other program commands.
INPUT (ARR)     !Wait for user to enter values in ARR array.
```

Program execution halts until input is received from the host.

Assume that a user enters: 20 30 40 <ENTER>, the program stores the three values in ARR(0), ARR(1), and ARR(2) and resumes execution. Since the user has entered only three numbers, the program will store 0 (zero) as the value in the elements: ARR(3) through ARR(9).



The user may enter the value of a keyboard function key, 0 (zero) followed by an integer from 1 to 12, in which case the value is stored separately in the standard variable: **FK**.

Example 2:

```
INT ARR(2)           !Declares a two-element array.
DISP "[FILL AXIS NUMBER],[FILL DISTANCE TO GO]"
                   !Display the text in the Terminal
INPUT (ARR)         !This command stops the program until the two
                   !variables are entered into the ARR array.
SET FPOS(ARR(0))=0  !Set the feedback position to zero.
ENABLE (ARR(0))     !Enable the drive specified by the value of ARR(0).
PTP/RE (ARR(0)), (ARR(1))
                   !Move the ARR(0) axis to position ARR(1)
```

2.7.7 LOOP...END

Description

The **LOOP** command structure provides a fixed number of command list repetitions set by the definition of an exact value, or any expression. **LOOP** control structures must conclude with **END**.

Syntax

LOOP *expression*

command_list

END

Arguments

| | |
|---------------------|------------------------------------------------------------------------|
| <i>expression</i> | Defines the number of loops. It can also be a specific integer number. |
| <i>command_list</i> | Any set of commands that are to be repeated. |

Comments

- > If the *expression* result is zero or negative, the command list is not executed and the program continues from the next line after **END**.
- > If the *expression* result is not an integer the command rounds-off the number to the closest integer.

Examples

Example 1:

```
LOOP 10           !Do 10 repetitions.
J=J+1            !J is increased by 1 for each repetition.
END              !End LOOP.
STOP            !Ends program.
```

Example 2:

```
LOOP ARR           !Do repetitions equal to value labeled by ARR.
J=J+1             !J is increased by 1 for each repetition.
END               !End LOOP.
STOP              !Ends program.
```

Example 3:

```
LOOP YY*325/TT    !Do repetitions equal to the value of the
                  !expression YY*325/TT
J=J+1             !J is increased by 1 for each repetition.
END               !End LOOP.
STOP              !Ends program.
```

2.7.8 ON...RET

Description

ON...RET defines an autoroutine. An autoroutine consists of a condition, and a body. Autoroutines must conclude with **RET**. The autoroutine condition is checked every MPU cycle. Once the autoroutine condition is met, the autoroutine interrupts, executes the lines in the body (until **RET**), and then transfers execution control back to the interrupted program line.



The controller should never directly execute **ON**. If the program execution flow comes to **ON**, the controller asserts a runtime error and aborts the program. To avoid this error, use **ON** after **STOP/STOPALL**.

Syntax

ON condition

```
auto_routine body
```

RET

Comments

- > Once the buffer in which an autoroutine is compiled, that autoroutine is enabled.
- > The controller implements edge-detection in autoroutine condition verification. If a condition becomes true, the controller activates the autoroutine only once. If afterwards the condition remains true, the controller does not activate the autoroutine again. The condition must become false and then become true again in order to activate the autoroutine again.
- > Only one autoroutine can be active in a buffer at a time.

Related ACSPL+ Commands

[ENABLEON](#), [DISABLEON](#)

Examples

Example 1:

Demonstrates a typical use of an autoroutine for processing controller faults. The autoroutine provides an error message when a drive alarm on 0 axis occurs.

```

STOP                                !STOP must precede the autoroutine.
ON FAULT(0).#DRIVE                  !Activate autoroutine when bit
                                     !FAULT(0).#DRIVE changes from 0 to 1
DISABLE 0                            !Disable drive X
DISP "0 Axis Drive Alarm"           !Display message
RET                                  !End of autoroutine

```

Example 2:

Illustrates how all variables, not only faults, can be used in autoroutine conditions. Assuming that output OUT0.6 is connected to a LED indicator, the following autoroutine signals the motion state bit to activate the indicator, and deactivate it when the 0 axis is no longer in motion.

```

STOP                                !STOP must precede the autoroutine.
ON MST(0).#MOVE                     !When MST(0).#MOVE bit changes from 0 to
                                     !1 (signaling that the axis is moving)
OUT0.6 = 1                           !Set output 6 to 1 (turn on the LED)
RET                                  !End of autoroutine
ON ^MST(0).#MOVE                    !When MST(0).#MOVE bit changes from 1 to
                                     !0 (signaling that the axis is no longer
                                     !moving)
OUT0.6 = 0                           !Set output 6 to 0 (turn off the LED)
RET                                  !End of autoroutine

```

2.7.9 TILL

Description

TILL delays program execution until the result of a specified expression is met, or the return value is non-zero.

Syntax

TILL *expression* [*time_out*]

Arguments

| | |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>expression</i> | Defines how long to pause the program execution |
| <i>time_out</i> | Used to specify a time-out in milliseconds. Use <i>time_out</i> to limit the time that an expression can return zero. |

Related ACSPL+ Commands

[WAIT](#)

Examples

Example 1:

```
TILL ^MST(0).#MOVE      !Wait for 0 axis termination
```

Example 2:

```
TILL IN0.0=1, 2000      !Wait until input #0 =1 or 2000 milliseconds
```

Example 3

```
TILL RPOS(0)<>0      !Wait until RPOS(0) is not equal to 0
```

2.7.10 WAIT

Description

WAIT delays program execution for the specified number of milliseconds.

Syntax

WAIT *wait_time*

Arguments

wait_time

Defines how long to delay the program execution. *wait_time* can also be defined by an expression.

Related ACSPL+ Commands

[TILL](#)

Examples

Example 1:

```
WAIT 2000      !wait 2000 milliseconds
```

Example 2:

```
WAIT YY=65/TT      !Milliseconds defined by an expression
```

2.7.11 WHILE...END

Description

The **WHILE** command structure provides repetitive execution of commands list as long as a condition is satisfied. If the condition was not satisfied when checked for the first time, program execution continues from the command following **END**.

WHILE control structures must conclude with **END**.

Syntax

WHILE *condition*

command_list

END

Arguments

| | |
|---------------------|------------------------------------------------------------------------|
| <i>condition</i> | Condition controlling the execution of the WHILE <i>command_list</i> . |
| <i>command_list</i> | List of commands to be executed. |

Related ACSPL+ Commands

[IF, ELSEIF, ELSE...END, TILL](#)

Examples

Example 1:

```
WHILE S_ST.#DC          !Indicate data collection active.
  OUT0.0=^OUT0.0        !Blink LED.
  WAIT 200              !Blink period is 200msec.
END
```

Example 2:

```
WHILE (ABS(PE(0)>20))   !Do if the 0 axis Position Error is greater
                      !than 20.
  DISP "PE is :", PE(0)
  Display the 0 axis Position Error
END
```

Example 3:

```
WHILE 1                !Run forever
  PTP/e 0, 1000
  PTP/e 0, -1000
END
```

2.8 Program Management Commands

The Program Management commands are:

| Command | Description |
|---------------------------|----------------------------------------------|
| DISABLEON | Disables autoroutine activation in a buffer. |
| ENABLEON | Enables autoroutine activation in a buffer. |
| PAUSE | Suspends program execution in a buffer. |
| RESUME | Resumes program execution in a buffer. |
| START | Activates program execution in a buffer. |

| Command | Description |
|------------------------------|-----------------------------------------------------------------------------------|
| STOP/STOPALL | Terminates program execution in a buffer. STOPALL terminates all programs. |

2.8.1 *DISABLEON*

Description

DISABLEON disables autoroutine activation in a buffer. The command has the same functionality as [PFLAGS.#NOAUTO=1](#).

Syntax

DISABLEON [(*buffer-number*)]

Arguments

| | |
|----------------------|--------------------------------------------------------------------------------------------------------|
| <i>buffer-number</i> | Optional argument, if no buffer number is mentioned, the command disables autoroutines in all buffers. |
|----------------------|--------------------------------------------------------------------------------------------------------|

Related ACSPL+ Commands

[ON...RET](#), [ENABLEON](#)

2.8.2 *ENABLEON*

Description

ENABLEON enables autoroutine activation in a buffer. The command has the same functionality as [PFLAGS.#NOAUTO=0](#).

Syntax

ENABLEON [(*buffer-number*)]

Arguments

| | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer-number</i> | A number between 0 and 63 specifying the buffer in which the autoroutine is to be enabled.. If no buffer number is given, the command enables autoroutines in all buffers. |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Related ACSPL+ Commands

[ON...RET](#), [DISABLEON](#)

2.8.3 *PAUSE*

Description

PAUSE suspends program execution in a specific buffer.

Syntax

PAUSE *buffer-number*

Arguments

| | |
|----------------------|------------------------------------------------------------------------------------------|
| <i>buffer-number</i> | A number between 0 and 63 specifying the buffer in which the program is to be suspended. |
|----------------------|------------------------------------------------------------------------------------------|

Related ACSPL+ Commands

[RESUME](#)

Example

```
PAUSE 5          !Pauses buffer five
```

2.8.4 RESUME

Description

RESUME resumes the program execution in a specific buffer after the execution was paused.

Syntax

RESUME *buffer-number*

Arguments

| | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer-number</i> | A number between 0 and 63 specifying the buffer in which the program execution that was suspended by the PAUSE command is to resumed. |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

Related ACSPL+ Commands

[PAUSE](#)

Example

```
RESUME 5          !Resume buffer five
```

2.8.5 START

Description

START activates program execution at a specified line number, or a specific label in a specified buffer that is different from the buffer where **START** is enabled.

Syntax

START[/s] *buffer-number, line-number | label*

Arguments

| | |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>buffer-number</i> | A number between 0 and 63 specifying the buffer. |
| <i>line-number</i> <i>label</i> | The line number of the program within the buffer where execution is to begin, or a label identifying the line number. |

Switches

| Switch | Comments |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /s | The "/s" switch causes the program to execute from the beginning in simulation mode up to the indicated line or label, after which execution continues normally. No motion is executed up to that point, but G-Code modality changes are registered. Execution continues normally from the indicated line or label, including motion commands. |

Comments

- > Unless line #1 is used in the **START** command, it is recommended to use a label since line numbers are prone to change if programs lines are deleted or inserted.
- > **START** executes successfully if the target buffer is loaded with a program, compiled, but not running. Otherwise, **START** causes a run-time error and aborts the current program.
- > The program activated by **START** executes concurrently with the program containing the **START** command, and other active programs.
- > If the whole program needs to be run in simulation mode, for early run-time problem detection for example, the **START/s** command can be used without specifying a line number or label argument.
- > Simulation mode does not affect ACSPL+ lines, which will run normally.

Related ACSPL+ Commands

STOP/STOPALL, BREAK

Examples

Example 1:

```
START 8,15           !Start program in buffer 8 line 15
```

Example 2:

```
START 8,Start_here  !Start program in buffer 8, at the line identified
                    !by the Start_here label.
```

Example 3 (simulation mode):

```
1 !Buffer 0
2 ENABLE(X,Y)
3 N20 G00 X0
4 N30 G01 X20 F2000
5 N40 G01 X10
6 N50 G01 X20
7 !Radius compensation
8 N10 G42 D10
9 RC:
10 N60 G01 X0
```



```

11 N30 G01 X20 F4000
12 N40 G01 X10
13 N50 G01 X20
14 N60 G01 X0
15 N70 M02
!Buffer 1
START/s 0,10 !Execution of buffer 0 in simulation mode until line 10
STOP

```

2.8.6 STOP/STOPALL

Description

STOP terminates program execution in the specified buffer. **STOPALL** terminates program execution in all buffers.

Syntax

STOP *buffer-number*

Arguments

| | |
|----------------------|-------------------------------------------------------------------------------------------------|
| <i>buffer-number</i> | A number between 0 and 63 specifying the buffer in which the program execution is to be halted. |
|----------------------|-------------------------------------------------------------------------------------------------|

COM Library Methods and .NET Library Methods

StopBuffer

C Library Functions

acsc_StopBuffer

Example

```
STOP 5           !Terminate program execution in buffer five.
```

2.9 Ethernet/IP ACSPL+ Support Commands

The following ACSPL+ functions set assembly configuration and get existing assembly configuration. For more information refer to *SPiPlus EtherNet/IP User Guide*.

2.9.1 EIPGETATTR

Description

EIPGETATTR returns value of a specific attribute. It can be a class attribute or an instance attribute.

Syntax

int eipgetattr(int class, int instance, int attr)

Arguments

| Class | The following classes are supported: | |
|-----------------|-------------------------------------------------------------------------------------------------------------------|---------------------|
| | Class Code | Class Name |
| | 0x01 | Identity |
| | 0x02 | Message Router |
| | 0x06 | Connection Manager |
| | 0xF4 | Port |
| | 0xF5 | TCP/IP Interface |
| | 0xF6 | Ethernet Link |
| | 0x04 | Assembly |
| | 0x64 | ACSPL+ Command |
| 0x65 | ACSPL+ Variable | |
| Instance | For the class attribute, this parameter should 0. Otherwise, the specific instance should be specified as follows | |
| | Class Code | Supported Instances |
| | 0x01 | 1 |
| | 0x02 | 1 |
| | 0x06 | 1 |
| | 0xF4 | 1,2,3 |
| | 0xF5 | 1 |
| | 0xF6 | 1 |
| | 0x04 | |
| | 0x64 | |
| 0x65 | | |

| Attr | Specifies the attribute as follows: | | |
|-------------|-------------------------------------|-----------------|--------------------|
| | Class Code | Class Supported | Instance Supported |
| | | Attributes | Attributes |
| | 0x01 | 1 | 1,2,3,4,5,6 |
| | 0x02 | 1 | 2 |
| | 0x06 | 1 | 1..8 |
| | 0xF4 | 1,2,3 | 1 |
| | 0xF5 | 1 | 1,2 |
| | 0xF6 | 1 | 1,7,8 |
| | 0x04 | 1 | |
| 0x64 | 1,100 | | |
| 0x65 | 1 | | |

Return Value

Returns value of a specific attribute.

-1 is returned in case of illegal parameters.

2.9.2 EIPGETIND1**Description**

EIPGETIND1 returns the first index of the requested ACSPL+ standard or user-defined variable in a one-dimensional array. The indexes start from 0.

Syntax

```
int eipgetind1(int instance, int element )
```

Arguments

| | | |
|-----------------|------------------------------------------------------------------------------------------|-------------------------|
| instance | Assembly instance. The following instances are supported: | |
| | Instance | Instance Name |
| | > 0x64 (100) | Input Integer Assembly |
| | > 0x65 (101) | Output Integer Assembly |
| | > 0x66 (102) | Input Real Assembly |
| > 0x67 (103) | Output Real Assembly | |
| element | The index of element in the corresponding assembly. Only indexes 0 to 123 are supported. | |

Return Value

0 is returned in case of scalar variable.
 -1 is returned in case of illegal index parameter.

2.9.3 EIPGETIND2

Description

EIPGETIND2 returns the first index of the requested ACSPL+ standard or user-defined variable in a two-dimensional array. The indexes start from 0.

Syntax

int eipgetind2(int instance, int element)

Arguments

| | |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>instance</i> | Assembly instance. The following instances are supported: Instance Instance Name > 0x64 (100) Input Integer Assembly > 0x65 (101) Output Integer Assembly > 0x66 (102) Input Real Assembly > 0x67 (103) Output Real Assembly |
| <i>element</i> | The index of element in the corresponding assembly. Only indexes 0 to 123 are supported. |

Return Value

0 is returned in case of scalar variable or one-dimensional array.
 -1 is returned in case of illegal index parameter.

2.9.4 EIPGETTAG

Description

EIPGETTAG returns the tag number of requested ACSPL+ standard or user-friendly variable. User-friendly variables tags start from index 1000.

Syntax

int eipgettag(int instance, int element)

Arguments

| <i>instance</i> | <p>Assembly instance. The following instances are supported:</p> <table border="1"> <thead> <tr> <th>Instance</th> <th>Instance Name</th> </tr> </thead> <tbody> <tr> <td>> 0x64 (100)</td> <td>Input Integer Assembly</td> </tr> <tr> <td>> 0x65 (101)</td> <td>Output Integer Assembly</td> </tr> <tr> <td>> 0x66 (102)</td> <td>Input Real Assembly</td> </tr> <tr> <td>> 0x67 (103)</td> <td>Output Real Assembly</td> </tr> </tbody> </table> | Instance | Instance Name | > 0x64 (100) | Input Integer Assembly | > 0x65 (101) | Output Integer Assembly | > 0x66 (102) | Input Real Assembly | > 0x67 (103) | Output Real Assembly |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|---------------|--------------|------------------------|--------------|-------------------------|--------------|---------------------|--------------|----------------------|
| Instance | Instance Name | | | | | | | | | | |
| > 0x64 (100) | Input Integer Assembly | | | | | | | | | | |
| > 0x65 (101) | Output Integer Assembly | | | | | | | | | | |
| > 0x66 (102) | Input Real Assembly | | | | | | | | | | |
| > 0x67 (103) | Output Real Assembly | | | | | | | | | | |
| <i>element</i> | The index of element in the corresponding assembly. Only indexes 0 to 123 are supported. | | | | | | | | | | |

Return Value

-1 is returned in case of illegal index parameter.

2.9.5 EIPSETASM**Description**

EIPSETASM sets the assembly configuration.

Syntax

eipsetasm(int instance, int element, int tag, int first, int second)

Arguments

| <i>instance</i> | <p>Assembly instance. The following instances are supported:</p> <table border="1"> <thead> <tr> <th>Instance</th> <th>Instance Name</th> </tr> </thead> <tbody> <tr> <td>> 0x64 (100)</td> <td>Input Integer Assembly</td> </tr> <tr> <td>> 0x65 (101)</td> <td>Output Integer Assembly</td> </tr> <tr> <td>> 0x66 (102)</td> <td>Input Real Assembly</td> </tr> <tr> <td>> 0x67 (103)</td> <td>Output Real Assembly</td> </tr> </tbody> </table> | Instance | Instance Name | > 0x64 (100) | Input Integer Assembly | > 0x65 (101) | Output Integer Assembly | > 0x66 (102) | Input Real Assembly | > 0x67 (103) | Output Real Assembly |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|---------------|--------------|------------------------|--------------|-------------------------|--------------|---------------------|--------------|----------------------|
| Instance | Instance Name | | | | | | | | | | |
| > 0x64 (100) | Input Integer Assembly | | | | | | | | | | |
| > 0x65 (101) | Output Integer Assembly | | | | | | | | | | |
| > 0x66 (102) | Input Real Assembly | | | | | | | | | | |
| > 0x67 (103) | Output Real Assembly | | | | | | | | | | |
| <i>element</i> | The index of element in the corresponding assembly. Only indexes 0 to 123 are supported. | | | | | | | | | | |
| <i>tag</i> | The tag number of required ACSPL+ standard or user-defined variable. User-defined variables tags start from index 1000. | | | | | | | | | | |
| <i>first</i> | The first index of required ACSPL+ standard or user-defined variable. The indexes start from 0. Used only if variable is one or two-dimension array. Should be 0 for scalar variable. | | | | | | | | | | |
| <i>second</i> | The first index of required ACSPL+ standard or user-defined variable. The indexes start from 0. Used only if variable is two-dimensional array. Should be 0 for scalar variable or one-dimensional array. | | | | | | | | | | |

2.10 Laser Control Commands

The laser commands are:

| Command | Description |
|---------------------------|-----------------------------------------------------------------|
| LCENABLE | Enables a pulse generation process with current set parameters. |
| LCDISABLE | Stops a pulse generation process, including tickle pulses. |

2.10.1 LCENABLE

The **LCENABLE** command enables a pulse generation process with current set parameters. If initially frequency or duty cycle is set to zero, than pulse generation will be pending till positive value will be set. This approach allows synchronize the pulse generation with actual motion, when, for example, frequency or duty cycle is updated as a function of motion reference velocity.

Syntax

LCENABLE(*Index*)

Arguments

| Arguments | Comment |
|-----------|-------------------------------------------------------------|
| Index | Refers to any of the axis that are allocated for the laser. |

2.10.2 LCDISABLE

The **LCDISABLE** command stops a pulse generation process, including tickle pulses.

Syntax

LCDISABLE(*Index*)

Arguments

| Arguments | Comment |
|-----------|-------------------------------------------------------------|
| Index | Refers to any of the axis that are allocated for the laser. |

2.11 Input Shaping Commands

The input shaping commands are as follows.

| Command | Description |
|----------------------------|-------------------------|
| INSHAPEON | Activate input shapping |
| INSHAPEOFF | Terminate input shaping |

2.11.1 INSHAPEON

Description

The **INSHAPEON** function starts Input Shape algorithm for specified axis. The result is a dynamic output signal equal to the convolution of the input signal and the convolution pulses.

Syntax

INSHAPEON Axis_Index, T_array, A_array

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| T_array | One-dimensional array specifying the times of each convolution in milliseconds. |
| A_array | One-dimensional array specifying the amplitudes of each convolution pulse |

Return Value

None

Comments

Vectors T_array and A_array define characteristics of the convolution pulses. The array sizes should be identical.

Vector T_array contains real numbers, so fractional numbers may be specified. However, the position of each pulse is rounded to a multiple of the controller cycle. If the controller cycle is one millisecond, the numbers in T_array are rounded to integers. The elements of T_array must be arranged in ascending order.

The sum of A_array entries must equal 1.

See the Using the Convolve Web Site chapter in the *ACSPL+ Programmers Guide* to get the explanation how to calculate the T_array and A_array parameters.

This function is supported in version 3.00 and higher..

Examples

```
global real CnvT(5), CnvA(5), CnvB(420)
VEL(0) = 120
ACC(0) = VEL(0)*10
JERK(0) = ACC(0)*10

CnvT(0)=0; CnvT(1)=64 ; CnvT(2)=68 ;CnvT(3)=72; CnvT(4)=139
CnvA(0)=25345/1e5; CnvA(1)=160/1e5; CnvA(2)=30987/1e5; CnvA(3)=18949/1e5;
CnvA(4)=24559/1e5

enable 0
InShapeOn 0, CnvT, CnvA
```

```
ptp/e 0,0
ptp/e 0,50
till ^MST(0).#MOVE
InShapeOff 0
stop
```

```
!INPUTSHAPE EXAMPLE: IF CTIME < 1

!In this case we need to multiply CnvT array by CTIME. Here CTIME = 0.5
global real CnvT(5), CnvA(5), CnvB(420)
CnvT(0)=0*CTIME; CnvT(1)=1*CTIME ; CnvT(2)=214*CTIME ;CnvT(3)=253*CTIME;
CnvT(4)=501*CTIME
CnvA(0)=22960/1e5; CnvA(1)=10361/1e5; CnvA(2)=3186/1e5; CnvA
(3)=45767/1e5;
CnvA(4)=17726/1e5
enable 0
InShapeOn 0, CnvT, CnvA
ptp/e 0,0
ptp/e 0,30
till ^MST(0).#MOVE
!InShapeOff 0
stop
```

2.11.2 INSHAPEOFF

Description

The **INSHAPEOFF** function stops the Input Shape algorithm for the specified axis.

Syntax

INSHAPEOFF Axis_Index

Arguments

Axis_Index

Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

Comments

This variable is supported in version 3.00 and higher.

Return Value

None

3. ACSPL+ Variables

ACSPL+ supports data types of integer, real, and matrix. Matrix variables are a two-dimensional array of real values.

This chapter covers the ACSPL+ variables. ACSPL+ has a complete set of built-in variables for use in setting values that control ACSPL+ programs. The ACSPL+ variables are divided into the following categories:

- > [Axis Configuration Variables](#)

Along with the following subgroups:

- > [Brake Variables](#)
- > [Feedback Variables](#)
- > [Safety Limits Variables](#)
- > [Axis State Variables](#)
- > [Data Collection Variables](#)
- > [Input and Output Variables](#)
- > [Monitoring Variables](#)
- > [Motion Variables](#)
- > [Program Execution Control Variables](#)
- > [Safety Control Variables](#)
- > [Induction Motor Variables](#)
- > [Nanomotion Variables](#)
- > [Servo-Loop Variables](#)

Along with the following subgroups:

- > [Servo-Loop Current Variables](#)
- > [Servo-Loop Velocity Variables](#)
- > [Servo-Loop Velocity Notch Filter Variables](#)
- > [Servo-Loop Velocity Low Pass Filter Variables](#)
- > [Servo-Loop Velocity Bi-Quad Filter Variables](#)
- > [Servo-Loop Position Variables](#)
- > [Servo-Loop Compensations Variables](#)
- > [Servo-Loop Miscellaneous Variables](#)
- > [Commutation Variables](#)
- > [System Configuration Variables](#)
- > [Communication Variables](#)
- > [Miscellaneous](#)

The ACSPL+ programming variables are:

Table 3-1. Alphabetical Listing of All ACSPL+ Variables

| Name | Description | Variable Group |
|----------|-----------------------------------------|---------------------------------------|
| ACC | Acceleration | Motion |
| AFLAGS | Axis Flags | Axis Configuration |
| AIN | Analog Inputs | Input and Output |
| AOUT | Analog Outputs | Input and Output |
| APOS | Axis Position | Motion |
| AST | Axis State | Axis State |
| BAUD | Serial Communication Baud Rate | Communication |
| BOFFTIME | Brake Deactivation Time | Axis Configuration - Brake |
| BONTIME | Brake Activation Time | Axis Configuration - Brake |
| CERRA | Critical Position Error in Accelerating | Axis Configuration - Safety Limits |
| CERRI | Critical Position Error in Idle | Axis Configuration - Safety Limits |
| CERRV | Critical Position Error in Moving | Axis Configuration - Safety Limits |
| CFG | Configuration Mode | System Configuration |
| COMMCH | Communication Channel | Communication |
| COMMFL | Communication Flags | Communication |
| CONID | Controller Identification | Communication |
| CTIME | Controller Cycle Time | System Configuration |
| DAPOS | Delayed Axis Position | Motion |
| DCN | Axis DC, Number of Samples | Data Collection |

| Name | Description | Variable Group |
|----------|------------------------------------------------------------------------------|------------------------------------|
| DCOM | Drive Command | Servo-Loop |
| DCP | Axis DC, Period | Data Collection |
| DEC | Deceleration | Motion |
| DELI | Delay on Transition to Idle State | Axis Configuration - Safety Limits |
| DELV | Delay on Transition to Velocity State | Axis Configuration - Safety Limits |
| DISPCH | Default Communication Channel | Communication |
| E_AOFFS | Sets user-defined offset for absolute encoder. | Axis Configuration - Feedback |
| E2_AOFFS | Sets user-defined offset for absolute encoder secondary feedback | Axis Configuration - Feedback |
| E_FREQ | Primary Encoder Frequency | Axis Configuration - Feedback |
| E_SCMUL | Primary Encoder Sin-Cos Multiplier | Axis Configuration - Feedback |
| E_TYPE | Primary Encoder Type | Axis Configuration - Feedback |
| E2FAC | Secondary Encoder Factor | Axis Configuration - Feedback |
| E2_FLAGS | Contains configuration bits for secondary feedback absolute encoder. | Axis Configuration - Feedback |
| E2_FREQ | Defines the maximum encoder pulse frequency (in MHz) for secondary feedback. | Axis Configuration - Feedback |
| E2OFFS | Secondary Encoder Offset | Axis Configuration - Feedback |
| E2_PAR_A | Sets the encoder data transmission frequency | Axis Configuration - Feedback |
| E2_PAR_B | Sets the encoder data control CRC code | Axis Configuration - Feedback |

| Name | Description | Variable Group |
|----------|--------------------------------------------------------------------------------|------------------------------------|
| E2_PAR_C | Sets the interval (in microseconds) of encoder position reading | Axis Configuration - Feedback |
| E2_PAR_D | Defines number of status bits (LSB) in the real-time position data | Axis Configuration - Feedback |
| E2_PAR_E | Defines a mask for setting error bits | Axis Configuration - Feedback |
| E2_SCMUL | Specifies the Sin-Cos multiplication factor for the secondary feedback encoder | Axis Configuration - Feedback |
| ECERR | Contains EtherCAT Error Code | Safety Control |
| ECHO | Echo Communication Channel | Communication |
| ECST | Contains EtherCAT Status | Safety Control |
| EFAC | Primary Encoder Factor | Axis Configuration - Feedback |
| ENTIME | Enable Time | Axis Configuration |
| EOFFS | Primary Encoder Offset | Axis Configuration - Feedback |
| E_PAR_A | | Feedback Variables |
| E_PAR_B | | Feedback Variables |
| E_PAR_C | | Feedback Variables |
| EPOS | Shows the encoder feedback | Feedback |
| ERRA | Position Error in Accelerating | Axis Configuration - Safety Limits |
| ERRI | Position Error in Idle | Axis Configuration - Safety Limits |
| ERRV | Position Error in Moving | Axis Configuration |
| EXTIN | Extended Inputs (HSSI) | Input and Output |
| EXTOUT | Extended Outputs (HSSI) | Input and Output |

| Name | Description | Variable Group |
|-------------------|----------------------------------------------------------------------------|-------------------------------|
| F2ACC | Defines the feedback acceleration value of the axis in secondary feedback. | Feedback |
| F2POS | Secondary Feedback Position | Motion |
| F2VEL | Secondary Feedback Velocity | Motion |
| FACC | Primary Feedback Acceleration | Motion |
| FAULT | Faults | Safety Control |
| FDEF | Default Response Mask | Safety Control |
| FK | Function Key | Miscellaneous |
| FMASK | Fault Mask | Safety Control |
| FPOS | Primary Feedback Position | Axis State |
| FVEL | Primary Feedback Velocity | Motion |
| FVFIL | Primary Feedback Velocity Filter | Axis Configuration - Feedback |
| G_01WCS...G_12WCS | Used for defining one of the 12 work-piece coordinate systems | System Configuration |
| GACC | Group Acceleration | Motion |
| GATEWAY | Gateway Address for 1 st Ethernet | Communication |
| GJERK | Group Jerk | Motion |
| GMOT | Motion Number | Motion |
| GMQU | Motion Queue | Motion |
| GMTYPE | Motion Type | Motion |
| GPATH | Group Path | Motion |
| GPEXL | Indicates the GSP program executed block | System Configuration |
| GPHASE | Motion Phase | Motion |

| Name | Description | Variable Group |
|---------|----------------------------------------------------------------------------------------|----------------------|
| GRTIME | Remaining Motion Time | Motion |
| GSEG | Motion Segment | Motion |
| GSFREE | Free Motion Segments | Motion |
| GUFAC | Holds the conversion factor from 'Common Physical Units' in [mm] to 'Controller Units' | System Configuration |
| GVEC | Group Vector | Motion |
| GVEL | Group Velocity | Motion |
| IENA | Interrupt Enable/Disable | System Configuration |
| IMASK | Interrupt Mask | System Configuration |
| IN | Digital Inputs | Input and Output |
| IND | Primary Index Position | Axis State |
| ISENA | Specific Interrupt Enable/Disable | System Configuration |
| IST | Index State | Axis State |
| JERK | Jerk | Motion |
| JITTER | Time difference between physical timer interrupt and start of SC real-time task | SPiiPlusSC |
| KDEC | Kill Deceleration | Motion |
| M2ARK | Secondary Mark Position | Axis State |
| MARK | Primary Mark Position | Axis State |
| MERR | Motor Error | Safety Control |
| MFF | Master Feed Forward | Axis Configuration |
| MFLAGS | Motor Flags | Axis Configuration |
| MFLAGSX | Extended Motor Flags | Axis Configuration |

| Name | Description | Variable Group |
|--------|--------------------------------------------------------------------------------|---------------------------|
| MPOS | Master Position | Motion |
| MSSYNC | Difference between master clock and bus clock | SPiiPlusSC |
| MST | Motor State | Axis State |
| NST | Reads the status of EtherCAT Sync and GPRT errors for each axis in the system. | Axis State |
| NVEL | Minimal Velocity | Motion |
| ONRATE | Autoroutine Rate | Program Execution Control |
| OUT | Digital Outputs | Input and Output |
| PCHARS | Program Size in Characters | Program Execution Control |
| PE | Non-Critical Position Error | Motion |
| PERL | Program Error Line | Program Execution Control |
| PERR | Program Error | Program Execution Control |
| PEXL | Program Executed Line | Program Execution Control |
| PFLAGS | Program Flags | Program Execution Control |
| PLINES | Program Size in Lines | Program Execution Control |
| PRATE | Program Rate | Program Execution Control |
| PST | Program State | Program Execution Control |
| RACC | Reference Acceleration | Motion |
| | RMS current of axis | Axis State |

| Name | Description | Variable Group |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| ROFFS | Reference Offset | Axis State |
| RPOS | Reference Position | Axis State |
| RVEL | Reference Velocity | Motion |
| RVFIL | Reference Velocity Filter | Axis Configuration - Feedback |
| S_DCN | System DC, Number of Samples | Data Collection |
| S_DCP | System DC, Period | Data Collection |
| S_ERR | System Error | Safety Control |
| S_FAULT | System Faults | Safety Control |
| S_FDEF | System Default Response Mask | Safety Control |
| S_FLAGS | System Flags | System Configuration |
| S_FMASK | System Fault Mask | Safety Control |
| S_SAFIN | System Safety Inputs | Safety Control |
| S_SAFINI | System Safety Inputs Inversion | Safety Control |
| S_SETUP | Bit mask defining various system settings | System Configuration |
| S_ST | State of System Data Collection | Data Collection |
| SAFIN | Safety Inputs | Safety Control |
| SAFINI | Safety Inputs Inversion | Safety Control |
| SC2COFFS | Defines the Sin-Cos Sine offset in secondary feedback | Axis Configuration - Feedback |
| SC2GAIN | Sin-Cos encoder gain compensation variable used to compensate the Cosine signal for an improper amplitude relative to the Sine signal in secondary feedback | Axis Configuration - Feedback |

| Name | Description | Variable Group |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| SC2PHASE | Sin-Cos encoder phase compensation variable and is used to compensate the Cosine signal for an improper phase difference relative to the Sine signal in secondary feedback. | Axis Configuration - Feedback |
| SC2SOFFS | Defines the Sin-Cos Sine offset in secondary feedback. | Axis Configuration - Feedback |
| SCCOFFS | Sin-Cos Offset (Cosine) | Axis Configuration - Feedback |
| SCGAIN | Cosine gain compensation | Axis Configuration - Feedback |
| SCPHASE | Cosine phase compensation | Axis Configuration - Feedback |
| SCSOFFS | Sin-Cos Offset (Sine) | Axis Configuration - Feedback |
| SETTLE | Settling Time | Axis Configuration |
| S2LABITS | Used for setting the total number of absolute position bits for an absolute encoder connected to a secondary feedback | Axis Configuration - Feedback |
| SLABITS | Absolute Position Bits | Axis Configuration - Feedback |
| SLAFF | Acceleration Feed Forward | Servo-Loop - Compensations |
| SLBIASA | Current Phase A Bias | Servo-Loop - Current |
| SLBIASB | Current Phase B Bias | Servo-Loop - Current |
| SLCFIELD | Induction Motor Excitation | Induction Motor |
| SLCHALL | Hall Shift | Commutation |
| SLCNP | Number of Motor Poles | Commutation |
| SLCOFFS | Commutation Offset | Commutation |

| Name | Description | Variable Group |
|----------|-------------------------------------------------------------------------------------------------|-------------------------------|
| SLCORG | Commutation Origin | Commutation |
| SLCPRD | Commutation Period | Commutation |
| SLCROUT | Commutation Feedback | Servo-Loop - Miscellaneous |
| SLCSLIP | Induction Motor Slip Factor | Induction Motor |
| SLDRAIF | DRA frequency | Servo-Loop - Position |
| SLDRAIF | Provides an Idle Factor to the SLDRA variable. | Servo-Loop - Position |
| SLDRX | Maximum DRA correction | Servo-Loop - Position |
| SLDZMAX | Maximum Dead Zone position | Servo-Loop - Nanomotion |
| SLDZMIN | Minimum Dead Zone position | Servo-Loop - Nanomotion |
| SLEBIASA | Defines encoder hardware Sine offset | Axis Configuration - Feedback |
| SLEBIASB | Defines encoder hardware Cosine offset | Axis Configuration - Feedback |
| SLEBIASC | Defines the Sin-Cos encoder's hardware compensation for the Sine offset in secondary feedback | Axis Configuration - Feedback |
| SLEBIASD | Defines the Sin-Cos encoder's hardware compensation for the Cosine offset in secondary feedback | Axis Configuration - Feedback |
| SLFRC | Static Friction | Servo-Loop - Compensations |
| SLFRCD | Dynamic Friction | Servo-Loop - Compensations |
| SLHROUT | Sets the Hall state routing | Commutation |

| Name | Description | Variable Group |
|----------|-------------------------------------------------------------------------------------|----------------------------|
| SLIFILT | Internal Current filter | Servo-Loop - Current |
| SLIKI | Integrator Gain | Servo-Loop - Current |
| SLIKP | Integrator Proportional Gain | Servo-Loop - Current |
| SLIFILT | Internal Current filter | Servo-Loop - Current |
| SLILI | Determines output voltage | Servo-Loop - Current |
| SLIOFFS | Current Command Offset | Servo-Loop - Current |
| SLLIMIT | Soft Left Limit | Safety Limits |
| SLLROUT | Sets the HW limits routing for the specified axis | Commutation |
| SLP2ROUT | Sets the feedback routing of the secondary feedback position for the specified axis | Servo Loop - Miscellaneous |
| SLPKITF | Increases position loop integrator coefficient | Servo-Loop - Position |
| SLPKP | Proportional Position Gain | Servo-Loop - Position |
| SLPKPIF | Provides an Idle Factor to the SLPKP variable | Servo-Loop - Position |
| SLPKPSF | Provides a Settling Factor to the SLPKP variable | Servo-Loop - Position |
| SLPKPTF | Increases position loop proportional coefficient | Servo-Loop - Position |
| SLPMAX | Modulo Axis Upper Limit | Axis Configuration |
| SLPMIN | Modulo Axis Lower Limit | Axis Configuration |
| SLPROUT | Position Feedback Routing | Servo-Loop - Miscellaneous |

| Name | Description | Variable Group |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| SLSTHALL | The hall state of each axis | Commutation |
| SLTFWID | Determines distance to target at which position and velocity loops gains increase by 50% Servo-Loop - Miscellaneous SLVBODD Bi-Quad filter | Servo-Loop - Miscellaneous |
| SLVBODD | Bi-Quad filter damping ratio denominator | Servo-Loop - Filter |
| SLVBODF | Bi-Quad filter algorithm denominator | Servo-Loop - Filter |
| SLVBOND | Bi-Quad filter damping ratio numerator | Servo-Loop - Filter |
| SLVBONF | Bi-Quad filter algorithm numerator | Servo-Loop - Filter |
| SLVKI | Velocity Integrator Coefficient | Servo-Loop - Velocity |
| SLVKIIF | Provides an Idle Factor to SLVKI variable | Servo-Loop - Velocity |
| SLVKISF | Provides a Settle Factor to the SLVKI variable | Servo-Loop - Velocity |
| SLVKITF | Increases velocity loop integrator coefficient | Servo-Loop - Velocity |
| SLVKP | Proportional Velocity Gain | Servo-Loop - Velocity |
| SLVKPIF | Provides an Idle Factor to the SLVKP variable | Servo-Loop - Velocity |
| SLVKPSF | Provides a Settle Factor to the SLVKP variable | Servo-Loop - Velocity |
| SLVKPTF | Increases the velocity loop proportional coefficient | Servo-Loop - Velocity |
| SLVLI | Integrator Velocity Limit | Servo-Loop - Velocity |
| SLVNATT | Notch Filter Attenuation | Servo-Loop - Filter |
| SLVNRQ | Notch Filter Frequency | Servo-Loop - Filter |
| SLVNWID | Notch Filter Width | Servo-Loop - Filter |

| Name | Description | Variable Group |
|----------|------------------------------------------------------------------------------------------------------------|----------------------------|
| SLVRAT | Velocity Feed Forward Ratio | Servo-Loop - Velocity |
| SLVROUT | Velocity Feedback | Servo-Loop - Miscellaneous |
| SLVSOF | Low-Pass Filter Bandwidth | Servo-Loop - Filter |
| SLVSOFD | Low-Pass Filter Damping | Servo-Loop - Filter |
| SLZFF | Defines zero velocity feed forward position | Servo-Loop - Nanomotion |
| SRLIMIT | Soft Right Limit | Safety Limits |
| STEPF | Stepper Factor | Axis Configuration |
| STEPW | Stepper Pulse Width | Axis Configuration |
| STODELAY | Configures the delay time between the STO fault indication and the default response (disable) to the fault | Safety Control |
| SUBNET | Subnet Mask for 1 st Ethernet | Communication |
| SYNC | Slave Sync counter | Safety Control |
| TARGRAD | Target Radius | Axis Configuration |
| TCPIP | IP Address for 1 st Ethernet | Communication |
| TCPIP2 | IP Address for 2 nd Ethernet | Communication |
| TCPPORT | TCP Port Number | Communication |
| TIME | Elapsed Time | Monitoring |
| TPOS | Target Position | Motion |
| UDPPORT | UDP Port Number | Communication |
| USAGE | MPU Usage | Monitoring |
| VEL | Velocity | Motion |

| Name | Description | Variable Group |
|----------|--------------------------------------|------------------------------------|
| VELBRK | Brake Velocity | Axis Configuration - Brake |
| XACC | Maximum Acceleration | Axis Configuration - Safety Limits |
| XARRSIZE | Maximum Array Size | Miscellaneous |
| XCURI | Maximum Current in Idle | Axis Configuration - Safety Limits |
| XCURCDB | Threshold of the current vector peak | Axis Configuration - Safety Limits |
| XCURV | Maximum Current in Moving | Axis Configuration - Safety Limits |
| XRMS | RMS Current Limit | Axis Configuration - Safety Limits |
| XRMST | RMS Current Time Constant | Axis Configuration - Safety Limits |
| XSACC | Maximum Slave Acceleration | Axis Configuration - Safety Limits |
| XVEL | Maximum Velocity | Axis Configuration - Safety Limits |

3.1 Axis Configuration Variables

The Axis Configuration variables are:

| Name | Description |
|----------|---------------------------------------------------------------------------------------------------------------------------|
| AFLAGS | Axis Flags |
| ENTIME | Enable Time |
| ESTBITS | ESTBITS represents the single turn resolution (number of bits) of the absolute encoder. |
| E2STBITS | E2STBITS represents the single turn resolution (number of bits) of the absolute encoder for the secondary feedback |

| Name | Description |
|-----------------|--------------------------------------------------------------------------------------------------------------------------|
| EMTBITS | EMTBITS represents the multi-turn resolution (number of bits) of the absolute encoder. |
| E2MTBITS | E2MTBITS represents the multi-turn resolution (number of bits) of the absolute encoder for the secondary feedback |
| MFF | Master Feed forward |
| MFLAGS | Motor Flags |
| MFLAGSX | Extended Motor Flags |
| MSTIMEA | Time elapsed from start of motion up to first entering the settled zone |
| MSTIMEB | Time elapsed from start of motion up to first entering the settled zone |
| MSTIMEC | Time elapsed from start of motion up to first entering the settled zone |
| PEGQUE | Count of PEG FIFO, one entry per axis |
| SETTLE | Axis Settling Time Parameter |
| SETTLE | Axis Settling Time Parameter |
| SLCFIELD | Defines magnetic field component of an induction motor |
| SLCSLIP | Defines the slip constant component of an induction motor |
| SLPMAX | Specifies the upper limit of modulo axis |
| SLPMIN | Specifies the lower limit of modulo axis |
| STEPF | Stepper Factor |
| STEPW | Stepper Pulse Width |
| TARGRAD | Axis settling target envelope parameter |

3.1.1 AFLAGS

Description

AFLAGS is an integer array, with one element for each axis in the system, each element of which contains a set of 4 bits.

Syntax

AFLAGS*axis_index*.*bit_designator* = 1|0

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The meanings of bit_designator are given in AFLAGS .meanings of bit_designator |

Table 3-2. AFLAG Bit Description

| Bit Name | No. | Description |
|----------|-----|----------------|
| #NOS | 0 | No S-Profile |
| #SEMIS | 1 | Semi-S-Profile |
| #AUX | 2 | Auxiliary Axis |
| #NOGROUP | 3 | Disable Group |

Tag

3

Comments

Currently, only **AFLAGS**(*axis_index*).**#NOGROUP** can be set in the variable. **AFLAGS**(*axis_index*).**#NOGROUP** disables the axis in a group, so that any group or motion command that includes the axis in a group will fail.

Other bits are reserved for future use, and must be set to zero.

Accessibility

Read-Write



AFLAGS values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**.

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.1.2 ENTIME

Description

ENTIME is a real array, with one element for each axis in the system, and is used for controlling the execution of [ENABLE/ENABLE ALL](#).

Syntax

ENTIME(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308, Default = 50. |

Tag

37

Comments

Since the drive enable process is relatively long, (50 - 100msec) **ENTIME** defines a time duration in msec between [ENABLE/ENABLE ALL](#) and the moment the controller considers the drive as enabled and all faults as [FAULT\(axis_index\).#PE](#), [FAULT\(axis_index\).#CPE](#), [FAULT\(axis_index\).#DRIVE](#) are triggered.

Exact usage of the variable depends on the flag bit [MFLAGS\(axis_index\).#ENMODE](#) (bit 19). See [MFLAGS](#).

Accessibility

Read-Write



ENTIME values cannot be modified if protection is applied to this variable through [SPiiPlus MMI Application Studio](#) → [Toolbox](#) → [Application Development](#) → [Protection](#)

Related ACSPL+ Commands

[ENABLE/ENABLE ALL](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.3 ESTBITS

Description

ESTBITS is an integer array with one element for each axis. It represents the single turn resolution (number of bits) of the absolute encoder.

Syntax

ESTBITS(axis_index)

Arguments

| | |
|-------|---------------------------------------------------------------------------------------------------------------|
| index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | Value ranges from 0 to 64. |

Tag

Comments

The **ESTBITS** variable can only be set by the **ENCINIT()** function.

Related ACSPL+ Variables

EMTBITS, SLABITS

Accessibility

Read-only



SLABITS compatibility is ensured.

If **EMTBITS** and **ESTBITS** are 0, **SLABITS** is still a Read-Write variable.

If **ESTBITS** is not 0 or **EMTBITS** is not 0 (in other words, it was changed by a call to **ENCINIT()**), **SLABITS** cannot be assigned a new value, and the firmware will return an error if an assignment is attempted.

3.1.4 E2STBITS

Description

E2STBITS is an integer array with one element for each axis. It represents the single turn resolution (number of bits) of the absolute encoder for the secondary feedback.

Syntax

E2STBITS(axis_index)

Arguments

| | |
|-------|---------------------------------------------------------------------------------------------------------------|
| index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | Value ranges from 0 to 64. |

Tag

Comments

The **E2STBITS** variable can only be set by the **ENCINIT()** function.

Related ACSPL+ Variables

E2MTBITS, SLABITS

Accessibility

Read-only

3.1.5 EMTBITS

Description

EMTBITS is an integer array with one element for each axis. It represents the multi-turn resolution (number of bits) of the absolute encoder.

Syntax

EMTBITS(axis_index)

Arguments

| | |
|-------|---------------------------------------------------------------------------------------------------------------|
| index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | Value ranges from 0 to 64. |

Tag

Comments


The **EMTBITS** variable can only be set by the **ENCINIT()** function.

Related ACSPL+ Variables

ESTBITS, SLABITS

Accessibility

Read-only



SLABITS compatibility is ensured.
 If **EMTBITS** and **ESTBITS** are 0, **SLABITS** is still a Read-Write variable.
 If **ESTBITS** is not 0 or **EMTBITS** is not 0 (in other words, it was changed by a call to **ENCINIT()**), **SLABITS** cannot be assigned a new value, and the firmware will return an error if an assignment is attempted.

3.1.6 E2MTBITS

Description

E2MTBITS is an integer array with one element for each axis. It represents the multi-turn resolution (number of bits) of the absolute encoder for the secondary feedback.

Syntax

E2MTBITS(axis_index)

Arguments

| | |
|-------|---------------------------------------------------------------------------------------------------------------|
| index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | Value ranges from 0 to 64. |

Tag**Comments**

The **E2MTBITS** variable can only be set by the **ENCINIT()** function.

Related ACSPL+ Variables**E2STBITS, SLABITS****Accessibility**

Read-only

3.1.7 MFF**Description**

MFF is a real array, with one element for each axis in the system, and is used for specifying the feed forward time, in milliseconds, for **MPOS** calculations.

Syntax

MFF(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 10, Default = 2. |

Tag

87

Comments

The controller calculates the **MPOS** value according to a formula supplied by the **MASTER** command. A non-zero **MFF** value provides additional extrapolation of the calculated value to the predicted value at the current time plus **MFF**. The purpose is to compensate delay introduced by the controller and the external circuits.

The default value of **MFF** depends on the controller model so that it compensates the delay introduced by the controller itself. Increase the **MFF** value if you want to compensate additional delay introduced by sensor or other circuits.

Accessibility

Read-Write



MFF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[MASTER](#)

Related ACSPL+ Variables

[ENTIME](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.8 MFLAGS

Description

MFLAGS is an integer array, with one element for each axis in the system, each element of which contains a set of bits used for configuring the motor.

Syntax

MFLAGS(*axis_index*).*bit_designator* = 0/1

Arguments


| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The MFLAGS bit designators are given in MFLAGS Bit Designators . |


Table 3-3. MFLAGS Bit Designators


| Name | No. | Description |
|--------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #DUMMY | 0 | <p>0 (default) = The axis is defined as non-dummy. 1 = The axis is defined as dummy. A dummy axis is an inactive axis which is not connected to a drive and a motor. When an axis is defined as a dummy the following apply:</p> <ul style="list-style-type: none"> > In SPiiPlus controllers, excluding Control Modules, the two analog outputs of the axis can be used as General Purpose outputs. > In all SPiiPlus products: all faults of the axis are disabled. > In all SPiiPlus products ENABLE/ENABLE ALL is disabled. |
| #OPEN | 1 | <p>0 (default): Closed-loop control - for servo motors only. 1: Open-loop control. In open loop control the user defines the value of the command/s to the drive with variable DCOM.</p> |
| #MICRO | 2 | <p>0 (default): If MFLAGS.#PHASE2 = 1 – motor operated in full step mode. If MFLAGS.#PHASE2 = 0 – bit is ignored 1: If MFLAGS.#PHASE2 = 1 – motor operated in micro step mode. If MFLAGS.#PHASE2 = 0 – #MICRO bit should be cleared</p> |
| #HOME | 3 | <p>0 (default): Axis homing procedure not done. 1: Axis homing procedure done. This bit is changed by the ACSPL+ HOME command after a successful homing process. E_TYPE and other encoder initialization processes will reset the bit. The controller clears the bit during power-up.</p> |

| Name | No. | Description |
|----------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #STEPPER | 4 | <p>#STEPPER is applicable only for PDMnt products.</p> <p>0 (default): Axis is defined as a servo motor.</p> <p>1: Axis is defined as a pulse-direction stepper motor in open loop.</p> |
| #ENCLOOP | 5 | <p>#ENCLOOP is not applicable for NT products.</p> <p>0 (default): Stepper feedback loop is not active.</p> <p>1: The axis provides a stepper feedback loop. In this case the FPOS(axis_index) variable will indicate the number of pulses that were sent to the pulse-direction stepper drive and not the encoder counts - if an encoder is connected and #STEPENC (bit 6 = 1). #ENCLOOP is effective only if #STEPPER (bit 4) = 1.</p> |
| #STEPENC | 6 | <p>0 (default): Stepper encoder feedback loop is not active.</p> <p>1: The controller provides an encoder feedback to the pulse-direction stepper. In this case, the FPOS (axis_index) variable will indicate the quadrature encoder counts, and not the stepper pulse-direction pulses if #ENCLOOP (bit 5 = 1).</p> <p>#STEPENC is effective only if #STEPPER (bit 4) = 1. The encoder feedback is used for monitoring the axis position by a user application, and does not affect the open loop control of the stepper motor.</p> |
| #NANO | 7 | <p>#NANO is applicable only for UDMnt-x (new revision) and UDIhp-x products.</p> <p>0 (default): Defines the axis as a servo or stepper motor.</p> <p>1: Defines the axis as a Nanomotion piezo ceramic motor.</p> |
| #BRUSHL | 8 | <p>0 (default): The motor is a non-DC brushless type or the amplifier provides commutation itself and uses one $\pm 10V$ input from the controller.</p> <p>1: The controller provides commutation for the DC brushless motor. The bit must be set only if the controller is connected to a three-phase amplifier with two-phase input commands.</p> |

| Name | No. | Description |
|----------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #BRUSHOK | 9 | <p>#BRUSHOK is applicable only for DC Brushless motors when #BRUSHL(bit 8)=1.</p> <p>0 (default): DC brushless motor is not commutated.</p> <p>1: DC brushless motor is commutated. After power-up the controller clears the bit. The controller automatically sets this bit =1 only after a successful commutation or auto-commutation processes.</p> |
| #PHASE2 | 10 | <p>0 (default): Two phase motor is not selected.</p> <p>1: Two phase motor is selected.</p> |
| #DBRAKE | 11 | <p>0 (default): Dynamic brake is disabled</p> <p>1: The controller will apply dynamic braking to stop the motor when the axis is disabled and the feedback velocity is less than the predefined value of VELBRK. See the <i>SPiiPlus Setup Guide</i>, Dynamic Brake for a complete explanation.</p> |
| #INVENC | 12 | <p>0 (default): Encoder counting is non-inverted.</p> <p>1: The controller inverts the direction of encoder counting. This does not affect the motion direction.</p> |
| #INVDOUT | 13 | <p>0 (default): Drive output command/s are not inverted.</p> <p>1: The controller inverts the drive output command/s. This effectively inverts the direction of the motion and the sign of the feedback.</p> |
| #NOTCH | 14 | <p>0 (default): Notch filter is disabled.</p> <p>1: Notch filter is active</p> |
| #NOFILT | 15 | <p>0 (default): The control algorithm includes a second-order filter specified by bandwidth SLVSOF.</p> <p>1: The control algorithm by-passes the second-order filter.</p> |

| Name | No. | Description |
|-----------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #BI_QUAD | 16 | 0 (default): First BiQuad Filter is disabled. 1: First BiQuad Filter is active. |
| #DEFCON | 17 | 0: CONNECT is allowed. See CONNECT . 1 (default): The controller applies the default connection between APOS and RPOS (RPOS) . While #DEFCON = 1, the controller does not accept CONNECT for the axis. The bit is reset to 1 every time the controller is restarted. |
| #FASTSC | 18 | 0 (default): not available 1: For working with 5mHz SIN-COS encoders  Bit must be set to 1 in the middle of quadrant. |
| #ENMOD | 19 | 0: Enable time is defined by ENTIME , or when the drive switches its drive alarm signal to the inactive state - whichever comes first. If the signal remains active more than ENTIME milliseconds, ENABLE/ENABLE ALL fails. 1 (default): The value of ENTIME defines the enable time. |
| #DUALLOOP | 20 | 0 (default): The control algorithm implements a regular single-loop scheme. 1: The control algorithm implements a dual-loop scheme. See <i>SPIiPlus Setup Guide, Appendix B</i> for further details. |
| #LINEAR | 21 | 0 (default): The axis is defined as rotary. 1: The axis is defined as linear. |

| Name | No. | Description |
|-----------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #ABSCOMM | 22 | <p>0 (default): Controller doesn't automatically retrieve the commutation phase according to the absolute encoder after powerup.</p> <p>1: The axis is using absolute encoder feedback for commutation and the controller automatically retrieves the commutation phase after powerup.</p> |
| #BRAKE | 23 | <p>0 (default): The controller does not provide mechanical brake control.</p> <p>1: The brake is deactivated when the motor is enabled and activated when the motor is disabled. For detailed description, see commands ENABLE/ENABLE ALL and DISABLE/DISABLEALL, and variables BOFFTIME, BONTIME variables.</p> |
| #HSSI | 24 | <p>#HSSI is not currently supported by NT products.</p> <p>0 (default): Direct drive connection.</p> <p>1: Remote drive connection via HSSI.</p> |
| #GANTRY | 25 | <p>0 (defaults): standard (SISO) PIV control scheme.</p> <p>1: gantry control (MIMO) scheme.</p> |
| #BI_QUAD1 | 26 | <p>(default): Second BiQuad Filter is disabled.</p> <p>1: Second BiQuad Filter is active.</p> |
| #HALL | 27 | <div style="border: 1px solid black; border-radius: 10px; padding: 10px; margin-bottom: 10px;">  <p>#HALL applies only to DC brushless motors when #BRUSHL (bit 8) =1.</p> </div> <p>0 (default): Commutation is not based on Hall signals.</p> <p>1: Commutation is based on Hall signals.</p> |

| Name | No. | Description |
|----------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #INVHALL | 28 |  <p>#INVHALL applies only for DC brushless motors when #BRUSHL (bit 8) =1.</p> <p>0 (default): Motor Hall signals counting direction is not inverted. 1: Motor Hall signals counting direction is inverted.</p> |
| #MODULO | 29 | <p>0 (default): Axis is defined as non-modulo. 1: Axis is defined as MODULO. In MODULO mode, physically, the motion of the axis is not limited, but each time RPOS goes out of the defined SLPMIN...SLPMAX range, the controller brings RPOS into range by changing the internal offset EOFFS. See SLPMAX and SLPMIN.</p> |
| #USER1 | 30 | <p>The functionality of this bit can be defined by the user. 0 (default): Functionality not defined 1: User defined functionality</p> |
| #USER2 | 31 | <p>The functionality of this bit can be defined by the user. 0 (default): Functionality not defined 1: User defined functionality</p> |

Tag

88

Comments

MFLAGS is typically configured using the **SPiiPlus MMI Application Studio → Toolbox → Setup → Adjuster** when setting the Dynamic Brake.

Use direct bit assignment for on-the-fly changes, for example, from closed-loop operation to the open-loop and vice versa.

Accessibility

Read-Write



MFLAGS values cannot be modified if protection is applied to this variable through
SPIIPlus MMI Application Studio → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[CONNECT](#), [ENABLE/ENABLE ALL](#), [GETCONF](#)

Related ACSPL+ Variables

[SLPROUT](#), [APOS](#), [RPOS](#)

COM Library Methods and .NET Library Methods

[ReadVariable](#), [WriteVariable](#)

C Library Functions

[acsc_ReadInteger](#), [acsc_WriteInteger](#)

3.1.9 MFLAGSX

Description

MFLAGSX is an integer array with one element for each axis in the system, each element of which contains a set of bits used for configuring the motor. It is an extension of the **MFLAGS** variable.

Syntax

```
MFLAGSX(Axis_Index).bit_designator = 0|1
```

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Bit_designator | An MFLAGSX bit designator as described below |

MFLAGSX Bit Designators

| Bit Name | No. | Description |
|------------------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #STCLFULL | 0 | 0 (default): Do not use position correction for Stepper motors working under closed loop. 1: Activate Full mode of the stepper closed loop position correction mechanism . This bit is mutually exclusive to #STCLEND and #STCLSP , as such only one of them may be 1 at a time. |
| #STCLEND | 1 | 0 (default): Do not use position correction for Stepper motors working under closed loop 1: Activate End mode of the stepper closed loop position correction mechanism |

| Bit Name | No. | Description |
|------------------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | This bit is mutually exclusive to #STCLFULL and #STCLSP , as such only one of them may be 1 at a time. |
| #STCLSP | 2 | 0 (default): Do not use servo processor closed-loop stepper algorithm 1: Use servo processor closed-loop stepper algorithm Mutually exclusive to #STCLFULL , #STCLEND . |
| #VOLTMODE | 3 | 0: Use current mode for control 1: Use voltage mode instead of current mode to compensate for low resolution |
| #HLIMSWAP | 4 | 0: Left/Right limit signal swapping is disabled 1: Left/Right limit signal swapping is enabled |
| #SATPROT | 5 | 1: Saturation Protection enabled (default) 0: Saturation Protection disabled |

Tag

357

Comments

When saturation protection (**MFLAGSX.#SATPROT=1**) is currently available for the following products: NPMpm, UDMxx, IDMxx.

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKI, SLSKP, SLSDZ

Accessibility

Read-Write



MFLAGSX values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio→Toolbox→Application Development→Protection

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.1.10 MODULOMD

Description

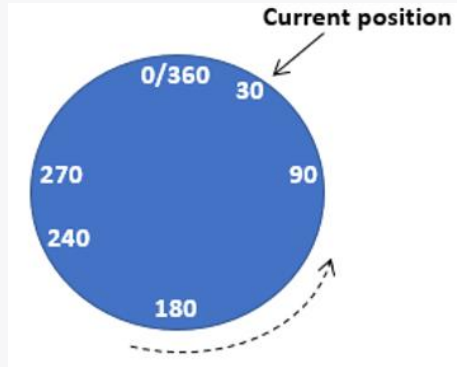
MODULOMD is an integer array, with one element for each axis in the system, the elements of which are used for storing the mode of a modulo axis.

Syntax

```
MODULOMD(axis_index) = mode
```

Arguments

| | |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| mode | <p>Valid values are:</p> <ul style="list-style-type: none"> > 0(default) – The axis feedback changes between the specified minimal (SLPMIN) and maximal(SLPMAX) positions. The reference position RPOS of the modulo axis changes in the range from SLPMIN to SLPMAX inclusively. Physically, the motion of the modulo axis is not limited, but each time when the RPOS comes out from range SLPMIN...SLPMAX, the controller brings RPOS into the range by changing the internal offset EOFFS. In the case of a default connection, the modulo operation also affects the APOS value (APOS=RPOS). > 1 (No-Pass-Through-Rollover) - Every Motion from will avoid moving through the Rollover point (where SLPMIN and SLPMAX are topologically united). This also applies to Motion Commands that exceed the Modulo range as set by SLPMAX(\$) and SLPMIN(\$). <p>Example:</p> <p>Assume SLPMIN(axis_index)=0 and SLPMAX(axis_index)=360, FPOS (axis_index) is reported as 30, and a Relative point-to-point command is issued:</p> <pre>PTP/r axis_index, 570</pre> <p>Internal calculations will determine that the target position is $(30 + 570) \% 360 = 240$, motion will be in positive direction passing through 180 point to 240.</p> |



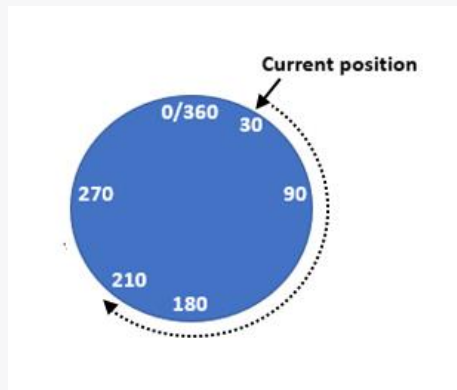
- > **2 (Positive motion only)** - Every Motion Command (excluding Jog) will cause the Axis to move in the positive Direction to the modulo of the Target Position. This also applies to motion commands that exceed the range set by **SLPMAX**(axis_index) and **SLPMIN**(axis_index).

Example 1:

Assume **SLPMIN**(axis_index)=0 and **SLPMAX**(axis_index)=360, **FPOS** (axis_index) is reported as 30, and a relative point to point command is issued:

```
PTP/r axis_index, 540
```

Internal calculations will determine that the target position is $(30 + 540) \% 360 = 210$, motion will be in positive direction passing through 180 to 210.



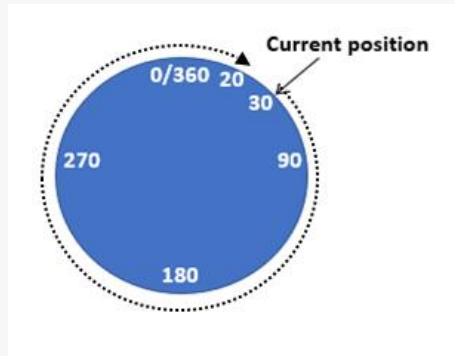
Example 2:

Assume **SLPMIN**(axis_index)=0 and **SLPMAX**(axis_index)=360, **FPOS** (axis_index) is reported as 30, and a point-to-point command is issued:

```
PTP axis_index, 20
```

Internal calculations determine that the target position is $20 \% 360 = 20$, the motion will be in positive direction passing through 180 and

360 point to 20 point.



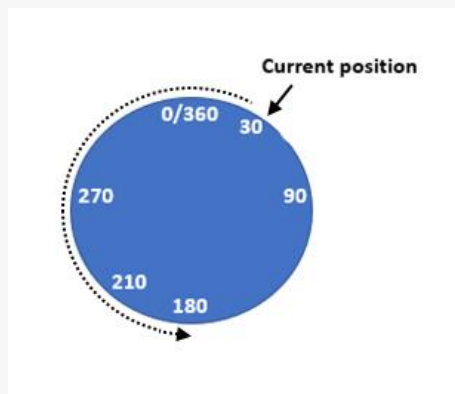
- > **3 (Negative motion only)** - Every Motion Command (excluding Jog) will cause the Axis to move in the negative direction to the modulo of the target position. This also applies to motion commands that exceed the Modulo range as set by **SLP_{MAX}**(axis_index) and **SLP_{MIN}**(axis_index).

Example:

Assume **SLP_{MIN}**(axis_index)=0 and **SLP_{MAX}**(axis_index)=360, **FPOS**(axis_index) is reported as 30, and a Relative point-to-point command is issued:

```
PTP/r axis_index, 540
```

Internal calculations will determine that the target position is $(30 + 540) \% 360 = 210$, motion will be in negative direction passing through 0 and 270 point to 210.



- > **4 (Shortest-Path)** - Each motion command will be analyzed to determine the shortest rotational distance between current position and target position, possibly moving through the Roll-Over point. This also applies to motion Commands that exceed the modulo range as set by **SLP_{MAX}**(axis_index) and **SLP_{MIN}**(axis_index).

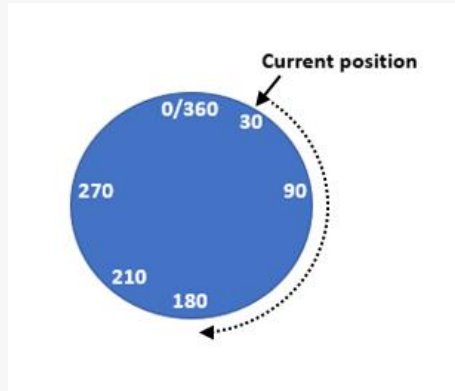
Example 1:

Assume **SLP_{MIN}**(axis_index)=0 and **SLP_{MAX}**(axis_index)=360, **FPOS**(axis_index) is reported as 30, and a relative point-to-point command

is issued:

```
PTP/r axis_index, 510
```

Internal calculations determine that the target position is $(30 + 510) \% 360 = 180$, motion will be in positive direction passing through 120 point to 180.

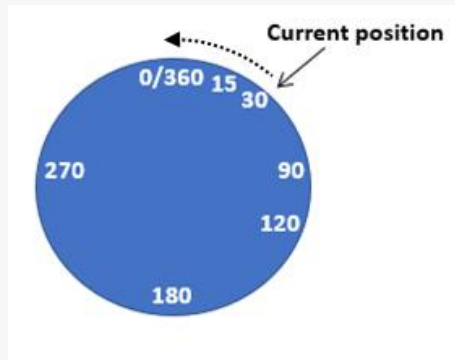


Example 2:

Assume $SLPMIN(axis_index)=0$ and $SLPMAX(axis_index)=360$, $FPOS(axis_index)$ is reported as 30, and a point-to-point command is issued:

```
PTP axis_index, 0
```

Internal calculations determine that the target position is $0 \% 360 = 0$, motion will be in negative direction passing through 15 point to 0.



Tag

417

Comments

The **MFLAGS(axis).#MODULO** bit needs to be set(1) for the axis to operate as a modulo axis.

All motion commands are calculated such that the resulting motion spans less than the Modulo range ($SLPMAX(axis) - SLPMIN(axis)$).

Axis modulo mode will not change the **EPOS** value as it does for **RPOS** and **FPOS**.

If an axis modulo mode has been turned off (**MFLAGS(axis).#MODULO=0**), then, **EPOS** is updated with the current **FPOS** value (by definition, **EPOS = FPOS** in a non-modulo mode).

Related ACSPL+ Variables

SLPMIN, SLPMAX, MFLAGS.#MODULO

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.1.11 PEGQUE

Description

PEGQUE is an integer array with one element for each PEG engine in the system, the elements of which store the current state of PEG FIFO for that engine (the items count in the FIFO) . The parameter is updated as long as its value is different from 0.

Syntax

PEGQUE

Arguments

Tag

371

Comments

This variable is supported in version 3.00 and higher.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger

3.1.12 SETTLE

Description

SETTLE is a real array, with one element for each axis in the system, and is used for setting the Settling Time, it controls **MST(axis_index).#INPOS**.

Syntax

SETTLE(axis_index) = value

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 1.79769e+308, Default = 0. |

Tag

123

Comments

When the motor is not moving, the controller compares the position error (**PE** value) and the target envelope (**TARGRAD** value) every MPU cycle. **#INPOS** is raised when **PE** drops to within the range (-**TARGRAD**, +**TARGRAD**) and remains within the range for a period of time equal or greater than **SETTLE**.

If the motor starts to move or **PE** goes out of the range, **MST**(axis_index).**#INPOS** is cleared.

Accessibility

Read-Write



SETTLE values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

TARGRAD, **MST**, **PE**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.13 SLPMAX**Description**

SLPMAX is a real array, with one element for each axis in the system, the elements of which are used for storing the maximum range of a modulo axis.

Syntax

SLPMAX(axis_index) = value

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from -1.79769e+308 to 1.79769e+308, Default = 8000. |

Tag

194

Comments

SLPMAX stores the maximum range of a modulo axis, see [MFLAGS](#). **#MODULO** (bit 29). **SLPMAX** can be changed only when the motor is disabled.

Accessibility

Read-Write



SLPMAX values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[SLPMIN](#), [EOFFS](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.14 SLPMIN

Description

SLPMIN is a real array, with one element for each axis in the system, the elements of which are used for storing the minimum range of a modulo axis.

Syntax

SLPMIN(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from -1.79769e+308 to 1.79769e+308, Default = 0. |

Tag

195

Comments

SLPMIN stores the minimum range of a modulo axis, see [MFLAGS](#). **#MODULO** (bit 29). **SLPMIN** can be changed only when the motor is disabled.

Accessibility

Read-Write



SLP_{MIN} values cannot be modified if protection is applied to this variable through
 SPiPlus MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Variables

SLP_{MAX}, EOFFS

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.15 STEPF

Description

STEPF is a real array, with one element for each axis in the system, and is used for defining the ratio between one stepper pulse and user units. See the *SPiPlus Setup Guide*, Stepper Drive section for more information.

Syntax

STEPF(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 1e-015 to 1e+015, Default = 1. |

Tag

129

Comments

STEPF = 1 (default) means that motion is performed in motor steps. For example,

PTP/R 0,320

will move the motor by 320 steps from the current position.

If another unit is required for motion programming, the user must configure an appropriate value for **STEPF**. For example, a controlled plant provides a gear ratio of 500 motor pulses per inch. If the motion programming unit must be provided in inches, the configured **STEPF** value must be 0.002 (1/500).

Accessibility

Read-Write



STEPF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.16 STEPW

Description

STEPW is a real array, with one element for each axis in the system, and is used for defining the pulse width, in milliseconds, for the stepper motor. See the *SPiiPlus Setup Guide*, Stepper Drive section for more information.

Syntax

STEPW(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from $12.0 \cdot 10^{-6}$ (120 nanoseconds) to 0.050 (50 microseconds), Default = 0.001. |

Tag

130

Comments

The value defines the width of the pulses generated on the pulse output for stepper control.

Accessibility

Read-Write



STEPW values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.1.17 TARGRAD

Description

TARGRAD is a real array, with one element for each axis in the system, and is used for defining the parameters for **MST**(axis_index).**#INPOS**.

Syntax

TARGRAD(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 2.22507e-308, 1.79769e+308, Default = 1. |

Tag

132

Comments

When the motor is enabled but in a standstill position, the controller compares **PE** to the target envelope (**TARGRAD**) each MPU cycle. **#INPOS** = 1 when **PE** moves into the defined range (-**TARGRAD**, +**TARGRAD**) and remains within that range for a period of time equal or greater than defined by **SETTLE**.

If the motor starts to move or goes out of range, **#INPOS** is cleared.

Accessibility

Read-Write



TARGRAD values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

SETTLE, **MST**(axis_index).**#INPOS**, **PE**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.2 Brake Variables

The Brake variables are:

| Name | Description |
|--------------------------|------------------------------------------------------------------------------------------------------|
| BOFFTIME | Brake Deactivation Time |
| BONTIME | Brake Activation Time |
| MBRKROUT | Set the supplier for the mechanical break output signal of an axis as a specified digital output bit |
| VELBRK | Braking Velocity |

3.2.1 BOFFTIME

Description

BOFFTIME is a real array, with one element for each axis in the system, and is used for specifying the brake release time in milliseconds.

Syntax

BOFFTIME(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from -1.79769e+308 to 1.79769e+308, Default = 50. |

Tag

9

Accessibility

Read-Write



BOFFTIME values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Comments

See the *ACSPL+ Programmer's Guide* for information about using a mechanical brake on system startup.

Related ACSPL+ Commands

[ENABLE/ENABLE ALL](#) - The brake is deactivated automatically when the ENABLE command is executed.

Related ACSPL+ Variables

[MFLAGS](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.2.2 BONTIME

Description

BONTIME is a real array, with one element for each axis in the system, and is used for specifying the brake engagement time in milliseconds.

Syntax

BONTIME(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from -1.79769e+308 to 1.79769e+308, Default = 50. |

Tag

10

Accessibility

Read-Write



BONTIME values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Comments

See the *ACSPL+ Programmer's Guide* for information about using a mechanical brake on system startup.

Related ACSPL+ Commands

[DISABLE/DISABLEALL](#), [FCLEAR](#), [DISABLEALL](#)



The brake is activated automatically when the **DISABLE** command is executed.

Related ACSPL+ Variables

[MERR](#), [FPOS](#), [RPOS](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.2.3 MBRKROUT

Description

MBRKROUT is an integer array, with one element for each axis in the system, and is used for setting the supplier for the mechanical brake output signal of an axis as a specified digital output bit(**ACSPL+ OUT**).

Syntax

MBRKROUT(Axis_Index) = value

Arguments

| | |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis | Designates the specific axis. Valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | <p>Value is a 4-digit number (decimal): <NN00> where:</p> <ul style="list-style-type: none"> > NN - digital output index (00-99) > 00 – specified input bit (00-31) <p>The default value is -1, in which case the mechanical brake default assignment will be used.</p> <p>If Value is set to -2, the assignment for the axis is canceled and the output reverts to standard digital output behavior</p> |

Tag

381

Comments

This variable can be saved to flash memory. In other words, the mapping of a mechanical brake output using this variable will automatically occur on controller boot without the need to reconfigure it .

The following errors are supported:

- > Error 3330: "Invalid value, digital output index should range between 0-99 and bit index should range between 0-31"
- > Error 3332: "This output is already mapped as a mechanical brake to a different axis"
- > The ACSPL+ variable bit **MFLAGS**(Axis_Index). **#BRAKE** should be set to 1 in order to enable the mechanical brake for the specified axis.

This variable is supported in V3.03 and higher.

Examples

```
MBRKROUT(0) = 1000 ! output index 10 and bit 0
                  ! supplies the mechanical brake signal for axis 0.
```

```
MBRKROUT(0) = -2 ! Cancels assignment for axis 0. In other words,
                  ! the output reverts to general purpose digital output
```

Related ACSPL+ Variables**MFLAGS****Accessibility**

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.2.4 VELBRK**Description**

VELBRK is a real array, with one element for each axis in the system, and is used for defining dynamic braking.

Syntax

VELBRK(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 1.79769e+308, Default = 0. |

Tag

140

Comments

If **MFLAGS**(*axis_index*).#**DBRAKE** is = 1, the controller will apply dynamic braking to stop the motor when the axis is disabled and the feedback velocity is less than the predefined value of **VELBRK**.

Accessibility

Read-Write



VELBRK values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

MFLAGS(*axis_index*).#**DBRAKE**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3 Feedback Variables

The Feedback variables are:

| Name | Description |
|----------|------------------------------------------------------------------------------|
| E_AOFFS | Sets user-defined offset for absolute encoder |
| E_FLAGS | Configuration bits for absolute encoder. |
| E2_AOFFS | Sets user-defined offset for absolute encoder secondary feedback |
| E2_FLAGS | Defines the Encoder Direction Inverse |
| E_FREQ | Encoder frequency |
| E2_FREQ | Defines the maximum encoder pulse frequency (in MHz) for secondary feedback. |
| E_PAR_A | Data transmission actual frequency. |
| E2_PAR_A | Sets the encoder data transmission frequency in MHz |
| E_PAR_B | Data control CRC code. |
| E2_PAR_B | Sets the encoder data control CRC code |
| E_PAR_C | The interval (in microseconds) of encoder position reading. |
| E2_PAR_C | Sets the interval (in microseconds) of encoder position reading. |
| E_PAR_D | Number of status bits (LSB) in the real-time position data (SLABITS). |
| E2_PAR_D | Defines the number of status bits (LSB) in the real-time position data |
| E_PAR_E | MASK for setting error bits. |
| E2_PAR_E | Defines a mask for setting error bits. |
| E_SCMUL | Encoder Sin-Cos Multiplier |
| E2_SCMUL | Specifies the Sin-Cos multiplication factor for the encoder |

| Name | Description |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| E_TYPE | Encoder Type |
| E2_TYPE | Defines the encoder type for the secondary feedback |
| EFAC | Encoder Factor |
| E2FAC | Secondary Encoder Factor |
| EOFFS | Axis Encoder Offset |
| E2OFFS | Axis Encoder Offset for Secondary Encoder |
| EPOS | Shows the encoder feedback |
| HOMEDF | Integer array defining homing method for each axis |
| HOMEVELI | Array of doubles defining the default homing velocity used by each axes for index search during ACSPL+ HOME command |
| HOMEVELL | Array of doubles defining the default homing velocity used for limit search during ACSPL+ HOME command |
| FVFIL | Primary Feedback Velocity Filter |
| F2ACC | Defines the feedback acceleration value of the axis |
| RVFIL | Reference Velocity Filter |
| SCSOFFS | Defines the sine offset. |
| SCCOFFS | Defines the cosine offset. |
| SC2COFFS | Defines the Sin-Cos Cosine offset |
| SC2GAIN | A Sin-Cos encoder gain compensation variable used to compensate the Cosine signal for an improper amplitude relative to the Sine signal |

| Name | Description |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| SC2PHASE | A Sin-Cos encoder phase compensation variable and is used to compensate the Cosine signal for an improper phase difference relative to the Sine signal |
| SC2SOFFS | Defines Sin-Cos Sine offset |
| SLEBIASA | Defines firmware sine offset. |
| SLEBIASB | Defines hardware cosine offset. |
| SLEBIASC | Defines the Sin-Cos encoder's hardware compensation for the Sine offset |
| SLEBIASD | Defines the Sin-Cos encoder's hardware compensation for the Cosine offset |
| SLABITS | Total number of absolute position bits for an absolute encoder |
| S2LABITS | Sets the total number of absolute position bits for an absolute encoder connected to a secondary feedback |
| SCGAIN | Feedback gain compensation variable |
| SCPHASE | Feedback phase compensation variable |

3.3.1 E_AOFFS

Description

E_AOFFS is an integer array, with one element for each axis in the system, and is used for setting user-defined offset for absolute encoder.

Comments

Modifying **E_AOFFS** causes absolute encoder initialization.

The **EOFFS** variable is being modified according to **E_AOFFS**' value:

EOFFS=EOFFS-E_AOFFS.

Tag

300

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.2 E_FREQ

Description

E_FREQ is an integer array, with one element for each axis in the system, and is used for defining the maximum encoder pulse frequency (in MHz).

Syntax

E_FREQ(*axis_index*) = *value*

Arguments

| | |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value of each member can be one of: <ul style="list-style-type: none"> > 2 (shown as 2MHz but practically is 2.5MHz, default for an analog Sin-Cos encoder) > 20 (default for a digital encoder) > 60 |

Tag

27

Comments

The encoder is represented in the controller as a synchronous state machine that is activated by a clock, with programmable frequency in the SPiiPlus processor.

E_FREQ provides three optional clock rates that define the maximum encoder pulse frequency (in MHz) measured after an internal 4x multiplication.

In general, using a higher **E_FREQ** enables to read a higher rate of encoder input. However, the electrical noise immunity is reduced and Encoder Error **FAULT** might occur. Per case, it is recommended to use the lowest possible **E_FREQ** that does not generate an Encoder Error **FAULT**. In case of an Encoder Error, do **FCLEAR** (*axis_index*) and try a higher **E_FREQ** value.

For more information, see the "Encoder Input Clock" section in the *SPiiPlus Setup Guide*.

Accessibility

Read-Write



E_FREQ values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.3 E2_AOFFS**Description**

E2_AOFFS is an integer array, with one element for each axis in the system, and is used for setting user-defined offset for absolute encoders for the secondary feedback.

Comments

Modifying E2_AOFFS causes absolute encoder initialization.

The E2OFFS variable is modified according to E2_AOFFS' value:

E2OFFS=E0FFS-E2_AOFFS.

Tag

377

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.4 E2_FREQ**Description**

E2_FREQ is an integer array, with one element for each axis in the system, and is used for defining the maximum encoder pulse frequency (in MHz) for secondary feedback.

Syntax

E2_FREQ(axis_index)=value

Arguments

| | |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | Value of each member can be one of the following: <ul style="list-style-type: none"> > 2 (shown as 2MHz but practically is 2.5 MHz, default for an analog Sin-Cos encoder) > 20 (default for a digital encoder) > 60 |

Tag

303

Comments

The encoder is represented in the controller as a synchronous machine that is activated by a clock, with programmable frequency in the SPiiPlus processor.

E2_FREQ provides three optional clock rates that define the maximum encoder pulse frequency (in MHz) measured after an internal 4x multiplication.

In general, using a higher E2_FREQ enables to read a higher rate of encoder input. However, the electrical noise immunity is reduced and Encoder Error FAULT might occur. Per case, it is recommended to use the lowest possible E2_FREQ that does not generate an Encoder Error FAULT. In case of an Encoder Error, FCLEAR(axis_index) command should be executed and a higher E2_FREQ value should be set.

For more information, see the "Encoder Input Clock" section in the *SPiiPlus Setup Guide*.

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.5 E_FLAGS

Description

E_FLAGS is an integer array, with one element for each axis in the system. Each element contains different configuration bits for absolute encoder.

Syntax

E_FLAGS(axis_index) = value Arguments

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| bit_designator | The meanings of bit_designator are given in Table 3-4 . |

Table 3-4. E_FLAGS Bit Description

| Bit Name | No. | Description |
|-----------|-----|------------------------------------------------------------------------------------------|
| #ERRLOGIC | 0 | Encoder Error Logics |
| #UNSIGND | 1 | Unsigned Mode |
| #INVERSE | 2 | Encoder Direction Inverse |
| #HALLCORR | 3 | 0: continuous commutation correction is disabled 1: continuous commutation is enabled |

| Bit Name | No. | Description |
|-----------|-----|---------------------------------------------------------------|
| #GRAYCODE | 4 | 0: Gray Code mode is disabled 1: Gray Code mode is enabled |

Comments

#ERRLOGIC defines the status bits logic (0 or 1). The default value is 0. The Encoder 1 Error (#ENC) bit of the ACSPL+ **FAULTS** variable is triggered if the error is latched (based on the error bit logics).

If E_PAR_E is not set (0), **#ERRLOGIC**'s value has no meaning.

#UNSIGNED defines the Absolute Encoder Position mode. It can be unsigned (1) or signed (0). The default mode is signed for backward compatibility.

#INVERSE

When the bit is ON, the DSP is being notified and sends the position inverted. FW inverts the EOFFS variable as well.

The bits applies for all types of encoders.

The bit change event causes the FW to reinitialize the absolute encoder.

The bit cannot be changed if the axis is enabled.

Changing this bit may require changing the drive polarity (MFLAGS bit 13) or repeat commutation.

In case of a brushless motor, after the bit is changed the commutation will no longer be correct and the user should repeat the Adjuster commutation.

In case of a brush motor, the user should re-verify the drive polarity in the Adjuster by running Open Loop Verification.



Failure to repeat the Adjuster Commutation and Open Loop Verification may result critical position error, over current faults and in certain cases motor run away.

Tag

268

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteg

3.3.6 E2_FLAGS

Description

E2_FLAGS is an integer array, with one element for each axis in the system. Used for the secondary feedback. Each element contains different configuration bits for absolute encoder, in current version

it includes 3 bits.

E2_FLAGS Bit Description

| Bit Name | Number | Description |
|----------|--------|---------------------------|
| #ERLOGIC | 0 | Encoder Error Logics |
| #UNSIGND | 1 | Unsigned Mode |
| #INVERSE | 2 | Encoder Direction Inverse |

#ERRLOGIC defines the logics of the status bits – can be 0 or 1. If E2_PAR_E is not set (value equals to 0), ERRLOGIC value has no meaning. The default value is 0. Encoder 1 Error (#ENC) bit of the ACSPL+ FAULTS variable is triggered if error is latched (based on the error bit logics).

#UNSIGNED defines the Absolute Encoder Position mode – can be unsigned (1) or signed (0). The default mode is signed for backwards compatibility.

#INVERSE defines if the position is inverted on the DSP level. If the bit is set, the DSP sends the position inverted. FW inverts EOFFS variable.

Arguments

| | |
|----------------|-----------------------------------------------------------------------------------------------------|
| axis | The specific axis index. Valid numbers are: 0,1... up to the number of axes in the system, minus 1. |
| bit_designator | The meanings of bit_designator are defined in the table above. |

Tag

268

Comments

The bits applies for all types of encoders.

The bit change event causes the FW to reinitialize the absolute encoder.

Accessibility

Read-Write

Com Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.7 E_PAR_A

Description

E_PAR_A is used for setting the encoder data transmission actual frequency in MHz. It is a double array, with one element for each axis in the system.

Syntax

E_PAR_A(axis) = value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------|
| <i>axis</i> | The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1. |
| <i>value</i> | The encoder data transmission actual frequency in MHz ranging from 1.25 to 10. |



For the IDMsM/ECMsM/UDMsM products using the EnDAT encoder, only the following values are allowed for the **value** parameter: 0.1, 0.2, 1, 2, 4, 8, 16.



This variable is supported only for BiSS encoders (in all products) and EnDAT encoders (IDMsM/ECMsM/UDMsM products only).

Tag

248

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.8 E2_PAR_A

Description

E2_PAR_A is used for setting the encoder data transmission actual frequency in MHz. it is a double array, with one element for each axis in the system. Used for Secondary Feedback.

Syntax

E2_PAR_A (axis_index)=value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------------|
| <i>axis</i> | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| <i>value</i> | The encoder data transmission actual frequency in MHz ranging from 1.25 to 10. |



For the IDMsM/ECMsM/UDMsM products using the EnDAT encoder, only the following values are allowed for the **value** parameter: 0.1, 0.2, 1, 2, 4, 8, 16.



This variable is supported only for BiSS encoders (in all products) and EnDAT encoders (IDMsm/ECMsm/UDMsm products only).

Tag

304

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.9 E_PAR_B**Description**

E_PAR_B is used for setting the encoder data control CRC code. It is an integer array, with one element for each axis in the system.

Syntax**E_PAR_B**(axis) = value**Arguments**

| | |
|--------------|------------------------------------------------------------------------------------------------------|
| axis | The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1. |
| value | The encoder data control CRC code ranging from 0 to 255. |

Tag

267

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.10 E2_PAR_B**Description**

E2_PAR_B is used for setting the encoder data control CRC code. It is an integer array, with one element for each axis in the system. Used for Secondary Feedback.

Syntax

E2_PAR_B (axis_index)=value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------------|
| axis | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | The encoder data control CRC ranging from 0 to 255. |

Tag

305

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.11 E_PAR_C

Description

E_PAR_C is used for setting the interval (in microseconds) of encoder position reading. It is an integer array, with one element for each axis in the system. The default value is 0 which means 50 microseconds.

Syntax

E_PAR_C(axis) = value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------|
| axis | The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1. |
| value | The interval of encoder position reading in microseconds, valid range [0,..3] |

Valid Values

| | |
|----------|------------------|
| 0 | 50 microseconds |
| 1 | 100 microseconds |
| 2 | 200 microseconds |
| 3 | 400 microseconds |

Tag

258

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.12 E2_PAR_C

Description

E2_PAR_C is used for setting the interval (in microseconds) of encoder position reading. it is an integer array, with one element for each axis in the system. Used for Secondary Feedback. The default value is 0 which means 50 microseconds.

Syntax

E2_PAR_C (axis_index)=value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------------|
| <i>axis</i> | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| <i>value</i> | The interval of encoder position reading in microseconds valid range [0,..3] |

Valid Values

| | |
|---|---------------------------|
| 0 | 50 microseconds (default) |
| 1 | 100 microseconds |
| 2 | 200 microseconds |
| 3 | 400 microseconds |

Tag

306

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.13 E_PAR_D

Description

E_PAR_D defines number of status bits (LSB) in the real-time position data (**SLABITS**). The status bits include warning and error bits. These bits are not part of the real-time position.

E_PAR_D is represented as an integer array with the size of maximum axes; each element for each axis in the system. The default value is 0 which means no status bits at all. Maximum value is 16.

Syntax

E_PAR_D(axis) = value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------|
| <i>axis</i> | The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1. |
| <i>value</i> | Number of status bits; the range is [0, 16]. |

Tag

266

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.14 E2_PAR_D

Description

E2_PAR_D defines number of status bits (LSB) in the real-time position data. The status bits include warning and error bits. These bits are not part of the real-time position. it is an integer array, with one element for each axis in the system. Used for Secondary Feedback. The default value is 0 which means no status bits at all. Maximum value is 16.

Syntax

E2_PAR_D (axis_index)=value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------------|
| <i>axis</i> | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| <i>value</i> | Number of status bits; the range is [0,16]. |

Tag

307

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.15 E_PAR_E**Description**

E_PAR_E defines a mask for setting error bits. It is an integer array, with one element for each axis in the system. The default value is 0, which means no error bit is set. All bits that are not defined in the mask but covered by **E_PAR_D** are considered as warning bits.

Syntax**E_PAR_E**(axis) = value**Arguments**

| | |
|--------------|----------------------------------------------------------------------------------------------------------|
| <i>axis</i> | The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1. |
| <i>value</i> | Error bits MASK; minimum value is 0 (default – no error bits). The range is [0,2 ^{E_PA_E_D-1}] |

Tag

267

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.16 E2_PAR_E**Description**

E2_PAR_E defines a mask for setting error bits. It is an integer array, with one element for each axis in the system. Used for the secondary feedback. The default value is 0, which means no encoder error will be triggered. All bits that are not set in the mask but covered by **E2_PAR_D** are defined as warning bits.

Syntax**E2_PAR_E**(axis_index)=value

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------------|
| <i>axis</i> | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| <i>value</i> | Error bits MASK; minimum value is 0 (default – no error bits). The range is [0,2E2_PAR_D-1] |

Tag

308

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.17 E_SCMUL**Description**

E_SCMUL is an integer array, with one element for each axis in the system, and is used for specifying the Sin-Cos multiplication factor for the encoder.

Syntax**E_SCMUL(*axis_index*) = *value*****Arguments**

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2 to 16, Default = 10. |

Tag

28

Comments

E_SCMUL specifies the Sin-Cos multiplication factor as a power of 2. The maximum value of 16 corresponds to a multiplication of $65536 = 2^{16}$. The minimum value of 2 corresponds to a multiplication of $4 = 2^2$.

Accessibility

Read-Write



E_SCMUL values cannot be modified if protection is applied to this variable through
SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.18 E2_SCMUL

Description

E2_SCMUL is an integer array, with one element for each axis in the system, and is used for specifying the Sin-Cos multiplication factor for the encoder. Used for secondary feedback.

Syntax

E2_SCMUL (axis_index)=value

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | Value ranges from 2 to 16, default=10. |

Tag

30

Comments

E2_SCMUL specifies the Sin-Cos multiplication factor as a power of 2. The maximum value of 16 corresponds to a multiplication of $65536 = 2^{16}$. The minimum value of 2 corresponds to a multiplication of $4 = 2^2$.

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.19 E_TYPE

Description

E_TYPE is an integer array, with one element for each axis in the system, and is used for defining the encoder type.

Syntax**E_TYPE(*axis_index*) = *value*****Arguments**

| | |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | <p>value of each element can be one of:</p> <ul style="list-style-type: none"> > 0 - Up-down counter > 1 - Clock direction counter > 2 - Quadrature single-ended encoder > 3 - Quadrature encoder (Default) > 4 - 250 KHz / 500 KHz Sin-Cos encoder multiplier > 5 - HSSI encoder > 9 - Resolver > 10 - EnDat 2.2 > 11 - Smart-Abs > 12 - Panasonic > 13 - BiSS-A/B/C > 14 - Hiperface > 17 - BiSS-SSI > 18 - 10 MHz Sin-Cos encoder multiplier > 19- Sanyo Denki |

Tag

29

Comments

The most common encoder type is quadrature, which corresponds to the default value 3.

A value of 2 is supported by the following products:

- > UDMlc-x-048 (where x is either 2 or 4)
- > UDlIt-x / UDlhp-x (where x is either 2 or 4)

A value of 9 is supported only by the following products:

- > SPiiPlus CMnt-x-320 (where x is either 1 or 2)
- > UDMpm-x-320 (where x is either 1 or 2)

A value of 18 is supported by the following products:

- > UDMnt-x (where x is either 1 or 2)
- > UDlhp-x (where x is either 2 or 4)

As a configuration variable, the variable can be changed only if the controller is in configuration mode.

Accessibility

Read-Write



E_TYPE values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

Related ACSPL+ Variables

None

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.20 E2_TYPE

Description

E2_TYPE is an integer array, with one element for each axis in the system, and is used for defining the encoder type for the secondary feedback.

Syntax

E2_TYPE(axis_index)=value

Arguments

| | |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | <p>value of each member can be one of the following:</p> <ul style="list-style-type: none"> > 0 – not defined > 1 - Clock direction counter > 2 - Quadrature single-ended encoder > 3 - Quadrature encoder (Default) > 4 - 250 KHz / 500 KHz Sin-Cos encoder multiplier > 5 - HSSI encoder > 9 - Resolver > 10 - EnDat 2.2 > 11 - Smart-Abs > 12 - Panasonic > 13 - BiSS-C > 14 – Hiperface > 17 - SSI > 18 - 10 MHz Sin-Cos encoder multiplier > 19 – Sanyo Denki |

Tag

31

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.21 EFAC

Description

EFAC is a real array, with one element for each axis in the system, and is used for defining a factor between the raw feedback in encoder counts and the **FPOS** value calculated by the controller.

Syntax

EFAC(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value of each member ranges between 1e-15 to 1e+15, Default = 1. |

Tag

36

Comments

When reading the feedback position from the SP, the controller executes feedback transform according to the formula:

$$FPOS = FP * EFAC + EOFFS$$

where **FPOS** is the controller feedback position in user units, **FP** is an SP-calculated feedback position in encoder counts, **EFAC** is a user-defined value of the corresponding **EFAC** factor, and **EOFFS** represents an offset.

As a configuration variable, the **EFAC** value is normally defined by **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



EFAC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

All standard variables that are based on position units.

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.22 E2FAC**Description**

E2FAC is a real array, with one element for each axis in the system, and is used for defining a factor between the secondary raw feedback in encoder counts and the **F2POS** value calculated by the controller.

Syntax

E2FAC(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value of each member ranges between 1e-15 to 1e+15, Default = 1. |

Tag

32

Comments

When reading the secondary feedback position from the SP, the controller executes feedback transform according to the formula:

$$F2POS = FP2 * E2FAC + E2OFFS$$

where **F2POS** is the secondary controller feedback position in user units, **FP2** is an SP-calculated secondary feedback position in encoder counts, **E2FAC** is a user-defined value of the corresponding **E2FAC** factor, and **E2OFFS** represents an offset.

As a configuration variable, the **E2FAC** value is normally defined by **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



E2FAC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

F2POS, **E0FFS**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.23 EOFFS

Description

EOFFS is a real array, with one element for each axis in the system, and is used for defining the offset between the raw feedback from the encoder counts and the **FPOS** value calculated by the controller.

Syntax

EOFFS(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value can be any integer. |

Tag

38

Comments

EOFFS provides the offset between the raw feedback in encoder counts and the **FPOS** value calculated by the controller. The value of **EOFFS** changes when the set command defines a new origin for an axis.

When reading the feedback position from the SP, the controller executes feedback transform according to the formula:

$$FPOS = FP * EFAC + EOFFS$$

where **FPOS** is the controller feedback position in user units, **FP** is an SP-calculated feedback position in encoder counts, **EFAC** is a user-defined value of the corresponding EFAC factor, and **EOFFS** represents an offset.

Accessibility

Read-Only

Related ACSPL+ Variables

[EFAC](#), [FPOS](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.3.24 E2OFFS

Description

E2OFFS is a real array, with one element for each axis in the system, and is used for defining the offset between the raw feedback from the secondary encoder counts and the **FPOS** value calculated by the controller.

Syntax

E2OFFS(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value can be any integer. |

Tag

34

Comments

E2OFFS provides the offset between the raw feedback from the secondary encoder (in encoder counts) and the **F2POS** value calculated by the controller. The value of **E2OFFS** changes when the set command defines a new origin for the axis's secondary feedback.

When reading the secondary feedback position from the SP, the controller executes feedback transform according to the formula:

$$F2POS = FP2 * E2FAC + E2OFFS$$

where **F2POS** is the secondary controller feedback position in user units, **FP2** is an SP-calculated secondary feedback position in encoder counts, **E2FAC** is a user-defined value of the corresponding **E2FAC** factor, and **E2OFFS** represents an offset.

Accessibility

Read-Only

Related ACSPL+ Variables

[E2FAC](#), [F2POS](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.3.25 EPOS

Description

EPOS is a real array, one element for each feedback, and is used for showing the encoder feedback. Not affected by Gantry mode.

Comments

EPOS value is affected by ACSPL+ **SET** command only when applied to **FPOS** variable with the same index (a relevant offset is being added to EPOS value).

EPOS variable can be useful for displaying the encoder position in Gantry mode.

Tag

299

Accessibility

Read only

Com Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.3.26 FVFIL**Description**

FVFIL is a real array, with one element for each axis in the system, and is used for setting the intensity (in %) of the filter that the controller uses when calculating **FVEL**.

Syntax

FVFIL(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 0 to 100, Default = 30. |

Tag

54

Comments

FVFIL = 0 corresponds to no filtering. In this case the controller calculates **FVEL** as the derivative of the **FPOS** variable:

$$\Delta = (FPOS_n - FPOS_{n-1}) * K$$

$$FVEL_n = \Delta$$

where K is a scaling factor that reduces **FVEL** to user units per second.

As the **FPOS** value is supplied by a discrete physical sensor, the **FVEL** value calculated without filtering contains a considerable amount of noise.

Non-zero value of **FVFIL** provides additional filtering in the **FVEL** calculation according to the formula:

$$FVEL_n = \Delta * ((1 - FVFIL) / 100) + FVEL_{n-1} * (FVFIL / 100)$$

Accessibility

Read-Write



FVFILE values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

FVEL

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.27 F2ACC**Description**

F2ACC is a real array, with one element for each axis in the system, and is used for defining the feedback acceleration value of the axis. Used for secondary feedback.

Tag

317

Accessibility

Read-Only

Com Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.3.28 HOMEDEF**Description**

HOMEDEF is an integer array with one element for each axis in the system, and defines the default homing method for an axis used by the ACSPL+ **HOME** command.

Default value is 0, and should be set to a valid homing method number.

Syntax

```
HOMEDEF(Axis_Index) = Value
```

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------|
| Axis_ Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|----------------|---------------------------------------------------------------------------------------------------------------|

Homing Methods

Table 3-5. Homing Methods

| Method Number | Explanation |
|---------------|----------------------------------------------------------------------|
| 1 | Homing Method 1: Homing on the negative limit switch and index pulse |
| 2 | Homing Method 2: Homing on positive limit switch and index pulse |
| 17 | Homing Method 17: Homing on Negative Limit Switch |
| 18 | Homing Method 18: Homing on Positive Limit Switch |
| 33/34 | Homing Method 33 and 34: Homing on the index pulse |
| 37 | Homing Method 37: Homing on current position |
| 50 | Homing Method 50: Negative Hard Stop and index pulse (ACS Specific) |
| 51 | Homing Method 51: Positive Hard Stop and index pulse (ACS Specific) |
| 52 | Homing Method 52: Negative Hard Stop (ACS Specific) |
| 53 | Homing Method 53: Positive Hard Stop (ACS Specific) |

TAG

358

Comments

HOMEDEF should be used with the **HOME** command. **HOME**(<axis>) will use the homing method based on HOMEDEF variable.\

Related ACSPL+ Commands

HOME

Accessibility

Read-Write

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.3.29 HOMEVELI

Description

HOMEVELI is a double array with one element for each axis in the system, and defines the default homing velocity used for index search during ACSPL+ HOME command.

Default value is 0. If the value is 0, the velocity is calculated based on the **HomingVel** parameter.

Syntax

```
HOMEVELL(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 0 to 1e100. |

Tag

360

Comments

HOMEVELL should be used with the **HOME** command. The following homing methods are affected by this parameter: 17,18,33,34,50,51

This variable is supported in version 3.00 and higher.

Related ACSPL+ Commands and Variables

HOME, HOMEDEF, HOMEVELL

Accessibility

Read-Write

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.3.30 HOMEVELL

Description

HOMEVELL is a double array with one element for each axis in the system, and defines the default homing velocity used for limit search during ACSPL+ **HOME** command. Default value is 0.

If the value is 0, the velocity is calculated based on the HomingVel parameter.

Syntax

```
HOMEVELL(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 0 to 1e100. |

Tag

359

Comments

HOMEVELL should be used with the HOME command. The following homing methods are affected by this parameter: 1,2,17,18

This variable is supported in version 3.00 and higher.

Related ACSPL+ Commands and Variables

HOME, HOMEDEF, HOMEVELI

Accessibility

Read-Write

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.3.31 RVFIL

Description

RVFIL is a real array, with one element for each axis in the system, and is used for specifying the power of the filter that the controller uses to calculate RVEL.

Syntax

RVFIL(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value is a percent ranging from 0 to 100, Default = 0. |

Tag

110

Comments

The value is specified as a percent where 0 means no filtering. In this case the controller calculates RVEL as the first difference of RPOS as follows:

$$\Delta = (RPOS - RPOS_{n-1}) * K$$

$$RVEL_n = \Delta$$

where K is a scaling factor that translates RVEL to position units per second.

A non-zero value for RVFIL provides additional filtering in the RVEL calculation according to the formula:

$$RVEL_n = \Delta * (1 - (RVFIL/100)) + (RVEL_{n-1} * (RVFIL/100))$$

The default value of **RVFIL** is zero. No filtering is required as long as the axis motion is not a **MASTER-SLAVE** motion. In this case **RPOS** is a calculated value and the first difference provides a smooth approximation of velocity.

If an axis is involved in a **MASTER-SLAVE** motion, **RPOS** usually contains a signal from a discrete physical sensor that causes a certain amount of noise in the first difference. Increase the value of **RVFIL** if a smoother approximation is required.

Accessibility

Read-Write



RVFIL values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[MASTER](#)

Related ACSPL+ Variables

[RVEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.32 SCSOFFS

Description

SCSOFFS is a real array, with one element for each axis in the system, and is used for defining a Sin-Cos encoder's software compensation for the Sine offset.

Syntax

SCSOFFS(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | <i>value</i> designates the offset ranging from -32766 to 32766. |

Tag

202

Comments

The digital range corresponds to ± 0.5 Volts. This number is used to modify the offset of the Sin-Cos encoder signal related to the axis. The ratio between this number and the offset is 9.6.

Accessibility

Read-Write



SCSOFFS values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.33 SCCOFFS**Description**

SCCOFFS is a real array, with one element for each axis in the system, and is used for defining a Sin-Cos encoder's software compensation for the Cosine offset.

Syntax

SCCOFFS(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | <i>value</i> designates the offset ranging from -32766 to 32766. |

Tag

203

Comments

The digital range corresponds to ± 0.5 Volts. This number is used to modify the offset of the Sin-Cos encoder signal related to the axis. The ratio between this number and the offset is 9.6.

Accessibility

Read-Write



SCCOFFS values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.34 SC2COFFS

Description

SC2COFFS is a real array, with one element for each axis in the system, and is used for defining the Sin-Cos Cosine offset. Used for secondary feedback.

Syntax

SC2COFFS (axis_index)=value

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | value designates the offset ranging from -32766 to 32766. |

Tag

311

Comments

The digital range corresponds to ± 0.5 Volts. This number is used to modify the offset of the Sin-Cos encoder signal related to the axis. The ratio between this number and the offset is 9.6.

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.35 SC2GAIN

Description

SC2GAIN is a real array, with one element for each axis in the system, and is a Sin-Cos encoder gain compensation variable used to compensate the Cosine signal for an improper amplitude relative to the Sine signal. Used for secondary feedback.

Syntax

SC2GAIN (axis_index)=value

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | value ranges between 0.5 and 1.5; Default: 1. |

Tag

312

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.36 SC2PHASE**Description**

SC2PHASE is a real array, with one element for each axis in the system, and is a Sin-Cos encoder phase compensation variable and is used to compensate the Cosine signal for an improper phase difference relative to the Sine signal. Used for Secondary Feedback.

Syntax**SC2PHASE** (axis_index)=value**Arguments**

| | |
|-------------------|------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | value in degrees, the value ranges between -15 and 15; default is 0. |

Tag

313

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.37 SC2SOFFS**Description**

SC2SOFFS is a real array, with one element for each axis in the system, and is used for defining the Sin-Cos Sine offset. Used for secondary feedback.

Syntax**SC2SOFFS** (axis_index)=value

Arguments

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates the offset ranging from -32766 to 32766. |

Tag

314

Comments

The digital range corresponds to ± 0.5 Volts. This number is used to modify the offset of the Sin-Cos encoder signal related to the axis. The ratio between this number and the offset is 9.6.

Accessibility

Read-Write

Com Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.38 SLEBIASA

SLEBIASA is a real array, with one element for each axis in the system, and is used for defining a Sin-Cos encoder's hardware compensation for the Sine offset.

Syntax**SLEBIASA(*axis_index*) = *value*****Arguments**

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates the offset ranging from -50 to 50; Default: 0. |

Tag

164

Comments



SLEBIASA performs the same function as **SCSOFFS** with the difference being that it corresponds to hardware offset compensation of encoder signals. Only certain SPiiPlus products: SPiiPlusNT-HP, CMnt, and UDMpc have an option for hardware offset compensation. Hardware compensation has some advantages over software compensation, such as, the possibility to get analog signals out of saturation, and making hardware based features like PEG more accurate.

SLEBIASA is normally set as part of the SPiiPlus MMI Application Studio **Sin-Cos Encoder Analyzer** tool routine. The tool first calculates the software compensation variable (**SCSOFFS**), and then writes the final value to the hardware variable **SLEBIASA** and resets the software one. Then during verification phase new value for **SCSOFFS** is found and stored along with previously found **SLEBIASA**.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.39 SLEBIASB

SLEBIASB is a real array, with one element for each axis in the system, and is used for defining a Sin-Cos encoder's hardware compensation for the Cosine offset.

Syntax

SLEBIASB(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value designates the offset ranging from -50 to 50; Default: 0. |

Tag

165

Comments



SLEBIASB performs the same function as **SCCOFFS** with the difference being that it corresponds to hardware offset compensation of encoder signals. Only certain SPiiPlus products: SPiiPlusNT-HP, CMnt, and UDMpc have an option for hardware offset compensation. Hardware compensation has some advantages over software compensation, such as, the possibility to get analog signals out of saturation, and making hardware based features like PEG more accurate.

SLEBIASB is normally set as part of the SPiiPlus MMI Application Studio **Sin-Cos encoder analyzer** tool routine. The tool first calculates the software compensation variable (**SCCOFFS**), and then writes the final value to the hardware variable **SLEBIASB** and resets the software one. Then during verification phase new value for **SCCOFFS** is found and stored along with previously found **SLEBIASB**.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.40 SLEBIASC

Description

SLEBIASC is a real array, with one element for each axis in the system, and is used for defining the Sin-Cos encoder's hardware compensation for the Sine offset. Used for secondary feedback.

Syntax

SLEBIASC (axis_index)=value

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| value | value designates the offset ranging from -50 to 50; default: 0. |

Tag

315

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.41 SLEBIASD

Description

SLEBIASD is a real array, with one element for each axis in the system, and is used for defining the Sin-Cos encoder's hardware compensation for the Cosine offset. Used for secondary feedback.

Syntax

SLEBIASD (axis_index)=value

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates the offset ranging from -50 to 50; default: 0. |

Tag

316

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.3.42 SLABITS

Description

SLABITS is an integer array, with one element for each axis in the system, and is used for setting the total number of absolute position bits for an absolute encoder.

Syntax

SLABITS(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 16 to 50, Default = 49. |

Tag

220

Comments

SLABITS is used for setting the total number of absolute position bits for an absolute encoder. This is the sum of the multi-turn resolution bits and the turn resolution bits. For example, for an encoder with 17 bits turn resolution and 16 bits multi-turn resolution, **SLABITS** should be set to 33.

Accessibility

Read-Write



SLABITS values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[E_TYPE](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.43 S2LABITS

Description

S2LABITS is an integer array, with one element for each axis in the system, and is used for setting the total number of absolute position bits for an absolute encoder connected to a secondary feedback.

Syntax

S2LABITS(axis_index)=value

| | |
|-------------------|---------------------------------------------------------------------|
| <i>axis_index</i> | Specific axis, range: 0 up to number of axis in the system minus 1. |
| <i>value</i> | Value ranges between 16 to 50, default is 49. |

Comments

S2LABITS is used for setting the total number of absolute position bits for an absolute encoder. This is the sum of the multi-turn resolution bits and the turn resolution bits. For example, for an encoder with 17 bits turn resolution and 16 bits multi-turn resolution, **SLABITS** should be set to 33.

Tag

310

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, Write Variable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.44 SCGAIN

SCGAIN is a real array, with one element for each axis in the system, and is a Sin-Cos encoder gain compensation variable used to compensate the Cosine signal for an improper amplitude relative to the Sine signal.

Syntax

SCGAIN(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0.5 and 1.5; Default: 1. |

Tag

204

Comments

The value of **SCGAIN** is normally set by the SPiPlus MMI Application Studio **Sin Cos Encoder Analyzer** tool routine when calculating the optimum encoder compensation.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.3.45 SCPHASE

SCPHASE is a real array, with one element for each axis in the system, and is a Sin-Cos encoder phase compensation variable and is used to compensate the Cosine signal for an improper phase difference relative to the Sine signal.

Syntax

SCPHASE(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value in degrees, the value of which ranges between -15 and 15; Default: 0. |

Tag

205

Comments

The value of **SCPHASE** is normally set by the SPiiPlus MMI Application Studio **Sin Cos Encoder Analyzer** tool routine when calculating the optimum encoder compensation.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.4 Axis State Variables

The Axis State variables are:

| Name | Description |
|--------|-----------------------------------------------------------------------|
| AST | Axis State |
| IND | Index Position |
| IST | Index State |
| AFLAGS | Mark Position |
| MARK | Secondary Mark Position |
| MST | Motor State |
| NST | Status of EtherCAT Sync and GPRT errors for each axis in the system. |
| | RMS current |
| ROFFS | Reads the offset calculated by the controller in the connect formula. |

3.4.1 AST

Description

AST is an integer array, with one element for each axis in the system, the elements of which contain a set of bits used for displaying the current axis state.

Syntax

[command] AST(axis_index).bit_designator

Arguments

| | |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>command</i> | Typical commands are DISP and the like. |
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. In the case of bit 2 (#PEGREADY) this parameter designates the PEG engine, not the axis. |
| <i>bit_designator</i> | A description of the AST bit designators is given in Table 3-6 . |

Table 3-6. AST Bit Descriptions

| Bit Name | No. | Description |
|-----------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #LEAD | 0 | 1 = axis is leading in a group |
| #DC | 3 | 1 = Axis data collection is in progress |
| #PEGREADY | 4 | 1 = all values are loaded and the Incremental/Random PEG is ready to respond to movement When referring to this bit, <i>axis_index</i> designates the PEG engine rather than the axis. |
| #MOVE | 5 | 1 = Axis is involved in a motion |
| #ACC | 6 | 1 = Axis in accelerating motion state |
| #BUILDUP | 7 | 1 = Segments build-up |
| #VELLOCK | 8 | 1 = Slave is synchronized to master in velocity lock mode - slave velocity strictly follows the master velocity. |
| #POSLOCK | 9 | 1 = Slave is synchronized to master in position lock mode - slave position strictly follows the master position. |
| #TRIGGER | 11 | 1 = Produces an interrupt to the host application, enabled by IENA.26 |
| #NEWSEGM | 16 | The controller sets the bit to inform that a new segment is required to be provided by the application. The bit is set starvation_margin ms before the starvation condition occurs. The starvation condition is indicated by #STARV bit. |

| Bit Name | No. | Description |
|-----------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #STARV | 17 | <p>The controller sets the bit to indicate starvation condition. The starvation condition means that there are not enough further segments to continue the motion with required velocity. In this case, the controller starts decelerating the motion with $\frac{1}{2}$ JERK in order to prevent motion discontinuity and avoid mechanical jerks. Once the application begins supplying segments at a sufficient rate, the controller returns the motion back to normal condition.</p> <p>Note, that often the starvation condition causes inefficient velocity generation and increases the time required for completing the required motion path.</p> |
| #ENCWARN | 18 | Indicates if there is an encoder warning. Cleared by the ACSPL+ FCLEAR command. |
| #ENC2WARN | 19 | Indicates if there is an secondary encoder warning. Cleared by the ACSPL+ FCLEAR command. |
| #INRANGE | 20 | Laser In Range |
| #LCTICKLE | 21 | <p>0: tickle mode is off</p> <p>1: tickle mode is active</p> |
| #LCMODUL | 22 | <p>0: modulation is off</p> <p>1: modulation is active</p> |
| #FOLLOWED | 23 | <p>0: Axis in regular mode</p> <p>1: Axis in slave mode and follows the profile generated by RTC6</p> |
| #HOLD | 24 | <p>0: hold is off</p> <p>1: hold is in progress</p> |
| #INHOMING | 25 | <p>0: homing is not in process</p> <p>1: homing is in process</p> |
| #DECOMPON | 26 | <p>0: dynamic error compensation is switched off</p> <p>1: dynamic error compensation is switched on</p> <p>The bit is set to 0 in the following cases:</p> <ul style="list-style-type: none"> > Calling ERRORMAPOFF function > Calling ERRORUNMAP function (in case there are no other active dynamic error compensation) |

| Bit Name | No. | Description |
|----------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <p>zones of the specified axis)</p> <ul style="list-style-type: none"> > During homing process initiated by HOME command (switched ON back at the end of the process) > Encoder Error / Encoder Not Connected faults > Changing one of the encoder-related parameters (E_TYPE, E_FREQ, E_SCMUL, SLPBITS, E_AOFSS) > Changing encoder routing (SLPROUT) |
| #INSHAPE | 27 | <p>0: Input Shaping not Active 1: Input Shaping Active</p> |
| #ENCPROC | 29 | <p>0: Encoder Initialization not in process 1: Encoder Initialization in process</p> |

Tag

7

Accessibility

Read-Only

Related ACSPL+ Commands[MASTER](#), [SLAVE](#)**Related ACSPL+ Variables**[MST](#)**COM Library Methods and .NET Library Methods**

ReadVariable, GetAxisState

C Library Functions

acsc_ReadInteger, acsc_GetAxisState

3.4.2 IND**Description**

IND is a real array, with one element for each axis in the system, the elements of which store the position of the last encountered encoder index in user-defined units. The variable operates in connection with [IST](#)(axis_index).#**IND**.

Tag

72

Comments

After power-up, `IST(axis_index).#IND` is reset and the value of `IND` is undefined because an index capture has not yet occurred. When the motor encounters an encoder index, `IST(axis_index).#IND` is raised and the current `FPOS` position is latched to `IND`.

Subsequent index values are ignored as long as `#IND` remains raised.

To resume the latching logic `IST(axis_index).#IND` must be explicitly cleared by the command `IST(axis_index).#IND=0`.

Accessibility

Read-Only



`IND` values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Variables

`IST`, `FPOS`

COM Library Methods and .NET Library Methods

`ReadVariable`, `GetIndexState`, `ResetIndexState`

C Library Functions

`acsc_ReadReal`, `acsc_GetIndexState`, `acsc_ResetIndexState`

3.4.3 `IST`

Description

`IST` is an integer array, with one element for each axis in the system, the elements of which contain a set of bits that indicate the state of the `IND` and the `MARK` variables for the given axis.

Syntax

`IST(axis_index).bit_designator = value`

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The <code>IST</code> has four bit designators: <ul style="list-style-type: none"> > <code>#IND</code> (bit 0) - Primary encoder index > <code>#IND2</code> (bit 1) - Secondary encoder index > <code>#MARK</code> (bit 2) - Mark 1 > <code>#MARK2</code> (bit 3) - Mark 2 |
| <i>value</i> | <code>value</code> can be zero or non-zero. |

Tag

79

Comments

The controller processes Index/Mark signals as follows: when an Index/Mark signal is encountered for the first time, the controller latches **FPOS** or **F2POS** to one of the variables **IND**, **MARK**, **M2ARK** and **sets the corresponding IST bit = 1**.

When finding an Index for the first time, the correct procedure is:

1. Start by setting the index flag to 1: **IST(axis).#IND=1**.

Then reset the flag to 0: **IST(axis).#IND=0**.

This puts the system in the correct mode for finding the Index.

As long as an **IST** bit is raised, the controller does not latch another value to the corresponding variable. To resume the latching logic, the user application must explicitly reset the corresponding **IST** bit to 0.

Accessibility

Read-Write

Related ACSPL+ Variables

FPOS, **F2POS**, **IND**, **MARK**, **M2ARK**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetIndexState, ResetIndexState

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_GetIndexState, acsc_ResetIndexState

3.4.4 M2ARK**Description**

M2ARK is a real array, with one element for each axis in the system, the elements of which store the position of the last encountered MARK2 signal in **IST(axis_index).#MARK2**.

Tag

84

Comments

After power-up **IST(axis_index).#MARK2** is reset and the value of **M2ARK** is undefined. When the motor encounters a MARK2 signal, the bit is raised and the current **FPOS** position is latched to **IST(axis_index).#MARK2**.

If the motion continues and the motor encounters another M2ARK signal, the new value is ignored as long as **IST(axis_index).#MARK2 = 1**.

To resume the latching logic, **IST(axis_index).#MARK2** must be explicitly cleared with the command **IST(axis_index).#MARK2=0**.

Example

```
ON IST (Axis) .#MARK2           ! When axis MARK is tripped
DISP "MARK Position:",M2ARK (Axis) ! Display the MARK position
IST (Axis) .#MARK2=0          ! Reset MARK flag
RET
```

Accessibility

Read-Only

Related ACSPL+ Variables

IST, FPOS, F2POS, MARK

IST, FPOS, F2POS, MARK

COM Library Methods and .NET Library Methods

ReadVariable, GetIndexState,

C Library Functions

acsc_ReadReal, acsc_GetIndexState

3.4.5 MARK

Description

MARK is a real array, with one element for each axis in the system, the elements of which store the position of the last encountered MARK1 signal in **IST(axis_index).#MARK**.

Tag

85

Comments

After power-up, **IST(axis_index).#MARK** is reset and the value of **MARK** is undefined. When the motor encounters a MARK1 signal, the bit is raised and the current **FPOS** position is latched to **IST(axis_index).#MARK**.

If the motion continues and the motor encounters another MARK1 signal, the new value is ignored as long as **IST(axis_index).#MARK=1**.

To resume the latching logic, **IST(axis_index).#MARK** must be explicitly cleared with the command **IST(axis_index).#MARK=0**.



MARK values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection.

Example

```
ON IST (Axis) .#MARK           ! When axis MARK is tripped
DISP "MARK Position:",MARK (Axis) ! Display the MARK position
IST (Axis) .#MARK=0          ! Reset MARK flag
RET
```

Accessibility

Read-Only

Related ACSPL+ Variables

IST, FPOS, F2POS, IND, M2ARK, IENA

COM Library Methods and .NET Library Methods

ReadVariable, GetIndexState

C Library Functions

acsc_ReadReal, acsc_GetIndexState

3.4.6 MST**Description**

MST is an integer array, with one element for each axis in the system. The elements of which contain a set of bits that display the current motor state, as given in [Table 3-7](#), for the given axis.

Syntax**MST(*node_index*).*bit_designator* = 1|0****Arguments**

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The meanings of bit_designator are given in 3.4.6 . |

Table 3-7. MST Bit Descriptions.

| Bit Name | No | Description |
|----------|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #ENABLED | 0 | 0: motor is disabled 1: motor is enabled. |
| #OPEN | 1 | 0: motor is operating with closed loop control 1: motor is operating with open loop control. |
| #INPOS | 4 | 0: Motor is moving or is out of range 1: Motor is not moving and has reached the target position (see variables TARGRAD and SETTLE) |
| #MOVE | 5 | 0 = Axis is not involved in a motion 1 = Axis is involved in a motion |
| #ACC | 6 | 0 = Motor is not accelerating 1 = Motor is accelerating. |
| #INTARGA | 25 | 0 = No motion has settled in Target Radius A |

| Bit Name | No | Description |
|----------|----|-------------------------------------------------------------------------------------------|
| | | 1 = Motion has settled in Target Radius A |
| #INTARGB | 26 | 0 = No motion has settled in Target Radius B 1 = Motion has settled in Target Radius B |
| #INTARGC | 27 | 0 = No motion has settled in Target Radius C 1 = Motion has settled in Target Radius C |

Tag

90

Accessibility

Read-Only

Related ACSPL+ Commands

All motion commands.

Related ACSPL+ Variables[FPOS](#), [F2POS](#), [APOS](#), [RPOS](#)**COM Library Methods and .NET Library Methods**

ReadVariable, GetMotorState

C Library Functions

acsc_ReadInteger, acsc_GetMotorState

Example

```
WHILE MST(0) .#MOVE
WAIT 300
```

3.4.7 RMSM**Description**

RMSM is a real array with one element for each axis in the system, the elements of which store the motor RMS current for an axis (in % of drive peak). The value ranges between 0 and 100.

Tag

363

Comments

This variable is supported in ADK versions 2.70 and higher.

Accessibility

Read-Only

ACSPL+ Variables

XRMSM, XRMSTM

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.4.8 RMSD

Description

RMSD is a real array, with one element for each axis in the system, the elements of which store the drive RMS current for an axis (in % of drive peak). The value ranges between 0 and 100.

Tag

362

Comments

This variable is supported in ADK versions 2.70 and higher.

Accessibility

Read-Only

Related ACSPL+ Variables

XRMSD, XRMSTD

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.4.9 NST

Description

NST is an integer array, with one element for each EtherCAT node in the system, each element of which contains a set of 2 bits. The variable enables users to differentiate between different causes of servo processor alarm faults.



An axis not associated to a physical drive will have a high Servo Processor Alarm fault.

Syntax

NST(*node_index*).*bit_designator* = 1|0

Arguments

| | |
|-----------------------|----------------------------------------------------------------------------------------------------------------|
| <i>node_index</i> | Designates the specific node, valid numbers are: 0, 1, 2, ... up to the number of nodes in the system minus 1. |
| <i>bit_designator</i> | The meanings of bit_designator are given in Table 3-8 . |

Table 3-8. NST Bit Description

| Bit Name | Bit No. | Description |
|----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #SYNC | 0 | Sync error |
| #GPRT | 1 | GPRT error |
| #MISSING | 2 | EtherCAT Node Missing |
| #SPDC | 3 | 0: SPDC data collection not active 1: SPDC data collection active |
| #FSYNC | 4 | FPGA Sync Fault |
| #SPRT | 5 | 0: Real-time data transfer process from the MPU to a corresponding Servo Processor is not active 1: Real-time data transfer process from the MPU to a corresponding Servo Processor is active |
| #LCI | 10 | LCI Fault |

Tag

229

Comments

If the #SYNC or #GPRT error bit is set in **NST**, there will be a network error in all axes, and servo processor alarm in all the axes related to the node related to the **NST**. These faults indicate a problem in the interface between the firmware and the node.

The setting of the #SYNC error bit means that one or more slaves are out of synchronization with the master.

The setting of the #GPRT error bit means that the queue (the size, of which, is 400) for the GPRT commands (commands that are sent by request) was full and some commands to be sent were lost. For example, for the SPiiPlusDC-LT-4 controller 8 such commands can be sent every cycle, these commands can be found in a reserved place in the EtherCAT telegram for the command1,...,command8.

The following commands will cause the NST.#SPRT bit to be 1:

- > **SPINJECT**
- > **SPRT**
- > **ASSIGNPEG/f**
- > **BPTP/2** (20 kHz motion profile)
- > **FOLLOW** (in case of customized servo algorithm for 20 kHz motion profile)

SPINJECT, **SPRT**, **ASSIGNPEG/f**, **BPTP/2** and **FOLLOW** are mutually exclusive, meaning only one of the features can be active at the given time. So the **NST.#SPRT** bit should be checked before using any of these commands.

FCLEAR for any axis associated with the node axis will reset all bits of the **NST** variable of that node.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadInteger

C Library Functions

acsc_ReadInteger

3.5 Safety Limits Variables

The Safety Limits variables are:

| Name | Description |
|---------|---------------------------------------------------|
| CERRA | Critical Position Error (Accelerating) |
| CERRI | Critical Position Error (Idle) |
| CERRV | Critical Position Error (Velocity) |
| DELI | Delay on Transition to Idle State |
| DELV | Delay on Transition to Velocity State |
| ERRA | Tolerable Error (Accelerating) |
| ERRI | Tolerable Error (Idle) |
| ERRV | Tolerable Error (Velocity) |
| SLLIMIT | Software Left Limit-feedback count down limit |
| SLLROUT | Sets the HW limits routing for the specified axis |
| SRLIMIT | Software Right Limit-feedback count up limit |
| XACC | Over Acceleration fault parameter |
| XCURCDB | Threshold of the current vector peak |
| XCURI | Maximum idle motor current |
| XCURK | Current limit during kill operation |
| XCURV | Maximum drive current during motion |
| XRMS | Drive RMS over current fault parameter |
| XRMSD | Drive RMS over current fault parameter |

| Name | Description |
|------------------------|----------------------------------------|
| XRMSM | Motor RMS over current fault parameter |
| XRMST | Drive RMS Current Time Constant |
| XRMSTD | Drive RMS Current Time Constant |
| XRMSTM | Motor RMS Current Time Constant |
| XSACC | Maximum slave axis acceleration |
| XVEL | Over Velocity fault parameter |

3.5.1 CERRA

Description

CERRA is a real array, with one element for each axis in the system, and is used for defining the Position Error criterion for acceleration/deceleration states.

Syntax

CERRA(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value of each member ranges between 2.22507e-308 and 1.79769e+308, Default = 1000. |

Tag

11

Comments

CERRA defines critical position error fault ([FAULT](#)(*axis_index*).#**CPE**) criterion when the motor is in acceleration or deceleration motion states.

As a configuration variable, the **CERRA** value is normally defined by **SPIIPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



CERRA values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#)(*axis_index*).#**CPE**

[CERRI](#), [CERRV](#), [DELI](#), [DELV](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.2 CERRI

Description

CERRI is a real array, with one element for each axis in the system, and is used for defining the critical Position Error when the motor is idle.

Syntax

CERRI(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value of each member ranges between 2.22507e-308 and 1.79769e+308, Default = 1000. |

Tag

12

Comments

As a configuration variable, the **CERRI** value is normally defined by **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



CERRI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#)(*axis_index*).#CPE

[CERRA](#), [CERRV](#), [DELI](#), [DELV](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.3 CERRV

Description

CERRV is a real array, with one element for each axis in the system, and is used for defining the critical Position Error when the motor is moving with constant velocity.

Syntax

CERRV(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value of each member ranges between 2.22507e-308 and 1.79769e+308, Default = 1000. |

Tag

13

Comments

As a configuration variable, the **CERRV** value is normally defined by **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



CERRV values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#)(*axis_index*).#CPE

[CERRA](#), [CERRI](#), [DELI](#), [DELV](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.4 DELV

Description

DELV is a real array, with one element for each axis in the system, and is used for defining the delay of transition to the Constant Velocity state.

Syntax

DELV(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value of each member ranges between 2.22507e-308 and 1.79769e+308, Default = 50. |

Tag

24

Comments

DELV is defined in msec and applies a delay when the motion state changes to a constant velocity state (**GPHASE**(axis_index) = 4).

DELV affects the following faults:

- > **FAULT**(axis_index).#PE
- > **FAULT**(axis_index).#CPE

Accessibility

Read-Write



DELV values cannot be modified if protection is applied to this variable through **SPiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

ERRA, **ERRI**, **ERRV**, **DELI**, **FAULT**.#CPE, **FAULT**.#PE

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.5 DELI

Description

DELI is a real array, with one element for each axis in the system, and is used for defining the delay of transition to the Idle state.

Syntax

DELI(axis_index) = value

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value of each member ranges between 2.22507e-308 and 1.79769e+308, Default = 50. |

Tag

23

Comments

DELI is defined in milliseconds and applies a delay when the motion state changes from any motion state to idle (**GPHASE**(axis_index) = 0 or 12).

DELI affects the following faults and current limits:

- > **FAULT**(axis_index).#**PE**
- > **FAULT**(axis_index).#**CPE**
- > **XCURI**(axis_index)
- > **XCURV**(axis_index)

Accessibility

Read-Write



DELI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

ERRA, **ERRI**, **ERRV**, **DELV**, **FAULT**(axis_index).#**CPE**, **FAULT**(axis_index).#**PE**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.6 E_ERR**Description**

E_ERR is an integer array for each axis. It contains the encoder error code that was identified during the encoder initialization process.

The encoder errors range from 5121 to 5128 and are latched in the **E_ERR** variable. The error codes are specified in [Table 6-5](#) in the Error Codes section.

Comments

This variable is supported in version 3.00 and higher.

Accessibility

Read-Only

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadInteger()

3.5.7 *ERRA***Description**

ERRA is a real array, with one element for each axis in the system, and is used for defining the Position Error criterion for Acceleration/Deceleration states.

Syntax

ERRA(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308, Default = 100. |

Tag

39

Comments

ERRA defines the maximum tolerable position error (**FAULT**(*axis_index*).#**PE**) when the motor is moving with acceleration.

As a configuration variable, the **ERRA** value is normally defined by **SPIIPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



ERRA values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

FAULT(*axis_index*).#**PE**, **FDEF**

ERRI, **ERRV**, **DELI**, **DELV**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable,

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.8 *ERRI*

Description

ERRI is a real array, with one element for each axis in the system, and is used for defining the maximum tolerable Position Error (**FAULT**(axis_index).#PE) when the motor is idle.

Syntax

ERRI(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308, Default = 100. |

Tag

40

Comments

As a configuration variable, the **ERRI** value is normally defined by **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



ERRI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

FAULT(axis_index).#PE

ERRA, **ERRV**, **DELI**, **DELV**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.9 *ERRV*

Description

ERRV is a real array, with one element for each axis in the system, and is used for defining the maximum tolerable Position Error (**FAULT**(axis_index).#PE) when the axis is moving with constant velocity.

Syntax**ERRV**(*axis_index*) = *value***Arguments**

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 2.22507e-308 to 1.79769e+308, Default = 100. |

Tag

41

Comments

As a configuration variable, the **ERRV** value is normally defined by **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** during the setup procedure of the system.

Accessibility

Read-Write



ERRV values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables[FAULT](#)(*axis_index*).#PE[ERRA](#), [ERRI](#), [DELI](#), [DELV](#)**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.10 SLLIMIT**Description**

SLLIMIT is a real array, with one element for each axis in the system, and is used for defining the minimum allowed Left position for the motor.

Syntax**SLLIMIT**(*axis_index*) = *value***Arguments**

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from -1.79769e+308 to 1.79769e+308, Default = 2e+014. |

Tag

124

Comments

If reference position [RPOS](#) is less than this value, a software Left Limit fault results and bit [FAULT](#) ([\(axis_index\).#SLL](#)) is = 1.

Accessibility

Read-Write



SLLIMIT values cannot be modified if protection is applied to this variable through **SPIiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[RPOS](#), [FAULT](#)([\(axis_index\).#SLL](#)), [SRLIMIT](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.11 SLLROUT

HW Limits Routing is available using ACSPL+ variable: **SLLROUT**

Description

SLLROUT is an integer array, with one element for each axis in the system, and is used for setting the HW limits routing for the specified axis.

Syntax

SLLROUT(<axis>)=value

Arguments

| Value | HW Limits |
|-------|----------------------|
| 0 | According to SLPROUT |
| 001 | From channel 0 |
| 101 | From channel 1 |
| 201 | From channel 2 |
| 301 | From channel 3 |

Comments

If **SLLROUT(<axis>)=0**, the routing is being done according to SLPROUT. In particular, if SLPROUT (<axis>)=0, the HW limits are being taken from the axis itself.

Tag

318

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.5.12 SRLIMIT

Description

SRLIMIT is a real array, with one element for each axis in the system, and is used for defining the minimum allowed Right position for the motor.

Syntax

SRLIMIT(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from -1.79769e+308 to 1.79769e+308, Default = 2e+014. |

Tag

128

Comments

If reference position **RPOS** is greater than this value, a software Right Limit fault results and **FAULT** (axis_index).#SRL is = 1.

Accessibility

Read-Write



SRLIMIT values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection.

Related ACSPL+ Variables

RPOS, **FAULT**(axis_index).#SRL, **SLLIMIT**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.13 XACC

Description

XACC is a real array, with one element for each axis in the system, and is used for defining the maximum allowed acceleration for the motor.

Syntax

XACC(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 2.22507e-308 to 1.79769e+308, Default = 1e+007. |

Tag

141

Comments

If the reference acceleration **RACC** exceeds this value, the Acceleration Limit fault is activated and bit **#AL** is set in variable **FAULT**.

Accessibility

Read-Write



XACC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

FAULT(*axis_index*).**#AL**, **RACC**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.14 XCURCDB

Description

XCURCDB is a real array, the size of which is determined by the total number of axes in the system. It is used for defining the threshold of the current vector peak.

Syntax

```
XCURCDB(index) = value
```

Arguments

| | |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| index | A number between 0 up to a maximum number of axes in the system minus 1. |
| value | value range is [0,100] (in percentage) Default value: 0. If the value is 0, the actual value will be minimum of ACSPL+ XRMSM and XRMSD . |

Comments

The parameter is relevant only if the Controlled Current Dynamic Brake Mode is active.

This variable is supported in version 3.10 and higher.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.15 XCURI

Description

XCURI is a real array, with one element for each axis in the system, and is used for limiting the drive output when the motor is enabled but in standstill position. **XCURI** is defined as a percentage of the maximum peak output. For products that incorporate the Drive Power Electronics (UDMnt, etc..), this value directly limits the Output Current. For products that output Voltage Signals to external Amplifiers (universal analog drive controllers such as the UDI), this value limits the Output Signal to percentage of $\pm 10V$.

Syntax

```
XCURI(axis_index) = value
```

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 100, Default = 50. |

Tag

142

Comments

XCURI is defined as a percentage of the maximum peak output. For drive command products, the command scales the voltage output range. For example, in the UDMnt-10/20, the maximum output

voltage is ±20V. Setting **XCURI** to 50 will limit the drive output to ±10V when the motor is idle. For other products setting **XCURI** will scale the peak current output.

Accessibility

Read-Write



XCURI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT\(axis_index\).#CL](#), [XRMS](#), [XCURV](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.16 XCURK

Description

XCURK is a double array with one element for each axis in the system. It sets the current limit (in percentage) for the axis to be applied during a KILL MOTION operation.

Syntax

```
XCURK (Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 0 to 100. Default value is 10. |

Tag

375

Comments

The value transferred to the DSP is:

$$\frac{XCURV * XCURK}{10000}$$

This variable is supported in version 3.00 and higher.

Related ACSPL+ Commands and Variables

[XCURV](#)

Accessibility

Read-Write

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadDouble(), acsc_WriteDouble()

3.5.17 XCURV**Description**

XCURV is a real array, with one element for each axis in the system, and is used for limiting the drive output when the motor is moving. **XCURV** is defined as a percentage of the maximum peak output. For products that incorporate the Drive Power Electronics (UDMnt, etc...), this value directly limits the Output Current. For products that output Voltage Signals to external Amplifiers (universal analog drive controllers such as the UDI), this value limits the Output Signal to percentage of $\pm 10V$.

Syntax**XCURV**(*axis_index*) = *value***Arguments**

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 100, Default = 50. |

Tag

143

Comments

XCURV is defined as a percentage of the maximum peak output. For drive command products, the command scales the voltage output range. For example, in the UDMnt-10/20, the maximum output voltage is $\pm 20V$. Setting **XCURV** to 50 will limit the drive output to $\pm 10V$ when the motor is idle. For other products setting **XCURV** will scale the peak current output.

Accessibility

Read-Write



XCURV values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection.

Related ACSPL+ Variables[FAULT](#)(*axis_index*).#CL, [XRMS](#), [XCURI](#)**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.18 XRMS

Description

XRMS is a real array, with one element for each axis in the system, and is used for setting the maximum allowable rms current for the motor. **XRMS** is defined as a percentage of the maximum peak output. For products that incorporate the Drive Power Electronics (UDMnt, etc...), this value directly limits the Output Current. For products that output Voltage Signals to external Amplifiers (universal analog drive controllers such as the UDI), this value limits the Output Signal to percentage of $\pm 10V$.

Syntax

XRMS(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 100, Default = 50. |

Tag

144

Comments

The SP program calculates **RMS** of the corresponding motor. If the calculated value exceeds the **XRMS** value, an overcurrent fault occurs. **XRMS** is defined as a percentage of the maximum output voltage.

Accessibility

Read-Write



XRMS values cannot be modified if protection is applied to this variable through **SPiiPlus** MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Variables

[FAULT](#)(*axis_index*).#CL, [XCURI](#), [XCURV](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.19 XRMSD

Description

XRMSD is a real array, with one element for each axis in the system, and is used for setting the maximum allowable RMS current on the drive, as opposed to **XRMSM** which sets the maximum allowed RMS current for the motor. **XRMSD** is defined as a percentage of the maximum peak output.

For products that incorporate the Drive Power Electronics (UDMnt, etc...), this value directly limits the Output Current. For products that output Voltage Signals to external Amplifiers (universal analog drive controllers such as the UDI), this value limits the Output Signal to percentage of $\pm 10V$.

Syntax

```
XRMSD(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axes_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 0 to 100, Default = 50. |

Tag

345

Comments

XRMSD is used to define the desired maximum current for the drive.

The SP program calculates RMS of the corresponding controller drive output. If the calculated value exceeds the **XRMSD** value, an overcurrent fault occurs. **XRMSD** is defined as a percentage of the maximum output current.

Related ACSPL+ Variables

XRMSD, **FAULT(Axis_Index).#CL**, **XCURI**, **XCURV**

Accessibility

Read-Write



XRMSD values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable(), GetFault

C Library Function

acsc_ReadReal(), acsc_WriteReal(), acsc_GetFault

3.5.20 XRMSM

Description

XRMSM is a real array, with one element for each axis in the system, and is used for setting the maximum allowable RMS current on the motor, as opposed to **XRMSD** which sets the maximum allowed RMS current for the drive. **XRMSM** is defined as a percentage of the maximum peak output. For products that incorporate the Drive Power Electronics (UDMnt, etc...), this value directly limits the Output Current. For products that output Voltage Signals to external Amplifiers (universal analog drive controllers such as the UDI), this value limits the Output Signal to percentage of $\pm 10V$.

Syntax

```
XRMSM(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 0 to 100, Default = 100. |

Tag

351

Comments

The SP program calculates RMS of the corresponding controller drive output. If the calculated value exceeds the **XRMSM** value, an overcurrent fault occurs. **XRMSM** is defined as a percentage of the maximum output current.

XRMSM is used to define the desired maximum current for the motor.

Related ACSPL+ Variables

XRMSM, **FAULT**(Axis_Index).#**CL**, **XCURI**, **XCURV**

Accessibility

Read-Write



XRMSM values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio→Toolbox→Application Development→Protection.

.NET Library Method

ReadVariable(), WriteVariable(), GetFault

C Library Function

acsc_ReadReal(), acsc_WriteReal(), acsc_GetFault

3.5.21 XRMST

Description

XRMST is a real array, with one element for each axis in the system, and is used for setting the time constant in milliseconds for the **XRMS** to activate the overcurrent protection. For calculation of **XRMS** activation time, see *SPiiPlus Setup Guide*.

Syntax

```
XRMST(axis_index) = value
```

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 200 to 3260, Default = 3230. |

Tag

145

Accessibility

Read-Write



XRMST values cannot be modified if protection is applied to this variable through **SPIiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT\(axis_index\).#CL](#), [XCURI](#), [XCURV](#), [XRMS](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFault

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetFault

3.5.22 XRMSTD

Description

XRMSTD is a real array, with one element for each axis in the system, and is used for setting the time constant in milliseconds for **XRMSD** to activate the overcurrent protection for the drive. For calculation of **XRMSD** activation time, see *SPIiPlus Setup Guide*.

Syntax

```
XRMSTD (Axis_Index) = Value
```

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 200 to 60,000, Default = 3230. |

Tag

346

Related ACSPL+ Variables

[XRMSD](#), [FAULT\(Axis_Index\).#CL](#), [XCURI](#), [XCURV](#)

Accessibility

Read-Write



XRMSTM values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio >Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable(), GetFault

C Library Function

acsc_ReadReal(), acsc_WriteReal(), acsc_GetFault



If Drive XRMS protection is triggered, the error 5049 "Drive Overcurrent" is given.

3.5.23 XRMSTM**Description**

XRMSTM is a real array, with one element for each axis in the system, and is used for setting the time constant in milliseconds for **XRMSM** to activate the overcurrent protection for the motor. For calculation of **XRMSM** activation time, see *SPiiPlus Setup Guide*.

Syntax

```
XRMSTM(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 200 to 600,000, Default = 20,000. |

Tag

352

Related ACSPL+ Variables

XRMSM, FAULT(Axis_Index).#CL, XCURI, XCURV

Accessibility

Read-Write




XRMSTM values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable(), GetFault

C Library Function

acsc_ReadReal(), acsc_WriteReal(), acsc_GetFault



If Motor XRMS protection is triggered, the error 5048 "Motor Overcurrent" is given.

3.5.24 XSACC

Description

XSACC is a real array, with one element for each axis in the system, and is used for defining the maximum allowed slave acceleration in **MASTER - SLAVE** motion.

Syntax

XSACC(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 2.22507e-308 to 1.79769e+308, Default = 1e+007. |

Tag

146

Comments

When a slave is synchronized to a master, the controller verifies the slave acceleration against the **XSACC** value each MPU cycle. If the slave acceleration exceeds **XSACC**, the motion falls out of synchronization. The controller tries to regain synchronism by having the slave pursue the master with the maximum allowed motion parameters.

Accessibility

Read-Write



XSACC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

Related ACSPL+ Commands

MASTER, SLAVE

COM Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.5.25 XVEL

Description

XVEL is a real array, with one element for each axis in the system, and is used for defining the maximum allowed velocity for the axis.

Syntax

XVEL(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 2.22507e-308 to 1.79769e+308, Default = 2e+006. |

Tag

147

Comments

If **RVEL** reference velocity exceeds **XVEL**, **FAULT**(axis_index).#VL = 1.



Trying to adjust the position and velocity loops when **XVEL** is not correctly set will produce poor results. Verify that **XVEL** is correctly defined to fit the application and other requirements before adjusting the loops.

Accessibility

Read-Write



XVEL values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

FAULT(axis_index).#VL, **RVEL**, **FVEL**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.6 Data Collection Variables

The Data Collection variables are:

| Name | Description |
|-----------------------|-------------------------------------------|
| DCN | Axis Data Collection, Number of Samples |
| DCP | Axis Data Collection, Period |
| S_DCN | System Data Collection, Number of Samples |
| S_DCP | System Data Collection, Period |
| S_ST | State of System Data Collection |

3.6.1 DCN

Description

DCN is an integer array, with one element for each axis in the system, the elements of which store the number of data collection samples per given axis.

Tag

19

Comments

DCN stores a defined number of axis data collection samples, as follows:

1. While an axis data collection is in progress **DCN** displays the index of the array element that stores the next sample.
2. When an axis data collection terminates for the corresponding axis, **DCN** stores the number of actually collected samples. If the data collection terminates automatically, the variable is always equal to the requested number of samples specified in **DC**. If **STOPDC** terminates data collection, **DCN** may contain less than the specified number of samples.

If an axis data collection is in progress, **DCN** increments each time the next sample is stored. When the data collection terminates, the **DCN** holds the last value, until the next data collection starts for the same axis.

Accessibility

Read-Only

Related ACSPL+ Commands

[DC](#), [STOPDC](#)

Related ACSPL+ Variables

[AST](#), [DCP](#)

COM Library Methods and .NET Library Methods

ReadVariable, DataCollection, StopCollect, WaitCollectEnd

C Library Functions

acsc_ReadInteger, acsc_DataCollectionExt, acsc_StopCollect, acsc_WaitCollectEnd

3.6.2 DCP

Description

DCP is a real array, with one element for each axis in the system, the elements of which store the axis data collection samples based on a specified sampling *period*. When an axis data collection terminates, **DCP** stores the sampling period.

Tag

21

Comments

DCP is generally equal to the specified *period*, however because *period* is rounded to an integer number of controller cycles, the actual *period* may differ from the *period* specified in the **DC** command.

If **DC/t** (temporal data collection) was executed, **DCP** may be greater than the requested minimal period.

When a system data collection starts, **DCP** is assigned a real data collection period.

Accessibility

Read-Only

Related ACSPL+ Commands

[DC](#), [STOPDC](#)

Related ACSPL+ Variables

[AST](#), [DCN](#)

COM Library Methods and .NET Library Methods

[ReadVariable](#), [DataCollection](#), [StopCollect](#), [WaitCollectEnd](#)

C Library Functions

[acsc_ReadReal](#), [acsc_DataCollectionExt](#), [acsc_StopCollect](#), [acsc_WaitCollectEnd](#)

3.6.3 S_DCN

Description

S_DCN is a scalar integer that stores a defined number of system data collection samples.

Tag

111

Comments

S_DCN stores a defined number of system data collection samples, as follows:

1. While a system data collection is in progress **S_DCN** displays the index of the array element that stores the *next* sample.
2. When a system data collection terminates, the variable stores the number of actually collected samples. If the data collection terminates automatically, the variable is always equal to the requested number of samples specified in the **dc** command. If the data

collection terminates due to the [STOPDC](#) command, the variable may be less than the requested number of samples.

For *cyclic* data collection **S_DCN** displays the current number of collected samples and changes as follows:

1. At the start of data collection, **S_DCN** is assigned with zero.
2. With each sampling, **S_DCN** is incremented until it reaches the specified size of the sample array
3. **S_DCN** remains unchanged - the newest sample overwrites the oldest, so the total number of samples remains the same.

As long as cyclic data collection is in progress, the application cannot use the sample array. After the cyclic data collection finishes, the controller repacks the sample array so that the first element represents the oldest sample and the last element represents the most recent sample.

Accessibility

Read-Only

Related ACSPL+ Commands

[DC](#), [STOPDC](#)

Related ACSPL+ Variables

[S_ST](#), [S_DCP](#)

COM Library Methods and .NET Library Methods

ReadVariable, DataCollection, StopCollect, WaitCollectEnd

C Library Functions

acsc_ReadInteger, acsc_DataCollectionExt, acsc_StopCollect, acsc_WaitCollectEnd

3.6.4 S_DCP

Description

S_DCP is real variable that stores the period of system data collection sampling.

Tag

112

Comments

When a system data collection terminates, the **S_DCP** stores the sampling *period*. Unless a temporal data collection was executed, the variable is always equal to the requested period specified in the [DC](#) command.



S_DCP is generally equal to the specified period, however because the period is rounded to an integer number of controller cycles, the actual period may differ from the period specified in the **DC** command.

Accessibility

Read-Only

Related ACSPL+ Commands

[DC](#), [STOPDC](#)

Related ACSPL+ Variables

[S_ST](#), [S_DCN](#)

COM Library Methods and .NET Library Methods

ReadVariable, DataCollection, StopCollect, WaitCollectEnd

C Library Functions

acsc_ReadReal, acsc_DataCollectionExt, acsc_StopCollect, acsc_WaitCollectEnd

3.6.5 S_ST

Description

S_ST is a scalar integer variable that provides the state of System Data Collection.

Tag

120

Comments

S_ST provides a bit that indicates if system data collection is currently in progress.

Bit 3:

0 - System data collection off

1 - System data in progress.

Accessibility

Read-Only

Related ACSPL+ Commands

[DC](#), [STOPDC](#)

Related ACSPL+ Variables

[S_DCN](#), [S_DCP](#)

COM Library Methods and .NET Library Methods

ReadVariable, DataCollection, StopCollect, WaitCollectEnd

C Library Functions

acsc_ReadInteger, acsc_DataCollectionExt, acsc_StopCollect

3.7 Input and Output Variables

The Input and Output variables are:

| Name | Description |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| AIN | Analog Inputs |
| AINOFFS | Percent offset of analog signal from external source |
| AINSCALE | Define scale of analog input signal |
| AOUT | Analog Outputs |
| COMMCH | Returns the last activated communication channel. |
| DCOM | Drive command - in open loop |
| DOUT | Drive output. |
| EXTIN | Extended digital inputs (HSSI) |
| EXTOUT | Extended digital outputs (HSSI) |
| IN | General Purpose Digital Inputs |
| OUT | General Purpose Digital Outputs |
| SPIRXN | Number of words transmitted from SPI interface |
| SPIST | Integer array with one element for each EtherCAT node in the system. It shows the current state of the SPI communication channel. |

3.7.1 AIN

Description

AIN is a real array, the size of which is determined by the total number of analog input signals in the system, and is used for defining the level of an analog signal from an external source such as a sensor or a potentiometer.

Syntax

```
value = AIN(index)
```

Arguments

| | |
|--------------|---------------------------------------------------------------------------------------------------|
| <i>index</i> | A number between 0 up to the maximum number of analog input signals minus one. |
| <i>value</i> | value is the scaling, by percent, of the signal and ranges from -100 to +100, Default = 0. |

Tag

4

Comments

None

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable, GetAnalogInput

C Library Functions

acsc_ReadReal, acsc_GetAnalogInput

3.7.2 AINOFFS

Description

AINOFFS is a real array, the size of which is determined by the total number of analog input signals in the system and is used for defining the percent offset of an analog signal from an external source such as a sensor or a potentiometer.

Syntax

```
AINOFFS(index) = value
```

| | |
|-------|---------------------------------------------------------------------------------|
| index | A number between 0 up to the maximum number of analog inputs signals minus one. |
| value | Value is the offset, by percent, ranges from -100 to +100, Default = 0. |

Tag

378

Comments

- > If used in combination with the analog input scaling feature(**AINSCALE**) then, the order of operations will be first offset and then scaling.
- > The **AIN** variable range is now between -200 to +200.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

AIN, AINSCALE

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.7.3 AINSCALE

Description

AINSCALE is a real array, the size of which is determined by the total number of analog input signals in the system and is used for defining the scaling of an analog signal from an external source such as a sensor or a potentiometer.

Syntax

```
AINSCALE(index) = value
```

Arguments

| | |
|-------|---------------------------------------------------------------------------------|
| index | A number between 0 up to the maximum number of analog inputs signals minus one. |
| value | Value is the scaling factor, ranges from -2 to +2, Default = 1. |

Tag

391

Comments

If combined with the analog input offset feature(**AINOFFS**) then the order of operations will be offset and then scaling.

The **AIN** variable range is between -200 to +200.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

AIN, AINOFFS

Accessibility

Read-Write

3.7.4 AOUT

Description

AOUT is a real array, the size of which is determined by the total number of analog output signals in the system, and is used for defining the level of a general purpose analog signal that is sent to an external device.

Syntax

```
AOUT(index) = value
```


Arguments

| | |
|--------------|---------------------------------------------------------------------------------------------------|
| <i>index</i> | A number between 0 up to the maximum number of analog output signals minus one. |
| <i>value</i> | value is the scaling, by percent, of the signal and ranges from -100 to +100, Default = 0. |

Tag

5

Comments

1. Some aspects of **AOUT** are model-dependent, including the number of analog outputs and type of analog inputs (differential or single-ended).
2. In SPiiPlus controllers (not CM) **AOUT** can be used only when the axis is defined as Dummy - see [MFLAGS](#).
To define the analog output command to a drive (connected to a motor) in open loop, refer to [DCOM](#).

Accessibility

Read-Write

Related ACSPL+ Variables[MFLAGS](#)**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable, GetAnalogOutput, SetAnalogOutput

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetAnalogOutput, acsc_SetAnalogOutput

3.7.5 DOUT**Description****DOUT** is an integer array that stores the drive command (velocity loop output) for each axis.**Tag**

26

Comments**DOUT** values range from -32767 to 32767, which indicates +/-100% command.In open loop mode (**MFLAGS.1=1**), **DOUT** is determined by the **DCOM** variable.In closed loop mode (**MFLAGS.1=0**), **DOUT** is determined by **DCOM** plus the velocity loop output.In gantry mode (**MFLAGS.25=1**) **DOUT** of the primary axis indicates the longitudinal (force) command and **DOUT** of the secondary axis indicates the rotational (torque) command.**DOUT** is updated at the MPU rate (**CTIME**) in most products, see note below.

The following ACS products only support lower rate update (once per 100msec) of DOUT:



- > UDMpc
- > CMnt
- > UDMpm
- > MC4U with SPiiPlus NT-HP/LT/LD
- > MC4U with SPiiPlus DC-HP/LT/LD
- > SPiiPus SAnt

Accessibility

Read-Only, ReadVariable

COM Library Methods and .NET Library Methods

ReadVariable, C Library Functions, acsc_ReadInteger

3.7.6 EXTIN

Description

EXTIN is an integer array, the size of which is determined by the total number of SPI and HSSI input signals in the system, and reads the current state of the inputs. The number of inputs depends on the number of SPI and HSSI modules in the system.

For details about the HSSI, see the *HSSI Modules Hardware Guide*.

Comments

The **SPIRXN** variable is updated every cycle with the number of active elements.

Tag

42

Accessibility

Read-Only

Related ACSPL+ Variables

EXTOUT, IN

COM Library Methods and .NET Library Methods

ReadVariable, GetExtInput, GetExtInputPort

C Library Functions

acsc_ReadInteger, acsc_GetExtInput, acsc_GetExtInputPort

3.7.7 EXTOUT

Description

EXTOUT is an integer array, the size of which is determined by the total number of SPI and HSSI output signals in the system, which can be used for reading or setting the current state of the outputs. The number of outputs depends on the number of HSSI modules and SPI inputs in the system.

For details about the HSSI, see the *HSSI Modules Hardware Guide*.

Syntax

EXTOUT(*index*) = *value*

Arguments

| | |
|--------------|----------------------------------------------------------------|
| <i>index</i> | A number between 0 and 511. |
| <i>value</i> | value ranges from -2147483648, 2147483647, Default = 0. |

Comments

When used with an SPI interface in master mode, the **EXOUT** function should be used.

The **SPICFG** function sets the number of elements which contain data.

Tag

43

Accessibility

Read-Write

Related ACSPL+ Variables

EXTIN, OUT

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetExtOutput, SetExtOutput, GetExtOutputPort, SetExtOutputPort

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_GetExtOutput, acsc_SetExtOutput, acsc_GetExtOutputPort, acsc_SetExtOutputPort

3.7.8 IN

Description

IN is an integer array, the size of which is determined by the total number of digital input signals in the system, and stores the current state of the General Purpose digital inputs.

Syntax

IN(*port*).*bit*

Arguments

| | |
|-------------|---------------------------------------------------------------------------|
| <i>port</i> | A number between 0 and the total number of ports in the system minus one. |
| <i>bit</i> | bit can be 0-31. |

Tag

71

Comments

General Purpose inputs are represented by bits 0..31 of **IN**(*port*). Each bit reports the state of one General Purpose input.

For example, entering the query command **?IN(0).0** in the SPiiPlus MMI Application Studio **Communication Terminal** will return "0" if inputs #0 is non-active or "1" when active.



In some SPiiPlus controllers, the digital input pins can also be used as **MARK**.

Accessibility

Read-Only

Related ACSPL+ Variables

[OUT](#), [EXTOUT](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetInput, GetInputPort

C Library Functions

acsc_ReadInteger, acsc_GetInput, acsc_GetInputPort

3.7.9 OUT

Description

OUT is an integer array, the size of which is determined by the total number of digital output signals in the system, and can be used for reading or writing the current state of the General Purpose digital outputs.

Syntax

OUT(port).bit

Arguments

| | |
|-------------|-----------------------------------------------------------------------|
| <i>port</i> | A number between 0 the total number of ports in the system minus one. |
| <i>bit</i> | <i>bit</i> can be 0-31. |

Tag

94

Comments

General purpose outputs are represented by bits 0..31 of **OUT(port)**. Each bit reports the state of one general purpose output for the given port.

For example, the query command **?OUT(23).0 = 1** through the SPiiPlus MMI Application Studio **Communication Terminal** will activate the outputs #0 of port 23.



In some SPiiPlus controllers the digital output pins can also be used as **PEG** - see .

Accessibility

Read-Write

Related ACSPL+ Variables

IN, EXTIN

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, SetOutput, GetOutput, SetOutputPort, GetOutputPort

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_SetOutput, acsc_GetOutput, acsc_SetOutputPort, acsc_GetOutputPort

3.7.10 SPIRXN**Description**

SPIRXN is a variable that shows the number of actual words that contain data transmitted from the SPI external interface. The range is from 0 to 8.

Accessibility

Read-only

Comments

SPIRXN value is 0 if the SPI is disabled. This variable is supported by the UDMsm, IDMSm and ECMsm products only.

3.7.11 SPIST**Description**

SPIST is an integer array with one element for each EtherCAT node in the system. It shows the current state of the SPI communication channel.

Bit Field

The contents of each entry are interpreted according to the following table

| Bit | Name | Description |
|------|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| 0 | SPI RX Overflow | 1 - more than 16 SPI words received 0 - No Event |
| 1 | SPI TX Full | 1 - Attempt to add another SPI word for transmission while the SPI TX FIFO is full. The FIFO depth is 16. 0 - No Event |
| 2-15 | | Reserved |

Comments

If a specific element has the value 0, then there are no errors in that node

TAG

385

Accessibility

Read-only

3.8 Monitoring Variables

The Monitoring variables are:

| Name | Description |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BCODEUSG | An integer array, with one element for each buffer in the system. It shows the actual memory allocated by a program code in KB, for each buffer. |
| BCODECFG | An integer array, with one element for each buffer in the system. It is used for configuration of the amount of memory in KB that is pre-allocated for program code for each buffer. |
| BGLOBCFG | Sets the amount of memory pre-allocated for global variables in the D-Buffer. |
| BGLOBUSG | A scalar that shows the amount of memory used by the global variables defined in the D-Buffer. |
| BSRCCFG | An integer array with one element for each buffer in the system. It is used for configuration of the amount of memory in KB that is pre-allocated for a program source for each buffer. |
| BSRCUSG | An integer array, with one element for each buffer in the system. It shows, in KB, the actual memory allocated by a program source for each buffer. |
| BVARUSG | An integer array with one element for each buffer in the system. It shows, in KB, the actual memory allocated by a program local variables for each buffer. |
| BVARCFG | An integer array with one element for each buffer in the system. It is used for configuration of the amount of memory that is pre-allocated for local variables in KB, for each buffer. |
| JITTER | Elapsed time between the physical timer interrupt and the SC real-time task starts working. |
| MSSYNC | Time difference between the master clock and the bus clock. |
| USGBUF | Stores the amount of MPU usage as a percentage of the specific ACSPL+ buffer in the controller cycle during the execution of real-time tasks. |
| USGTRACE | Stores the amount of MPU usage as a percentage of the specific real-time task in the controller cycle during the execution of real-time tasks. |

| Name | Description |
|-------------------------|-----------------------------------------------------------------------------|
| SOFTIME | Parameter which specifies the EtherCAT frame delivery time in microseconds. |
| TIME | Elapsed Time |
| USAGE | MPU Usage |

3.8.1 *BCODECFG*

Description

BCODECFG is an integer array, with one element for each buffer in the system. It is used for configuration of the amount of memory in KB that is pre-allocated for program code for each buffer.

Syntax

```
BCODECFG(Buffer_Index) = value
```

Arguments

| | |
|--------------|---------------|
| Buffer_Index | Buffer index. |
|--------------|---------------|

Tag

409

Comments

The values are in KB.

The default value is 128KB.

BCODEUSG can be useful for setting the value of the **BCODECFG** variable.

A new value takes affect only after controller reboot.

Related ACSPL+ Variables

[BCODEUSG](#)

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.8.2 *BCODEUSG*

Description

BCODEUSG is an integer array, with one element for each buffer in the system. It shows the actual memory allocated by a program code in KB, for each buffer.

Syntax

```
[command] BCODEUSG (Buffer_Index)
```

Arguments

Buffer_Index

Buffer index.

Tag

410

Comments

- > The values are in KB.
- > For empty buffer, the value is 0.
- > **BCODEUSG** can be useful for setting the value of the **BCODECFG** variable.

Related ACSPL+ Variables

BCODECFG

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.8.3 BGLOBCFG

Description

BGLOBCFG is a scalar. It sets the amount of memory pre-allocated for global variables in the D-Buffer.

Syntax

```
BGLOBCFG = value
```

Arguments

NONE

Tag

413

Comments

- > The value is in KB.
- > The default value is 1024KB.
- > **BGLOBUSG** can be useful for setting the value of the **BGLOBCFG** variable.

- > Setting the **BGLOBCFG** variable changes the value of **BGLOBCFG** only if the **S_SETUP.#VRMEMVAR** bit is ON
- > The **BGLOBCFG** variable applies only for global variables defined in the D-Buffer.
- > A new value takes effect only after controller reboot.

Related ACSPL+ Variables

BGLOBUSG

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, Write Variable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.8.4 *BGLOBUSG*

Description

BGLOBUSG is a scalar that shows the amount of memory used by the global variables defined in the D-Buffer.

Syntax

```
value = BGLOBUSG
```

Arguments

NONE

Tag

414

Comments

- > The value is in KB.
- > **BGLOBUSG** can be useful for setting the value of the **BGLOBCFG** variable.
- > The **BGLOBUSG** variable has a valid value only if **S_SETUP.#VRMEMVAR** bit is ON
- > The **BGLOBUSG** variable applies only for global variables defined in the D-Buffer

Related ACSPL+ Variables

BGLOBCFG

Accessibility

Read-only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.8.5 *BSRCUSG*

Description

BSRCUSG is an integer array, with one element for each buffer in the system. It shows, in KB, the actual memory allocated by a program source for each buffer.

Syntax

```
value = BSRCUSG(Buffer_Index)
```

Arguments

| | |
|--------------|---------------|
| Buffer_Index | Buffer index. |
|--------------|---------------|

Tag

412

Comments

- > The values are in KB
- > For empty buffers the value is 0
- > **BSRCUSG** can be useful for setting the value of the **BSRCCFG** variable
- > The value should equal the size of the source code

Related ACSPL+ Variables

BSRCCFG

Accessibility

Read-only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.8.6 *BSRCCFG*

Description

BSRCCFG is an integer array with one element for each buffer in the system. It is used for configuration of the amount of memory in KB that is pre-allocated for a program source for each buffer.

Syntax

```
BSRCCFG(Buffer_Index) = value
```

Arguments

| | |
|--------------|---------------|
| Buffer_Index | Buffer index. |
|--------------|---------------|

Tag

411

Comments

- > The values are in KB.
- > The default value is 64KB.
- > **BSRCUSG** can be useful for setting the value of the **BSRCCFG** variable.
- > A new value takes effect only after controller reboot.

Related ACSPL+ Variables

BSRCUSG

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.8.7 BVARUSG

Description

BVARUSG is an integer array with one element for each buffer in the system. It shows, in KB, the actual memory allocated by a program local variables for each buffer.

Syntax

```
value = BVARUSG(Buffer_Index)
```

Arguments

| | |
|--------------|---------------|
| Buffer_Index | Buffer index. |
|--------------|---------------|

Tag

416

Comments

- > The values are in KB.
- > For an empty buffer, the value is 0.
- > **BVARUSG** can be useful for setting the value of the **BVARCFG** variable.
- > The **BVARUSG** variable has a valid value only if **S_SETUP.#VRMEMVAR** bit is ON

Related ACSPL+ Variables

BVARCFG

Accessibility

Read-only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.8.8 BVARCFG

Description

BVARCFG is an integer array with one element for each buffer in the system. It is used for configuration of the amount of memory that is pre-allocated for local variables in KB, for each buffer.

Syntax

```
BVARCFG(Buffer_Index) = value
```

Arguments

Buffer_Index

Buffer index.

Tag

415

Comments

- > The values are in KB.
- > The default value is 10KB.
- > **BVARUSG** can be useful for setting the value of the **BVARCFG** variable.
- > The **BVARCFG** variable takes effect only if **S_SETUP.#VRMEMVAR** bit is ON.
- > A new value takes effect only after controller rebooted.

Related ACSPL+ Variables

BVARCFG

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.8.9 JITTER

Description

JITTER is a real variable that contains the time, in microseconds, that elapsed from the physical timer interrupt until the SC real-time task starts working. This parameter shows the influence of overall hosting PC load on real-time endurance of SC.

Tag

224

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.8.10 MSSYNC

Description

MSSYNC is a real variable that contains the difference, in microseconds, between clocks of the master and the bus. This parameter shows how close the synchronization is between the two clocks.

Tag

225

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.8.11 USGBUF

Description

USGBUF array stores the amount of MPU usage as a percentage of the specific ACSPL+ buffer in the controller cycle during the execution of real-time tasks.

USGBUF is a real array with one element for each ACSPL+ buffer as follows:

- > USGBUF(0) - Buffer 0
- > USGBUF(63) - Buffer 63
- > USGBUF(64) - D-Buffer
- > USGBUF(65) - Buffer for immediate commands execution (C / COM libraries, communication terminal, etc.)
- > USGBUF(66) - Buffer for MACRO execution

Tag

246

Comments

The real-time tasks always have the greatest priority. If the usage reaches 80% or more, the response time of the controller deteriorates. In addition, it is dangerous and may cause jerks in the motion profile.

The **USGBUF** variable should be used for debugging purposes only and should not be used in real production applications. In order to use a tracing mechanism, bit 1 of **S_SETUP** variable should be set to 1.

3.8.12 USGTRACE**Description**

USGTRACE array is used for storing the amount of MPU usage as a percentage of the specific real-time task in the controller cycle during the execution of real-time tasks.

USGTRACE is a real array with one element for each real-time task according to the following list:

- > USGTRACE(0) - EtherCAT communication (including communication jitter)
- > USGTRACE(1) - reading inputs, prerequisite operations for motion generator
- > USGTRACE(2) - motion generator and real-time objects
- > USGTRACE(3) - operations on axes and Servo Processor interfaces
- > USGTRACE(4) - execution of ACSPL+ buffers
- > USGTRACE(5) - writing outputs, house keeping operations



Values 6...9 are not used and reserved for future needs

Tag

245

Comments

The real-time tasks always have the greatest priority. If the usage reaches 80% or more, the response time of the controller deteriorates. In addition, it is dangerous and may cause jerks in the motion profile.

The **USGTRACE** variable should be used for debugging purposes only and should not be used in real production applications. In order to use a tracing mechanism, bit 1 of **S_SETUP** variable should be set to 1.

3.8.13 SOFTIME**Description**

SOFTIME is a read-only real array, with one element for each EtherCAT node in the system, which specifies the EtherCAT frame delivery time in microseconds.

Tag

255

Accessibility

Read Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

Comments

A Bit 2 (#SOFTIME) setting enables measuring the EtherCAT frame delivery time. Currently this feature is supported by the following products only:

- > "IOMnt (rev.B2 and higher)
- > "PDMnt (rev. B3 and higher)
- > "SDMnt (rev. B2 and higher)

3.8.14 TIME**Description**

TIME is a real variable that provides the defines the elapsed time (in milliseconds) from the controller power-up.

Tag

134

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.8.15 USAGE**Description**

USAGE is a real variable used for storing the amount of MPU usage as a percentage of the real-time tasks in the controller cycle during the execution of real-time tasks.

Tag

137

Comments

The real-time tasks always have the greatest priority. If the usage reaches 80% or more, the response time of the controller deteriorates, in addition, it is dangerous and may cause jerks in the motion profile.

The **USAGE** variable value can be used in autoroutines for halting the application should usage exceed a certain value.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.8.16 USAGESIM

Description

USAGESIM is a real variable used for storing MPU usage as a percentage of the real-time tasks in the controller cycle during the execution of real-time tasks in the simulator.

Tag

376

Comments

The **USAGESIM** variable should be used for debugging purposes only and should not be used in real production applications.

Since the simulator runs in a non-real-time OS, USAGESIM can frequently be above 100% .

This variable is supported in version 3.00 and higher.

Accessibility

Read-only

Related ACSPL+ Variables

[USAGE](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9 Motion Variables

The Motion variables are:

| Name | Description |
|-------------------------|----------------------------------------------------------------------------------------------------|
| ACC | Default Acceleration |
| APOS | Axis Position |
| APOSFLT | Integer array storing the current desired motor position, including the filtering operation result |

| Name | Description |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| CERRK | Critical Position Error criterion for the KILL state |
| DEC | Default Deceleration |
| DECOMP | Error Correction |
| DELK | Defines the time delay after a kill process when CERRK is used to indicate a critical position error. |
| DAPOS | Delayed Axis Position |
| FACC | Feedback Acceleration |
| FEEDRF | Feedrate Override |
| FPOS | Feedback Position |
| F2POS | Secondary Feedback Position |
| FVEL | Feedback Velocity |
| F2VEL | Secondary Feedback Velocity |
| GACC | Group Acceleration |
| GJERK | Group Jerk |
| GMOT | Motion Number |
| GMQU | Motion Queue |
| GMTYPE | Motion Type |
| GPATH | Group Path |
| GPHASE | Motion Phase |
| GRTIME | Remaining Motion Time |
| GSEG | Motion Segment |
| GSFREE | Free Motion Segments |
| GSNAP | A real array, with one element for each axis in the system, and is used for defining the current calculated snap vector snap of group motion. |
| GVEC | Group Vector |

| Name | Description |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GVEL | Group Velocity |
| JERK | Default jerk |
| KDEC | Default kill deceleration |
| LPOS | Axis position in LCS |
| NVEL | Sets a non-zero axis velocity in stepper motor applications |
| SETTLEA | Sets the time to wait inside the target radius before triggering |
| SETTLEB | Sets the time to wait inside the target radius before triggering |
| SETTLEC | Sets the time to wait inside the target radius before triggering |
| SLSFF | A SNAP feed forward parameter. |
| SNAP | A real array with one element for each axis in the system, and is used for defining the snap (jerk derivative) of the motion profile in user-defined units. |
| STLTIMEA | Last settling time of axis, according to SETTLEA and TARGRADA criteria |
| STLTIMEB | Last settling time of axis, according to SETTLEB and TARGRADB criteria |
| STLTIMEC | Last settling time of axis, according to SETTLEC and TARGRADC criteria |
| SETTLEC | Sets the time to wait inside the target radius before triggering |
| TARGRADA | Sets the target radius around which you wish the motion to settle. |
| TARGRADB | Sets the target radius around which you wish the motion to settle. |
| TARGRADC | Sets the target radius around which you wish the motion to settle. |
| TPOS | Target position for track motion |
| PE | Position Error |
| RACC | Reference Acceleration |
| PPOS | Reference Position, 20kHz |
| PRFLTIME | Holds the time(In milliseconds) passed from the moment that the move starts until the motion profile ends |

| Name | Description |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PPOSCOMP | Reference Jerk |
| RJERK | Reference Position |
| RPOSCOMP | Reference Position |
| RPOSDEL | Actual motor motion delay |
| RSNAP | A real array, with one element for each axis in the system, and is used for defining the current calculated reference snap (jerk derivation) in user-defined units. |
| RVEL | Reference Velocity |
| VEL | Velocity |

3.9.1 ACC

Description

ACC is a real array, with one element for each axis in the system, and is used for defining the motion profile acceleration.

Syntax

ACC(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308, Default = 100000. |

Tag

1

Comments

For single-axis motion, **ACC** defines the axis acceleration. If the axis is a leading axis in a group, **ACC** defines the vector acceleration of the common motion.

If **ACC** is changed when a motion is in progress, the change does not affect currently executing motions, or motions that were created before the change.

Accessibility

Read-Write



ACC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[IMM](#), [SLAVE](#), and all motion commands where the profile is generated by the controller.

Related ACSPL+ Variables

[DEC](#), [JERK](#), [KDEC](#), [VEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, SetAcceleration. GetAcceleration, SetAccelerationImm

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_SetAcceleration. acsc_GetAcceleration, acsc_SetAccelerationImm

3.9.2 APOS

Description

APOS is a real array, with one element for each axis in the system, and is used for defining the current reference value for the axis in user-defined units.

Syntax

See [SET](#)

Tag

6

Comments

APOS is updated each MPU cycle if a motion that involves the axis is in progress.

If the corresponding motor has a default connection ([MFLAGS](#)(axis_index).#**DEFCON** =1), **APOS** = [RPOS](#).

Accessibility

Read-Only - Can be changed using [SET](#).

Related ACSPL+ Commands

[MASTER](#), [SLAVE](#)

Related ACSPL+ Variables

[MPOS](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.3 APOSFILT

Description

APOSFILT is real array with one element for each axis in the system. The array elements store the current desired motor position, including the filtering operation result, such as input shaping.



APOSFILT updates on every controller cycle according to the filtering algorithm. When the filtering algorithm is not configured, **APOSFILT = APOS**.

TAG

368

Comments

This variable is supported in version 3.00 and higher.

ACCESSIBILITY

Read-Only

RELATED ACSPL+ COMMANDS

All motion commands

RELATED ACSPL+ VARIABLES

FPOS, RPOS, APOS,

.NET LIBRARY METHODS

ReadVariable(), WriteVariable()

C LIBRARY FUNCTIONS

acsc_ReadReal(), acsc_WriteReal()

3.9.4 CERRK

Description

CERRK is a real array, with one element for each axis in the system, and is used for defining the Critical Position Error criterion for the **KILL** state.

Syntax

```
CERRK(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308 to 1.79769e+308. Default: 1000 |

Tag

347

Comments

CERRK defines the maximum tolerable critical position error (**FAULT(axis_index).#PE**) during the kill operation.

The value of **CERRK** is always equal or higher than that of **CERRA**.

If **CERRK** is assigned a new value which is lower than **CERRA**, **CERRK** is set to **CERRA** automatically.

If **CERRA** is assigned a new value which is higher than **CERRK**, **CERRK** is set to **CERRA** automatically.

Related ACSPL+ Variables

FAULT(Axis_Index).#PE, FDEF

Accessibility

Read-Write



CERRK values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.5 DAPOS**Description**

DAPOS is a real array, with one element for each axis in the system. **DAPOS** reads the delayed Axis Position value which is synchronized with **RPOS** and **FPOS**.

Syntax

DAPOS is activated as part of the SPiiPlus MMI Application Studio **Scope**.

Tag

18

Comments

Use **DAPOS** only to view the axis position in the **Scope** when comparing the axis position to the **RPOS**.

In the SPiiPlus, **APOS** (axis position) is not synchronized with **RPOS** and **FPOS** and are characterized by a few msec delay.

When implementing a non-default **CONNECT**, it may be necessary to monitor **APOS** versus **RPOS** with the Scope.

Accessibility

Read-Only

Related ACSPL+ Commands

[CONNECT](#)

Related ACSPL+ Variables

[APOS, RPOS](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.6 DEC

Description

DEC is a real array, with one element for each axis in the system and is used for specifying the motion profile deceleration in milliseconds.

Syntax

DEC(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308, Default = 100000. |

Tag

22

Comments

For single-axis motion, **DEC** defines axis deceleration. If the axis is a leading axis in a group, **DEC** defines the vector deceleration of the common motion.

If **DEC** is changed when a motion is in progress, the change does not affect currently executing motions or motions that were created before the change.

Accessibility

Read-Write



DEC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[HALT](#) , [IMM](#), [SLAVE](#)

Related ACSPL+ Variables

[ACC](#), [JERK](#), [KDEC](#), [VEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, SetDeceleration. GetDeceleration, SetDecelerationImm

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_SetDeceleration. acsc_GetDeceleration, acsc_SetDecelerationImm

3.9.7 DECOMP**Description**

DECOMP is a real array, with one element for each axis in the system, and is used for displaying the error correction for the mechanical error compensation that was applied to the axis. **DECOMP** displays the difference between **RPOS** and **RPOSCOMP**.

Tag

344

Accessibility

Read-Only

Related ACSPL+ Commands

All motion commands

Related ACSPL+ Variables

FPOS, RPOS, RPOSCOMP, APOS, PE .

NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.8 DELK**Description**

DELK is a real array, with one element for each axis in the system, and is used for defining the time delay after a kill process in which we still use **CERRK** to indicate a critical position error.

Syntax

```
DELK(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 0 to 1000. |

Tag

350

Comments

CERRK defines the maximum tolerable critical position error (**FAULT(axis_index).#CPE**) during the kill operation. After the kill operation in order to ensure a smooth transition, we still use **CERRK** to define the maximal tolerable position error, for **DELK** time.

Related ACSPL+ Variables

FAULT(Axis_Index).#CPE, CERRK

Accessibility

Read-Write



DELK values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.9 FACC**Description**

FACC is a real array, with one element for each axis in the system, and is used for defining the feedback acceleration value of the axis.

Tag

46

Accessibility

Read-Only

Related ACSPL+ Variables

[FVEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetAcceleration

C Library Functions

acsc_ReadReal, acsc_GetAcceleration

3.9.10 FPOS**Description**

FPOS is a real array, with one element for each axis in the system, and is used for defining the current feedback position for the motor.

Tag

52

Comments

The user can shift the origin of feedback position using [SET](#).

The user can select the units of feedback position by setting the [EFAC](#) variable.

Accessibility

Read-Only

Related ACSPL+ Commands

[SET](#)

Related ACSPL+ Variables

[APOS](#), [RPOS](#), [SLPROUT](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetFPosition, SetFPosition

C Library Functions

acsc_ReadReal, acsc_GetFPosition, acsc_SetFPosition

3.9.11 F2POS**Description**

F2POS is a real array, with one element for each axis in the system, and is used for defining the current secondary feedback value for the motor in user-defined units.

Tag

44

Comments

The user can shift the origin of secondary feedback position using [SET](#).

The user can select the units of secondary feedback position by setting the [E2FAC](#) variable.

The application needs to explicitly clear [IST\(axis_index\).#IND2](#) in order to resume the latching logic.

Accessibility

Read-Only - Can be changed by [SET](#).

Related ACSPL+ Commands

[SET](#), [SLP2ROUT](#)

Related ACSPL+ Variables

[IST](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.12 FVEL

Description

FVEL is a real array, with one element for each axis in the system, the elements of which store the measured velocity.

Tag

53

Accessibility

Read-Only

Related ACSPL+ Variables

[FVFIL](#), [RVEL](#), [XVEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetFVelocity

C Library Functions

acsc_ReadReal, acsc_GetFVelocity

3.9.13 F2VEL

Description

F2VEL is a real array, with one element for each axis in the system, the elements of which store the measured secondary velocity.

Tag

45

Accessibility

Read-Only

Related ACSPL+ Variables

[FVFIL](#), [RVEL](#), [XVEL](#), [FVEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetFVelocity

C Library Functions

acsc_ReadReal, acsc_GetFVelocity

3.9.14 FEEDRF

Description

FEEDRF is real array, with one element for each axis in the system, the elements of which store the feedrate factor. The feedrate factor modifies the calculation of motion velocity for all relevant motion profiles.

Examples

```
FEEDRF (2) = 1.23
```

```
IMM FEEDRF(0) = 0.5
```

Comments

This variable may be updated immediately using the "IMM" qualifier.

The allowed range is 0.1 to 2.0.

Acceleration and Jerk are NOT affected, which actually changes the trajectory characteristics (may result in Triangular instead of Trapezoidal Trajectory).

It takes effect on the next Trajectory calculation, according to the specified velocity of that trajectory; situation varies according to PTP Switches.

In case of group motion, the FEEDRF of the leading Axis will be in effect.

Motion Modes in which FEEDRF is supported: PTP, JOG, TRACK, MPTP, XSEG

Accessibility

Read-Write

Related ACSPL+ Commands

IMM, PTP, JOG, TRACK, MPTP, XSEG

Related ACSPL+ Variables

VEL

3.9.15 GACC

Description

GACC is a real array, with one element for each axis in the system, and is used for deriving the vector acceleration of a group motion.

For example when the three axes 0, 1 and 2 are moving as a group, the **GACC** is calculated by:

$$GACC = \sqrt{(Acceleration_0)^2 + (Acceleration_1)^2 + (Acceleration_2)^2}$$



GPATH, **GVEL**, **GACC**, **GPHASE**, **GJERK**, and **GRTIME** Variables are updated while the motion is in progress.

Tag

55

Accessibility

Read-Only

Related ACSPL+ Commands

GROUP

Related ACSPL+ Variables

GVEL, **GJERK**, **GPATH**, **GPHASE**, **GRTIME**

COM Library Methods and .NET Library Methods

ReadVariable, GetAcceleration

C Library Functions

acsc_ReadReal, acsc_GetAcceleration

3.9.16 GJERK**Description**

GJERK is a real array, with one element for each axis in the system, and is used for deriving the vector acceleration of a group motion.

For example when the three axes 0, 1 and 2 are moving as a group, the **GJERK** is calculated by:

$$GJERK = \sqrt{(Jerk_0)^2 + (Jerk_1)^2 + (Jerk_2)^2}$$



GPATH, **GVEL**, **GACC**, **GJERK**, **GPHASE**, and **GRTIME** variables are updated while the motion is in progress.

Tag

57

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.17 GMOT**Description**

GMOT is an integer array, with one element for each axis in the system, and defines the ordinal number of the current motion.

Tag

58

Comments

The **GMOT** value is valid only if one of the following is true:

- > Single-axis motion in progress
- > The axis is a leading axis in a group and motion in the group is in progress

After power-up, **GMOT** is zero and increments each time a motion of the corresponding axis/axis group terminates.

GMOT resets to zero each time the axis group is created or split.

Accessibility

Read-Only.

Related ACSPL+ Commands

[GROUP](#), [SPLIT](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.9.18 GMQU**Description**

GMQU is an integer array, with one element for each axis in the system, and defines the total number of motions in the motion queue including the currently executing motion. The maximum motion queue per axis is 5.

Tag

59

Comments

GMQU is valid only if one of the following is true:

- > Single-axis motion in progress
- > The axis is a leading axis in a group and motion in the group is in progress

After power-up **GMQU** is zero. The variable is incremented by one each time a new motion of the corresponding axis/axis group is issued. It is decremented by one each time a motion of the corresponding axis/axis group terminates.

GMQU resets to zero each time an axis is regrouped, i.e., a group that contains the axis is created or split-up.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.9.19 GMTYPE**Description**

GMTYPE is an integer array, with one element for each axis in the system. The MPU updates **GMTYPE** each time a motion involving the corresponding axis or axis group starts or terminates.

Tag

60

Comments

GMTYPE is updated according to the type of the motion as follows:

- 0 - no motion
- 1 - **PTP** motion
- 2 - **MPTP...ENDS** motion
- 3 - **TRACK** motion
- 4 - **MSEG...ENDS** motion
- 5 - **JOG** motion
- 6 - **SLAVE** motion
- 7 - **PATH...ENDS** motion
- 8 - **PVSPLINE...ENDS** motion
- 10 - **XSEG...ENDS** motion
- 11 - **BPTP** motion
- 12 - **BSEG...ENDS** motion
- 43 - **BPTP/2** motion using 20 kHz control

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.9.20 GPATH**Description**

GPATH is a real array, with one element for each axis in the system. **GPATH** defines the current path value, defined as the distance from the motion origin to the current motion point, or in the case of Extended Segmented Motion, the distance from the beginning of the first segment.

Tag

61

Comments

GPATH updates each MPU cycle if one of the following is true:

- > Single-axis motion in progress
- > The axis is a leading axis in a group and motion in the group is in progress

If either of these conditions is not true, **GPATH** retains its previous value.

For single-axis motion, **GPATH** defines a positive distance from the initial point of the motion.

If the axis is a leading axis, **GPATH** defines a vector distance along the trajectory from the motion origin.



GPATH, **GVEL**, **GACC**, **GJERK**, **GPHASE**, and **GRTIME** variables are updated while the motion is in progress.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.21 GPHASE

Description

GPHASE is an integer array, with one element for each axis in the system. **GPHASE** defines the current phase of a motion.

Tag

62

Comments

GPHASE can have the following values:

0 - no motion

1 - acceleration buildup

2 - constant acceleration

3 - acceleration finishing

4 - constant velocity

5 - deceleration buildup

6 - constant deceleration

7 - deceleration finishing

8 - kill deceleration

9 - asynchronous phase of master-slave motion

10 - synchronous phase of master-slave motion

11 - stalled phase of master-slave motion.

13 - dwell phase in **JOG** or **MPTP...ENDS** motions, or no defined target point in **PATH...ENDS**, **PVSPLINE...ENDS** or **MPTP...ENDS** motions.

31 – Jerk buildup when acceleration buildup

- 32 – Jerk finishing when acceleration buildup
- 33 – Jerk buildup when acceleration finishing
- 34 – Jerk finishing when acceleration finishing
- 35 – Jerk buildup when deceleration buildup
- 36 – Jerk finishing when deceleration buildup
- 37 – Jerk buildup when deceleration finishing
- 38 – Jerk finishing when deceleration finishing



GPATH, **GVEL**, **GACC**, **GJERK**, **GPHASE**, and **GRTIME** variables are updated while the motion is in progress.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.9.22 GRTIME

Description

GRTIME is a real array, with one element for each axis in the system. **GRTIME** defines an estimated value of time (in milliseconds) remaining until the end of the current motion.

Tag

63

Comments

GRTIME updates each MPU cycle if one of the following is true:

- > Single-axis motion in progress
- > The axis is a leading axis in a group and motion in the group is in progress

GRTIME does not update if the motion is **JOG** or **MASTER SLAVE**. If **GRTIME** does not update, it retains its previous value.

Normally, 1-2 msec after motion starts, **GRTIME** accepts the correct value. In rare cases, the **GRTIME** value remains high during motion phases 1 and 2, and accepts correct value at the beginning of phase 3.



GPATH, **GVEL**, **GACC**, **GJERK**, **GPHASE**, and **GRTIME** variables are updated while the motion is in progress.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.23 GSEG**Description**

GSEG is an integer array, with one element for each axis in the system. **GSEG** defines the ordinal number of the currently executing segment.

Tag

64

Comments

GSEG updates only under one of the following conditions:

- > Single-axis motion in progress, or
- > The axis is a leading axis in a group and motion in the group is in progress

If either of these conditions is not true, **GSEG** retains its previous value.

GSEG updates as follows:

- > If the current motion in the axis/axis group is not **MSEG...ENDS**, the **GSEG** value is -1.
- > The value resets to zero when a multi segment motion starts
- > The value increments each time when the motion passes from one segment to the next.
- > The value decrements each time the motion passes from one segment to the previous (possible only in master-slave motion).
- > Because the motion returns to the start point in cyclic motion, **GSEG** may appear greater than the number of a segment in the motion, if the motion overruns the segment sequence in positive direction.
- > For master-slave cyclic motion, **GSEG** may appear negative, if the motion overruns the segment sequence in a negative direction.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.9.24 GSFREE

Description

GSFREE is an integer array, with one element for each axis in the system. **GSFREE** is updated for the leading axis with the number of free cells in the segment queue.

Tag

65

Comments

If **GSFREE** is zero, the segment queue is full and the next coming **POINT** or **MPOINT** command will be delayed until the required number of cells are freed.

Accessibility

Read-Only

Related ACSPL+ Variable

[GSEG](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.9.25 GSNAP

Description

GSNAP is a real array, with one element for each axis in the system, and is used for defining the current calculated snap vector snap of group motion.

Tag

407

3.9.26 GVEC

Description

GVEC is a real array, with one element for each axis in the system. **GVEC** is updated each MPU cycle, if a motion involving the axis is in progress. If the motion is not in progress, **GVEC** retains its previous value.

Tag

66

Comments

In single-axis motion, **GVEC** = 1 or -1, depending on the motion direction.

In multi-axis group motion, **GVEC** values for all axes in the group are updated each MPU cycle and together build up a tangent vector for the motion trajectory.

GVEC can also be used for retrieving a tangent vector.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.27 GVEL

Description

GVEL is a real array, with one element for each axis in the system, and is used for deriving the vector velocity of a group motion.

For example when the three axes 0, 1 and 2 are moving as a group, the **GVEL** is calculated by:

$$GVEL = \sqrt{(Velocity_0)^2 + (Velocity_1)^2 + (Velocity_2)^2}$$



GPATH, **GVEL**, **GACC**, **GJERK**, **GPHASE**, and **GRTIME** variables are updated while the motion is in progress

Tag

67

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.28 JERK

Description

JERK is a real array, with one element for each axis in the system, and is used for defining the jerk of the motion profile.

Syntax

JERK(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 2.22507e-308 to 1.79769e+308, Default = 2e+007. |

Tag

80

Comments

For single-axis motion, **JERK** defines the axis jerk. If the axis is a leading axis in a group, **JERK** defines vector jerk of the common motion.

If **JERK** is changed when a motion is in progress, the change does not affect currently executing motions, or motions that were created before the change.

Accessibility

Read-Write



JERK values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[IMM](#), and all motion commands where the profile is generated by the controller.

Related ACSPL+ Variables

[ACC](#), [DEC](#), [KDEC](#), [VEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetJerk, SetJerk, SetJerkImm

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_GetJerk, SetJerk, acsc_SetJerkImm

3.9.29 KDEC**Description**

KDEC is a real array, with one element for each axis in the system, and is used for defining deceleration when a motion is killed by the user or fails due to a fault.

Syntax

KDEC(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308, Default = 100000. |

Tag

81

Comments

For single-axis motion, the value defines axis deceleration. If the axis is a leading axis in a group, **KDEC** defines the vector deceleration when the common motion is killed or fails.

If **KDEC** is changed when a motion is in progress, the change does not affect currently executing or motions that were created before the change.

Accessibility

Read-Write



KDEC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[ACC](#), [DEC](#), [JERK](#), [VEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, SetKillDeceleration, GetKillDeceleration, SetKillDecelerationImm

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_SetKillDeceleration, acsc_GetKillDeceleration, acsc_SetKillDecelerationImm

3.9.30 LPOS

Description

LPOS is a real array, with one element for each axis in the system, the elements of which store the axis position in the Local Coordinate System.

Syntax

LPOS(*axis_index*) = *value*

Arguments

| | |
|-------------------|--------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid values are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2.22507e-308 to 1.79769e+308; default = 50. |

Tag

372

Accessibility

Read-Only

Related ACSPL+ Variables

"FPOS" on page 353, "RPOS" on page 375, "APOS" on page 348

.NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.9.31 MPOS

Description

MPOS is a real array, with one element for each axis in the system, and defines the current master position value for the axis in user units.

Tag

89

Comments

MASTER must precede **MPOS** for the specified axis. **MPOS** updates each controller cycle according to the formula specified in **MASTER**.

Accessibility

Read-Only

Related ACSPL+ Commands

[MASTER](#), [SLAVE](#)

Related ACSPL+ Variables

[FPOS](#), [F2POS](#), [APOS](#)

COM Library Methods and .NET Library Methods

ReadVariable, SetMaster, Slave

C Library Functions

acsc_ReadReal, acsc_SetMaster, acsc_Slave

3.9.32 MTIMEA

Description

MTIMEA returns the time elapsed from start of motion up to first entering the settled zone, using **SETTLEA** and **TARGRADA** to determine the time and radius required for settling.

Syntax

```
Value = MTIMEA (Axis_Index)
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|

Tag

341

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADA**. This indicates if the motor is in the target zone, if it stays in the target zone for at least **SETTLEA** time - bit **MST.#INTARGA** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEA** and is only valid when the **#INTARGA** bit is on.

Related ACSPL+ Variables

TARGRADA, SETTLEA, MST(axis_index).#INTARGA

Accessibility

Read-only



MSTIMEA values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.



MSTIMEA is only updated if the Move & Settle feature is enabled by using **SETCONF(318)** to enable either single mode or auto mode.

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadReal()

3.9.33 *MSTIMEB*

Description

MSTIMEB returns the time elapsed from start of motion up to first entering the settled zone, using **SETTLEB** and **TARGRADB** to determine the time and radius required for settling.

Syntax

```
Value = MSTIMEB (Axis_Index)
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|

Tag

342

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADB**. This indicates if the motor is in the target zone, if it stays in the target zone for at least **SETTLEB** time - bit **MST.#INTARGB** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIME_B** and is only valid when the **#INTARGB** bit is on.

Related ACSPL+ Variables

TARGRADB, SETTLEB, MST(axis_index).#INTARGB

Accessibility

Read-only



MSTIMEB values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.



MSTIMEB is only updated if the Move & Settle feature is enabled by using **SETCONF(318)** to enable either single mode or auto mode.

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadReal()

3.9.34 *MSTIMEC*

Description

MSTIMEC returns the time elapsed from start of motion up to first entering the settled zone, using **SETTLEC** and **TARGRADC** to determine the time and radius required for settling.

Syntax

```
Value = MSTIMEC(Axis_Index)
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|

Tag

343

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADC**. This indicates if we the motor is in the target zone, if it stays in the target zone for at least **SETTLEC** time - bit **MST.#INTARGC** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEC** and is only valid when the **#INTARGC** bit is on.

Related ACSPL+ Variables

TARGRADC, SETTLEC, MST(axis_index).#INTARGC

Accessibility

Read-only



MSTIMEC values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.



MSTIMEC is only updated if the Move & Settle feature is enabled by using **SETCONF(318)** to enable either single mode or auto mode.

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadReal()

3.9.35 NVEL

Description

NVEL is a real array, with one element for each axis in the system, and is used for specifying the start and the end velocities for an axis in stepper motor applications.

Syntax

NVEL(axis_index) = value

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from 0 to 1.79769e+308, Default = 0. |

Tag

91

Comments

1. An **NVEL** element affects the motion of the corresponding axis and the multi-axis motions if the axis is a leading axis in the group.
2. If an element is zero, the normal motion profile starts from zero velocity and finishes at zero velocity.
If an element is non-zero, at the beginning of motion the velocity immediately jumps to the **NVEL** value and then continues the regular motion profile. At the end of the motion, the motion approaches the final point at the velocity specified by **NVEL**, and then immediately drops to zero. For example, **KILL/KILLALL** and **HALT** slow the velocity to the value specified in **NVEL**, and then the velocity drops to zero.

Accessibility

Read-Write



NVEL values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection.

Related ACSPL+ Commands

IMM, and all motion commands where the profile is generated by the controller.

Related ACSPL+ Variables

VEL

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.9.36 PE**Description**

PE is a real array, with one element for each axis in the system, and is used for displaying the difference between **RPOS** and **FPOS** (the current position error) denoting a noncritical position error.



The **PE** value is valid only if the motor is enabled.

Tag

98

Accessibility

Read-Only

Related ACSPL+ Commands

All motion commands.

Related ACSPL+ Variables

FPOS, RPOS, FAULT

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.37 PPOS**Description**

PPOS is real array, with one element for each axis in the system, the elements of which store the current desired motor reference position. Unlike **RPOS**, this holds the current value, rather than a value taking into account the delay for reading the actual current position from the encoder.

Tag

364

Comments

When the motor is disabled, **RPOS = FPOS**.

This variable is supported in ADK versions 2.70 and higher.

Accessibility

Read-Only

Related ACSPL+ Commands

SET, CONNECT, and all motion commands.

Related ACSPL+ Variables

FPOS, RVEL, RACC

COM Library Methods and .NET Library Methods

ReadVariable, GetRPosition, SetRPosition

C Library Functions

acsc_ReadReal, acsc_GetRPosition, acsc_SetRPosition

3.9.38 PPOSCOMP**Description**

PPOSCOMP is real array, with one element for each axis in the system, the elements of which store the current desired motor reference position including dynamic error compensation. Unlike **RPOSCOMP**, this holds the current value, rather than a value taking into account the delay for reading the actual current position from the encoder.

Tag

365

Comments

This variable is supported in ADK versions 2.70 and higher.

Accessibility

Read-Only

Related ACSPL+ Commands

All motion commands.

Related ACSPL+ Variables

PPOS, FPOS, RPOS, APOS, PE, DECOMP

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.39 PRFLTIME

Description

PRFLTIME is a real array, the size of which is determined by the total number of axes in the system. It holds the time(In milliseconds) passed from the moment that the move starts until the motion profile ends.

Syntax

```
value = PRFLTIME(index)
```

Arguments

| | |
|-------|---------------------------------------------------------------------------------------------------------------|
| index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|-------|---------------------------------------------------------------------------------------------------------------|

Tag

405

Comments

- > The **PRFLTIME** variable is updated with the latest profile time for the relevant axes.
- > If we have multi-axes move, the profile time for all the involved axes is the same and will be updated once the profile has ended for all the axes.
- > A kill/error event is regarded as profile end.
- > If the profile starts and ends in the same cycle (for example, move from the current axis position to the same position), the profile time will be 0.

Related ACSPL+ Variables

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable()

C Library Functions

acsc_ReadReal()

3.9.40 RACC

Description

RACC is a real array, with one element for each axis in the system, and defines the current reference acceleration value for the motor in user-defined units.

Tag

106

Comments

RACC updates each controller cycle, and is calculated by digital differentiation of [RVEL](#).

Accessibility

Read-Only

Related ACSPL+ Commands

All motion related commands.

Related ACSPL+ Variables

[RVEL](#), [RPOS](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

Real

3.9.41 RJERK

Description

RJERK is a real array, with one element for each axis in the system, and defines the current calculated reference jerk value for the motor in user-defined units.

RJERK is updated each controller cycle and is calculated as digital differentiation of [RACC](#).

Tag

259

Accessibility

Read-Only

3.9.42 ROFFS

Description

ROFFS is a real array, with one element for each axis in the system, the elements of which store the Reference Offset.

Tag

107

Comments

As long as the motor is in the default connection (`MFLAGS(axis).#DEFCON = 1`), offset **ROFFS** is zero. Once a user specifies connect formula such as:

`CONNECT RPOS(0) = F(...)`

the controller calculates offset **ROFFS(0)** to prevent a sudden change in **RPOS(0)** that may cause the motor to jump. The controller then calculates:

RPOS(0) = F(...) + ROFFS(0)

each controller cycle.

The controller recalculates **ROFFS** to prevent motor jump when the commands `CONNECT`, `SET`, `ENABLE/ENABLE ALL`, `DISABLE/DISABLEALL`, `KILL` are executed. **ROFFS** reads the current value of the offset.



Watching the **ROFFS** value facilitates development and debugging of applications with complex kinematics.

Accessibility

Read-Only

Related ACSPL+ Commands

`CONNECT`, `SET`, `ENABLE/ENABLE ALL`, `DISABLE/DISABLEALL`, `KILL/KILLALL`

Related ACSPL+ Variables

`MFLAGS(axis_index).#DEFCON` (bit 17 = Default Connection)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.43 RPOS**Description**

RPOS is real array, with one element for each axis in the system, the elements of which store the current desired motor reference position.

Tag

108

Comments

RPOS updates each MPU cycle according to the connection specified for the motor, see `CONNECT`.

When the motor is disabled, **RPOS** = **FPOS**.

Accessibility

Read-Only

Related ACSPL+ Commands

SET, CONNECT, and all motion commands.

Related ACSPL+ Variables

FPOS, RVEL, RACC

COM Library Methods and .NET Library Methods

ReadVariable, GetRPosition, SetRPosition

C Library Functions

acsc_ReadReal, acsc_GetRPosition, acsc_SetRPosition

3.9.44 RPOSCOMP**Description**

RPOSCOMP is real array, with one element for each axis in the system, the elements of which store the current desired motor reference position including dynamic error compensation.



RPOSCOMP updates every controller cycle according to the configured dynamic error compensation, see **ERRORMAP1D**, **ERRORMAPN1D**, **ERRORMAP2D**, **ERRORMAPN2D**.

When the dynamic error compensation is not configured, **RPOSCOMP** = **RPOS**.

Tag

348

Accessibility

Read-Only

Related ACSPL+ Commands

All motion commands.

Related ACSPL+ Variables

FPOS, RPOS, APOS, PE, DECOMP

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.45 RPOSDEL**Description**

RPOSDEL shows the actual delay time that is currently set. The delay value is rounded (ceiling function) to 50 µsec. At the beginning of the motion, which is delayed, the parameter indicates the specified delay. At the end of the motion, the delay is gradually reduced to zero.

Syntax

```
INT val = RPOSDEL
```

Tag

329

Accessibility

Read-Only

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadInteger()

3.9.46 RSNAP

Description

RSNAP is a real array, with one element for each axis in the system, and is used for defining the current calculated reference snap (jerk derivation) in user-defined units.

Tag

408

3.9.47 RVEL

Description

RVEL is a real array, with one element for each axis in the system, the elements of which store the current motor reference velocity in user-defined units.

Tag

109

Accessibility

Read-Only

Related ACSPL+ Commands

All motion commands.

Related ACSPL+ Variables

[RPOS](#), [RACC](#), [FVEL](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetRVelocity

C Library Functions

acsc_ReadReal, acsc_GetRVelocity

3.9.48 SETTLEA

Description

SETTLEA allows you to set the time to wait inside the TARGRADA before triggering MST.#INTARGA.

Syntax

```
SETTLEA(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308, 1.79769e+308, Default = 0. |

Tag

338

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADA**. This indicates if the motor is in the target zone, if it stays in the target zone for at least **SETTLEA** time - bit **MST.#INTARGA** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEA** and is only valid when the **#INTARGA** bit is on.

Related ACSPL+ Variables

TARGRADA, MST(axis_index).#INTARGA, MSTIMEA

Accessibility

Read-Write



SETTLEA values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection .

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.49 SETTLEB

Description

SETTLEB allows you to set the time to wait inside the TARGRADB before triggering MST.#INTARGB.

Syntax

```
SETTLEB(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308, 1.79769e+308, Default = 0. |

Tag

339

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADB**. This indicates if the motor is in the target zone, if it stays in the target zone for at least **SETTLEB** time - bit **MST.#INTARGB** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEB** and is only valid when the **#INTARGB** bit is on.

Related ACSPL+ Variables

TARGRADB, MST(axis_index).#INTARGB, MSTIMEB

Accessibility

Read-Write



SETTLEB values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection .

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.50 SLSFF

Description

SLSFF is a SNAP feed forward parameter.

3.9.51 SETTLEC

Description

SETTLEC allows you to set the time to wait inside the TARGRADC before triggering MST.#INTARGC.

Syntax

```
SETTLEC(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308, 1.79769e+308, Default = 0. |

Tag

340

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADC**. This indicates if we the motor is in the target zone, if it stays in the target zone for at least **SETTLEC** time - bit **MST.#INTARGC** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEC** and is only valid when the **#INTARGC** bit is on.

Related ACSPL+ Variables

TARGRADC, MST(axis_index).#INTARGC, MSTIMEC

Accessibility

Read-Write



SETTLEC values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection .

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.52 SNAP

Description

SNAP is a real array with one element for each axis in the system, and is used for defining the snap (jerk derivative) of the motion profile in user-defined units.

Syntax

```
SNAP(axis_index) = value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value value ranges from 2.22507e-308 to 1.79769e+308, Default = 1e+009 |

Tag

406

3.9.53 STLTIMEA**Description**

STLTIMEA is a real array, with one element for each axis in the system. Each element holds the last settling time of the axis, i.e., the time passed since the profile generation has been completed up to first entering the settled zone, using **SETTLEA** and **TARGRADA** to determine the time and radius required for settling.

Syntax

```
value = STLTIMEA(axis_index)
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|

Tag

392

Comments

- > In cases where the motion settled within the specified radius before the profile has finished, **STLTIME** = 0. Otherwise, **STLTIME** [axis_index] = **MSTIME**[axis_index] - **PRFLTIME** [axis_index].
- > This feature has two operating modes, which are controlled through **SETCONF** with key 318, where index indicates the axis number and the value the mode, value of 0 is the default and deactivates the feature. See **SETCONF** documentation for more details.
- > Only motions which support **TPOS** may be used to measure the settle time. These motions are: **PTP**, **MPTP**, and **TRACK**.
- > Once the **MST**(axis_index).#INTARGA bit is set(1), the **STLTIMEA** variable holds an updated value of the axis settling time.

Related ACSPL+ Variables

MSTIMEA, **TARGRADA**, **SETTLEA**, **MST**(axis_index).#INTARGA

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.54 STLTIMEB**Description**

STLTIMEB is a real array, with one element for each axis in the system. Each element holds the last settling time of the axis, i.e., the time passed since the profile generation has been completed up to first entering the settled zone, using **SETTLEB** and **TARGRADB** to determine the time and radius required for settling.

Syntax

```
value = STLTIMEB(axis_index)
```

Arguments

axis_index

Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

Tag

393

Comments

- > In cases where the motion settled within the specified radius before the profile has finished, **STLTIME** = 0. Otherwise, **STLTIME** [axis_index] = **MSTIME**[axis_index] - **PRFLTIME** [axis_index].
- > This feature has two operating modes, which are controlled through **SETCONF** with key 318, where index indicates the axis number and the value the mode, value of 0 is the default and deactivates the feature. See **SETCONF** documentation for more details.
- > Only motions which support **TPOS** may be used to measure the settle time. These motions are: **PTP**, **MPTP**, and **TRACK**.
- > Once the **MST**(axis_index).#INTARGB bit is set(1), the **STLTIMEB** variable holds an updated value of the axis settling time.

Related ACSPL+ Variables**MSTIMEB, TARGRADB, SETTLEB, MST**(axis_index).#INTARGB**Accessibility**

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.55 STLTIMEC

Description

STLTIMEC is a real array, with one element for each axis in the system. Each element holds the last settling time of the axis, i.e., the time passed since the profile generation has been completed up to first entering the settled zone, using **SETTLEC** and **TARGRADC** to determine the time and radius required for settling.

Syntax

```
value = STLTIMEC(axis_index)
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|

Tag

392

Comments

- > In cases where the motion settled within the specified radius before the profile has finished, **STLTIME** = 0. Otherwise, **STLTIME** [axis_index] = **MSTIME**[axis_index] - **PRFLTIME** [axis_index].
- > This feature has two operating modes, which are controlled through **SETCONF** with key 318, where index indicates the axis number and the value the mode, value of 0 is the default and deactivates the feature. See **SETCONF** documentation for more details.
- > Only motions which support **TPOS** may be used to measure the settle time. These motions are: **PTP**, **MPTP**, and **TRACK**.
- > Once the **MST**(axis_index).#INTARGC bit is set(1), the **STLTIMEC** variable holds an updated value of the axis settling time.

Related ACSPL+ Variables

MSTIMEC, **TARGRADC**, **SETTLEC**, **MST**(axis_index).#INTARGC

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.9.56 TARGRADA

Description

TARGRADA is a variable designed to define the target radius around which you wish the motion to settle.

Syntax

```
TARGRADA(Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308, 1.79769e+308, Default = 1. |

Tag

335

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADA**. This indicates if the motor is in the target zone, if it stays in the target zone for at least **SETTLEA** time - bit **MST.#INTARGA** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEA** and is only valid when the **#INTARGA** bit is on.

Related ACSPL+ Variables

SETTLEA, MST(axis_index).#INTARGA, MSTIMEA

Accessibility

Read-Write



TARGRADA values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.57 TARGRADB

Description

TARGRADB is a variable designed to define the target radius around which you wish the motion to settle.

Syntax

```
TARGRADB(Axis_Index) = Value
```


Arguments

| | |
|-------------|---------------------------------------------------------------------------------------------------------------|
| Axis_ Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308, 1.79769e+308, Default = 1. |

Tag

336

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADB**. This indicates if we the motor is in the target zone, if it stays in the target zone for at least **SETTLEB** time - bit **MST.#INTARGB** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEB** and is only valid when the **#INTARGB** bit is on.

Related ACSPL+ Variables

SETTLEB, MST(axis_index).#INTARGB, MSTIMEB

Accessibility

Read-Write



TARGRADB values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.58 TARGRADC

Description

TARGRADC is a variable designed to define the target radius around which you wish the motion to settle.

Syntax

```
TARGRADC (Axis_Index) = Value
```

Arguments

| | |
|-------------|---------------------------------------------------------------------------------------------------------------|
| Axis_ Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value ranges from 2.22507e-308, 1.79769e+308, Default = 1. |

Tag

337

Comments

We compare the distance between current position and target position to the given target radii, represented by **TARGRADC**. This indicates if we the motor is in the target zone, if it stays in the target zone for at least **SETTLEC** time - bit **MST.#INTARGC** is raised, respectively, and depending on the operating mode – further inspection will be stopped or continued.

The time from the beginning of motion until first entering the settled zone is represented by **MSTIMEC** and is only valid when the **#INTARGC** bit is on.

Related ACSPL+ Variables

SETTLE_C, MST(axis_index).#INTARGC, MSTIMEC

Accessibility

Read-Write



TARGRADC values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio > Toolbox > Application Development > Protection.

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadReal(), acsc_WriteReal()

3.9.59 TPOS**Description**

TPOS is a real array, with one element for each axis in the system, and is used for defining or updating the target position in TRACK motion.

Syntax**TPOS**(axis_index) = value**Arguments**

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from -1.79769e+308 to 1.79769e+308, Default = 0. |

Tag

135

Comments

The controller update occurs as follows:

- > When the controller executes **PTP** motion, the axes' target coordinates are stored in the **TPOS** elements.
- > During **MPTP...ENDS** motion, the controller updates the target coordinates each time motion to the next point starts.
- > When the controller executes **TRACK** motion, the axes' target coordinates are stored in the **TPOS** elements.

Accessibility

Read-Write



TPOS values cannot be modified if protection is applied to this variable through SPiiPlus
 MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Commands

TRACK

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, Track

C Library Functions

acsc_ReadReal, acsc_WriteReal, acsc_Track

3.9.60 VEL

Description

VEL is a real array, with one element for each axis in the system, and is used for defining the default velocity of the motion profile. If a motion command does not specify a specific velocity, the default value is used.

Syntax

VEL(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from -1.79769e+308 to 1.79769e+308, Default = 10000. |

Tag

139

Comments

For single-axis motion, the value defines axis velocity. If the axis is a leading axis in a group, its value defines a vector velocity of common motion.

If **VEL** is changed when a motion is in progress, the change does not affect currently executing motions or motions that were created before the change.

Accessibility

Read-Write



VEL values cannot be modified if protection is applied to this variable through **SPiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands

[IMM](#), and all motion commands where the profile is generated by the controller.

Related ACSPL+ Variables

[ACC](#), [DEC](#), [JERK](#), [KDEC](#),

COM Library Methods and .NET Library Methods

[ReadVariable](#), [WriteVariable](#), [SetVelocity](#), [GetVelocity](#), [SetVelocityImm](#)

C Library Functions

[acsc_ReadReal](#), [acsc_WriteReal](#), [acsc_SetVelocity](#), [acsc_GetVelocity](#), [acsc_SetVelocityImm](#)

3.10 Program Execution Control Variables

The Program Execution Control variables are:

| Name | Description |
|------------------------|----------------------------|
| ONRATE | Autoroutine Rate |
| PCHARS | Program Size in Characters |
| PERL | Program Error Line |
| PERR | Program Error |
| PEXL | Executed Line |
| PFLAGS | Program Flags |
| PLINES | Number of Lines |
| PRATE | Program Rate |
| PST | Program State |

3.10.1 ONRATE

Description

ONRATE is an integer array with one element for each program buffer plus one for the D-Buffer and is used for controlling the autoroutine execution rate.

Syntax

ONRATE(*buffer_index*) = *value*

Arguments

| | |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer_index</i> | buffer index - a number between 0 and the total number of buffers minus one (the highest number is that of the D-Buffer). |
| <i>value</i> | value ranges from 1 to 10, Default = 1. |

Tag

93

Comments

ONRATE is set through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Program Manager** → **Program Buffer Parameters**.

When an autoroutine executes in the program buffer, the execution rate is **ONRATE** lines per each MPU cycle. The normal rate of program execution (when no autoroutine is activated) is defined by **PRATE**.

Accessibility

Read-Write



ONRATE values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.10.2 PCHARS

Description

PCHARS is an integer array with one element for each program buffer plus one for the D-Buffer that stores the total number of characters stored in the buffer.

Tag

95

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.10.3 PERL

Description

PERL is an integer array with one element for each program buffer plus one for the D-Buffer that stores the line number where the error occurred.

Tag

99

Comments

If an error occurs during ACSPL+ program execution, the controller stores the line number where the error occurred in the corresponding element of the **PERL** array.

Accessibility

Read-Only

Related ACSPL+ Variables

[PERR](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetProgramError

C Library Functions

acsc_ReadInteger, acsc_GetProgramError

3.10.4 PERR

Description

PERR is an integer array with one element for each program buffer plus one for the D-Buffer that stores an error code.

Error codes are found in [Table 6-2](#) and [Table 6-3](#).

Tag

100

Comments

If an error occurs during ACSPL+ program execution, the controller stores the error code in the corresponding element of the **PERR** array.

Accessibility

Read-Only

Related ACSPL+ Variables[PERL](#)**COM Library Methods and .NET Library Methods**

ReadVariable, GetProgramError

C Library Functions

acsc_ReadInteger, acsc_GetProgramError

3.10.5 PEXL**Description**

PEXL is an integer array with one element for each program buffer plus one for the D-Buffer that stores the number of the currently executed line.

Tag

101

Comments

PEXL stores the number of the currently executed line in the buffer. If the program has not executed, the variable reads zero.

Accessibility

Read-Only

Related ACSPL+ Variables[PERL](#), [PERR](#)**COM Library Methods and .NET Library Methods**

ReadVariable

C Library Functions

acsc_ReadInteger



PEXL does not support D-Buffer.

3.10.6 PFLAGS**Description**

PFLAGS is an integer array with one element for each program buffer plus one for the D-Buffer, each element of which contains a set of bits that defines the behavior of the program buffer.

When the #JIT(Just In Time) bit is ON the controller waits for a file loading operation (via MMI or host application program) and can start executing the commands in the buffer immediately after the loading process is completed.

The Just in Time buffer acts as a FIFO for the ACSPL+ commands which are read from the file. After the file is loaded, the buffer can be executed any number of times.



The **#JIT** bit can be set to ON only if the buffer is empty. Error 3204 "JIT and Dynamic modes require the buffer to be empty" is returned if the buffer is not empty.

Syntax


PFLAGS(buffer_index).(bit) = 0|1

Arguments

| | |
|---------------------|--------------------------------------------------------------------------|
| buffer_index | buffer index - a number between 0 and 64 (64 being the D-Buffer). |
| bit | Table 3-9 |

Table 3-9. PFLAGS Bit Description 1

| Bit Name | No. | Description |
|----------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #NOAUTO | 0 | 0 (default): Autoroutines (if exist in the buffer) are enabled. 1: Autoroutines (if exist in the buffer) are disabled. |
| #NOEDIT | 1 | 0 (default): User program can be viewed and edited. 1: User program can be viewed but cannot be edited. <div data-bbox="651 1061 740 1164" data-label="Image"> </div> Supported after applying protection, see Protection Wizard in the <i>MMI Application Studio User Guide</i> . |
| #DYNAMIC | 2 | 0 (default): Buffer works in normal order. 1: Buffer works in Dynamic mode. |
| #JIT | 3 | 0 (default): Buffers works in normal order. 1: Buffer works in JIT mode. |
| #PRIVLG | 4 | 0 (default): Buffer works in normal order. 1: Sets the buffer as privileged which means that the program in the buffer can change the values of protected variables, start and stop other ACSPL+ programs, and execute any other action that in a regular buffer would cause a protection violation. |
| #DEBUG | 5 | 0 (default): Buffer works in normal order. 1: Not applicable (obsolete option) |

| Bit Name | No. | Description |
|----------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #NOVIEW | 6 | <p>0 (default): The program is visible in the buffer. 1: The program in the buffer hidden from being viewed.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>Supported after applying protection, see Protection Wizard in the <i>MMI Application Studio User Guide</i>.</p> </div> |

Comments

The bit cannot be applied for the D-Buffer. Attempt to setting the bit for the D-Buffer will result in error 3200 "JIT is not allowed for D-Buffer".

Tag

102

Accessibility

Read-Write



PFLAGS values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

Example

```
PFLAGS (0) #JIT=1
```

3.10.7 PLINES**Description**

PLINES is an integer array with one element for each program buffer plus one for the D-Buffer each element of which contains the total number of lines stored in the associated buffer.

Tag

103

Accessibility

Read-Only

Related ACSPL+ Variables

[PCHARS](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger

3.10.8 PRATE**Description**

PRATE is an integer array with one element for each program buffer, each element of which is used for defining the program execution rate for that buffer.

Syntax**PRATE**(*buffer_index*) = *value***Arguments**

| | |
|---------------------|---------------------------------------------------------------------------------------------------------|
| <i>buffer_index</i> | buffer index - a number between 0 and N, N being the number of buffers in the product or system. |
| <i>value</i> | value ranges from 1 to 10, Default = 1. |

Tag

104

Comments

PRATE is set through **SPIiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Program Manager** → **Program Buffer Parameters**.

PRATE defines the program execution rate. The execution rate is **PRATE** lines per each MPU cycle.

PRATE is used only if no autoroutine is activated in the buffer. While an autoroutine is executed, **ONRATE** defines execution rate.

For example, if the controller is configured so that **PRATE(2)** is 1, but **ONRATE(2)** is 4, the program in Buffer 2 will be executed one line per one controller cycle, and any autoroutine specified in Buffer 2 that interrupts the program will be executed four lines per one controller cycle. When the **RET** command that terminates the autoroutine is executed, the controller switches back to the rate of one line per one cycle.

Accessibility

Read-Write



PRATE values cannot be modified if protection is applied to this variable through **SPIiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables**ONRATE****COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.10.9 PST

Description

PST is an integer array with one element for each program buffer plus one for the D-Buffer each element of which contains a set of bits that display the current state of the given program buffer.

The **PST** bits are detailed in [Table 3-10](#).

Table 3-10. PST Bit Description

| Bit Name | No. | Description |
|-----------|-----|---------------------------------------------------------------------------------------------------------------------------|
| #COMPILED | 0 | 0: Program in buffer is not compiled 1: Program in buffer is compiled |
| #RUN | 1 | 0: Program is not running. 1: Program is running. |
| #SUSPEND | 2 | 0: Program in buffer is not suspended. 1: Program is suspended after STEP or due to a breakpoint in debug mode. |
| #DEBUG | 5 | 0: Buffer works in normal order (default). 1: Not applicable (obsolete option) |
| #AUTO | 7 | 0: Autoroutine is not running 1: Autoroutine is running. |

Tag

105

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable, GetProgramState

C Library Functions

acsc_ReadInteger, acsc_GetProgramState

3.11 Safety Control Variables

The Safety Control variables are:

| Name | Description |
|----------|-----------------------------------------------------------------------------------------------------------------------|
| ECALERR | Contains Ethernet slave error code |
| CERRA | Critical Position Error (Accelerating) |
| CERRI | Critical Position Error (Idle) |
| CERRV | Critical Position Error (Velocity) |
| DELI | Delay on Transition to Idle State |
| DELV | Delay on Transition to Velocity State |
| E_ERR | An integer array for each axis containing the encoder error code identified during the encoder initialization process |
| ECEXTST | EtherCAT state of the SPiiPlusES slave |
| ECEXTERR | SPiiPlusES EtherCAT error code (based on Application Level Error Code). |
| FAULT | Faults |
| FAULTSIM | Fault Simulation |
| ECST | Contains Ethernet status |
| ECERR | Contains Ethernet error code |
| FDEF | Default Response Mask |
| FMASK | Fault Mask |
| HLLROUT | Integer array used for mapping between an axis hardware left limit to a specified digital input bit |
| HRLROUT | Integer array used for mapping between an axis hardware righthlimit to a specified digital input bit |
| MERR | Motor Error |
| SAFIN | Safety Inputs |
| SAFINI | Safety Inputs Inversion |
| S_ERR | System Error |
| S_FAULT | System Faults |

| Name | Description |
|-----------------------|------------------------------------------------------------------------------------------------------------|
| <code>S_FDEF</code> | System Default Response Mask |
| <code>S_FMASK</code> | System Fault Mask |
| <code>S_SAFIN</code> | System Safety Inputs |
| <code>S_SAFINI</code> | System Safety Inputs Inversion |
| <code>SS1TIME</code> | Store SS1-t channel A time between emergency stop request and drive switching to torque off mode |
| <code>SS12TIME</code> | Store SS1-t channel B time between emergency stop request and drive switching to torque off mode |
| <code>STODELAY</code> | Configures the delay time between the STO fault indication and the default response (disable) to the fault |
| <code>SYNC</code> | Slave synchronization indicator |

3.11.1 `E_ERR`

Description

`E_ERR` is an integer array for each axis. It contains the encoder error code that was identified during the encoder initialization process.

The encoder errors range from 5121 to 5128 and are latched in the `E_ERR` variable. The error codes are specified in [Table 6-5](#) in the Error Codes section.

Comments

This variable is supported in version 3.00 and higher.

Accessibility

Read-Only

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadInteger()

3.11.2 `ECALERR`

Description

`ECALERR` is an integer array for each EtherCAT slave in the configuration (ENI file). It contains the AL Status Code error of the slave, when the value "0" indicates no error.

Tag

328

Accessibility

Read-Only

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadInteger()

The error codes are defined according to AL Status Code (ETG 1020).

The error codes are listed in [Table 6-8](#).

3.11.3 ECERR**Description**

ECERR is a scalar variable containing an EtherCAT error code. The EtherCAT error codes are given in [Table 6-7](#).

Syntax**ECERR****Arguments**

None

Tag

239

Comments

Any EtherCAT error sets **ECST.#OP** to false and the error code is latched in **ECERR**.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.11.4 ECEXTERR**Description**

ECEXTERR is a scalar (INT) variable representing the EtherCAT error code of the SPiiPlusES (based on Application Level Error Code). The error code range is 7000-7999. The error codes are specified in [Table 6-8](#).



Can be used only by SPiiPlusES.

Syntax

EEXTERR

Arguments

None

Comments

If the controller is not SPiiPlusES, the value is always 0.

Tag

321

Accessibility

Read-Only

Com Library Methods and .NET Library Methods

ReadVariable

C Library Functions

Acsc_ReadInteger

3.11.5 EEXTST

Description

EEXTST is a scalar (INT) variable representing the EtherCAT state of the SPiiPlusES slave. The state is reflected in the relevant bits.



Can be used only by SPiiPlusES.

| Bit | Designator | Description |
|-----|------------|-------------------------------------------------------|
| 0 | #INIT | SPiiPlusES is in INIT state |
| 1 | #BOOT | SPiiPlusES is in BOOT state (currently not supported) |
| 2 | #PREOP | SPiiPlusES is in PREOP state |
| 3 | #SAFEOP | SPiiPlusES is in SAFEOP state |
| 4 | #ES_OP | SPiiPlusES is in OP state |
| 5 | #ES_DC | Distributed Clocks are ON |
| 6 | #WATCHDOG | PDI Watchdog status. 0: expired, 1: reloaded |
| 7 | #LNKPORTA | Physical Link Port A |

| Bit | Designator | Description |
|-----|------------|--------------------------------|
| 8 | #LNKPORTB | Physical Link Port B |
| 9 | #EXTSYNC | Synchronized to External Clock |

Syntax**ECEXTST****Arguments**

None

Comments

Comments

If the controller is not SPiiPlusES, , MPU3U, or IDMsm, all bits are 0. If external EtherCAT master is not connected, the slave is in the INIT state.

Bit 9 #EXTSYNC is relevant for IDMsm only.

Tag

320

Accessibility

Read-Only

Com Library Methods and .NET Library Methods

ReadVariable

C Library Functions

Accsc_ReadInteger

3.11.6 ECST**Description**

ECST is a scalar variable affecting the EtherCAT state. The EtherCAT state is reflected in the first six bits as given in [Table 3-11](#).

Table 3-11. ECST Bits

| Bit | Designator | Description |
|-----|------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | #SCAN | The scan process was performed successfully, that is, the Master was able to detect what devices are connected to it. |
| 1 | #CONFIG | There is no deviation between XML and actual setup. The Master succeeded to initialize the network by steps described in configuration file. |

| Bit | Designator | Description |
|-----|------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 2 | #INITOK | All bus devices are successfully set to INIT state. The Master started all devices to the initial state. |
| 3 | #CONNECTED | Indicates valid Ethernet cable connection to the master. The physical link of EtherCAT cable is OK on the Master side. |
| 4 | #INSYNC | If DCM is used, indicates synchronization between the Master and the bus. |
| 5 | #OP | The EtherCAT bus is operational. The Master successfully turned each Slave into full operational mode and the bus is ready for full operation. |
| 6 | #DCSYNC | Distributed clocks are synchronized. |
| 7 | #RINGMODE | Ring topology mode is selected. |
| 8 | #RINGCOMM | Ring Communication is active. |
| 9 | #EXTCONN | External clock is connected |
| 10 | #DCXSYNC | External clock/slaves are synchronized |

Syntax

`ECST.bit_designator = 1|0`

Arguments

None

Tag

238

Comments

All bits (except #INSYNC in some cases) should be true for proper bus functioning.

For monitoring the bus state, checking bit #OP is sufficient. Any bus error will reset the #OP bit.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.11.7 FAULT

Description

FAULT is an integer array, with one element for each axis in the systems, the elements of which contain a set of bits that stores axis-related fault bits.

The fault bits are detailed in [Table 3-12](#).

Table 3-12. Axis Fault Bits

| Bit | Fault | Fault Description |
|-----|---------|---------------------------------------------------------------------------------------------------------------------------------|
| 0 | #RL | Hardware Right Limit. 1 = Right limit switch is activated. |
| 1 | #LL | Hardware Left Limit. 1 = Left limit switch is activated. |
| 2 | #NT | Network Error. 1 = EtherCAT network error is activated. |
| 4 | #HOT | Motor Overheat. 1 = Motor's temperature sensor indicates overheat. |
| 5 | #SRL | Software Right Limit. 1 = Axis reference position (RPOS) is greater than the software right limit margin (SRLIMIT). |
| 6 | #SLL | Software Left Limit. 1 = Axis reference position (RPOS) is less than the software left limit margin (SLLIMIT). |
| 7 | #ENCNC | Encoder Not Connected. 1 = Primary encoder (for digital encoder type only) is not connected. |
| 8 | #ENC2NC | Encoder 2 Not Connected. 1 = Secondary encoder (for digital encoder type only) is not connected. |
| 9 | #DRIVE | Drive Alarm. 1 = Signal from the drive reports a failure. |
| 10 | #ENC | Encoder Error. 1 = Primary encoder miscounts. |

| Bit | Fault | Fault Description |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11 | #ENC2 | Encoder 2 Error. 1 = Secondary encoder miscounts. |
| 12 | #PE | Position Error. 1 = Position error (PE) has occurred. PE is defined by the following variables: <ul style="list-style-type: none"> > ERRI - Maximum position error while the axis is idle > ERRV - Maximum position error while the axis is moving with constant velocity > ERRA - Maximum position error while the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to ERRI > DELV - Delay on transition from ERRA to ERRV |
| 13 | #CPE | >Critical Position Error. 1 = Position error (#PE) exceeds the value of the critical limit. #CPE errors occur outside normal range of operation and #CPE > #PE. The critical limit depends on the axis state and is defined by the following variables: <ul style="list-style-type: none"> > CERRI if the axis is idle (not moving) > CERRV if the axis is moving with constant velocity > CERRA if the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to CERRI > DELV - Delay on transition from ERRA to CERRV |
| 14 | #VL | Velocity Limit. 1 = Absolute value of the reference velocity (RVEL) exceeds the limit defined by the XVEL parameter. |
| 15 | #AL | Acceleration Limit. 1 = Absolute value of the reference acceleration (RACC) exceeds the limit defined by the XACC parameter. |
| 16 | #CL | Current Limit. |

| Bit | Fault | Fault Description |
|-----|---------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| | | 1 = RMS current calculated in the Servo Processor exceeds the limit value defined by the XRMS parameter. |
| 17 | #SP | Servo Processor Alarm. 1 = Axis Servo Processor loses its synchronization with the MPU. The fault indicates a fatal problem in the controller. |
| 18 | #STO | Safe Torque Off 1 = STO is active |
| 20 | #HSSINC | Hssi Not Connected. 1 = HSSI module is not connected. |

Tag

47

Comments

FAULT indicates axis related fault bits as detected by the safety mechanism. When each of the faults is active (such as Left Limit), the corresponding fault bit becomes = 1 while the fault is active, and automatically reverts to 0 when the fault is no longer active.

- > Each fault can be masked by **FMASK**.
- > The logic of some faults can be inverted by **SAFINI**.
- > The default response of each fault can be disabled by **FDEF**. In this case, any customized default response can be implemented by autoroutines - see **ON...RET**.

For a list of **S_FAULT** related system fault bits see [S_FAULT Fault Bits](#)

Accessibility

Read-Only



FAULT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[S_FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetFault

C Library Functions

acsc_ReadInteger, acsc_GetFault

3.11.8 FAULTSIM

Description

FAULTSIM is an integer array, with one element for each axis in the system. Each such element consists of bits representing the errors in the axis or the system itself. This variable is used to simulate these faults and raising a certain bit will trigger the fault the axis/system thereby raising **FAULT** and **S_FAULT**.

The fault bits are indicated in the following table:

| Bit | Fault | Fault Description |
|-------------|---------|------------------------------------------------------------------------------------------------------------------------------|
| Axis Faults | | |
| 0 | #RL | Hardware Right Limit 1 = Right limit switch is activated |
| 1 | #LL | Hardware Left Limit 1 = Left limit switch is activated. |
| 2 | #NT | Network Error 1 = EtherCAT network error is activated |
| 4 | #HOT | Motor Overheat 1 = Motor's temperature sensore indicates overheating |
| 5 | #SRL | Software Left Limit 1 = Axis reference postion (RPOS) is greater than the software right limit margin (SRLIMIT). |
| 6 | #SLL | Software Left Limit 1 = Axis reference position (RPOS) is less than the software left limit margin (SLLIMIT). |
| 7 | #ENCNC | Encoder Not Connected 1 = Primary encoder (for digital encoder type only) is not connected. |
| 8 | #ENC2NC | Encoder 2 Not Connected 1 = Secondary encoder (for digital encoder type only) is not connected. |
| 9 | #DRIVE | Drive Fault 1 = Signal from the drive reports a failure |
| 10 | #ENC | Encoder Error 1 = Primary encoder miscounts. |

| Bit | Fault | Fault Description |
|-------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis Faults | | |
| 11 | #ENC2 | Encoder 2 Error 1 = Secondary encoder miscounts. |
| 12 | #PE | 1 = Position error (PE) has occurred. PE is defined by the following variables: <ul style="list-style-type: none"> > ERRI - Maximum position error while the axis is idle > ERRV - Maximum position error while the axis is moving with constant velocity > ERRA - Maximum position error while the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to ERRI > DELV - Delay on transition from ERRA to ERRV |
| 13 | #CPE | Critical Position Error 1 = Position error (#PE) exceeds the value of the critical limit. #CPE errors occur outside normal range of operation and #CPE > #PE. The critical limit depends on the axis state and is defined by the following variables: <ul style="list-style-type: none"> > CERRI if the axis is idle (not moving) > CERRV if the axis is moving with constant velocity > CERRA if the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to CERRI > DELV - Delay on transition from ERRA to CERRV |
| 14 | #VL | Velocity Limit 1 = Absolute value of the reference velocity (RVEL) exceeds the limit defined by the XVEL parameter. |
| 15 | #AL | Acceleration Limit 1 = Absolute value of the reference acceleration (RACC) exceeds the limit defined by the XACC parameter. |
| 16 | #CL | Current Limit 1 = RMS current calculated in the Servo Processor exceeds the limit value defined by the XRMS parameter. |

| Bit | Fault | Fault Description |
|----------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis Faults | | |
| 17 | #SP | Servo Processor 1 = Axis Servo Processor loses its synchronization with the MPU. The fault indicates a fatal problem in the controller. |
| 20 | #HSSINC | HSSI Not Connected 1 = HSSI module is not connected. |
| System Faults | | |
| 23 | #EXTNT | External Network Error |
| 24 | #TEMP | MPU Overheat Fault Activated at CPU temperature > 90°C or System temperature > 70°C Default response - none |
| 25 | #PROG | Program Fault 1 = Run time error occurs in one of the executing ACSPL+ programs. |
| 26 | #MEM | Memory Overflow 1 = User application requires too much memory. |
| 27 | #TIME | MPU Overuse 1 = User application consumes too much time in the controller cycle. |
| 28 | #ES | Hardware Emergency Stop 1 = ES signal is activated. |
| 29 | #INT | Servo Interrupt 1 = The servo interrupt that defines the controller cycle is not generated. The fault indicates a fatal controller problem. |
| 30 | #INTGR | File Integrity 1 = The integrity of the user application in controller RAM is checked by the controller at power-up and whenever an #IR Terminal command is issued. |

| System Faults | | |
|---------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31 | #FAILURE | <p>Component Failure</p> <p>1 = An MC4U hardware component other than the drive, such as the Power Supply, I/O card, or encoder card, has failed.</p> |
| Axis Faults | | |
| | | <p>When the bus voltage is not supplied to the MC4U, a component failure fault is reported. The fault is system wide and prevents all axes from operating unless the fault is masked or bus voltage is supplied to the power supply.</p> <p>When a component failure is reported, the affected power supply is identified by its address. To determine the faulty unit, use the MMI System Viewer and Diagnostics</p> |

Tag

332

Comments

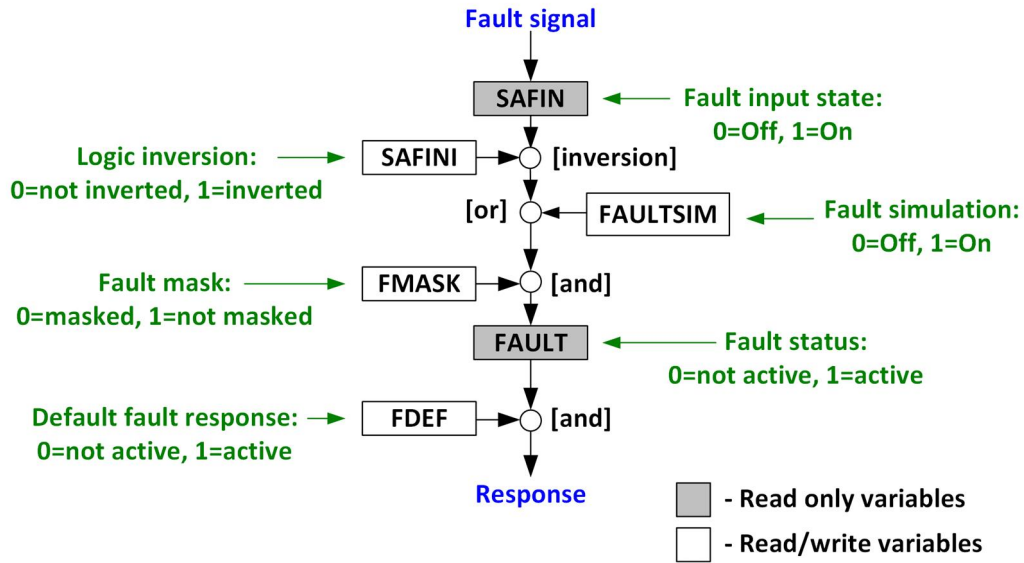
This variable allows you to simulate Axis & System faults, by raising a certain bit on a given axis – you effectively set the fault and the appropriate response will be triggered.



In order to reset some faults you must use **FCLEAR** command, setting the **FAULTSIM** bit to 0 might not be enough.

FAULTSIM variable is not saved to flash and will be reset upon controller restart.

FAULTSIM variable does not interact with **SAFINI/SAFIN** variables and therefore does not affect the variables.



Accessibility

Read/Write

Related ACSPL+ Variables

FAULT, S_FAULT, SAFINI, S_SAFINI, FMASK, S_FMASK

3.11.9 FDEF

Description

FDEF is an integer array, with one element for each axis in the system, the elements of which contain a set of bits used for setting a default response to an axis fault.

Syntax

`FDEF(axis_index)[bit_designator] = value`

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The FDEF bit designators are given in FDEF Bit Description . |
| <i>value</i> | value ranges from -2147483648 to 2147483647, Default = -1. |

Table 3-13. FDEF Bit Description

| Bit | Fault | Fault Description | Default Response (FDEF) |
|-----|---------|-----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | #RL | Hardware Right Limit 1 = Right limit switch is activated. | The controller kills the violating axis. As long as the fault is active, the controller kills any motion that tries to move the axis in the direction of the limit; however, motion within the permissible range is allowed. |
| 1 | #LL | Hardware Left Limit 1 = Left limit switch is activated. | Same as for #RL. |
| 2 | #NT | Network Error 1 = EtherCAT network error detected. | Halts all program buffers and waits for receipt of network Sync signal. |
| 4 | #HOT | >Motor Overheat 1 = Motor's temperature sensor indicates overheat. | None. |
| 5 | #SRL | Software Right Limit 1 = Axis reference position (RPOS) is greater than the software right limit margin (SRLIMIT). | The controller kills the violating axis. As long as the fault is active, the controller kills any motion that tries to move the axis in the direction of the limit. Motion in the direction out of the limit is allowed. |
| 6 | #SLL | Software Left Limit 1 = Axis reference position (RPOS) is less than the software left limit margin (SLLIMIT). | Same as #SRL. |
| 7 | #ENCNC | Encoder Not Connected 1 = Primary encoder (for digital encoder type only) is not connected. | The controller disables the violating axis. |
| 8 | #ENC2NC | Encoder 2 Not Connected | No default response. |

| Bit | Fault | Fault Description | Default Response (FDEF) |
|-----|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | 1 = Secondary encoder (for digital encoder type only) is not connected. | |
| 9 | #DRIVE | Drive Fault 1 = Signal from the drive reports a failure. | The controller disables the violating axis. This fault is only detected when the axis is enabled. To catch this fault in an ACSPL+ program, write an autoroutine. |
| 10 | #ENC | Encoder Error 1 = Primary encoder miscounts. | The controller disables the violating axis. The faults remain active until the user resolves the problems and enables the axis again or executes FCLEAR . |
| 11 | #ENC2 | Encoder 2 Error 1 = Secondary encoder miscounts. | Same as #ENC. |
| 12 | #PE | Position Error. 1 = Position error (PE) has occurred. PE is defined by the following variables: <ul style="list-style-type: none"> > ERRI - Maximum position error while the axis is idle > ERRV - Maximum position error while the axis is moving with constant velocity > DELI - Delay on transition from ERRA to ERRI > DELV - Delay on transition from | None. |

| Bit | Fault | Fault Description | Default Response (FDEF) |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| | | ERRA to ERRV | |
| 13 | #CPE | <p>Critical Position Error</p> <p>1 = Position error (#PE) exceeds the value of the critical limit.</p> <p>#CPE errors occur outside normal range of operation and #CPE > #PE.</p> <p>The critical limit depends on the axis state and is defined by the following variables:</p> <ul style="list-style-type: none"> > CERRI if the axis is idle (not moving) > CERRV if the axis is moving with constant velocity > CERRA if the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to CERRI > DELV - Delay on transition from CERRA to ERRV | The controller disables the violating axis. |
| 14 | #VL | <p>>Velocity Limit</p> <p>1 = Absolute value of the reference velocity (RVEL) exceeds the limit defined by the XVEL parameter.</p> | The controller kills the violating axis. |

| Bit | Fault | Fault Description | Default Response (FDEF) |
|-----|---------|--------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| 15 | #AL | >Acceleration Limit 1 = Absolute value of the reference acceleration (RACC) exceeds the limit defined by the XACC parameter. | The controller kills the violating axis. |
| 16 | #CL | Current Limit 1 = RMS current calculated in the Servo Processor exceeds the limit value defined by the XRMS XpParameter. | The controller disables the violating axis. |
| 17 | #SP | Servo Processor Alarm 1 = Axis Servo Processor loses its synchronization with the MPU. The fault indicates a fatal problem in the controller. | The controller disables the violating axis and kills the motion that involves the axis. |
| 18 | #STO | Safe Torque Off 1 = STO is activated | Blocks the PWM signals to the power stage of the drive |
| 20 | #HSSINC | Hssi Not Connected 1 = HSSI module is not connected. | None. |

Tag

48

Comments

When an **FDEF** bit = 1, the controller executes the default response when the corresponding fault occurs. If the **FDEF** bit = 0, the default response is disabled.

Not every fault has a default response. For a fault that has no default response, the corresponding **FDEF** bit is inoperative.

Accessibility

Read-Write



FDEF values cannot be modified if protection is applied to this variable through **SPiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

[ReadVariable](#), [WriteVariable](#), [GetResponseMask](#), [SetResponseMask](#), [GetFaultMask](#), [SetFaultMask](#)

C Library Functions

[acsc_ReadInteger](#), [acsc_WriteInteger](#), [acsc_GetResponseMask](#), [acsc_SetResponseMask](#), [acsc_GetFaultMask](#), [acsc_SetFaultMask](#)

3.11.10 FMASK

Description

FMASK is an integer array, with one element for each axis in the system, the elements of which contain a set of bits used for enabling or disabling each axis fault bit.

Syntax

FMASK(*axis_index*)[*.bit_designator*] = *value*

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The FDEF bit designators are given in FMASK Bit Description . |
| <i>value</i> | value ranges from -2147483648 to 2147483647, Default=1040414435. |

Table 3-14. FMASK Bit Description

| Bit | Fault | Fault Description |
|-----|-------|----------------------------------------------------------------------|
| 0 | #RL | Hardware Right Limit 1 = Right limit switch is activated. |
| 1 | #LL | Hardware Left Limit 1 = Left limit switch is activated. |
| 2 | #NT | Network Error 1 = EtherCAT network error detected. |
| 4 | #HOT | Motor Overheat 1 = Motor's temperature sensor indicates overheat. |

| Bit | Fault | Fault Description |
|-----|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5 | #SRL | Software Right Limit 1 = Axis reference position (RPOS) is greater than the software right limit margin (SRLIMIT). |
| 6 | #SLL | Software Left Limit 1 = Axis reference position (RPOS) is less than the software left limit margin (SLLIMIT). |
| 7 | #ENCNC | Encoder Not Connected 1 = Primary encoder (for digital encoder type only) is not connected. |
| 8 | #ENC2NC | Encoder 2 Not Connected 1 = Secondary encoder (for digital encoder type only) is not connected. |
| 9 | #DRIVE | Drive Fault 1 = Signal from the drive reports a failure. |
| 10 | #ENC | Encoder Error 1 = Primary encoder miscounts. |
| 11 | #ENC2 | Encoder 2 Error 1 = Secondary encoder miscounts. |
| 12 | #PE | Position Error 1 = Position error (PE) has occurred. PE is defined by the following variables: <ul style="list-style-type: none"> > ERRI - Maximum position error while the axis is idle > ERRV - Maximum position error while the axis is moving with constant velocity > ERRA - Maximum position error while the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to ERRI > DELVDELV - Delay on transition from ERRAERRA to ERRV |

| Bit | Fault | Fault Description |
|-----|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13 | #CPE | <p>Critical Position Error</p> <p>1 = Position error (#PE) exceeds the value of the critical limit.</p> <p>#CPE errors occur outside normal range of operation and #CPE > #PE.</p> <p>The critical limit depends on the axis state and is defined by the following variables:</p> <ul style="list-style-type: none"> > CERRI if the axis is idle (not moving) > CERRV if the axis is moving with constant velocity > CERRA if the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to CERRI > DELV - Delay on transition from CERRA to ERRV |
| 14 | #VL | <p>Velocity Limit</p> <p>1 = Absolute value of the reference velocity (RVEL) exceeds the limit defined by the XVEL parameter.</p> |
| 15 | #AL | <p>Acceleration Limit</p> <p>1 = Absolute value of the reference acceleration (RACC) exceeds the limit defined by the XACC parameter.</p> |
| 16 | #CL | <p>Current Limit</p> <p>1 = RMS current calculated in the Servo Processor exceeds the limit value defined by the XRMS parameter.</p> |
| 17 | #SP | <p>Servo Processor Alarm</p> <p>1 = Axis Servo Processor loses its synchronization with the MPU. The fault indicates a fatal problem in the controller.</p> |
| 18 | #STO | <p>Safe Torque Off</p> <p>1 = STO is active</p> |
| 20 | #HSSINC | <p>Hssi Not Connected</p> <p>1 = HSSI module is not connected.</p> |

Tag

51

Comments

The default value = 1 and causes the controller to check for the fault associated with that bit, as follows:

0 = the corresponding **FAULT** bit is disabled.

1 = the corresponding **FAULT** is enabled and examined each MPU cycle.

Accessibility

Read-Write



FMASK values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [S_FDEF](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetResponseMask, SetResponseMask, GetFaultMask, SetFaultMask

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_GetResponseMask, acsc_SetResponseMask, acsc_GetFaultMask, acsc_SetFaultMask

3.11.11 HLLROUT

Description

HLLROUT is an integer array with one element for each axis in the system, and it is used for mapping the hardware left limit of an axis to a specified digital input bit (ACSPL+ **IN**).

Syntax

HLLROUT(Axis_Index) = value

Arguments

| | |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis | Designates the specific axis. Valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value is a 4-digit number (decimal): <NN00> where: <ul style="list-style-type: none"> > NN - digital input index (00-99) > 00 – specified input bit (00-31) The default value is -1, in which case mapping will not occur. |

Tag

379

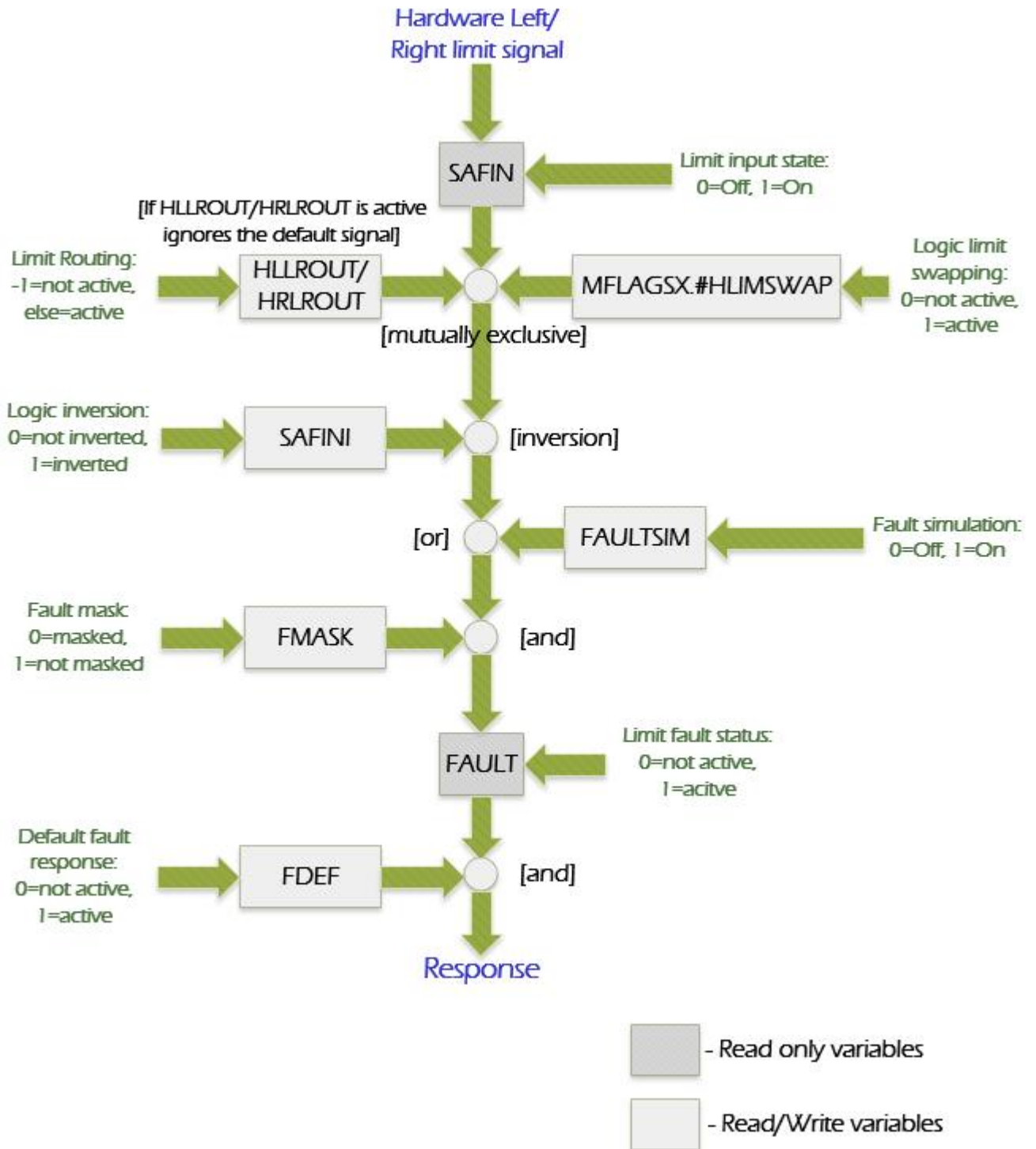
Comments

A value of -1 disables the mapping of the digital input to the hardware limit and restores the default behavior.

The following errors are supported:

- > Error 3329: "Invalid value, digital input index should range between 0-99 and bit index should range between 0-31"
- > Error 3332: "Hardware limit swapping(**MFLAGSX.#HLIMSWAP**) and limit routing are mutually exclusive"

The following diagram illustrates the behavior of the firmware when the left limit signal is set:



Related ACSPL+ Variables
 HLLROUT, IN

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.11.12 HRLROUT

Description

HRLROUT is an integer array with one element for each axis in the system, and it is used for mapping the hardware right limit of an axis to a specified digital input bit (ACSPL+ **IN**).

Syntax

HRLROUT(Axis_Index) = value

Arguments

| | |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Axis | Designates the specific axis. Valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Value | Value is a 4-digit number (decimal): <NN00> where: > NN - digital input index (00-99) > 00 – specified input bit (00-31) The default value is -1, in which case mapping will not occur. |

Tag

379

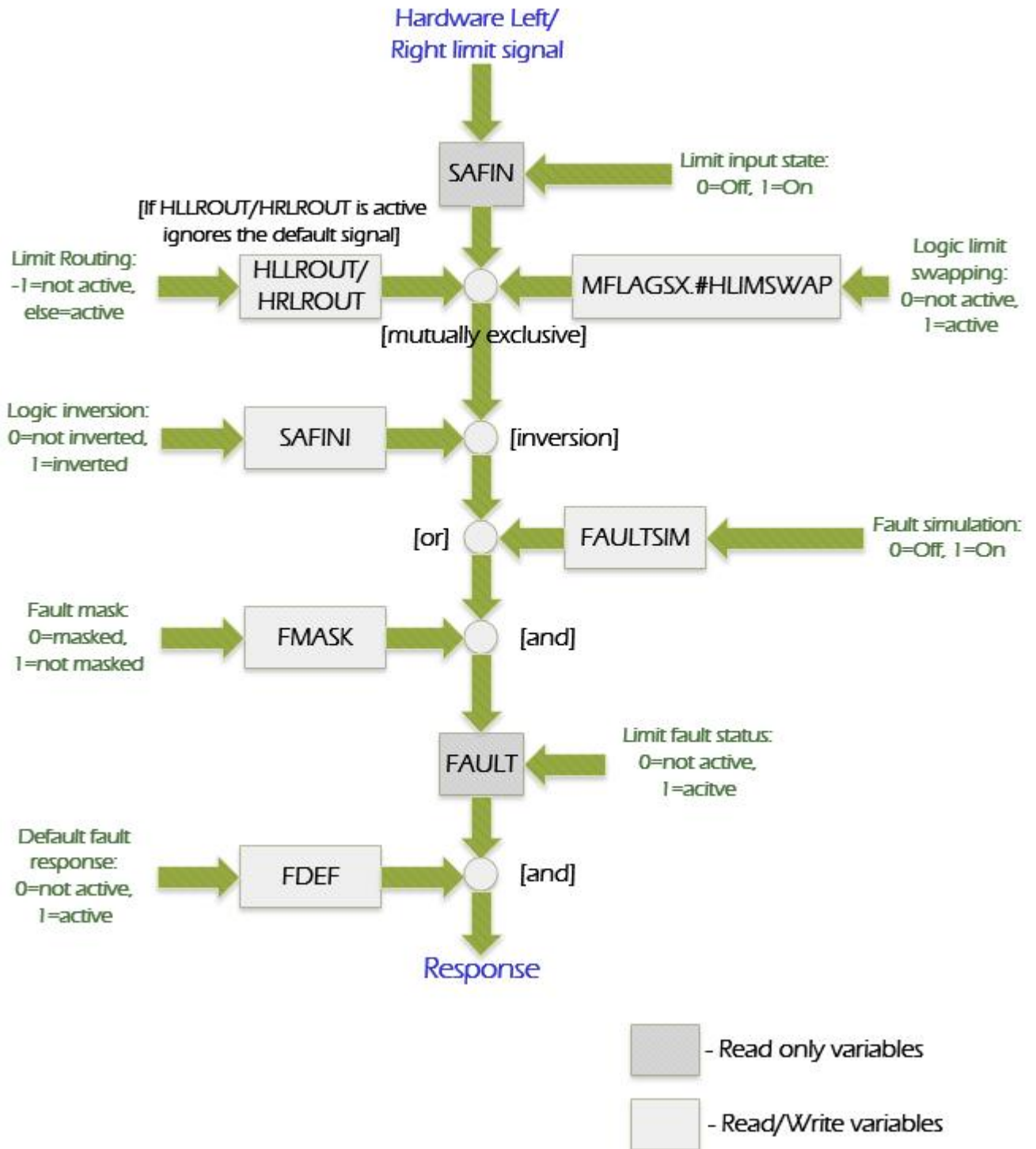
Comments

A value of -1 disables the mapping of the digital input to the hardware limit and restores the default behavior.

The following errors are supported:

- > Error 3329: "Invalid value, digital input index should range between 0-99 and bit index should range between 0-31"
- > Error 3332: "Hardware limit swapping(**MFLAGSX.#HLIMSWAP**) and limit routing are mutually exclusive"

The following diagram illustrates the behavior of the firmware when the left limit signal is set:



Related ACSPL+ Variables

HLLROUT, IN

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.11.13 MERR

Description

MERR is an integer array, with one element for each axis in the system, the elements of which store a code indicating the termination cause of the last motion of an axis.



An error code= 5027, "Motor Failed: Servo Processor Alarm" fault is activated when an axes does not have a physical drive associated to it.

The **MERR** return values are listed in [Table 6-4](#)

Tag

86

Comments

MERR is updated every time the axis motion is terminated. MERR stores the last termination code until either [FCLEAR](#) or [ENABLE/ENABLE ALL](#) is executed.

Accessibility

Read-Only

Related ACSPL+ Commands

[FCLEAR](#)

Related ACSPL+ Variables

[PERR](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetMotorError

C Library Functions

acsc_ReadInteger, acsc_GetMotorError

3.11.14 SAFIN

Description

SAFIN is an integer array, with one element for each axis in the system, the elements of which contain a set of bits that indicates the raw state, before processing, of the axis safety inputs.

SAFIN(<axis>).17 (#STO1) and **SAFIN(<axis>).18 (#STO2)** present the status of the emergency stop request (24V switched off).

For the products that supports SS1-t:

SAFIN(<axis>).19 (#SS11) and **SAFIN(<axis>).20 (#SS12)** present the status of switching in the torque off mode (5V switched off).

SAFIN(<axis>).16 (#SS1TERR) presents the status of SS1-t timing error.

The value of each element in the array ranges from -2147483648 to 2147483647, Default=0.

Tag

121

Comments

1. The **SAFIN** uses the same bit numbers as in **S_SAFIN** and as the corresponding faults in **FAULT** and **S_FAULT**.
2. **SAFIN** is normally read-only. However, when working with the Simulator, read/write is permitted to simulate safety inputs.
3. Only the **SAFIN** bits below are valid.

| Bit Name | No. | Description |
|----------|-----|--------------------------------------------------------------|
| #RL | 0 | Hardware Right Limit |
| #LL | 1 | Hardware Left Limit |
| #HOT | 4 | Motor Overheat |
| #DRIVE | 9 | Drive Fault |
| #SS1TERR | 16 | Status of SS1-t timing error |
| #STO1 | 17 | Status of the emergency stop request (24V switched off) |
| #STO2 | 18 | Status of the emergency stop request (24V switched off) |
| #SS11 | 19 | Status of switching in the torque off mode (5V switched off) |
| #SS12 | 20 | Status of switching in the torque off mode (5V switched off) |
| #ES | 28 | Hardware Emergency Stop 1 = ES signal is activated |
| #COMP | 31 | Component Failure |

Accessibility

Read-Only



SAFIN can be written to when working with the SPiPlus Simulator.

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.11.15 SAFINI

Description

SAFINI is an integer array, with one element for each axis in the system, the elements of which contain a set of bits defining the active state of the **axis** safety input variable (**SAFIN**) specifying inversion of the signal input logic, if required.

Syntax

SAFINI(*axis_index*)[*bit_designator*] = *value*

Arguments

| | |
|-----------------------|-------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0, 1, 2, .. up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | The valid SAFINI bits are given in Table 3-15 . |
| <i>value</i> | value ranges from -2147483648 to 2147483647, Default=0. |

Table 3-15. SAFINI Valid Bits

| Bit Name | No. | Description |
|----------|-----|-------------------------------------------------------|
| #RL | 0 | Hardware Right Limit |
| #LL | 1 | Hardware Left Limit |
| #HOT | 4 | Motor Overheat |
| #DRIVE | 9 | Drive Fault |
| #ES | 28 | Hardware Emergency Stop 1 = ES signal is activated |

Tag

122

Comments

1. When a **SAFINI** bit=0, the corresponding signal is not inverted and the high voltage state is considered active.
2. When a **SAFINI** bit=1, the bit is inverted and the low voltage state is considered active.

Accessibility

Read-Write



SAFINI values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection.

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [S_SAFINI](#).

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, SetSafetyInputPortInv, GetSafetyInputPortInv

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_SetSafetyInputPortInv, acsc_GetSafetyInputPortInv

3.11.16 S_ERR**Description**

S_ERR is a scalar integer that contains the code of the initialization error set during powerup.

The error codes are specified in [Table 6-6](#).

Tag

113

Accessibility

Read-Only

Related ACSPL+ Variables

None

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.11.17 S_FAULT**Description**

S_FAULT is a scalar integer variable consisting of a set of bits equating to the occurrence of faults. **S_FAULT** has two categories of bits, Axis Faults and System Faults (faults that are not related to any specific axis).

The **S_FAULT** bits are described in [Table 3-16](#).

Table 3-16. S_FAULT Fault Bits

| Bit | Fault | Fault Description |
|-------------|---------|--------------------------------------------------------------------------------------------------------------------------------|
| Axis Faults | | |
| 0 | #RL | Hardware Right Limit 1 = Right limit switch is activated. |
| 1 | #LL | Hardware Left Limit 1 = Left limit switch is activated. |
| 2 | #NT | Network Error. 1 = EtherCAT network error is activated. |
| 4 | #HOT | Motor Overheat 1 = Motor's temperature sensor indicates overheat. |
| 5 | #SRL | Software Right Limit 1 = Axis reference position (RPOS) is greater than the software right limit margin (SRLIMIT). |
| 6 | #SLL | Software Left Limit 1 = Axis reference position (RPOS) is less than the software left limit margin (SLLIMIT). |
| 7 | #ENCNC | Encoder Not Connected 1 = Primary encoder (for digital or SinCos encoder types) is not connected. |
| 8 | #ENC2NC | Encoder 2 Not Connected 1 = Secondary encoder (for digital or SinCos encoder types) is not connected. |
| 9 | #DRIVE | Drive Fault 1 = Signal from the drive reports a failure. |
| 10 | #ENC | Encoder Error 1 = Primary encoder miscounts. |
| 11 | #ENC2 | Encoder 2 Error 1 = Secondary encoder miscounts. |

| Bit | Fault | Fault Description |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12 | #PE | <p>Position Error</p> <p>1 = Position error (PE) has occurred.</p> <p>PE is defined by the following variables:</p> <ul style="list-style-type: none"> > ERRI - Maximum position error while the axis is idle > ERRV - Maximum position error while the axis is moving with constant velocity > ERRA - Maximum position error while the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to ERRI > DELV - Delay on transition from ERRA to ERRV |
| 13 | #CPE | <p>Critical Position Error</p> <p>1 = Position error (#PE) exceeds the value of the critical limit.</p> <p>#CPE errors occur outside normal range of operation and #CPE > #PE.</p> <p>The critical limit depends on the axis state and is defined by the following variables:</p> <ul style="list-style-type: none"> > CERRI if the axis is idle (not moving) > CERRV if the axis is moving with constant velocity > CERRA if the axis is accelerating or decelerating > DELI - Delay on transition from ERRA to CERRI > DELV - Delay on transition from ERRA to CERRV |
| 14 | #VL | <p>Velocity Limit.</p> <p>1 = Absolute value of the reference velocity (RVEL) exceeds the limit defined by the XVEL parameter.</p> |
| 15 | #AL | <p>Acceleration Limit</p> <p>1 = Absolute value of the reference acceleration (RACC) exceeds the limit defined by the XACC parameter.</p> |
| 16 | #CL | <p>Current Limit</p> <p>1 = RMS current calculated in the Servo Processor exceeds the limit value defined by the XRMS parameter.</p> |
| 17 | #SP | Servo Processor Alarm |

| Bit | Fault | Fault Description |
|---------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | 1 = Axis Servo Processor loses its synchronization with the MPU. The fault indicates a fatal problem in the controller. |
| 18 | #STO | Safe Torque Off 1 = STO is active |
| 20 | #HSSINC | HSSI Not Connected 1 = HSSI module is not connected. |
| System Faults | | |
| 23 | #EXTNT | External Network Error |
| 24 | #TEMP | MPU Overheat Fault Activated at CPU temperature > 90°C or System temperature > 70°C Default response - none. |
| 25 | #PROG | Program Fault 1 = Run time error occurs in one of the executing ACSPL+ programs. |
| 26 | #MEM | Memory Overflow 1 = User application requires too much memory. |
| 27 | #TIME | MPU Overuse 1 = User application consumes too much time in the controller cycle. |
| 28 | #ES | Hardware Emergency Stop 1 = ES signal is activated. |
| 29 | #INT | Servo Interrupt 1 = The servo interrupt that defines the controller cycle is not generated. The fault indicates a fatal controller problem. |
| 30 | #INTGR | File Integrity 1 = The integrity of the user application in controller RAM is checked by the controller at power-up and whenever an #IR Terminal command is issued. |

| Bit | Fault | Fault Description |
|-----|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31 | #FAILURE | <p>Component Failure</p> <p>1 = An MC4U hardware component other than the drive, such as the Power Supply, I/O card, or encoder card, has failed.</p> <p>When the bus voltage is not supplied to the MC4U, a component failure fault is reported. The fault is system wide and prevents all axes from operating unless the fault is masked or bus voltage is supplied to the power supply.</p> <p>When a component failure is reported, the affected power supply is identified by its address. To determine the faulty unit, use the MMI System Viewer and Diagnostics</p> |

Tag

114

Comments

An **S_FAULT** bit, such as Left Limit, will be = 1 whenever one or more Left Limit fault bits are = 1. In this manner, **S_FAULT** provides an indication of the aggregate state of each **FAULT** bit.

Accessibility

Read-Only



S_FAULT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetFault

C Library Functions

acsc_ReadInteger, acsc_GetFault

3.11.18 S_FDEF**Description**

S_FDEF is a scalar integer variable consisting of a set of bits for defining the default response for the system faults contained in **S_FAULT**. **S_FDEF** is connected to **S_FAULT** in the same way that **FDEF** is connected with **FAULT**.


Syntax

S_FDEF[*bit_designator*] = *value*

Arguments

| | |
|-----------------------|-------------------------------------------------------------------------------------------|
| bit_designator | The S_FDEF bits and associated responses are given in Table 3-17 . |
| value | value ranges from -2147483648 to 2147483647, Default = 1. |

Table 3-17. S_FDEF Bit Description

| Bit | Fault | Fault Description | Default Response (S_FDEF) |
|-----|--------|-------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23 | #EXTNT | External Network Error | The controller disables all axes. |
| 24 | #TEMP | MPU Overheat | S_SETUP.#USGTEMP = 0: No response S_SETUP.#USGTEMP=1: Default response is to disable all axes |
| 25 | #PROG | Program Fault 1 = Run time error occurs in one of the executing ACSPL+ programs. | The controller kills all axes. |
| 26 | #MEM | Memory Overflow 1 = User application requires too much memory. | The controller kills all axes. |
| 27 | #TIME | MPU Overuse 1 = User application consumes too much time in the controller cycle. | No default response. |
| 28 | #ES | Hardware Emergency Stop 1 = ES signal is activated. | The controller disables all axes, and sets the offset of each axis to 0. <div style="border: 1px solid black; border-radius: 10px; padding: 5px; display: inline-block;">  It does not stop the program buffers. </div> |

| Bit | Fault | Fault Description | Default Response (S_FDEF) |
|-----|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| 29 | #INT | Servo Interrupt 1 = The servo interrupt that defines the controller cycle is not generated. The fault indicates a fatal controller problem. | The controller disables all axes. |
| 30 | #INTGR | File Integrity 1 = The integrity of the user application in controller RAM is checked by the controller at power-up and whenever an #IR Terminal command is issued. | No default response |
| 31 | #FAILURE | Component Failure 1 = An MC4U hardware component other than the drive, such as the Power Supply, I/O card, or encoder card, has failed. | No default response The user has to supply a user-defined fault response. |

Tag

115

Comments

The default value for all **S_FDEF** bits is 1, which enables the default response. If an **S_FDEF** bit = 0, the default response is disabled.

Accessibility

Read-Write



S_FDEF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetResponseMask, SetResponseMask, GetFaultMask, SetFaultMask

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_GetResponseMask, acsc_SetResponseMask, acsc_GetFaultMask, acsc_SetFaultMask

3.11.19 S_FMASK

Description

S_FMASK is scalar integer variable consisting of a set of bits for enabling or disabling the system faults contained in **S_FAULT**. **S_FMASK** is connected to **S_FAULT** in the same way that **FMASK** is connected with **FAULT**.

Syntax

S_FMASK[*bit_designator*] = *value*

Arguments

| | |
|-----------------------|--------------------------------------------------------------------------------------------|
| <i>bit_designator</i> | The S_FMASK bits and associated responses are given in Table 3-18 . |
| <i>value</i> | value ranges from -2147483648 to 2147483647, Default=0. |

Table 3-18. S_FMASK Bit Description

| Bit | Fault | Fault Description |
|-----|--------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 23 | #EXTNT | External Network Error |
| 24 | #TEMP | MPU Overheat |
| 25 | #PROG | Program Fault 1 = Run time error occurs in one of the executing ACSPL+ programs. |
| 26 | #MEM | Memory Overflow 1 = User application requires too much memory. |
| 27 | #TIME | MPU Overuse 1 = User application consumes too much time in the controller cycle. |
| 28 | #ES | Hardware Emergency Stop 1 = ES signal is activated. |
| 29 | #INT | Servo Interrupt 1 = The servo interrupt that defines the controller cycle is not generated. The fault indicates a fatal controller problem. |
| 30 | #INTGR | File Integrity |

| Bit | Fault | Fault Description |
|-----|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | 1 = The integrity of the user application in controller RAM is checked by the controller at power-up and whenever a #IR immediate command is issued. |
| 31 | #FAILURE | Component Failure 1 = An MC4U hardware component other than the drive, such as the Power Supply, I/O card, or encoder card, has failed. |

Tag

117

Comments

The **S_FMASK** default value = 1 and causes the controller to check for the fault associated with that bit, as follows:

0: The corresponding **FAULT** bit is disabled

1: The corresponding **FAULT** is enabled and examined each MPU cycle.

Accessibility

Read-Write



S_FMASK values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetFaultMask, SetFaultMask

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_GetFaultMask, acsc_SetFaultMask

3.11.20 S_SAFIN**Description**

S_SAFIN is a scalar integer variable that indicates the raw state of the **#ES** bit (Emergency Stop) input stored in the **SAFIN** variable and indicates the **#FAILURE** bit (system fault) stored in the **S_FAULT** variable.

The value ranges from -2147483648 to 2147483647, Default=0.

Tag

118

Comments

S_SAFIN uses the same bit numbers as in **SAFIN** and as the corresponding faults in **FAULT** and **S_FAULT**, but only the **#ES** bit is meaningful.



S_SAFIN can be written to when working with the SPiiPlus Simulator.

Accessibility

Read-Only

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [SAFINI](#), [S_SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, GetSafetyInputPort

C Library Functions

acsc_ReadInteger, acsc_GetSafetyInputPort

3.11.21 S_SAFINI

Description

S_SAFINI is a scalar integer variable used for defining the active state of the **system** safety input variable ([S_SAFIN](#)) specifying inversion of the signal input logic, if required.

Tag

119

Comments

1. When a **S_SAFINI** bit=0, the corresponding signal is not inverted and the high voltage state is considered active.
2. When a **S_SAFINI** bit=1, the bit is inverted and the low voltage state is considered active.

Accessibility

Read-Write



S_SAFINI values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Variables

[FAULT](#), [S_FAULT](#), [FDEF](#), [S_FDEF](#), [FMASK](#), [S_FMASK](#), [SAFIN](#), [S_SAFIN](#), [SAFINI](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, SetSafetyInputPortInv, GetSafetyInputPortInv

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_SetSafetyInputPortInv, acsc_GetSafetyInputPortInv

3.11.22 *SS11TIME*

Description

SS11TIME is a integer array with one element for each EtherCAT node in the system, the elements of which store the last SS1-t channel A time between the emergency stop request (24V switched off) and the point in time when the drive in fact switched the torque off mode (5V switched off). The value ranges between 0 and 500. The user can read this value in order to determine whether the system stops motion within the time required by the system functional safety requirements.



SS11TIME should be used for SS1-t diagnostics only

Tag

373

Comments

This variable is supported in version 3.00 and higher

Accessibility

Read-Only

Related ACSPL+ Variables

SS12TIME

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.11.23 *SS12TIME*

Description

SS12TIME is a integer array with one element for each EtherCAT node in the system, the elements of which store the last SS1-t channel B time between the emergency stop request (24V switched off) and the point in time when the drive in fact switched the torque off mode (5V switched off). The value ranges between 0 and 500. The user can read this value in order to determine whether the system stops motion within the time required by the system functional safety requirements.



SS12TIME should be used for SS1-t diagnostics only

Tag

374

Comments

This variable is supported in version 3.00 and higher

Accessibility

Read-Only

Related ACSPL+ Variables

SS11TIME

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.11.24 STODELAY

Description

STODELAY is a real array, with one element for each axis in the system. It is used to configure the delay time between the STO fault indication and the default response (disable) to the fault. During this time the user can activate his own response (auto-routine) to kill the motion.

Syntax

STODELAY(*axis*) = *value*

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis</i> | The specific axis index. Valid numbers are: 0,1... up to the number of axes in the system, minus 1. |
| <i>value</i> | The value is the delay time between the STO fault indication and the default response (disable) to the fault. value can range between 0 (minimum) to 200 (maximum). The default is 50. |

Tag

319

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.11.25 SYNC

Description

SYNC is an integer array (one element per each slave node) the elements of which contain a slave synchronization indicator for the node.

Tag

222

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.12 Induction Motor Variables

The Induction Motor variables are:

| Name | Description |
|--------------------------|-----------------------------|
| SLCFIELD | Induction Motor Excitation |
| SLCSLIP | Induction Motor Slip Factor |

3.12.1 SLCFIELD

Description

SLCFIELD is a real array with one element for each axis in the system. It is used along with [SLCSLIP](#) for controlling permanent magnet (PM) synchronous motors (so called "DC Brushless motors").

SLCFIELD defines the magnetic field component.

Syntax

SLCFIELD (*axis_index*)>= *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value provides the percentage of the peak current of the amplifier ranging from 0 to 25. |

Tag

217

Comments

Vector control, also known as Field Oriented Control, is a control technique that imitates the DC motor operation and applies it to AC motors: PM synchronous motors (DC brushless motors) and induction motors. This technique decomposes the motor current into two independent components:

- > The magnetizing (direct) component that influences the total magnetic flux.
- > The torque-producing (quadrature) component that influences the generated torque. This component is perpendicular to the magnetizing component.

In PM synchronous motors, the magnetizing component is in phase with the permanent magnet field. The goal is usually to keep this component zero, so all the current is dedicated to torque production. This maximizes the torque/current ratio and improves the efficiency and dynamic performance.

The magnetic field variable, **SLCFIELD**, is usually set equal to the nominal magnetizing current of the motor. It should be around the "knee" of the magnetizing curve of the motor - it should be high enough to maximize the torque constant of the motor, but must not be too high to prevent magnetic saturation. The exact value is provided by the motor manufacturer, but it can also be estimated based on 10-40% of the nominal current.

The value of **SLCFIELD** is expressed as percentage of the peak current of the amplifier and is calculated according to the following formula:

$$Field = \frac{\sqrt{2}I_{mag}}{I_{peak}} 100$$

where:

| | | |
|------------|----------------------------------------------------------|----------------------------------------------------------------|
| I_{mag} | The nominal excitation current of the motor (rms value). | Provided by the manufacturer, of 10-40% of the nominal current |
| I_{peak} | The amplitude of the maximum current of the amplifier. | |



SLCFIELD = 0 identifies an induction motor.

Accessibility

Read-Write



SLCFIELD values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

Related ACSPL+ Variables

SLCSLIP

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.12.2 SLCSLIP**Description**

SLCSLIP is a real array with one element for each axis in the system. It is used along with **SLCFIELD** for controlling permanent magnet (PM) synchronous motors (so called "DC Brushless motors").

SLCSLIP defines the slip constant.

Syntax

SLCSLIP(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value designates the slip frequency ranging from 0 to 5000. |

Tag

218

Comments

The range of **SLCSLIP** is typically 200-1000. A reference value is calculated as follows:

$$SLCSLIP = 131 \times f_n [Hz] \times s_n [\%] \times \frac{I_{peak}}{\sqrt{2}I_n}$$

where:

f_n - The nominal supply frequency

s_n - The nominal slip, given by:

where:

n_s - The synchronous velocity

n_n - The nominal velocity

I_{peak} - The peak current of the amplifier

I_n - The nominal motor current (rms)

For example, the nominal data of a 1.05kW motor is:

$n = 2870\text{rpm}$, $n = 3000\text{rpm}$, $I_n = 8.1\text{A}$, and $f_n = 50\text{Hz}$

So the nominal slip of the motor is:

$$s = \frac{3000-2870}{3000} = 0.043$$

For $I_{peak} = 10\text{A}$, the value of **SLCSLIP** will be:

$$SLCSLIP = 131 \times 50 \times 0.043 \frac{10}{\sqrt{2} \times 8.1} = 248$$

Accessibility

Read-Write



SLCSLIP values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection.

Related ACSPL+ Variables

[SLCFIELD](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13 Nanomotion Variables

The UDIhp-x / UDMnt-x (new revision) products' control algorithm supports Nanomotion motors, based on Nanomotion servo algorithm. Each UDIhp-x / UDMnt-x (new revision) products can support up to four Nanomotion axes using AB1 amplifier or two Nanomotion axes using AB2 amplifier (with automatic DC/AC mode switching).



AB2 amplifiers are not supported by SPiiPlus ADK Suite 2.40 or later. If AB2 amplifiers are used and there is not a need to upgrade the FW, it is recommended to continue using FW 2.30.03.

If an upgrade is needed, consult ACS Applications and the relevant DSP will be provided.

To activate the Nanomotion algorithm set the seventh bit of **MFLAGS** variable to 1

> **MFLAGS(<axis>).7 = 1**, or alternatively **MFLAGS(<axis>).#NANO = 1**

The following variables should be used in support of Nanomotion piezo-ceramic motor motion:

| Name | Description |
|----------|---------------------------------------------------------------------------------------------------------------|
| SLDZMIN | Parameter which specifies the minimum position of the Dead Zone (when the servo is turned off) |
| SLDZMAX | Parameter which specifies the maximum position of the Dead Zone (when the servo is turned on). |
| SLDZTIME | Parameter which specifies the duration (in msec) required for settling after entering the SLDZMIN. |
| SLZFF | Parameter which specifies the distance from target to stop the velocity Feed Forward. |
| SLFRC | Parameter which specifies the initial non-zero command to overcome the static friction in positive direction. |
| SLFRCN | Parameter which specifies the initial non-zero command to overcome the static friction in negative direction. |
| SLHRS | Parameter which specifies the modulation ratio of the drive command. |
| SLVKPDCF | Parameter which specifies the multiplication factor of the velocity loop gain (SLVKP) in DC mode. |
| SLPKPDCF | Parameter which specifies the multiplication factor of the position loop gain (SLPKP) in DC mode. |
| SLVKIDCF | Parameter which specifies the multiplication factor of the velocity loop integrator (SLVKI) in DC mode. |

3.13.1 SLDZMIN

Description

SLDZMIN is a real array, with one element for each axis in the system, and is used for defining the minimum position of the Dead Zone (when Servo is turned off).

Syntax

SLDZMIN(*axis_index*) = *value*

Arguments

| | |
|-------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges between 0 to 1.79769e+308, Default = 1. |

Tag

162

Comments

The Dead Zone mechanism stops the motor when the position approaches the target within the value of **SLDZMIN**. The value depends on the system specifications; usually **SLDZMIN** is between 1.0 to 2.0 counts (with an equivalent value in user units).

Accessibility

Read-Write



SLDZMIN values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.2 SLDZMAX

Description

SLDZMAX is a real array, with one element for each axis in the system, and is used for defining the maximum position of the Dead Zone.

Syntax

SLDZMIN(*axis_index*) = *value*

Arguments

| | |
|-------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges between 0 to 1.79769e+308, Default = 10. |

Tag

163

Comments

The Dead Zone mechanism starts the motor again when the error radius increases above the value **SLDZMAX**. The value depends on the system specifications; usually **SLDZMAX** is between 4.0 to 10.0 counts (with an equivalent value in user units).

Accessibility

Read-Write



SLDZMAX values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.3 SLDZTIME**Description**

SLDZTIME is a real array, with one element for each axis in the system, which defines the duration (in msec) required for settling after entering the **SLDZMIN**. Only after this duration the controller starts monitoring the position error and returns the servo if |PE| exceeds **SLDZMAX**.

Syntax**SLDZTIME**(*axis_index*) = *value***Arguments**

| | |
|--------------------------|-----------------------------------------------------------------------|
| <i>axis_index</i> | <i>axis_index</i> designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | <i>value</i> ranges between 0.1 to 1.79769e+308, Default = 20. |

Tag

251

Accessibility

Read-Write



SLDZMAX values cannot be modified if protection is applied to this variable through SPIiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.4 SLZFF**Description**

SLZFF is a real array, with one element for each axis in the system, and is used for defining the distance from target to stop the velocity Feed Forward.

Syntax**SLZFF**(*axis_index*) = *value*

Arguments

| | |
|--------------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges between 0 to 1.79769e+308, Default = 300. |

Tag

189

Comments

Using **SLZFF** improves settling time by stopping the feed forward velocity when the axis is getting close to the target position. The distance from the target is defined by **SLZFF** (in user units). The proper value of **SLZFF** depends on the total moving mass and the resolution of the encoder. Usually:

- > For an HR1 motor with encoder resolution of 0.1 μ M, set **SLZFF** to 100 - 300 counts (with an equivalent value in user units).
- > For an HR8 motor with encoder resolution of 0.1 μ M, set **SLZFF** to 300 - 400 counts (with an equivalent value in user units).

Accessibility

Read-Write



SLZFF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.5 SLFRC**Description**

SLFRC is a real array, with one element for each axis in the system, which defines initial non-zero command to overcome the static friction in a positive direction.

Syntax

$$\text{SLFRC}(\text{axis_index}) = \text{value}$$
Arguments

| | |
|--------------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges between 0 to 50, Default = 0. |

Tag

167

Comments

SLFRC is expressed as a percentage of the maximum output.

Accessibility

Read-Write



SLFRC values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.6 SLFRCN

Description

SLFRCN is a real array, with one element for each axis in the system, which defines initial non-zero command to overcome the static friction in a negative direction.

Syntax

SLFRCN(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------|
| <i>axis_index</i> | <i>axis_index</i> designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | <i>value</i> ranges between 0 to 50, Default = 0. |

Tag


250

Comments

SLFRCN is expressed as a percentage of the maximum output.

Accessibility

Read-Write



SLFRCN values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.7 SLHRS

Description:

SLHRS is a real array, with one element for each axis in the system, which defines the modulation ratio of the drive command.

Syntax:

SLHRS(*axis_index*) = *value*

Arguments

| | |
|-------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges from 0 to 100, Default = 100. |

Tag:

169

Accessibility

Read-Write



DCOM values cannot be modified if protection is applied to this variable through **SPiiPlus**
MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, Write Variable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.8 SLVKPDCF

Description

SLVKPDCF is a real array, with one element for each axis in the system, which defines multiplication factor of the velocity loop gain (**SLVKP**) in DC mode. The normal gain is increased by setting a **SLVKPDCF** value larger than 1.

Syntax

SLPKPDCF *axis_index* = *value*

Arguments

| | |
|-------------------|-------------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges between -1.79769e+308 to 1.79769e+308, Default = 1. |

Tag

252

Accessibility

Read-Write



SLPKPDCF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.9 SLPKPCDF**Description**

SLPKPDCF is a real array, with one element for each axis in the system, which defines multiplication factor of the position loop gain (**SLPKP**) in DC mode. The normal gain is increased by setting the **SLPKPDCF** to a value larger than 1.

Syntax**SLPKPDCF** *axis_index* = *value***Arguments**

| | |
|-------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | <i>axis_index</i> designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | <i>value</i> ranges between 0 to 1.79769e+308, Default = 1. |

Tag

253

Accessibility

Read-Write



SLPKPDCF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.13.10 SLVKIDCF

Description

SLVKIDCF is a real array, with one element for each axis in the system, which defines multiplication factor of the velocity loop integrator (**SLVKI**) in DC mode. The normal gain is increased by setting the **SLVKIDCF** to a value larger than 1.

Syntax

SLVKIDCF *axis_index* = *value*

Arguments

| | |
|-------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | value ranges between 0 to 1.79769e+308, Default = 1. |

Tag

254

Accessibility

Read-Write



SLVKIDCF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14 Servo-Loop Variables



Servo-Loop variables are fully accessible at the ACSPL+ level. While ACSPL+ programs generally do not refer to servo-loop variables at run time, an advanced program could change a servo-loop variable on-the-fly to provide adaptive control

The servo-loop variables are used for configuration and adjustment, and are set through **SPiiPlus MMI Application Studio** → **Setup** → **Adjuster**.

The Servo-Loop variable is:

| Name | Description |
|------|---------------|
| DCOM | Drive Command |

Additional servo-loop variables are grouped as follows:

- > [Servo-Loop Current Variables](#)
- > [Servo-Loop Velocity Variables](#)
- > [Servo-Loop Velocity Notch Filter Variables](#)
- > [Servo-Loop Velocity Low Pass Filter Variables](#)
- > [Servo-Loop Velocity Bi-Quad Filter Variables](#)
- > [Servo-Loop Position Variables](#)
- > [Servo-Loop Compensations Variables](#)
- > [Servo Loop Stepper Variables](#)
- > [Servo-Loop Miscellaneous Variables](#)
- > [Non-Linear Control Variables](#)

3.14.1 DCOM

Description

DCOM is a real array, with one element for each axis in the system. It is used for defining the commanded current as a percentage of the maximum drive command..

Syntax

DCOM(*axis_index*) = *value*

Arguments

| | |
|--------------------------|--------------------------------------------------------------------|
| <i>axis_index</i> | <i>axis_index</i> designates the specific axis: 0 - X, 1 - Y, etc. |
| <i>value</i> | <i>value</i> ranges from -100 to 100, Default = 0. |

Tag

20

Comments

DCOM defines a percentage of the maximum current command **XCURV**. When operating in the open loop mode (**MFLAGS.1=1**), **DCOM** supplies this current directly to the motor windings.

DCOM defines a percentage of the maximum drive command, for example, the UDI provides differential drive output from -10 to +10V. Therefore, assigning 100 to **DCOM** provides +10V on the drive output, -100 corresponds to -10V and 0 to 0V.

When operating in the closed loop mode (**MFLAGS.1=0**), **DCOM** offsets the normal commanded output current from the drive..

Accessibility

Read-Write



DCOM values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

Related ACSPL+ Commands[MFLAGS](#)**COM Library Methods and .NET Library Methods**

ReadVariable, Write Variable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2 Servo-Loop Current Variables

The Servo-Loop Current variables are:

| Name | Description |
|-------------------------|--------------------------------------------------------------|
| SLBIASA | Current phase A bias |
| SLBIASB | Current phase B bias |
| SLBIASC | Current phase C bias |
| SLIKI | Integrator gain |
| SLIKP | Integrator proportional gain |
| SLIFILT | Internal current filter |
| SLIOFFS | Offset to be added to the result of the current loop control |
| SLIUI | Used to limit the drive's output voltage |

3.14.2.1 SLBIASA**Description**

SLBIASA is a real array, with one element for each axis in the system, and is used for defining offset of the current in phase "S" or offset in command of phase "S".

Syntax

SLBIASA(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between -10 to 10, Default = 0. |

Tag

149

Comments

SLBIASA is expressed as a percentage of the maximum controller voltage output.

1. For **integrated models**: **SLBIASA** is read-only and displays the measured value of the current input bias.
2. For **non-integrated models**: **SLBIASA** is read-write and specifies the bias of the drive output. The controller uses the value only for brushless motors commutated by the controller.

Accessibility

Read-Only (integrated models)

Read-Write (nonintegrated models)



SLBIASA values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection Wizard**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.2 SLBIASB

Description

SLBIASB is a real array, with one element for each axis in the system, and is used for defining offset of the current in phase "T" or offset in command of phase "T".

Syntax

SLBIASB(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between -10 to 10, Default = 0. |

Tag

150

Comments

SLBIASB is expressed as a percentage.

1. For **integrated models**: **SLBIASB** is read-only and displays the measured value of the current input bias.
2. For **nonintegrated models**: **SLBIASB** is read-write and specifies the bias of the drive output. The controller uses the value only for brushless motors commutated by the controller.

Accessibility

Read-Only (integrated models)

Read-Write (nonintegrated models)



SLBIASB values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.3 SLBIASC**Description**

SLBIASC is a real array, with one element for each axis in the system, and is used for defining offset of the current in phase "R" or offset in command of phase "R".

Syntax

```
SLBIASC(axis_index)=value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges between -10 to 10, Default = 0. |

Tag

404

Comments

SLBIASC is expressed as a percentage of the maximum controller voltage output.

1. For integrated models: **SLBIASC** is read-only and displays the measured value of the current input bias.
2. For non-integrated models: **SLBIASC** is read-write and specifies the bias of the drive output. The controller uses the value only for brushless motors commutated by the controller.

Related ACSPL+ Variables**SLBIASA, SLBIASB****Accessibility**

Read-Only (integrated models)

Read-Write (nonintegrated models)



SLBIASC values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio -> Toolbox -> Application Development -> Protection Wizard

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.4 SLIKI

Description

SLIKI is a real array, with one element for each axis in the system, and is used for specifying the current loop.

Syntax

SLIKI(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0 - 65000. |

Tag

170

Comments

SLIKI is active only in integrated models.

Accessibility

Read-Write



SLIKI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.5 SLIKP

Description

SLIKP is a real array, with one element for each axis in the system, and is used for specifying the current loop proportional coefficient.

Syntax

SLIKP(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0 - 256000, Default = 1000. |

Tag

171

Comments

SLIKP is active only in integrated models.

Accessibility

Read-Write



SLIKP values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.6 SLIFILT

Description

SLIFILT is a real array, with one element for each axis in the system, and is used for defining the UDM current filter frequency.

Syntax

SLIFILT(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0-5000. |

Tag

226

Accessibility

Read-Write



SLIFILT values cannot be modified if protection is applied to this variable through
SPIiPlus MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Variables and .NET Library Methods

SLIKI, SLIKP

COM Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.7 SLIOFFS**Description**

SLIOFFS is a real array, with one element for each axis in the system, and is used for offset to be added to the result of the current loop control.

Syntax

$$\text{SLIOFFS}(\text{axis_index}) = \text{value}$$
Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between -50 to 50, Default = 0. |

Tag

172

Comments

The variable contains value in percents of maximal drive output.

The primary goal of the variable is to compensate for an active component of the motor load. For example, in a vertical axis the weight of the carriage can be compensated.

The variable is valid for DC brush and brushless motors.

Normally, the variable is changed in the process of adjustment (use **SPIiPlus MMI Application Studio → Toolbox → Setup → Adjuster Wizard**).

Accessibility

Read-Write



SLIOFFS values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.2.8 SLUI

Description

SLUI is a real array, with one element for each axis in the system, and is used to limit the drive's output voltage. If raised, a higher speed can be achieved for the given drive (higher output voltage).

Syntax

SLUI(*axis_index*) = *value*

Arguments

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value is a percentage of the maximal drive output, and consequently ranges between 0 to 100; Default: 88. |

Tag

221

Comments



It is recommended to set value **SLUI(axis)=97** to get a higher speed only for SPiiPlus CMnt, SPiiPlus UDMpm and SPiiPlus UDMpc

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3 Servo-Loop Velocity Variables

The Servo-Loop Velocity variables are:

| Name | Description |
|---------|---------------------------------------------------------------------------------------------|
| SLCRAT | Defines gear ratio between velocity feedback resolution and commutation feedback resolution |
| SLVKI | Sets velocity integrator coefficient |
| SLVKIIF | Provides an Idle Factor to the SLVKI variable |
| SLVKISF | Provides a Settle Factor to the SLVKI variable |
| SLVKP | Sets the proportional velocity gain. |
| SLVKPIF | Provides an Idle Factor to the SLVKP variable |
| SLVKPSF | Provides a Settle Factor to the SLVKP variable |
| SLVLI | Integrator velocity limit |
| SLVRAT | Velocity feed forward ratio |

3.14.3.1 SLCRAT

Description

SLCRAT is a real array, with one element for each axis in the system defining the ratio between the velocity feedback resolution and the commutation feedback resolution. It is used during the commutation phase.

Syntax

SLCRAT(*axis_index*)

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between -8.38861e+006 to 8.38861e+006, Default = 1. |

Tag

154

Comments

Velocity feed forward compensates for the velocity feedback, achieving zero position error at constant velocity.

Accessibility

Read-Write



SLCRAT values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.2 SLVKI

Description

SLVKI is a real array, with one element for each axis in the system, and is used for specifying the velocity loop integrator coefficient.

Syntax

SLVKI(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0 to 20000; Default = 200. |

Tag

179

Accessibility

Read-Write



SLVKI values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.3 SLVKIIF

Description

SLVKIIF is a real array with one element for each axis in the system. It is used for providing an Idle Factor to the **SLVKI** (Integrator Gain - Velocity) variable.

Syntax

SLVKIIF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 100.0. |

Tag

233

Accessibility

Read-Write



SLVKIIF values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.4 SLVKISF

Description

SLVKISF is a real array with one element for each axis in the system. It is used for providing a Settle Factor to the **SLVKI** variable.

Syntax

SLVKISF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 100.0. |

Tag

234

Accessibility

Read-Write



SLVKISF values cannot be modified if protection is applied to this variable through
 SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.5 SLVKITF

Description

SLVKITF increases the velocity loop integrator coefficient when the axis is close to the target position. It is a real array with one element for each axis in the system.

Syntax

SLVKITF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 1.0 and 100.0. Default = 1. |

Tag

271

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.6 SLVKP

Description

SLVKP is a real array, with one element for each axis in the system, and is used for specifying the velocity loop proportional coefficient.

Syntax

SLVKP(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0 to 16777215, Default = 100. |

Tag

180

Accessibility

Read-Write



SLVKP values cannot be modified if protection is applied to this variable through **SPIiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.7 SLVKPIF

Description

SLVKPIF is a real array with one element for each axis in the system. It is used for providing an Idle Factor to the SLVKP (Proportional Gain - Velocity) variable.

Syntax

SLVKPIF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 100.0. |

Tag

235

Accessibility

Read-Write



SLVKPIF values cannot be modified if protection is applied to this variable through **SPIiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.8 SLVKPSF

Description

SLVKPSF is a real array with one element for each axis in the system. It is used for providing a Settle Factor to the SLVKP variable.

Syntax

SLVKPSF *axis_index* = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | A real value ranging between 0.0 and 100.0. |

Tag

236

Accessibility

Read-Write



SLVKPSF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.9 SLVKPTF

Description

SLVKPTF increases the velocity loop proportional coefficient when the axis is close to the target position. It is a real array with one element for each axis in the system.



SLVKPTF is supported for the NanoPWM drives only.

Syntax

SLVKPTF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 1.0 and 100.0. Default = 1. |

Tag

272

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.10 *SLVLI*

Description

SLVLI is a real array, with one element for each axis in the system, and is used for providing an integrator limit for the velocity of the specified axis.

Syntax

SLVLI(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0 to 100, Default = 50. |

Tag

181

Comments

SLVLI is expressed as a percentage of the maximum value.

Accessibility

Read-Write



SLVLI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.3.11 SLVRAT

Description

SLVRAT is a real array, with one element for each axis in the system. It defines the velocity feed forward ratio during the commutation phase..

Syntax

SLVRAT(*axis_index*)

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between -8.38861e+006 to 8.38861e+006, Default = 1. |

Tag

187

Comments

Velocity feed forward compensates for the velocity feedback, achieving zero position error at constant velocity.

Accessibility

Read-Write



SLVRAT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.4 Servo-Loop Velocity Notch Filter Variables

Notch Filter variables serve for notching the velocity servo-loop. The Servo-Loop Velocity Notch Filter variables are:

| Name | Description |
|---------|---------------------------|
| SLVNFRQ | Notch filter frequency. |
| SLVNWID | Notch filter width. |
| SLVNATT | Notch filter attenuation. |

3.14.4.1 SLVNFRQ

Description

SLVNFRQ is a real array, with one element for each axis in the system, and is used for providing a notch filter frequency.

Syntax

SLVNFRQ(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0.1 to 4000, Default = 300. |

Tag

182

Comments

SLVNFRQ is expressed in Hz.

Accessibility

Read-Write



SLVNFRQ values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.4.2 *SLVNWID***Description**

SLVNWID is a real array, with one element for each axis in the system, and is used for providing a notch filter width.

Syntax

SLVNWID(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0.1 to 4000, Default = 30. |

Tag

183

Comments

SLVNWID is expressed in Hz.

Accessibility

Read-Write



SLVNWID values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.4.3 *SLVNATT***Description**

SLVNATT is a real array, with one element for each axis in the system, and is used for providing the attenuation of the notch frequency at frequency specified by **SLVNFRQ**.

Syntax

SLVNATT(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0.05 to 20; Default = 5. |

Tag

184

Accessibility

Read-Write



SLVNATT values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.5 Servo-Loop Velocity Low Pass Filter Variables

Low Pass Filter variables serve for setting the velocity low pass filtering parameters. The Servo-Loop Velocity Low Pass Filter variables are:

| Name | Description |
|---------|-----------------------|
| SLVSOF | Sets filter bandwidth |
| SLVSOFD | Sets filter damping |

3.14.5.1 SLVSOF**Description**

SLVSOF is a real array, with one element for each axis in the system, and is used for providing a second order filter bandwidth for the velocity of the specified axis.

Syntax

SLVSOF(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0.1 to 4000, Default = 700. |

Tag

185

Comments

SLVSOF is expressed in Hz.

Accessibility

Read-Write



SLVSOF values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.5.2 SLVSOFD**Description**

SLVSOFD is a real array, with one element for each axis in the system, and is used for providing a second order filter damping factor for the velocity of the specified axis.

Syntax

$$\text{SLVSOFD}(\text{axis_index}) = \text{value}$$
Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges between 0.3 to 1, Default = 0.707. |

Tag

186

Accessibility

Read-Write



SLVSOFD values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.6 Servo-Loop Velocity Bi-Quad Filter Variables

Bi-Quad Filter variables serve for setting the velocity bi-quad filtering parameters. The Servo-Loop Velocity Bi-Quad Filter variables are:

| Name | Description |
|---------|----------------------------------------------------------|
| SLVBODD | Sets the damping ratio denominator for a Bi-Quad filter. |
| SLVBODF | Sets the denominator value of the Bi-Quad filter. |
| SLVBOND | Sets the damping ratio numerator for a Bi-Quad filter. |
| SLVBONF | Sets the numerator value of the Bi-Quad filter. |

3.14.6.1 SLVBODD

Description

SLVBODD is a real array, with one element for each axis in the system, and is used for setting the damping ratio denominator for a Bi-Quad filter to the velocity loop control in addition to the existing 2nd order Low-pass and Notch filters for the given axis.

Syntax

SLVBODD(*axis_index*) = *value*

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates denominator value of the Bi-Quad damping ratio ranging from 0.1 to 1. See <i>SPIIPlus ACSPL+ Programmer Guide</i> . |

Tag

208

Comments

The Bi-Quad filter is the most general 2nd order filter. It has two poles and two zeros. It can be thought of as a high-pass filter in series with a low-pass filter.

Accessibility

Read-Write



SLVBODD values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.6.2 SLVBODF

Description

SLVBODF is a real array, with one element for each axis in the system, and is used for setting the denominator natural frequency value of the Bi-Quad filter algorithm applied to the velocity loop control in addition to the existing 2nd order Low-pass and Notch filters for the given axis.

Syntax

SLVBODF(*axis_index*) = *value*

Arguments

| | |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates denominator value of the Bi-Quad filter algorithm ranging from 0.1 to 4000 [Hz]. See <i>SPiiPlus ACSPL+ Programmer Guide</i> . |

Tag

209

Comments

The Bi-Quad filter is the most general 2nd order filter. It has two poles and two zeros. It can be thought of as a high-pass filter in series with a low-pass filter.

Accessibility

Read-Write



SLVBODF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.6.3 SLVBOND

Description

SLVBOND is a real array, with one element for each axis in the system, and is used for setting the damping ratio numerator for a Bi-Quad filter to the velocity loop control in addition to the existing 2nd order Low-pass and Notch filters for the given axis.

Syntax

SLVBOND(*axis_index*) = *value*

Arguments

| | |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates numerator value of the Bi-Quad damping ratio ranging from 0.1 to 1. See <i>SPiiPlus ACSPL+ Programmer Guide</i> . |

Tag

210

Comments

The Bi-Quad filter is the most general 2nd order filter. It has two poles and two zeros. It can be thought of as a high-pass filter in series with a low-pass filter.

Accessibility

Read-Write



SLVBOND values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.6.4 SLVBONF**Description**

SLVBONF is a real array, with one element for each axis in the system, and is used for setting the numerator natural frequency value of the Bi-Quad filter algorithm applied to the velocity loop control in addition to the existing 2nd order Low-pass and Notch filters for the given axis.

Syntax**SLVBONF(*axis_index*) = *value*****Arguments**

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates numerator value of the Bi-Quad filter algorithm ranging from 0.1 to 4000 [Hz]. See <i>SPiiPlus ACSPL+ Programmer Guide</i> . |

Tag

211

Comments

The Bi-Quad filter is the most general 2nd order filter. It has two poles and two zeros. It can be thought of as a high-pass filter in series with a low-pass filter.

Accessibility

Read-Write



SLVBONF values cannot be modified if protection is applied to this variable through
SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7 Servo-Loop Position Variables

The Servo-Loop Position variables are:

| Name | Description |
|-------------------------|-------------------------------------------------------------------------------|
| SLDRA | Defines disturbance rejection. |
| SLDRAIF | Provides an Idle Factor to the SLDRA. |
| SLDRX | Defines the maximum DRA correction for a given axis. |
| SLPKI | Specifies the position loop integrator coefficient |
| SLPKIIF | Provides the Idle Factor to the SLPKI (Integrator Gain - Position) variable |
| SLPKISF | Provides the Settle Factor to the SLPKI (Integrator Gain - Position) variable |
| SLPLI | Defines the limit for the position of the specified axis |
| SLPKP | Sets the proportional coefficient of the position for the specified axis. |
| SLPKPIF | Provides an Idle Factor to the SLPKP variable. |
| SLPKPSF | Provides an Settle Factor to the SLPKP variable. |

3.14.7.1 SLDRA

Description

SLDRA is a real array, with one element for each axis in the system, and is used for defining the DRA frequency for the given axis.

The ACS proprietary Disturbance Rejection Algorithm (DRA) is used to improve the disturbance rejection response of the servo, and helps to minimize the position error during the settling phase and shorten the settling time.

Syntax

SLDRA(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates the DRA frequency ranging from 0 to 1500 [Hz]. |

Tag

206

Comments

The most common use of DRA is to improve the settling of systems mounted on passive isolation platforms. Passive isolation is typically used to isolate systems from disturbances transmitted from the floor. They employ a seismic mass supported on a soft spring made of rubber, metal, or air. The spring's damping action absorbs vibrations above the spring's resonance. For this reason, passive isolation manufacturers usually try to lower spring resonant frequency to increase the effective isolation range. When a servo force is applied to generate motion, it also acts on the isolated stationary base, causing it to vibrate. Because the frequency is low (usually below 1 Hz, to 10 Hz) and damping is very light, the isolation system continues vibrating long after the motion profile has ended. This vibration acts as disturbance to the servo system, introduces position error, and extends the settling time.

The DRA is used to minimize the latter effect and improve the position error during settling.

Accessibility

Read-Write



SLDRA values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.2 SLDRAIF

Description

SLDRAIF is a real array with one element for each axis in the system. It is used for providing an Idle Factor to the **SLDRA** variable.

Syntax**SLDRAIF** *axis_index* = *value***Arguments**

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 100.0. |

Tag

230

Accessibility

Read-Write



SLDRAIF values cannot be modified if protection is applied to this variable through
SPIiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.3 SLDRX**Description**

SLDRX is a real array, with one element for each axis in the system, and is used for defining the maximum DRA correction for the given axis.

The ACS proprietary Disturbance Rejection Algorithm (DRA) is used to improve the disturbance rejection response of the servo, and helps to minimize the position error during the settling phase and shorten the settling time.

Syntax**SLDRX** (*axis_index*) = *value***Arguments**

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value designates the DRA correction ranging from 0 to 2^{23} . |

Tag

207

Comments

The most common use of DRA is to improve the settling of systems mounted on passive isolation platforms. Passive isolation is typically used to isolate systems from disturbances transmitted from the floor. They employ a seismic mass supported on a soft spring made of rubber, metal, or air. The spring's damping action absorbs vibrations above the spring's resonance. For this reason, passive isolation manufacturers usually try to lower spring resonant frequency to increase the effective isolation range. When a servo force is applied to generate motion, it also acts on the isolated stationary base, causing it to vibrate. Because the frequency is low (usually below 1 Hz, to 10 Hz) and damping is very light, the isolation system continues vibrating long after the motion profile has ended. This vibration acts as disturbance to the servo system, introduces position error, and extends the settling time.

The DRA is used to minimize the latter effect and improve the position error during settling.

Accessibility

Read-Write



SLDRX values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.4 SLPKI

Description

SLPKI is a real array, with one element for each axis in the system, and is used for specifying the position loop integrator coefficient.

Syntax

SLPKI(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0 to 20000; Default = 0. |

Tag

174

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.5 SLPKIIF

Description

SLPKIIF is a real array with one element for each axis in the system. It is used for providing an Idle Factor to the [SLPKI](#) (Integrator Gain - Position) variable.

Syntax

SLPKIIF(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 to 100.0; Default = 1. |

Tag

260

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

acsc_ReadReal, acsc_WriteReal

3.14.7.6 SLPKISF

Description

SLPKISF is a real array with one element for each axis in the system. It is used for providing a Settle Factor to the [SLPKI](#) (Integrator Gain - Position) variable.

Syntax

SLPKISF(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 to 100.0; Default = 1. |

Tag

261

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.7 *SLPKITF*

Description

SLPKITF increases the velocity loop proportional coefficient when the axis is close to the target position. It is a real array with one element for each axis in the system.



SLPKITF is supported for the NanoPWM drives only.

Syntax

SLPKITF(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 to 100.0; Default = 1. |

Tag

269

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.8 *SLPLI*

Description

SLPLI is a real array, with one element for each axis in the system, and is used for providing an integrator limit for the position of the specified axis.

Syntax

SLPLI(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0 to 100; Default = 0. |

Tag

176

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.9 SLPKP

Description

SLPKP is a real array, with one element for each axis in the system, and is used for setting the proportional coefficient of the position for the specified axis.

Syntax

SLPKP(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0 to 16777215, Default = 0. |

Tag

175

Comments

Motor movement during commutation largely depends on the servo-loop parameters.

COMMUT will not operate properly if **SLPKP** is set to zero, or the integrator is very low.

Accessibility

Read-Write



SLPKP values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

Related ACSPL+ Commands

[COMMUT](#)

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.10 SLPKPIF

Description

SLPKPIF is a real array with one element for each axis in the system. It is used for providing an Idle Factor to the [SLPKP](#) variable.

Syntax

SLPKPIF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 100.0. |

Tag

231

Accessibility

Read-Write



SLPKPIF values cannot be modified if protection is applied to this variable through
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.11 SLPKPSF

Description

SLPKPSF is a real array with one element for each axis in the system. It is used for providing a Settle Factor to the **SLPKP** variable.

Syntax

SLPKPSF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 100.0. |

Tag

232

Accessibility

Read-Write



SLPKPSF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.7.12 SLPKPTF

Description

SLPKPTF increases the position loop proportional coefficient when the axis is close to the target position. It is a real array with one element for each axis in the system.



SLPKPTF is supported for the NanoPWM drives only.

Syntax

SLPKPTF *axis_index* = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 1.0 and 100.0. Default = 1. |

Tag

270

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.8 Servo-Loop Compensations Variables

Servo-Loop Compensations variables are used to set various parameters that provide compensation to overcome certain motion problems. The Servo-Loop Compensation variables are:

| Name | Description |
|--------|----------------------------------------------------|
| SLAFF | Defines acceleration feed forward for a given axis |
| SLFRCD | Compensation for dynamic friction |

3.14.8.1 SLAFF

Description

SLAFF is a real array, with one element for each axis in the system, and is used for specifying the acceleration feed forward of the specified axis.

Syntax

SLAFF(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 0 to 16777215; Default: 0. |

Tag

148

Accessibility

Read-Write



SLAFF values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.8.2 SLFRCD

Description

SLFRCD is a real array, with one element for each axis in the system, and is used for providing dynamic friction compensation at the maximum velocity.

Syntax

SLFRCD(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges between 0% to 5% of maximum command. |

Tag

168

Comments

SLFRCD provides dynamic compensation at the maximum velocity **XVEL**. For lower velocities, the compensation is reduced proportionally with the velocity. The value of **SLFRCD** is given as a percentage (range is 0 to 50%) of the maximum command.

Accessibility

Read-Write



SLFRCD values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.9 Servo Loop Stepper Variables

This section contains a collection of variables employed for calculation of the position correction for stepper motors.

| Name | Description |
|-------------------------|-----------------------------------------------------------|
| MFLAGSX | Extended Motor Flags |
| SLSMZ | Array of dead zone values for position correction |
| SLSKI | Array of integral gain values for position correction |
| SLSKP | Array of proportional gain values for position correction |
| SLSMC | Array of maximum allowed stepper correction values |
| SLSOUT | Returns calculated stepper correction for given axis |
| SLSRL | Array of rate limiter values for position correction |

3.14.9.1 MFLAGSX

Description

MFLAGSX is an integer array with one element for each axis in the system, each element of which contains a set of bits used for configuring the motor. It is an extension of the **MFLAGS** variable.

Syntax

```
MFLAGSX(Axis_Index).bit_designator = 0|1
```

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| Bit_designator | An MFLAGSX bit designator as described below |

MFLAGSX Bit Designators

| Bit Name | No. | Description |
|------------------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #STCLFULL | 0 | 0 (default): Do not use position correction for Stepper motors working under closed loop. 1: Activate Full mode of the stepper closed loop position correction mechanism . This bit is mutually exclusive to #STCLEND and #STCLSP , as such only one of them may be 1 at a time. |
| #STCLEND | 1 | 0 (default): Do not use position correction for Stepper motors working under closed loop 1: Activate End mode of the stepper closed loop position correction mechanism This bit is mutually exclusive to #STCLFULL and #STCLSP , as such only one of them may be 1 at a time. |
| #STCLSP | 2 | 0 (default): Do not use servo processor closed-loop stepper algorithm 1: Use servo processor closed-loop stepper algorithm Mutually exclusive to #STCLFULL , #STCLEND . |
| #VOLTMODE | 3 | 0: Use current mode for control 1: Use voltage mode instead of current mode to compensate for low resolution |
| #HLIMSWAP | 4 | 0: Left/Right limit signal swapping is disabled 1: Left/Right limit signal swapping is enabled |
| #SATPROT | 5 | 1: Saturation Protection enabled (default) 0: Saturation Protection disabled |

Tag

357

Comments

When saturation protection (**MFLAGSX.#SATPROT=1**) is currently available for the following products: NPMpm, UDMxx, IDMxx.

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKI, SLSKP, SLSDZ

Accessibility

Read-Write



MFLAGSX values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio→Toolbox→Application Development→ Protection

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.14.9.2 SLSDZ

Description

SLSDZ is a real array with one element for each axis in the system, it is used for the closed loop operation of steppers and represents the dead zone of the position correction.

Syntax

```
SLSDZ (Axis_Index) = Value
```

Arguments

| Axis_Index | Description |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |

Tag

356

Comments

Dead zone [user units]. Inside this zone ($|PE| < \mathbf{SLSDZ}$) algorithm is inactive.

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKI, SLSKP, SLSRL, MFLAGSX

Accessibility

Read-Write



SLSDZ values cannot be modified if protection is applied to this variable through SPiiPlus MMI Application Studio→Toolbox→Application Development→ Protection

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.14.9.3 SLSKI

Description

SLSKI is a real array with one element for each axis in the system, it is used for the closed loop operation of steppers and represents the integral gain[rad/sec] of the position correction.

Syntax

```
SLSKI (Axis_Index) = Value
```

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------|
| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|

Tag

353

Comments

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKP, SLSRL, SLSDZ

Accessibility

Read-Write



SLSKI values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio**→**Toolbox**→**Application Development**→**Protection**

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.14.9.4 SLSKP

Description

SLSKP is a real array with one element for each axis in the system, it is used for the closed loop operation of steppers and represents proportional gain of the position correction.

Syntax

```
SLSKP (Axis_Index) = Value
```

ArgumentsAxis_
Index

Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

Tag

354

Comments

Proportional gain is unitless.

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKI, SLSRL, SLSDZ, MFLAGSX

Accessibility

Read-Write



SLSKP values cannot be modified if protection is applied to this variable **through SPiiPlus MMI Application Studio→Toolbox→Application Development→Protection**

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.14.9.5 SLSMC**Description**

SLSMC is a real array with one element for each axis in the system, it is used for the closed loop operation of steppers and represents the maximum allowed stepper correction.

Syntax

```
SLSMC(Axis_Index) = Value
```

ArgumentsAxis_
Index

Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

Tag

360

Comments

Maximum correction value [user units]. If this value is reached FAULT.#PE is set.

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKI, SLSKP, SLSRL, SLSOUT, MFLAGSX

Accessibility

Read-Write



If the accumulated correction of a closed-loop algorithm reaches the maximum limit as defined by **SLSMC**, a non-critical Position Fault error is raised (error 5022).

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.14.9.6 SLSOUT**Description**

SLSOUT is a real array with one element for each axis in the system, it is used for the closed loop operation of steppers and represents the calculated stepper correction

Syntax

```
SLSOUT (Axis_Index)
```

Arguments

| Axis_Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------------|---------------------------------------------------------------------------------------------------------------|
| | |

Tag

359

Comments

Output of the PI loop [user units].

This variable is supported in ADK versions 2.70 and higher.

SLSOUT is reset upon disabling of an axis, or upon setting **RPOS** using the set command.**Related ACSPL+ Variables**

SLSKI, SLSKP, SLSRL, MFLAGSX

Accessibility

Read only

.NET Library Method

ReadVariable()

C Library Function

acsc_ReadInteger()

3.14.9.7 SLSRL

Description

SLSRL is a real array with one element for each axis in the system, it is used for the closed loop operation of steppers and represents the rate limiter of the position correction.

Syntax

```
SLSRL(Axis_Index) = Value
```

Arguments

| | |
|-------------|---------------------------------------------------------------------------------------------------------------|
| Axis_ Index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|-------------|---------------------------------------------------------------------------------------------------------------|

Tag

355

Comments

Rate limiter of the PI output [user unit/sec].

This variable is supported in ADK versions 2.70 and higher.

Related ACSPL+ Variables

SLSKI, SLSKP, SLSDZ

Accessibility

Read-Write



SLSRL values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio**→**Toolbox**→**Application Development**→**Protection**

.NET Library Method

ReadVariable(), WriteVariable()

C Library Function

acsc_ReadInteger(), acsc_WriteInteger()

3.14.10 Servo-Loop Miscellaneous Variables

This section contains a collection of miscellaneous variables employed for specific servo-loop purposes. The Servo-Loop Miscellaneous variables are:

| Name | Description |
|--------------------------|--------------------------------------------------------------------------|
| SLCROUT | Commutation feedback routing , see Commutation Variables |
| SLGCAXN | Specifies the complementary gantry axis |
| SLPROUT | Position feedback routing |
| SLP2ROUT | Sets the feedback routing of the secondary feedback position |
| SLVROUT | Velocity feedback routing |

3.14.10.1 SLCROUT

Description

SLCROUT is an integer array, with one element for each axis in the system, and is used for setting the feedback routing of the velocity commutation for the specified axis.

Syntax

SLCROUT(*axis_index*) = *value*

Arguments

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | The value values and the feedback sources associated with them are given in Table 3-19 . Default = 0. |

Table 3-19. SLCROUT Values

| SLCROUT | FACC (SPonly) |
|---------|-----------------------------------------------------------------|
| 0 | According to E_TYPE velocity |
| 001 | From channel 0 quadrature velocity or Absolute Encoder velocity |
| 002 | From channel 0 SINCOS velocity |
| 003 | From channel 0 HSSI velocity |
| 004 | From analog input 0 |
| 005 | From channel 0 resolver velocity |
| 101 | From channel 1 quadrature velocity Absolute Encoder velocity |
| 102 | From channel 1 SINCOS velocity |

| SLCROUT | FACC (SPonly) |
|---------|--------------------------------------------------------------|
| 103 | From channel 1 HSSI velocity |
| 104 | From analog input 1 |
| 105 | From channel 1 resolver velocity |
| 201 | From channel 2 quadrature velocity Absolute Encoder velocity |
| 202 | From channel 2 SINCOS velocity |
| 203 | From channel 2 HSSI velocity |
| 204 | From analog input 2 |
| 205 | From channel 2 resolver velocity |
| 301 | From channel 3 quadrature velocity Absolute Encoder velocity |
| 302 | From channel 3 SINCOS velocity |
| 303 | From channel 3 HSSI velocity |
| 304 | From analog input 3 |
| 305 | From channel 3 resolver velocity |

Tag

159

Accessibility

Read-Write



SLCROUT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.14.10.2 SLGCAXN

Description

SLGCAXN is a read-only integer array, with one element for each axis in the system, which specifies the complementary gantry axis. The value can be viewed SPiiPlus MMI Application Studio

Communication Terminal window or its value can be assigned to another variable, for example:

Var = **SLGCAXN**(axis_index).



In order to change gantry axis allocation, the **SETCONF**(267) command should be used.

Syntax

SLGCAXN(axis_index)

Arguments

| | |
|-------------------|-------------------------------------------------------------|
| axis_index | axis_index designates the specific axis: 0 - X, 1 - Y, etc. |
|-------------------|-------------------------------------------------------------|

Tag

256

Accessibility

Read Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.14.10.3 *SLPROUT*

Description

SLPROUT is a real array, with one element for each axis in the system, and is used for setting the feedback routing of the position for the specified axis.

Syntax

SLPROUT(axis_index) = value

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | The value values and the feedback sources associated with them are given in Table 3-20 . Default = 0. |

Table 3-20. SLPROUT Values

| SLPROUT | FPOS |
|---------|--------------------------------------------------------------|
| 0 | According to E_TYPE position |
| 001 | From channel 0 quadrature position Absolute Encoder position |
| 002 | From channel 0 SINCOS position |
| 003 | From channel 0 HSSI position |
| 004 | From analog input 0 |
| 005 | From channel 0 resolver position |
| 101 | From channel 1 quadrature position Absolute Encoder position |
| 102 | From channel 1 SINCOS position |
| 103 | From channel 1 HSSI position |
| 104 | From analog input 1 |
| 105 | From channel 1 resolver position |
| 201 | From channel 2 quadrature position Absolute Encoder position |
| 202 | From channel 2 SINCOS position |
| 203 | From channel 2 HSSI position |
| 204 | From analog input 2 |
| 205 | From channel 2 resolver position |
| 301 | From channel 3 quadrature position Absolute Encoder position |
| 302 | From channel 3 SINCOS position |
| 303 | From channel 3 HSSI position |
| 304 | From analog input 3 |
| 305 | From channel 3 resolver position |

Tag

177

Comments

The controller supports a standard control loop configuration where 0 feedback position (**FPOS**) is obtained from the 0 encoder, FPOS(1) from the 1 encoder, etc.

SLPROUT¹ 0 indicates FPOS is from an alternative sensor, for example, if **SLPROUT(0)** is 0104, FPOS is obtained from an analog input 0 rather than from the encoder. In this case, the feedback source could be a potentiometer or any other device that produces analog voltage proportional to the motor position.

The meaning of the routing value depends on the axis and the controller model. For example, a value of 1 specified for the 0 or 2 axis selects the 0 encoder, the same value for the 1 or 2 axis selects the 1 encoder.

Accessibility

Read-Write



SLPROUT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.10.4 SLP2ROUT

SPL2ROUT is variable for setting the feedback routing of the secondary feedback position.

Description

SPL2ROUT is an integer array, one element for each axis in the system, and is used for setting the feedback routing of the secondary feedback position for the specified axis.

Syntax

SPL2ROUT(<axis>)=value

Arguments

The value values and the feedback sources associated with them are given below. The default value is 0.

| Value | F2POS |
|-------|-----------------------------------------------------------------|
| 0 | Secondary Feedback is not used |
| 001 | From channel 0 quadrature position or Absolute Encoder position |
| 002 | From channel 0 SINCOS position |
| 003 | From channel 0 HSSI position101 |

| Value | F2POS |
|-------|-----------------------------------------------------------------|
| 004 | From analog input 0 |
| 005 | From channel 0 resolver position |
| 101 | From channel 1 quadrature position or Absolute Encoder position |
| 102 | From channel 1 SINCOS position |
| 103 | From channel 1 HSSI position |
| 104 | From analog input 1 |
| 105 | From channel 1 resolver position |
| 201 | From channel 2 quadrature position or Absolute Encoder position |
| 202 | From channel 2 SINCOS position |
| 203 | From channel 2 HSSI position |
| 204 | From analog input 2 |
| 205 | From channel 2 resolver position |
| 301 | From channel 3 quadrature position or Absolute Encoder position |
| 302 | From channel 3 SINCOS position |
| 303 | From channel 3 HSSI position |
| 304 | From analog input 3 |
| 305 | From channel 3 resolver position |

Comments

SLP2ROUT variable can be used for routing when applied on secondary feedback only.

Tag

301

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.14.10.5 SLTFWID

Description

SLTFWID determines the distance to the target at which the position and velocity loops gains will be increased by 50%. It is a real array with one element for each axis in the system.



SLTFWID is supported for the NanoPWM drives only.

Syntax

SLTFWID(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | A real value ranging between 0.0 and 4294967295. Default = 0. |

Tag

273

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.10.6 SLVROUT

Description

SLVROUT is a real array, with one element for each axis in the system, and is used for setting the feedback routing of the velocity for the specified axis.

Syntax

SLVROUT(*axis_index*) = *value*

Arguments

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | The value values and the feedback sources associated with them are given in Table 3-21 . Default = 0. |

Table 3-21. SLVROUT Values

| SLVROUT | FVEL (SP only) |
|---------|--------------------------------------------------------------|
| 0 | According to E_TYPE velocity |
| 001 | From channel 0 quadrature velocity Absolute Encoder velocity |
| 002 | From channel 0 SINCOS velocity |
| 003 | From channel 0 HSSI velocity |
| 004 | From analog input 0 |
| 005 | From channel 0 resolver velocity |
| 101 | From channel 1 quadrature velocity Absolute Encoder velocity |
| 102 | From channel 1 SINCOS velocity |
| 103 | From channel 1 HSSI velocity |
| 104 | From analog input 1 |
| 105 | From channel 1 resolver velocity |
| 201 | From channel 2 quadrature velocity Absolute Encoder velocity |
| 202 | From channel 2 SINCOS velocity |
| 203 | From channel 2 HSSI velocity |
| 204 | From analog input 2 |
| 205 | From channel 2 resolver velocity |
| 301 | From channel 3 quadrature velocity Absolute Encoder velocity |
| 302 | From channel 3 SINCOS velocity |
| 303 | From channel 3 HSSI velocity |
| 304 | From analog input 3 |
| 305 | From channel 3 resolver velocity |

Tag

188

Accessibility

Read-Write



SLVROUT values cannot be modified if protection is applied to this variable through
SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.14.11 Non-Linear Control Variables

See *Non-Linear Control Application Note* for more information about use of non-linear control variables.

The following variables are used to implement non-linear control.

| Name | Description |
|-------|--------------------------------------------------------------------------------------|
| SLPAP | Holds the exponent (α) of the Position-Loop Proportional Non-Linear Control |
| SLPDP | The Linear Range (δ) of the Position-Loop Proportional Non-Linear Control |
| SLPAI | The exponent (α) of the Position-Loop Integral Non-Linear Control |
| SLPDI | The Linear Range (δ) of the Position-Loop Integral Non-Linear Control |
| SLVAP | The exponent (α) of the Velocity-Loop Proportional Non-Linear Control |
| SLVDP | The Linear Range (δ) of the Velocity-Loop Proportional Non-Linear Control |
| SLVAI | The exponent (α) of the Velocity-Loop Integral Non-Linear Control |
| SLVDI | he Linear Range (δ) of the Velocity-Loop Integral Non-Linear Control |

3.14.11.1 SLPAP

Description

SLPAP is a real array, the size of which is determined by the total number of axes in the system.
SLPAP holds the exponent (α) of the Position-Loop Proportional Non-Linear Control.

Syntax

SLPAP(index) = *value*

Comments

This variable is supported in version 3.10 and higher.

3.14.11.2 SLPDP

Description

SLPDP is a real array, the size of which is determined by the total number of axes in the system. This is the Linear Range (δ) of the Position-Loop Proportional Non-Linear Control.

Syntax

SLPDP(index) = value

Arguments

| | |
|-------|------------------------------------------------------|
| index | Axis index, from 0 to number of axes in the system-1 |
| value | Default value is 1. The range is [0,1] |

Tag

Comments

A value of 1 sets the Linear range of the Non-Linear Gain curve to the Position-Error Limit.

If the value is not default and the required license for this feature is missing, error 3300 "Non Linear Control License is required" is given.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

SLPAP, SLPAI, SLPDI, SLVAP, SLVDP, SLVAI, SLVDI

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.14.11.3 SLPAI

Description

SLPAI is a real array, the size of which is determined by the total number of axes in the system. This is the exponent (α) of the Position-Loop Integral Non-Linear Control.

Syntax

SLPAI(index) = value

Arguments

| | |
|-------|---------------------------------------------------------------------|
| index | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 |
| value | Default value is 1. The range is [0,100] |

Tag**Comments**

A value of 1 sets the Linear Range of the Non-Linear Gain curve to the Position-Error Limit.

If the value is not default and the required license for this feature is missing, error 3300 "Non Linear Control License is required" is given.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

SLPAP, SLPDP, SLPAL, SLVAP, SLVDP, SLVAI, SLVDI

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.14.11.4 SLPDI**Description**

SLPDI is a real array, the size of which is determined by the total number of axes in the system. This is the Linear Range (δ) of the Position-Loop Integral Non-Linear Control.

Syntax

SLPDI(index) = value

Arguments

| | |
|-------|---------------------------------------------------------------------|
| index | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 |
| value | Default value is 1. The range is [0,1] |

Tag**Comments**

A value of 1 sets the Linear Range of the Non-Linear Gain curve to the Position-Error Limit.

If the value is not default and the required license for this feature is missing, error 3300 "Non Linear Control License is required" is given.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

SLPAP, SLPDP, SLPAL, SLVAP, SLVDP, SLVAI, SLVDI

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.14.11.5 SLVAP**Description**

SLVAP is a real array, the size of which is determined by the total number of axes in the system. This is the exponent (α) of the Velocity-Loop Proportional Non-Linear Control.

Syntax**SLVAP**(index) = value**Arguments**

| | |
|-------|---------------------------------------------------------------------|
| index | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 |
| value | Default value is 1. The range is [0,1] |

Tag**Comments**

A value of 1 sets the Velocity-Loop Proportional Gain to Linear.

If the value is not default and the required license for this feature is missing, error 3300 "Non Linear Control License is required" is given.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables**SLPAP, SLPDP, SLPAI, SLPDI, SLVDP, SLVAI, SLVDI****Accessibility**

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.14.11.6 SLVDP**Description**

SLVDP is a real array, the size of which is determined by the total number of axes in the system. This is the Linear Range (δ) of the Velocity-Loop Proportional Non-Linear Control.

Syntax**SLVDP**(index) = value

Arguments

| | |
|-------|---------------------------------------------------------------------|
| index | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 |
| value | |

Switches**Return Value**

None

Comments

This variable is supported in version 3.10 and higher.

Example

3.14.11.7 *SLVAI*

Description

SLVAI is a real array, the size of which is determined by the total number of axes in the system. This is the exponent (α) of the Velocity-Loop Integral Non-Linear Control.

Syntax

SLVAI(index) = value

Arguments

| | |
|-------|---------------------------------------------------------------------|
| index | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 |
| value | Default value is 1. The range is [0,1] |

Tag**Comments**

A value of 1 sets the Velocity-Loop Integral Gain to Linear.

If the value is not default and the required license for this feature is missing, error 3300 "Non Linear Control License is required" is given.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

SLPAP, SLPDP, SLPAI, SLPDI, SLVDP, SLVAI, SLVDI

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.14.11.8 SLVDI

Description

SLVDI is a real array, the size of which is determined by the total number of axes in the system. This is the Linear Range (δ) of the Velocity-Loop Integral Non-Linear Control.

Syntax

```
SLVDI(index) = value
```

Arguments

| | |
|-------|---------------------------------------------------------------------|
| index | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 |
| value | Default value is 1. The range is [0,1] |

Tag

Comments

A value of 1 sets the Linear Range of the Non-Linear Gain curve to the Maximum Current Command.

If the value is not default and the required license for this feature is missing, error 3300 "Non Linear Control License is required" is given.

This variable is supported in version 3.10 and higher.

Related ACSPL+ Variables

SLPAP, SLPDP, SLPAI, SLPDI, SLVDP, SLVAI, SLVDI

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable(), WriteVariable()

C Library Functions

acsc_ReadReal(), acsc_WriteReal()

3.15 Commutation Variables

The Servo-Loop Commutation variables are:

| Name | Description |
|-------------------------|--------------------|
| SLCHALL | Hall Shift |
| SLCNP | Number of Poles |
| SLCOFFS | Commutation Offset |
| SLCORG | Commutation Origin |

| Name | Description |
|--------------------------|----------------------------|
| SLCPRD | Commutation Period |
| SLHROUT | Setting Hall State Routing |
| SLSTHALL | Getting Hall State |



The low-level variables in this section are normally not used by the user. Generally, these variables are defined during the axis adjustment using **SPiiPlus MMI Application Studio** → **Toolbox** → **Setup** → **Adjuster** or the [COMMUT](#) command.

3.15.1 SLCHALL

Description

SLCHALL is an integer array, with one element for each axis in the system, and serves for storing the Hall shift.

Tag

192

Comments

The **Adjuster** commutation program calculates this parameter and saves it.



Do not change this parameter manually.

Accessibility

Read-Write



SLCHALL values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.15.2 SLCNP

Description

SLCNP is an integer array, with one element for each axis in the system, and defines the number of poles for a rotary motor.

Syntax

SLCNP(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 2 to 1000, Default = 4. |

Tag

152

Comments

For linear motors, set **SLCNP**=2.

Accessibility

Read-Write



SLCNP values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.15.3 SLCOFFS

Description

SLCOFFS is a real array, with one element for each axis in the system, and defines a commutation offset in electrical degrees to be added to the commutation phase.

Syntax

SLCOFFS(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from -60 to 60, Default=0. |

Tag

153

Comments

SLCOFFS defines **SLCOFFS** is valid only if a brushless motor is specified (**MFLAGS**(axis_index).#**BRUSHL** = 1).

Assignment to **SLCOFFS** immediately changes the commutation phase. Use **SLCOFFS** to introduce a small correction to the commutation phase.

Accessibility

Read-Write



SLCOFFS values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.15.4 SLCORG

Description

SLCORG is a real array, with one element for each axis in the system, that defines the commutation phase in electrical degrees at the point of origin which is usually at the index point.

Syntax

SLCORG(axis_index) = value

Arguments

| | |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| axis_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| value | value ranges from: <ul style="list-style-type: none"> > -60 to 60, Default=0 > 0 to 360, Default=0 |

Tag

156

Comments

SLCORG is valid only if a brushless motor has been specified, i.e., **MFLAGS**(axis_index).#**BRUSHL** = 1.

Accessibility

Read-Write



SLCORG values cannot be modified if protection is applied to this variable through
SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.15.5 SLCPRD

Description

SLCPRD is a real array, with one element for each axis in the system, that defines the servo-loop commutation period.

Syntax

SLCPRD(*axis_index*) = *value*

Arguments

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>value</i> | value ranges from 256 to 16777215, Default=8000. |

Tag

158

Comments

SLCPRD defines the feedback counts per revolution for rotary motors and the feedback counts per two magnetic pitches for linear motors.

Accessibility

Read-Write



SLCPRD values cannot be modified if protection is applied to this variable through
SPIIPlus MMI Application Studio → Toolbox → Application Development → Protection

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.15.6 SLHROUT

Hall Routing is available using ACSPL+ variable: **SLHROUT**

Description

SLHROUT is an integer array, with one element for each axis in the system, and is used for setting the Hall state routing for the specified axis.

Syntax

SLHROUT(<axis>)=value

| Value | SLSTHALL |
|-------|----------------|
| 0 | Default |
| 001 | From channel 0 |
| 101 | From channel 1 |
| 201 | From channel 2 |
| 301 | From channel 3 |

Comments

Hall state is an integer number within the range [0,5]. `Getconf(262,<index>)` returns Hall state of axis with index <index>. If `SLHROUT(<index>)` is not 0, `Getconf(262,<index>)` returns the hall state of the channel based on the table defined above.

Tag

302

Accessibility

Read-Write

Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.15.7 SLSTHALL**Description**

SLSTHALL is an integer array, with one element for each axis in the system, and is used for getting the Hall state of each axis.

The value is an integer number, in range of {-1,5}. -1 means invalid Hall state.

Comments

The **SLSTHALL** variable's value is being updated every cycle.

Tag

196

Accessibility

Read only

Com Library Methods and .NET Library Methods

ReadVariable

3.16 System Configuration Variables

The System Configuration variables are:

| Name | Description |
|-------------------|-----------------------------------------------------------------------------------------------------------|
| CFG | Configuration Mode |
| CTIME | Controller Cycle Time |
| EXTFAC | Conversion factor between the SL2-100 protocol transferred units (microns [μm]) and ACS user units. |
| FOLLOWCH | Maps an axis to a data channel of a SLEC module |
| G_01WCS...G_12WCS | Defines one of the 12 work-piece coordinate systems a user can set as part of the CNC setup/configuration |
| GPEXL | Indicates the GSP program executed block |
| GSPEXL | Indicates the line in a running buffer that called a subroutine |
| GUFAC | Holds the value a conversion factor, from 'Common Physical Units' in [mm] to 'Controller Units' |
| IENA | Interrupt Enable/Disable |
| IMASK | Interrupt Mask |
| ISENA | Interrupt-Specific Enable/Disable |
| S_FLAGS | System Flags |

| Name | Description |
|--------------------------|-------------------------------|
| S_SETUP | System Settings |
| XSEGAMAX | Maximal processing angles |
| XSEGAMIN | Minimal processing angles |
| XSEGRMAX | Maximal arc radius difference |
| XSEGRMIN | Minimal arc radius |

3.16.1 CFG

Description

CFG is an integer variable that indicates the application protection configuration mode.

Syntax

CFG = *value*

Arguments

| | |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>value</i> | <p>value can be one of the following:</p> <ul style="list-style-type: none"> 0: Controller is in protected mode 1: Controller is in normal mode |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tag

14

Comments

An attempt to assign a value to a protected variable when **CFG** = 0 causes an error.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Commands

acsc_ReadInteger

3.16.2 CTIME

CTIME is a real variable that defines the controller cycle time.

Syntax

CTIME = *value*

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------|
| <i>value</i> | value can be 0.2, 0.25, 0.5, or 1.0 milliseconds (depending on the controller model). |
|--------------|----------------------------------------------------------------------------------------------|

Tag

17

Comments

Many operations in the controller are synchronized to the controller cycle. For example, profile generation is executed each controller cycle.



If **CTIME** is used, before running, the program has to be saved to the controller and controller restarted.

CTIME is normally set through the SPiiPlus MMI Application Studio **EtherCAT Configurator** - see *SPiiPlus MMI Application Studio User Guide*.

For information about valid **CTIME** values see the EtherCAT Cycle Rate section in the Installation Guide for the relevant controller.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadReal, acsc_WriteReal

3.16.3 EXTFAC

Description

EXTFAC is a real array, with one element for each axis in the system. It is used as a conversion factor between the SL2-100 protocol transferred units (microns [μm]) and ACS user units.

Syntax

EXTFAC(axis)

Argument

| | |
|------|---------------------------------------------------------------------------------------|
| axis | Axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|------|---------------------------------------------------------------------------------------|

3.16.4 FOLLOWCH

Description

FOLLOWCH is an integer array, with one element for each axis in the system, the elements of which maps an axis to a data channel of a SLEEC module.

Syntax

FOLLOW(axis)xbit

Argument

| axis | Axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
|-------|---------------------------------------------------------------------------------------|
| bit | A description of the AST bit designators is: |
| bit | Description |
| 0-15 | Data channel of SLEC |
| 16-31 | Unit ID of SLEC |

Example 1

For an EtherCAT system that includes the following devices:

- > SPiiPlusEC motion controller and EtherCAT master
- > SPiiPlusCMHv EtherCAT control module
- > SLEC. EtherCAT node

SLEC node number 1:

- > Axis 0 will follow channel 0
- > Axis 2 will follow channel 1

```
FOLLOWCH (0) = 0x00010000
FOLLOWCH (2) = 0x00010001
```

Example 2

For an EtherCAT system that includes the following devices:

- > SpiiPlusEC motion controller and EtherCAT master
- > SLEC EtherCAT node
- > MC4U(8 axes)
- > MC4U(8 axes)
- > SLEC.EtherCAT node



Each MC4U includes 2 drives with 4 axes each for a total of 16 total axes..

SLEC node number 0:

- > Axes 0 will follow channel 0
- > Axes 1 will follow channel 1.

SLEC node number 3:

- > Axes 8 will follow channel 0

> Axes 9 will follow channel 1.

```
FOLLOWCH (0) = 0x00000000
FOLLOWCH (1) = 0x00000001
FOLLOWCH (8) = 0x00050000
FOLLOWCH (9) = 0x00050001
```

3.16.5 G_01WCS...G_12WCS

Description

G_01WCS to **G_012WCS** are each a real array, with one element for each axis in the system (up to 9 axes). Each is used for defining one of the 12 work-piece coordinate systems a user can set as part of the CNC setup/configuration. During the execution of a GSP program, a user can select one of them to work. If user choses going back to work in the Machine Coordinate System, he can then chose to clear it.

Syntax

N/A

Arguments

N/A

Tag

284...295

Accessibility

Read Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.16.6 GPEXL

Description

GPEXL is an integer array, with one element for each program buffer in the system. It indicates the GSP program executed block. If block executes motion then GPEXL will indicate this line till this motion completion.

If a buffer is not running, GPEXL equals 0.

Syntax

N/A

Arguments

N/A

Tag

296

Accessibility

Read Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.16.7 GSPEXL

Description

GSPEXL is an integer array, with one element for each program buffer in the system. It indicates the line in a running buffer that called a subroutine.

If a buffer is not running, GSPEXL equals 0.

Syntax

N/A

Arguments

N/A

Tag

347

Accessibility

Read Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.16.8 GUFAC

Description

GUFAC is a real array, with one element for each program buffer in the system. Each entry in the array holds the value a conversion factor, from 'Common Physical Units' in [mm] to 'Controller Units' (In the world of ACSPL+, those 'Controller Units' are sometimes related to as 'User Units'). As part of GSP modality data, default value of GUFAC is 1.0 for all buffers.

Syntax

N/A

Arguments

N/A

Tag

298

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadReal

3.16.9 IENA

Description

IENA is a 23-bit mask variable used for enabling or disabling software and hardware interrupts from a specific source.

Syntax

IENA.*bit_designator* = 1|0

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------|
| <i>bit_designator</i> | The meanings of bit_designator are given in Table 3-22 . |
|-----------------------|---------------------------------------------------------------------------------|

Table 3-22. IENA Bit Description

| Bit | Interrupt |
|-----|----------------------------------------------------------------------|
| 7 | Enable MARK1 0 interrupt |
| 8 | Enable MARK2 0 interrupt |
| 9 | Enable MARK1 2 interrupt |
| 10 | Enable M2ARK2 2 interrupt |
| 11 | Enable MARK1 4 interrupt |
| 12 | Enable M2ARK2 4 interrupt |
| 13 | Enable MARK1 5 interrupt |
| 14 | Enable M2ARK2 5 interrupt |
| 15 | Enable Emergency Stop interrupt |
| 16 | Enable Physical motion end interrupt |
| 17 | Enable Logical motion end interrupt |
| 18 | Enable Motion failure (Motion interruption due to a fault) interrupt |

| Bit | Interrupt |
|-----|---------------------------------------------------------------|
| 19 | Enable Motor failure (Motor disable due to a fault) interrupt |
| 20 | Enable Program termination interrupt |
| 21 | Enable Dynamic buffer interrupt |
| 22 | Enable ACSPL+ interrupt by INTERRUPT command |
| 23 | Enable Digital input interrupt |
| 24 | Enable Motion start interrupt |
| 25 | Enable Motion phase interrupt |
| 26 | Enable ACSPL+ interrupt by TRIGGER command |

Tag

69

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.10 IMASK**Description**

IMASK is an integer array, with one element for each axis in the system, the elements of which contain a set of bits that define which motor index and mark signals are processed.

Syntax

IMASK(*axis_index*).*bit_designator* = *value*

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>axis_index</i> | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| <i>bit_designator</i> | <p>IMASK has four bit designators:</p> <ul style="list-style-type: none"> > #IND (bit 0) - Primary encoder index > #IND2 (bit 1) - Secondary encoder |

| | |
|--------------|----------------------------------------------------------------------------|
| | index > #MARK (bit 2) - Mark1 > #MARK2 (bit 3) - Mark2 |
| value | value ranges from -2147483648 to 2147483647, Default = 13 |

Tag

70

Comments

If a bit is zero, the controller neither analyzes or latches the corresponding **INDEX** or **MARK** signal.

Every axis does not provide all **INDEX** and **MARK** signals. The secondary encoder index is available only if a secondary encoder is used. **MARK** signals are only available for axes 0, 1, 4, and 5.

Accessibility

Read-Write



IMASK values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, GetIndexState, ResetIndexState

C Library Functions

acsc_ReadInteger, acsc_WriteInteger, acsc_GetIndexState, acsc_ResetIndexState

3.16.11 ISENA**Description**

ISENA is an integer array, with one element for each axis in the system, the elements of which contain a set of 8 bits used for enabling or disabling software interrupts within a specific interrupt status bit for a specific axis or buffer. Each element corresponds to one software interrupt status bit and specifies which axis, buffers or inputs are enabled to cause interrupt.

Syntax


```
ISENAarray_index.bit_designator = 1|0
```

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------|
| array_index | Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| bit_designator | The meanings of bit_designator are given in Table 3-23 . |

Table 3-23. ISENA Bit Description

| Bit | Interrupt |
|-----|-------------------------------------------------------------|
| 0 | Controls Physical Motion End interrupt. |
| 1 | Controls Logical Motion End interrupt. |
| 2 | Controls Motion Failure interrupt. |
| 3 | Controls Motor Failure interrupt. |
| 4 | Controls Program Termination interrupt. |
| 5 | Controls Command Execution interrupt (dynamic buffer only). |
| 6 | Controls ACSPL+ interrupt (by INTERRUPT command). |
| 7 | Controls Digital Input interrupt. |



If the Physical Motion is PEG-related, then bit 0 is not supported in NT 1.0.

Tag

78

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.12 S_FLAGS

Description

S_FLAGS is an integer variable containing a set of bits that define different settings for the controller.

Syntax

S_FLAGS.*bit_designator* = 1|0

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------|
| <i>bit_designator</i> | The meanings of bit_designator are given in Table 3-24 . |
|-----------------------|---------------------------------------------------------------------------------|

Table 3-24. S_FLAGS Bit Description

| Bit Name | No. | Description | Remarks |
|----------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #S_FLAGS | 1 | <p>S_FLAGS.1 controls whether the controller allots a controller cycle to processing non-executable lines such as comments, empty lines and labels.</p> <p>0: Comments, empty lines, labels are skipped during execution and not allotted a controller cycle. (Default)</p> <p>1: These lines are each allotted a controller cycle.</p> | Changes to S_FLAGS.1 take effect only after a program is recompiled. |
| #FCLEAR | 2 | <p>0: Sets the controller to regular mode. (Default)</p> <p>1: Sets the controller to strict mode</p> | <p>In the regular mode the next motion command simply clears the reason for the previous KILL/KILLALL.</p> <p>In the strict mode, the next motion command cannot activate the motion and fails. Motion cannot be activated as long as the reason for the previous KILL is non-zero for any involved motor.</p> <p>Motion can continue only after clearing the MERR variable with ENABLE/ENABLE ALL or FCLEAR.</p> |

Tag

116

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.13 S_SETUP

Description

An integer variable containing a bit mask for defining various settings for the system.

Syntax


S_SETUP.bit_designator = 1|0

Arguments

bit_designator

The **S_SETUP** bit designators are given in [Table 3-25](#).

Table 3-25. S_SETUP Bit Designators

| Name | No. | Description |
|-----------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #USGTRACE | 1 | 0 (default) - Usage tracing is disabled 1 - Enables usage tracing (for debugging purposes only) |
| #SOFTIME | 2 | 0 (default) - EtherCAT frame delivery time measurement is disabled 1 - Enables EtherCAT frame delivery time measurement (for debugging purposes only) |
| #FRMLOSS | 3 | 0 (default) - single EtherCAT frame loss is not allowed. Every frame loss causes an EtherCAT error. 1 - single EtherCAT frame loss is allowed without causing an EtherCAT error. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> A single EtherCAT frame loss mode can affect the performance of ACS units, if this mode is in use, then the ACS units MUST be connected before non-ACS units in the network.</div> |
| #ENHPROT | 4 | 0 - backward compatible application protection. 1 (default) - allow enhanced application protection. |
| #CONFPROT | 5 | 0 (default) - prevent system reconfiguration In Protected Mode. 1 - allow system reconfiguration In Protected Mode. |
| #GMODE | 6 | 0 - backward compatible gantry mode for PEG, MARK, INDEX. 1 (default) - enable enhanced gantry mode for PEG, MARK, INDEX. |

| Name | No. | Description |
|------------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #POSSYNC | 7 | 0 - backward compatibility for FPOS/RPOS synchronization 1 (default) - enable FPOS/RPOS synchronization |
| #ESDBMODE | 8 | SPiiPlusES Debug Mode |
| #IOMNTMAP | 9 | 0 - IOMNT units are not mapped to ACSPL+ IN and OUT variables 1 (default) - IOMNT units are mapped to ACSPL+ IN and OUT variables |
| #IRMSLEG* | 10 | 0 (default) - RMS Protection is separated between Motor and Drive 1 - Legacy RMS protection only, using XRMS+XRMST |
| #USGTEMP | 11 | 0: Do not use default response for Motor Overheat, MPU Overheat or MPU Overuse 1: Exception default response for Motor Overheat, MPU Overheat or MPU Overuse is 'DISABLE'. |
| #HOME402 | 13 | CiA402 Homing Methods Enabled |
| #COMM402 | 14 | CiA402 Drive uses ACSPL+ COMMUT Command |
| #FASTPWUP | 15 | Optimized power-up time |
| #CTDBRAKE | 19 | When the bit is ON, products that support Controlled Current Dynamic Brake will operate in the new mode. If the bit is OFF, Dynamic Brake is applied. Default value is ON. |
| #SINCOSREP | 20 | When the bit is ON, the mechanism compensation for any quadrature misalignment between the Analog counter and the quadrature counter during initialization is applied. The bit is ON by default. |



*Starting from version 2.60, after changing the RMSLEG bit, the system should be reconfigured using the System Setup component

Tag

240

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.14 XSEGAMAX**Description**

A real variable that defines the maximal angle required when configuring look-ahead processing angles.

Syntax

XSEGAMAX= value

Tag

262

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.15 XSEGAMIN**Description**

A real variable that defines the minimal angle required when configuring look-ahead processing angles.

Syntax

XSEGAMIN= value

Tag

263

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.16 XSEGRMAX**Description**

A real variable that defines the maximal radius difference required when configuring arcs.

Syntax

XSEGRMAX= value

Tag

264

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.16.17 XSEGRMIN

Description

A real variable that defines the minimal arc radius required when configuring arcs.

Syntax

XSEGRMIN= value

Tag

265

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.17 Communication Variables

Communication variables are used for establishing various communication parameters.

The Communication variables are:

| Name | Description |
|---------------|--------------------------------|
| BAUD | Serial Communication Baud Rate |
| COMMCH | Communication Channel |
| COMMFL | Communication Flags |
| CONID | Controller Identification |

| Name | Description |
|---------|------------------------------------------------------------------------------------------|
| DISPCH | Default Communication Channel |
| ECHO | Echo Communication Channel |
| GATEWAY | Contains the address of a network router that serves accessing another network segments. |
| SUBNET | Used to determine to what subnet an IP address belongs. |
| TCPIP | IP Address for 1 st Ethernet |
| TCPIP2 | IP Address for 2 nd Ethernet |
| TCPPORT | TCP port identifier |
| UDPPORT | UDP port identifier |

3.17.1 BAUD

Description

BAUD is an integer variable that defines the serial communication rate, given in bits per second.

Syntax

BAUD = *value*

Arguments

| | |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| value | <p>value can be one of the following:</p> <ul style="list-style-type: none"> > 300 > 1200 > 4800 > 9600 > 19200 > 57600 > 115200 |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tag

8

Comments

Changes to **BAUD** take effect only after controller restart.

Accessibility

Read-Write



BAUD values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable, OpenComSerial

C Library Commands

acsc_ReadInteger, acsc_WriteInteger, acsc_OpenComSerial

3.17.2 COMMCH

Description

COMMCH is an integer that stores a number representing the last activated communication channel.

Table 3-26. COMMCH Values

| Value | Description |
|-------|-------------------------------|
| 1 | Serial port 1 |
| 2 | Serial port 2 |
| 6 | Ethernet network (TCP) |
| 7 | Ethernet network (TCP) |
| 8 | Ethernet network (TCP) |
| 9 | Ethernet network (TCP) |
| 10 | Ethernet Point-to-Point (UDP) |
| 12 | PCI bus |
| 16 | MODBUS Slave |
| 36 | Ethernet network (TCP) |
| 37 | Ethernet network (TCP) |
| 38 | Ethernet network (TCP) |
| 39 | Ethernet network (TCP) |

Tag

15

Comments

When queried through a communication channel, **COMMCH** reads the number of the current communication channel.

COMMCH can be used in [SEND](#), or assigned to [DISPCH](#).

Accessibility

Read-Only.

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.17.3 COMMFL



This variable is for advanced users. Changing the default values of these bits is not recommended!

Description

COMMFL is a scalar variable containing a set of bits that affect controller communication.

Syntax

COMMFL.*bit_designator* = 1|0

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------------|
| <i>bit_designator</i> | The COMMFL bits and the meanings of their values are given in Table 3-27 . |
|-----------------------|---------------------------------------------------------------------------------------------------|

Table 3-27. COMMFL Bit Descriptions

| Bit Name | Bit No. | Description |
|----------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #VERBOSE | 0 | Controls error message form. 0 = The controller provides the error number only to the C Library function or COM method, when an error occurs. 1 = The controller provides an extended message |
| | 1 | 1 = Enable motor messages |
| | 2 | 1 = Enable Axis messages |
| | 3 | 1 = Enable Program messages |
| #SAFEMSG | 4 | 1 = Controller sends unsolicited messages in Safe communication format |

| Bit Name | Bit No. | Description |
|----------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #CSUMMSG | 6 | 1 = A checksum is included in unsolicited messages. Normally the user does not need to change this bit. |
| #NOCOMM | 7 | Controls the communication in protected mode. 1 = The controller ignores any command received via communication channels except the queries that start from '?' character. The bit is not effective if the controller is in configuration mode. The default value is 0. |
| #NOQUERY | 8 | Controls the communication in protected mode. 1 = The controller ignores any query received via communication channels. The bit is not effective if the controller is in configuration mode. The default value is 0. |

Tag

16

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Commands

acsc_ReadInteger, acsc_WriteInteger

3.17.4 CONID**Description****CONID** is an integer variable that contains the controller identification.**Syntax****CONID** = *value***Arguments**

| | |
|--------------|-----------------------------------------|
| value | An integer ranging between 0 and 65536. |
|--------------|-----------------------------------------|

Tag

193

Comments

The controller identification can be used for many different purposes like Modbus Slave ID, CAN Slave ID or user-defined unique ID within the user network.



According to the Modbus specification, the controller must have an individual address from 1 to 247.

In order to specify the Modbus Slave address, **CONID** should be initialized to the Modbus Slave address value.

By default, the variable has zero value.

Accessibility

Read-Write



CONID values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Commands

acsc_ReadInteger, acsc_WriteInteger

3.17.5 ECHO

Description

ECHO is a 10-member mask integer variable that defines an echo communication channel.

Syntax

ECHO = *channel_number*

Arguments

channel_number

The values of **channel_number** and their meanings are given in [Table 3-28](#).

Table 3-28. ECHO Channel Numbers

| Channel Number | Channel Name |
|----------------|------------------------|
| -1 | Echo NOT active |
| -2 | All channels |
| 1 | Serial port 1 |
| 2 | Serial port 2 |
| 6 | Ethernet network (TCP) |
| 7 | Ethernet network (TCP) |

| Channel Number | Channel Name |
|----------------|-------------------------------|
| 8 | Ethernet network (TCP) |
| 9 | Ethernet network (TCP) |
| 10 | Ethernet Point-to-Point (UDP) |
| 12 | PCI bus |
| 16 | MODBUS Slave |
| 36 | Ethernet network (TCP) |
| 37 | Ethernet network (TCP) |
| 38 | Ethernet network (TCP) |
| 39 | Ethernet network (TCP) |

Tag

35

Comments

If **ECHO** specifies a valid communication channel, the controller sends an echo of each command received from any communication channel to the specified channel. Address each channel as follows:

The default value for **ECHO** is -1, in order to select an echo channel, the user needs to select a channel number.

ECHO cannot be saved to flash. After power-up the value is set to -1.

Use **DISPCH** to configure the communication channels related to the controller. Use **COMMCH** to retrieve the current controller channel's physical connection (only the channel that is connected to the terminal on which the query is sent).

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.17.6 DISPCH**Description**

DISPCH is a scalar integer variable that defines a communication channel between the controller and a host application, SPiiPlus MMI Application Studio or any device connected to the controller's

communication ports.

Syntax

DISPCH = *channel_number*

Arguments

| | |
|-----------------------|--------------------------------------------------------------------------------------------------|
| <i>channel_number</i> | The values of channel_number and their meanings are given in Table 3-29 . |
|-----------------------|--------------------------------------------------------------------------------------------------|

Table 3-29. DISPCH Channel Numbers

| Channel Number | Description |
|----------------|-------------------------------------------------------------------------------------------|
| -2 | All channels |
| -1 | No default channel is specified, the command uses the last channel activated by the host. |
| 1 | Serial port 1 |
| 2 | Serial port 2 |
| 6 | Ethernet network (TCP) |
| 7 | Ethernet network (TCP) |
| 8 | Ethernet network (TCP) |
| 9 | Ethernet network (TCP) |
| 10 | Ethernet Point-to-Point (UDP) |
| 12 | PCI bus |
| 16 | MODBUS Slave |
| 36 | Ethernet network (TCP) |
| 37 | Ethernet network (TCP) |
| 38 | Ethernet network (TCP) |
| 39 | Ethernet network (TCP) |

Tag

25

Comments

DISPCH is relevant only to messages sent with **DISP** and **SEND** (described also as “Unsolicited Messages”).

In order to view unsolicited messages in the SPiiPlus MMI Application Studio **Communication Terminal** window, select the check box in the lower right corner of the Terminal window to enable **Show Unsolicited Messages**.

If **DISPCH** specifies a valid communication channel, all unsolicited messages (messages that are sent with **DISP** and **SEND** from the program buffers) are sent to this channel irrespective of the channel used for immediate commands.

Accessibility

Read-Write

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.17.7 GATEWAY

Description

GATEWAY is an integer variable used for setting the address of a network router that serves for accessing another network segment.



The configuration is only available for the first Ethernet port.

Syntax

GATEWAY = *value*

Arguments

value

value an hexadecimal number - format: 0xAABBCCDD.

Tag

227

Comments

The **GATEWAY** address **value** consists of 4 individual bytes, each byte containing a decimal number ranging from 0 to 255. The bytes, when read, include a dot between each byte, with the least significant byte of the value representing the first decimal number. For example, the value 0x0100000A is the address: 10.0.0.1.

If controller is configured to obtain network settings from a DHCP server, **GATEWAY** contains the gateway address received from DHCP server

Accessibility

Read-Write



GATEWAY values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.17.8 SUBNET**Description**

SUBNET is an integer variable used to determine to what subnet an IP address belongs.



The configuration is only available for the first Ethernet port.

Syntax**SUBNET** = *value***Arguments***value***value** an hexadecimal number - format: 0xAABBCCDD.**Tag**

228

Comments

The **SUBNET value** consists of 4 individual bytes, each byte containing a decimal number ranging from 0 to 255. The bytes, when read, include a dot between each byte, with the least significant byte of the value representing the first decimal number. For example, the value 0x00FFFFFF represents mask 255.255.255.0.

If controller is configured to obtain network settings from a DHCP server, **SUBNET** contains the address received from DHCP server.

Accessibility

Read-Write



SUBNET values cannot be modified if protection is applied to this variable through **SPIIPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.17.9 TCPIP

Description

TCPIP is an integer variable used for setting the TCP/IP for the Ethernet port N1.

Syntax

TCPIP = *value*

Arguments

| | |
|--------------|----------------------------------------------------------|
| value | value an hexadecimal number - format: 0xAABBCCDD. |
|--------------|----------------------------------------------------------|

Tag

133

Comments

If **TCPIP** has a non-zero value, the controller uses the value as its TCP/IP address. In this case, other configuration parameters receive the following default values:

- > Subnet mask - 255.255.255.0
- > Gateway address - no gateway, i.e., no routing is supported

If **TCPIP** is zero, the controller uses the DHCP protocol to receive the network configuration from the DHCP server. The network configuration received from the DHCP server includes the following parameters:

- > Controller's TCP/IP address
- > Subnet mask
- > Gateway address

The **TCPIP** variable value has to be in hex, for example:

```
TCPIP=0x6400000a
```

assigns a TCP/IP address of: 10.0.0.100. Note that the address is calculated starting from the least significant byte of the value.

To retrieve the assigned address in an ACSPL+ program, use the [GETCONF](#) function with key 310.

Accessibility

Read-Write



TCPIP values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Commands

acsc_ReadInteger, acsc_WriteInteger, acsc_GetEthernetCards (to find all SPiiPlus controllers in the network segment)

3.17.10 TCPIP2

Description

TCPIP2 is an integer variable used for setting the TCP/IP for a second Ethernet port: N2.

Syntax

TCPIP2 = *value*

Arguments

value

value an hexadecimal number - format: 0xAABBCCDD.

Tag

198

Comments

If **TCPIP2** is zero, the address will be automatically obtained at the controller start-up through DHCP protocol. The default address for second Ethernet port is 192.168.0.100.

The **TCPIP2** variable value has to be in hex: 0xAABBCCDD, for example:

```
TCPIP=0x6400000a
```

assigns a TCP/IP address of: 10.0.0.100. Note that the address is calculated starting from the least significant byte of the value.

To retrieve the assigned address in an ACSPL+ program, use the [GETCONF](#) function with key 310.

Accessibility

Read-Write



TCPIP2 values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Commands

acsc_ReadInteger, acsc_WriteInteger, acsc_GetEthernetCards (to find all SPiiPlus controllers in the network segment)

3.17.11 TCPSPORT

Description

TCPSPORT is a scalar integer that stores a number representing a TCP port.

Syntax

TCPSPORT = *Port_number*

Arguments

| | |
|--------------------|-----------------------------------------|
| <i>Port_number</i> | An integer ranging between 0 and 65536. |
|--------------------|-----------------------------------------|

Tag

200

Comments

TCPSPORT defines Ethernet ports in the controller for TCP. By default, this variable is set to 701. In order to establish communication with the controller through a port different from default port numbers, the following should be done:

1. Set **TCPSPORT** to a value other than 701.



Some of the ports are used by the controller firmware and cannot be used. It is recommended to use ports starting from 1024.

2. Save system parameters to the flash.
3. Restart the controller.
4. Try to establish communication using new ports by providing them in client user application. If communication isn't established, try to set other values.



This new port value is used by the client user application only. The SPiiPlus Tools and SPiiPlus C/COM Library continue to use the default ports.

Accessibility

Read-Write



TCPSPORT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.17.12 UDPPORT

Description

UDPPORT is a scalar integer that stores a number representing a UDP port.

Syntax

UDPPORT = *Port_number*

Arguments

Port_number

An integer ranging between 0 and 65536.

Tag

201

Comments

UDPPORT defines Ethernet ports in the controller for UDP. By default, it is set to 700. In order to establish communication with the controller through different from default port numbers, the following should be done:

1. Set **UDPPORT** to a value other than 700.



Some of the ports are used by the controller firmware and cannot be used. It is recommended to use ports starting from 1024.

2. Save system parameters to the flash.
3. Restart the controller.
4. Try to establish communication using new ports by providing them in client user application. If communication isn't established, try to set other values.



This new port value is used by the client user application only. The SPiiPlus Tools and SPiiPlus C/COM Library continue to use the default ports.

Accessibility

Read-Write



UDPPORT values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio** → **Toolbox** → **Application Development** → **Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

3.18 Miscellaneous

The Miscellaneous variables are:

| Name | Description |
|----------|------------------------------------------------|
| FK | Function Key |
| STATIC | Tag used to define a global variable as STATIC |
| XARRSIZE | Maximum Array Size |

3.18.1 FK

Description

FK is a scalar integer and is used by the controller to store function keys in an input string processed with the **INPUT** command.

Tag

50

Comments

If the controller encounters a zero character in the input string, it stores the value of the next character in **FK**.

This is the usual way for loading function key codes (F1 - F12).



An autoroutine can be used to respond to changes in **FK**.

Accessibility

Read-Only

COM Library Methods and .NET Library Methods

ReadVariable

C Library Functions

acsc_ReadInteger

3.18.2 STATIC

Description

A new tag can be used to define a global variable as **STATIC**.

STATIC variables can only be defined in the D-buffer and once defined can only be freed using the **#VGV** command, that is: removing the definition from the D-buffer does not remove the variable.

Use the **#VGS/#VGSF** command to list all static variables in the system.

Syntax

```
GLOBAL INT/REAL STATIC Var
```

3.18.3 XARRSIZE

Description

XARRSIZE is a scalar integer that stores maximum size of an array.

Syntax

XARRSIZE = value

Arguments

value

value ranges from 10,000 to 10,000,000 (elements); Default = 100,000

Tag

219

Comments

By default the maximum size for a user array is 100,000 elements; if, however, an application requires larger arrays, the user may change this value to accommodate a larger array size. However the following should be taken into consideration:

1. Defining large arrays may use too much memory and may cause an out of memory fault. To avoid this, the RAM size available for user data in the specific controller model should be checked. One element in an array requires 8 bytes of RAM, 131,072 elements require 1 MB.
2. The processing time required for operations on large arrays in ACSPL+ may cause an over usage fault. Therefore, arrays should be defined only with the size actually required for the application.



It is strongly recommended that users change the **XARRSIZE** variable only if necessary, and that such changes be tested under safe conditions.

Accessibility

Read-Write



XARRSIZE values cannot be modified if protection is applied to this variable through **SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection**

COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

C Library Functions

acsc_ReadInteger, acsc_WriteInteger

4. ACSPL+ Functions

ACSPL+ functions are divided into the following categories:

- > [Arithmetical Functions](#)
- > [Matrix Functions](#)
- > [Miscellaneous Functions](#)
- > [EtherCAT Functions](#)
- > [CoE Functions](#)
- > [Modbus Functions](#)
- > [Servo Processor Functions](#)
- > [Signal Processing Functions](#)
- > [Laser Control Functions](#)
- > [Dynamic Error Compensation](#)

This chapter covers the ACSPL+ functions.

The ACSPL+ Functions, in alphabetical order, are:

| Function | Description |
|------------------------------|-------------------------------------------------------------------------|
| ABS | Calculates the absolute value. |
| ACOS | Calculates the arc cosine. |
| AINOFFS | Percent offset of analog signal from external source |
| AINSCALE | Define scale of analog input signal |
| ASIN | Calculates the arc sine. |
| ATAN | Calculates the arctangent. |
| ATAN2 | Calculates the arctangent of X/Y. |
| AxListAsMask | Mask for defining axes. |
| AVG | Finds the average of all values in an array. |
| BCOPY | Copies a given number of bytes from a source array into a target array. |
| CEIL | Calculates the ceiling of a value. |
| COPY | The function copies one array to another. |

| Function | Description |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COERead | Read CoE slave Object Directory entry. |
| COS | Calculates the cosine. |
| COEWRIte | Write into CoE slave Object Directory. |
| DEADZONE | Implements dead-zone routine. |
| DSIGN | Implements a dynamic version of the standard SIGN function. |
| DSTR | Converts a string to an integer array. |
| ECIN | Copy EtherCAT offset data into ACSPL+ variable |
| ECGETGRPIND | Returns an array that contains optional groups' indexes that are part of the current configuration |
| ECGETOPTGRP | Returns number of actually connected optional groups, not including the mandatory group. |
| ECGETSLAVES | Returns the number of EtherCAT slaves in the network |
| ECGRPINFO | <ul style="list-style-type: none"> > Fills array with nodes' indexes which are members of a given optional group > Returns the number of members in the group. |
| ECOUT | Copy data of ACSPL+ variable into EtherCAT offset |
| ECRESCUE | Triggers execution of a rescue scan of the EtherCAT network |
| ECSAVECFG | Saves to flash an array of optional groups based on current configuration, based on <code>ecgetgrpind()</code> function. |
| ECSAVEDCNF | Returns array that contains optional groups' indexes that are part of the last saved configuration |
| ECUNMAP | Undoes any results from running ECIN and ECOUT |
| ECUNMAPIN | Undoes any results from running ECIN to a specific offset. |
| ECUNMAPOUT | Undoes any results from running ECOUT to a specific offset. |
| EDGE | Returns 1 on positive edge of x. |
| ENCREAD | Read encoder parameters |

| Function | Description |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| ERRORMAP1D | Configures and activates 1D error correction for the mechanical error compensation for the specified zone, |
| ERRORMAP2D | Configures and activates 2D error correction for the mechanical error compensation of the 'axis0' command for the specified zone, |
| ERRORMAPN1D | Configures and activates 1D error correction for the mechanical error compensation for the specified zone |
| ERRORMAPN2D | Configures and activates 2D error correction for the mechanical error compensation of the 'axis0' command for the specified zone, |
| ERRORMAPOFF | Deactivates error mapping correction for the mechanical error compensation for the specified zone. |
| ERRORMAPON | Activates error correction for the mechanical error compensation for the specified zone. |
| #ERRORMAPREP | Generates a report of all activated zones of error mapping for all axes in the system. |
| ERRORUNMAP | Deactivates error correction for the mechanical error compensation for the specified zone. |
| EXP | Calculates the exponent. |
| FLOOR | Calculates the floor of a value. |
| GETCONF | Reads hardware and firmware parameters. |
| GETSP | Reads a value from the specified SP address. |
| GETSPA | Retrieves address of the SP variable specified by name. |
| GETSPV | Reads a value from the specified SP variable name. |
| HYPOT | Calculates the hypotenuse. |
| INP | Reads data characters from the specified channel and stores them into integer array. |
| INTGR | Implements an integrator with DEADZONE and SAT . |
| LAG | Provides delayed switching on argument change (anti-bouncing effect). |

| Function | Description |
|---------------------------|--------------------------------------------------------------------------------------------------------|
| <code>LCDelayGet</code> | Returns the actual currently configured delay in microseconds. |
| <code>LCDelaySet</code> | Sets the pulse generation delay in microseconds. |
| <code>LCFixedDist</code> | Initializes the fixed distance pulse firing mode. |
| <code>LCFixedInt</code> | Initializes the fixed distance pulse firing mode. |
| <code>LCMODULATION</code> | Initializes Pulse modulation mode and sets initial values for the unit internal registers. |
| <code>LCOutputGet</code> | Returns the laser outputs configuration. |
| <code>LCOutputSet</code> | Configures the laser physical outputs. |
| <code>LCRandomDist</code> | Initializes either array based pulse firing mode or gating mode. |
| <code>LCSignalGet</code> | Returns the laser control signal (LCS) output conditioning state. |
| <code>LCSignalSet</code> | Configures the laser control signal (LCS) output conditioning state. |
| <code>LCStop</code> | Stops any previously initialized laser mode and resets the previously defined mode parameters. |
| <code>LCTickle</code> | Initializes Tickle mode. |
| <code>LCZone</code> | Sets a laser operation zone area. |
| <code>LCZoneGet</code> | Returns the limits of the laser operation zone that was previously defined |
| <code>LCZoneSet</code> | Changes the minimal and/or maximal laser operationzone limit. |
| <code>LDEXP</code> | Calculates a value of $x*2^{exp}$. |
| <code>LOG</code> | Calculates the natural logarithm. |
| <code>LOG10</code> | Calculates the base-10 logarithm. |
| <code>MAP</code> | Implements a table-defined function with constant step. |
| <code>MAPB</code> | One-dimensional uniform b-spline. |
| <code>MAPN</code> | One-dimensional non-uniform linear interpolation (replaces obsolete MAPBY1 and MAPBY2). |

| Function | Description |
|----------|-----------------------------------------------------------------------------------------------------|
| MAPNB | One-dimensional non-uniform B-spline. |
| MAPNS | One-dimensional non-uniform Catmull-Rom spline. |
| MAPS | One-dimensional uniform Catmull-Rom spline. |
| MAP2 | Implements a table-defined function with two arguments and constant step along each argument. |
| MAP2B | Two-dimensional uniform B-spline. |
| MAP2N | Two-dimensional non-uniform linear interpolation (replaces obsolete MAP2FREE). |
| MAP2NB | Two-dimensional non-uniform B-spline. |
| MAP2NS | Two-dimensional non-uniform Catmull-Rom spline. |
| MAP2S | Two-dimensional uniform Catmull-Rom spline. |
| MATCH | Calculates axis position that matches current reference position of the same axis with zero offset. |
| MAX | Finds the maximum value in an array. |
| MAXI | Finds the element with maximum value in an array or in a section of an array and returns its index. |
| MIN | Finds the minimum value in an array. |
| MINI | Finds the element with minimum value in an array or in a section of an array and returns its index. |
| NUMTOSTR | Converts a number to an ASCII string. |
| POW | Calculates x raised to the power of y. |
| RAND | Implements a random number generator. |
| ROLL | Calculates a result rolled-over to within one pitch. |
| SAT | Implements a saturation characteristic. |
| SETCONF | Writes hardware and firmware parameters. |
| SETSP | Sets a value for the specified SP address. |

| Function | Description |
|----------|--------------------------------------------------------------------------------------------------|
| SETSPV | Sets a value for the specified SP variable name. |
| SETVAR | Writes a value to a system or user variable, scalar or array, that was declared as a Tag number. |
| SIGN | Returns -1, 0 or 1 depending on the sign of x. |
| SIN | Calculates the sine. |
| SQRT | Calculates the square root. |
| STR | Converts an integer array to a string. |
| STRTONUM | Converts an ASCII string representing a number to the number it represents. |
| SYSINFO | Returns certain system information based on the argument that is specified. |
| TAN | Calculates the tangent. |

4.1 *Arithmetical Functions*

The Arithmetical functions are:

| Function | Description |
|----------|-----------------------------------|
| ABS | Calculates the absolute value |
| ACOS | Calculates the arc cosine |
| ASIN | Calculates the arc sine |
| ATAN | Calculates the arctangent |
| ATAN2 | Calculates the arctangent of Y/X |
| CEIL | Calculates the ceiling of a value |
| COS | Calculates the cosine |
| EXP | Calculates the exponent |
| FLOOR | Calculates the floor of a value |
| HYPOT | Calculates the hypotenuse |
| LDEXP | Calculates a value of $x*2^{exp}$ |

| Function | Description |
|----------|-----------------------------------------------|
| LOG | Calculates the natural logarithm |
| LOG10 | Calculates the base 10 logarithm |
| POW | Calculates x raised to the power of y |
| SIGN | Returns -1, 0 or 1 depending on the sign of x |
| SIN | Calculates the sine |
| SQRT | Calculates the square root |
| TAN | Calculates the tangent |
| ROUND | Rounds REAL to INTEGER |

4.1.1 ABS

Description

ABS calculates the absolute value

Syntax

ABS(*input*)

Arguments

input

input can be a real number or an expression.

Return Value

Real number - returns the absolute value of the input.

Error Conditions

None

Example

```
XX = ABS (-3.14)
DISP XX !Output = 3.14
```

4.1.2 ACOS

Description

ACOS calculates arc cosine.

Syntax

ACOS(*input*)

Arguments

input **input** can be a real number or an expression.

Return Value

Real number - returns the arc cosine of the argument in the range from 0 to π radians.

Error Conditions

The value of **input** must be between -1 to 1, otherwise the function returns Error 3045, Numerical Error in Standard Function.

Example

```
XX = ACOS (-1)
DISP XX                !Output = 3.141592654
```

4.1.3 ASIN

Description

ASIN calculates the arc sine.

Syntax

ASIN(*input*)

Arguments

input **input** can be a real number or an expression.

Return Value

Real number - returns the arc sine of XX in the range $-\pi/2$ to $\pi/2$.

Error Conditions

The value of **input** must be between -1 to 1, otherwise the function returns Error 3045, Numerical Error in Standard Function.

Example

```
XX = ASIN (-1)
DISP XX                !Output = -1.570796327
```

4.1.4 ATAN

Description

ATAN calculates the arctangent.

Syntax

ATAN(*input*)

Arguments

input **input** can be a real number or an expression.

Return Value

Real number - returns the arctangent of the input in the range $-\pi/2$ to $\pi/2$ radians.

Error Conditions

None

Example

```
XX = ATAN(-1)
DISP XX                !Output = -0.7853981634
```

4.1.5 ATAN2

Description

ATAN2 calculates the arctangent of **X/Y**.

Syntax

ATAN2(*X_input*,*Y_input*).

Arguments

| | |
|----------------|-------------------------------------------------------|
| X_input | X_input can be a real number or an expression. |
|----------------|-------------------------------------------------------|

| | |
|----------------|-------------------------------------------------------|
| Y_input | Y_input can be a real number or an expression. |
|----------------|-------------------------------------------------------|

Return Value

Real number - returns the arctangent value of **X_input**, **Y_input**. **ATAN2** calculates a value in the range of $-\pi$ to π radians using the signs of both parameters to determine the quadrant of the return value. If both parameters are 0, the function returns 0. **ATAN2** is well defined even if Y equals 0.

Error Conditions

None

Example

```
X_input = -1; Y_input = 0
XX=ATAN2(X_input,Y_input)
DISP XX                !Output = -1.5708
```

4.1.6 CEIL

Description

CEIL calculates the ceiling of a value.

Syntax

CEIL(*input*)

Arguments

| | |
|--------------|-----------------------------------------------------|
| input | input can be a real number or an expression. |
|--------------|-----------------------------------------------------|

Return Value

Integer number - returns a value that represents the smallest integer that is ³ **input**.

Error Conditions

None

Example

```
XX=CEIL(3)
YY=CEIL(-3)
ZZ=CEIL(2.1)
TT=CEIL(-2.1)
DISP XX, YY, ZZ, TT !Output = 3 -3 3 -2
```

4.1.7 COS

Description

COS calculates the cosine.

Syntax

COS(*input*)

Arguments

input

input can be a real number or an expression (which is treated as radians by the function).

Return Value

Real number - returns the cosine of X in the range of -1 to 1.

Error Conditions

None

Example

```
XX = COS(-3.141592654)
DISP XX !Output = -1
```

4.1.8 EXP

Description

EXP calculates the e^{input}

Syntax

EXP(*input*)

Arguments

input

input can be a real number or an expression.

Return Value

Real number - returns the exponential value of **input**. On overflow, the function returns the largest real number.

Error Conditions

None

Example

```
XX = EXP (1)
DISP XX !Output = 2.718281828
```

4.1.9 FLOOR

Description

FLOOR calculates the floor of a value.

Syntax

FLOOR(*input*)

Arguments

input

input can be a real number or an expression.

Return Value

Integer number - returns a value representing the largest integer that is \leq to **input**.

Error Conditions

None

Example

```
XX=FLOOR (3)
YY=FLOOR (-3)
ZZ=FLOOR (2.1)
TT=FLOOR (-2.1)
DISP XX,YY,ZZ,TT !Output = 3 -3 2 -3
```

4.1.10 HYPOT

Description

HYPOT calculates the hypotenuse of a right triangle

Syntax

HYPOT(*X_input*, *Y_input*)

Arguments

X_input

X_input can be a real number or an expression.

Y_input

Y_input can be a real number or an expression.

Return Value

Real number - calculates the length of the hypotenuse of a right triangle, given the length of the two sides **X_input** and **Y_input**. **HYPOT** is equivalent to the square root of $X^2 + Y^2$.

Error Conditions

None

Example

```
XX=HPOT(3,4)
DISP XX           !Output = 5
```

4.1.11 LDEXP**Description**

LDEXP calculates the value of $x*2^{exp}$.

Syntax

LDEXP(*X_input*, *Y_input*)

Arguments

| | |
|----------------|-------------------------------------------------------|
| X_input | X_input can be a real number or an expression. |
| Y_input | Y_input can be a real number or an expression. |

Return Value

Real number - returns the value of $X_input*2^{Y_input}$. On overflow **LDEXP** returns the largest real number with a sign, depending on the sign of **X_input**

Error Conditions

None

Example

```
XX= LDEXP(1,2)
DISP XX           !Output = 4
```

4.1.12 LOG**Description**

LOG calculates the natural logarithm.

Syntax

LOG(*input*)

Arguments

| | |
|--------------|-----------------------------------------------------|
| input | input can be a real number or an expression. |
|--------------|-----------------------------------------------------|

Return Value

Real number - returns the natural logarithm of **input**.

Error Conditions

input must be > 0, otherwise the function returns Error 3045, Numerical Error in Standard Function.

Example

```
XX=LOG(2.718281829)
DISP XX                !Output = 1
```

4.1.13 LOG10

Description

LOG10 calculates the base 10 logarithm.

Syntax

LOG10(*input*)

Arguments

| | |
|--------------|-----------------------------------------------------|
| <i>input</i> | input can be a real number or an expression. |
|--------------|-----------------------------------------------------|

Return Value

Real number - returns the base 10 logarithm of **input**.

Error Conditions

input must be >0, otherwise the function returns Error 3045, Numerical Error in Standard Function.

Example

```
XX=LOG10(10)
DISP XX                !Output = 1
```

4.1.14 POW

Description

POW calculates X raised to the power of Y.

Syntax

POW(*X_input*, *Y_input*)

Arguments

| | |
|----------------|-------------------------------------------------------|
| <i>X_input</i> | X_input can be a real number or an expression. |
|----------------|-------------------------------------------------------|

| | |
|----------------|-------------------------------------------------------|
| <i>Y_input</i> | Y_input can be a real number or an expression. |
|----------------|-------------------------------------------------------|

Return Value

Real number - returns the value of **(X_input)^{Y_input}**.

Error Conditions

None

Example

```
XX=POW(2,3)
DISP XX           !Output = 8
```

4.1.15 SIGN

Description

SIGN returns -1, 0 or 1 depending if the input is negative, zero or positive.

Syntax

SIGN(*input*)

Arguments

| | |
|--------------|-----------------------------------------------------|
| <i>input</i> | <i>input</i> can be a real number or an expression. |
|--------------|-----------------------------------------------------|

Return Value

Real number - returns:

-1 if *input* < 0;

0 if *input* = 0;

1 if *input* > 0

Error Conditions

None

Example

```
XX=SIGN(-5), SIGN(0), SIGN(5)
DISP XX           !Output = -1 0 1
```

4.1.16 SIN

Description

SIN calculates the sine.

Syntax

SIN(*input*)

Arguments

| | |
|--------------|---------------------------------------------------------------------------------------------------|
| <i>input</i> | <i>input</i> can be a real number or an expression (which is treated as radians by the function). |
|--------------|---------------------------------------------------------------------------------------------------|

Return Value

Real number - returns the sine value of *input* in the range of -1 to 1.

Error Conditions

None

Example

```
XX=SIN(1.570796327)
DISP XX                !Output = 1
```

4.1.17 SQRT**Description****SQRT** calculates the square root.**Syntax****SQRT**(*input*)**Arguments*****input*****input** can be a real number or an expression.**Return Value**Real number - returns the square root of **input**.**Error Conditions****input** must be ≥ 0 , otherwise the function returns Error 3045, Numerical Error in Standard Function.**Example**

```
XX=SQRT(4)
DISP XX                !Output = 2
```

4.1.18 TAN**Description****TAN** calculates the tangent.**Syntax****TAN**(*input*)**Arguments*****input*****input** can be a real number or an expression (which treated as radians by the function).**Return Value**Real number - returns the tangent value of **input**.**Error Conditions**

None

Example

```
REAL PI
PI = 3.141592654
DISP TAN(PI/4)                !Output = 1
```

4.1.19 ROUND

Description

Round a REAL number to closest integer value

Syntax

ROUND(input)

Arguments

Input - Arbitrary real number

Return Value

Closest integer value

Comments

ROUND calculates the closest integer number according to arithmetical rules.

This command is supported in ADK versions 2.70 and higher.

Examples:

```
DISP (ROUND (1.5))                !Output = 2
DISP (ROUND (1.4))                !Output = 1
DISP (ROUND (-1.5))               !Output = -2
DISP (ROUND (-1.4))               !Output = -1
```

4.2 Matrix Functions

The matrix functions are:

| Function | Overloaded Operator | Description |
|--------------|---------------------|------------------------------------|
| MATRIXADD | + | Add matrices |
| MATRIXSUB | - | Subtract matrices |
| MATRIXMUL | * | Multiply matrices |
| MATRIXMULSCA | * | Multiply matrix by scalar |
| MATRIXMULEW | .* | Element-wise matrix multiplication |
| MATRIXDIV | / | Matrix division |

| Function | Overloaded Operator | Description |
|---------------------------|---------------------|------------------|
| <code>MATRIXIDENT</code> | | Identity matrix |
| <code>MATRIXTRANS</code> | | Transpose matrix |
| <code>MATRIXINVERT</code> | | Invert matrix |

4.2.1 Matrix Type

ACSPL+ supports a **MATRIX** type starting with version 3.10.

A **MATRIX** is defined as 2-dimensional **REAL** array.

Syntax

- > `MATRIX A(N)(N)` – A is square matrix of size NxN where N is a positive constant.
- > `MATRIX B(N)(M)` – B is NxM matrix having N rows and M columns, where N and M are positive constants.

Example

```
MATRIX A(2)(2) !Defines a square matrix of 2x2 size
```

4.2.1.1 Matrix Initialization in Compilation Time

1. Regular Initialization (similar to 2D arrays initialization):
`MATRIX A(M)(N)` can be initialized by a 2D array (array of M rows, each of N size)
 - a. Syntax:

```
MATRIX A(2)(3) = ((1,2,3), (4,5,6))
```

- b. Preconditions:
 - i. Number of rows of initialization values = matrix first dimension
 - ii. Number of columns of initialization values = matrix second dimension
- c. Error conditions:
 - i. If preconditions are violated, then a compilation error will occur.
- d. Example:

```
!///Compilation-time initializations///!  
MATRIX A(2)(2)=((1,2), (3,4)) !Regular initialization of a 2x2 matrix
```

2. Default Initialization:
A matrix which not initialized by the user on definition is automatically initialized with zeros.
 - a. Syntax:

```
MATRIX A(M)(N)
```

Example:

```
!///Compilation-time initializations///!
MATRIX C(2) (2) !Default initialization, filled by zeros
```

4.2.2 MATRIXADD**Description**

Add two matrices.

Syntax

```
MATRIXADD([in] MATRIX A, [in] MATRIX B, [out] MATRIX C)
```

Arguments

| | |
|----------|-----------------------|
| MATRIX A | First operand matrix |
| MATRIX B | Second operand matrix |
| MATRIX C | Result Matrix |

Return Value

None

Comments

1. All argument matrices must be of the same size.
2. If preconditions are violated, then a compilation error will occur.
3. This function is supported in versions 3.11 and higher

Example

```
MATRIX A(2) (2) = ((1, 2), (3, 4))
MATRIX B(2) (2) = ((5, 6), (7, 8))
MATRIX C(2) (2)
MATRIXADD(A, B, C)
```

4.2.3 MATRIXSUB**Description**

Subtract two matrices.

Syntax

```
MATRIXSUB([in] MATRIX A, [in] MATRIX B, [out] MATRIX C)
```

Arguments

| | |
|----------|-----------------------|
| MATRIX A | First operand matrix |
| MATRIX B | Second operand matrix |
| MATRIX C | Result Matrix |

Return Value

None

Comments

1. All argument matrices must be of the same size.
2. If preconditions are violated, then a compilation error will occur.
3. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (2) (2) = ( (1, 2) , (3, 4) )
MATRIX B (2) (2) = ( (5, 6) , (7, 8) )
MATRIX C (2) (2)
MATRIXSUB (B, A, C)
```

4.2.4 MATRIXMUL**Description**

Multiply matrices

Syntax

MATRIXMUL([in] MATRIX A, [in] MATRIX B, [out] MATRIX C)

Arguments

| | |
|----------|-----------------------|
| MATRIX A | First operand matrix |
| MATRIX B | Second operand matrix |
| MATRIX C | Result Matrix |

Return Value

None

Comments

1. Matrix A's column dimension must equal Matrix B's row dimension.
2. The dimension of Matrix C's rows are equal to the row dimension of Matrix A
3. The dimension of Matrix C's columns are equal to the column dimension of Matrix B
4. If preconditions are violated, then a compilation error will occur.
5. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (3) (2) = ( (1, 2) , (3, 4) , (5, 6) )
MATRIX B (2) (2) = ( (7, 8) , (9, 10) )
MATRIX C (3) (2)
MATRIXMUL (A, B, C)
```

Example With Overloaded Operator

```
MATRIX A (3) (2) = ( (1, 2) , (3, 4) , (5, 6) )
MATRIX B (2) (2) = ( (7, 8) , (9, 10) )
MATRIX C (3) (2)
C=A*B
```

4.2.5 MATRIXMULSCA

Description

Multiply matrix by scalar value.

Syntax

MATRIXMULSCA([in] MATRIX A, [in] REAL S, [out] MATRIX C)

Arguments

| | |
|----------|----------------------|
| MATRIX A | First operand matrix |
| REAL S | Scalar operand |
| MATRIX C | Result Matrix |

Return Value

None

Comments

1. The result matrix is of the same size as operand matrix.
2. If preconditions are violated, then a compilation error will occur.
3. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (3) (2) = ( (1, 2) , (3, 4) , (5, 6) )
REAL r=7
MATRIX C (3) (2)
MATRIXMULSC (A, r, C)
MATRIXMULSC (A, 5.5, C)
STOP
```

4.2.6 MATRIXMULEW

Description

Element-wise matrix multiplication

Syntax

MATRIXMULEW([in] MATRIX A, [in] REAL S, [out] MATRIX C)

Arguments

| | |
|----------|----------------------|
| MATRIX A | First operand matrix |
| REAL S | Scalar operand |
| MATRIX C | Result Matrix |

Return Value

None

Comments

1. All parameter matrices must be of the same size.
2. If preconditions are violated, then a compilation error will occur.
3. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (2) (2) = ( (1, 2) , (3, 4) )
MATRIX B (2) (2) = ( (5, 6) , (7, 8) )
MATRIX C (2) (2)
MATRIXMULEW (A, B, C)
```

4.2.7 MATRIXDIV

Description

Matrix division

Syntax

MATRIXDIV([in] MATRIX A, [in] MATRIX B, [out] MATRIX C)

Arguments

| | |
|----------|-----------------|
| MATRIX A | Dividend matrix |
| MATRIX B | Divisor matrix |
| MATRIX C | Result Matrix |

Return Value

None

Comments

1. Matrix B is a square matrix.
2. The dimension of Matrix A's columns are equal to the order of Matrix A
3. C dimension = (A's rows, B's order) = A dimension
4. If preconditions are violated, then a compilation error will occur.
5. If Matrix B is not convertible (determinant(B) == 0) a run-time error will occur.
6. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (2) (3) = ( (1, 2, 3) , (4, 5, 6) )
MATRIX B (3) (3) = ( (1, 2, 3) , (4, 5, 6) , (7, 8, 1) )
MATRIX C (2) (3)
MATRIXDIV (A, B, C)
```

Example With Overloaded Operator

```
MATRIX A (2) (3) = ( (1, 2, 3) , (4, 5, 6) )
MATRIX B (3) (3) = ( (1, 2, 3) , (4, 5, 6) , (7, 8, 1) )
MATRIX C (2) (3)
C=A/B
```

4.2.8 MATRIXIDENT

Description

Generate identity matrix, a matrix filled with values of 1 on the main diagonal and all other elements are 0.

Syntax

MATRIXIDENT([in out] MATRIX A)

Arguments

MATRIX A

Square Matrix

Return Value

Identity matrix of dimension matching the input matrix.

Comments

1. The result matrix must be square.
2. If preconditions are violated, then a compilation error will occur.
3. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (3) (3)
MATRIXIDENT (A)
```


4.2.9 MATRIXTRANS

Description

Transpose a matrix.

Syntax

MATRIXTRANS([in]MATRIX A, [out] MATRIX B)

Arguments

| | |
|----------|----------------------------------------|
| MATRIX A | Input matrix to be transposed |
| MATRIX B | Output, transposed version of MATRIX A |

Return Value

None

Comments

1. The target matrix should be defined so that its row number is equal to the target matrix column number, and its column number is equal to target matrix row number.
2. If preconditions are violated, then a compilation error will occur.
3. This function is supported in versions 3.11 and higher

Example

```
MATRIX A (2) (3) = ( (1, 2, 3) , (4, 5, 6) )
MATRIX B (3) (2)
MATRIXTRANS (A, B)
```

4.2.10 MATRIXINVERT

Description

Invert a matrix.

Syntax

MATRIXINVERT([in] MATRIX A, [out] MATRIX B)

Arguments

| | |
|----------|--------------------------------------|
| MATRIX A | Input matrix to be inverted |
| MATRIX B | Output, inverted version of MATRIX A |

Return Value

None

Comments

1. The argument matrix must be square.
2. The result matrix should be of the same size as the argument matrix.
3. If preconditions are violated, then a compilation error will occur.

4. This function is supported in versions 3.11 and higher

Example

```
MATRIX A(3)(3) = ((1,2,3), (4,5,6), (7,8,1))
MATRIX B(3)(3)
MATRIXINV(A,B)
```

4.3 Miscellaneous Functions

The Miscellaneous functions are:

| Function | Description |
|----------|-----------------------------------------------------------------------------------------------|
| GETCONF | Reads hardware and firmware parameters. |
| SYSINFO | Returns certain system information based on the argument that is specified. |
| GETVAR | Reads the current value of the variable and returns it as a real number. |
| SETCONF | Writes hardware and firmware parameters. |
| SETVAR | Provide write access to all ACSPL+ variables and to user variables declared with <i>tag</i> . |
| STR | Converts an integer array to a string. |
| STRTONUM | Converts an ASCII string representation of a number to the number it represents. |
| NUMTOSTR | Converts a number to an ASCII string. |
| BCOPY | Copies bytes from a source array to a target array. |
| SS1RESET | Resets the values of the last SS1-t channel A and channel B times |
| ENCREAD | Read encoder parameters |

4.3.1 GETCONF



GETCONF should be used only by knowledgeable users.

Description

GETCONF retrieves system configuration data that was configured by **SETCONF**.



Some keys relate to data that is set by the system and not by **SETCONF**, for keys set by **SETCONF** see **SETCONF** Arguments.

Syntax

GETCONF(key,index)

Arguments


| | |
|--------------|---------------------------------------------------------------|
| key | Specifies the configured feature. |
| index | Specifies the axis, buffer, or type of information requested. |

Return Value


GETCONF return values are described in [Table 4-1](#) according to key.

Table 4-1. GETCONF Return Values

| Key | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26 | <p>Returns the mask that determines, for each digital input, whether the leading or trailing signal edge triggers an interrupt.</p> <p>The mask contains a bit for each available input signal. The location of bits in the mask corresponds to the location of bits in variable INO.</p> <p>For each bit:</p> <p>1: The controller generates an interrupt on the falling edge of the corresponding input signal.</p> <p>0: Controller generates an interrupt on the rising edge of the corresponding input signal.</p> <p>After power-up, all bits in the mask = 0.</p> |
| 37 | <p>Returns the mask that determine whether a digital input triggers on a single edge, or on both edges. If value = 0, the trigger edge is determined by key 26.</p> <p>The location of bits in the mask corresponds to the location of bits in variable INO.</p> <p>1: The controller generates an interrupt on both edges.</p> <p>0: The controller generates an interrupt on one edge.</p> <p>After power-up the mask contains 0 in all bits.</p> |
| 71 | <p>Used to view the actual assignment of digital outputs to PEG states and PEG pulses outputs.</p> <p>Returns the bit code according to or for SPiiPlusNT/DC-LT/HP/LD-x, or for SPiiPlus CMnt-x-320/UDMpm-x-320, depending on the axis.</p> |
| 72 | <p>Used to view the actual encoder PEG engine assignment.</p> |

| Key | Value - Bit and Explanation |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Returns the bit code according to or for SPiiPlusNT/DC-LT/HP/LD-x, or for SPiiPlus CMnt-x-320/UDMpm-x-320, depending on the axis. |
| 73 | Used to view the actual output pins assignment for PEG engines. Returns the bit code according to or ,for SPiiPlusNT/DC-LT/HP/LD-x, or or for SPiiPlus CMnt-x-320/UDMpm-x-320, depending on the axis. |
| 74 | Returns the saved EtherCAT topology configuration Index = 1 : EtherCAT topology mode that is saved on the controller's non-volatile memory: 0: line topology mode 1: ring topology mode 2: two lines topology mode Index =2 : Number of nodes connected to the EtherCAT master's main line that is saved on the controller's non-volatile memory Index =3 : Number of nodes connected to the EtherCAT master's redundant line that is saved on the controller's non-volatile memory. |
| 76 | Returns the the System or MPU temperature (in degrees °C): 0: Current System temperature 1: Current temperature in controller's CPU/MPU 2: Current DSP temperature (IDMsm/sa, ECMsm/sa, UDMsa) |
| 78 | Returns the status if fast loading of Random PEG arrays for the relevant Servo Processor is activated or deactivated: 0: fast loading of Random PEG arrays is deactivated 1: fast loading of Random PEG arrays is activated. <div style="border: 1px solid black; border-radius: 10px; padding: 10px; margin-top: 10px;"> Index is the axis of the relevant Servo Processor: 0, 1, 2, ..., up to total number of axes in system minus 1.</div> |
| 79 | Returns the maximum USAGE value since power-up or since last call to the SETCONF(79) command. |
| 80 | Returns the UnitID of the network unit that the specified digital input is assigned to. Index = 0, 1, 2... up to total number of digital inputs in the system minus 1. |

| Key | Value - Bit and Explanation |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 81 | Returns the UnitID of the network unit that the specified digital output is assigned to. Index = 0, 1, 2... up to total number of digital outputs in the system minus 1. |
| 82 | Returns the UnitID of the network unit that the specified analog input is assigned to. Index = 0, 1, 2... up to total number of analog inputs in the system minus 1. |
| 83 | Returns the UnitID of the network unit that the specified analog output is assigned to. Index = 0, 1, 2... up to total number of analog outputs in the system minus 1. |
| 86 | Returns the number of allowed single EtherCAT frames that were actually lost. |
| 99 | Index = 0: Returns number of regular ACSPL+ program buffers. Index = 7: Returns total number of axes to which the controller is configured. Index = 8:Key Returns the maximum number of data bytes in the SAFE format message. |
| 203 | Returns the value of MFLAGS.1 (Open Loop) 1: Open loop 0: Not open loop |
| 204 | Returns the value of MFLAGS.9 (Commutation OK). 1: Commutation OK) 0: Commutation not OK) |
| 214 | Only valid for brushless motors (MFLAGS.8 = 1). Returns the commutation phase (degrees) at the current point. |
| 216 | Only valid for brushless motors (MFLAGS.8 = 1). Returns the commutation state (MFLAGS.9): 0: Commutation is not OK (not initialized) 1: Commutation is OK. |
| 229 | Returns the mechanical brake output: 1: Mechanical brake is inactive |

| Key | Value - Bit and Explanation |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | 0: Mechanical brake is active |
| 246 | <p>Upon receipt of a Drive Alarm signal, the controller stores a general Drive Alarm code (5019) in the MERR variable. The extended Drive Fault status code can be obtained by executing GETCONF(246, Axis).</p> <p>The following extended Drive Fault statuses are supported (the MERR code appears in brackets) by the DDM3U Motor Drive:</p> <ul style="list-style-type: none"> > Drive Alarm (5060) > Drive Alarm: Short circuit (5061) > Drive Alarm: External Protection Activated (5062) > Drive Alarm: Power Supply Too Low (5063) > Drive Alarm: Power supply too high (5064) > Drive Alarm: Temperature too high (5065) > Drive Alarm: Power Supply 24VF1 (5066) > Drive Alarm: Power Supply 24VF2 (5067) > Drive Alarm: Emergency Stop (5068) > Drive Alarm: Power down (5069) > Drive Alarm: Phase lost. (5070) > Drive Alarm: Drive not ready (5071) > Drive Alarm: Over current (5072) > Not in use (reserved) (5073) > Drive Alarm: Damper is not ok (5074) > Drive Alarm: Digital Drive Interface not Connected (5075) <p>Using the GETCONF function, the faults 5064, 5065, 5069, 5071 can be read before the ENABLE command is executed.</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;">  <p>The GETCONF function provides a delay until the extended Drive Fault is received. This behavior differs from the implementation in the SPiiPlus CM, where the extended Drive Fault status is stored in MERR immediately upon receipt of the Drive Alarm signal.</p> </div> |
| 253 | <p>Returns the state of the STO signals for the axis selected by index</p> <p>bit 0: ST01</p> <p>bit 1: ST02</p> |

| Key | Value - Bit and Explanation |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 262 | Returns the current Hall state, which can be 0, 1, 2, 3, 4, or 5, of the axis given by axis_def (a number: 0, 1, 2, ... up to the number of axes in the system minus 1). It returns -1 for invalid states. |
| 265 | <p>When a SIN-COS encoder is used, there are rare cases in which a homing repeatability error of 1 quadrant (quarter of a sine-period) may occur. This key is used for supporting SIN-COS repeatability. For example:</p> <pre> !!!Original homing procedure here ... TILL IST(AXIS).#IND; !Index was found ! Move to a middle of a quadrant, close to the index location PTP(AXIS), IND(AXIS) + POW(2, (E_SCMUL(AXIS)-3)) *EFAC(AXIS) TILL ^MST(AXIS).#MOVE WAIT 1000 ! Repeatability correction SET FPOS(AXIS) = FPOS(AXIS) - IND(AXIS) - GETCONF(265,AXIS) PTP(AXIS), 0 ! 0 = index location </pre> |
| 270 | <p>If an axis is enabled while moving, the motor back-EMF may generate high currents during the ENABLE process which can potentially damage the drive or the motor.</p> <p>To avoid such damage the controller should check the motor velocity during the ENABLE process and triggers a fault (error 5104 – “motor is moving”) if it is above a threshold. The threshold is proportional to SLCPRD (commutation period) for a brushless motor (MFLAGS(.8=1). It is proportional to XVEL(maximal velocity) for a DC-brush motor (MFLAGS(.8=0)</p> <p>Usually the user should not modify the factor, but in special specific cases it may need to be increased. A typical example where modification might be needed is a dual loop system with high resolution encoder on the load and low resolution encoder at the motor (used for commutation). The threshold may be multiplied by a factor using a special SETCONF command.</p> <ul style="list-style-type: none"> > SETCONF(270, <axis>, <value>) The value is – 1.0 by default. > GETCONF(270, <axis>) returns the current value of the factor. <p>The SETCONF command should be executed after each controller powerup.</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;">  <p>Because of the potential damage to the drive and the motor, the user is advised to set this value only after consulting the factory. Contact support@acsmotioncontrol.com.</p> </div> |
| 301 | Returns the size of the segment queue in segmented motions (MPTP...ENDS , MSEG...ENDS , PATH...ENDS , PVSPLINE...ENDS). |

| Key | Value - Bit and Explanation |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 310 | Returns an integer value that contains the TCP/IP address currently assigned to the controller. The index argument has to be zero, for example, GETCONF(310, 0) If a TCP/IP protocol is not configured, or not supported, the return value is zero. |
| 312 | Returns the RAM load in percentage, the amount of total physical memory, or the amount of free physical memory as: 0 – memory load in percentage 1 – amount of total physical memory 2 – amount of free physical memory |
| 318 | Returns the current mode of operation of move & settle: 0 – Feature is off, no sampling of settling times around radius 1 – Single mode, measure settling time up to first successful settle 2 – Auto mode, keep measuring even after first successful settle |
| 322 | Returns 1 if the channel is connected, 0 otherwise |

COM Library Methods

GetConf

C Library Functions

acsc_GetConf

Examples

Example 1:

```
?B/GETCONF (229, 0)
```

Returns the following mask:

```
00000000,00000000,00000000,00000001
```

Reports the actual state of the mechanical brake for the given axis. The output is presented in binary base (/B).

Example 2:

```
?X/GETCONF (229, 0)
```

Output:

```
00000001
```

Reports the actual state of the output pins of the mechanical brake for the given axis in hexadecimal format.

Example 3:

```
?X/GETCONF (310,0)
```

Output:

6e00000a

The address of a controller whose TCP/IP address is 10.0.0.110

4.3.2 SYSINFO

Description

SYSINFO retrieves a value related to the SPiiPlus controller system based on the argument that is specified.

Syntax

SYSINFO(*Int*)

Arguments

| | |
|------------|---------------------------------------------|
| <i>Int</i> | A positive integer ranging between 1 to 16. |
|------------|---------------------------------------------|

Return Value

Returns a value based on the specified *Int*.

The possible values of *Int* and the associated return values are given in [Table 4-2](#).

Table 4-2. SYSINFO Return Values

| Int | Value Returned |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | The SPiiPlus model number. For all EtherCAT products, returns 60. A returned value of ≤ 0 means that the connection is to the Simulator. Note: To get the product ID, use the function ECGETPID. |
| 2 | The SPiiPlus version number. |
| 10 | Number of regular ACSPL+ program buffers. |
| 11 | D-Buffer index. |
| 13 | Total number of axes in the current configuration, whether a single SPiiPlus controller or an EtherCAT network. |
| 14 | Number of EtherCAT nodes |
| 15 | Number of data collection channels per DSP. |
| 16 | EtherCAT support: |

| Int | Value Returned |
|-----|----------------|
| | 1 - Yes |
| | 0 - No |

COM Library Methods

None

C Library Functions

acsc_SysInfo

4.3.3 GETVAR

Description

GETVAR retrieves a value from a variable (ACSPL+ or user-defined variable, scalar or array) that was declared as a Tag number.

Syntax

GETVAR(*Tag*, [*Index1*, *Index2*])

Arguments

| | |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Tag | The variable Tag number (positive integer) |
| Index1, Index2 | If the variable is an array, the indexes point to the specific location in the array. If the variable is scalar, omit the indexes. |

Return Value

Returns the value of the variable specified by the **Tag**.

Example

```
GLOBAL REAL TAG 1001 EE(2) (2)
                                !Defines user variable array EE as Tag 1001.
SETVAR (15,1001,1,1)           !Sets value 15 to user variable array
                                !described in Tag 1001 cell (1)(1).
DISP GETVAR (1001,1,1)         !Display the value in user variable array
                                !designated by Tag 1001 cell (1)(1).
STOP                           !Ends program
```

The controller displays the return value of the **GETVAR** function which is = 15.

4.3.4 SETCONF



SETCONF should be used only by knowledgeable users.

Description

SETCONF defines system configuration data.

All the keys that can be set by **SETCONF** listed in [SETCONF Arguments](#) can also be retrieved by [GETCONF](#).

Syntax

SETCONF(*key,index,value*)

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| key | Specifies the configured feature. |
| index | Specifies axis or buffer number. |
| value | <p>The set of bit states for the defined key. <i>value</i> is set up according to a 16-bit binary template, illustrated in 16-bit Binary value Template. The controller strips all leading zeros. The controller understands <i>value</i> in binary, hexadecimal, or decimal format.</p> <p>The following prefixes determine <i>value</i> format:</p> <p>OB - binary</p> <p>OH - hexadecimal</p> <p>A decimal value does not require any prefix.</p> |

Table 4-3. 16-bit Binary *value* Template


| | | | | | | | | | | | | | |
|-----------------------------------------------------------|---|---|---|---|---|----|---|---|---|---|---|---|---|
| Bit | 3 | 3 | 2 | 2 | 2 | .. | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 1 | 0 | 9 | 8 | 7 | . | | | | | | | |
| Pre fix [O B] [O H] <i>val ue</i> | 0 | 0 | 0 | 0 | 0 | .. | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SETCONF arguments are detailed in [Table 4-4](#).

Table 4-4. SETCONF Arguments

| Key | Index | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26 | Don't Care | <p>Sets the mask that determines, for each digital input, whether the leading or trailing signal edge triggers an interrupt.</p> <p>The mask contains a bit for each available input signal. The location of bits in the mask corresponds to the location of bits in variable INO.</p> <p>For each bit -</p> <p>1: The controller generates an interrupt on the falling edge of the corresponding input signal.</p> <p>0: Controller generates an interrupt on the rising edge of the corresponding input signal.</p> <p>After power-up, all bits in the mask = 0.</p> |
| 37 | | <p>Sets the mask that determine whether a digital input triggers on a single edge, or on both edges. If value = 0, the trigger edge is determined by key 26.</p> <p>The location of bits in the mask corresponds to the location of bits in variable INO.</p> <p>1: The controller generates an interrupt on both edges.</p> <p>0: The controller generates an interrupt on one edge.</p> <p>After power-up the mask contains 0 in all bits.</p> |
| 79 | | <p>Clears the stored maximum usage value. This value is the maximum system usage since power-up or a previous call to SETCONF(79). See GETCONF(79).</p> |
| 86 | | <p>Resets the counter of allowed single EtherCAT frames that were actually lost</p> |
| 203 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Sets the value of MFLAGS.1 (Open Loop)</p> <p>1: Open loop</p> <p>0: Not open loop</p> |
| 204 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Sets the value of MFLAGS.9 (Commutation OK).</p> <p>1: Commutation OK</p> |


| Key | Index | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | 0: Commutation not OK |
| 214 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Valid only for brushless motors (MFLAGS.8 = 1).</p> <p>SETCONF with the 214 key sets the commutation offset at the current position to the specified <i>value</i> (in degrees).</p> <p>There are, however, two cases that have to be considered when using SETCONF:</p> <ul style="list-style-type: none"> > Motor Not Commutated (MFLAGS.9=0) If the motor has not yet been commutated, SETCONF sets the commutation offset at the current position to the value specified by the <i>value</i> argument. > Motor Commutated (MFLAGS.9=1) The behavior of SETCONF for commutated motors in SPiiPlus NT controllers is different from that in non-NT SPiiPlus controllers. For NT controllers once a motor is commutated, the commutation phase has an additional 90°. SETCONF modifies the commutation phase prior to this 90° addition. Therefore, to change the commutation phase of a commutated motor, it is recommended that the user enter <code>?GETCONF(214,axis)</code> in the Communication Terminal and subtract 90 from the returned value. From this the user can calculate what the value of the <i>value</i> argument should be to get the proper phase. |
| 216 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Valid only for brushless motors (MFLAGS.8 = 1).</p> <p>If value = 1:</p> <ul style="list-style-type: none"> > MFLAGS.9=0 (Commutation is not OK) > MFLAGS bits 1, 4, 5, 6 are set to zero > MFLAGS.8 is set to 1 > DCOM is reset to zero > Sets RPOS = FPOS |

| Key | Index | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <p>If value = 0:</p> <ul style="list-style-type: none"> > MFLAGS.9=0 (Commutation is not OK) > MFLAGS bits 1, 4, 5, 6 are set to zero > MFLAGS.8 is set to 10 > Variable DCOM is reset to zero. |
| 217 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Valid only for brushless motors (MFLAGS.8 = 1) where the encoder has encountered the index and IND contains a valid value.</p> <p>value: Not Relevant.</p> <ul style="list-style-type: none"> > Adjusts the commutation offset so that the commutation phase at the last index point is equal to the specified value in degrees. > MFLAGS.9=0 (Commutation is not OK) > MFLAGS bits 1, 4, 5, 6 are set to zero > MFLAGS.8 is set to 1 > DCOM is reset to zero > Sets RPOS = FPOS |
| 229 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Mechanical brake output:</p> <p>1: Deactivates mechanical brake output. 0: Activates mechanical brake output. The motor must be disabled to execute this setting.</p> |
| 246 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>SETCONF(246, Axis, 0) is used to clear the fault status on all axes that relate to the DDM3U Motor Drive that handles the specified axis.</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;">  <p>The FCLEAR command does not clear the fault status in the MC4U, SETCONF(246, Axis, 0) has to be used instead.</p> </div> |

| Key | Index | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 249 | Axis: Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>There are situations where automatic current bias measurement (in ACS control modules) can lead to problems. For example, when running an air bearing stage, the automatic current offset measurement may be inconsistent with every enable due to air bearing stage movement or drift during enable. Thus any small offset can create a relatively large oscillation during constant velocity.</p> <p>SETCONF(249, Axis, 0) disables the automatic current bias measurement for the given axis.</p> |

| Key | Index | Value - Bit and Explanation |
|-----|------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 267 | 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>Changes the gantry pair's allocation to all axes within the Servo Processor that the specified axis belongs to.</p> <p>Gantry pair's allocation is done according to the following:</p> <p>0: for pairs (0,1) and (2,3)</p> <p>1: for pairs (0,2) and (1,3)</p> <p>For example:</p> <p>4 axes MC4U system</p> <p style="padding-left: 40px;">Command for setting pairs (0,1) and (2,3) - SETCONF(267,0,0)</p> <p style="padding-left: 40px;">Command for setting pairs (0,2) and (1,3) - SETCONF(267,0,1)</p> <p>8-axes MC4U system</p> <p style="padding-left: 40px;">Command for setting pairs (0,1) and (2,3) - SETCONF(267,0,0)</p> <p style="padding-left: 40px;">Command for setting pairs (0,2) and (1,3) - SETCONF(267,0,1)</p> <p style="padding-left: 40px;">Command for setting pairs (4,5) and (6,7) - SETCONF(267,4,0)</p> <p style="padding-left: 40px;">Command for setting pairs (4,6) and (5,7) - SETCONF(267,4,1)</p> <p>Two 8-axes MC4U systems connected in the network</p> <p style="padding-left: 40px;">Command for setting pairs (0,1) and (2,3) - SETCONF(267,0,0)</p> <p style="padding-left: 40px;">Command for setting pairs (0,2) and (1,3) - SETCONF(267,0,1)</p> <p style="padding-left: 40px;">Command for setting pairs (4,5) and (6,7) - SETCONF(267,4,0)</p> <p style="padding-left: 40px;">Command for setting pairs (4,6) and (5,7) - SETCONF(267,4,1)</p> <p style="padding-left: 40px;">Command for setting pairs (8,9) and (10,11) - SETCONF(267,8,0)</p> |

| Key | Index | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <p>Command for setting pairs (8,10) and (9,11) - SETCONF(267,8,1)</p> <p>Command for setting pairs (12,13) and (14,15) - SETCONF(267,12,0)</p> <p>Command for setting pairs (12,14) and (13,15) - SETCONF(267,12,1)</p> |
| 270 | Axis: 0, 1, 2, ..., up to total number of axes in system minus 1 | <p>If an axis is enabled while moving, the motor back-EMF may generate high currents during the ENABLE process which can potentially damage the drive or the motor.</p> <p>To avoid such damage the controller should check the motor velocity during the ENABLE process and triggers a fault (error 5104 – “motor is moving”) if it is above a threshold. The threshold is proportional to SLCPRD (commutation period) for a brushless motor (MFLAGS().8=1). It is proportional to XVEL (maximal velocity) for a DC-brush motor (MFLAGS().8=0)</p> <p>Usually the user should not modify the factor, but in special specific cases it may need to be increased. A typical example where modification might be needed is a dual loop system with high resolution encoder on the load and low resolution encoder at the motor (used for commutation). The threshold may be multiplied by a factor using a special SETCONF command.</p> <ul style="list-style-type: none"> > SETCONF(270, <axis>, <value>) The value is – 1.0 by default. > GETCONF(270, <axis>) returns the current value of the factor. <p>The SETCONF command should be executed after each controller powerup.</p> |

| Key | Index | Value - Bit and Explanation |
|-----|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <div style="border: 1px solid black; padding: 10px;">  <p>Because of the potential damage to the drive and the motor, the user is advised to set this value only after consulting the factory. Contact support@acsmotioncontrol.com.</p> </div> |
| 302 | 2 1 | <p>The following decimal values specify a communication channel for special input:</p> <ul style="list-style-type: none"> 3: Set the channel to Modbus Master mode. 2: Set the channel to Modbus Slave mode 1: Assigns the channel for special input. 0: Set the channel to regular command processing mode (default channel mode). <p>If a channel is assigned for special input, the controller does not process commands from this channel. Output to the channel is provided by regular DISP and SEND commands.</p> <p>index specifies the channel.</p> |
| 303 | 2 1 | <p>The value sets the baud rate for the specified serial channel, where the baud rate is the decimal value.</p> <p>115200 (default), 57600, 19200, 9600, 4800, 2400, 1200, 600, 300.</p> <p>index specifies the channel.</p> |

| Key | Index | Value - Bit and Explanation |
|-----|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 304 | 2 1 | <p>Sets communication options:</p> <p>Bit 2 -</p> <p>1: extended stop bit 0: normal stop bit</p> <p>Bit 3 -</p> <p>1: check parity 0: no parity</p> <p>Bit 4 -</p> <p>1: even parity 0: odd parity</p> <p>index specifies the channel.</p> |
| 306 | -1 | <p>In order to be able to receive messages, use specific channels (SEND and DISP commands)</p> <p>1 - Enable receiving messages on a specific channel 0 - Stop receiving messages from a host application</p> |
| 308 | 19 18 17 | <p>When the controller acts as a Modbus Master, the function establishes or closes the Modbus TCP connection with the Slave device using the specified slaveIP. Up to three slaves can be connected to the master controller (17, 18 or 19). Value=SlaveIP.</p> <p>The slaveIP value specifies the Modbus Slave device IP address. The slaveIP address is calculated as follows: If the Slave has the following address: 192.168.1.10, the slaveIP parameter should be $10*2^{24} + 1*2^{16} + 168*2^8 + 192 = 167880896$ (0x0A01A8C0).</p> <p>If the specified channel is already open, the function closes the opened channel and then opens new one.</p> <p>If slaveIP is zero, the function closes the TCP connection.</p> |
| 309 | Don't Care | <p>Defines the sequence for the two 16-bit Modbus interface registers -</p> |

| Key | Index | Value - Bit and Explanation |
|-----|-------|-------------------------------------------------------------------------------------------------------------------|
| | | 1: Low word will be first, then high word 0: High word will be first, then low word (default configuration) |

| Key | Index | Value - Bit and Explanation |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 310 | 0 | <p>Used for providing access to the controller TCP/IP address. Where:</p> <p>Value = TCP/IP Address, which is a 32-bit (four bytes) integer, each byte of which contains one part of the TCP/IP address, for example, 0x6400000a assigns address 10.0.0.100 (the number is read from right to left and fills in the address left to right).</p> <p>If value is zero, SETCONF activates a new execution of the DHCP protocol and obtains a new TCP/IP address from the host (the host may configure the same address as before). SETCONF does not change the TCPIP variable. After power-up, the controller is initialized with the TCP/IP address set in the TCPIP variable.</p> <p>There are several limitations when using SETCONF(310):</p> <ul style="list-style-type: none"> > If the TCPIP variable stored in the flash is zero, SETCONF(310) must be used only with zero address argument. In other words, if the controller is configured for dynamic addressing, assigning static addresses is not allowed. > If the TCPIP variable stored in the flash is not zero, SETCONF(310) must be used only with non-zero address arguments. In other words, if the controller is configured for static addressing, switching to a dynamic address is not allowed. > SETCONF(310) has a long execution time. During this time, communication with the controller is impossible using any communication channel. Use SETCONF(310) only within the controller initialization sequence. Avoid attempts to communicate with the controller and the motor ENABLE command or motion commands while SETCONF(310) is in progress. |

| Key | Index | Value - Bit and Explanation |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 318 | Axis: 0, 1, 2, ..., up to total number of axis in system minus 1 | <p>Move & Settle:</p> <p>0 – Feature is off, no sampling of settling times around radius</p> <p>1 – Single mode, measure settling time up to first successful settle</p> <p>2 – Auto mode, keep measuring even after first successful settle.</p> |
| 322 | <ul style="list-style-type: none"> > TCP Server Communication Port > Communication Channel (channels 26-29 are reserved for this task, no other channels may be used) <p>Index value calculated as follows:</p> $\text{Index} = \text{Port} * 100 + \text{Channel}$ | <p>The value is the TCP Server IP Address</p> <p>The IP address is calculated as follows: If the TCP Server has the following address: 192.168.1.10, the IP parameter should be $10 * 2^{24} + 1 * 2^{16} + 168 * 2^8 + 192 = 167880896$ (0x0A01A8C0).</p> <p>If the specified channel is already open, the function closes the opened channel and then opens new one.</p> <p>If an IP is zero, the function closes the TCP connection.</p> |
| 325 | 0 | <p>Range of Values: 0-400</p> <ul style="list-style-type: none"> > 0 – Matrix related operations will not be performed in real-time. > 400 - Operations on matrices with up to 400 elements (e.g., 20x20) will be performed in real-time. Operations on matrices larger than 400 are not allowed to run in real-time due to critical impact on MPU usage. |

Return Value

None

COM Library Methods

SetConf

C Library Functions

acsc_SetConf

Example

The following example illustrates setting the value of MFLAGS.1 to 1, configuring axis 2 to open loop control:

```
SETCONF(203, 1, 1)
```

4.3.5 SETVAR

Description

SETVAR writes a value to an ACSPL+ or user variable, scalar or array, that was declared as a Tag number.

Syntax

SETVAR(*value*, *Tag*, [*Index1*, *Index2*])

Arguments

| | |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| value | A real or integer value assigned to the variable. |
| Tag | The variable Tag number (positive integer). |
| Index1, Index2 | If the variable is an array, the indexes point to the specific location in the array. If the variable is scalar , omit the indexes. |

Return Value

None

Error Conditions

None

Example

```
GLOBAL REAL TAG 1001 EE(2)(2)    !Defines user variable array EE as Tag 1001
SETVAR (15,1001,1,1)           !Sets value 15 to user variable array
                                !described in Tag 1001 cell (1)(1).
DISP GETVAR (1001,1,1)         !Retrieves the value in user variable array
                                !described in Tag 1001 cell (1)(1).
STOP                            !Ends program
```

The controller displays the **GETVAR** return value which is = 15.

4.3.6 STR

Description

STR converts an integer array to a string. Each element of the array is interpreted as an ASCII character.

Syntax

string **STR**(*array_name*, [*start_index*,] [*number*])

Arguments

| | |
|--------------------|----------------------------------------------------|
| <i>array_name</i> | Name of user-defined integer array. |
| <i>start_index</i> | Index in the array from which to start converting. |
| <i>number</i> | Number of characters to convert. |

Comments

If an element value is in the range from 0 to 255, it is directly converted to the corresponding ASCII character. Otherwise, the value will be cyclic, based on 256.

If **start_index** is omitted, the assignment starts from the first element of the array.

If neither **start_index** nor **number** is specified, the conversion takes all elements of the array. If only **start_index** is specified, the conversion takes all characters from the specified index until the end of the array. **number** limits the number of characters in the resulting string.

Return Value

String composed of the array elements interpreted as characters.

Example

```
GLOBAL INT BIBI (3)
BIBI (0) =65
BIBI (1) =67
BIBI (2) =83
DISP STR (BIBI)
STOP
```

The function transforms each of the array members of BIBI to ASCII code characters. When **DISP** is applied, the displayed value will be: "ACS"

4.3.7 STRTONUM**Description**

This function converts ASCII encoded element string to a number.

Syntax

double **STRTONUM**(*Source, Type, From, N*)

Arguments

| | |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Source</i> | An integer array |
| <i>Type</i> | Designates the format of the converted number, it can be: <ul style="list-style-type: none"> > 0 - decimal > 1- hex > 2 - floating point |

| | |
|-------------|---------------------------------------------------------------------|
| From | Index of the first element in the array that may contain the number |
| N | The number of elements in source that may be used |

Return Value

The number converted from the ASCII string

Example

```

real number
int string(4);
number = 0
string(0) = 49; !1
string(1) = 50; !2
string(2) = 46; !.
string(3) = 50; !2
number = strtonum(string, 0, 0, 4);
disp"number = %f", number;
number = strtonum(string, 1, 0, 4);
disp"number = %f", number;
number = strtonum(string, 2, 0, 4);
disp"number = %f", number;
stop
! output:
! number = 12.000000
! number = 18.000000
! number = 12.200000
    
```

4.3.8 NUMTOSTR

Description

This function converts a number to a string of elements where each element is an ASCII encoded value.

Syntax

int NUMTOSTR(*Number, Target, Type, From, N*)

Arguments

| | |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number | The number to be converted |
| Target | An integer array used in which to put the results |
| Type | Designates the format of the number to be converted, it can be: <ul style="list-style-type: none"> > 0 - decimal > 1- hex > 2 - floating point |

| | |
|-------------|---------------------------------------------------------------------|
| From | Index of the first element in the array that may contain the number |
| N | The number of elements in target that may be used |

Return Value

The number of elements used.

Example

```

real number
int target(12);
int i;
int j;
number = 12.2
i=0; loop 12 target(i) = 0; i = i+1; end;
j = numtostr(number,target,0,0,12)
disp "target = %X,%X,%X,%X,%X num elements = %d", target(0),target
(1),target(2),target(3),target(4), j
i=0; loop 12 target(i) = 0; i = i+1; end;
j = numtostr(number,target,1,0,12)
disp "target = %X,%X,%X,%X,%X num elements = %d", target(0),target
(1),target(2),target(3),target(4), j
i=0; loop 12 target(i) = 0; i = i+1; end;
j = numtostr(number,target,2,0,12)
disp "target = %X,%X,%X,%X,%X num elements = %d", target(0),target
(1),target(2),target(3),target(4), j
stop
! output:
! target = 31,32,0,0,0 num elements = 2
! target = 63,0,0,0,0 num elements = 1
! target = 31,32,2E,32,30 num elements = 9

```

4.3.9 BCOPY**Description**

This function can be used to copy bytes from the source array to the target array.

Syntax

int BCOPY(Source_array, Target_array, CopyBytes, S_SkipBytes, T_SkipBytes, From, N)

Arguments

| | |
|---------------------|----------------------------------------------------------------------------------------------------|
| Source_array | Array containing input data |
| Target_array | Array containing output data |
| CopyBytes | Number specifying how many bytes will each copy operation |
| S_SkipBytes | Number specifying how many bytes will be skipped in the source array following each copy operation |

| | |
|--------------------|----------------------------------------------------------------------------------------------------|
| T_SkipBytes | Number specifying how many bytes will be skipped in the target array following each copy operation |
| From | Index of the first element in the source array that may be used |
| N | The number of elements in the source that may be used |

Return Value

The number of elements (in the source) used.

Comments

The function copies **CopyBytes** from the source array to the target array, skip **S_SkipBytes** in the source and skip **T_SkipBytes** in the target, and then copy **CopyBytes** again. The operation starts from the **From** element in source and is applied to a maximum of **N** elements. This means no bytes from any element other than the specified **N** elements will be affected. Elements less than **N** can be affected if the target array is too short to complete the whole operation.

The difference between elements and bytes throughout this function should be noted.

Examples

In the examples that follow use is made of:

`bcopy(source,target,1,3,0)` - Copy every 4 byte element from source to a 1 byte element in target

`bcopy(source,target,2,2,0)` - Copy every 4 byte element from source to a 2 byte element in target

`bcopy(source,target,1,0,3)` - Copy every 1 byte element from source to a 4 byte element in target

`bcopy(source,target,2,0,2)` - Copy every 2 byte element from source to a 4 byte element in target

1. Unpacking a single element in source to 1, 2, or 4 elements in target

```
int source(1)
int target(4)
int j
source(0) = 0x01020304;
j = bcopy(source,target,4,0,0,0,1) => target = 1020304,0,0,0 num elements = 1
j = bcopy(source,target,2,0,2,0,1) => target = 304,102,0,0 num elements = 1
j = bcopy(source,target,1,0,3,0,1) => target = 4,3,2,1 num elements = 1

stop
```

2. Packing multiple elements in source to 1 element in target.

```
int source(4)
int target(1)
int i
int j
target(0) = 0;
i = 0
loop 4
```

```

source(i) = 1;
i=i+1;
end
j = bcopy(source,target,4,0,0,0,4) => target(0) = 1 j = 1
j = bcopy(source,target,2,2,0,0,4) => target(0) = 10001 j = 2
j = bcopy(source,target,1,3,0,0,4) => target(0) = 1010101 j = 4
stop
    
```

4.3.10 SS1RESET

Description

Resets the values of the last SS1-t channel A and channel B times.



This function should be used for SS1-t diagnostics only.

Syntax

SS1RESET [/f] axis

Arguments

| | |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. Axis parameter can be any axis number of the same unit. |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|

Comments

This variable is supported in version 3.00 and higher

Return Value

None

4.3.11 MDURATION

Description

The **MDURATION** function calculates time of certain motion profile phase. The function can be executed in buffer or terminal, and it will wait till the execution is completed.

Syntax

```

REAL MDURATION(MotionType, Distance, Vel, Acc, Dec, Jerk, phaseNumber
(optional), EncoderFactor(optional))
    
```

Arguments

| | |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MotionType | An integer number representing the motion type 0: PTP motion Currently only PTP motion is supported |
| Distance | A real number representing the distance of motion. The velocity in [user units] |
| Vel | A real number representing the desired velocity. The velocity in . Velocity ranges from -1.79769e+308 to 1.79769e+308. |
| Acc | A real number representing the desired acceleration. The acceleration in . Acceleration ranges from 2.22507e-308 to 1.79769e+308. |
| Dec | A real number representing the desired deceleration. The acceleration in . Deceleration ranges from 2.22507e-308 to 1.79769e+308. |
| Jerk | A real number representing the desired jerk. The jerk in . Jerk ranges from 2.22507e-308 to 1.79769e+308. |
| phaseNumber | (Optional, Integer) Determines which motion profile component will be returned. 0: t – Motion overall time [sec] – default 1: t1 – Acceleration buildup [sec] 2: t2 – Constant Acceleration [sec] 3: t3 – Acceleration finishing [sec] 4: t4 – Constant velocity [sec] 5: t5 – Deceleration buildup [sec] 6: t6 – Constant deceleration [sec] 7: t7 – Deceleration finishing [sec] |
| EncoderFactor | A real representing the factor between the raw feedback in encoder counts and the FPOS value calculated by the controller. EncoderFactor ranges between 1e-15 to 1e+15, Default = 1. |

Return Value

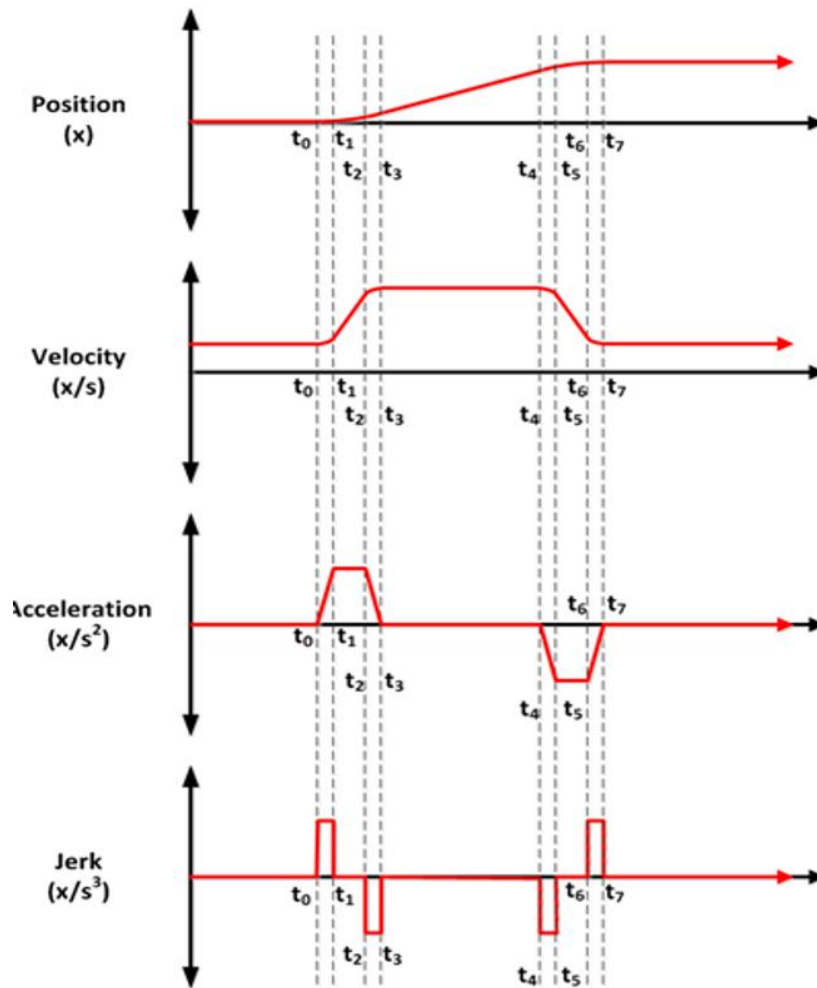
Returns motion profile time according to `phaseNumber`.

Comments

Assume that the motion will start and complete when velocity and acceleration are 0.

Input Shaping is not supported

`phaseNumber` options corresponds to a third order point-to-point profile:



Assume that the motion will start and complete when velocity and acceleration are 0.
 Input Shaping is not supported.

Error Conditions

The function detects the following error conditions.

- > **3207** - phaseNumber parameter is not between 0 to 7
- > **3412** - The MotionType parameter is intended to be one of supported motion types. Currently only PTP motion is supported.
- > **3045** - At least one of struct members is infinite.
- > **3041** - Vel, Acc, Dec, Jerk, Or EncoderFactor is out of range.

Examples

```
! Example 1
! MotionType = 0 - PTP
! Distance = 20000, Vel= 10000, Acc = 100000, Dec = 100000, Jerk =
20000000
! phaseNumber = default = 0 - Motion overall time in seconds
```

```
! EncoderFactor = default = 1
V0 = MDURATION(0, 20000, 10000, 100000, 100000, 20000000)
! Return value: Motion overall time V0 = 2.105[sec]
STOP
```

```
! Example 2
! MotionType = 0 - PTP
! Distance = 20000, Vel= 10000, Acc = 100000, Dec = 100000, Jerk =
20000000
! phaseNumber = 4 - Constant velocity
! EncoderFactor = default = 1
V0 = MDURATION(0, 20000, 10000, 100000, 100000, 20000000, 4)
! Return value: Constant velocity V0 = 1.895[sec]
STOP
```

```
! Example 3
! MotionType = 0 - PTP
! Distance = 20000, Vel= 10000, Acc = 100000, Dec = 100000, Jerk =
20000000
! phaseNumber = 4 - Constant velocity
! EncoderFactor = 2
V0 = MDURATION(0, 20000, 10000, 100000, 100000, 20000000, 4, 2)
! Return value: Constant velocity V0 = 1.895[sec]
STOP
```

4.4 Array Processing Functions

The Array Processing functions are:

| Function | Description |
|----------|-----------------------------------------------------------------------------------------------------|
| AVG | Finds the average of all values in an array. |
| COPY | Copies data from one user array to another. |
| DSHIFT | Shifts all of the elements of the array to one position left. |
| FILL | Fills an array or a section of array with the specified value. |
| MAX | Finds the maximal value in an array |
| MAXI | Finds the maximal value in an array or in a section of array and returns its index. |
| MIN | Finds the minimal value in an array. |
| MINI | Finds the element with minimal value in an array or in a section of an array and returns its index. |

| Function | Description |
|---------------------|----------------------------------------------------------------|
| <code>SIZEOF</code> | Returns number of columns or number of rows of the given array |

4.4.1 *AVG*

Description

AVG finds the average of all values in an array.

Syntax

AVG(*array_name*)

Arguments

| | |
|-------------------|-------------------------------------------------------------|
| <i>array_name</i> | The name of an array that has been declared in the program. |
|-------------------|-------------------------------------------------------------|

Return Value

Real number - returns the average of all elements in **array_name**.

Example

```
REAL Ar (3)
Ar (0) = 1; Ar (1)=0.5; Ar (2)=3;
DISP AVG (Ar)
!Output = 1.5
```

4.4.2 *COPY*

Description

COPY copies data from one user array to another.

Syntax

COPY(*source, destination, from_source_row, to_source_row, from_source_col, to_source_col, from_destination_row, to_destination_row, from_destination_col, to_destination_col*)

Arguments

| | |
|------------------------|-------------------------------------------------------------------------------------------------|
| <i>source</i> | The name of an array that has been declared in the program from which the data is to be copied. |
| <i>destination</i> | The name of an array that has been declared in the program to which the data is to be copied. |
| <i>from_source_row</i> | The index of the first row of the source to begin copying. |
| <i>to_source_row</i> | The index of the last row of the source to end copying. |

| | |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <i>from_source_col</i> | The index of the first column of the source to begin copying. Used only for matrix type arrays, otherwise it can be omitted. |
| <i>to_source_col</i> | The index of the last column of the source to end copying. Used only for matrix type arrays, otherwise it can be omitted. |
| <i>from_destination_row</i> | The index of the first row of the destination to begin copying into. |
| <i>to_destination_row</i> | The index of the last row of the destination to end copying into. |
| <i>from_destination_col</i> | The index of the first column of the destination to begin copying into. Used only for matrix type arrays, otherwise it can be omitted. |
| <i>to_destination_col</i> | The index of the last column of the destination to begin copying into. Used only for matrix type arrays, otherwise it can be omitted. |

Comments

If the matrix indexes are omitted, the entire source matrix will be copied to the destination matrix.

If the destination matrix has different dimensions than the source matrix then the destination matrix will use the source matrix values to fill each row completely and move to the next row.

Return Value

None

Error Conditions

Error 3034, Illegal index value - the destination matrix is smaller than the source matrix.

Example

```

INT GLOBAL SOURCE(3)(3), DESTINATION(3)(3)      !Define source and destination
                                                !matrices
!----- Assign values to SOURCE matrix -----
-
SOURCE(0)(0)=1;SOURCE(0)(1)=2;SOURCE(0)(2)=3;SOURCE(1)(0)=4;SOURCE(1)
(1)=5;
SOURCE(1)(2)=6;SOURCE(2)(0)=7;SOURCE(2)(1)=8;SOURCE(2)(2)=9
!----- Assign values to DESTINATION matrix -----
----
COPY(SOURCE,DESTINATION,0,1,0,2,1,2,0,2)      !COPY command -
                                                !The program copies the first
                                                !two rows from the source
                                                !matrix to the destination
                                                !matrix second two rows. The
                                                !matrices are as follows:
                                                !Source:
                                                !1 2 3

```

```
STOP
```

```
!4 5 6
!7 8 9
!Destination:
!0 0 0
!1 2 3
!4 5 6
!Ends program
```

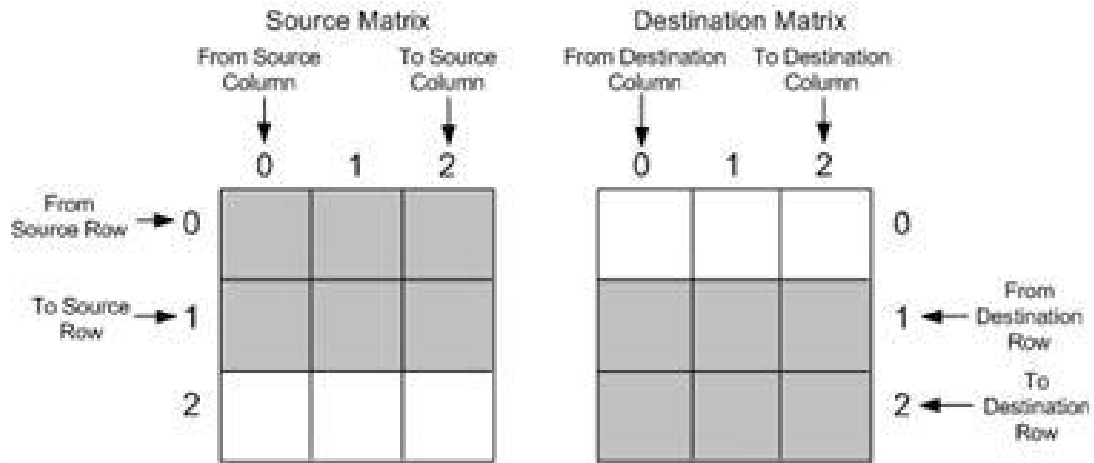


Figure 4-1. Illustration of COPY Function

4.4.3 DSHIFT

Description

DSHIFT shifts all elements of the array to one position left.

Syntax

real **DSHIFT**(*array, value, index*)

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>array</i> | An array of real with the maximal size of 10,000 elements |
| <i>value</i> | The real value to be inserted to the array. The Value is inserted to position of the element with the index Index. |
| <i>index</i> | The position of the element the Value is inserted to. The Index should be equal or less than declared size of the array |

Return Value

The first element (element with index 0) of the array.

Comments

Each time the function is called the first element of the array (element with the index 0) is returned. All of the other elements of the array shift one position to the left (element with index 1 to 0, element with index 2 to 1, etc.). The Value parameter is inserted to the element with the index Index.

This function only works with arrays with up to 10,000 elements.

The function is useful when it is necessary to delay data for a number of the controller cycles.

Example 1

Example 1 shows how to delay the result of the motion trajectory generator.

```
global real Array(20)    ! Array for delay
global int Axis         ! Axis index
Axis=0\
fill(APOS(Axis), Array) ! FILL array with initial APOS value
MFLAGS(Axis).#DEFCON=0 ! Motion trajectory delay is implemented
                        ! using CONNECT function
! Motion trajectory is delayed for 10 controller cycles
CONNECT RPOS(Axis) = dshift(Array, APOS(Axis),8)
DEPENDS Axis, Axis
stop
```

Example 2

The example 2 shows how to delay the pulses of laser generator. In this example it's assumed that the laser control module generates pulses as function of vector velocity of the axis. The PFGPAR parameter is used to transfer a vector velocity value to the laser control module.

```
global real Array(20)    ! Array for delay
global int Axis         ! Axis index
Axis=0
fill(0, Array)         ! FILL array with zero values
! Laser control module pulses are delayed for 10 controller cycles
while (1); PFGPAR(Axis) = dshift(Array, GVEL(Axis), 9); end;
stop
```

4.4.4 FILL

Description

FILL fills an array or a section of array with a specified value.

Syntax

FILL (*real value, destination, from_array_row, to_array_row, from_array_column, to_array_column*)

Arguments

| | |
|-----------------------|------------------------------------------------------------------------------------------------|
| <i>real value</i> | Any real number. |
| <i>destination</i> | The name of an array that has been declared in the program to which the value is to be copied. |
| <i>from_array_row</i> | The index of the first row of the destination array to begin filling. |
| <i>to_array_row</i> | The index of the last row of the destination array to end filling. |

| | |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <i>from_array_column</i> | The index of the first column of the destination array to begin filling. Used only for matrix type arrays, otherwise it can be omitted. |
| <i>to_array_column</i> | The index of the last column of the destination array to end filling. Used only for matrix type arrays, otherwise it can be omitted. |

Comments

If the matrix indexes are omitted, the entire matrix will be filled with the requested value.

Example

```
GLOBAL ARR(6)(3)
FILL(3,ARR,0,4,1,2)
STOP

!The program fills ARR with the value 3, from row 0 to row 4,
!and from column 1 to column 2. After executing the program,
!the resulting ARR values are:
!0      3      3
!0      3      3
!0      3      3
!0      3      3
!0      3      3
!0      0      0
```

4.4.5 MAX**Description**

MAX finds the maximum value in an array or in a section of an array.

Syntax

MAX(array_name, From1, To1, From2, To2)

Arguments

| | |
|--------------------------|-------------------------------------------------------------|
| <i>array_name</i> | The name of an array that has been declared in the program. |
| From1 | The initial element |
| To1 | The final element |
| From2 | The initial element |
| To2 | The final element |

Return Value

Real number - returns the maximum element in the array.

Example

```
REAL AR(3)
AR(0) = 1; AR(1)=0.5; AR(2)=3;
DISP MAX(AR) !Output = 3
```

4.4.6 MAXI

Description

MAXI finds the maximum value in an array or in a section of array and returns its index.

Syntax

MAXI(*array_name*, *From1*, *To1*, *From2*, *To2*)

Arguments

| | |
|-------------------|-------------------------------------------------------------|
| <i>array_name</i> | The name of an array that has been declared in the program. |
| From1 | The initial element |
| To1 | The final element |
| From2 | The initial element |
| To2 | The final element |

Return Value

Integer - **MAXI** returns the index of maximum element in the array, or in the specified section of the array. In case of a two-dimensional array only the column index is returned.

Error Conditions

None

Example

```
REAL AR(3)
AR(0)= 1; AR(1)= 0.5; AR(2)= 3
DISP MAXI(AR) !Output = 2
```

4.4.7 MIN

Description

MIN finds the minimum value in an array or in a section of an array.

Syntax

MIN(*array_name*, *From1*, *To1*, *From2*, *To2*)

Arguments

| | |
|-------------------|-------------------------------------------------------------|
| <i>array_name</i> | The name of an array that has been declared in the program. |
|-------------------|-------------------------------------------------------------|

Return Value

Real number - returns the minimum element in the array.

Error Conditions

None

Example

```
REAL Ar(3)
Ar(0) = 1; AR(1)=0.5; Ar(2)=3;
DISP MIN(Ar) !Output = 0.5
```

4.4.8 MINI**Description**

MINI finds the element with the minimum value in an array or in a section of an array and returns its index.

Syntax

MINI(array_name, From1, To1, From2, To2)

Arguments

| | |
|-------------------|-------------------------------------------------------------|
| <i>array_name</i> | The name of an array that has been declared in the program. |
| From1 | The initial element |
| To1 | The final element |
| From2 | The initial element |
| To2 | The final element |

Return Value

Integer - **MINI** returns the index of the minimum element in array X, or in the specified section of array x. In case of a two-dimensional array, only the column index is returned.

Example

```
REAL AR(3)
AR(0)= 1; AR(1)= 0.5; AR(2)= 3
DISP MINI(AR) !Output = 1
```

4.4.9 SIZEOF**Description**

The function returns number of columns or number of rows of the given array (can be user-defined or ACSPL+). If the array is one-dimension, number of rows is always one and the number of columns represents the number of elements.

Syntax

```
sizeof(Array, index(optional) )
```

Arguments

| | |
|-------|-------------------------------------------------|
| Array | The array name, can be local or global |
| Index | Optional parameter. If specified, can be 1 or 2 |

Return Value

- > No index is specified: total number of elements in the array
- > Index parameter equals to 1: number of columns for two-dimensional array, or number of elements for one-dimensional array
- > Index parameter equals to 2: number of rows for two-dimensional array, or 1 for one-dimensional array.

Comments

In case of wrong parameters, the corresponding runtime error will be generated. The function is intended to be used for arrays only, meaning that an error is generated if a scalar is given as a parameter.

Example

```
I0=sizeof(NST)
```

4.5 EtherCAT Functions

EtherCAT functions include:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| COGETSIZE | Returns the size, in bits, of a specific entry in the object dictionary of a specific slave. |
| ECCLOSEPORT | The function closes the specified port of specified EtherCAT node. |
| ECCLRREG | Clears the contents of error counters registers. |
| ECEXTIN | Used for mapping input variables (TxPDO) to non-ACS EtherCAT network. |
| ECEXTOUT | Used for mapping output variables (RxPDO) from non-ACS EtherCAT network to the SPiiPlusES. |
| ECGETGRPIND | The function returns an array that contains optional groups' indexes that are part of the current configuration (including mandatory group, which is 0). |

| | |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| ECGETPID | Returns product ID of the node. |
| ECGETMAIN | The function returns the number of EtherCAT slaves connected to the main EtherCAT line. |
| ECGETOFFSET | The function returns offset of the specified variable of the specific EtherCAT node. |
| ECGETOPTGRP | The function returns number of actually connected optional groups, not including the mandatory group. |
| ECGETRED | The function returns the number of EtherCAT slaves connected to the redundant EtherCAT line. |
| ECGETREG | Gets the contents of ESCs error counters registers. |
| ECGETSLAVES | Returns the number of slaves in an EtherCAT network. |
| ECGETSTATE | Returns the state of the node. |
| ECGETVID | Returns vendor ID of the node. |
| ECGRPINFO | The function fills array with nodes' indexes which are members of a given optional group. In addition, it returns the number of members in the group. |
| ECIN | Copies the EtherCAT network input variable to an ACSPL+ variable. |
| ECOUT | Copies ACSPL+ variable to EtherCAT network output variable. |
| ECRESCAN | Triggers the system to rescan the EtherCAT network after a slave has been removed or been added in order to refresh the network composition data. |
| ECRESCUE | Triggers execution of a rescue scan of the EtherCAT network |
| ECREPAIR | Returns the system back to the operational state if one or more slaves underwent a reset or power cycle. |
| ECSAVECFG | Saves the current network topology configuration into the non-volatile memory. |
| ECSAVEDCNF | The function returns array that contains optional groups' indexes that are part of the last saved configuration (including mandatory group, which is 0). |
| ECUNMAP | Resets mapping of ECIN and ECOUT . |

| | |
|-----------------------------|-------------------------------------------------------------------------|
| ECUNMAPIN | Resets mapping of ECIN of a specified offset. |
| ECUNMAPOUT | Resets mapping of ECOUT of a specified offset. |
| FOEDOWNLOAD | Downloads a file over EtherCAT from the controller's flash to slave |
| FOEUPLOAD | Upload file over EtherCAT and saves it to the controller's flash memory |
| PDOEXT | Prints the PDO configuration defined by master (SPiiPlusES only) |

4.5.1 COEGETSIZE

Description

COEGETSIZE returns the size, in bits, of a specific entry in the object dictionary of a specific slave.

Syntax

int **COEGETSIZE**(*Slave, Index, Subindex*)

Arguments

| | |
|-----------------|------------------------------------------------------------|
| <i>Slave</i> | An integer representing slave number, starting from 0 |
| <i>Index</i> | An integer representing index in the object dictionary |
| <i>Subindex</i> | An integer representing sub-index in the object dictionary |

Return Value

Size in bits of the entry in the object dictionary.

Example

```
I0=coegetsize(0,0x1000,0)

!Return value: 32 bits. Object 0x1000 usually means the device type.
```

Comments:

The function returns the received value or fails with runtime error. The function cannot be used in the SPiiPlus MMI Application Studio Communication Terminal. The function delays the buffer execution on its line until it's successful or fails the whole buffer with timeout or other error.

4.5.2 ECCLOSEPORT

Description

The function closes the specified port of specified EtherCAT node.

Syntax

Int **ECCLOSEPORT**(*name, index*)

Arguments

| | |
|--------------|-------------------------------------------------------------------|
| Index | EtherCAT node index. |
| Port | EtherCAT port index (0 - EtherCAT IN port, 1 - EtherCAT OUT port) |

Return Value

None

Comments

This function can only be used if ring topology is configured and ring communication is active. In case there is a cable / port failure on a specific EtherCAT node, it is recommended (as long as the erroneous situation exists) that the machine should start up directly in the line topology mode. This is very useful, because it prevent a manual stop of the machine with a suspicious communication link / device).

Example

```

ECDCLOSEPORT(<slave index>, <port index>)
FCLEAR ALL
ECSAVECFG
STOP

```

In this example of the AUTOEXEC program that should run on power-up in this case:

The first parameter is the slave index in the network and the second parameter is the port index of the slave that should be closed.

After "fclear All", two Lines Are Stored Into The Controller's Non-volatile Memory.

4.5.3 ECCLRREG**Description**

ESC Error Counters Registers Clear. The **ECCLRREG** function clears the contents of the error counters registers.

Syntax

```
void ECCLRREG(index,offset)
```

Arguments

| | |
|---------------|-----------------------------------------|
| Index | EtherCAT slave index. |
| Offset | Register offset in the Beckhoff memory. |

Return Value

None

Comments

When the Offset value is -1, all error counters in all slaves are cleared. Otherwise, only the specific register at the specified Offset is cleared.

After executing the **ECCLRREG** function, we recommend to execute the **FCLEAR** function without parameters before running **ECGETREG**.

Example

Run the following code example in a Program Buffer.

```
ECCLRREG (0, 0x310)
FCLEAR
STOP
```

You can also enter this code in the SPiiPlus MMI Application Studio Connection Terminal: `ECCLRREG (0, -1)`.

4.5.4 ECEXTIN

Description

The **ECEXTIN** function is used for mapping input variables (TxPDO) to non-ACS EtherCAT network.



Can be used only by SPiiPlusES.

Syntax

```
ECEXTIN (int PDOIndex, int Index, int Subindex, string Varname)
```

Arguments

| | |
|----------|----------------------------------------------------------------------|
| PDOIndex | One of the TxPDO indexes defined by external master's configuration. |
| Index | Index of the variable mapped to the TxPDO |
| Subindex | Sub-Index of the variable mapped to the TxPDO |
| Varname | Valid name of a global ACSPL+ variable. |

Return Value

None

Comments

Once the function is called successfully, the Firmware copies the value of the network input variable into the ACSPL+ variable every controller cycle. The input is from the external EtherCAT master (e.g. TwinCAT) point of view (a value provided by SPiiPlusES to the external master).

There is no restriction on number of the mapped network variables.

The mapping is allowed only when the SPiiPlusES is in OP state.

All types supported by the SPiiPlusES are also supported by the ECEXTIN function.



Error 3309 "Function is supported only by SPiiPlusES" is returned when called on a controller which is not SPiiPlusES.

If SPiiPlusES is not in OP state, error 3310 "SPiiPlusES is not in OP state. PDO is not enabled" error is given.

The mapping is rested by the ECUNMAP command.

COM Library Methods

None

C Library Functions

None

Example

```
D-Buffer:
GLOBAL INT ActualPositionTwinCAT

Regular Buffer:
ecextin (0x1A01,0x6064,0, ActualPositionTwinCAT)

STOP
```

4.5.5 ECEXTOUT

Description

The **ECEXTOUT** function is used for mapping output variables (RxPDO) from non-ACS EtherCAT network to the SPiiPlusES.



Can be used only by SPiiPlusES.

Syntax

```
ECEXTOUT (int PDOIndex, int Index, int Subindex, string Varname)
```

Arguments

| | |
|----------|----------------------------------------------------------------------|
| PDOIndex | One of the RxPDO indexes defined by external master's configuration. |
| Index | Index of the variable mapped to the TxPDO |
| Subindex | Sub-Index of the variable mapped to the TxPDO |

Varname

Valid name of a global ACSPL+ variable.

Return Value

None

Comments

Once the function is called successfully, the Firmware copies the value of the network output variable into the ACSPL+ variable every controller cycle. The output is from the external EtherCAT master (e.g. TwinCAT) point of view (a value provided to SPiiPlusES by the external master).

There is no restriction on number of the mapped network variables.

Error 3309 "Function is supported only by SPiiPlusES" is returned when called on a controller which is not SPiiPlusES.

If SPiiPlusES is not in OP state, error 3310 "SPiiPlusES is not in OP state. PDO is not enabled" error is given.



The mapping is allowed only when the SPiiPlusES is in OP state.

All types supported by the SPiiPlusES are also supported by the ECEXTOUT function.

The mapping is rested by the ECUNMAP command.

COM Library Methods

None

C Library Functions

None

Example

```
D-Buffer:
GLOBAL INT TargetPositionTwinCAT

Regular Buffer:
ecextout (0x1601,0x607A,0, TargetPositionTwinCAT)

STOP
```

4.5.6 ECGETGRPIND

Description

The function returns an array that contains optional groups' indexes that are part of the current configuration (including mandatory group, which is 0). The array ends with {-1}.

Syntax

ECGETGRPIND(groups_array)

Arguments

groups_array

Array of type INT, filled with groups' indexes. {-1} marks the end.

Return Value

None.

Example:

```
int groups (5)
ecgetgrpind(groups) ! groups array is filled with: {0,1,-1,0,0},
!meaning that there is one optional group (with index 1) defined in the
actual system.
```

4.5.7 ECGETPID**Description**

ECGETPID returns product ID of the node.

Syntax

ECGETPID(index)

Arguments

Index

EtherCAT slave index, starting from 0.

Return Value

Product ID of the node.

4.5.8 ECGETMAIN**Description**

The function returns the number of EtherCAT slaves connected to the main EtherCAT line.

Syntax

ECGETMAIN()

Return Value

The number of EtherCAT slaves connected to the main EtherCAT line

4.5.9 ECGETOFFSET**Description**

The function returns offset of the specified variable of the specific EtherCAT node. The "/b" switch is used to retrieve the offset on the variable in bits.

SyntaxInt ECGETOFFSET[/b](*name, index, InOut(optional), occurrence(optional)*)

Arguments

| | |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Name | Name of variable as shown in System Setup of #ETHERCAT command. |
| Index | EtherCAT node index. |
| InOut (optional) | Can be Output (=0) or Input (=1). By default, the variable is being searched in Inputs and, if not found, in Outputs. |
| Occurrence (optional) | By default the value is 0. |

Function Options

| | |
|-----------|----------------------------------------|
| /b | Returns the bit offset of the variable |
|-----------|----------------------------------------|

Comments

The offset in bits can be calculated by following method:

Offset in Bytes as provided by **#ETHERCAT** report.

The return value of the **ECGETOFFSET/B** will be equal to:

<Offset in Bytes>*8+<Offset in Bits>

Return Value

Offset of the specified variable.

Example

```
GLOBAL INT IN0_OFFSET
! Assuming that IOMnt is the first EtherCAT node in the network
IN0_OFFSET = ECGETOFFSET("Digital Inputs 0", 0)
! Assuming that WAGO is the first EtherCAT node in the network.
! Returns offset of network variable (PDO) "Channel 1 Data", slave 0,
Output, first occurrence
I0=ECGETOFFSET("Channel 1 Data",0)
!Returns offset of network variable (PDO) "Channel 1 Data", slave 0,
Input, first occurrence
I1=ECGETOFFSET("Channel 1 Data",0,1,1)
!Returns offset of network variable (PDO) "Channel 1 Data", slave 0,
Input, second occurrence
```

Example

```
WAGO_Offset_Bit, WAGO_Offset_Byte
WAGO_Offset_Bit =ECGETOFFSET/b ("Input(s).Channel 2, Word 1",0)
WAGO_Offset_Byte = ECGETOFFSET ("Input(s).Channel 2, Word 1",0)
```

4.5.10 *ECGETOPTGRP*

Description

The function returns number of actually connected optional groups, not including the mandatory group.

Syntax

ECGETOPTGRP()

Arguments

None.

Return Value

Returns number of actually connected optional groups, not including the mandatory group.

Example:

```
?ecgetoptgrp ()
2
```

4.5.11 *ECGETRED*

Description

The function returns the number of EtherCAT slaves connected to the redundant EtherCAT line.

Syntax

ECGETRED()

Return Value

The number of EtherCAT slaves connected to the redundant EtherCAT line.

4.5.12 *ECGETREG*

Description

ESC Error Counters Registers (Beckhoff Memory). The ESCs have numerous error counters that help you detect and locate errors. The ECGETREG function enables you to view the contents of these registers.

Syntax

int **ECGETREG**(*index,offset*)

Arguments

| | |
|---------------|-----------------------------------------|
| Index | EtherCAT slave index. |
| Offset | Register offset in the Beckhoff memory. |

Return Value

None

Comments

The following table lists supported error counter registers.

Table 4-5. Supported Error Counter Registers

| Offset | Name | Description |
|--------|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x300 | Port Error Counter (CRC A) | Error Counted at the Auto-Forwarded (per port). Each register contains two counters: <ul style="list-style-type: none"> > Invalid Frame Counter: 0x300/2/4/6 > RX Error Counter: 0x301/3/5/7 |
| 0x302 | Port Error Counter (CRC B) | |
| 0x304 | Port Error Counter (CRC C) | |
| 0x306 | Port Error Counter (CRC D) | |
| 0x308 | Forwarded RX Error Counter (CRC A/B) | Invalid frame with marking from previous ESC detected (per port). |
| 0x309 | Forwarded RX Error Counter | |
| 0x30A | Forwarded RX Error Counter (CRC C/D) | |
| 0x30B | Forwarded RX Error Counter | |
| 0x30C | ECAT Processing Unit Error Counter | Invalid frame passing the EtherCAT Processing Unit (additional checks by processing unit). |
| 0x30D | PDI Error Counter | Physical Errors detected by the PDI. |
| 0x310 | Lost Link Counter, Port A (IN) | Link Lost events (per port). |
| 0x311 | Lost Link Counter, Port B (OUT) | |
| 0x312 | Lost Link Counter, Port C | |
| 0x313 | Lost Link Counter, Port D | |



If a cable is unplugged, we recommend using the **FCLEAR** command before using **ECGETREG**.
The mapping is allowed only when stack is operational.

Example

Run the following code example in a Program Buffer.

```
I0=ECGETREG (0, 0x310)
STOP
```

You can also enter this code in the SPiiPlus MMI Application Studio Connection Terminal:

```
?ECGETREG (0, 0x310)
```

4.5.13 ECGETSLAVES

Description

This function is used to retrieve the number of slaves in an EtherCAT network.

Syntax

ECGETSLAVES()

Arguments

None

Return Value

Number of EtherCAT slaves in the network.

Comments

If a slave was added or removed, the **ECRESCAN** command should be used before using **ECGETSLAVES** again.

4.5.14 ECGETSTATE

Description

ECGETSTATE returns the state of the node.

Syntax

ECGETSTATE(index)

Arguments

Index

EtherCAT slave index, starting from 0.

Return Value

INIT, PREOP, SAFEOP, OP.

4.5.15 ECGETVID

Description

ECGETVID returns the vendor ID of the node.

Syntax

ECGETVID(index)

Arguments

| | |
|--------------|----------------------------------------|
| <i>Index</i> | EtherCAT slave index, starting from 0. |
|--------------|----------------------------------------|

Return Value

The ACS vendor ID of the node.

4.5.16 ECGRPINFO

Description

The function fills array with nodes' indexes which are members of a given optional group. In addition, it returns the number of members in the group.

Syntax

ECGRPINFO(group_index, nodes_array)

Arguments

| | |
|--------------------|--------------------------------------------------------------------|
| <i>group_index</i> | Optional Group Index, starting from 0 (0 is the mandatory group) |
| <i>nodes_array</i> | Array of type INT, filled with nodes' indexes. {-1} marks the end. |

Return Value

Returns the number of the members in the specified optional group.

Example:

```
int nodes (5)
I0=ecgrpinfo(1,nodes) ! returns number of nodes in optional group 1
! nodes array is filled with: {0,-1,0,0,0}
```

4.5.17 ECIN

Description

This function is used to copy the EtherCAT network input variable at the corresponding EtherCAT offset into the specified ACSPL+ variable.

Syntax

ECIN[b](int offset, Varname)

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| offset | Internal EtherCAT offset (in bytes or in bits) of network variable derived from the SPiiPlus MMI Application Studio Communication Terminal ETHERCAT command. <ul style="list-style-type: none"> > No switch: EtherCAT offset in bytes > "/b" switch: the offset is in bits |
| Varname | Valid name of ACSPL+ variable, global or standard. |

Return Value

None

Comments

Once the function is called successfully, the Firmware copies the value of the network input variable at the corresponding EtherCAT offset into the specified ACSPL+ variable, every controller cycle.

There is no restriction on number of mapped network variables.



The mapping is allowed only when stack is operational.

In the event of wrong parameters or stack state, the function will produce a corresponding runtime error.



It is recommended to use the **ECGETOFFSET** function to retrieve the offset, in bytes or in bits.

COM Library Methods

None

C Library Functions

acsc_MapEtherCATInput

Example

```

D-Buffer:
GLOBAL INT IOMNT_IN(4)
GLOBAL INT IOMNT_OUT(4)

Regular Buffer:
AUTOEXEC:

INT IN0_OFFSET
INT IN1_OFFSET
INT IN2_OFFSET
INT IN3_OFFSET

INT OUT0_OFFSET
INT OUT1_OFFSET
INT OUT2_OFFSET
INT OUT3_OFFSET

! Assuming that IOMnt is the first EtherCAT node in the network
IN0_OFFSET = ECGETOFFSET("Digital Inputs 0", 0)
IN1_OFFSET = ECGETOFFSET("Digital Inputs 1", 0)
IN2_OFFSET = ECGETOFFSET("Digital Inputs 2", 0)
IN3_OFFSET = ECGETOFFSET("Digital Inputs 3", 0)

OUT0_OFFSET = ECGETOFFSET("Digital Outputs 0", 0)
OUT1_OFFSET = ECGETOFFSET("Digital Outputs 1", 0)
OUT2_OFFSET = ECGETOFFSET("Digital Outputs 2", 0)
OUT3_OFFSET = ECGETOFFSET("Digital Outputs 3", 0)

ECIN(IN0_OFFSET, IOMNT_IN(0))
ECIN(IN1_OFFSET, IOMNT_IN(1))
ECIN(IN2_OFFSET, IOMNT_IN(2))
ECIN(IN3_OFFSET, IOMNT_IN(3))

ECOUT(OUT0_OFFSET, IOMNT_OUT(0))
ECOUT(OUT1_OFFSET, IOMNT_OUT(1))
ECOUT(OUT2_OFFSET, IOMNT_OUT(2))
ECOUT(OUT3_OFFSET, IOMNT_OUT(3))

STOP

```

4.5.18 ECOUT**Description**

This function is used to copy the value of ACSPL+ variable into the network output variable at the corresponding EtherCAT offset.

Syntax

ECOUT[/b>[/r] (*int offset, Vname*)

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| offset | Internal EtherCAT offset (in bytes or in bits) of network variable derived from the SPiiPlus MMI Application Studio Communication Terminal ETHERCAT command. <ul style="list-style-type: none"> > No switch: EtherCAT offset in bytes > "/b" switch: the offset is in bits |
| Varname | Valid name of ACSPL+ variable, global or standard. |

Switches

| | |
|-----------|------------------------------------------------------------------------------------------------------------------------------|
| /r | Copies the EtherCAT network output (RxPDO) variable at the corresponding EtherCAT offset into the specified ACSPL+ variable. |
|-----------|------------------------------------------------------------------------------------------------------------------------------|

Return Value

None

Comments

Once the function is called successfully, the Firmware copies the value of the specified ACSPL+ variable network input variable into the given EtherCAT offset, every controller cycle.

There is no restriction on number of mapped network variables.



Mapping is allowed only when stack is operational. ACS recommends retrieving the offset using ACSPL+ `ECGETOFFSET()`.

In the event of wrong parameters or stack state, the function will produce corresponding runtime error.



It's recommended to use the `ECGETOFFSET` function to retrieve the offset, in bytes or in bits.

COM Library Methods

None

C Library Functions

`acsc_MapEtherCATOutput`

Example 1

```

D-Buffer:
GLOBAL INT IOMNT_IN(4)
GLOBAL INT IOMNT_OUT(4)
Regular Buffer:
AUTOEXEC:
INT IN0_OFFSET

```

```

INT IN1_OFFSET
INT IN2_OFFSET
INT IN3_OFFSET
INT OUT0_OFFSET
INT OUT1_OFFSET
INT OUT2_OFFSET
INT OUT3_OFFSET
! Assuming that IOMnt is the first EtherCAT node in the network
IN0_OFFSET = ECGETOFFSET("Digital Inputs 0", 0)
IN1_OFFSET = ECGETOFFSET("Digital Inputs 1", 0)
IN2_OFFSET = ECGETOFFSET("Digital Inputs 2", 0)
IN3_OFFSET = ECGETOFFSET("Digital Inputs 3", 0)
OUT0_OFFSET = ECGETOFFSET("Digital Outputs 0", 0)
OUT1_OFFSET = ECGETOFFSET("Digital Outputs 1", 0)
OUT2_OFFSET = ECGETOFFSET("Digital Outputs 2", 0)
OUT3_OFFSET = ECGETOFFSET("Digital Outputs 3", 0)
ECIN(IN0_OFFSET, IOMNT_IN(0))
ECIN(IN1_OFFSET, IOMNT_IN(1))
ECIN(IN2_OFFSET, IOMNT_IN(2))
ECIN(IN3_OFFSET, IOMNT_IN(3))
ECOUT(OUT0_OFFSET, IOMNT_OUT(0))
ECOUT(OUT1_OFFSET, IOMNT_OUT(1))
ECOUT(OUT2_OFFSET, IOMNT_OUT(2))
ECOUT(OUT3_OFFSET, IOMNT_OUT(3))
STOP

```

Example 2

```

D-Buffer:
GLOBAL INT TargetPosition
Regular Buffer:
!Assuming the first EtherCAT node in the network has "Target Position"
network variable
ecout/r ( ECGETOFFSET("Target Position",0),TargetPosition)
STOP

```

4.5.19 ECREPAIR

Description

ECREPAIR serves to return the system back to the operational state if one or more slaves underwent a reset or power cycle. It provides an ability to recover EtherCAT network when there is a need to replace unit for maintenance without the need to perform commutation, homing, etc., to all other units within the EtherCAT network.

Syntax

ECREPAIR

Comments

ECREPAIR performs the following steps:

1. Detects which nodes are not communicating
2. Brings all slaves to the EtherCAT OP state
3. Establishes inter-slaves and master-slaves synchronization
4. Downloads Servo Processor programs to repaired nodes only
5. Restores all Servo Processor variable values accordingly

ECREPAIR can take a long time to complete; therefore it is recommended calling **ECREPAIR** from a Program Buffer. It is possible to execute the **ECREPAIR** command from the SPiiPlus MMI Application Studio Communication Terminal; however, in this case the communication timeout should be configured to be longer.



It is strongly recommended not to take any other actions until **ECREPAIR** completes.

Once the process is complete, the system state can be evaluated through:

- > **ECST**
- > **SYNC** values
- > Servo Processor Alarm indication on all axes
- > The **View System Configuration** task of the System Configuration Wizard

If all of these indicators show normal operation status, the **ECREPAIR** operation was successful.

EtherCAT slaves that were operational before **ECREPAIR** activation will keep their feedback and commutation valid.

4.5.20 **ECRESCAN**

Description

ECRESCAN triggers the system to rescan the EtherCAT network after a slave has been removed or been added in order to refresh the network composition data.

Syntax

ECRESCAN

Arguments

None

Comments

The command can be entered either through a Program Buffer or via the SPiiPlus MMI Application Studio **Communication Terminal**.

During controller power-up, the controller automatically detects the EtherCAT network change and informs the user, making a call to **ECRESCAN** unnecessary.

4.5.21 **ECRESCUE**

Description

ECRESCUE triggers execution of a rescue scan of the EtherCAT network. The scan will detect the failure location preventing return of EtherCAT frames to the EtherCAT master.

Syntax

```
ECRESCUE
```

Arguments

None

Comments

The command can be entered through a Program Buffer in the SPiiPlus MMI Application Studio.

Procedure when using an application developed by the user:

1. Issue the **ECRESCUE** command to execute EtherCAT network rescue scan
2. Check **ECERR** value
3. In case of **ECERR** = 6024, Issue the **?ECGETMAIN()** and **?ECGETRED()** commands to identify the location of the failure
4. The controller must be rebooted after running the command **ECREPAIR**

In most cases the **ECRESCUE** command shouldn't be explicitly executed by the user. During the controller power up, the controller automatically detects the EtherCAT network failure and informs the user.

4.5.22 ECSAVECFG

Description

The command saves to flash an array of optional groups based on current configuration, based on `ecgetgrpind()` function. For example, actual configuration that contains 1 optional group will be represented as: {0,1,-1}. This array will be read upon power-up of the controller. Actual configuration will be checked against the approved configuration. If not identical, error 6016, "The actual network configuration doesn't match the last approved configuration.", is displayed.

Syntax

```
ECSAVECFG
```

Arguments

None

Return Value

None.

Example:

```
ecsavecfg
```

4.5.23 ECSAVEDCNF

Description

The function returns array that contains optional groups' indexes that are part of the last saved configuration (including mandatory group, which is 0). The array ends with {-1}.

Syntax

```
ECSAVEDCNF(groups_array)
```

Arguments

| | |
|---------------------|---------------------------------------------------------------------|
| groups_array | Array of type INT, filled with groups' indexes. {-1} marks the end. |
|---------------------|---------------------------------------------------------------------|

Return Value

None.

Example:

```
int groups (5)
ecsavedcnf (groups) ! groups array is filled with: {0,1,-1,0,0},
>!meaning that there is one optional group (with index 1) in the last
saved configuration.
```

4.5.24 ECUNMAP

Description

This function is used to reset all previous mapping defined by [ECIN](#) and [ECOUT](#).

Syntax

ECUNMAP

Arguments

None

Return Value

None

Comments

The mapping is allowed only when stack is operational.

COM Library Methods

None

C Library Functions

acsc_UnmapEtherCATInputsOutputs

4.5.25 ECUNMAPIN

Description

This function is used to reset all previous mapping defined by [ECIN](#) to a specific offset.

Syntax

ECUNMAPIN(*ECOffset*)

Arguments

| | |
|-----------------|---------------------------------------------------------------------------------------------|
| ECOffset | An integer providing the offset to which a variable was mapped using ECIN . |
|-----------------|---------------------------------------------------------------------------------------------|

Return Value

None

Comments

The mapping is allowed only when stack is operational.

Example

Given the previous execution of `ECIN(48,I0)`, `ECUNMAPIN(48)` will unmap only I0.

4.5.26 ECUNMAPOUT**Description**

This function is used to reset all previous mapping defined by `ECOUT` to a specific offset.

Syntax

`ECUNMAPOUT(ECOffset)`

Arguments

| | |
|-----------------------|---------------------------------------------------------------------------------------------|
| <code>ECOffset</code> | An integer providing the offset to which a variable has been mapped by <code>ECOUT</code> . |
|-----------------------|---------------------------------------------------------------------------------------------|

Return Value

None

Example

Assuming previous execution of `ECOUT(48,I0)` and `ECOUT(50,I1)`, executing `ECUNMAPOUT(48)` will unmap only I0.

4.5.27 FOEDOWNLOAD**Description**

The function downloads a file over EtherCAT from the controller's flash to slave.

Syntax

```
FOEDOWNLOAD[/b]( string LocalPath, string TargetFileName, int SlaveIndex)
```

Arguments

| | |
|----------------|-----------------------------------------------------|
| LocalPath | The path of the file in ACS controller's flash |
| TargetFileName | The name of the file to be transferred to the slave |
| SlaveIndex | Index of the slave in the EtherCAT network |

Flags

| | |
|-----------------|------------------------------------------------------------------------------------------------------------|
| <code>/b</code> | With switch "b", the FW will attempt to set the slave to BOOTSTRAP mode before the FoE transfer operation. |
|-----------------|------------------------------------------------------------------------------------------------------------|

Comments

Switch `/b`: even if the slave doesn't support BOOTSTRAP, the firmware will attempt to set it back to OP state without executing the FoE operation.

The function blocks the ACSPL+ buffer until completion.

It cannot be called from the terminal.

The following errors are supported in case of a failure:

- > Error 3317 "FoE Error: Access Denied"
- > Error 3308 "The disk is full or the file is too big"
- > Error 3307 "FoE Protocol is not support by Slave"
- > Error 3040 "Unable to open file"

This function is supported in version 3.00 and higher.



This function is intended for use with EtherCAT slaves that support the FoE protocol.

Example

```
FOEDOWNLOAD/b ("C:\Test.txt","Test",0)
```

4.5.28 FOEUPLOAD

Description

The function reads a file over EtherCAT from a slave and saves it to the controller's flash memory.

Syntax

```
FOEUPLOAD( string LocalPath, string TargetFileName, int MaxFileSize, int SlaveIndex)
```

Arguments

| | |
|----------------|-------------------------------------------------------------------------------|
| LocalPath | The path of the file as it will be saved to the ACS controller's flash memory |
| TargetFileName | The name of the file on the slave |
| MaxFileSize | The maximum size of the file (in bytes) |
| SlaveIndex | Index of the slave in the EtherCAT network |

Comments

- > The function blocks the ACSPL+ buffer processing until completion. It cannot be called from the terminal. For complete list of errors, see the FoE Errors table.
- > The function is intended for use with EtherCAT slaves that support the FoE protocol.

Example

```
! Reads file "FileToRead" from slave 0, saves it as "Text.xml"
! Maximum size is 10MB
FOEUPLOAD("Text.xml","FileToRead",10000000,0)
STOP
```

4.5.29 PDOEXT

The PDOEXT command may be used with ACS DS402 products. The command prints the PDO configuration defined by the master that controls the device.

Example

| In/Out | ObjectIndex | SubIndex | Size | PdoIndex |
|--------|-------------|----------|------|----------|
| In | 0x6041 | 0x00 | 16 | 0x1a02 |
| In | 0x606c | 0x00 | 32 | 0x1a02 |
| In | 0x6841 | 0x00 | 16 | 0x1a11 |
| In | 0x6864 | 0x00 | 32 | 0x1a11 |
| Out | 0x6040 | 0x00 | 16 | 0x1602 |
| Out | 0x60ff | 0x00 | 32 | 0x1602 |
| Out | 0x6840 | 0x00 | 16 | 0x1611 |
| Out | 0x687a | 0x00 | 32 | 0x1611 |

4.6 CoE Functions

CoE functions are required for SDO transfers in CoE. SDO are part of the cyclic EtherCAT data transfer. It is impossible to define a generic function for any kind of mailbox transfer, as protocols like EoE, FoE and VoE have their own definitions. So CoE is supported first.



The SPiiPlus MMI Application Studio **Communication Terminal** [ETHERCAT](#) command reports for every slave if it has Mailbox support.



CoE functions cannot be used with ACS EtherCAT slaves since the CoE protocol is not supported.

The CoE functions are:

| Function | Description |
|------------|----------------------------------------------------------------------------------------------|
| COERead | Read CoE slave Object Directory entry. |
| COEWriTE | Write into CoE slave Object Directory. |
| COEGETSiZE | Returns the size, in bits, of a specific entry in the object dictionary of a specific slave. |

4.6.1 COERead

Description

This function is used to read Object Dictionary entry from CoE slave.

This function with the "/d" switch provides the capability of reading a double type (64 bit).

In addition, parameter slave with the "-1" value means SPiiPlusES' Object Dictionary. The following objects can be read from the SPiiPlusES' Object Dictionary:

- > CiA402 objects, range: 0x6000-0x9FFF

This function with the "/l" (non-capital L) switch is an extension to the existing COERead function, and enables reading of objects bigger than 64 bits, e.g. strings.

Syntax

COERead/d[/size] (int slave,int Index,int Subindex)

COERead/l (int Slave, int Index, int Subindex, int Len, int Array[])

Arguments

| | |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| size | 1/2/4 number of bytes in the Object Dictionary /f for floating (32 bit) or /d for double (64 bit) |
| slave | Slave number (which can be obtained by running the SPiiPlus MMI Application Studio Communication Terminal ETHERCAT command). -1 means "SPiiPlusES" |
| Index | Index in the Object Dictionary. |
| Subindex | Sub-index in the Object Dictionary. |
| Len | Number of bytes to read if "l" switch is used. The maximum value is 100. |
| Array | A user-defined one-dimensional integer ACSPL+ array that will store the result value. Each element is treated as a single byte, so size of Array should be at least "Len". |

Return Value

COERead/d returns the value stored in the variable in the specified **Index** of the Object Dictionary.

COEREAD/l has no return value, the result is found in the **Array** parameter

Comments

If the object doesn't exist, error 3303 "Error SDO: Object doesn't exist in the Object Dictionary" is returned. In case of wrong parameters, the corresponding runtime error will be generated. The function cannot be used in the **Communication Terminal**. The function delays the buffer execution on its line until it is successful or fails the whole buffer with timeout or other error.

COM Library Methods

None

C Library Functions

None

Example 1

```
COEREAD/4 (0,0x6040,0)
```

This reads 4 bytes from **slave** 0, at **Index** 0x6040, **Subindex** 0 and returns the value that is stored in the variable at this **Index**.

Example 2

```
V0=Coeread/d (0,0x2801,1)
```

```
STOP
```

In the example above, 8 bytes are read from slave 0, index 0x2801, subindex 1. The returned value is being stored in the V0 global REAL variable.

Example 3

4.6.2 COEWRITE

Description

This function is used to write a value into the CoE slave Object Dictionary.

This function with the "/d" switch is an extension to the existing coeread function, and provides the capability of reading a double type (64 bit).

This function with the "/l" (non-capital L) switch is an extension to the existing COEREAD function, and enables reading of objects bigger than 64 bits, e.g. strings.

Syntax

COEWRITE/d[/size] (int slave,int Index,int Subindex,double Value)

COEREAD/l (int Slave, int Index, int Subindex, int Len, int Array[])

Arguments

| | |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| size | 1, 2 or 4 - the number of bytes in the Object Dictionary /f for floating.(32 bit) or /d for double (64 bit) |
| slave | Slave number (which can be obtained by running the SPiiPlus MMI Application Studio Communication Terminal ETHERCAT command) |
| Index | Index in the Object Dictionary. |
| Subindex | Sub-index in the Object Dictionary. |
| Value | The value to be written. |
| Len | Number of bytes to read if "l" switch is used. The maximum value is 100. |
| Array | A user-defined one-dimensional integer ACSPL+ array that will store the result value. Each element is treated as a single byte, so size of Array should be at least "Len". |

Return Value

None

Comments

If the object doesn't exist, error 3303 "Error SDO: Object doesn't exist in the Object Dictionary" is returned. In case of wrong parameters, the corresponding runtime error will be generated. The function cannot be used in the **Communication Terminal**. The function delays the buffer execution on its line until it is successful or fails the whole buffer with timeout or other error.

COM Library Methods

None

C Library Functions

None

Example 1

```
COEWRITE/4 (0,0x6041,0,0x0)
```

This writes the **Value** 0 (4 bytes) into **slave** 0, at **Index** 0x6041, **Subindex** 0.

Example 2

```
Coewrite/d (0,0x2801,1,555.666)
```

```
STOP
```


In the example above, the value "555.666" is being written to slave 0, index 0x2801, subindex 1.

Example 3

4.7 Modbus Functions

| Function | Description |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>MBOPEN</code> | Establishes a Modbus TCP connection with a Modbus server device using the server's IP address |
| <code>MBGETHANDLE</code> | Returns the server's communication handle if a connection has been established using the specified IP address |
| <code>MBCLOSE</code> | Closes an open Modbus TCP connection using the server's communication handle or IP address |
| <code>MBREADHREG</code> | Maps a Modbus server's holding register to a specified ACSPL+ variable. The value in the register is then read to the variable, either once or at a specified interval. |
| <code>MBREADIREG</code> | Used to map a Modbus Server's input register to a specified ACSPL+ variable. The value in the register is then read to the variable, either once or at a specified interval. |
| <code>MBWRITEHREG</code> | The function is used to map a client's ACSPL+ variable to a Modbus server's register. The value in the client variable is then written to the server's register, either once or at a defined interval. |
| <code>MBREADCOIL</code> | The function is used to map a Modbus server coil to a specified ACSPL+ variable and read the state of the coil |
| <code>MBWRITECOIL</code> | The function is used to map a Modbus server coil to a specified ACSPL+ variable and write the variable's value to the coil. |
| <code>MBREADDIN</code> | The function maps a Modbus server discrete input to an ACSPL+ variable |
| <code>MBUNMAP</code> | Unmaps a specific request using the request ID |
| <code>MBCLEAR</code> | Clears the error codes from the Modbus requests error array (MBERR) and reactivates the requests that experienced the error condition |
| <code>MBERR</code> | Holds the most recent Modbus error code that occurred for the request during the communication process |
| <code>#MBMAPREP</code> | Displays a report of all the active Modbus connections and mapped variables |

4.7.1 MBOPEN

Description

MBOPEN establishes a Modbus TCP connection with a Modbus server device using the server's IP address.

Syntax

int **MBOPEN**[/switches] (server_ip[, server_id, word_order])

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| server_ip | The server IP address as a string. For example, "192.168.1.10". |
| server_id | (Optional) Modbus server ID (1 to 247), the default value is 1 |
| word_order | (Optional) Specifies the order or sequence in which words (registers) are received or sent. To transfer 32-bit (4 byte) or 64-bit (8 byte) values using the Modbus protocol, you must define the order in which the registers are received. (Most vendors choose to map the least significant word onto the lower address of the register pair). <ul style="list-style-type: none"> > big-endian (0) > little-endian (1) (default) |

Return Value

On success: A positive integer value that is used as a communication handle for the server

On failure: A runtime error will occur

Comments

- > Up to three servers can be connected to the client controller at any given time.
- > A runtime error will occur if the client fails to open a connection with the server.
- > If a connection with the specified IP address is already open, the function will return the communication handle for the server.
- > To communicate with an ACS server device, a valid **server_id** must be specified (the ACS server device has a **CONID** variable that specifies the server's ID).
- > Some I/O devices might have a Modbus connection timeout; after a period of inactivity in the channel, the connection is closed (this behavior and the timeout period are usually user-defined).

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates how to open a connection to a Modbus server on controller power-up by using the AUTOEXEC label.

```
AUTOEXEC:
GLOBAL INT server_handle
!Opens a Modbus TCP connection with a server device
server_handle = MBOPEN("10.0.0.100")
STOP
```

Example 2

This example demonstrates how to open a connection to a Modbus server by specifying the server_id and word_order optional arguments.

```
GLOBAL INT server_handle, server_id, word_order
server_id= 10
word_order = 0 !big-endian
!Opens a Modbus TCP connection with a server device
server_handle = MBOPEN("10.0.0.100", server_id, word_order)
STOP
```

Related ACSPL+ Commands

MBCLOSE, MBGETHANDLE

4.7.2 MBGETHANDLE

Description

MBGETHANDLE returns the server's communication handle if a connection has been established using the specified IP address.

Syntax

int **MBGETHANDLE**(IP_address)

Arguments

| | |
|------------|-------------------------|
| IP_address | IP address, as a string |
|------------|-------------------------|

Return Value

On success: Server's communication handle

On failure: -1

Example

This example shows how to verify the existence of a connection to a Modbus server device.

```
GLOBAL INT server_handle
!Checks if a connection with the specified IP has been established
server_handle = MBGETHANDLE("10.0.0.100")
IF server_handle <> -1
    DISP "There is an open connection with the specified IP"
END
STOP
```

Related ACSPL+ Commands

MBOPEN, MBCLOSE

4.7.3 MBCLOSE

Description

Closes an open Modbus TCP connection using the server's communication handle or IP address

Syntax

int **MBCLOSE**[/switches](server_communication_handle|server_ip_address)

Arguments

| | |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| server_communication_handle | The server's communication handle, as received from the MBOPEN function or The server's IP address (when used with the /s switch) |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|

Switches


| | |
|----|----------------------------------------------------------------------------------------------------------------|
| /s | Using the function with the /s switch specifies that the server IP address is provided, rather than the handle |
|----|----------------------------------------------------------------------------------------------------------------|

Return Value

On success: 0

On failure: A runtime error will occur

Comments



Closing a connection will stop all the active Modbus mapping requests.

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates how to open and close a connection to a Modbus server device using the device handle.

```
GLOBAL INT server_handle
!Opens a Modbus TCP connection with a server device
server_handle = MBOPEN("10.0.0.100")
MBCLOSE(server_handle) !Closes the connection
STOP
```

Example 2

This example demonstrates how to open and close a connection to a Modbus server device using the IP address.

```
GLOBAL INT server_handle
!Opens a Modbus TCP connection with a server device
```

```
server_handle = MBOPEN("10.0.0.100")
MBCLOSE/s("10.0.0.100") !Closes the connection
STOP
```

Related ACSPL+ Commands

MBOPEN, MBGETHANDLE

4.7.4 MBREADHREG

Description

The **MBREADHREG** function maps a Modbus server's holding register to a specified ACSPL+ variable. The value in the register is then read to the variable, either once or at a specified interval.

Syntax

For Scalar variables

MBREADHREG[/switches] server_handle, mapped_scalar, starting_address[, request_frequency]

For Arrays

MBREADHREG/a server_handle, mapped_array, array_index, starting_address[, number_of_elements, request_frequency]

Arguments

| | |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| server_handle | The server's communication handle (received from the MBOPEN function) |
| mapped_scalar/mapped_array | A valid ACSPL+ variable name (can be either static or standard, and scalar or array) |
| array_index (if /a switch is used) | Starting index for the mapping |
| starting_address | The starting address of the register to be mapped to the specified ACSPL+ variable. |
| number_of_elements | (Optional) The number of elements that will be mapped (sequentially from the starting address). Element type depends on the switch that has been used. For example, the /f switch specifies that each element is regarded as a single-precision floating-point number (32 bits, 2 registers). |
| request_frequency | (Optional) Specifies the Modbus request frequency (in milliseconds). The request is sent every "request_frequency" period. Default value: 10 milliseconds Minimum value: 5 milliseconds |

Switches

| Switch Name | Description |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /s | Indicates a one-time read (as opposed to reading the value from the client at the requested frequency). The function copies the value from the server's register to the mapped variable. |
| /2 | The data read should be regarded as integer type (16 bit). |
| /f | The data read should be regarded as float type (32 bit). |
| /d | The data read should be regarded as double type (64 bit). |
| /a | The mapped variable is an array. |

Return Value

On success: A unique request ID

On failure: A runtime error will occur

Comments

- > If the mapping request tries to read a location that does not exist on the server, a runtime error will occur (the user will know if they tried to access an invalid address), and the request will not be created.
- > If an error occurs during the Modbus mapping process, the mapping will stop, and the **MBERR**(request_id) will hold the error code. Also, the request will become inactive. To remove or restore a Modbus request, see the **MBUNMAP** and **MBCLEAR** functions.
- > The request ID can be used to check if an error has occurred for the request using the **MBERR**(request_id). Also, it can be used to unmap a specific request. (The request ID is unique per request).
- > If no switch is specified, the data to be mapped is regarded as 32-bit (2 registers) integer value.
- > Each element is converted to the format of the user's mapped variable type. For example, if the user mapped an Integer variable using the /f suffix (each element from the server device is regarded as a single-precision floating-point value), the value of the element is converted into a two's-complement integer representation before it is stored.
- > Loss of precision can occur when converting different data types. For example, reading a real (floating point) value to an integer element will result in truncation, which may cause a loss of precision.
- > A runtime error will occur if the specified mapped_variable length is incompatible with the specified number_of_elements.
- > The maximum number of registers that can be mapped is 32. (32 shorts, 16 integers, 16 floats, or 8 doubles).
- > The maximum number of mapping requests is 32 per server device.

- > If repeated timeout errors occur during the mapping process, it is recommended to use a single-time mapping. Such errors may indicate that the server device is having problems keeping up with the request rate. (This might be required for Festo AG & Co. KG and Beckhoff I/O devices.)

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates the mapping of 2 holding registers to the client's ACSPL+ global scalar variable, where the value read is treated as an integer number.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 0           !First holding register address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADHREG server_handle, mapped_scalar, starting_address
  STOP
```

Example 2

This example demonstrates the mapping of 2 holding registers to the client's ACSPL+ global scalar variable, where the value read is treated as a single-precision floating-point number.

```
D-Buffer:
  GLOBAL STATIC REAL mapped_scalar
Buffer 1:
  INT server_handle, starting address, request_id
  starting_address = 0           !First holding register address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADHREG/f server_handle, mapped_scalar, starting_
address
  STOP
```

Example 3

This example demonstrates the mapping of 10 holding registers to the client's ACSPL+ global array, where each element, is treated as a single-precision floating-point number (2 registers).

```
D-Buffer:
  GLOBAL STATIC REAL mapped_array(20)
Buffer 1:
  INT server_handle, starting_address, starting_index, number_of_
elements, request_id
  starting_index = 10
  starting_address = 0           !First holding register address
  number_of_elements = 5
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADHREG/fa server_handle, mapped_array, starting_
```

```
index, starting_address, number_of_elements
STOP
```

Example 4

This example demonstrates a one-time mapping of 2 holding registers to the client’s ACSPL+ global scalar variable, where the value read is treated as a single-precision floating-point number.

```
D-Buffer:
GLOBAL STATIC REAL mapped_scalarBuffer 1:
INT server_handle, starting_address, request_id
starting_address = 0 !First holding register address
server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
request_id = MBREADHREG/fs server_handle, mapped_scalar, starting_
address
STOP
```

Figure 4-2 illustrates the mapping of two Modbus device registers to a 32-bit little-endian variable.

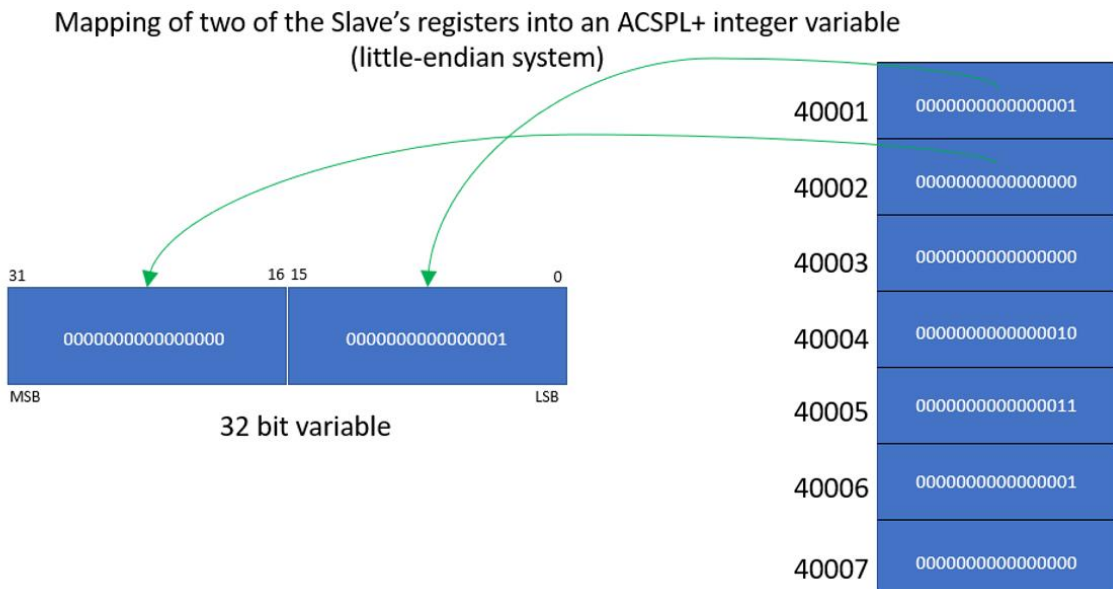


Figure 4-2. Example Mapping

Related ACSPL+ Commands

MBOPEN, MBCLOSE, MBWRITEREG, MBWRITECOIL

4.7.5 MBREADIREG

Description

The **MBREADIREG** function is used to map a Modbus Server’s input register to a specified ACSPL+ variable. The value in the register is then read to the variable, either once or at a specified interval.

Syntax

MBREADIREG[/switches] server_handle, mapped_scalar, starting_address[, request_frequency]

MBREADIREG/a server_handle, mapped_array, array_index, starting_address[, number_of_elements, request_frequency]

Arguments

| | |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| server_handle | The server's communication handle (received from the MBOPEN function) |
| mapped_scalar / mapped_array | A valid ACSPL+ variable name (can be either static or standard, and scalar or array) |
| array_index (if the /a switch is used) | Starting index for the mapping |
| starting_address | The starting address of the register to be mapped to the specified ACSPL+ variable. |
| number_of_elements | (Optional) The number of elements that will be mapped (sequentially from the starting address). The element type depends on the switch that is used. For example, the /f switch specifies that each element is regarded as a single-precision floating-point number (32 bits, 2 registers). |
| request_frequency | (Optional) The Modbus request frequency (in milliseconds). The request is sent every "request_frequency" period. Default value: 10 milliseconds Minimum value: 5 milliseconds |

Switches

| Name | Description |
|------|-------------------------------------------------------------------------------------------------------------|
| /s | Indicates a one-time read. The function copies the value from the server's register to the mapped variable. |
| /2 | The data read should be interpreted as integer type (16 bit). |
| /f | The data read should be interpreted as float type (32 bit). |
| /d | The data read should be interpreted as double type (64 bit). |
| /a | The mapped variable is an array. |

Return Value

On success: A unique request ID

On failure: A runtime error will occur

Comments

- > If the mapping request tries to read a location that does not exist on the server, a runtime error will occur (the user will know if they tried to access an invalid address), and the request will not be created.
- > If an error occurs during the Modbus mapping process, the mapping will stop, and the **MBERR**(request_id) will hold the error code. Also, the request will become inactive. To remove or restore a Modbus request, see the **MBUNMAP** and **MBCLEAR** functions.
- > The request ID can be used to check if an error has occurred for the request using the **MBERR**(request_id). Also, it can be used to unmap a specific request. (The request ID is unique per request).
- > If no switch is specified, the data to be mapped is regarded as 32-bit (2 registers) integer value.
- > Each element is converted to the format of the user's mapped variable type. For example, if the user mapped an Integer variable using the /f suffix (each element from the server device is regarded as a single-precision floating-point value), the value of the element is converted into a two's-complement integer representation before it is stored.
- > Loss of precision can occur when converting different data types. For example, reading a real (floating point) value to an integer element will result in truncation, which may cause a loss of precision.
- > A runtime error will occur if the specified mapped_variable length is incompatible with the specified number_of_elements.
- > The maximum number of registers that can be mapped is 32. (32 shorts, 16 integers, 16 floats, or 8 doubles).
- > The maximum number of mapping requests is 32 per server device.
- > If repeated timeout errors occur during the mapping process, it is recommended to use a single-time mapping. Such errors may indicate that the server device is having problems keeping up with the request rate. (This might be required for Festo AG & Co. KG and Beckhoff I/O devices.)

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates the mapping of 2 input registers to the client's ACSPL+ global scalar variable, where the value read is interpreted as an integer number.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address
  starting_address = 0           !First input register address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADIREG server_handle, mapped_scalar, starting_address
STOP
```

Example 2

This example demonstrates the mapping of 2 input registers to the client's ACSPL+ global scalar variable, where the value read is interpreted as a single-precision floating-point number.

```
D-Buffer:
  GLOBAL STATIC REAL mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 0           !First input register address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADIREG/f server_handle, mapped_scalar, starting_
address
STOP
```

Example 3

This example demonstrates the mapping of 10 input registers to the client's ACSPL+ global array, where each element is interpreted as a single-precision floating-point number (2 registers).

```
D-Buffer:
  GLOBAL STATIC REAL mapped_array(20)
Buffer 1:
  INT server_handle, starting_address, starting_index, number_of_
elements, request_id
  starting_index = 10
  starting_address = 0           !First input register address
  number_of_elements = 5
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADIREG/fa server_handle, mapped_array, starting_
index, starting_address, number_of_elements
STOP
```

Example 4

This example demonstrates a one-time mapping of 2 input registers to the client's ACSPL+ global scalar variable, where the value read is interpreted as a single-precision floating-point number.

```
D-Buffer:
  GLOBAL STATIC REAL mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 0 !First input register address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADIREG/fs server_handle, mapped_scalar, starting_
address
STOP
```

Figure 4-3 illustrates the mapping of two Modbus device registers to a 32-bit little-endian variable.

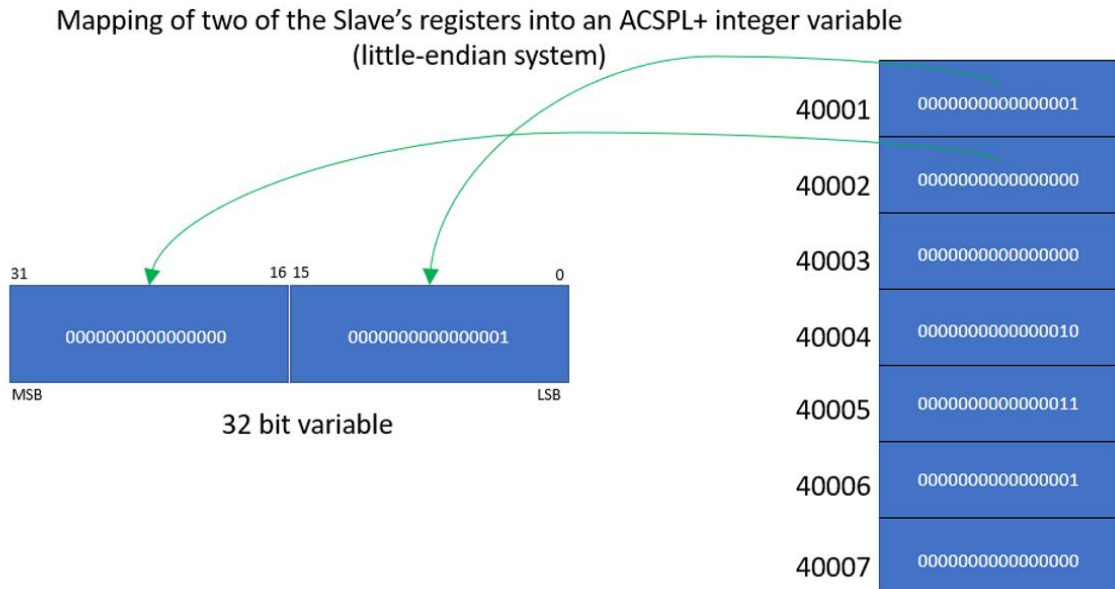


Figure 4-3. Example Mapping

4.7.6 MBWRITEHREG

Description

The **MBREADIREG** function is used to map a client's ACSPL+ variable to a Modbus server's register. The value in the client variable is then written to the server's register, either once or at a defined interval.

Syntax

MBWRITEHREG[/switches] server_handle, mapped_scalar, starting_address[, request_frequency]

MBWRITEHREG/a server_handle, mapped_array, array_index, starting_address[, number_of_elements, request_frequency]

Arguments

| | |
|-----------------------------------------------|---------------------------------------------------------------------------------------|
| server_handle | The server's communication handle (received from the MBOPEN function). |
| mapped_scalar / mapped_array | A valid ACSPL+ variable name (may be either static or standard, and scalar or array). |
| array_index (if the /a switch is used) | The array index from which mapping will begin. |
| starting_address | The starting address of the register to be mapped to the specified ACSPL+ variable. |

| | |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| number_of_elements | (Optional) The number of elements to be mapped (sequentially from the starting address). The element type depends on the switch used. For example, the /f suffix specifies that each element is interpreted as a single-precision floating-point number (32 bits, 2 registers). |
| request_frequency | The Modbus request frequency (in milliseconds). The request is sent every "request_frequency" period. Default value: 10 milliseconds Minimum value: 5 milliseconds |

Switches

| | |
|-----------|-----------------------------------------------------|
| /s | Generates a single write request |
| /2 | The data should be written as a 16-bit integer type |
| /f | The data should be written as a 32-bit float type |
| /d | The data should be written as a 64-bit double type |
| /a | The mapped variable is an array |

Return Value

On success: A unique request ID

On failure: A runtime error will occur

Comments

- > If the mapping request tries to write a location that does not exist on the server, a runtime error will occur (the user will know if they tried to access an invalid address), and the request will not be created.
- > If an error occurs during the Modbus mapping process, the mapping will stop, and the **MBERR**(request_id) will hold the error code. Also, the request will become inactive. To remove or restore a Modbus request, see the **MBUNMAP** and **MBCLEAR** functions.
- > The request ID can be used to check if an error has occurred for the request using the **MBERR**(request_id). Also, it can be used to unmap a specific request. (The request ID is unique per request).
- > If no switch is specified, the data to be mapped is regarded as 32-bit (2 registers) integer value.
- > Each element is converted to the format of the user's mapped variable type. For example, if the user mapped an Integer variable using the /f suffix (each element from the server device is regarded as a single-precision floating-point value), the value of the element is converted into a two's-complement integer representation before it is stored.

- > Loss of precision can occur when converting different data types. For example, reading a real (floating point) value to an integer element will result in truncation, which may cause a loss of precision.
- > A runtime error will occur if the specified mapped_variable length is incompatible with the specified number_of_elements.
- > The maximum number of registers that can be mapped is 32. (32 shorts, 16 integers, 16 floats, or 8 doubles).
- > The maximum number of mapping requests is 32 per server device.
- > If repeated timeout errors occur during the mapping process, it is recommended to use a single-time mapping. Such errors may indicate that the server device is having problems keeping up with the request rate. (This might be required for Festo AG & Co. KG and Beckhoff I/O devices.)

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates the mapping of 2 input registers to the client's ACSPL+ global scalar variable, where the mapped variable value is interpreted as an integer value.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  mapped_scalar = 100
  starting_address = 0 !First input register address
  server_handle = MBOpen("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITEHREG server_handle, mapped_scalar, starting_
address
STOP
```

Example 2

This example demonstrates the mapping of 2 input registers to the client's ACSPL+ global scalar variable, where the mapped variable value is interpreted as a single-precision floating-point number.

```
D-Buffer:
  GLOBAL STATIC REAL mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  mapped_scalar = 100.25
  starting_address = 0 !First input register address
  server_handle = MBOpen("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITEHREG/f server_handle, mapped_scalar, starting_
address
STOP
```

Example 3

This example demonstrates the mapping of 10 input registers to the client's ACSPL+ global array, where each element is interpreted as a single-precision floating-point number (2 registers).

```
D-Buffer:
  GLOBAL STATIC REAL mapped_array(20)
Buffer 1:
  INT server_handle, starting_address, starting_index, number_of_
elements, request_id
  starting_index = 10
  starting_address = 0 !First input register address
  number_of_elements = 5
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITEHREG/fa server_handle, mapped_array, starting_
index, starting_address, number_of_elements
STOP
```

Example 4

This example demonstrates a one-time mapping of 2 input registers to the client's ACSPL+ global scalar variable, where the mapped variable value is interpreted as a single-precision floating-point number.

```
D-Buffer:
  GLOBAL STATIC REAL mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  mapped_scalar = 1.1
  starting_address = 0 !First input register address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITEHREG/fs server_handle, mapped_scalar, starting_
address
STOP
```

Figure 4-4 illustrates the mapping of a 32-bit little-endian variable to two Modbus device registers.

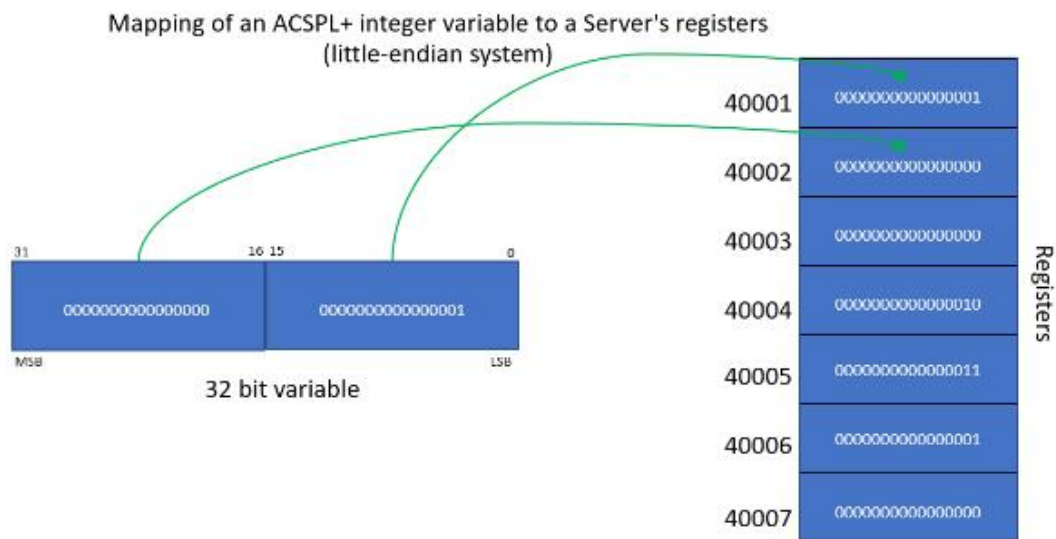


Figure 4-4. Example Mapping

4.7.7 MBREADCOIL

Description

The **MBREADCOIL** function is used to map a Modbus server coil to a specified ACSPL+ variable and read the state of the coil.

Syntax

MBREADCOIL[/switches] server_handle, mapped_scalar, starting_address[, number_of_coils, request_frequency]

MBREADCOIL/a server_handle, mapped_array, array_index, starting_address[, number_of_coils, request_frequency]

Arguments

| | |
|-----------------------------------------------|---------------------------------------------------------------------------------------|
| server_handle | The server's communication handle (received from the MBOPEN function) |
| mapped_scalar / mapped_array | A valid ACSPL+ variable name (may be either static or standard, and scalar or array) |
| array_index (if the /a switch is used) | The array index from which mapping will begin. |
| starting_address | The starting address of the coils to be mapped to the specified ACSPL+ variable. |
| number_of_coils | (Optional) The number of coils to map (sequentially from starting_address) |

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| request_frequency | (Optional) The Modbus request frequency (in milliseconds). The request is sent every "request_frequency" period. Default value: 10 milliseconds Minimum value: 5 milliseconds |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Switches

| | |
|-----------|---------------------------------------------------------------------------------------------------------------------|
| /s | A one-time read. Copies the value from specified coils to mapped_scalar . |
| /a | The mapped variable is an array |
| /b | Bitwise mapping. Coils are mapped to bits of the specified variable rather than to successive elements of an array. |

Return Value

On success: A unique request ID

On failure: A runtime error will occur

Comments

- > If the mapping request tries to read to or write from a location that does not exist on the server, a runtime error will occur (the user will know if they tried to access an invalid address), and the request will not be created.
- > If an error occurs during the Modbus mapping process, the mapping will stop and **MBERR** (request_id) will hold the error code. Also, the request will become inactive. To remove or restore a Modbus request, see the **MBUNMAP** and **MBCLEAR** functions.
- > The request ID can be used to check if an error has occurred for the request using **MBERR** (request_id). The request ID can also be used for unmapping a specific request. (The request ID is unique per request.)
- > A runtime error will occur if the specified mapped_scalar length is incompatible with the specified number_of_coils.
- > The maximum number of coils that can be mapped is 32.
- > The maximum number of mapping requests is 32 per server device.
- > If repeated timeout errors occur during the mapping process, it is recommended to use a single-time mapping. Such errors may indicate that the Server device is having problems keeping up with the request rate. (This might be required for Festo AG & Co. KG and Beckhoff I/O devices.)

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates mapping a server coil to the client's ACSPL+ global variable.

```
D-Buffer:
GLOBAL STATIC INT mapped_scalar
```

```

Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADCOIL server_handle, mapped_scalar, starting_address
STOP

```

Example 2

This example demonstrates mapping three server coils to the client's ACSPL+ global array.

```

D-Buffer:
  GLOBAL STATIC INT mapped_array(5)
Buffer 1:
  INT server_handle, starting_address, number_of_coils
  INT array_index, request_id
  array_index = 2
  number_of_coils = 3
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADCOIL/a server_handle, mapped_array, array_index,
starting_address, number_of_coils
STOP

```

Example 3

This example demonstrates bitwise mapping of 20 server coils to the client's ACSPL+ global variable.

```

D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, number_of_coils, request_id
  number_of_coils = 20
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADCOIL/b server_handle, mapped_scalar, starting_
address, number_of_coils
STOP

```

Example 4

This example demonstrates a one-time mapping from a server coil to the client's ACSPL+ global variable.

```

D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection

```

```
request_id = MBREADCOIL/s server_handle, mapped_scalar, starting_
address
STOP
```

Figure 4-5 illustrates the mapping of a server coil to an integer variable.

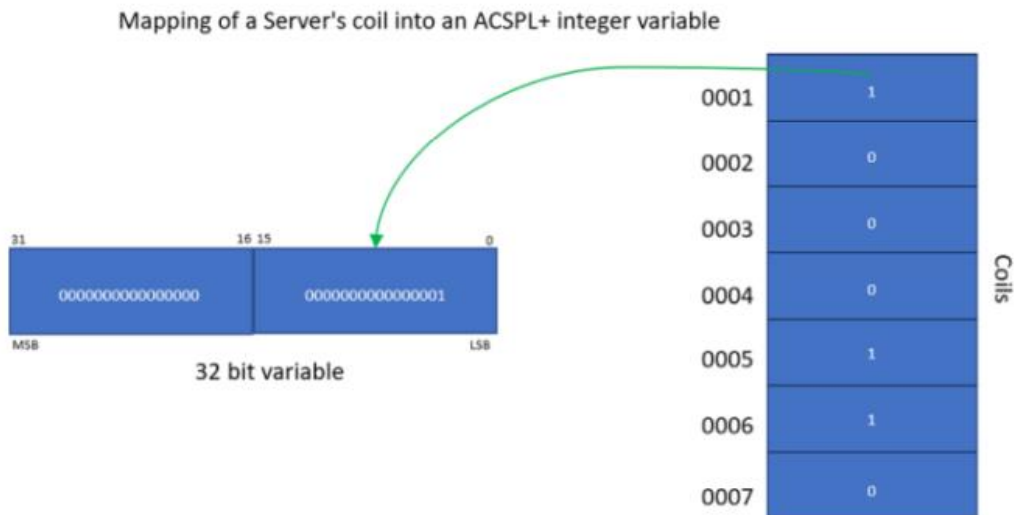


Figure 4-5. Example Mapping

4.7.8 MBWRITECOIL

Description

The **MBWRITECOIL** function is used to map a Modbus server coil to a specified ACSPL+ variable and write the variable's value to the coil.

Syntax

MBWRITECOIL[/switches] server_handle, mapped_scalar, starting_address[, number_of_coils, request_frequency]

MBWRITECOIL/a server_handle, mapped_array, array_index, starting_address[, number_of_coils, request_frequency]

Arguments

| | |
|-----------------------------------------------|--------------------------------------------------------------------------------------|
| server_handle | The server's communication handle (received from the MBOPEN function) |
| mapped_scalar / mapped_array | A valid ACSPL+ variable name (can be either static or standard, and scalar or array) |
| array_index (if the /a switch is used) | The array index from which mapping will begin. |

| | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| starting_address | The starting address of the coils to be mapped to the specified ACSPL+ variable. |
| number_of_coils | (Optional) The number of coils to map (sequentially from starting_address) |
| request_frequency | (Optional) The Modbus request frequency (in milliseconds). The request is sent every "request_frequency" period. Default value: 10 milliseconds Minimum value: 5 milliseconds |

Switches

| | |
|-----------|---------------------------------------------------------------------------------------------------------------------|
| /s | A one-time write. Copies the value from specified coils to mapped_scalar . |
| /a | The mapped variable is an array |
| /b | Bitwise mapping. Coils are mapped to bits of the specified variable rather than to successive elements of an array. |

Return Value

On success: A unique request ID

On failure: A runtime error will occur

Comments

- > If the mapping request tries to read from or write to a location that does not exist on the server, a runtime error will occur (the user will know if they tried to access an invalid address), and the request will not be created.
- > If an error has occurred during the mapping process, the mapping for the request will stop and MBERR(request_id) will hold the error code that occurred. Also, the request will become inactive. To remove or restore a Modbus request, see the **MBUNMAP** and **MBCLEAR** functions.
- > The request ID can be used to check if an error has occurred for the request using MBERR (request_id). Also, it can be used for unmapping a specific request. (The request ID is unique per request.)
- > A runtime error will occur if the specified mapped_scalar length is incompatible with the specified number_of_coils.
- > The maximum number of coils that can be mapped is 32.
- > The maximum number of mapping requests is 32 per server device.
- > If repeated timeout errors occur during the mapping process, it is recommended to use a single-time mapping. Such errors may indicate that the Server device is having problems keeping up with the request rate. (This might be required for Festo AG & Co. KG and Beckhoff I/O devices.)

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates mapping a server coil to the client's ACSPL+ global variable.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  mapped_scalar = 1
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITECOIL server_handle, mapped_scalar, starting_
address
STOP
```

Example 2

This example demonstrates mapping three server coils to the client's ACSPL+ global array.

```
D-Buffer:
  GLOBAL STATIC INT mapped_array(5)
Buffer 1:
  INT server_handle, starting_address, number_of_coils
  INT array_index, request_id
  number_of_coils = 3
  array_index = 2
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITECOIL/a server_handle, mapped_array, array_index,
starting_address, number_of_coils
STOP
```

Example 3

This example demonstrates bitwise mapping of 20 server coils to the client's ACSPL+ global variable.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, number_of_coils, request_id
  number_of_coils = 20
  starting_address = 1           !First Coil address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBWRITECOIL/b server_handle, mapped_scalar, starting_
address, number_of_coils
STOP
```

Example 4

This example demonstrates a one-time mapping from a server coil to the client's ACSPL+ global variable.

```
D-Buffer:
    GLOBAL STATIC INT mapped_scalar
Buffer 1:
    INT server_handle, starting_address, request_id
    starting_address = 1           !First Coil address
    server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
    request_id = MBWRITECOIL/s server_handle, mapped_scalar, starting_
    address
    STOP
```

Figure 4-6 illustrates the mapping of an integer variable to a server coil.

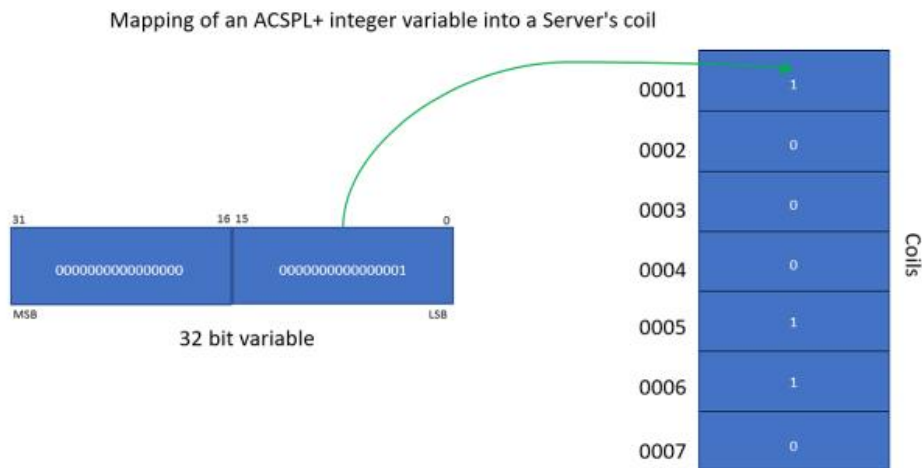


Figure 4-6. Example Mapping

4.7.9 MBREADDIN

Description

The **MBREADDIN** function maps a Modbus server discrete input to an ACSPL+ variable.

Syntax

MBREADDIN[/switches] server_handle, mapped_scalar, starting_address[, number_of_discrete_inputs, request_frequency]

MBREADDIN/a server_handle, mapped_array, array_index, starting_address[, number_of_discrete_inputs, request_frequency]

Arguments

| | |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| server_handle | The server's communication handle (received from the MBOPEN function) |
| mapped_scalar / mapped_array | A valid ACSPL+ variable name (can be either static or standard, and scalar or array) |
| array_index (if the /a switch is used) | The array index from which mapping will begin. |
| starting_address | The starting address of the discrete inputs to be mapped to the specified ACSPL+ variable. |
| number_of_discrete_inputs | (Optional) The number of discrete inputs to map (sequentially from starting_address) |
| request_frequency | (Optional) The Modbus request frequency (in milliseconds). The request is sent every "request_frequency" period. Default value: 10 milliseconds Minimum value: 5 milliseconds |

Switches

| | |
|-----------|-------------------------------------------------------------------------------------------------------------------------------|
| /s | A one-time read. Copies the value from the specified discrete input to mapped_scalar . |
| /a | The mapped variable is an array |
| /b | Bitwise mapping. Discrete inputs are mapped to bits of the specified variable rather than to successive elements of an array. |

Return Value

On success: A unique request ID

On failure: A runtime error will occur

Comments

- > If the mapping request tries to read to or write from a location that does not exist on the server, a runtime error will occur (the user will know if they tried to access an invalid address), and the request will not be created.
- > If an error has occurred during the mapping process, the mapping for the request will stop, and **MBERR**(request_id) will hold the error code that occurred. Also, the request will become inactive. To remove or restore a Modbus request, see the **MBUNMAP** and **MBCLEAR** functions.

- > The request ID can be used to check if an error has occurred for the request using **MBERR** (request_id). Also, it can be used for unmapping a specific request. (The request ID is unique per request.)
- > A runtime error will occur if the specified mapped_scalar length is incompatible with the specified number_of_discrete_inputs.
- > The maximum number of discrete inputs that can be mapped is 32.
- > The maximum number of mapping requests is 32 per server device.
- > If repeated timeout errors occur during the mapping process, it is recommended to use a single-time mapping. Such errors may indicate that the Server device is having problems keeping up with the request rate. (This might be required for Festo AG & Co. KG and Beckhoff I/O devices.)

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates mapping a server's discrete input to the client's ACSPL+ global variable.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 1           !First discrete input address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADDIN server_handle, mapped_scalar, starting_address
STOP
```

Example 2

This example demonstrates mapping three of the server's discrete inputs to the client's ACSPL+ global array.

```
D-Buffer:
  GLOBAL STATIC INT mapped_array(5)
Buffer 1:
  INT server_handle, starting_address, number_of_coils
  INT array_index, request_id
  array_index = 2
  number_of_discrete_inputs = 3
  starting_address = 1           !First discrete input address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADDIN/a server_handle, mapped_array, array_index,
starting_address, number_of_discrete_inputs
STOP
```


Example 3:

This example demonstrates bitwise mapping of 20 of the server's discrete inputs to the client's ACSPL+ global variable.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, number_of_discrete_inputs,
  request_id
  number_of_discrete_inputs = 20
  starting_address = 1           !First discrete input address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADDIN/b server_handle, mapped_scalar, starting_
  address, number_of_discrete_inputs
STOP
```

Example 4

This example demonstrates a one-time mapping from a server's discrete input to the client's ACSPL+ global variable.

```
D-Buffer:
  GLOBAL STATIC INT mapped_scalar
Buffer 1:
  INT server_handle, starting_address, request_id
  starting_address = 1           !First discrete input address
  server_handle = MBOPEN("10.0.0.100") !Opens a Modbus connection
  request_id = MBREADDIN/s server_handle, mapped_scalar, starting_
  address
STOP
```

4.7.10 MBUNMAP

Description

The **MBUNMAP** function unmaps a specific request using the request ID.

Syntax

MBUNMAP [request_id]

Arguments

| | |
|-------------------|---------------------------------------------------------------------------------------|
| request_id | (Optional) A request ID received from the MBREAD or MBWRITE function |
|-------------------|---------------------------------------------------------------------------------------|

Return Value

The number of requests that have been unmapped (0 or 1).

Comments

- > Unmapping a request will also clear its error code(if such exists) from the MBERR array.
- > If no request ID is specified, all active Modbus requests will be removed.

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates the unmapping of a specific Modbus request using the request ID.

```
MBUNMAP request_id
STOP
```

Example 2

This example demonstrates the unmapping of all active Modbus requests.

```
MBUNMAP
STOP
```

4.7.11 MBCLEAR

Description

The **MBCLEAR** function clears the error codes from the Modbus requests error array (**MBERR**) and reactivates the requests that experienced the error condition.

Syntax

MBCLEAR [request_id]

Arguments

| | |
|-------------------|----------------------------------------------------------------------------------------|
| request_id | (Optional) A request ID received from the MBREAD or MBWRITE function. |
|-------------------|----------------------------------------------------------------------------------------|

Return Value

None

Comments

If **request_id** is not specified, the entire Modbus requests error array(MBERR) will be cleared (set to 0) and all request that experienced errors will be reactivated.

See [Section 7.3, ACSPL+ Runtime Errors](#) for supported error codes

Example 1

This example demonstrates how to clear a Modbus error and reactivate the relevant request.

```
MBCLEAR request_id
STOP
```

Example 2

This example demonstrates how to clear all Modbus errors and reactivate all Modbus requests.

```
MBCLEAR
STOP
```

4.7.12 MBERR

Description

MBERR is an integer array with one element for each possible Modbus request ID (between 0 - 95). It holds the most recent Modbus error code that occurred for the request during the communication process.

Table 4-6. Modbus Error Codes

| Error Code | Description |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | No Error |
| 8001 | Illegal Function - Function code received in the query is not recognized or allowed by the server. |
| 8002 | Illegal Data Address - Data addresses of some or all the required entities are not allowed or do not exist in the server. |
| 8003 | Illegal Data Value - A value contained in the query data field is not an allowable value for the server. |
| 8004 | Server Device Failure - An unrecoverable error occurred while the server was attempting to perform the requested action. |
| 8005 | Acknowledge - Server has accepted the request and is processing it, but a long duration of time is required. This response is returned to prevent a timeout error from occurring in the client. |
| 8006 | Server Device Busy - Server is engaged in processing a long-duration command. The client should retry later. |
| 8012 | Transaction Identifier Mismatch – The received response transaction identifier did not match the expected value. |
| 8013 | Response Length Mismatch - The received response length did not match the expected length. |
| 8014 | Response Out of Bounds – The received response value is invalid. This may be due to the value being out of the allowed range, an attempt to write to a protected variable, etc. |
| 8015 | Timeout – No response has been received from the server device for the timeout period(5 seconds). The connection with the server device has been terminated. |
| 8016 | Connection Closed - The connection with the server device has been closed. |

Comments

MBERR can be cleared by unmapping the erroneous request using the **MBUNMAP** function or by activating the erroneous request using the **MBCLEAR** function.

Example

This example demonstrates monitoring the error state of a Modbus mapping request.

```
D-buffer:
    GLOBAL INT request_id
Buffer 1:
    request_id = MBWRITEHREG server_handle, mapped_scalar, starting_
address

ON MBERR(request_id)
DISP "An error has occurred for the request, request ID=",request_id
!Here we decide how we would like to handle this error, un-mapping
!the request, closing the communication channel, or clearing the
!fault and restore the request to an active state
.
.
.
MBCLEAR request_id !For example, clearing the error of the request,
!and restoring it to an active state

RET
STOP
```

Tag

403

Accessibility

Read-Only

4.7.13 #MBMAPREP

Description

The **#MBMAPREP** command displays a report of all the active Modbus connections and mapped variables.

Syntax

```
#MBMAPREP
```

Example

```
#MBMAPREP
    Modbus Server Address:10.0.0.103 (channel:18)
    =====
Server ID
=====
1

Endianness
=====
Little-Endian

Mapped Variables
=====
None

    Modbus Server Address:10.0.0.31 (channel:19)
    =====
Server ID
=====
1

Endianness
=====
Little-Endian

Mapped Variables
=====
1)Variable Name:bit_status | Type:Integer
   Mapping Type:Write Coils
   Starting Address:0
   Number of Coils:1
   Bit Mapping:False
   Request Frequency:5.000000 [milliseconds]
   Request State:Sleeping
   Request ID:64
   Request Error Code:0
```

4.8 Servo Processor Functions

SPiiPlus™, a proprietary ACS Motion Control Servo Processor (SP), executes the real-time tasks such as implementation of the real time control algorithms. Each SPiiPlus can control from two to eight axes (depending on the product). The SPiiPlus includes all the necessary peripherals that are needed

for a high performance axis control, such as encoder counters, Digital-to-Analog interface, smart inputs and outputs.

Servo Processor functions are used to read and monitor SP values.



The number of Servo Processors is model-dependent. Check the controller Hardware Guide for the specific SPiPlus model.

The Servo Processor functions are:

| Function | Description |
|------------------------|---------------------------------------------------------|
| GETSP | Reads a value from the specified SP address |
| GETSPA | Retrieves address of the SP variable specified by name. |
| GETSPV | Reads a value from the specified SP variable name |
| SETSP | Sets a value for the specified SP address |
| SETSPV | Sets a value for the specified SP variable name |

4.8.1 [GETSP](#)

Description

GETSP reads a value from the specified SP address.

The read value is treated as a signed integer number.

The minimal delay for the function response is more than three MPU cycles.

Syntax

GETSP(*int SP, int Address*)

Arguments

| | |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <i>SP</i> | The SP number in the system. Use <code>getconf(260, axis)</code> , where <code>axis</code> is axis in the system to get the SP number. |
| <i>Address</i> | Address within the SP. Get the address by using <code>getspa(SPnumber, varname)</code> command. |



For **Address** it is recommended using the memory address as obtained by [GETSPA](#).

Return Value

Value read from the specified SP address.

Error Conditions

The function causes an error if an SP number is specified other than 0–3, or if an illegal address is specified.

Example

```
REAL PAR_ADDRESS
PAR_ADDRESS=GETSPA(0,"PE(0)")    ! Get the memory address of PE(0) in SP#0
GETSP(0, PAR_ADDRESS)           ! The return value is the PE(0) in SP#0
```

4.8.2 GETSPA

Description

GETSPA retrieves the SP address of the specified SP variable.

Syntax

int **GETSPA**(*SP_number*, "*SP_variable*")

Arguments

| | |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>SP_number</i> | The SP number in the system. |
| <i>SP_variable</i> | String representing the name of an SP variable. In order to avoid unexpected results, place SP_variable in quotes. |

Return Value

Address of the variable in the SP memory, or -1 if the variable does not exist. The return value can be used in **GETSP** and **SETSP**.

Example

```
XX= GETSPA(0,"axes[0].PE")      !Retrieve the address Position error variable
of axis 0 in SP 0.
```

4.8.3 GETSPV



This function is obsolete and is replaced by **GETSP**.

4.8.4 SETSP



The **SETSP** function is for advanced users only. Misuse of this function may damage the drive.

Description

SETSP writes a value to the specified SP address.

Syntax

SETSP(*int SP_number, int Address, value*)

Arguments

| | |
|------------------|-----------------------------------------------------------------------------|
| SP_number | The SP number in the system. |
| Address | Address within the SP. Look up addresses by using getspa function |
| value | The value to write to the address. |

Error Conditions

Error 3126, Illegal SP number.

Example

```
REAL PAR_ADDRESS
PAR_ADDRESS=GETSPA(0, "axes[0].params[0].SLVKP")      !Gets the memory
address of axes[0].params[0].SLVKP
SETSP(0, PAR_ADDRESS, 1000)                          !Changes the value of the axes
[0].params[0].SLVKP in SP 0 to 1000.
```

4.8.5 SETSPV



This function is obsolete and is replaced by [SETSP](#).

4.9 Signal Processing Functions

The Signal Processing functions are:

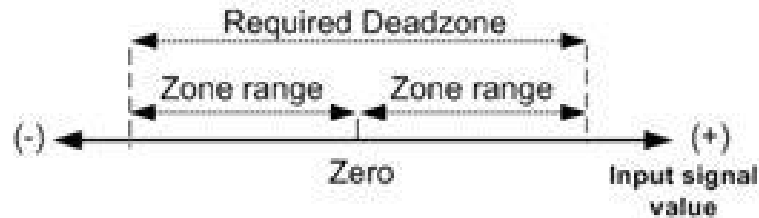
| Function | Description |
|--------------------------|-------------------------------------------------------------------------------------|
| COPY | The function copies one array to another. |
| DEADZONE | Implements dead-zone routine |
| DSIGN | Implements a dynamic version of the standard SIGN function. |
| DSTR | Converts a string to an integer array. |
| EDGE | Returns 1 on positive edge of x |
| INP | Reads data characters from the specified channel and stores them into integer array |

| Function | Description |
|----------|--------------------------------------------------------------------------------------------------------------------|
| INTGR | Implements an integrator with DEADZONE and SAT . |
| LAG | Provides delayed switching on argument change (anti-bouncing effect) |
| MAP | Implements a table-defined function with constant step |
| MAPB | Implements a one-dimensional uniform B-spline interpolation |
| MAPN | One-dimensional non-uniform linear interpolation (replaces the obsolete MAPBY1 and MAPBY2 functions) |
| MAPS | One-dimensional non-uniform linear interpolation (replaces the obsolete MAPBY1 and MAPBY2 functions) |
| MAPNB | One-dimensional non-uniform b-spline |
| MAPNS | One-dimensional non-uniform Catmull-Rom spline |
| MAPS | One-dimensional uniform Catmull-Rom spline |
| MAP2 | Implements a table-defined function with two arguments and constant step along each argument. |
| MAP2B | Two-dimensional uniform b-spline |
| MAP2N | Two-dimensional non-uniform linear interpolation (replaces the obsolete MAP2FREE function) |
| MAP2NB | Two-dimensional non-uniform b-spline |
| MAP2NS | Two-dimensional non-uniform Catmull-Rom spline |
| MAP2S | Two-dimensional uniform Catmull-Rom spline |
| MATCH | Calculates axis position that matches current reference position of the same axis with zero offset. |
| RAND | Implements a random number generator. |
| ROLL | Calculates a result rolled-over to within one pitch. |
| SAT | Implements a saturation characteristic |

4.9.1 DEADZONE

Description

DEADZONE returns values based on a defined analog or other input signal with a defined symmetrical or asymmetrical dead zone around zero.



DEADZONE is useful for anti-bouncing effect.

Syntax

DEADZONE (*input_signal*, *zone1*[,*zone2*])

Arguments

| | |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <i>input_signal</i> | Any real number or integer expression |
| <i>zone1</i> | Any real number or integer expression for a symmetrical deadzone. See Example 1 . |
| <i>zone2</i> | Any real number or integer expression, required for an asymmetrical deadzone requires two values. See Example 2 . |

Return Value

DEADZONE returns a **real number** as follows:

If $-zone < input_signal < zone$, return value = 0

If $input_signal < -zone$, return value = $input_signal + zone$

If: $input_signal > zone$, return value = $input_signal - zone$

Error Conditions

A negative zone range returns Error 3045, Numerical Error in Standard Function.

Examples

Example 1

Symmetrical Dead Zone

In this example, illustrated in [Figure 4-7](#), *input_signal* ranges from -20: +20, and creates a symmetrical dead zone from -10: +10.

```
return_value = DEADZONE(input_signal,10)
```

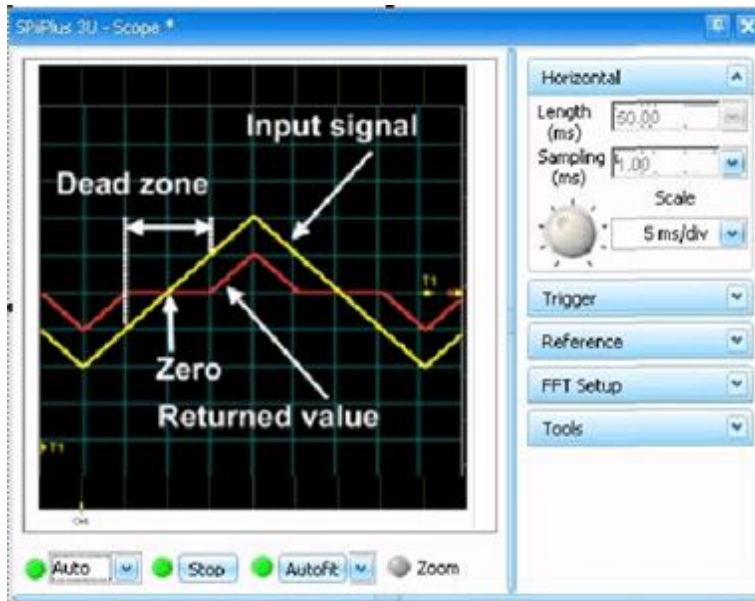


Figure 4-7. Symmetrical Dead Zone Example

Example 2

Asymmetrical Dead Zone

In this example, illustrated in [Figure 4-8](#), input_signal ranges from -20: +20, and creates an asymmetrical dead zone from -5 – +15.

```
return_value = DEADZONE(input_signal,-5,10)
```

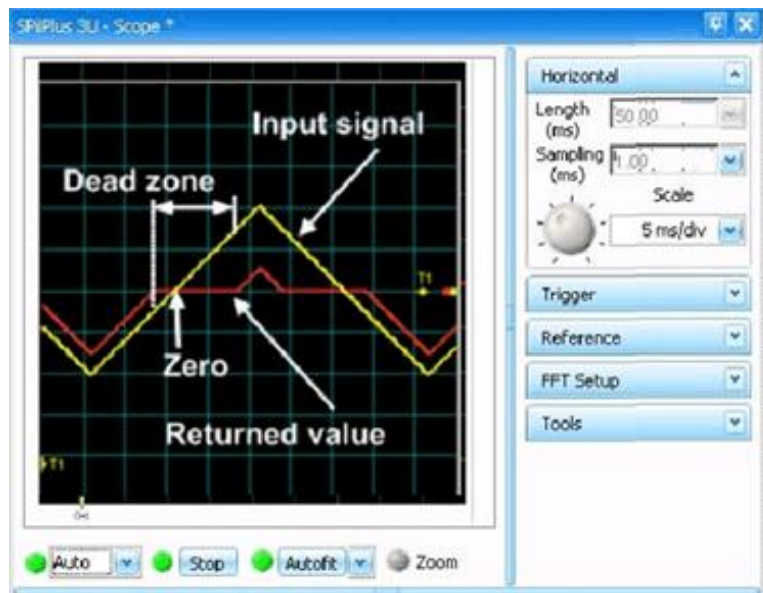


Figure 4-8. Asymmetrical Dead Zone Example

4.9.2 *DSIGN*

Description

DSIGN returns values between -1 to 1 based on a defined input variable with defined delay time and ramp time.

Syntax

real **DSIGN**(*X*, *Delay_Time*, *Ramp_Time*)

Arguments

| | |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <i>X</i> | Input variable name declared as real or real expression. |
| <i>Delay_Time</i> | Real number that determines on zero crossing of <i>X</i> , continues the previous return value for defined delay time in msec. |
| <i>Ramp_Time</i> | Real number that controls the rate of the returned value change between -1 to 1 (and vice versa), defined in msec. |

Return Values

0: when the input variable is < zero while **Ramp Time** = 0.

1: when the input variable is > zero while **Ramp Time**= 0.

-1 to 1: when the input variable changes between positive to negative (or vice versa), while **Ramp Time** is > 0.

Error Conditions

Delay time and ramp time should be positive. If any of them becomes negative, Error 3045, Numerical Error in Standard Function appears.

Example

```
REAL XX, YY
YY = DSIGN(XX, 50, 100)
STOP
```

Figure 4-9 illustrates the **DSIGN** function.

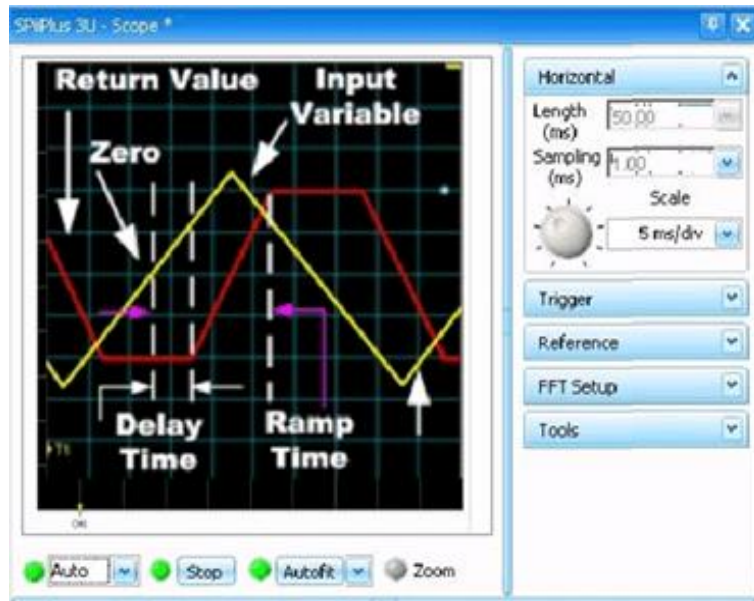


Figure 4-9. DSIGN Function Example

4.9.3 DSTR

Description

DSTR converts a string to an integer array based on an ASCII transformation code. The function decomposes a string to characters and assigns the characters to the sequential elements of the variable array.

Each ASCII character is represented as its numerical value and stored in a separate element of the array.

Syntax

```
int DSTR(string, array_name, [start_index,] [number])
```

Arguments

| | |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>string</i> | String of characters enclosed in double quotation marks. |
| <i>array_name</i> | User-defined integer array. |
| <i>start_index</i> | Index in the array from where the transformed numbers will be placed. If <i>start_index</i> is omitted, the assignment starts from the first element of the array. |
| <i>number</i> | Number of characters from the string to be transformed. If <i>number</i> is omitted, the function assigns all characters of the string. If <i>number</i> is specified, the function assigns the specified number of characters. In both cases the assignment stops when the last array element is reached. |

Return Value

The number of actually transformed characters.

Example

In the example below **DSTR** transforms each of "ACS-Motion_Control" characters to its corresponding numeric value and assigns them to a user defined array "BIBI" (each character to a separate array member. BIBI content is as follows:

```
65 67 83 45 84 69 67 72 56 48
```

```
GLOBAL INT BIBI(10)
DSTR("ACS-Motion_Control",BIBI)
STOP
```

4.9.4 EDGE

Description

EDGE returns 0 or 1, based on a defined input variable. **EDGE** is mainly useful in PLC implementation when an action must be taken once a condition becomes true.

Syntax

```
real EDGE(X)
```

Arguments

| | |
|----------|----------------------------------------------|
| X | Input real variable name or real expression. |
|----------|----------------------------------------------|

Return Value

EDGE returns 1 when the input variable changes from 0 to (+1) or from 0 to (-1) until the input variable changes to other values. The function returns 0 in all other cases.

Error Conditions

None

Example

```
GLOBAL REAL XX,XI,YY           !Declares three local variables
XX=-5 ; XI=1                   !Assigns values to the variables XX and XI.
WHILE 1                         !Run the following routine forever
  XX=XX+XI
  YY = EDGE (XX)
  IF (XX>5) | (XX<-5) XI=-XI    !Keeps XX between -5 to 5.
END !Ends IF END               !Ends WHILE
STOP                            !Ends program
```

Figure 4-10 illustrates the **EDGE** function.

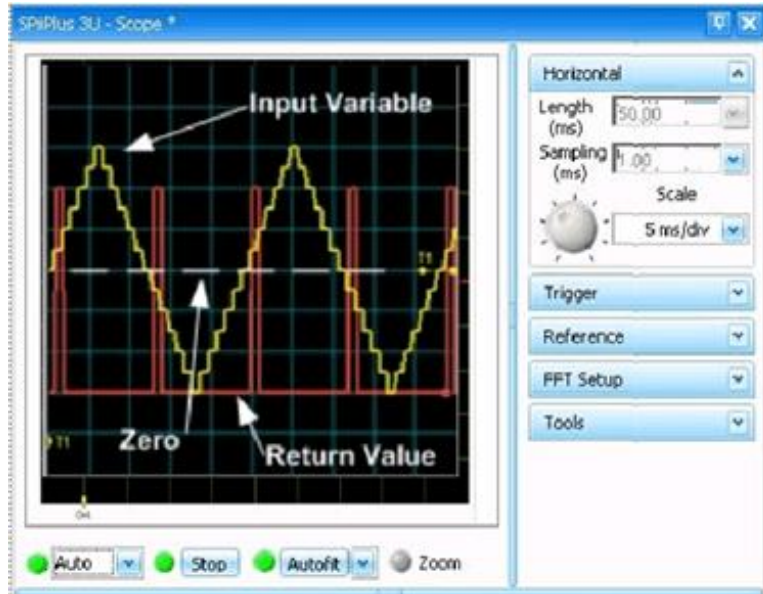


Figure 4-10. EDGE Function Example

4.9.5 INTGR

Description

INTGR returns an integrator with optional deadzone and saturation limits.

Syntax

real **INTGR**(*X*, *Deadzone*, *Min*, *Max*[, *Initial_Value*])

Arguments

| | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>X</i> | Real variable name or real expression. |
| <i>Deadzone</i> | A real number. When the Deadzone value falls into the range of - Deadzone to + Deadzone , INTGR retains its previous value as if <i>X</i> = 0. |
| <i>Min</i> | A real number representing the low integrator saturation limit. |
| <i>Max</i> | A real number representing the high integrator saturation limit. |
| <i>Initial_Value</i> | A real number setting the initial value of the return value. (optional) |

Return Value

INTGR returns a value in the range from *Min* to *Max*.

Error Conditions

None

Example

Input variable *XX* is changing between -20 to +20.

```

GLOBAL REAL YY, XX           !Defines global real variables.
REAL ZZ                     !Defines increment variable as real
MARK:                       !Set GOTO line
ZZ=0.1; XX=-20             !Set variable values
WHILE XX<=20               !Set upper limit for XX
    XX=XX+ZZ               !Set increment for XX
    YY=INTGR (XX, 5, -10, 10, 20)
END                          !End WHILE
WHILE XX>=-20              !Set lower limit for XX
    XX=XX-ZZ               !Set increment for XX
    YY=INTGR (XX, 5, -10, 10, 20)
END                          !End WHILE
GOTO MARK                   !GOTO MARK to create a program loop
STOP                        !End program
    
```

Figure 4-11 illustrates the **INTGR** function.

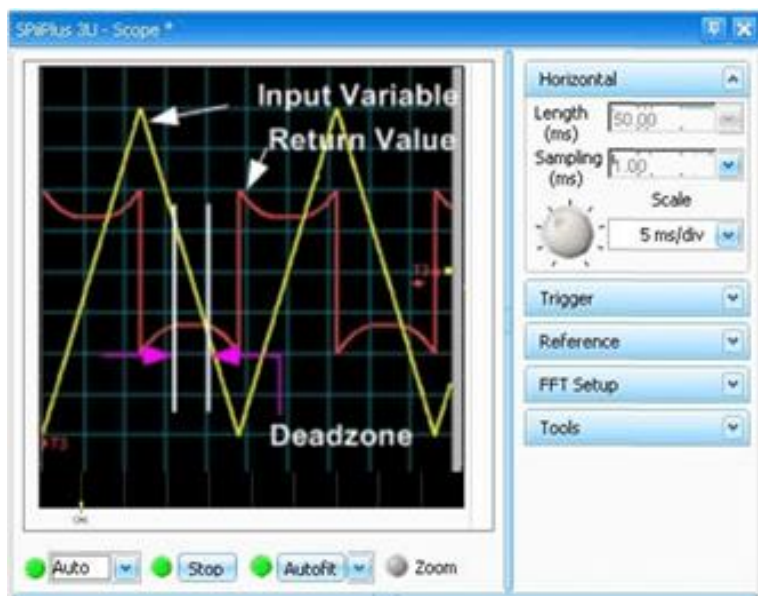


Figure 4-11. INTGR Function Example

4.9.6 LAG

Description

LAG returns values based on a defined input signal with defined up delay or down delay. This function is useful for an anti-bouncing effect.

Syntax

LAG(*X*, *up_delay*, *down_delay*)

Arguments

| | |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X | Real variable name or real expression. Although LAG accepts any real or integer argument for X , the typical use implies a logical X argument, for example, an integer variable or expression that supplies only 0 or 1. If X supplies values other than 0 or 1, LAG treats any non-zero value as 1. |
| p_delay | A real number representing the delay on the positive edge in msec. |
| down_delay | A real number representing the delay on the negative edge in msec. |

Return Value

1: when **X** remains non-zero for at least **up_delay** msec.

0: when **X** remains zero for at least **down_delay** msec.

Error Conditions

None

Example

The example here demonstrates **LAG** by generating a reference signal **XX** that changes from 0 to 1 every 100 msec, and based on **XX** and **YY**, implements a lag of 50msec and 20msec.

```

INT XX,YY,XT                                !Define three integers
XT=TIME; XX=0                               !XT equals the time elapsed from
                                             !startup. XX equal zero.

WHILE 1                                     !Run an infinite routine
IF TIME-XT>100; XX=^XX; XT=TIME; END      !Switch the value of XX - from ON
to
                                             !OFF
YY=LAG (XX,20,50)                          !YY is a returned value that is delayed
                                             !20msec after the transfer of XX from
                                             !0 to 1 and delayed 50msec after the
                                             !transfer of XX from 1 to 0

END                                         !Ends WHILE
STOP                                       !Ends program

```

Figure 4-12 illustrates the **LAG** function.

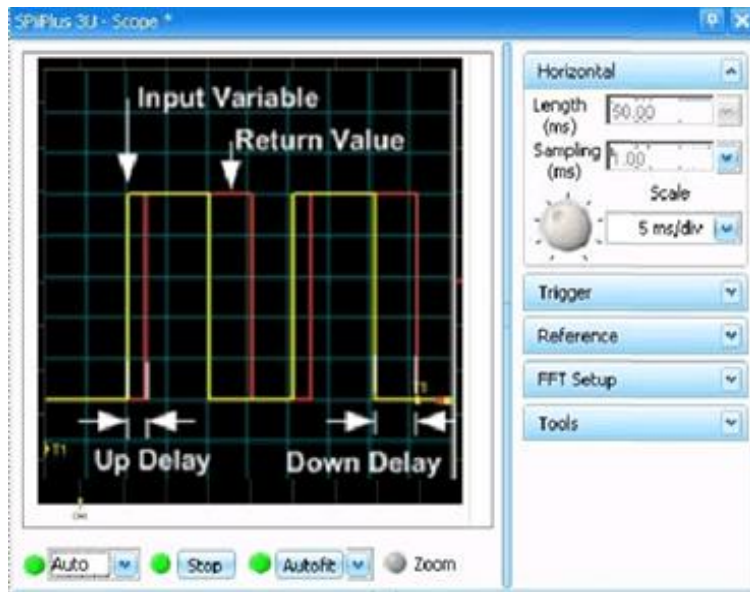


Figure 4-12. LAG Function Example

4.9.7 Interpolation Functions

There are a number of interpolation methods between points among them are Linear and Spline.

4.9.7.1 Linear interpolation

Linear interpolation of function $y(x)$ between distant points a and b are calculated as follows:

$$y(x_i) = \frac{y(b) - y(a)}{b - a} x_i + y(a) - a \frac{y(b) - y(a)}{b - a}, x_i \in [a, b]$$

4.9.7.2 Spline interpolation

A spline is a special function defined piecewise by polynomials. The spline is a piecewise polynomial function spread over an interval $[a, b]$ consists of polynomial pieces, such that:

$$a = t_0 < t_1 < t_2 < \dots < t_{k-2} < t_{k-1} = b$$

The points t_j are called **knots**. The vector is called a **knot vector** for the spline. If the knots are equidistantly distributed in the interval $[a, b]$, we say the spline is **uniform**, otherwise we say it is **non-uniform**.

In many cases functional dependence between two or more values cannot be expressed as an analytic formula. The most common presentation of those functions is a table of function values in specific points.

For example, a machine axis was graduated with an external laser interferometer. The result of graduation is a table like the following:

| | | | | | | | | |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Commanded position (x) | 100 | 200 | 300 | 400 | 500 | 600 | 700 | ... |
| Actual position (p) | 103 | 199 | 294 | 402 | 500 | 598 | 705 | ... |

The table defines a functional dependence $p=f(x)$ that cannot be expressed analytically.

The argument values for x in the definition table are knots, and the function values for p are control points.

A 3rd order polynomial spline provides an approximation of the table-driven function that can provide the function value not only in the knots, but at any point. Between each two knots the spline is expressed as:

$$p = a_0 + a_1x + a_2x^2 + a_3x^3 = \sum_{i=0}^3 a_i x^i$$

where coefficients a_0, a_1, a_2, a_3 have different values at different intervals.

The SPiiPlus controller also supports two-dimensional splines. In this case, the definition table is a two-dimensional matrix. Knot points are defined for two arguments x and y , and the matrix contains corresponding p values. Knot values divide the XY plane into rectangular cells. The matrix defines the function values in the cell vertices. Within each cell, the interpolating spline is expressed as:

$$p = \sum_{i,j=0}^3 a_{ij} x^i y^j$$

Many different spline approximations can be provided for one definition table. The SPiiPlus controller supports two kinds of splines: Catmull-Rom and B-Splines (see description below).

If the distance between the knots in the table is constant, the spline is called uniform. On the contrary, a non-uniform spline corresponds to a table that contains function values in arbitrary points. However, the definition table always arranges the knot values in ascending order, so that $x_i \leq x_{i+1}$.

All knot points constitute the **definition range** of the spline. Figure 4-13 illustrates definition range of a function defined with six non-uniform knots:

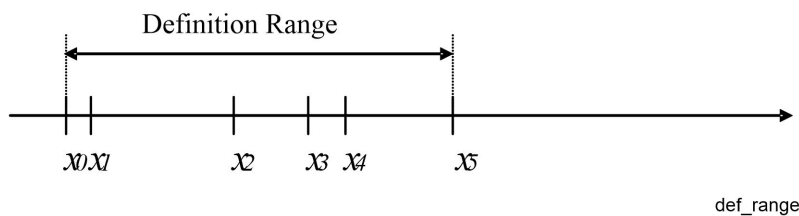


Figure 4-13. Spline Definition Range

In a two-dimensional case, definition range is a rectangular area, as illustrated in [Figure 4-14](#):

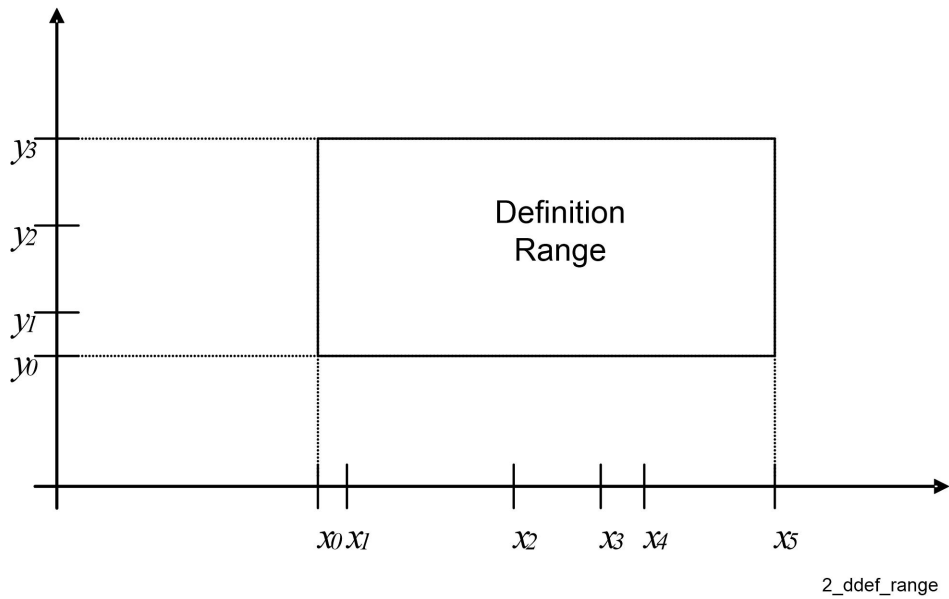


Figure 4-14. Two-Dimensional Spline Definition Range

One-Dimensional Catmull-Rom Spline

Assume a definition table provides N control points $p_0, p_1, p_2, \dots, p_{N-1}$ in knots $x_0, x_1, x_2, \dots, x_{N-1}$.

The Catmull-Rom construction process is described as follows:

- > At each internal knot x_i ($1 \leq i \leq N-2$) calculate the derivative:

$$v_i = (p_{i+1} - p_{i-1}) / (x_{i+1} - x_{i-1})$$
- > At the first and last knots assume zero derivative: $v_0 = v_{N-1} = 0$.
- > In interval i ($1 \leq i \leq N-2$), build a 3rd order polynomial $p=f(x)$ that satisfies four bound conditions $p_i, v_i, p_{i+1}, v_{i+1}$.
- > Beyond the definition range the spline is defined as follows:

$$p = p_0 \text{ if } x < x_0$$

$$p = p_{N-1} \text{ if } x > x_{N-1}$$

[Figure 4-15](#) illustrates a Catmull-Rom spline that interpolates a 5-component definition table.

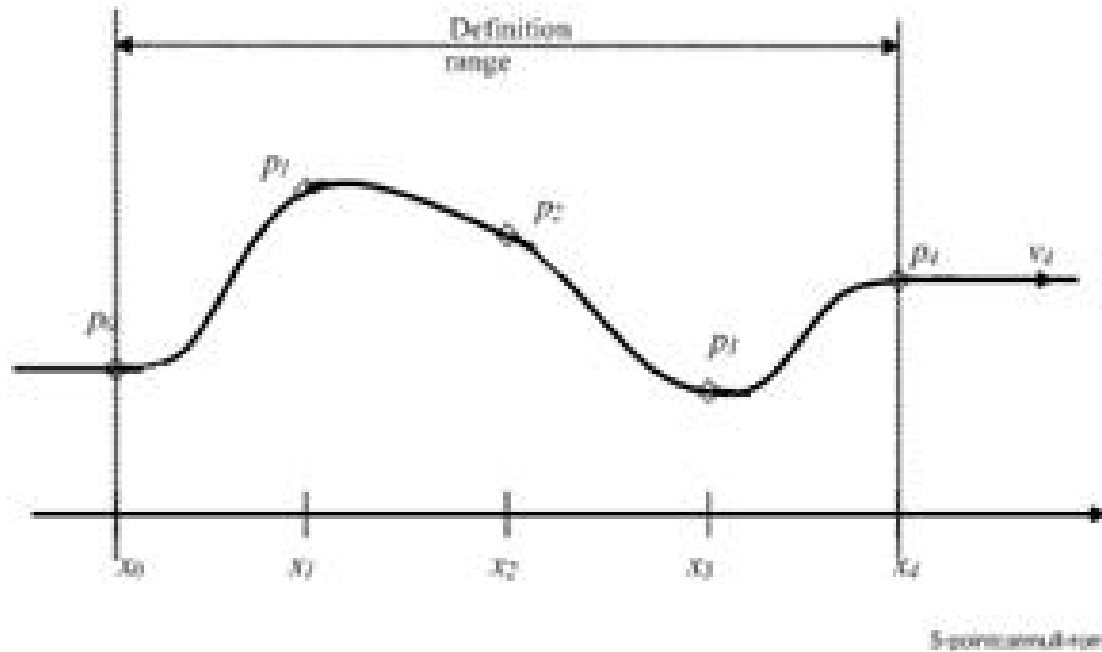


Figure 4-15. 5-Point Catmull-Rom Spline

Features of the Catmull-Rom spline:

- > The spline is C^1 -continuous, meaning the curve and its first derivative are continuous functions. The second derivative has discontinuities in the knot points.
- > The curve interpolates all control points; meaning that the curve goes through each control point.
- > At internal control point number i , the first derivative vector is parallel to the line connecting control points $i-1$ and $i+1$.
- > At the first and the last control points, the first derivative is zero.
- > The spline yields a constant value equal to p_0 on the interval from $-\infty$ to x_0 .
- > The spline yields constant value equal to p_{N-1} on the interval from x_{N-1} to $+\infty$.

One-Dimensional B-spline

Assume a definition table provides N control points $p_0, p_1, p_2, \dots, p_{N-1}$ in knots $x_0, x_1, x_2, \dots, x_{N-1}$.

Unlike the Catmull-Rom spline, a B-Spline does not go through the control points. Actually, it approximates the control points as illustrated in [Figure 4-16](#).

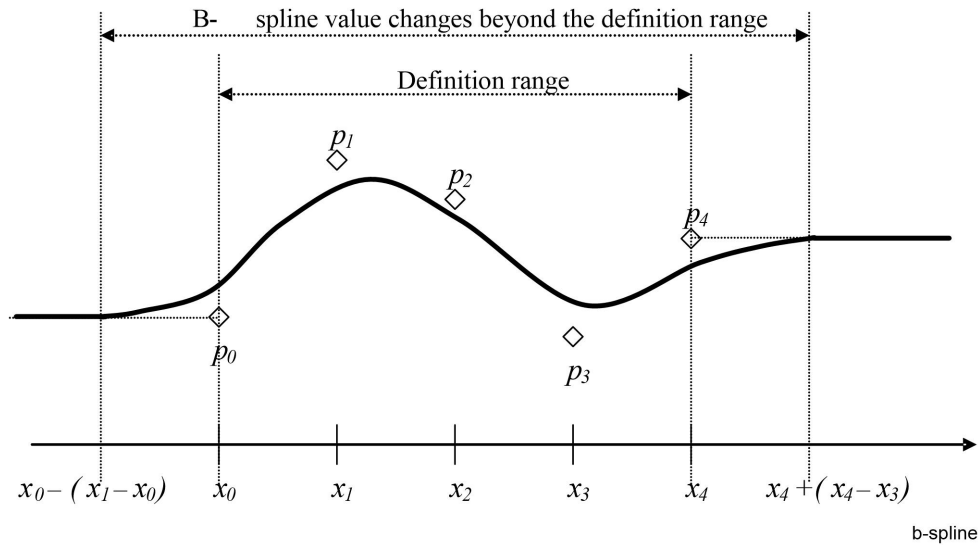


Figure 4-16. B-Spline - Approximation of Points

Compared to a Catmull-Rom spline, a B-Spline generates a smoother curve. Used in the controller, a B-spline provides continuous velocity and acceleration; where Catmull-Rom spline provides continuous velocity only, and acceleration may change by jumping at the control points.

Features of the B-Spline:

- > The spline is C^2 -continuous, meaning the curve, its first and second derivatives are all continuous functions.
- > The curve approximates the control points; and does not go through each control point.
- > The spline yields changing value in the interval from $x_0 - (x_1 - x_0)$ to $x_{N-1} + (x_{N-1} - x_{N-2})$.
- > The spline yields constant value equal to p_0 in the interval from $-\infty$ to $x_0 - (x_1 - x_0)$.
- > The spline yields constant value equal to p_{N-1} in the interval from $x_{N-1} + (x_{N-1} - x_{N-2})$ to $+\infty$.

Not-passing the control points is not always a drawback. If values $p_0, p_1, p_2 \dots p_{N-1}$ are obtained from some measuring process, the values include measuring error that has a stochastic component. B-Spline tends to filter out the stochastic error, thereby improving overall accuracy.

Two-Dimensional Splines

The SPiiPlus NT controllers support two-dimensional Catmull-Rom and B-Splines.

A two-dimensional spline approximates a definition table that provides $N \times M$ control points $p_{00}, p_{01}, \dots, p_{0,M-1}, p_{10}, p_{11}, \dots, p_{N-1,M-1}$ on the grid defined by knots $x_0, x_1, x_2 \dots x_{M-1}$ and $y_0, y_1, y_2 \dots y_{N-1}$.

A two-dimensional spline is defined as tensor product of two one-dimensional splines. Two-dimensional splines share many features with the corresponding one-dimensional splines, for example:

| Catmull-Rom Spline Surface | B-Spline Surface |
|-----------------------------|-----------------------------|
| C^1 -continuous | C^2 -continuous |
| Interpolates control points | Approximates control points |

A section of a two-dimensional spline surface along any direction provides a curve, which is a corresponding one-dimensional file. For example, a two-dimensional Catmull-Rom spline is cut on the grid line that corresponds to knot y_2 . The section is a one-dimensional Catmull-Rom spline built upon control points $p_{20}, p_{21}, p_{22}, \dots, p_{2,M-1}$.

The behavior of a two-dimensional spline beyond the definition range is more complex than in the case of a one-dimensional spline. Figure 4-17 illustrates the Catmull-Rom spline beyond the definition range:

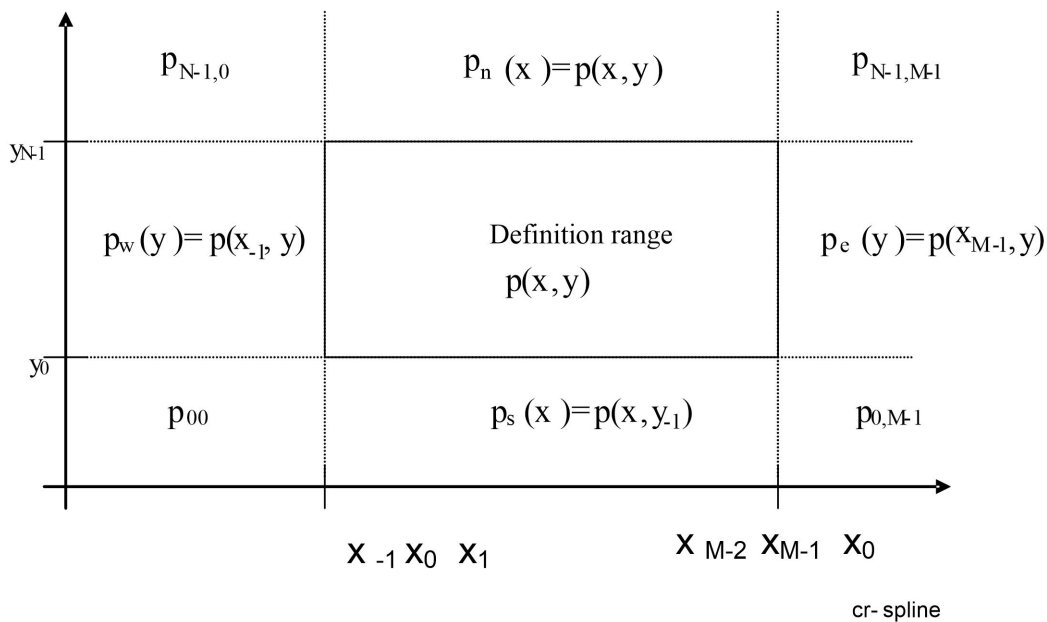


Figure 4-17. Catmull-Ron Spline Beyond the Definition Range

The behavior of Catmull-Rom depends on the area as follows:

- > Within definition range: function of two arguments $p(x, y)$.
- > Southeast to definition range: constant value equal to control point p_{00} .
- > Northeast to definition range: constant value equal to control point $p_{N-1,0}$.
- > Northwest to definition range: constant value equal to control point $p_{N-1,M-1}$.
- > Southwest to definition range: constant value equal to control point $p_{0,M-1}$.
- > South to definition range: function of x , $p_s(x) = p(x, y_0)$
- > North to definition range: function of x , $p_n(x) = p(x, y_{N-1})$
- > West to definition range: function of y , $p_w(y) = p(x_0, y)$
- > East to definition range: function of y , $p_e(y) = p(x_{M-1}, y)$

Similar to one-dimensional B-spline, two-dimensional B-spline has an extended range of result change. To specify the extended range, let us define four artificial knot points:

$$x_{-1} = x_0 - (x_1 - x_0)$$

$$x_M = x_{M-1} + (x_{M-1} - x_{M-2})$$

$$y_{-1} = y_0 - (y_1 - y_0)$$

$$y_N = y_{N-1} + (y_{N-1} - y_{N-2})$$

Then, the extended range spans from x_{-1} to x_M and from y_{-1} to y_N .

Area map of a B-spline is similar to the Catmull-Rom spline, but the extended range takes place of the definition range:

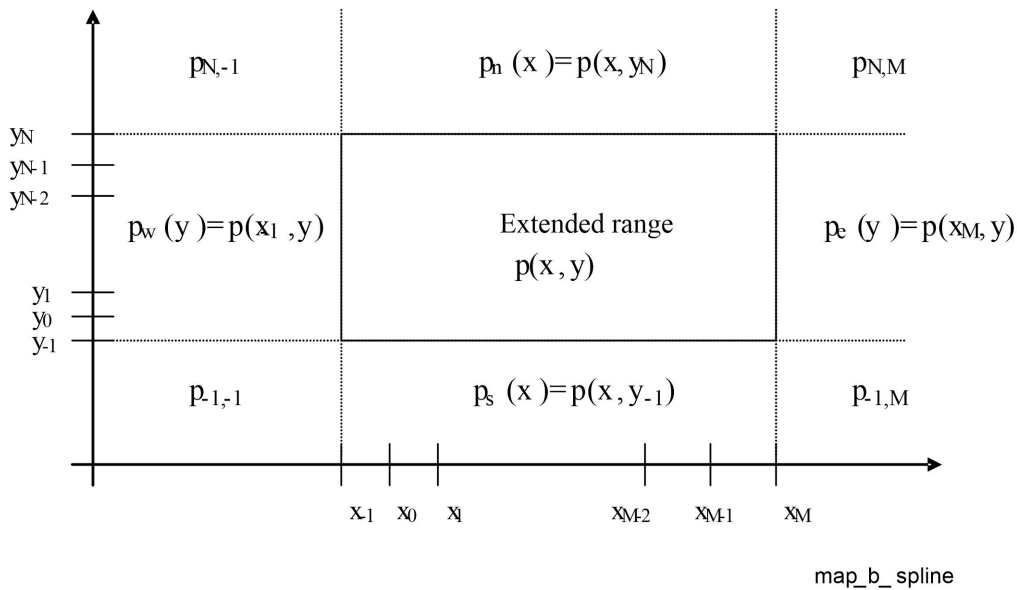


Figure 4-18. B-Spline Map

4.9.7.3 MAP

Description

MAP returns a value from an array of points per input variable value, base value of the variable and fixed defined intervals of the variable. The return values between the array points are linearly interpolated. **MAP** is useful for creating a correction table for mechanical error compensation.

Syntax

MAP(*X*, *array*, *base*, *step*)

Arguments

| | |
|---------------------|-------------------------------------------------------------------------------------------------|
| <i>X</i> | Real variable name or real expression. |
| <i>array</i> | The name of a real one-dimensional array that specifies points |
| <i>base</i> | A real number representing the value of XX that corresponds to the first point in array. |

| | |
|-------------|----------------------------------------------------------------------------------------------------------|
| step | A real number representing the value of XX that defines fixed intervals between the array points. |
|-------------|----------------------------------------------------------------------------------------------------------|

Return Value

MAP returns a linearly interpolated value from the **array** for each value of **XX**.

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined **array** size is less than the defined number of **array** points.
- > Error 3113, The step in the table is zero or negative, when the **step** argument is zero or negative.

Example

```

REAL ARRAY(5);REAL XX; REAL YY; REAL XI
!Defines array ARRAY, and variables XX, YY, and XI.
!----- Assign values to ARRAY array -----
ARRAY(0)= -10; ARRAY(1)=3; ARRAY(2)=5; ARRAY(3)=14; ARRAY(4)=12
!Assign values to ARRAY
XX=-20; XI=0.1 !Assign values to XX and XI
WHILE XX<20 !Set conditions
XX=XX+XI
YY=MAP(XX,ARRAY,-10,5) !MAP command where:
!XX is a variable name
!ARRAY is a one-dimensional point array
!-10 is the value of XX that corresponds to base
!point!5 is the value of XX that defines fixed intervals
!between the ARRAY points.
END !Ends mapping
STOP !Ends program
    
```

Table 4-7 shows the XX and YY values.

Table 4-7. MAP Array

| | | | | | | | | | |
|-------|-----|-----|------|----|---|----|----|-----|-----|
| XX | -20 | -10 | -7 | -5 | 0 | 5 | 10 | 15 | 20 |
| ARRAY | N/A | -10 | N/A | 3 | 5 | 14 | 12 | N/A | N/A |
| YY | -10 | -10 | -2.2 | 3 | 5 | 14 | 12 | 12 | 12 |

Figure 4-19 illustrates illustration of this **MAP** example on the Scope.

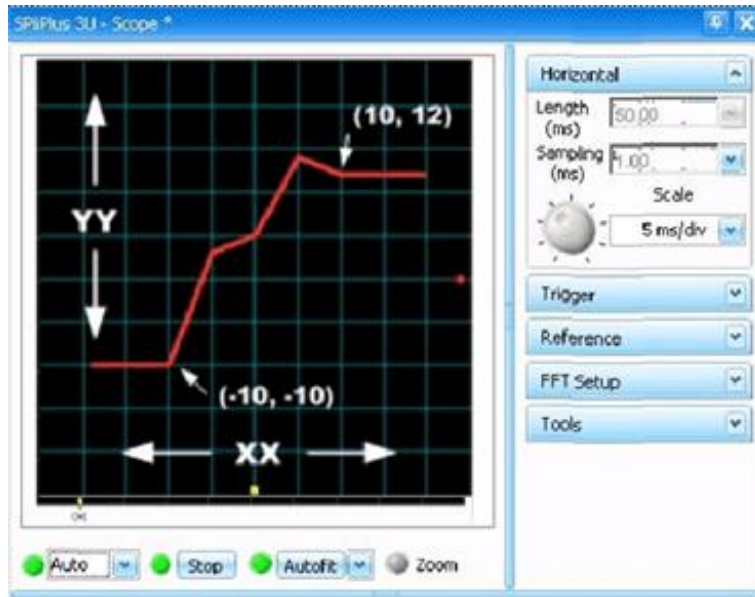


Figure 4-19. MAP Example on the Scope

4.9.7.4 MAPB

Description

MAPB returns a value from an array of points according to an input variable value, base value of the variable and fixed defined intervals of the variable. The return values between the array points are interpolated by a third order B-spline. **MAPB** is useful for creating a correction table for mechanical error compensation.

Syntax

MAPB(*XX*, *array*, *base*, *step*)

Arguments

| | |
|--------------|-----------------------------------------------------------------------------------------------------------------|
| <i>XX</i> | Real variable name or real expression. |
| <i>array</i> | The name of a real one-dimensional array that specifies points |
| <i>base</i> | A real number representing the value of <i>XX</i> that corresponds to the first point in <i>array</i> . |
| <i>step</i> | A real number representing the value of <i>XX</i> that defines fixed intervals between the <i>array</i> points. |

Return Value

MAPB returns the third order B spline interpolated value from the *array* according to the value of variable *XX*.

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined **array** size is less than the defined number of **array** points.
- > Error 3113, The step in the table is zero or negative, when the **step** argument is zero or negative.

Example

```

REAL ARRAY(5);REAL XX; REAL YY; REAL XI
!Defines array ARRAY, and variables XX,YY, and XI
!----- Assign point values to ARRAY array -----
---
ARRAY(0)= -10; ARRAY(1)=3; ARRAY(2)=5; ARRAY(3)=14; ARRAY(4)=12
XX=-20; XI=0.1 !Assign values to XX and XI
WHILE XX<20 !Set conditions
XX=XX+XIYY=MAPB(XX,ARRAY,-10,5) !MAPB command where:!XX is a variable
name
!ARRAY is a one-dimensional point array
!-10 is the value of XX that corresponds to base
!point
!5 is the value of XX that defines fixed
intervals
!between the ARRAY points.
END !Ends mapping
STOP !Ends program
    
```

Table 4-8 shows the XX and YY values.

Table 4-8. MAPB Array

| | | | | | | | | | |
|-----------|-----|-----------|-----------|------|------|-------|-------|-----|-----|
| XX | -20 | -10 | -7 | -5 | 0 | 5 | 10 | 15 | 20 |
| ARRA Y | N/A | -10 | N/A | 3 | 5 | 14 | 12 | N/A | N/A |
| YY | -10 | - 7.83 | - 2.45 | 1.16 | 6.16 | 12.33 | 12.33 | 12 | 12 |

Figure 4-20 illustrates this **MAPB** example on the Scope.

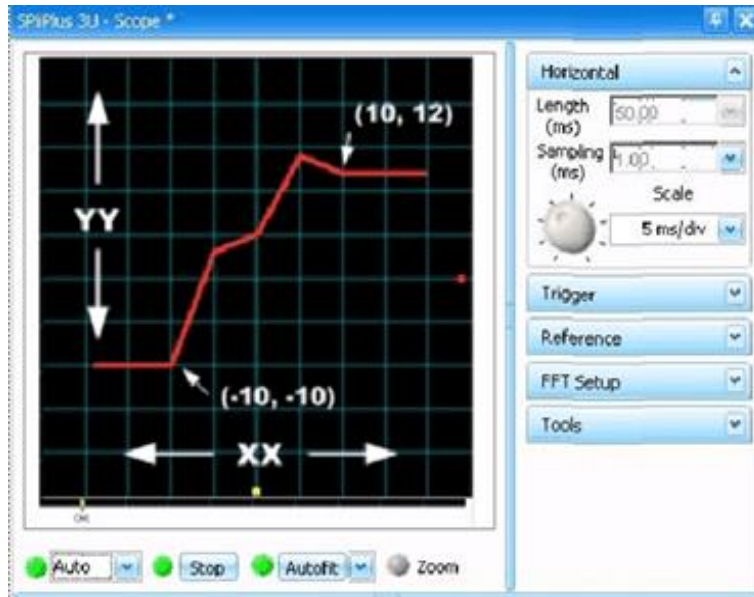


Figure 4-20. MAPB Example on the Scope

4.9.7.5 MAPN

Description

MAPN returns a value from an array of points according to the value of an input variable, and a look-up array. The return values between the array points are linearly interpolated. **MAPN** is useful for creating a correction table for mechanical error compensation.

Syntax

MAPN(*XX*, *X_table*, *Y_table*)

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------------|
| XX | Real variable name or real expression. |
| X_table | The name of a real one-dimensional array that specifies the values of XX that correspond to the point array. |
| Y_table | The name of a real one-dimensional array that specifies points. |

Return Value

MAPN returns a linearly interpolated value from the array according to the value variable **XX**.



- > To obtain a valid return value from the MAPN function after the Y_table array is changed, the buffer must be recompiled before calling the MAPN function.
- > If a MAPN function is called without recompiling the buffer, then the return value is calculated according to the values from the Y_table array that was provided to the function after the last compilation.

Error Conditions

The function detects the following error conditions:

- > The **Xtable** or **Ytable** argument has the wrong dimension
- > The arguments in **Xtable** are not arranged in ascending order.

Example

```
REAL Xtable(5);REAL Ytable(5);REAL XX; REAL YY; REAL XI!Defines arrays
Xtable, Ytable, and variables XX,
!YY, and XI.
!----- Assign values to Xtable array -----
Xtable(0)= -10;Xtable(1)=-3;Xtable(2)=6;Xtable(3)=12;Xtable(4)=14
!----- Assign values to Ytable array -----
Ytable(0)= -10;Ytable(1)=3;Ytable(2)=5;Ytable(3)=14;Ytable(4)=4
XX=-20; XI=0.1           !Assigns values to XX and XI.
WHILE XX<20             !Set conditions.
XX=XX+XI
YY=MAPN(XX,Xtable,Ytable)!MAPN where:   !XX is the defined variable.
                               !Xtable is a one-dimensional point array that
                               !specifies the values of XX corresponding to the
                               !point array.
                               !Ytable is a one-dimensional array that specifies
                               !points.
END                       !Ends mapping
STOP                      !Ends program
```

Table 4-9 shows the XX and YY values.

Table 4-9. MAPN Array

| | | | | | | | | | |
|-------|-----|-----|-------|----|---|----|----|-----|-----|
| XX | -20 | -10 | -7 | -3 | 6 | 12 | 14 | 15 | 20 |
| ARRAY | N/A | -10 | N/A | -3 | 6 | 12 | 12 | N/A | N/A |
| YY | -10 | -10 | -4.42 | -3 | 6 | 12 | 14 | 14 | 14 |

Figure 4-21 illustrates this **MAPN** example on the Scope.

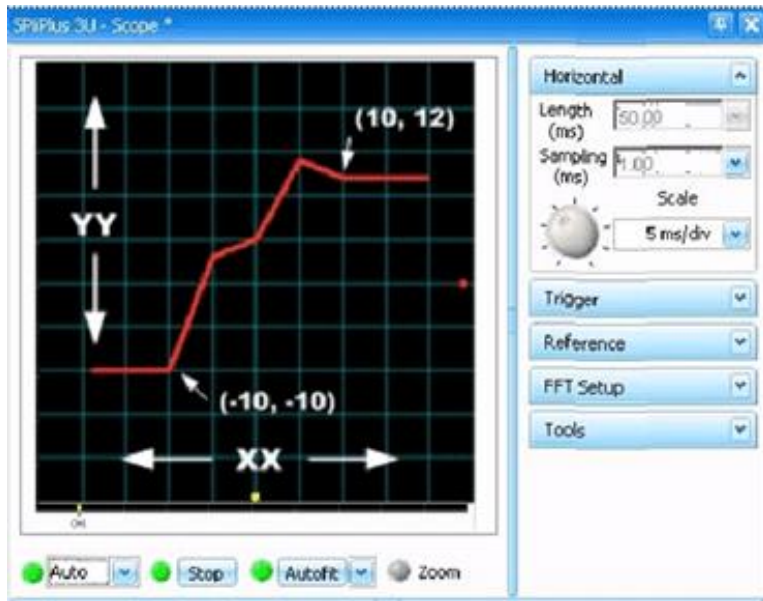


Figure 4-21. MAPN Example on the Scope

4.9.7.6 MAPNB

Description

MAPNB returns a value from a pre-defined control point array based on a defined variable and a look-up table. The return values between the table intervals are interpolated according to a third order B-spline. **MAPNB** is useful for creating a correction table for mechanical error compensation.

Syntax

MAPNB(*XX*, *X_table*, *Y_table*)

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------------|
| <i>XX</i> | Real variable name or real expression. |
| <i>X_table</i> | The name of a real one-dimensional array that specifies the values of XX that correspond to the point array. |
| <i>Y_table</i> | The name of a real one-dimensional array that specifies points. |

Return Value

MAPNB returns the third order B spline interpolated value from the **Y_table** per variable **XX** value.



- > To obtain a valid return value from the MAPNB function after the Y_table array is changed, the buffer must be recompiled before calling the MAPNB function.
- > If a MAPNB function is called without recompiling the buffer, then the return value is calculated according to the values from the Y_table array that was provided to the function after the last compilation.

Error Conditions

The function detects the following error conditions:

- > The **X_table** or **Y_table** argument has the wrong dimension
- > The arguments in **X_table** are not arranged in ascending order sequence.

Example

```

REAL Xtable(5);REAL Ytable(5);REAL XX; REAL YY; REAL XI
!Defines arrays Xtable, Ytable, and variables XX,
!YY, and XI.
!----- Assign values to Xtable array -----
Xtable(0)= -10; Xtable(1)=-3; Xtable(2)=6; Xtable(3)=12; Xtable(4)=14
!----- Assign values to Ytable array -----
Ytable(0)= -10; Ytable(1)=3; Ytable(2)=5; Ytable(3)=14; Ytable(4)=4
XX=-20; XI=0.1           !Assigns values to XX and XI.
WHILE XX<20             !Set conditions.
  XX=XX+XI
  YY=MAPNB(XX,Xtable,Ytable)!MAPNB where:
                           !XX is the defined variable.
                           !Xtable is a point array that specifies the values
                           !of XX corresponding to the point array.
                           !Ytable is an array that specifies Y points.
END                       !Ends mapping
STOP                      !Ends program

```

Table 4-10 shows the XX and YY values.

Table 4-10. MAPNB Array

| | | | | | | | | | |
|-----------|-----|-----------|-----------|-----------|------|------|----|-------|-----|
| XX | -20 | -10 | -7 | -3 | 6 | 12 | 14 | 15 | 20 |
| ARRA Y | N/A | -10 | N/A | -3 | 6 | 12 | 12 | N/A | N/A |
| YY | -10 | - 8.02 | - 4.65 | - 0.41 | 7.64 | 9.23 | 5 | 4.124 | 4 |

Figure 4-22 illustrates this **MAPNB** example on the Scope.

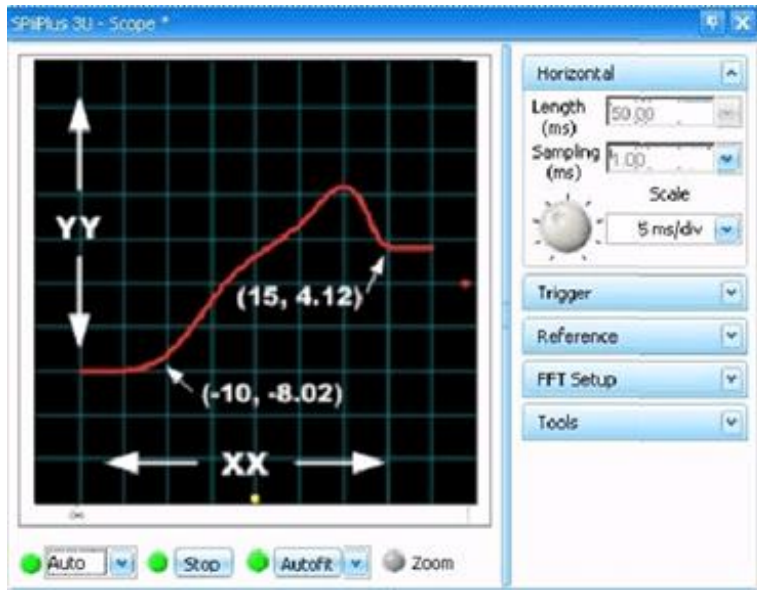


Figure 4-22. MAPNB Example on the Scope

4.9.7.7 MAPNS

Description

MAPNS returns a value from an array of points according to the value of an input variable, and a look-up array. The return values between the array points are interpolated according to a third order Catmull-Rom spline. **MAPNS** is useful for creating a correction table for mechanical error compensation.

Syntax

MAPNS(*XX*, *X_table*, *Y_table*)

Arguments

| | |
|----------------|---------------------------------------------------------------------------------------------------------------------|
| <i>XX</i> | Real variable name or real expression. |
| <i>X_table</i> | The name of a real one-dimensional array that specifies the values of XX that correspond to the point array. |
| <i>Y_table</i> | The name of a real one-dimensional array that specifies points. |

Return Value

MAPNS returns the third order Catmull-Rom spline interpolated value from the **X_table** per variable **XX** value.



- > To obtain a valid return value from the MAPNS function after the Y_table array is changed, the buffer must be recompiled before calling the MAPNS function.
- > If a MAPNS function is called without recompiling the buffer, then the return value is calculated according to the values from the Y_table array that was provided to the function after the last compilation.

Error Conditions

The function detects the following error conditions:

- > The **X_table** or **Y_table** argument has the wrong dimension
- > The arguments in **X_table** are not arranged in ascending order sequence.

Example

```

REAL Xtable(5);REAL Ytable(5);REAL XX; REAL YY; REAL XI
!Defines arrays Xtable, Ytable, and variables XX,
!YY, and XI
!----- Assign values to Xtable array -----
Xtable(0)= -10; Xtable(1)=-3; Xtable(2)=6; Xtable(3)=12; Xtable(4)=14
!----- Assign values to Ytable array -----
Ytable(0)= -10; Ytable(1)=3; Ytable(2)=5; Ytable(3)=14; Ytable(4)=4
XX=-20; XI=0.1                !Assigns values to XX and XI.
WHILE XX<20                    !Set conditions.
XX=XX+XI
YY=MAPNS (XX,Xtable,Ytable)    !MAPNS command where:
                                !XX is the defined variable.
                                !Xtable is a one-dimensional point array that
                                !specifies the values of XX corresponding to the
                                !point array.
                                !Ytable is a one-dimensional array that specifies
                                !points.
END                               !Ends mapping
STOP                              !Ends program

```

Table 4-11 shows the XX and YY values.

Table 4-11. MAPNS Array

| | | | | | | | | | |
|-------|-----|-----|-------|----|---|----|----|-----|-----|
| XX | -20 | -10 | -7 | -3 | 6 | 12 | 14 | 15 | 20 |
| ARRAY | N/A | -10 | N/A | -3 | 6 | 12 | 12 | N/A | N/A |
| YY | -10 | -10 | -4.42 | -3 | 6 | 12 | 14 | 14 | 14 |

Figure 4-23 illustrates this **MAPNS** example on the Scope.

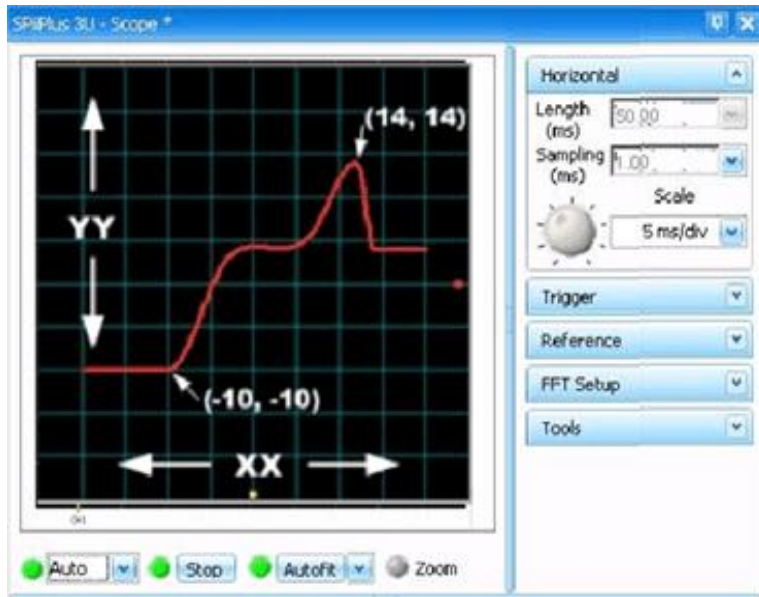


Figure 4-23. MAPNS Example on the Scope

4.9.7.8 MAPS

Description

MAPS returns a value from an array of points according to the value of an input variable, the base value of the variable, and the fixed defined intervals. The return values between the array-points are interpolated according to a third order Catmull-Rom spline. **MAPS** is useful for creating a correction table for mechanical error compensation.

Syntax

MAPS(*XX*, *array*, *base*, *step*)

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------|
| <i>XX</i> | Real variable name or real expression. |
| <i>array</i> | The name of a real one-dimensional array that specifies points |
| <i>base</i> | A real number representing the value of XX that corresponds to the first point in array. |
| <i>step</i> | A real number representing the value of XX that defines fixed intervals between the array points. |

Return Value

MAPS returns the third order Catmull-Rom spline interpolated value from the array according to the value of **XX**.

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined **array** size is less than the defined number of **array** points.
- > Error 3113, The step in the table is zero or negative, when the **step** argument is zero or negative.

Example

```

REAL ARRAY(5);REAL XX; REAL YY; REAL XI
                                !Defines ARRAY a 5 element array, XX, YY, and XI
!----- Assign values to ARRAY array -----
ARRAY(0)= -10; ARRAY(1)=3; ARRAY(2)=5; ARRAY(3)=14; ARRAY(4)=12
XX=-20; XI=0.1                    !Assigns values to XX and XI.
WHILE XX<20                        !Set conditions.
XX=XX+XI
YY=MAPS(XX,ARRAY,-10,5)           !MAPS command where:
                                !XX is the defined variable.
                                !ARRAY is a one-dimensional point array.
                                !-10 is the value of XX that corresponds to
                                !base point.
                                !5 is the value of XX that defines fixed
                                !intervals between the ARRAY points.
END                                !Ends mapping
STOP                              !Ends program

```

Table 4-12 shows the XX and YY values.

Table 4-12. MAPS Array

| | | | | | | | | | |
|-------|-----|-----|-------|----|---|----|----|-----|-----|
| XX | -20 | -10 | -7 | -5 | 0 | 5 | 10 | 15 | 20 |
| ARRAY | N/A | -10 | N/A | 3 | 5 | 14 | 12 | N/A | N/A |
| YY | -10 | -10 | -2.65 | 3 | 5 | 14 | 12 | 12 | 12 |

Figure 4-24 illustrates this **MAPS** example on the Scope.

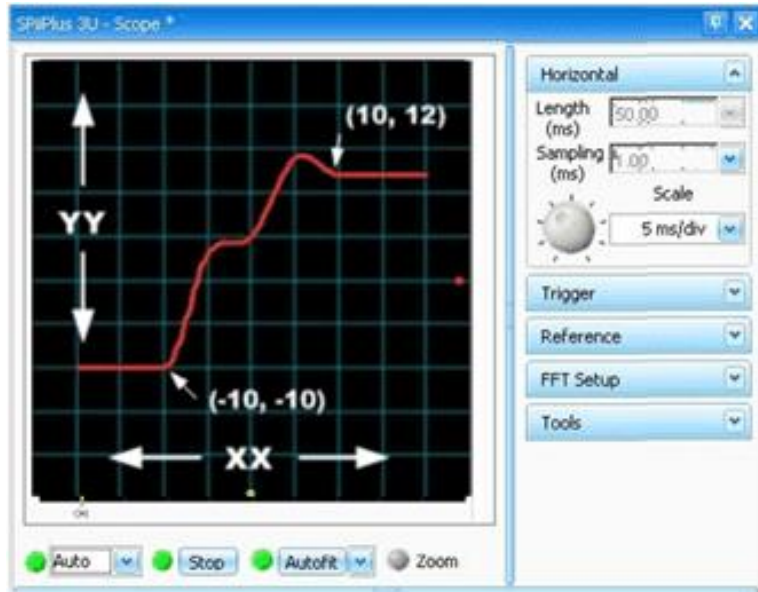


Figure 4-24. MAPS Example on the Scope

4.9.7.9 MAP2

Description

MAP2 returns a value from a two dimensional point array for each set of two input variable values, the base values of the input variables, and a fixed, defined interval for the variables. The return values between the array-points are linearly interpolated. **MAP2** is useful for dynamic error compensation.

Syntax

MAP2(*XX, ZZ, table, baseX, stepX, baseY, stepY*)

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------|
| XX | Real variable name or real expression. |
| ZZ | Real variable name or real expression. |
| table | The name of a real two-dimensional array that specifies points |
| baseX | A real number representing the value of XX that corresponds to the first point in the array. |
| stepX | A real number representing the value of XX that defines fixed intervals between the array points. |
| baseY | A real number representing the value of ZZ that corresponds to the first point in the array. |
| stepY | A real number representing the value of ZZ that defines fixed intervals between the array points. |

Return Value

MAP2 returns a linearly interpolated value from the array according to the value of variables **XX** and **ZZ**.

Error Conditions

The function detects the following error conditions:

- > Error 3072, Wrong array size, if **table** has only one dimension.
- > Error 3113, The step in the table is zero or negative, when the **stepX** or **stepY** arguments are zero or negative.

Example

```
GLOBAL REAL XI,XX,YY,ZZ, TABLE(5) (5)
                                !Defines real variables and 5x5 matrix.
XX=-20;ZZ=10;XI=0.1!Assigns values to the variable.
!----- Assign values to TABLE matrix -----
TABLE (0) (0)=2; TABLE (0) (1)=22; TABLE (0) (2)=6; TABLE (0) (3)=8; TABLE (0)
(4)=10;
TABLE (1) (0)=12; TABLE (1) (1)=-44; TABLE (1) (2)=16; TABLE (1) (3)=18;
TABLE (1) (4)=20; TABLE (2) (0)=22; TABLE (2) (1)=24; TABLE (2) (2)=26;
TABLE (2) (3)=68; TABLE (2) (4)=30; TABLE (3) (0)=12; TABLE (3) (1)=34;
TABLE (3) (2)=59; TABLE (3) (3)=-38; TABLE (3) (4)=40; TABLE (4) (0)=92;
TABLE (4) (1)=44; TABLE (4) (2)=46; TABLE (4) (3)=10; TABLE (4) (4)=90
WHILE ZZ<70
XX=XX+XI                                !Set conditions
ZZ=ZZ+XI
YY=MAP2 (XX,ZZ, TABLE, -10,5,20,10)!MAP2 command where:
                                !XX and ZZ are variables.
                                !TABLE is a 2x2 matrix that specifies points.
                                !-10 is the first XX point in the matrix.
                                !5 defines the fixed intervals between the XX
                                points.
                                !20 is the first ZZ point in the matrix.
                                !10 defines the fixed intervals between the ZZ
                                !points.
DISP XX,ZZ,YY                        !Displays values of XX, ZZ and YY.
END                                    !Ends MAP2.
STOP                                  !Ends program
```

Table 4-13 shows the YY values as a function of XX and ZZ arguments.

Table 4-13. MAP2

| | | XX | | | | | | |
|----|-----|-----|-----|-----|----|-----|----|----|
| ZZ | | -30 | -10 | -5 | 0 | 5 | 10 | 50 |
| | | 1 | 2 | 2 | 22 | 6 | 8 | 10 |
| | 20 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 30 | 12 | 12 | -44 | 16 | 18 | 20 | 20 |
| | 40 | 22 | 22 | 24 | 26 | 68 | 30 | 30 |
| | 50 | 12 | 12 | 34 | 59 | -38 | 40 | 40 |
| | 60 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |
| | 100 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |

| | |
|------|----------------------------------------------------------------|
| Key: | XX and ZZ values |
| | Interpolated YY values outside the boundaries of array "Table" |
| | "Table" array values |

Figure 4-25 illustrates this MAP2 example on the Scope.

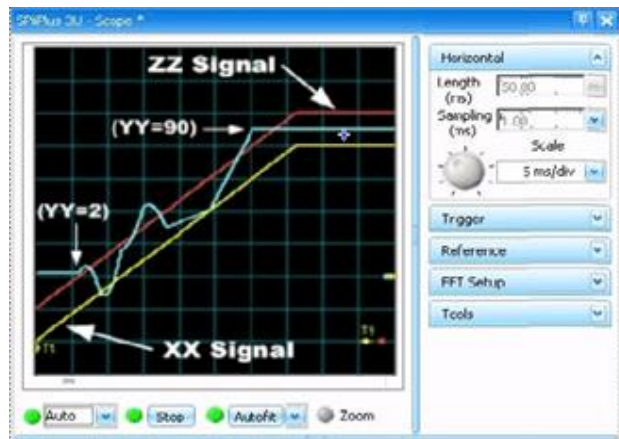


Figure 4-25. MAP2 Example on the Scope

4.9.7.10 MAP2B

Description

MAP2B returns a value from a two dimensional array of points according to the value of two input variables, where the base values of the input variables are fixed and the intervals of the input variables are defined. The return values between the array-points are interpolated according to a third order B-spline. **MAP2B** is useful for dynamic error compensation.

Syntax

MAP2B(XX,ZZ, table, baseX, stepX, baseY, stepY)

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------|
| XX | Real variable name or real expression. |
| ZZ | Real variable name or real expression. |
| table | The name of a real two-dimensional array that specifies points |
| baseX | A real number representing the value of XX that corresponds to the first point in the array. |
| stepX | A real number representing the value of XX that defines fixed intervals between the array points. |
| baseY | A real number representing the value of ZZ that corresponds to the first point in the array. |
| stepY | A real number representing the value of ZZ that defines fixed intervals between the array points. |

Return Value

MAP2B returns the third order B-spline interpolated value from the array according to the value of variables **XX** and **ZZ**.

The function detects the following error conditions:

- > Error 3072, Wrong array size, if **table** has only one dimension.
- > Error 3113, The step in the table is zero or negative, when the **stepX** or **stepY** arguments are zero or negative.

Example

```
GLOBAL REAL XI,XX,YY,ZZ,T(5)(5)
                                !Defines real variables and 5x5 matrix.
XX=-20;ZZ=10;XI=0.1!Assigns values to the variables.
!----- Assign values to T matrix -----
T(0)(0)=2;T(0)(1)=22;T(0)(2)=6;T(0)(3)=8;T(0)(4)=10; T(1)(0)=12;
T(1)(1)=-44;T(1)(2)=16;T(1)(3)=18; T(1)(4)=20;T(2)(0)=22;T(2)(1)=24;
T(2)(2)=26; T(2)(3)=68;T(2)(4)=30;T(3)(0)=12;T(3)(1)=34; T(3)(2)=59;
T(3)(3)=-38;T(3)(4)=40;T(4)(0)=92; T(4)(1)=44;T(4)(2)=46;T(4)(3)=10;
T(4)(4)=90
WHILE ZZ<70
XX=XX+XI                                !Set conditions
ZZ=ZZ+XI
YY=MAP2B(XX,ZZ,T,-10,5,20,10!MAP2B command where:
                                !XX and ZZ are variables.
                                !T is a 2x2 matrix that specifies points.
                                !-10 is the first XX point in the matrix.
                                !5 defines the fixed intervals between the XX points.
                                !20 is the first ZZ point in the matrix.
                                !10 defines the fixed intervals between the ZZ
```

```

!points.
DISP XX,ZZ,YY      !Displays values of XX, ZZ and YY.
END                !Ends MAP2B.
STOP              !Ends program
    
```

Table 4-14 shows the YY values as a function of XX and ZZ arguments.

Table 4-14. MAP2B

| | | XX | | | | | | |
|----|-----|-----|-----|-----|----|-----|----|----|
| ZZ | | -30 | -10 | -5 | 0 | 5 | 10 | 50 |
| | | 1 | 2 | 2 | 22 | 6 | 8 | 10 |
| | 20 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 30 | 12 | 12 | -44 | 16 | 18 | 20 | 20 |
| | 40 | 22 | 22 | 24 | 26 | 68 | 30 | 30 |
| | 50 | 12 | 12 | 34 | 59 | -38 | 40 | 40 |
| | 60 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |
| | 100 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |

Key:

| |
|----------------------------------------------------------------|
| XX and ZZ values |
| Interpolated YY values outside the boundaries of array "Table" |
| "Table" array values |

Figure 4-26 illustrates this MAP2B example on the Scope.

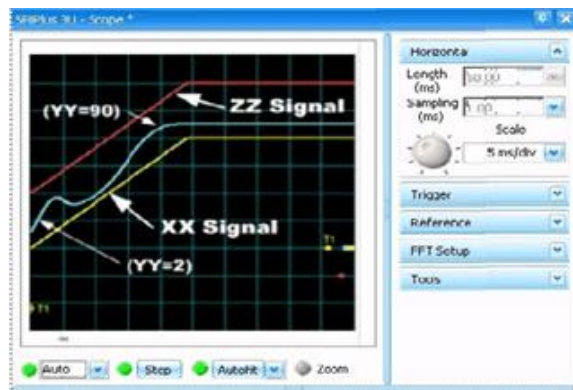


Figure 4-26. MAP2B Example on the Scope

4.9.7.11 MAP2N

Description

MAP2N returns a value from a two dimensional array of points according to the value of two input variables, with look-up arrays for each of the variables. The return values between the array-points are linearly interpolated. **MAP2N** is useful for dynamic error compensation.

Syntax

MAP2N(XX,ZZ, table, X_table, Y_table)

Arguments

| | |
|----------------|-------------------------------------------------------------------------------------------------------|
| XX | Real variable name or real expression. |
| ZZ | Real variable name or real expression. |
| table | The name of a real two-dimensional array that specifies points |
| X_table | A one-dimensional real array that specifies the values of XX that corresponds the point array. |
| Y_table | A one-dimensional real array that specifies the values of ZZ that corresponds the point array. |

Return Value

MAP2N returns the linearly interpolated value from the array according to the value of the variables **XX** and **ZZ**.



- > To obtain a valid return value from the MAP2N function after the Y_table array is changed, the buffer must be recompiled before calling the MAP2N function.
- > If a MAP2N function is called without recompiling the buffer, then the return value is calculated according to the values from the Y_table array that was provided to the function after the last compilation.

Error Conditions

The function detects the following error conditions:

- > Error 3072, Wrong array size, when **table** has only one dimension or **X_table** or **Y_table** have the wrong dimension. **X_table** must contain **M** elements and **Y_table** must contain **N** elements.
- > The values in **X_table** or **Y_table** are not sequenced in ascending order.

Example

```
GLOBAL REAL XI,XX,YY,ZZ,XTABLE(5),YTABLE(5),TABLE(5)(5)
                                !Defines real variables, two arrays and a 5x5 matrix.
XX=-90;ZZ=0;XI=0.1             !Assigns values to the variables.
!----- Assign values to XTABLE array -----
--
XTABLE(0)=13; XTABLE(1)=-8; XTABLE(2)=0; XTABLE(3)=12; XTABLE(4)=51
!----- Assign values to YTABLE array -----
--
YTABLE(0)=16; YTABLE(1)=21; YTABLE(2)=80; YTABLE(3)=82; YTABLE(4)=113
!----- Assign values to TABLE matrix -----
--
TABLE(0)(0)=2; TABLE(0)(1)=22; TABLE(0)(2)=6; TABLE(0)(3)=8; TABLE(0)
(4)=10
TABLE(1)(0)=12; TABLE(1)(1)=-44; TABLE(1)(2)=16; TABLE(1)(3)=18;
```

```

TABLE (1) (4)=20; TABLE (2) (0)=22; TABLE (2) (1)=24; TABLE (2) (2)=26;
TABLE (2) (3)=68; TABLE (2) (4)=30; TABLE (3) (0)=12; TABLE (3) (1)=34;
TABLE (3) (2)=59; TABLE (3) (3)=-38; TABLE (3) (4)=40; TABLE (4) (0)=92;
TABLE (4) (1)=44; TABLE (4) (2)=46; TABLE (4) (3)=10; TABLE (4) (4)=90
WHILE ZZ<200
XX=XX+XI!Set conditions
ZZ=ZZ+XI
YY=MAP2N(XX,ZZ, TABLE,XTABLE,YTABLE)!MAP2N command where:
                                !XX and ZZ are variables.
                                !TABLE is a 5x5 matrix specifying mapping points.
                                !XTABLE is an array specifying XX points.
                                !YTABLE is an array specifying ZZ points.
DISP XX,ZZ,YY                 !Displays values of XX, ZZ and YY.
END                             !Ends MAP2N.
STOP                            !Ends program

```

Table 4-15 shows the YY values as a function of XX and ZZ arguments.

Table 4-15. MAP2B

| | | XX | | | | | | |
|----|-----|-----|-----|-----|----|-----|----|-----|
| ZZ | | -90 | -13 | -8 | 0 | 12 | 51 | 100 |
| | 0 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 16 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 21 | 12 | 12 | -44 | 16 | 18 | 20 | 20 |
| | 80 | 22 | 22 | 24 | 26 | 68 | 30 | 30 |
| | 82 | 12 | 12 | 34 | 59 | -38 | 40 | 40 |
| | 113 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |
| | 150 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |

| | |
|-------------|-----------------------------------------------------------------|
| Key: | XX and ZZ values |
| | Interpolated YY values: outside the boundaries of array "Table" |
| | "Table" array values |

Figure 4-27 illustrates this MAP2N example on the Scope.

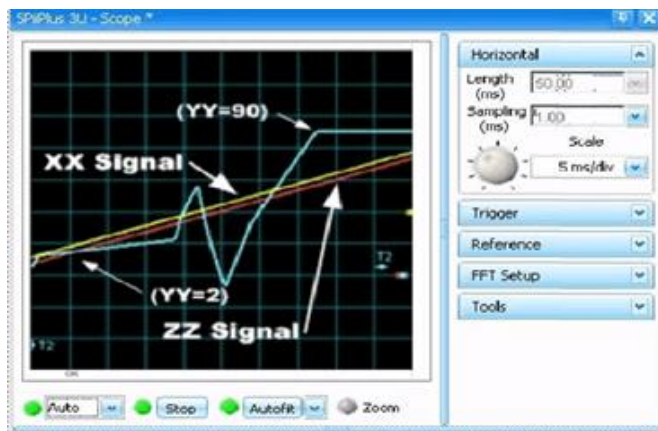


Figure 4-27. MAP2NB Example on the Scope

4.9.7.12 MAP2NB

Description

MAP2NB returns a value from a two-dimensional array of points based on the input values of two variables, with a separate look-up array for one variable, and a separate look-up array for the other variable. The return values between the array-points are interpolated according to a third-order B-spline. **MAP2NB** is useful for dynamic error compensation.

Syntax


MAP2NB(*XX,ZZ, Table, X_Table, Y_Table*)

Arguments

| | |
|----------------|-------------------------------------------------------------------------------------------------------|
| <i>XX</i> | Real variable name or real expression. |
| <i>ZZ</i> | Real variable name or real expression. |
| <i>table</i> | The name of a real two-dimensional array that specifies points |
| <i>X_table</i> | A one-dimensional real array that specifies the values of XX that corresponds the point array. |
| <i>Y_table</i> | A one-dimensional real array that specifies the values of ZZ that corresponds the point array. |

Return Value

MAP2NB returns the third-order B-spline interpolated value from the array according to the value of the variables **XX** and **ZZ**.



- > To obtain a valid return value from the MAP2NB function after the Y_table array is changed, the buffer must be recompiled before calling the MAP2NB function.
- > If a MAP2NB function is called without recompiling the buffer, then the return value is calculated according to the values from the Y_table array that was provided to the function after the last compilation.

Error Conditions

The function detects the following error conditions:

- > Error 3072, Wrong array size, when **table** has only one dimension or **X_table** or **Y_table** have the wrong dimension. **X_table** must contain **M** elements and **Y_table** must contain **N** elements.
- > The values in **X_table** or **Y_table** are not sequenced in ascending order.

Example

```

GLOBAL REAL XI,XX,YY,ZZ,XTABLE(5),YTABLE(5),TABLE(5)(5)
                                !Defines real variables, two arrays and a 5x5 matrix.
XX=-90;ZZ=0;XI=0.1            !Assigns values to the variables.
!----- Assign values to XTABLE array -----
--
XTABLE(0)=-13; XTABLE(1)=-8; XTABLE(2)=0; XTABLE(3)=12; XTABLE(4)=51
!----- Assign values to YTABLE array -----
--
YTABLE(0)=16; YTABLE(1)=21; YTABLE(2)=80; YTABLE(3)=82; YTABLE(4)=113
!----- Assign values to TABLE matrix -----
--
TABLE(0)(0)=2; TABLE(0)(1)=22; TABLE(0)(2)=6; TABLE(0)(3)=8; TABLE(0)
(4)=10;
TABLE(1)(0)=12; TABLE(1)(1)=-44; TABLE(1)(2)=16; TABLE(1)(3)=18;
TABLE(1)(4)=20; TABLE(2)(0)=22; TABLE(2)(1)=24; TABLE(2)(2)=26;
TABLE(2)(3)=68; TABLE(2)(4)=30; TABLE(3)(0)=12; TABLE(3)(1)=34;
TABLE(3)(2)=59; TABLE(3)(3)=-38; TABLE(3)(4)=40; TABLE(4)(0)=92;
TABLE(4)(1)=44; TABLE(4)(2)=46; TABLE(4)(3)=10; TABLE(4)(4)=90
WHILE ZZ<200
XX=XX+XI                        !Set conditions
ZZ=ZZ+XI
YY=MAP2NB(XX,ZZ,TABLE,XTABLE,YTABLE)
                                !MAP2NB command where:
                                !XX and ZZ are variables.
                                !TABLE is a 5x5 matrix specifying mapping points.
                                !XTABLE is an array specifying XX points.
                                !YTABLE is an array specifying ZZ points.
DISP XX,ZZ,YY                  !Displays values of XX, ZZ and YY.
END                              !Ends MAP2N.
STOP                            !Ends program
  
```

Table 4-16 shows the YY values as a function of XX and ZZ arguments.

Table 4-16. MAP2NB

| XX | | -90 | -13 | -8 | 0 | 12 | 51 | 100 |
|----|-----|-----|-----|-----|----|-----|----|-----|
| ZZ | 0 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 16 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 21 | 12 | 12 | -44 | 16 | 18 | 20 | 20 |
| | 80 | 22 | 22 | 24 | 26 | 68 | 30 | 30 |
| | 82 | 12 | 12 | 34 | 59 | -38 | 40 | 40 |
| | 113 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |
| | 150 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |

Key: XX and ZZ values
 Interpolated YY values outside the boundaries of array "Table"
 "Table" array values

MAP2B illustrates this MAP2NB example on the Scope.

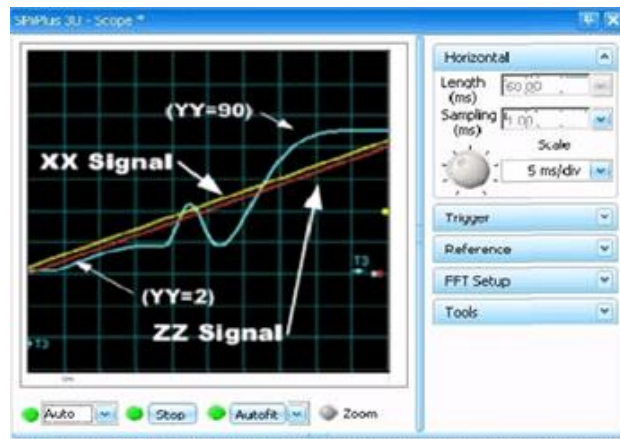


Figure 4-28. MAP2NB Example on the Scope

4.9.7.13 MAP2NS

Description

MAP2NS returns a value from a two-dimensional array of points based on the input values of two variables, with a separate look-up array for one variable, and a separate look-up array for the other variable. The return values between the array-points are interpolated according to a third-order Catmull-Rom spline. The **MAP2NS** function is useful for dynamic error compensation

Syntax

MAP2NS(XX,ZZ, table, X_table, Y_table)

Arguments

| | |
|----|----------------------------------------|
| XX | Real variable name or real expression. |
| ZZ | Real variable name or real expression. |

| | |
|----------------|-------------------------------------------------------------------------------------------------------|
| table | The name of a real two-dimensional array that specifies points |
| X_table | A one-dimensional real array that specifies the values of XX that corresponds the point array. |
| Y_table | A one-dimensional real array that specifies the values of ZZ that corresponds the point array. |

Return Value

MAP2NS returns the third-order Catmull-Rom interpolated value from the array according to the value of the variables **XX** and **ZZ**.



- > To obtain a valid return value from the MAP2NS function after the Y_table array is changed, the buffer must be recompiled before calling the MAP2NS function.
- > If a MAP2NS function is called without recompiling the buffer, then the return value is calculated according to the values from the Y_table array that was provided to the function after the last compilation.

Error Conditions

The function detects the following error conditions:

- > Error 3072, Wrong array size, when **table** has only one dimension or **X_table** or **Y_table** have the wrong dimension. **X_table** must contain **M** elements and **Y_table** must contain **N** elements.
- > The values in **X_table** or **Y_table** are not sequenced in ascending order.

Example

```

GLOBAL REAL XI,XX,YY,ZZ,XTABLE(5),YTABLE(5),TABLE(5)(5)
                                !Defines real variables, two arrays and a 5x5 matrix.
XX=-90;ZZ=0;XI=0.1            !Assigns values to the variables.
!----- Assign values to XTABLE array -----
--
XTABLE(0)=13; XTABLE(1)=-8; XTABLE(2)=0; XTABLE(3)=12; XTABLE(4)=51
!----- Assign values to YTABLE array -----
--
YTABLE(0)=16; YTABLE(1)=21; YTABLE(2)=80; YTABLE(3)=82; YTABLE(4)=113
!----- Assign values to TABLE matrix -----
--
TABLE(0)(0)=2; TABLE(0)(1)=22; TABLE(0)(2)=6; TABLE(0)(3)=8; TABLE(0)
(4)=10;
TABLE(1)(0)=12; TABLE(1)(1)=-44; TABLE(1)(2)=16; TABLE(1)(3)=18;
TABLE(1)(4)=20; TABLE(2)(0)=22; TABLE(2)(1)=24; TABLE(2)(2)=26;
TABLE(2)(3)=68; TABLE(2)(4)=30; TABLE(3)(0)=12; TABLE(3)(1)=34;
TABLE(3)(2)=59; TABLE(3)(3)=-38; TABLE(3)(4)=40; TABLE(4)(0)=92;
TABLE(4)(1)=44; TABLE(4)(2)=46; TABLE(4)(3)=10; TABLE(4)(4)=90
WHILE ZZ<200
XX=XX+XI                        !Set conditions
ZZ=ZZ+XI
YY=MAP2NS(XX,ZZ,TABLE,XTABLE,YTABLE)
                                !MAP2NS command where:
                                !XX and ZZ are variables.
                                !TABLE is a 5x5 matrix specifying mapping points.
                                !XTABLE is an array specifying XX points.
                                !YTABLE is an array specifying ZZ points.
DISP XX,ZZ,YY                  !Displays values of XX, ZZ and YY.
END                              !Ends MAP2N.
STOP                            !Ends program
    
```

Table 4-17 shows the YY values as a function of XX and ZZ arguments.

Table 4-17. MAP2NS

| | | XX | | | | | | |
|----|-----|-----|-----|-----|----|-----|----|-----|
| ZZ | | -90 | -13 | -8 | 0 | 12 | 51 | 100 |
| | 0 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 16 | 2 | 2 | 22 | 6 | 8 | 10 | 10 |
| | 21 | 12 | 12 | -44 | 16 | 18 | 20 | 20 |
| | 80 | 22 | 22 | 24 | 26 | 68 | 30 | 30 |
| | 82 | 12 | 12 | 34 | 59 | -38 | 40 | 40 |
| | 113 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |
| | 150 | 92 | 92 | 44 | 46 | 10 | 90 | 90 |

| | |
|------|----------------------------------------------------------------|
| Key: | XX and ZZ values |
| | Interpolated YY values outside the boundaries of array "Table" |
| | "Table" array values |

Figure 4-29 illustrates this MAP2NS example on the Scope.

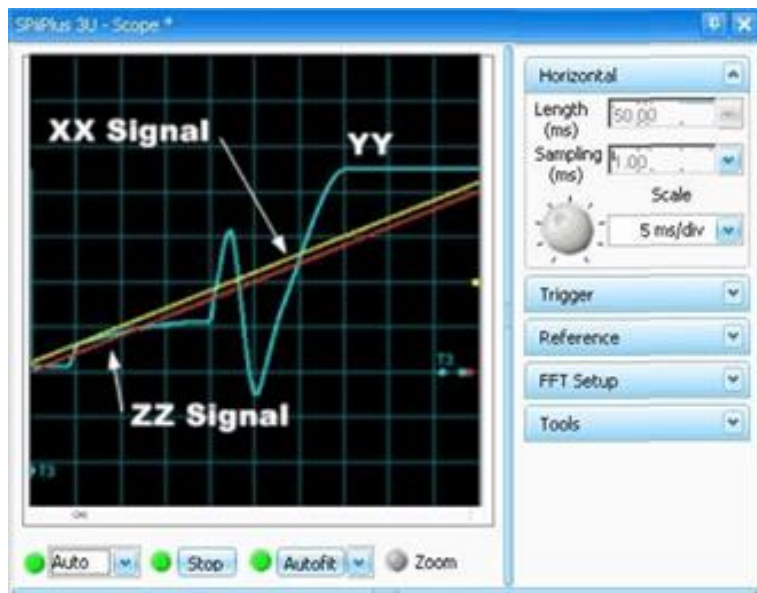


Figure 4-29. MAP2NS Example on the Scope

4.9.7.14 MAP2S

Description

MAP2S returns a value from a two-dimensional array of points based on the input values of two variables, the base values of the two variables, and the fixed defined intervals of the two variables. The return values between the array-points are interpolated according to a third order Catmull-Rom spline. The **MAP2S** function is useful for dynamic error compensation.

Syntax

MAP2S(XX,ZZ, table, baseX, stepX, baseY, stepY)

Arguments

| | |
|--------------|--------------------------------------------------------------------------------------------------------------|
| XX | Real variable name or real expression. |
| ZZ | Real variable name or real expression. |
| table | The name of a real two-dimensional array that specifies points |
| baseX | A real number representing the value of XX that corresponds to the first point in the array. |
| stepX | A real number representing the value of XX that defines the fixed intervals between the array points. |
| baseY | A real number representing the value of ZZ that corresponds to the first point in the array. |
| stepY | A real number representing the value of ZZ that defines the fixed intervals between the array points. |

Return Value

MAP2S returns the third order Catmull-Rom spline interpolated value from the array according to the value of variables **XX** and **ZZ**.

Error Conditions

The function detects the following error conditions:

- > Error 3072, Wrong array size, when **table** has only one dimension.
- > Error 3113, The step in the table is zero or negative, when the **stepX** or **stepY** arguments are zero or negative.

Example

```

GLOBAL REAL XI,XX,YY,ZZ, TABLE (5) (5)
                                !Defines real variables and 5x5 matrix.
XX=-20;ZZ=10;XI=0.1           !Assigns values to the variable.
!----- Assign values to TABLE matrix -----
TABLE (0) (0)=2; TABLE (0) (1)=22; TABLE (0) (2)=6; TABLE (0) (3)=8; TABLE (0)
(4)=10;
TABLE (1) (0)=12; TABLE (1) (1)=-44; TABLE (1) (2)=16; TABLE (1) (3)=18;
TABLE (1) (4)=20; TABLE (2) (0)=22; TABLE (2) (1)=24; TABLE (2) (2)=26;
TABLE (2) (3)=68; TABLE (2) (4)=30; TABLE (3) (0)=12; TABLE (3) (1)=34;
TABLE (3) (2)=59; TABLE (3) (3)=-38; TABLE (3) (4)=40; TABLE (4) (0)=92;
TABLE (4) (1)=44; TABLE (4) (2)=46; TABLE (4) (3)=10; TABLE (4) (4)=90
WHILE ZZ<70
XX=XX+XI                       !Set conditions
ZZ=ZZ+XI
YY=MAP2S (XX,ZZ, TABLE, -10,5,20,10) !MAP2S where:
                                !XX and ZZ are variables.
                                !TABLE is a 2x2 matrix that specifies points.
                                !-10 is the first XX point in the matrix.
                                !5 defines the fixed intervals between the XX points.
                                !20 is the first ZZ point in the matrix.
                                !10 defines the fixed intervals between the ZZ
                                !points.
DISP XX,ZZ,YY                 !Displays values of XX, ZZ and YY.
END                             !Ends MAP2S.
STOP                            !Ends program

```

Table 4-18 shows the YY values as a function of XX and ZZ arguments.

Table 4-18. MAP2S

| | | XX | | | | | | | |
|----|-----|-----|-----|-----|----|-----|----|----|--|
| ZZ | | -30 | -10 | -5 | 0 | 5 | 10 | 50 | |
| | 1 | 2 | 2 | 22 | 6 | 8 | 10 | 10 | |
| | 20 | 2 | 2 | 22 | 6 | 8 | 10 | 10 | |
| | 30 | 12 | 12 | -44 | 16 | 18 | 20 | 20 | |
| | 40 | 22 | 22 | 24 | 26 | 68 | 30 | 30 | |
| | 50 | 12 | 12 | 34 | 59 | -38 | 40 | 40 | |
| | 60 | 92 | 92 | 44 | 46 | 10 | 90 | 90 | |
| | 100 | 92 | 92 | 44 | 46 | 10 | 90 | 90 | |

Key:

| |
|----------------------------------------------------------------|
| XX and ZZ values |
| Interpolated YY values outside the boundaries of array "Table" |
| "Table" array values |

Figure 4-30 illustrates this MAP2S example on the Scope.

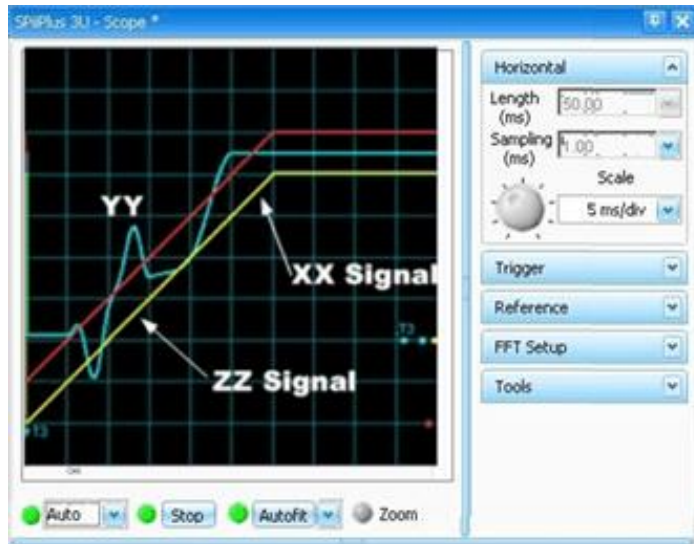


Figure 4-30. MAP2S Example on the Scope

4.9.7.15 MATCH

Description

MATCH calculates axis position (**APOS**) that matches the current reference position (**RPOS**), based on **CONNECT** between **APOS** and **RPOS** of the same axis. If there is no **CONNECT** command, the function returns the value of **RPOS**.

Syntax

real **MATCH** (*axis, from, to*)

Arguments

| | |
|-------------|-------------------------------------------------------------------------|
| axis | The axis upon which to apply the MATCH function. |
| from | The start point to begin searching for matching values of APOS . |
| to | The end point to stop searching for matching values of APOS . |

Comments

The function is useful only in the case of non-default connections.

The connection must be on the same axis.

The function succeeds if the unique root exists in the specified range. If there are several roots in the range, the function returns one of them. If the root does not exist, the function results an error.

Return Value

APOS that matches the current **RPOS** of the same axis.

Error Conditions

Error 3158 - The function cannot find a matching value. The **CONNECT** formula has no root or the root is not single.

Example

```

GLOBAL REAL ARRAY(6), XX
!----- Assign values to ARRAY array -----
ARRAY(0)= 60; ARRAY(1)=40; ARRAY(2)=90; ARRAY(3)=-40; ARRAY(4)=60;
ARRAY(5)=10
VEL(0)=800           !Set VEL(0)
MFLAGS(0).17=1
CONNECT RPOS(0)=APOS(0)+MAP(APOS(0),ARRAY,100,200)
                        !The program executes a CONNECT between RPOS and APOS
                        !for axis X. Where RPOS(0) receives the value of
                        !APOS(0) added to the value of a MAP function.

DEPENDS X,X
SET APOS(0)=0
SET RPOS(0)=MAP(APOS(0),ARRAY,100,200)
PTP X, 1300           !A PTP command sends the motor to 1300 counts,
                        !while the motor is in motion a MATCH function is
                        !calculated and displayed along with RPOS(0) and
                        !APOS(0).

WHILE MST(0).#MOVE
WAIT 300
XX=MATCH(X,0,1300)   !The MATCH function calculates the root of the
                        !current RPOS(0) based on the CONNECT formula.

DISP RPOS(0), APOS(0), XX
END
STOP                 !Ends program

```

The output of the program displays the following **RPOS(0)**, **APOS(0)** values, and the value of **MATCH**. Notice that **MATCH** values are very close to **APOS(0)** values:

| RPOS(0) | APOS(0) | MATCH |
|---------|---------|--------|
| 273.76 | 228 | 226.4 |
| 554 | 472.8 | 471.2 |
| 684 | 717.6 | 716 |
| 1005.6 | 962.4 | 960.8 |
| 1215.6 | 1207.2 | 1205.6 |
| 1310 | 1300 | 1300 |

4.9.7.16 RAND

Description

RAND generates a random number

Syntax

real **RAND**(*min*, *max*[,*seed*])

Arguments

| | |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>min</i> | Lower boundary of the interval from which randomized number will be selected. |
| <i>max</i> | Upper boundary of the interval from which randomized number will be selected. |
| <i>seed</i> | Sets the random sequence generator. The number range is limited only by the controller register size - 32bit. If omitted, the controller uses ACSPL+ TIME variable as the seed. |

Comments

RAND can generate only one random number per seed number (per interval). For generating a series of random numbers use **TIME** as the seed. In this case the series of random numbers will be in ascending order.

Return Value

The randomized generated number.

Error Conditions

None

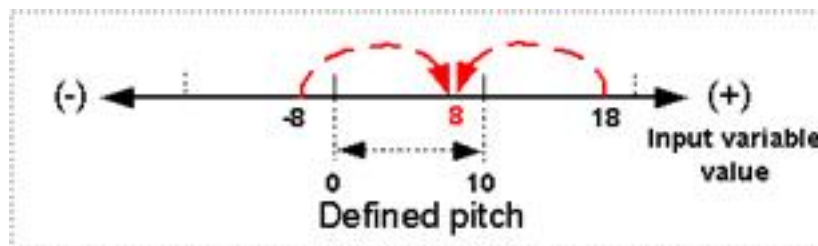
Example

```
RAND (-45, 45, TIME)
```

4.9.7.17 ROLL

Description

The **ROLL** function returns a result rolled-over to within a defined Pitch (range), as illustrated below.



Syntax

ROLL(*X*, *Pitch*)

Arguments

| | |
|--------------|--------------------------------------------------------------------------------------------------|
| <i>X</i> | Variable name declared as real or a real expression. |
| <i>Pitch</i> | A range from 0 to Pitch (positive real value) where the return value will be rolled-over. |

Return Value

X - if **X** falls within the range from 0 to **Pitch**.

or

$|X|/Pitch - \text{FLOOR}(|X|/Pitch)*Pitch$, if **X** is less than or greater than **Pitch**.

Error Conditions

None

Example:

```
YY=ROLL (XX, 10)                                !Input variable XX is changing between -
                                                  !20 to 20.
```

Figure 4-31 illustrates this **ROLL** example on the Scope.

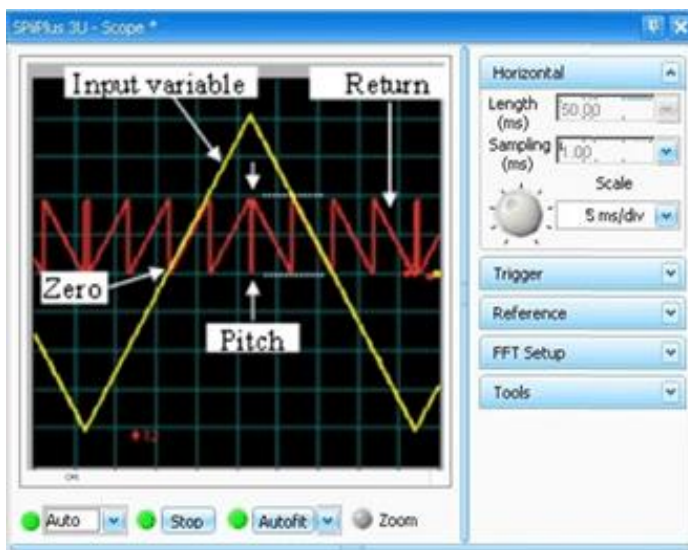


Figure 4-31. ROLL Example on the Scope

4.9.7.18 SAT

Description

SAT returns a result of defined saturation range

Syntax

real **SAT**(*X*, *Min*, *Max*)

Arguments

| | |
|------------|------------------------------------------------------|
| X | Variable name declared as real or a real expression. |
| Min | Low saturation value. Must be a real number. |
| Max | High saturation value. Must be a real number. |

Return Value

X - if the **X** value falls inside the SAT range.

Min - if **X** value is less than the SAT range.

Max - if **X** value is greater than SAT range.

Error Conditions

None

Example

```
GLOBAL REAL YY, ZZ, XX
EYAL :
ZZ=-20;XX=0.1
WHILE ZZ<=20
    ZZ=ZZ+XX
    YY=SAT (ZZ, -5, 15)
END
ZZ=20;XX=0.1
WHILE ZZ>=-20
    ZZ=ZZ-XX
    YY=SAT (ZZ, -5, 15)
END
GOTO EYAL
STOP
```

Figure 4-32 illustrates this **SAT** example on the Scope.

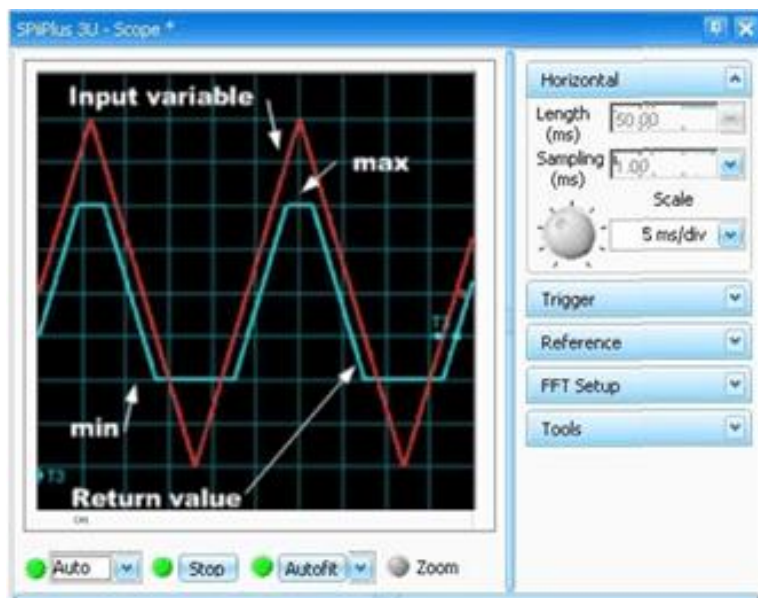


Figure 4-32. SAT Example on the Scope

4.10 Laser Control Functions

The laser control functions are:

| Function | Description |
|------------------------------|------------------------------------------------------------------------------------------------|
| LCMODULATION | Initializes Pulse modulation mode and sets initial values for the unit internal registers. |
| LCFixedDist | Initializes the fixed distance pulse firing mode. |
| LCFixedInt | Initializes the fixed distance pulse firing mode. |
| LCRandomDist | Initializes either array based pulse firing mode or gating mode. |
| LCTickle | Initializes Tickle mode. |
| LCZone | Sets a laser operation zone area. |
| LCZoneGet | Returns the limits of the laser operation zone that was previously defined. |
| LCZoneSet | Changes the minimal and/or maximal laser operation zone limit. |
| LCStop | Stops any previously initialized laser mode and resets the previously defined mode parameters. |
| LCSignalSet | Configures the laser control signal (LCS) output conditioning state. |
| LCSignalGet | Returns the laser control signal (LCS) output conditioning state. |
| LCOutputSet | Configures the laser physical outputs. |
| LCOutputGet | Returns the laser outputs configuration. |
| LCDelaySet | Sets the pulse generation delay in microseconds. |
| LCDelayGet | Returns the actual currently configured delay in microseconds. |
| AxListAsMask | Mask for defining axes. |

4.10.1 LCMODULATION

The function initializes Pulse modulation mode and sets initial values for the unit internal registers. Which Laser Control unit is initialized is determined by the index argument.

The Laser Control unit can be initialized irrespective if the corresponding axis is enabled or disabled.

Syntax

LCMODULATION (**Index**, **Mode**, **ModulationFrequency**, **Width**, **DutyCycle**, **AxesUsed**, [**MinValue**, **MaxValue**, **MinVelocity**, **MaxVelocity**])

Arguments

| Arguments | Comments |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | <p>Defines which Laser Control unit is referred.</p> <p>The system allocates 4 indexes (logical axes) for each LCM in the network. The last index of allocated logical axes should be specified.</p> <p>For example, if LCM gets network indexes (logical axes) 4,5,6,7, so index 7 should be used for referring the specific LCM module.</p> |
| Mode | <p>Operation mode:</p> <ol style="list-style-type: none"> 1. Mode with fixed frequency 2. Mode with fixed pulse width 3. Mode with fixed duty cycle <p>Modes 1 – 3 are incompatible modes, i.e. setting any of these modes automatically disables the previously set mode.</p> <p>In order to stop any of previously defined modes, 0 value is used.</p> |
| ModulationFrequency | <p>Pulse modulation frequency in Hz, range from 0.035Hz to 1MHz</p> <p>In <i>mode</i> = 1, the frequency is fixed and is not changed during the process.</p> <p>In <i>mode</i> = 2 or 3, the function only sets the initial frequency.</p> <p>If AxesUsed parameter is defined, during the process the controller will automatically update the frequency as function of vector velocity of the axes referred by AxesUsed.</p> <p>If AxesUsed parameter is zero, the application is responsible to update PFGPAR variable with frequency value.</p> |
| Width | <p>Pulse Width in milliseconds, range from 6.67nsec to 28.60sec.</p> <p>Only has effect in <i>mode</i> = 2.</p> |

| | |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DutyCycle | <p>Duty cycle in percentage, range from 0 to 100%.</p> <p>In <i>mode</i> = 1, the function only sets the initial duty cycle.</p> <p>If AxesUsed parameter is defined, during the process the controller will automatically update duty cycle as function of vector velocity of the axes referred by AxesUsed.</p> <p>If AxesUsed parameter is zero, the application is responsible to update PFGPAR variable with duty cycle value, see Duty cycle or frequency update for details.</p> <p>In <i>mode</i> = 2, the argument has no effect, as duty cycle is automatically calculated by the unit as function of frequency.</p> <p>In <i>mode</i> = 3, the duty cycle is fixed and is not changed during the process.</p> <p>In <i>mode</i> = 4, the parameters is ignored.</p> |
| AxisUsed | <p>Mask that defines which axes are used to calculate vector velocity for controlling duty cycle (mode 1) or frequency (modes 2 and 3).</p> <p>For example, the following mask defines that axes 0 and 2 should be used for vector velocity calculation:</p> <p>0b0101 (0b – is a prefix for binary value specification) or 0x5 (0x – is a prefix for hexadecimal value specification)</p> <p>To simplify the mask specification, the special ACSPL+ helper-function can be used:</p> <pre>AxesUsed = AxListAsMask(0,2)</pre> <p>Only first 32 axes (0..31) can be referred by AxesUsed. If no mask is specified, the application is responsible to update PFGPAR variable, see Duty cycle or frequency update.</p> |
| MinValue | <p>[Optional] Minimal value of duty cycle in percentage (if <i>mode</i> = 1) or frequency in Hz (if <i>mode</i> = 2 or <i>mode</i> = 3) if calculated vector velocity equals or below MinVelocity, see Figure 4-33.</p> <p>By default, minimal value of duty cycle is 0% and frequency is 0.035Hz .</p> |
| MaxValue | <p>[Optional] Maximal value of duty cycle in percentage (if <i>mode</i> = 1) or frequency in Hz (if <i>mode</i> = 2 or <i>mode</i> = 3) if calculated vector velocity equals or above MaxVelocity, see Figure 4-33.</p> <p>By default, maximal value of duty cycle is 100% and frequency is 1MHz.</p> |

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MinVelocity | <p>[Optional] Minimal velocity for duty cycle or frequency calculation. Below the minimal value, duty cycle or frequency will be limited by MinValue or default, if MinValue is not specified, see Figure 4-33.</p> <p>By default, minimal velocity is zero.</p> |
| MaxVelocity | <p>[Optional] Maximal velocity for duty cycle or frequency calculation. Above the maximal value, duty cycle or frequency will be limited by MaxValue or default, if MaxValue is not specified, see Figure 4-33.</p> <p>By default, maximal velocity is defined by XVEL parameter.</p> |

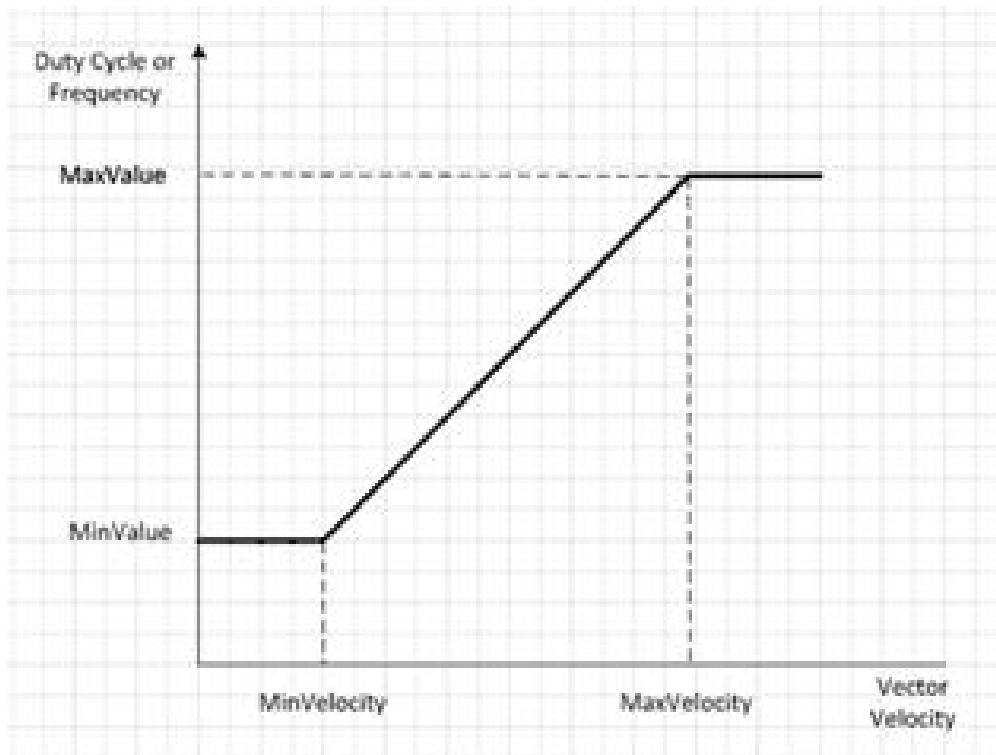


Figure 4-33. Velocity

Status

Modulation active/not active status is represented in AST.#LCMODUL (bit 22).

Example

Add example for regular mode and tickle mode initialization.

```
LCMODULATION (Laser, ! Laser index in the system
              1, ! Mode with fixed frequency
              5000, ! Frequency 5kHz
              0.001, ! Pulse width 1 microsec
              0, ! Initial duty cycle 0%
              0x03, ! Vector velocity of axis 0 and 1 defines duty cycle
              0, ! No Constant Spacing mode
```

```
0, 100, ! Minimal 0% and maximal 100% duty cycle
10, 1000) ! Minimal 10mm/sec and maximal 1000mm/sec velocity
```

Vector velocity indication (GVEL)

If **AxesUsed** parameter specifies involved axes mask, then the function calculates the vector velocity of all involved axes. The calculated vector velocity is indicated in GVEL parameter. The GVEL parameter index should correspond to the last (fourth) local logical axis of the specific LCM unit. For example, if LCM gets network indexes (logical axes) 4,5,6,7, so index 7 should be used for monitoring the calculated vector velocity.

4.10.1.1 Duty cycle or frequency update

The **LCMODULATION** function provides ability to automatically calculate vector velocity of the axes involved in a specific motion and update either a laser output duty cycle or frequency accordingly with the actual vector velocity.

In some cases, the user needs to implement more complex user-specific function for duty cycle or frequency update. In this case **AxesUsed** argument should be zero and the user is responsible to update duty cycle or frequency using **PGFPAR** parameter.

4.10.1.2 Duty cycle or Frequency monitoring

The controller provides ability to monitor an actual frequency or duty cycle at real-time by means of **LCFREQ** and **LCDC** controller variables.

LCFREQ and **LCDC** are real type array of 64 elements. Each element corresponds to the specific axis that belongs to LCM unit. The same index that is defined in **LCMODULATION** should be used as index for **LCFREQ** and **LCDC** reference.

| | Type | Accessibility | Description |
|---------------|------------------|---------------|-----------------------------------------------------------------------------------------------------|
| LCFREQ | Real Array 0..63 | Read-only | Indicates an actual frequency in [Hz] for modes 2 and 3, or configured frequency value for mode 1. |
| LCDC | Real Array 0..63 | Read-only | Indicates an actual duty cycle in [%] for modes 1, or configured duty cycle value for mode 2 and 3. |

4.10.2 LCFixedDist

The function initializes the fixed distance pulse firing mode. This mode is useful if a laser beam should be activated with specified fixed intervals between activations along the motion trajectory.

As pulses are generated along an actual motion trajectory. The **AxesUsed** argument defines which axes are used for multi-axis trajectory generation.

The arguments **StartPosition**, **StopPosition** defines start and last points in the user units relatively the user origin, defined by **InitOffset**. The pulse are generated with constant interval, defined by **Interval**.

Additionally, number of extra pulses, specified by **ExtraPulses**, with period, specified by **ExtraPeriod**, can be generated after each pulse at **Interval** position.

The **UserVector** argument provides ability to implement user-defined multi-axes trajectory. The **AxesUsed** argument should be zero in this case.

The **MinPosition** and **MaxPosition** arguments define an operation zone range along the multi-axes trajectory. The laser pulses start outputting after reaching **MinPosition** and stop outputting after reaching **MaxPosition**.


Syntax

```
LCFIXEDDIST (Index, AxesUsed, Width, StartPosition, Interval, StopPosition
[, InitOffset, ExtraPulses, ExtraPeriod, PulseResolution, UserVector,
MinPosition, MaxPosition])
```

Arguments

| Arguments | Comments |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | <p>Defines which Laser Control unit is referred.</p> <p>The system allocates 4 indexes (logical axes) for each LCM in the network. Third index of allocated logical axes should be specified.</p> <p>For example, if LCM gets network indexes (logical axes) 4,5,6,7, so index 6 should be used for referring the specific LCM module.</p> |
| AxesUsed | <p>Mask that defines the axes, which are used for generating pulses along the multi-axis motion trajectory.</p> <p>For example, the following mask defines that axes 0 and 2 should be used for vector velocity calculation:</p> <p>0b0101 (0b – is a prefix for binary value specification) or 0x5 (0x – is a prefix for hexadecimal value specification)</p> <p>Only first 32 axes (0..31) can be referred by AxesUsed.</p> <p>If AxesUsed is zero and UserVector is zero, the previously defined fixed distance pulse firing mode is canceled.</p> <p>The AxesUsed should be zero, if UserVector is used for user-defined trajectory calculation.</p> |
| Width | <p>Pulse Width in milliseconds, from 0.00002664msec (26.64nsec) to 1.745msec</p> |

| | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StartPosition | <p>Specifies a start position along the multi-axis motion trajectory. The pulses start generating after reaching this position.</p> <p>The StartPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |
| Interval | <p>Specifies an interval between pulses along the multi-axis motion trajectory.</p> <p>The Interval is specified in the user units of the axes, defined by AxesUsed</p> |
| StopPosition | <p>Specifies a last position along the multi-axis motion trajectory. The pulses stop generating after reaching this position.</p> <p>The StopPosition is specified relatively to the user origin, defined by InitOffset and the user units of the axes, defined by AxesUsed.</p> |
| InitOffset | <p>[Optional] Specifies an initial position offset of multi-axis trajectory relatively to the user origin.</p> <p>By default, the function initializes an initial position of multi-axis trajectory as current actual position of the involved axes at the moment the function is called, so InitOffset is zero.</p> <div data-bbox="544 999 1374 1429" style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>For all functions:</p> <ul style="list-style-type: none"> > For multi-axes-trajectory, the InitOffset should be omitted or set to zero. This causes the trajectory calculation to start from the current position. > For single-axis trajectory, if APOS of the axis, which is specified in AxesUsed, is provided as an argument, then both RPOS of the real axis and the virtual axis will match. </div> |
| ExtraPulses | <p>[Optional] Number of additional pulses to be generated with ExtraPeriod period after each pulse at each Interval position.</p> <p>By default, no extra pulses are generated.</p> |
| ExtraPeriod | <p>[Optional] Time period in milliseconds for ExtraPulses additional pulses to be generated after each pulse at each Interval position.</p> <p>By default, no extra pulses are generated.</p> |

| | |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>PulseResolution</p> | <p>[Optional] Pulse Resolution in user units.</p> <p>If the parameter is omitted or specified as zero, then the function automatically calculates the most optimal pulse resolution according with maximal velocity (XVEL) of the axes, defined by AxesUsed, and the internal pulse generator maximal frequency, which is 12.5MHz.</p> <p>An actual pulse resolution is indicated in STEPF parameter.</p> <p>If the user needs to modify the resolution (higher or lower), this parameter allows to do it.</p> <p>The width of internal pulse generator is initialized to its minimal value of 40nsec to allow achieving the maximal pulses frequency.</p> <div data-bbox="544 651 1374 875" style="border: 1px solid black; padding: 10px; margin-top: 10px;">  <p>For ALL functions - if the specified resolution causes the pulse generator frequency to reach its maximal value during the motion, then the Velocity Limit fault is raised and the LCM stops generating pulses.</p> </div> |
| <p>UserVector</p> | <p>[Optional] The user defined global real scalar variable that contains result of multi-axes trajectory calculation. This option is used if it's required that pulse will be generated along a user-defined multi-axes trajectory, which implements some custom coordinate system transformation.</p> <p>Usually, user defined global scalar variable should be updated each controller cycle, so the calculation should be done in a separate buffer inside an endless LOOP statement.</p> <p>The argument should be zero, if AxesMask is defined.</p> |
| <p>MinPosition</p> | <p>[Optional] Specifies a minimal position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position.</p> <p>The MinPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |
| <p>MaxPosition</p> | <p>[Optional] Specifies a maximal position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position.</p> <p>The MaxPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |

Example

```
LCFIXEDDIST (Laser, ! Laser index in the system
             0x03, ! Axis 0 and 1 defines multi-axis trajectory for
             generating pulses
```

```

0.001, ! Pulse width 1 microsec
100, ! First pulse position
10, ! Spacing between pulses is user 10 units (µm, mm)
000) ! Last pulse position

```

4.10.3 LCFixedInt

The function is similar to the **LCFixedDist** distance and also initializes the fixed distance pulse firing mode.

Unlike the **LCFixedDist** function, the **LCFixedInt** function does not provide the ability of Extra Pulses generation.

Also the **LCFixedInt** function does not contain the **StartPosition**, **StopPosition**, and **PulseResolution** specifications.

All other capabilities of **LCFixedDist** function are supported.

This mode is useful if a laser is to be activated at specified fixed intervals between activations along the motion trajectory.

As pulses are generated along an actual motion trajectory, the **AxesUsed** argument defines which axes are used for multi-axis trajectory generation.

The user origin is defined by **InitOffset**. The pulse are generated with constant interval, defines by **Interval**.

The **UserVector** argument provides ability to implement user-defined multi-axes trajectory. The **AxesUsed** argument should be zero in this case.

The **MinPosition** and **MaxPosition** arguments define an operation zone range along the multi-axes trajectory. The laser pulses start outputting after reaching **MinPosition** and stop outputting after reaching **MaxPosition**.

Syntax

```
LCFIXEDINT (Index, AxesUsed, Width, Interval, [, InitOffset, UserVector, MinPosition, MaxPosition])
```

Arguments

| Arguments | Comments |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | <p>Defines which Laser Control unit is referred.</p> <p>The system allocates 4 indexes (logical axes) for each LCM in the network. First index of allocated logical axes should be specified.</p> <p>For example, if LCM gets network indexes (logical axes) 4,5,6,7, so index 6 should be used for referring the specific LCM module.</p> |

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AxesUsed | <p>Mask that defines the axes, which are used for generating pulses along the multi-axis motion trajectory.</p> <p>For example, the following mask defines that axes 0 and 2 should be used for vector velocity calculation:</p> <p>0b0101 (0b – is a prefix for binary value specification) or 0x5 (0x – is a prefix for hexadecimal value specification)</p> <p>Only first 32 axes (0..31) can be referred by AxesUsed.</p> <p>If AxesUsed is zero and UserVector is zero, then the previously defined fixed distance pulse firing mode is canceled.</p> <p>The AxesUsed should be zero, if the UserVector is used for user-defined trajectory calculation.</p> |
| Width | <p>Pulse Width in milliseconds, range from 0.00008ms (80nsec) to 0.08188msec (81.88µsec).</p> |
| Interval | <p>Specifies an interval between pulses along the multi-axis motion trajectory.</p> <p>The Interval is specified in the user units of the axes, defined by AxesUsed</p> |
| InitOffset | <p>[Optional] Specifies an initial position offset of multi-axis trajectory relatively to the user origin.</p> <p>By default, the function initializes an initial position of multi-axis trajectory as current actual position of the involved axes at the moment the function is called, so InitOffset is zero.</p> |
| UserVector | <p>[Optional] The user defined global scalar variable that contains result of multi-axes trajectory calculation. This option is used if it's required that pulse will be generated along a user-defined multi-axes trajectory, which implements some custom coordinate system transformation.</p> <p>Usually, user defined global real scalar variable should be updated each controller cycle, so the calculation should be done in a separate buffer inside an endless LOOP statement.</p> <p>The argument should be zero, if AxesMask is defined.</p> |
| MinPosition | <p>[Optional] Specifies a minimal position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position.</p> <p>The MinPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |

MaxPosition

[Optional] Specifies a maximal position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position.

The **MaxPosition** is specified relatively to the user origin, defined by **InitOffset** and in the user units of the axes, defined by **AxesUsed**.

Example

```
LCFIXEDINT (Laser, ! Laser index in the system
            0x03, ! Axis 0 and 1 defines multi-axis trajectory for
generating pulses
            0.001, ! Pulse width 1 microsec
            10) ! Spacing between pulses is user 10 units (um, mm)
```

4.10.4 LCRandomDist

The function initializes either array based pulse firing mode or gating mode.

As pulses are generated along an actual motion trajectory, the **AxesUsed** argument defines which axes are used for multi-axis trajectory generation.

The Mode parameter defines either Pulse Firing or Gating mode is initialized.

The arguments **FirstIndex**, **LastIndex** define a first and a last elements in the **Points** array. Only elements from **FirstIndex** to **LastIndex** define points where pulses will be fired. Point coordinates are defined in the user units relatively to the user origin, defined by **InitOffset**.

The **States** argument is used for array based gating mode, where a laser is to be switched on/off at the predefined locations.

Additionally, the number of extra pulses, specified by **ExtraPulses**, with a period, specified by **ExtraPeriod**, can be generated after each pulse.

The **Inversion** argument defines if the pulse and state output need to be inverted.

The **UserVector** argument provides the ability to implement a user-defined multi-axes trajectory. The **AxesUsed** argument should be zero in this case.

The **MinPosition** and **MaxPosition** arguments define an operation zone range along the multi-axes trajectory. The laser pulses start outputting after reaching **MinPosition** and stop outputting after reaching **MaxPosition**.

Syntax

LCRANDOMDIST (**Index**, **Mode**, **AxesUsed**, **Width**, **FirstIndex**, **LastIndex**, **Points**[, **States**, **InitOffset**, **ExtraPulses**, **ExtraPeriod**, **Inversion**, **PulseResolution**, **UserVector**, **MinPosition**, **MaxPosition**])

Arguments

| Arguments | Comments |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 indexes (logical axes) for each LCM in the network. Third index of allocated logical axes should be specified. |

| | |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | For example, if LCM gets network indexes (logical axes) 4,5,6,7, then index 6 is to be used for referring the specific LCM module. |
| Mode | <p>Defines which mode, Pulse Firing or Gating, is initialized.</p> <p>1 – Pulse Firing mode</p> <p>2 – Gating mode</p> <p>For Gating mode, States argument should be specified.</p> |
| AxesUsed | <p>Mask that defines the axes, which are used for generating pulses along the multi-axis motion trajectory.</p> <p>For example, the following mask defines that axes 0 and 2 should be used for vector velocity calculation:</p> <p>0b0101 (0b – is a prefix for binary value specification) or</p> <p>0x5 (0x – is a prefix for hexadecimal value specification)</p> <p>Only first 32 axes (0..31) can be referred by AxesUsed.</p> <p>If AxesUsed is zero and UserVector is zero, then the previously defined fixed distance pulse firing mode is canceled.</p> <p>The AxesUsed should be zero, if the UserVector is used for the user-defined trajectory calculation.</p> |
| Width | Pulse Width in milliseconds, range from 0.00002664msec (26.64nsec) to 1.745msec |
| FirstPoint | Specifies a first element in the Points and States arrays. The pulses start generating after reaching the position defined in this element. |
| LastPoint | Specifies a last element in the Points array. The pulses stop generating after reaching the position defined in this element. |
| Points | <p>Array of points. Each element of the array defines the point where pulse should be fired.</p> <p>The point coordinates are specified relatively to the user origin, defined by InitOffset in the user units of the axes, defined by AxesUsed.</p> |
| States | <p>[Optional] Array of states. Each element of the array defines whether the laser should be switched On or OFF.</p> <p>States is only used for the Array based Gating Mode.</p> |
| InitOffset | <p>[Optional] Specifies an initial position offset of multi-axis trajectory relatively to the user origin.</p> <p>By default, the function initializes an initial position of multi-axis trajectory as current actual position of the involved axes at the moment the function is called, so InitOffset is zero.</p> |

| | |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ExtraPulses | <p>[Optional] Number of additional pulses to be generated with ExtraPeriod period after each pulse at each Interval position.</p> <p>By default, no extra pulses are generated.</p> |
| ExtraPeriod | <p>[Optional] Time period in milliseconds for ExtraPulses additional pulses to be generated after each pulse at each Interval position.</p> <p>By default, no extra pulses are generated.</p> |
| Inversion | <p>[Optional] Inversion of pulse and states output logic</p> <p>0 (default) – no inversion</p> <p>1 – inversion</p> |
| PulseResolution | <p>[Optional] Pulse Resolution in user units.</p> <p>If the parameter is omitted or specified as zero, the function automatically calculates the most optimal pulse resolution according with maximal velocity (XVEL) of the axes, defined by AxesUsed, and the internal pulse generator maximal frequency, which is 12.5MHz.</p> <p>An actual pulse resolution is indicated in STEPF parameter.</p> <p>If the user needs to modify the resolution (higher or lower), this parameter allows to do it.</p> <p>The width of internal pulse generator is initialized to its minimal value of 40nsec to allow achieving the maximal pulses frequency.</p> |
| UserVector | <p>[Optional] The user defined global real scalar variable that contains result of multi-axes trajectory calculation. This option is used if it's required that pulse will be generated along a user-defined multi-axes trajectory, which implements some custom coordinate system transformation.</p> <p>Usually, user defined global scalar variable should be updated each controller cycle, so the calculation should be done in a separate buffer inside an endless LOOP statement.</p> <p>The argument should be zero, if AxesMask is defined.</p> |
| MinPosition | <p>[Optional] Specifies a minimal position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position.</p> <p>The MinPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |

MaxPosition

[Optional] Specifies a maximal position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position.

The **MaxPosition** is specified relatively to the user origin, defined by **InitOffset** and in the user units of the axes, defined by **AxesUsed**.

Example

```
LCFIXEDDIST (Laser, ! Laser index in the system
0x03, ! Axis 0 and 1 defines multi-axis trajectory for generating pulses
0.001, ! Pulse width 1 microsec
100, ! First pulse position
10, ! Spacing between pulses is user 10 units (um, mm)
000) ! Last pulse position
```

4.10.5 LCTickle

The function initializes Tickle mode. In this mode the laser control unit generates a signal at constant frequency and with constant width. Usually, this mode is used for those types of lasers that require gas ionization, during the time period when laser processing is off. By maintaining this mode, the laser will respond faster and more predictably, when the laser processing resumes.

Once this mode is initialized, the laser control unit constantly generates pulses, irrespective to any other operational modes. By default, the tickle mode generate pulses to the same output pin, which is used for the operational modes. In this case, the tickle mode pulses are superimposed on operational mode pulses. If necessary, the tickle mode pulses can be redirected to other available pins, see [Physical outputs configuration](#) for details.

Syntax

LCTICKLE (**Index**, **Frequency**, **Width**[, **Connector**])

| Arguments | Comments |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | <p>Defines which laser control unit is referred.</p> <p>The system allocates 4 indexes (logical axes) for each LCM in the network. The last index of allocated logical axes should be specified.</p> <p>For example, if LCM gets network indexes (logical axes) 4,5,6,7, so index 7 should be used for referring the specific LCM module.</p> |
| Frequency | <p>Tickle frequency in Hz</p> <p>Minimal frequency is 1149Hz, maximal frequency is 100,000Hz</p> <p>Not any frequency can be configured, so the function automatically rounds the specified frequency to the nearest supported value.</p> <p>Actual value of the configured tickle frequency is updated in LCTFREQ variable. The same index that is defined in LCTICKLE should be used as index for LCTFREQ.</p> |

| | |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Width | <p>Pulse Width in milliseconds.</p> <p>Minimal width is 0.000107msec (106nsec), maximal width is 0.0273msec (27.3µsec)</p> <p>Not any width can be configured, so the function automatically rounds the specified width to the nearest supported value.</p> <p>Actual value of the configured tickle pulse width is updated in LCTWIDTH variable. The same index that is defined in LCTICKLE should be used as index for LCTWIDTH.</p> |
| Connector | <p>[Optional] Specifies which connector (J1/J2/J3) to output tickle pulses for:</p> <p>1 - J1</p> <p>2 - J2</p> <p>3 - J3</p> |

4.10.6 LCZone

Syntax

LCZONE (**Index**, **AxesUsed**, **MinPosition**, **MaxPosition** [, **InitOffset**, **PulseResolution**])

| Arguments | Comments |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | <p>Defines which Laser Control unit is referred.</p> <p>The system allocates 4 indexes (logical axes) for each LCM in the network. The Index parameter should corresponds to the first logical axis (for Zone 0), second logical axis (for Zone 1) or third logical axis (Zone 2) of the specific LCM unit.</p> <p>For example, if LCM gets network indexes (logical axes) 4,5,6,7, so index 4 should be used for referring Zone 0. It's recommended to refer only Zone 0 or Zone 1, as Zone 2 is used by LCFixedDist or LCFixedInt functions.</p> |
| AxesUsed | <p>Mask that defines the axes, which are used for generating single-axis or multi-axis trajectory that is used for comparing its actual position with operation zone boundaries. The laser pulses are allowed to be outputted only within operation zone.</p> <p>For example, the following mask defines that axes 0 and 2 should be used for vector velocity calculation:</p> <p>0b0101 (0b – is a prefix for binary value specification) or</p> <p>0x5 (0x – is a prefix for hexadecimal value specification)</p> <p>Only first 32 axes (0..31) can be referred by AxesUsed.</p> <p>If AxesUsed is zero, the previously defined zone is canceled.</p> |

| | |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MinPosition | <p>Specifies a minimal position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position.</p> <p>The MinPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |
| MaxPosition | <p>Specifies a maximal position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position.</p> <p>The MaxPosition is specified relatively to the user origin, defined by InitOffset and in the user units of the axes, defined by AxesUsed.</p> |
| InitOffset | <p>[Optional] Specifies an initial position offset of multi-axis trajectory relatively to the user origin.</p> <p>By default, the function initializes an initial position of multi-axis trajectory as current actual position of the involved axes at the moment the function is called, so InitOffset is zero</p> |
| PulseResolution | <p>[Optional] Pulse Resolution in user units.</p> <p>If the parameter is omitted or specified as zero, the function automatically calculates the most optimal pulse resolution according with maximal velocity (XVEL) of the axes, defined by AXESUSED, and the internal pulse generator maximal frequency, which is 12.5MHz.</p> <p>An actual pulse resolution is indicated in STEPF parameter.</p> <p>If the user needs to modify the resolution (higher or lower), this parameter allows to do it.</p> <p>The width of internal pulse generator is initialized to its minimal value of 40nsec to allow achieving the maximal pulses frequency.</p> |

Example

```
LCZONE (Laser, ! Laser index in the system
        0, ! Zone 0 is used
        0, ! Zone is defined for the axis 0
        1000, 2000) ! Laser zone range is 1000 ... 2000
```

Vector path indication (GPATH)

If the **AxesUsed** parameter specifies involved an axes mask, then the function calculates the single-axis or multi-axis trajectory path. The current value of the path is indicated in the **GPATH** parameter. The **GPATH** parameter index corresponds to the first logical axis (for Zone 0), second logical axis (for Zone 1) or third logical axis (Zone 2) of the specific LCM unit. For example, if LCM gets network indexes (logical axes) 4,5,6,7, then index 5 is to be used for monitoring the current vector path for the Zone 1.

4.10.6.1 LCZoneSet

The minimum and maximum zone limit can be changed using LCZoneSet function.

Syntax

LCZONESET (**Index**, **MinPosition**, **MaxPosition**)

Arguments

| Arguments | Comments |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Should specify the same zone that was defined by LaserZone function. |
| MinPosition | Specifies a minimal position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position. The MinPosition is specified relatively to the initial position and in the user units, calculated by LaserZone function. |
| MaxPosition | Specifies a maximal position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position. The MaxPosition is specified relatively to the initial position and in the user units, calculated by LaserZone function. |

Example

```
LCZONESET (Zone0, ! Laser index in the system
           1000, 2000) ! Laser zone range is 1000 ... 2000
```

4.10.6.2 LCZoneGet

The function **LCZoneGet** returns the limits of the laser operation zone that was previously defined by **LCZone** function or by **LCZoneSet** function.

Syntax

LCZONEGET (**Index**, **ZoneLimit**)

Arguments

| Arguments | Comments |
|------------------|-----------------------------------------------------------------------------|
| Index | Should specify the same zone that was defined by LaserZone function. |
| ZoneLimit | 0 – to get a minimal zone limit 1 - to get a maximal zone limit |

Return Value

The function returns the currently configured limit of the specified laser zone.

Example

```
MinLimit = LCZONEGET (Zone0, ! Laser index in the system
                    0) ! Minimal limit of the zone
```

4.10.7 LCStop

The function stops any previously initialized laser mode and resets the previously defined mode parameters. The function is useful in order to ensure that any of the previously defined modes are stopped.

Syntax

LCStop (**Index**)

Arguments

| Arguments | Comments |
|--------------|---------------------------------------------------------------------------------------|
| Index | Designates the specific axis to which stopping and resetting the mode will be applied |

4.10.8 LCSignalSet

The function configures LCS output conditioning state.

Syntax

LCSignalSet (**Index**, **ConditionMask**)

Arguments

| Arguments | Comments |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 axes for each LCM in the network. First of the allocated axes should be specified. |

ConditionMask

Bit code that defines the logic conditions for Laser Control Signal (LCS) outputting.

By default, the LCS is generated with no dependance on other signals. Means that only bit 11 (LCS) is set.

Using this function, the user can define additional conditions for LCS outputting. Each bit that is set to 1, adds the dependance of LCS on the specific internal signal.

For example, if necessary that LCS (bit 11) is generated only if InRange 0 signal (bit 4) is "1" and PEG State 0 (bit 12) is "1", then the bit code 0x00001810 should be defined: bits 4, 11 and 12 are set.

| Bit | Signal |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | P/D Pulse 0 |
| 1 | P/D Dir 0 |
| 2 | PEG Pulse The bit is automatically set by LCFixedDist function or LCRandomDist function in case of the Pulse Firing Mode |
| 3 | PEG Active |
| 4 | InRange 0 The bit is automatically set by LCZone function |
| 5 | P/D Pulse 1 |
| 6 | P/D Dir 1 |
| 7 | InRange 1 The bit is automatically set by LCZone function |
| 8 | P/D Pulse 2 |
| 9 | P/D Dir 2 |
| 10 | InRange 2 The bit is automatically set by LCZone function or LCRandomDist or LCFixedDist or LCFixedInt |
| 11 | Laser Control Signal (by default = 1) The bit is automatically set by LCModulation function |

| Bit | Signal |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| 12 | PEG State #0 The bit is automatically set by LCRandomDist function in case of the Gating mode |
| 13 | PEG State #1 |
| 14 | PEG_State #2 |
| 15 | Tickle output The bit is automatically set by LCTickle function if Connector argument is not specified or omitted |
| 16 | Input 0 |
| 17 | Input 1 |
| 18 | Input 2 |
| 19 | Input 3 |
| 20 | Input 4 |
| 21 | Input 5 |
| 22 | Input 6 |
| 23 | Input 7 |
| 24 | Output 0 |
| 25 | Output 1 |
| 26 | Output 2 |
| 27 | Output 3 |
| 28 | Output 4 |
| 29 | Output 5 |
| 30 | Output 6 |
| 31 | Output 7 |



If it is necessary to set new bits and to preserve the bits that have been previously set, it is recommended to begin by retrieving the current state by using the function **LCSignalGet** and then to set the new bits, see [LCS conditioning example](#).



Inputs 0-7 conditioning is not supported in SPiiPlusCMHP/BA.

4.10.9 LCSignalGet

The function returns LCS output conditioning state.

Syntax

LCSignalGet (**Index**)

Arguments

The function returns the value that represents the outputs configuration state depending on the **Connector** parameter.

| Arguments | Comments |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 axes for each LCM in the network. First of the allocated axes should be specified. |
| | Bit code that defines the logic conditions for Laser Control Signal (LCS) outputting. The meaning if the bits is desribed in LCSignalSet arguments table. |

Return value

The function returns bit code that defines the logic conditions for LCS outputting. The meaning if the bits is desribed in **LCSignalSet** arguments table.

4.10.10 LCS conditioning example

The example shows how to configure additional conditions to LCS outputting function.

```
! Configure laser output conditioning
! Bits 4(InRange0), 7(InRange1) are added to LCS conditioning
! Means that LCS is outputted only if both InRange0 and InRange1 signals
are "1"
LCSignalSet (Index, LCSignalGet (Index) | 0x0090)

! Add Tickle output to the LCS
! Add Tickle output (Bit 15) to LCS conditioning
! Means that tickle mode is superimposed on signal generated by PFG
```

```
module (modes 1, 2 or 3)
LCSignalSet (Index, LCSignalGet (Index) | 0x8000)
```

4.10.11 Physical outputs configuration

The LCM provides the following flexibility in digital outputs configuration: the logical outputs of the internal modules, like P/D pulse, PEG pulse, InRange can be redirected to available pins of J1, J2 and J3 connectors. The configuration is done by [LCOutputSet](#) function, described below. The outputs configuration can be retrieved using [LCOutputGet](#) function.

4.10.11.1 LCOutputSet

The LCOutputSet function configures LCM physical outputs.

Syntax

LCOutputSet (**Index**, **Connector**, **Code**)

| Arguments | Comments |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 axes for each LCM in the network. First of the allocated axes should be specified. |
| Connector | Specifies LCM connector: 01 – J1 02 – J2 03 – J3 |

| Arguments | Comments | | | |
|-----------|-----------------|--------------------------|--------------------|--------------------|
| Code | Connector value | Return value | #_PULSE output | #_DIR output |
| | 01 or 02 or 03 | 0x00 | None | None |
| | 01 | 0x01 *See note below. | Counter 0 A signal | Counter 0 B signal |
| | 01 | 0x02 *See note below. | Pulse 0 | Direction 0 |
| | 01 | 0x03 | In range 0 | "1" |
| | 01 or 02 or 03 | 0x04 | PEG pulse | PEG active |
| | 01 or 02 or 03 | 0x05 | PEG state 0 | PEG active |
| | 01 or 02 or 03 | 0x09 | Tickle | "1" |
| | 02 | 0x01 *See note below. | Counter 1 A signal | Counter 1 B signal |
| | 02 | 0x02 *See note below. | Pulse 1 | Direction 1 |
| | 02 | 0x03 | In range 1 | "1" |
| | 03 | 0x01 *See note below. | Counter 2 A signal | Counter 2 B signal |
| | 03 | 0x02 *See note below. | Pulse 2 | Direction 2 |
| | 03 | 0x03 | In range 2 | "1" |



*Only supported by LCM modules with the AqB and P/D output option.

4.10.11.2 LCOutputGet

The **LCOutputGet** function returns LCM outputs configuration.

Syntax

LCOutputGet (**Index**, **Connector**)

Arguments

The function returns the value that represents the outputs configuration state depending on the **Connector** parameter.

| Arguments | Comments |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 axes for each LCM in the network. First of the allocated axes should be specified. |
| Connector | Specifies LCM connector: 01 – J1 02 – J2 03 – J3 |

Return value

| Connector value | Return value | #_PULSE output | #_DIR output |
|-----------------|--------------|--------------------|--------------------|
| 01 or 02 or 03 | 0x00 | None | None |
| 01 | 0x01 | Counter 0 A signal | Counter 0 B signal |
| 01 | 0x02 | Pulse 0 | Direction 0 |
| 01 | 0x03 | In range 0 | "1" |
| 01 or 02 or 03 | 0x04 | PEG pulse | PEG active |
| 01 or 02 or 03 | 0x05 | PEG state 0 | PEG active |
| 01 or 02 or 03 | 0x09 | Tickle | "1" |
| 02 | 0x01 | Counter 1 A signal | Counter 1 B signal |
| 02 | 0x02 | Pulse 1 | Direction 1 |

| Connector value | Return value | #_PULSE output | #_DIR output |
|-----------------|--------------|--------------------|--------------------|
| 02 | 0x03 | In range 1 | "1" |
| 03 | 0x01 | Counter 2 A signal | Counter 2 B signal |
| 03 | 0x02 | Pulse 2 | Direction 2 |
| 03 | 0x03 | In range 2 | "1" |

4.10.12 LCDelaySet

The pulse generation delay can also be configured by the user using **LCDelaySet** function. The specified delay will override the default value that is set by default. The currently configured delay can be read using **LCDelayGet** function.

Syntax

LCDelaySet (**Index**, **Delay**)

Arguments

| Arguments | Comments |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 axes for each LCM in the network. First of the allocated axes should be specified. |
| Delay | Delay in microseconds: 0-CTIME*1000(μsec) |

4.10.13 LCDelayGet

The function returns the actual currently configured delay in microseconds.

Syntax

LCDelayGet (**Index**)

Arguments

| Arguments | Comments |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | Defines which Laser Control unit is referred. The system allocates 4 axes for each LCM in the network. First of the allocated axes should be specified. |

Return Value

The function returns the actual currently configured delay in microseconds.

4.10.14 AxListAsMask

In the functions **LcModulation**, **LcFixedDist**, **LcFixedInt**, **LcRandomDist**, **LcZone** an argument requires a mask to define the axes. The mask specification is defined with **AxListAsMask**.

Syntax

`AxListAsMask (axis_list)`

Arguements

| Arguments | Comments |
|------------------------|-----------------------------------------------|
| <code>axis_list</code> | List of the axes, separated by comma: 0, 2, 4 |

Return value

The function returns the integer value that represents the axis list as a mask.

Example

```
int AxisMask
AxisMask = AxListAsMask(0,2,4) ! AxisMask gets value 0x15 (0b00010101)
description in the manual.
```

4.11 Dynamic Error Compensation

The Dynamic Error Compensation functions are:

| Function | Description |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ERRORMAP1D</code> | Configures and activates 1D error correction for the mechanical error compensation for the specified zone, |
| <code>ERRORMAP2D</code> | Configures and activates 2D error correction for the mechanical error compensation of the 'axis0' command for the specified zone |
| <code>ERRORMAPN1D</code> | Configures and activates 1D error correction for the mechanical error compensation for the specified zone |
| <code>ERRORMAPN2D</code> | Configures and activates 2D error correction for the mechanical error compensation of the 'axis0' command for the specified zone, |
| <code>ERRORMAPA1D</code> | Configures and activates 1D error correction for the mechanical error compensation for the specified zone, so that the compensated reference position will be calculated by multiplying the scaling factor by the desired position so that the actual value will be closer to the desired value. |
| <code>ERRORMAPA2D</code> | Configures and activates 2D error correction for the mechanical error compensation of the specified axis for the specified zone, so that the compensated reference position will be calculated by taking into account the angle for the orthogonality correction so that the actual value will be closer to the desired value. |

| Function | Description |
|---------------------------|----------------------------------------------------------------------------------------------------|
| <code>ERRORMAPOFF</code> | Deactivates error mapping correction for the mechanical error compensation for the specified zone. |
| <code>ERRORMAPON</code> | Activates error correction for the mechanical error compensation for the specified zone. |
| <code>#ERRORMAPREP</code> | Generates a report of all activated zones of error mapping for all axes in the system. |
| <code>ERRORUNMAP</code> | Deactivates error correction for the mechanical error compensation for the specified zone. |

4.11.1 `ERRORMAP1D`

Description

The `ERRORMAP1D` function configures and activates 1D error correction for the mechanical error compensation for the specified zone, so that the compensated reference position will be calculated by subtracting the linearly (by default) interpolated error from the desired position so that the actual value will be closer to the desired value.

The calculation assumes fixed Intervals between points inside the zone.

Syntax

```
ERRORMAP1D[switches] axis, zone, base, step, correction_map, [referenced_
axis_or_analog_input]
```

Arguments

| | |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| base | A real number representing the axis command that corresponds to the first point in correction table for mechanical error compensation. |
| step | A real number representing the fixed interval distance between the two adjacent axis commands. |
| correction_map | The name of a real one-dimensional array that specifies correction table for mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |

referenced_
axis_or_analog_
input

[Optional] The index of the axis, or the index of the analog input that the mechanical error compensation will be calculated based on its feedback.

Switches

| | |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------|
| /p | Prevent applying dynamic error compensation on INDEX , MARK , and PEG values |
| /a | Specifies that the mechanical error compensation will be calculated based on the feedback from the axis specified by the optional parameter. |
| /i | Specifies that the mechanical error compensation will be calculated based on the feedback from the analog input indicated by the optional parameter. |

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined array size is less than the defined number of array points.
- > Error 3113, The step in the table is zero or negative, when the step argument is zero or negative.

Comments

If erroneous parameters are passed to the function, the corresponding runtime error will be generated. The function is intended to be used with arrays only, meaning that an error is generated if a scalar is passed as a parameter.

4.11.2 *ERRORMAPN1D*

Description

The **ERRORMAPN1D** function configures and activates 1D error correction for the mechanical error compensation for the specified zone, so that the compensated reference position will be calculated by subtracting the linearly (by default) interpolated error from the desired position so that the actual value will be closer to the desired value.

The calculation is based on an arbitrary network of points inside the zone.

Syntax

```
ERRORMAPN1D[switches] axis, zone, axis_command, correction_map,  
[referenced_axis_or_analog_input]
```

Arguments

| | |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| axis_command | The name of a real one-dimensional array that specifies axis command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_map | The name of a real one-dimensional array that specifies correction table for mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input | [Optional] The index of the axis or the index of the analog input to be used for the calculation of the mechanical error compensation. |

Switches

| | |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------|
| /p | Prevent applying dynamic error compensation on INDEX, MARK, and PEG values |
| /a | Specifies that the mechanical error compensation will be calculated based on the feedback from the axis specified by the optional parameter. |
| /i | Specifies that the mechanical error compensation will be calculated based on the feedback from the analog input indicated by the optional parameter. |

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined array size is less than the defined number of array points.
- > Error 3113, The step in the table is zero or negative, when the step argument is zero or negative.

Comments

In case of erroneous parameters, the relevant runtime error will be generated. The function is intended for use with arrays, meaning that an error is generated if a scalar is given as a parameter.

4.11.3 ERRORMAPA1D**Description**

The **ERRORMAPA1D** function configures and activates 1D error correction for the mechanical error compensation for the specified zone, so that the compensated reference position will be calculated

by multiplying the scaling factor by the desired position so that the actual value will be closer to the desired value.

Syntax

```
ERRORMAP1D[switches] axis, zone, scaling_factor, offset
```

Arguments

| | |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| scaling_factor | The scaling factor for the linear alignment that will be used for mechanical error compensation. The allowed range for the scaling factor is (0, 2.0). |
| offset | The offset for the linear alignment that will be used for mechanical error compensation. The offset is actually the mechanical error compensation for the 0-point location. |

Switches

| | |
|----|----------------------------------------------------------------------------|
| /p | Prevent applying dynamic error compensation on INDEX, MARK, and PEG values |
|----|----------------------------------------------------------------------------|

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined array size is less than the defined number of array points.
- > Error 3113, The step in the table is zero or negative, when the step argument is zero or negative.

Comments

In case of erroneous parameters, the corresponding runtime error will be generated. The function is intended to be used for arrays only, meaning that an error is generated if a scalar is given as a parameter.

This command is supported in ADK versions 2.70 and higher.

4.11.4 *ERRORMAP2D*

Description

The **ERRORMAP2D** function configures and activates 2D error correction for the mechanical error compensation of the **'axis0'** or **'axis1'** command (depending on the switch used) for the specified zone, so that the compensated reference position will be calculated by subtracting the linearly (by default) interpolated error from the desired position so that the actual value will be closer to the desired value.

Syntax

```
ERRORMAP2D[switches] (axis0, axis1), zone, base0, step0, base1, step1,
correction_map, [reference_axis_or_analog_input0, referenced_axis_or_
analog_input1]
```

Arguments

| | |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the first axis that the mechanical error compensation will be applied to. Valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 2D mechanical error compensation. Valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| base0 | A real number representing the 'axis0' command that corresponds to the first point in correction table for mechanical error compensation. |
| step0 | A real number representing the fixed interval distance between the two adjacent 'axis0' commands. |
| base1 | A real number representing the 'axis1' command that corresponds to the first point in correction table for mechanical error compensation. |
| step1 | A real number representing the fixed interval distance between the two adjacent 'axis1' commands. |
| correction_map | The name of a real two-dimensional array that specifies correction table for mechanical error compensation. The array type should be a GLOBAL REAL STATIC defined in the D-Buffer. |
| referenced_axis_or_analog_input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_axis_or_analog_input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /a | Specifies that the mechanical error compensation will be calculated based on the feedback from the axis specified by the optional parameter. |
| /i | Specifies that the mechanical error compensation will be calculated based on the feedback from the analog input indicated by the optional parameter. |
| /aj | Specifies that the first optional parameter will be treated as an analog input index. |
| /ak | Specifies that the second optional parameter will be treated as an analog input index. |

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined array size is less than the defined number of array points.
- > Error 3113, The step in the table is zero or negative, when the step argument is zero or negative.

Comments

If incorrect parameters are passed to the function the corresponding error will be generated. The function is intended to be used for arrays only, meaning that an error is generated if a scalar is given as a parameter.

4.11.5 *ERRORMAPN2D*

Description

The **ERRORMAPN2D** function configures and activates 2D error correction for the mechanical error compensation of the '**axis0**' or '**axis1**' command (depending on the switch used) for the specified zone, so that the compensated reference position will be calculated by subtracting the linearly (by default) interpolated error from the desired position so that the actual value will be closer to the desired value.

Syntax

```
ERRORMAPN2D[switches] (axis0, axis1), zone, axis0_command, axis1_command,
correction_map, [reference_axis_or_analog_input0,
referenced_axis_or_analog_input1]
```

Arguments

| | |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 2D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| axis0_command | The name of a real one-dimensional array that specifies 'axis0' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis1_command | The name of a real one-dimensional array that specifies 'axis1' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_map | The name of a real one-dimensional array that specifies correction table for mechanical error compensation. The array should be a GLOBAL REAL STATIC defined in the D-Buffer. |
| referenced_axis_or_analog_input0 | [Optional] The index of the first axis or the index of the first analog input providing the feedback used for calculation of the mechanical error compensation. |
| referenced_axis_or_analog_input1 | [Optional] The index of the second axis or the index of the second analog input providing the feedback used for calculation of the mechanical error compensation. |

Switches

| | |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default). |
| /1 | The mechanical error compensation will be applied to 'axis1'. |
| /a | Specifies that the mechanical error compensation will be calculated based on the feedback from the axis specified by the optional parameter. |
| /i | Specifies that the mechanical error compensation will be calculated based on the feedback from the analog input indicated by the optional parameter. |
| /aj | Specifies that the first optional parameter will be treated as an analog input index. |
| /ak | Specifies that the second optional parameter will be treated as an analog input index. |

Error Conditions

The function detects the following error conditions:

- > Error 2044, index is out of range, when the defined array size is less than the defined number of array points.
- > Error 3113, the step in the table is zero or negative, when the step argument is zero or negative.

Comments

If incorrect parameters are passed to the function the relevant error will be generated. The function is intended to be used for arrays only, meaning that an error is generated if a scalar is given as a parameter.

4.11.6 *ERRORMAPA2D*

Description

The **ERRORMAPA2D** function configures and activates 2D error correction for the mechanical error compensation of the specified axis for the specified zone, so that the compensated reference position will be calculated by taking into account the angle for the orthogonality correction so that the actual value will be closer to the desired value.

Syntax

```
ERRORMAPA2D[switches] (axis0, axis1), zone, angle
```

Arguments

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 2D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| angle | The angle for the orthogonality correction that will be used for mechanical error compensation. The allowed range for the angle is [-45°, 45°]. |

Switches

| | |
|----|------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |

Error Conditions

The function detects the following error conditions:

- > Error 2044, Index is out of range, when the defined array size is less than the defined number of array points.

- > Error 3113, The step in the table is zero or negative, when the step argument is zero or negative.

Comments

In case of wrong parameters, the corresponding runtime error will be generated. The function is intended to be used for arrays only, meaning that an error is generated if a scalar is given as a parameter.

This command is supported in ADK versions 2.70 and higher.

4.11.7 *ERRORMAP3DA*

Description

The **ERRORMAP3D3** function configures and activates 3D error correction for the mechanical error compensation of 'axis0', 'axis1', and 'axis2' for the specified zone, so that the compensated reference position will be calculated by adding the interpolated error from the desired position so that the actual value will be closer to the desired value. Interpolation is linear by default, other options are available.

The **ERRORMAP3DA** function receives the indices of three axes, the zone index, the base value of the 'axis0' command, the fixed defined interval of the 'axis0' command, the base value of the 'axis1' command, a fixed defined interval for the 'axis1' command, the base value of the 'axis2' command, the fixed defined interval of the 'axis2' command, and 10 2D correction tables correlated to the specified 'axis2' coordinates for mechanical error compensation.

Syntax

```
ERRORMAP3DA[command options] (axis0, axis1, axis2), zone, base1, step1,
base2, step2, base3, step3, correction_map0, correction_map1, correction_
map2, correction_map3 , correction_map4, correction_map5, correction_
map6, correction_map7, correction_map8, correction_map9[, reference_axis_
or_analog_input0, referenced _axis_or_analog_input1, referenced _axis_or_
analog_input2]
```

Arguments

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |

| | |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| base0 | A real number representing the 'axis0' command that corresponds to the first point in correction table for mechanical error compensation. |
| step0 | A real number representing the fixed interval distance between the two adjacent 'axis0' commands. |
| base1 | A real number representing the 'axis1' command that corresponds to the first point in correction table for mechanical error compensation. |
| step1 | A real number representing the fixed interval distance between the two adjacent 'axis1' commands. |
| base2 | A real number representing the 'axis2' command that corresponds to the first point in correction table for mechanical error compensation. |
| step2 | A real number representing the fixed interval distance between the two adjacent 'axis2' commands. |
| correction_ map0 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '0' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '1' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map2 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '2' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map3 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '3' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map4 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '4' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map5 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '5' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| correction_ map6 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '6' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map7 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '7' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map8 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '8' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map9 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '9' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter to be treated as an analog input index. |
| /ak | Specifies that the second optional parameter to be treated as an analog input index. |
| /al | Specifies that the third optional parameter to be treated as an analog input index. |

Error Conditions

The function detects the following error conditions.

- > **2044** - Index is out of range, when the defined array size is less than the defined number of array points.
- > **3113** - The step in the table is zero or negative, when the step argument is zero or negative.
- > **3384** - The specified suffix combination is invalid. Please see the documentation for more details.
- > **3413** - The supplied correction maps (2-dimensional arrays) should have the same dimensions. Please see the documentation for more details.
- > **3414** - The Specified referenced axes/analog inputs are invalid. Please see the documentation for more details.

Comments

If incorrect parameters are passed to the function the corresponding runtime error is generated. The function is intended to be used only with arrays; thus, an error is generated if a scalar is given as a parameter.

4.11.8 *ERRORMAP3D2*

Description

The **ERRORMAP3D2** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the interpolated error from the desired position so that the actual value will be closer to the desired value. Interpolation is linear by default; other options are available.

The **ERRORMAP3D2** function receives indexes of three axes, zone index, base value of the 'axis0' command, fixed defined interval of the 'axis0' command, base value of the 'axis1' command, fixed defined interval of the 'axis1' command, base value of the 'axis2' command, fixed defined interval of the 'axis2' command, and two 2D correction tables (in correlation to the specified 'axis2' coordinates) for mechanical error compensation.

Syntax

```
ERRORMAP3D2[command options] (axis0, axis1, axis2), zone, base0, step0,
base1, step1, base2, step2, correction_map0, correction_map1[, reference_
axis_or_analog_input0, referenced_axis_or_analog_input1, referenced _
axis_or_analog_input2]
```

Arguments

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| base0 | A real number representing the 'axis0' command that corresponds to the first point in correction table for mechanical error compensation. |
| step0 | A real number representing the fixed interval distance between the two adjacent 'axis0' commands. |
| base1 | A real number representing the 'axis1' command that corresponds to the first point in correction table for mechanical error compensation. |
| step1 | A real number representing the fixed interval distance between the two adjacent 'axis1' commands. |
| base2 | A real number representing the 'axis2' command that corresponds to the first point in correction table for mechanical error compensation. |
| step2 | A real number representing the fixed interval distance between the two adjacent 'axis2' commands. |
| correction_ map0 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '0' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '1' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter will be regarded as an analog input index. |
| /ak | Specifies that the second optional parameter will be regarded as an analog input index. |
| /al | Specifies that the third optional parameter will be regarded as an analog input index. |
| /e | Specifies that the extrapolation will be used for Z values that are beyond the original observation range. That is, we are assuming that existing trends will continue (by using the nearest correction table to estimate the correction). This method is subject to greater uncertainty and a higher risk of producing meaningless results than interpolation. |

Comments

- > When incorrect parameters are specified, the relevant compile-time or run-time error is generated. See [ACSPL+ Runtime Errors](#) for explanation of error 2044 and [ACSPL+ Compilation Errors](#) for explanations of errors 3113, 3384, 3413, 3414. The function is intended for use with arrays only; an error is generated if a scalar is given as a parameter.

Examples**Example 1**

This example uses two 2-dimensional arrays to create 2-dimensional correction maps that are used for the 3D Dynamic error compensation. Each map represents a different value (height) of the Z axis. An XYZ cuboid (rectangular prism) zone starts at coordinate -100 for X, 30 for Y, and 30 for Z. It has a fixed interval of 10 mm for each axis. The user units are represented in mm.

```
D-Buffer:
global real static X_Correction1(3) (10)
global real static X_Correction2(3) (10)

Buffer:
local int X_axis
local int Y_axis
local int Z_axis
local int zone
local int X_base
local int X_step
local int Y_base
local int Y_step
```

```

local int Z_base local int Z_step

X_axis=0      ! First moving axis
Y_axis=1      ! Second moving axis
Z_axis=2      ! Third moving axis
zone =0       ! Index of error mapping zone in use
X_base=-100   ! X starting coordinate of error mapping zone
Y_base=30     ! Y starting coordinate of error mapping zone
X_step=10     ! X interval in error map table
Y_step=10     ! Y interval in error map table
Z_base=30     ! Z starting coordinate of error mapping zone
Z_step=10     ! Z interval in error map table
              ! Correction table for Error Mapping for X axis
              ! of the first Z height - declared in D-Buffer
X_Correction1(0)(0)=0; X_Correction1(0)(1)=0.3; X_Correction1(0)(2)=-0.4;
X_Correction1(0)(3)=-0.7; X_Correction1(0)(4)=-0.1;
X_Correction1(0)(5)=0.22; X_Correction1(0)(6)=0.55; X_Correction1(0)(7)=0.2;
X_Correction1(0)(8)=-0.3;
X_Correction1(0)(9)=0;
X_Correction1(1)(0)=0; X_Correction1(1)(1)=0.4; X_Correction1(1)(2)=-0.4;
X_Correction1(1)(3)=-0.5; X_Correction1(1)(4)=-0.2;
X_Correction1(1)(5)=0.24; X_Correction1(1)(6)=0.3; X_Correction1(1)(7)=0.32;
X_Correction1(1)(8)=-0.13;
X_Correction1(1)(9)=0;
X_Correction1(2)(0)=0; X_Correction1(2)(1)=0.1; X_Correction1(2)(2)=-0.2;
X_Correction1(2)(3)=-0.4; X_Correction1(2)(5)=0.3;
X_Correction1(2)(6)=0.4; X_Correction1(2)(7)=0.33; X_Correction1(2)(8)=-0.1;
X_Correction1(2)(9)=0
! Correction table for Error Mapping for X axis of the second Z height
! - declared in D-Buffer
X_Correction2(0)(0)=0; X_Correction2(0)(1)=0.23; X_Correction2(0)(2)=-0.12;
X_Correction2(0)(3)=-0.3; X_Correction2(0)(4)=-0.2
X_Correction2(0)(5)=0.26; X_Correction2(0)(6)=0.15; X_Correction2(0)(7)=0.02;
X_Correction2(0)(8)=-0.15;
X_Correction2(0)(9)=0
X_Correction2(1)(0)=0; X_Correction2(1)(1)=0.24; X_Correction2(1)(2)=-0.14;
X_Correction2(1)(3)=-0.34;
X_Correction2(1)(4)=-0.2
X_Correction2(1)(5)=0.14; X_Correction2(1)(6)=-0.23; X_Correction2(1)(7)=-0.32; X_
Correction2(1)(8)=-0.13; X_Correction2(1)(9)=0
X_Correction2(2)(0)=0; X_Correction2(2)(1)=-0.1; X_Correction2(2)(2)=-0.32;
X_Correction2(2)(3)=-0.4; X_Correction2(2)(4)=0.15
X_Correction2(2)(5)=0.36; X_Correction2(2)(6)=0.44; X_Correction2(2)(7)=0.23; X_Correction2
(2)(8)=-0.11;
X_Correction2(2)(9)=0
! X axis error mapping function configuration
ERRORMAP3D2/0 (X_axis, Y_axis, Z_axis), zone, X_base, X_step, Y_base, Y_step, Z_base, Z_
step, X_Correction1, X_Correction2
! Enable error mapping
ERRORMAPON X_axis, zone
STOP

```

Example 2

This example uses two 2-dimensional arrays to create 2-dimensional correction maps, that are used for the 3D Dynamic error compensation. Each map represents a different analog input value. An XYZ cuboid (rectangular prism) zone starts at coordinate -100 for X, 30 for Y, and 30% for the Z specified analog input. It has a fixed interval of 10 mm for each axis, and 10 percent for the Z analog input. The user units are represented in mm.


```

D-Buffer:
global real static X_Correction1(3)(10)
global real static X_Correction2(3)(10)
Buffer:
local int X_axis
local int Y_axis
local int Z_axis
local int zone
local int X_base
local int X_step
local int Y_base
local int Y_step

local int Z_base
local int Z_step
local int Z_referenced_analog_input

X_axis=0      ! First moving axis
Y_axis=1      ! Second moving axis
Z_axis=2      ! Third moving axis
zone =0       ! Index of error mapping zone in use
X_base=-100   ! X starting coordinate of error mapping zone
Y_base=30     ! Y starting coordinate of error mapping zone
X_step=10     ! X interval in error map table
Y_step=10     ! Y interval in error map table
Z_base=30     ! Z starting percentage of analog input
Z_step=10     ! Z interval
Z_referenced_analog_input = 1 ! AIN(1)!Correction table for Error Mapping
for X axis of the first Z value - declared in D-Buffer
X_Correction1(0)(0)=0; X_Correction1(0)(1)=0.3; X_Correction1(0)(2)=-0.4;
X_Correction1(0)(3)=-0.7; X_Correction1(0)(4)=-0.1;
X_Correction1(0)(5)=0.22; X_Correction1(0)(6)=0.55;
X_Correction1(0)(7)=0.2; X_Correction1(0)(8)=-0.3;
X_Correction1(0)(9)=0;
X_Correction1(1)(0)=0; X_Correction1(1)(1)=0.4; X_Correction1(1)(2)=-0.4;
X_Correction1(1)(3)=-0.5; X_Correction1(1)(4)=-0.2;
X_Correction1(1)(5)=0.24; X_Correction1(1)(6)=0.3; X_Correction1(1)
(7)=0.32; X_Correction1(1)(8)=-0.13;
X_Correction1(1)(9)=0;
X_Correction1(2)(0)=0; X_Correction1(2)(1)=0.1; X_Correction1(2)(2)=-0.2;
X_Correction1(2)(3)=-0.4; X_Correction1(2)(5)=0.3;
X_Correction1(2)(6)=0.4; X_Correction1(2)(7)=0.33; X_Correction1(2)(8)=-
0.1; X_Correction1(2)(9)=0
! Correction table for Error Mapping for X axis of the second Z value -
declared in D-Buffer
X_Correction2(0)(0)=0; X_Correction2(0)(1)=0.23;
X_Correction2(0)(2)=-0.12; X_Correction2(0)(3)=-0.3;
X_Correction2(0)(4)=-0.2
X_Correction2(0)(5)=0.26; X_Correction2(0)(6)=0.15;

```

```

X_Correction2(0)(7)=0.02; X_Correction2(0)(8)=-0.15;
X_Correction2(0)(9)=0
X_Correction2(1)(0)=0; X_Correction2(1)(1)=0.24;
X_Correction2(1)(2)=-0.14; X_Correction2(1)(3)=-0.34;
X_Correction2(1)(4)=-0.2
X_Correction2(1)(5)=0.14;
X_Correction2(1)(6)=-0.23; X_Correction2(1)(7)=-0.32;
X_Correction2(1)(8)=-0.13; X_Correction2(1)(9)=0
X_Correction2(2)(0)=0; X_Correction2(2)(1)=-0.1;
X_Correction2(2)(2)=-0.32; X_Correction2(2)(3)=-0.4;
X_Correction2(2)(4)=0.15
X_Correction2(2)(5)=0.36; X_Correction2(2)(6)=0.44;
X_Correction2(2)(7)=0.23; X_Correction2(2)(8)=-0.11;
X_Correction2(2)(9)=0
! X axis error mapping function configuration
ERRORMAP3D2/0a1 (X_axis, Y_axis, Z_axis), zone, X_base, X_step, Y_base,
Y_step, Z_base, Z_step, X_Correction1, X_Correction2, X_axis, Y_axis, Z_
referenced_analog_input
! Enable error mapping
ERRORMAPON X_axis, zone
STOP

```

4.11.9 ERRORMAP3D3

Description

The **ERRORMAP3D3** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the interpolated error from the desired position so that the actual value will be closer to the desired value. Interpolation is linear by default; other options are available.

ERRORMAP3D3 function receives indexes of three axes, zone index, base value of the 'axis0' command, fixed defined interval of the 'axis0' command, base value of the 'axis1' command, fixed defined interval of the 'axis1' command, base value of the 'axis2' command, fixed defined interval of the 'axis2' command, and three 2D correction tables (in correlation to the specified 'axis2' coordinates) for mechanical error compensation.

Syntax

```

ERRORMAP3D3[command options] (axis0, axis1, axis2), zone, base0, step0,
base1, step1, base2, step2, correction_map0, correction_map1, correction_
map2[, reference_axis_or_analog_input0, referenced_axis_or_analog_
input1, referenced_axis_or_analog_input2]

```

Arguments

| | |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| base0 | A real number representing the 'axis0' command that corresponds to the first point in correction table for mechanical error compensation. |
| step0 | A real number representing the fixed interval distance between the two adjacent 'axis0' commands. |
| base1 | A real number representing the 'axis1' command that corresponds to the first point in correction table for mechanical error compensation. |
| step1 | A real number representing the fixed interval distance between the two adjacent 'axis1' commands. |
| base2 | A real number representing the 'axis2' command that corresponds to the first point in correction table for mechanical error compensation. |
| step2 | A real number representing the fixed interval distance between the two adjacent 'axis2' commands. |
| correction_ map0 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '0' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '1' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map2 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '2' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter will be regarded as an analog input index. |
| /ak | Specifies that the second optional parameter will be regarded as an analog input index. |
| /al | Specifies that the third optional parameter will be regarded as an analog input index. |
| /e | Specifies that the extrapolation will be used for Z values that are beyond the original observation range. That is, we are assuming that existing trends will continue (by using the nearest correction table to estimate the correction). This method is subject to greater uncertainty and a higher risk of producing meaningless results than interpolation. |

Comments

- > When incorrect parameters are specified, the relevant compile-time or run-time error is generated. See [ACSPL+ Runtime Errors](#) for explanation of error 2044 and [ACSPL+ Compilation Errors](#) for explanations of errors 3113, 3384, 3413, 3414. The function is intended for use with arrays only; an error is generated if a scalar is given as a parameter.

Example

This example uses three 2-dimensional arrays to create 2-dimensional correction maps, that are used for the 3D Dynamic error compensation. Each map represents a different value(height) of the Z axis. The zone starts at coordinate -100 for X, 30 for Y, and 30 for Z. It has a fixed interval of 10 mm for each axis. The user units are represented in mm.

```

D-Buffer:
global real static X_Correction1(3)(10)
global real static X_Correction2(3)(10)
global real static X_Correction3(3)(10)

Buffer:
local int X_axis
local int Y_axis
local int Z_axis
local int zone
local int X_base
local int X_step
local int Y_base
local int Y_step

local int Z_base
local int Z_step

X_axis=0      ! First moving axis
Y_axis=1      ! Second moving axis
Z_axis=2      ! Third moving axis
zone =0       ! Index of error mapping zone in use
X_base=-100   ! X starting coordinate of error mapping zone
Y_base=30     ! Y starting coordinate of error mapping zone
X_step=10     ! X interval in error map table
Y_step=10     ! Y interval in error map table
Z_base=30     ! Z starting coordinate of error mapping zone
Z_step=10     ! Z interval in error map table!Correction table for Error
Mapping for X axis of the first Z height - declared in D-Buffer
X_Correction1(0)(0)=0; X_Correction1(0)(1)=0.3; X_Correction1(0)(2)=-0.4;
X_Correction1(0)(3)=-0.7; X_Correction1(0)(4)=-0.1;
X_Correction1(0)(5)=0.22; X_Correction1(0)(6)=0.55;
X_Correction1(0)(7)=0.2; X_Correction1(0)(8)=-0.3;
X_Correction1(0)(9)=0;
X_Correction1(1)(0)=0; X_Correction1(1)(1)=0.4; X_Correction1(1)(2)=-0.4;
X_Correction1(1)(3)=-0.5; X_Correction1(1)(4)=-0.2;
X_Correction1(1)(5)=0.24; X_Correction1(1)(6)=0.3;
X_Correction1(1)(7)=0.32; X_Correction1(1)(8)=-0.13;
X_Correction1(1)(9)=0;
X_Correction1(2)(0)=0; X_Correction1(2)(1)=0.1; X_Correction1(2)(2)=-0.2;
X_Correction1(2)(3)=-0.4; X_Correction1(2)(5)=0.3;
X_Correction1(2)(6)=0.4; X_Correction1(2)(7)=0.33;
X_Correction1(2)(8)=-0.1; X_Correction1(2)(9)=0
! Correction table for Error Mapping for X axis of the second Z height -
declared in D-Buffer
X_Correction2(0)(0)=0; X_Correction2(0)(1)=0.23;
X_Correction2(0)(2)=-0.12; X_Correction2(0)(3)=-0.3;
X_Correction2(0)(4)=-0.2
X_Correction2(0)(5)=0.26; X_Correction2(0)(6)=0.15;

```

```

X_Correction2(0)(7)=0.02; X_Correction2(0)(8)=-0.15;
X_Correction2(0)(9)=0
X_Correction2(1)(0)=0; X_Correction2(1)(1)=0.24;
X_Correction2(1)(2)=-0.14; X_Correction2(1)(3)=-0.34;
X_Correction2(1)(4)=-0.2
X_Correction2(1)(5)=0.14; X_Correction2(1)(6)=-0.23;
X_Correction2(1)(7)=-0.32; X_Correction2(1)(8)=-0.13;
X_Correction2(1)(9)=0
X_Correction2(2)(0)=0; X_Correction2(2)(1)=-0.1;
X_Correction2(2)(2)=-0.32; X_Correction2(2)(3)=-0.4;
X_Correction2(2)(4)=0.15
X_Correction2(2)(5)=0.36; X_Correction2(2)(6)=0.44;
X_Correction2(2)(7)=0.23; X_Correction2(2)(8)=-0.11;
X_Correction2(2)(9)=0
! Correction table for Error Mapping for X axis of the third Z height -
declared in D-Buffer
X_Correction3(0)(0)=0; X_Correction3(0)(1)=0.23;
X_Correction3(0)(2)=-0.12; X_Correction3(0)(3)=-0.3;
X_Correction3(0)(4)=-0.2
X_Correction3(0)(5)=0.26; X_Correction3(0)(6)=0.15;
X_Correction3(0)(7)=0.02; X_Correction3(0)(8)=-0.15;
X_Correction3(0)(9)=0
X_Correction3(1)(0)=0; X_Correction3(1)(1)=0.24;
X_Correction3(1)(2)=-0.14; X_Correction3(1)(3)=-0.34;
X_Correction3(1)(4)=-0.2
X_Correction3(1)(5)=0.14; X_Correction3(1)(6)=-0.23;
X_Correction3(1)(7)=-0.32; X_Correction3(1)(8)=-0.13;
X_Correction3(1)(9)=0
X_Correction3(2)(0)=0; X_Correction3(2)(1)=-0.1;
X_Correction3(2)(2)=-0.32; X_Correction3(2)(3)=-0.4;
X_Correction3(2)(4)=0.15
X_Correction3(2)(5)=0.36; X_Correction3(2)(6)=0.44;
X_Correction3(2)(7)=0.23; X_Correction3(2)(8)=-0.11;
X_Correction3(2)(9)=0

! X axis error mapping function configuration
ERRORMAP3D3/0 (X_axis, Y_axis, Z_axis), zone, X_base, X_step, Y_base, Y_
step, Z_base, Z_step, X_Correction1, X_Correction2, X_Correction3
! Enable error mapping
ERRORMAPON X_axis, zone

```

4.11.10 *ERRORMAPN3D2*

Description

The **ERRORMAPN3D2** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the linearly (by default) interpolated error from the desired position so that the actual value will be closer to the desired value.

ERRORMAPN3D2 function receives indexes of three axes, zone index, 'axis0' command table, 'axis1' command table, 'axis2' command table, and two 2D correction tables (in correlation to the specified 'axis2' coordinates) for mechanical error compensation.

Syntax

```
ERRORMAPN3D2[command options] (axis0, axis1, axis2), zone, axis0_command,
axis1_command, axis2_command, correction_map0, correction_map1[,
reference_axis_or_analog_input0, referenced_axis_or_analog_input1,
referenced_axis_or_analog_input2]
```

Arguments

| | |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| axis0_command | The name of a real one-dimensional array that specifies 'axis0' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis1_command | The name of a real one-dimensional array that specifies 'axis1' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis2_command | The name of a real one-dimensional array that specifies 'axis2' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_map0 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to axis 2's first specified coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_map1 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to axis 2's second specified coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter will be regarded as an analog input index. |
| /ak | Specifies that the second optional parameter will be regarded as an analog input index. |
| /al | Specifies that the third optional parameter will be regarded as an analog input index. |
| /e | Specifies that the extrapolation will be used for Z values that are beyond the original observation range. That is, we are assuming that existing trends will continue (by using the nearest correction table to estimate the correction). This method is subject to greater uncertainty and a higher risk of producing meaningless results than interpolation. |

Comments

- > When incorrect parameters are specified, the relevant compile-time or run-time error is generated. See [ACSPL+ Runtime Errors](#) for explanation of error 2044 and [ACSPL+ Compilation Errors](#) for explanations of errors 3113, 3384, 3413, 3414. The function is intended for use with arrays only; an error is generated if a scalar is given as a parameter.

Example

```

D-Buffer:
global real static X_Correction1(3)(10)
global real static X_Correction2(3)(10)
global real static X_Axis_Coordinates(10)
global real static Y_Axis_Coordinates(3)
global real static Z_Axis_Coordinates(3)

Buffer:
local int X_axis
local int Y_axis
local int Z_axis
int Zone

X_axis = 0      ! First moving axis
Y_axis = 1      ! Second moving axis
Z_axis = 2      ! third moving axis
Zone=0          ! Index of error mapping zone in use
                ! Error map zone definition for X axis
X_Axis_Coordinates(0)=-100; X_Axis_Coordinates(1)=-90;
X_Axis_Coordinates(2)=-80;
X_Axis_Coordinates(3)=-70; X_Axis_Coordinates(4)=-60;
X_Axis_Coordinates(5)=-50;
X_Axis_Coordinates(6)=-40; X_Axis_Coordinates(7)=-30;
X_Axis_Coordinates(8)=-20;
X_Axis_Coordinates(9)=-10
! Error map zone definition for Y axis
Y_Axis_Coordinates(0)=30; Y_Axis_Coordinates(1)=40;
Y_Axis_Coordinates(2)=50
! Error map zone definition for Z axis
Z_Axis_Coordinates(0)=30; Z_Axis_Coordinates(1)=32;
! Correction table for Error Mapping for X axis of the first Z height -
declared in D-Buffer
X_Correction1(0)(0)=0; X_Correction1(0)(1)=0.3;
X_Correction1(0)(2)=-0.4; X_Correction1(0)(3)=-0.7;
X_Correction1(0)(4)=-0.1; X_Correction1(0)(5)=0.22;
X_Correction1(0)(6)=0.55; X_Correction1(0)(7)=0.2;
X_Correction1(0)(8)=-0.3; X_Correction1(0)(9)=0
X_Correction1(1)(0)=0; X_Correction1(1)(1)=0.4;
X_Correction1(1)(2)=-0.4; X_Correction1(1)(3)=-0.5;
X_Correction1(1)(4)=-0.2; X_Correction1(1)(5)=0.24;
X_Correction1(1)(6)=0.3; X_Correction1(1)(7)=0.32;
X_Correction1(1)(8)=-0.13; X_Correction1(1)(9)=0
X_Correction1(2)(0)=0; X_Correction1(2)(1)=0.1;
X_Correction1(2)(2)=-0.2; X_Correction1(2)(3)=-0.4;
X_Correction1(2)(4)=0.1; X_Correction1(2)(5)=0.3;
X_Correction1(2)(6)=0.4; X_Correction1(2)(7)=0.33;
X_Correction1(2)(8)=-0.1; X_Correction1(2)(9)=0
! Correction table for Error Mapping for X axis of the second Z height -

```

```

declared in D-Buffer
X_Correction2(0)(0)=0; X_Correction2(0)(1)=0.23;
X_Correction2(0)(2)=-0.12; X_Correction2(0)(3)=-0.3;
X_Correction2(0)(4)=-0.2; X_Correction2(0)(5)=0.26;
X_Correction2(0)(6)=0.15; X_Correction2(0)(7)=0.02;
X_Correction2(0)(8)=-0.15; X_Correction2(0)(9)=0
X_Correction2(1)(0)=0; X_Correction2(1)(1)=0.24;
X_Correction2(1)(2)=-0.14; X_Correction2(1)(3)=-0.34;
X_Correction2(1)(4)=-0.2; X_Correction2(1)(5)=0.14;
X_Correction2(1)(6)=-0.23; X_Correction2(1)(7)=-0.32;
X_Correction2(1)(8)=-0.13; X_Correction2(1)(9)=0
X_Correction2(2)(0)=0; X_Correction2(2)(1)=-0.1;
X_Correction2(2)(2)=-0.32; X_Correction2(2)(3)=-0.4;
X_Correction2(2)(4)=0.15; X_Correction2(2)(5)=0.36;
X_Correction2(2)(6)=0.44; X_Correction2(2)(7)=0.23;
X_Correction2(2)(8)=-0.11; X_Correction2(2)(9)=0

! X axis error mapping function configuration
ERRORMAPN3D2/0 (X_axis, Y_axis, Z_axis), Zone, X_Axis_Coordinates, Y_
Axis_Coordinates, Z_Axis_Coordinates, X_Correction1, X_Correction2

! Enable Error Mapping
ERRORMAPON X_axis, Zone
STOP

```

4.11.11 ERRORMAPN3D3

Description

The **ERRORMAPN3D3** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the linearly (by default) interpolated error from the desired position so that the actual value will be closer to the desired value.

ERRORMAPN3D3 function receives indexes of three axes, zone index, 'axis0' command table, 'axis1' command table, 'axis2' command table, and three 2D correction tables (in correlation to the specified 'axis2' coordinates) for mechanical error compensation.

Syntax

```

ERRORMAPN3D3[command options] (axis0, axis1, axis2), zone, axis0_command,
axis1_command, axis2_command, correction_map0, correction_map1,
correction_map2[, reference_axis_or_analog_input0, referenced_axis_or_
analog_input1, referenced_axis_or_analog_input2]

```

Arguments

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| axis0_ command | The name of a real one-dimensional array that specifies 'axis0' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis1_ command | The name of a real one-dimensional array that specifies 'axis1' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis2_ command | The name of a real one-dimensional array that specifies 'axis2' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map0 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to axis 2's first specified coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to axis 2's second specified coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map2 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to axis 2's third specified coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |

| | |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis_or_analog_input1 | input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_axis_or_analog_input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter will be regarded as an analog input index. |
| /ak | Specifies that the second optional parameter will be regarded as an analog input index. |
| /al | Specifies that the third optional parameter will be regarded as an analog input index. |
| /e | Specifies that the extrapolation will be used for Z values that are beyond the original observation range. That is, we are assuming that existing trends will continue (by using the nearest correction table to estimate the correction). This method is subject to greater uncertainty and a higher risk of producing meaningless results than interpolation. |

Comments

- > When incorrect parameters are specified, the relevant compile-time or run-time error is generated. See [ACSPL+ Runtime Errors](#) for explanation of error 2044 and [ACSPL+ Compilation Errors](#) for explanations of errors 3113, 3384, 3413, 3414. The function is intended for use with arrays only; an error is generated if a scalar is given as a parameter.

Example

```
D-Buffer:
global real static X_Correction1(3)(10)
global real static X_Correction2(3)(10)
global real static X_Correction3(3)(10)
global real static X_Axis_Coordinates(10)
```

```

global real static Y_Axis_Coordinates(3)
global real static Z_Axis_Coordinates(3)

Buffer:
local int X_axis
local int Y_axis
local int Z_axis
int Zone

X_axis = 0      ! First moving axis
Y_axis = 1      ! Second moving axis
Z_axis = 2      ! third moving axis
Zone=0          ! Index of error mapping zone in use! Error map zone definition
for X axis
X_Axis_Coordinates(0)=-100; X_Axis_Coordinates(1)=-90; X_Axis_Coordinates
(2)=-80;
X_Axis_Coordinates(3)=-70; X_Axis_Coordinates(4)=-60; X_Axis_Coordinates
(5)=-50;
X_Axis_Coordinates(6)=-40; X_Axis_Coordinates(7)=-30; X_Axis_Coordinates
(8)=-20;
X_Axis_Coordinates(9)=-10
! Error map zone definition for Y axis
Y_Axis_Coordinates(0)=30; Y_Axis_Coordinates(1)=40; Y_Axis_Coordinates
(2)=50
! Error map zone definition for Z axis
Z_Axis_Coordinates(0)=30; Z_Axis_Coordinates(1)=32; Z_Axis_Coordinates
(1)=34;
! Correction table for Error Mapping for X axis of the first Z height -
declared in D-Buffer
X_Correction1(0)(0)=0; X_Correction1(0)(1)=0.3; X_Correction1(0)(2)=-0.4;
X_Correction1(0)(3)=-0.7;
X_Correction1(0)(4)=-0.1; X_Correction1(0)(5)=0.22; X_Correction1(0)
(6)=0.55; X_Correction1(0)(7)=0.2;
X_Correction1(0)(8)=-0.3; X_Correction1(0)(9)=0
X_Correction1(1)(0)=0; X_Correction1(1)(1)=0.4; X_Correction1(1)(2)=-0.4;
X_Correction1(1)(3)=-0.5;
X_Correction1(1)(4)=-0.2; X_Correction1(1)(5)=0.24; X_Correction1(1)
(6)=0.3; X_Correction1(1)(7)=0.32;
X_Correction1(1)(8)=-0.13; X_Correction1(1)(9)=0
X_Correction1(2)(0)=0; X_Correction1(2)(1)=0.1; X_Correction1(2)(2)=-0.2;
X_Correction1(2)(3)=-0.4;
X_Correction1(2)(4)=0.1; X_Correction1(2)(5)=0.3; X_Correction1(2)
(6)=0.4; X_Correction1(2)(7)=0.33;
X_Correction1(2)(8)=-0.1; X_Correction1(2)(9)=0
! Correction table for Error Mapping for X axis of the second Z height -
declared in D-Buffer
X_Correction2(0)(0)=0; X_Correction2(0)(1)=0.23; X_Correction2(0)(2)=-
0.12; X_Correction2(0)(3)=-0.3;
X_Correction2(0)(4)=-0.2; X_Correction2(0)(5)=0.26; X_Correction2(0)

```

```

(6)=0.15; X_Correction2(0)(7)=0.02;
X_Correction2(0)(8)=-0.15; X_Correction2(0)(9)=0
X_Correction2(1)(0)=0; X_Correction2(1)(1)=0.24; X_Correction2(1)(2)=-
0.14; X_Correction2(1)(3)=-0.34;
X_Correction2(1)(4)=-0.2; X_Correction2(1)(5)=0.14; X_Correction2(1)(6)=-
0.23; X_Correction2(1)(7)=-0.32;
X_Correction2(1)(8)=-0.13; X_Correction2(1)(9)=0
X_Correction2(2)(0)=0; X_Correction2(2)(1)=-0.1; X_Correction2(2)(2)=-
0.32; X_Correction2(2)(3)=-0.4;
X_Correction2(2)(4)=0.15; X_Correction2(2)(5)=0.36; X_Correction2(2)
(6)=0.44; X_Correction2(2)(7)=0.23;
X_Correction2(2)(8)=-0.11; X_Correction2(2)(9)=0
! Correction table for Error Mapping for X axis of the third Z height -
declared in D-Buffer
X_Correction3(0)(0)=0; X_Correction3(0)(1)=0.23; X_Correction3(0)(2)=-
0.12; X_Correction3(0)(3)=-0.3; X_Correction3(0)(4)=-0.2
X_Correction3(0)(5)=0.26; X_Correction3(0)(6)=0.15; X_Correction3(0)
(7)=0.02; X_Correction3(0)(8)=-0.15;
X_Correction3(0)(9)=0
X_Correction3(1)(0)=0; X_Correction3(1)(1)=0.24; X_Correction3(1)(2)=-
0.14; X_Correction3(1)(3)=-0.34;
X_Correction3(1)(4)=-0.2
X_Correction3(1)(5)=0.14; X_Correction3(1)(6)=-0.23; X_Correction3(1)
(7)=-0.32; X_Correction3(1)(8)=-0.13; X_Correction3(1)(9)=0
X_Correction3(2)(0)=0; X_Correction3(2)(1)=-0.1; X_Correction3(2)(2)=-
0.32; X_Correction3(2)(3)=-0.4; X_Correction3(2)(4)=0.15
X_Correction3(2)(5)=0.36; X_Correction3(2)(6)=0.44; X_Correction3(2)
(7)=0.23; X_Correction3(2)(8)=-0.11;
X_Correction3(2)(9)=0

! X axis error mapping function configuration
ERRORMAPN3D3/0 (X_axis, Y_axis, Z_axis), Zone, X_Axis_Coordinates, Y_
Axis_Coordinates, Z_Axis_Coordinates, X_Correction1, X_Correction2, X_
Correction3

! Enable Error Mapping
ERRORMAPON X_axis, Zone
STOP

```

4.11.12 ERRORMAP3D5

Description

The **ERRORMAP3D5** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the interpolated error from the desired position so that the actual value will be closer to the desired value. Interpolation is linear by default; other options are available.

The **ERRORMAP3D5** function receives indices of three axes, zone index, the base value of the 'axis0' command, a fixed defined interval of the 'axis0' command, the base value of the 'axis1' command, a

fixed defined interval of the 'axis1' command, the base value of the 'axis2' command, a fixed defined interval of the 'axis2' command, and five 2D correction tables (in correlation to the specified 'axis2' coordinates) for mechanical error compensation.

Syntax

```
ERRORMAP3D5[command options] (axis0, axis1, axis2), zone, base1, step1,
base2, step2, base3, step3, correction_map0, correction_map1, correction_
map2, correction_map3, correction_map4[, reference_axis_or_analog_input0,
referenced_axis_or_analog_input1, referenced_axis_or_analog_input2]
```

Arguments

| | |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis to which the mechanical error compensation will be applied, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| base0 | A real number representing the 'axis0' command that corresponds to the first point in correction table for mechanical error compensation. |
| step0 | A real number representing the fixed interval distance between the two adjacent 'axis0' commands. |
| base1 | A real number representing the 'axis1' command that corresponds to the first point in correction table for mechanical error compensation. |
| step1 | A real number representing the fixed interval distance between the two adjacent 'axis1' commands. |
| base2 | A real number representing the 'axis2' command that corresponds to the first point in correction table for mechanical error compensation. |
| step2 | A real number representing the fixed interval distance between the two adjacent 'axis2' commands. |

| | |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| correction_ map0 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '0' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '1' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map2 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '2' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map3 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '3' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map4 | The name of a real two-dimensional array that specifies correction table for mechanical error compensation in relation to axis 2 step '4' coordinate. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter to be treated as an analog input index. |

| | |
|-----|--------------------------------------------------------------------------------------|
| /ak | Specifies that the second optional parameter to be treated as an analog input index. |
| /al | Specifies that the third optional parameter to be treated as an analog input index. |

Error Conditions

The function detects the following error conditions.

- > **2044** - Index is out of range, when the defined array size is less than the defined number of array points.
- > **3113** - The step in the table is zero or negative, when the step argument is zero or negative.
- > **3384** - The specified suffix combination is invalid. Please see the documentation for more details.
- > **3413** - The supplied correction maps (2-dimensional arrays) should have the same dimensions. Please see the documentation for more details.
- > **3414** - The Specified referenced axes/analog inputs are invalid. Please see the documentation for more details.

Comments

If incorrect parameters are passed to the function the corresponding runtime error is generated. The function is intended to be used only with arrays; thus, an error is generated if a scalar is given as a parameter.

4.11.13 *ERRORMAPN3D5*

Description

The **ERRORMAPN3D5** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the interpolated error from the desired position so that the actual value will be closer to the desired value. Interpolation is linear by default; other options are available.

ERRORMAPN3D5 function receives indices of three axes, zone index, 'axis0' command table, 'axis1' command table, 'axis2' command table, and 5 2D correction tables correlated to the specified 'axis2' coordinates for mechanical error compensation.

Syntax

```
ERRORMAPN3D5[command options] (axis0, axis1, axis2), zone, axis0_command,
axis1_command, axis2_command, correction_map0, correction_map1,
correction_map2, correction_map3, correction_map4[, reference_axis_or_
analog_input0, referenced_axis_or_analog_input1, referenced_axis_or_
analog_input2]
```

Arguments

| | |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| axis0_ command | The name of a real one-dimensional array that specifies 'axis0' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis1_ command | The name of a real one-dimensional array that specifies 'axis1' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis2_ command | The name of a real one-dimensional array that specifies 'axis2' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map0 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the first specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the second specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map2 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the third specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map3 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the fourth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |

| | |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| correction_ map4 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the fifth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter will be regarded as an analog input index. |
| /ak | Specifies that the second optional parameter will be regarded as an analog input index. |
| /al | Specifies that the third optional parameter will be regarded as an analog input index. |

Error Conditions

The function detects the following error conditions.

- > **2044** - Index is out of range, when the defined array size is less than the defined number of array points.
- > **3384** - The specified suffix combination is invalid. Please see the documentation for more details.
- > **3413** - The supplied correction maps (2-dimensional arrays) should have the same dimensions. Please see the documentation for more details.
- > **3414** - The Specified referenced axes/analog inputs are invalid. Please see the documentation for more details.

Comments

If incorrect parameters are passed to the function the corresponding runtime error is generated. The function is intended to be used only with arrays; thus, an error is generated if a scalar is given as a parameter.

4.11.14 *ERRORMAPN3DA*

Description

The **ERRORMAPN3DA** function configures and activates 3D error correction for the mechanical error compensation of the 'axis0' command, 'axis1' command, and 'axis2' command for the specified zone, so that the compensated reference position will be calculated by adding the interpolated error from the desired position so that the actual value will be closer to the desired value. Interpolation is linear by default; other options are available.

ERRORMAPN3DA function receives indexes of three axes, zone index, 'axis0' command table, 'axis1' command table, 'axis2' command table, and 10 2D correction tables (correlated to the specified 'axis2' coordinates) for mechanical error compensation.

Syntax

```
ERRORMAPN3DA[command options] (axis0, axis1, axis2), zone, axis0_command,
axis1_command, axis2_command, correction_map0, correction_map1,
correction_map2, correction_map3, correction_map4, correction_map5,
correction_map6, correction_map7, correction_map8, correction_map9[,
reference_axis_or_analog_input0, referenced_axis_or_analog_input1,
referenced_axis_or_analog_input2]
```

Arguments

| | |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis0 | The index of the axis that the mechanical error compensation will be applied to, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis1 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| axis2 | The index of the second axis participating in 3D mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index of the mechanical error compensation, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. |
| axis0_command | The name of a real one-dimensional array that specifies 'axis0' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis1_command | The name of a real one-dimensional array that specifies 'axis1' command values used for correction table of mechanical error compensation. The |

| | |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| axis2_ command | The name of a real one-dimensional array that specifies 'axis2' command values used for correction table of mechanical error compensation. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map0 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the first specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map1 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the second specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map2 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the third specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map3 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the fourth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map4 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the fifth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map5 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the sixth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map6 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the seventh specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map7 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the eighth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map8 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the ninth specified |

| | |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| correction_ map9 | The name of a real two-dimensional array that specifies a correction table for mechanical error compensation in relation to the tenth specified coordinate of the Z-axis. The array type should be GLOBAL REAL STATIC (defined in D-Buffer). |
| referenced_ axis_or_analog_ input0 | [Optional] The index of the first axis, or the index of the first analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input1 | [Optional] The index of the second axis, or the index of the second analog input whose feedback will be used to calculate the mechanical error compensation. |
| referenced_ axis_or_analog_ input2 | [Optional] The index of the second axis, or the index of the third analog input whose feedback will be used to calculate the mechanical error compensation. |

Switches

| | |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| /0 | The mechanical error compensation will be applied to 'axis0' (default) |
| /1 | The mechanical error compensation will be applied to 'axis1' |
| /2 | The mechanical error compensation will be applied to 'axis2' |
| /a | Specified that the mechanical error compensation will be calculated be on the feedback from the axis specified by the optional parameter. |
| /aj | Specifies that the first optional parameter will be regarded as an analog input index. |
| /ak | Specifies that the second optional parameter will be regarded as an analog input index. |
| /al | Specifies that the third optional parameter will be regarded as an analog input index. |

Error Conditions

The function detects the following error conditions.

- > **2044** - Index is out of range, when the defined array size is less than the defined number of array points.
- > **3384** - The specified suffix combination is invalid. Please see the documentation for more details.
- > **3413** - The supplied correction maps (2-dimensional arrays) should have the same dimensions. Please see the documentation for more details.

- > **3414** - The Specified referenced axes/analog inputs are invalid. Please see the documentation for more details.

Comments

If incorrect parameters are passed to the function the corresponding runtime error is generated. The function is intended to be used only with arrays; thus, an error is generated if a scalar is given as a parameter.

4.11.15 *ERRORMAPOFF*

Description

ERRORMAPOFF function receives axis index and zone index. The **ERRORMAPOFF** function deactivates error mapping correction for the mechanical error compensation for the specified zone.

Syntax

```
ERRORMAPOFF axis, zone
```

Arguments

| | |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. If '-1' is specified, all zones of specified axis will be affected. |

4.11.16 *ERRORMAPON*

ERRORMAPON function receives axis index and zone index. The **ERRORMAPON** function activates error correction for the mechanical error compensation for the specified zone.

Syntax

```
ERRORMAPON axis, zone
```

Arguments

| | |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. If '-1' is specified, all zones of specified axis will be affected. |

4.11.17 *#ERRORMAPREP*

Description

#ERRORMAPREP function generates a report of all activated zones of error mapping for all axes in the system.

Syntax

```
#ERRORMAPREP
```

4.11.18 ERRORUNMAP

Description

ERRORUNMAP function receives axis index and zone index. The **ERRORUNMAP** function deactivates error correction for the mechanical error compensation for the specified zone.

Syntax

```
ERRORUNMAP axis, zone
```

Arguments

| | |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| axis | The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1. |
| zone | The zone index, valid numbers are: 0, 1, 2, ... up to the maximum number of zones minus 1. If '-1' is specified, all zones of specified axis will be affected. |

5. ACSPL+ Standard Structures

5.1 LCI Standard Structure

The LCI structure is supported in versions 3.10 and higher.

5.1.1 LCI Functions

5.1.1.1 PowerPWMOut

Description

This function initializes Pulse Modulation mode .

Syntax

```
PowerPWMOut (Mode, Freq, Width, DutyCycle, [MinValue, MaxValue,
MinVelocity, MaxVelocity])
```

Arguments

| | |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mode | <p>Modulation mode:</p> <ul style="list-style-type: none"> 0 – Fixed parameters mode 1 - Fixed Frequency 2 - Fixed Pulse Width 3 - Fixed duty cycle <p>Modes 0 – 3 are incompatible modes, i.e. setting any of these modes automatically disables the previously set mode.</p> |
| Freq | <p>Pulse modulation frequency in Hz, range from 0.035Hz to 1MHz.</p> <p>In mode = 1, the frequency is fixed and is not changed during the process. In mode = 2 or 3, the function only sets the initial frequency.</p> |
| Width | <p>Pulse Width in milliseconds, range for 6.67 nsec to 28.60 sec.</p> |
| DutyCycle | <p>Duty cycle in percentage, range from 0% to 100%.</p> <p>When mode = 0, DutyCycle value is not applicable</p> <p>When mode = 1, the function only sets the initial duty cycle.</p> <p>When mode = 2, the argument has no effect, as duty cycle is automatically calculated by the unit as function of frequency.</p> <p>When mode = 3, the duty cycle is fixed and is not changed during the process</p> |
| MinValue | <p>[Optional] Minimum value of volatile parameter (depends on mode) , if calculated vector velocity is less than or equal to MinVel</p> |

| | |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MaxValue | [Optional] Maximum value of volatile parameter (depends on mode) , if calculated vector velocity is greater than or equal toMaxVel |
| MinVel | [Optional] Minimum velocity for parameter calculation. Below the minimum value, the parameter will be limited by MinValue or default, if MinValue is not specified By default, minimum velocity is zero.` |
| MaxVel | [Optional] Maximum velocity for parameter calculation. Above the maximum value, the parameter will be limited by MaxValue or default, if MaxValue is not specified. By default, maximum velocity is defined by XVEL parameter |

Example*5.1.1.2 PowerAnalogOut***Description**

The function defines the range of Analog Output value depending on the actual velocity

Syntax

```
PowerAnalogOut (AnalogOutInd, [MinValue, MaxValue, MinVelocity,
MaxVelocity] )
```

Arguments

| | |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AnalogOutInd | Analog output index (0 or 1) |
| MinValue | [Optional] Minimum value of volatile parameter if calculated vector velocity equals or below MinVel, default is 0. |
| MaxValue | [Optional] Maximum of volatile parameter if calculated vector velocity equals or above MaxVel, default is 100 (%) |
| MinVel | [Optional] Minimum velocity for parameter calculation. Below the minimum value, parameter will be limited by MinValue or default, if MinValue is not specified By default, Minimum velocity is zero. |

| | |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MaxVel | [Optional] Maximum velocity for parameter calculation. Above the maximum value, parameter will be limited by MaxValue or default, if MaxValue is not specified. By default, Maximum velocity is defined by XVEL parameter |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

```

!D-buffer declaration
LCI lc

!Program buffer
lc.MotionAxes = AxListAsMask(X,Y)
!Define laser power control by analog output
! Output value is 0 Volt for velocity < 10 and 10 Volt for velocity > VEL
(X)*2
! Use Analog output - 1
lc.PowerAnalogOut(1, 0, 10, 10, VEL(X)*2)

lc.LaserEnable()
```

5.1.1.3 PowerDigitalOut

Description

The function defines the range of Digital Output value depending on the actual velocity.

SYNTAX

```

PowerDigitalOut (SynchPulseWidth, [MinValue, MaxValue, MinVelocity,
MaxVelocity] )
```

Arguments

| | |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SynchPulseWidth | The width of the synchronization pulse in msec. The pulse fires every time when all Digital Outputs are updated and user can read the Power value |
| MinValue | [Optional] Minimum value of volatile parameter (depends on mode) , if calculated vector velocity equals or below MinVel. Default is 0 |
| MaxValue | [Optional] Maximum of volatile parameter (depends on mode) , if calculated vector velocity equals or above MaxVel. Default is 255 |
| MinVel | [Optional] Minimum velocity for parameter calculation. Below the minimum value, parameter will be limited by MinValue or by the default value if MinValue is not specified. By default, minimum velocity is zero. |

| | |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MaxVel | [Optional] Maximum velocity for parameter calculation. Above the maximum value, the parameter will be limited by MaxValue or by the default value if MaxValue is not specified. By default, maximum velocity is defined by the XVEL parameter. |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

5.1.1.4 FixedDistPulse

Description

The function initializes the fixed distance pulse firing mode. This mode is useful if a laser beam should be activated at specified fixed intervals between activations along an actual motion trajectory. The **AxesUsed** field defines which axes are used for multi-axis trajectory generation.

Syntax

```
int FixedDistPulse (PulseWidth, Interval[, StartPos, EndPos, InitialPos,
MinValue, MaxValue, MinVelocity, MaxVelocity])
```

Return Type

Occupied channel index as integer value

Arguments

| | |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PulseWidth | Pulse Width in milliseconds (40ns – 161ms) |
| Interval | Specifies an interval in user units between pulses along the multi-axis motion trajectory. |
| StartPos | [Optional] Specifies a start position in user units along the multi-axis motion trajectory. The pulses start generating after reaching this position. The value is 0 by default. |
| EndPos | [Optional] Specifies a last position in user units along the multi-axis motion trajectory. The pulses stop generating after reaching this position. If the parameter is omitted, then there is no limitation. |
| InitialPos | [Optional] Specifies an initial position offset relative to the user origin. By default InitialPos is zero. For a multi-axes-trajectory, the InitialPos should be omitted or set to zero. |
| MinValue | [Optional] Minimum value of volatile parameter (depends on mode), if calculated vector velocity is greater than or equal to MinVel |
| MaxValue | [Optional] Maximum of volatile parameter (depends on mode), if calculated vector velocity less than or equal to MaxVel |
| MinVelocity | [Optional] Minimum velocity for parameter calculation. Below the minimum value, parameter will be limited by MinValue or default, if MinValue is not specified |

| | |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | By default, minimum velocity is zero.` |
| MaxVelocity | [Optional] Maximum velocity for parameter calculation. Above the maximum value, parameter will be limited by MaxValue or default, if MaxValue is not specified. By default, maximum velocity is defined by XVEL parameter |

Comments

The ExtraPulses and ExtraPeriod fields define the extra pulses generation. If at least one of parameters is 0, no extra pulses are generated.

The PiercePulsesNum and PiercePulseWidth fields define the pierce pulse generation. If at least one of parameters is 0, no pierce pulses are generated.

The MotionAxes axes mask field is used for trajectory calculation.

The PulseResolution field is used for operation pulse resolution. If the default PulseResolution is 0, the pulse resolution should be calculated according to the **XVEL** parameter and Maximum LCI Frequency.

*5.1.1.5 DistanceArrPulse***Description**

The function initializes either array-based pulse firing mode.

Syntax

```
int DistanceArrPulse (arrPos, arWidth, ArraySize)
```

Return Type

Occupied channel index as integer value

Arguments

| | |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| arPos | Array of points. Each element in the array defines the point where a pulse should be fired. |
| arWidth | Pulse Width value or Pulse Width array in milliseconds. Each width in the array corresponds to a point in the Position array. The size of array should be equals to size of Position array. If the parameter is a real value, pulse width is applied for all elements in the Position array. |
| ArraySize | The size of Points and Width arrays. |

Comments

The ExtraPulses and ExtraPeriod fields define the extra pulses generation. If at least one of parameters is 0, no extra pulses are generated.

The MotionAxes axes mask field is used for trajectory calculation.

The PulseResolution field is used for operation pulse resolution. In case of the default PulseResolution is 0, the pulse resolution should be calculated according to the Comments

The ExtraPulses and ExtraPeriod fields define the extra pulses generating. If at least one of parameters is 0, no extra pulses are generated.

The MotionAxes axes mask field is used for trajectory calculation.

The PulseResolution field is used for operation pulse resolution. In case of the default PulseResolution is 0, the pulse resolution should calculate according to **XVEL** parameter and Maximum LCI Frequency.

Command requires "Array and Segment Based Modes" LCI ordering option to use.

Example

5.1.1.6 CoordinateArrPulse

Description

The function initializes either array-based pulse firing mode based on multiple axis position.

Syntax

```
int CoordinateArrPulse (ArraySize, WidthArr[, XPosArr, YPosArr , ZPosArr,
APosArr, BPosArr, CPosArr])
```

Return Type

Occupied first channel index as integer value

Arguments

| | |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ArraySize | The size of Points and Width arrays. |
| WidthArr | Pulse Width value or Pulse Width array in milliseconds. Each width in the array corresponds to a point in the Positions arrays. If the parameter is a real value, pulse width is applied for all items of Positions arrays. |
| XPosArr | (optional) Array of X axis positions. Each element of the array defines the point where a pulse should be fired. |
| YPosArr | (optional) Array of Y axis positions. Each element of the array defines the point where a pulse should be fired. |
| ZPosArr | (optional)Array of Z axis positions. Each element of the array defines the point where a pulse should be fired. |
| APosArr | (optional)Array of A axis positions.Each element of the array defines the point where a pulse should be fired. |
| BPosArr | (optional)Array of B axis positions. Each element of the array defines the point where a pulse should be fired. |

| | |
|---------|-----------------------------------------------------------------------------------------------------------------|
| CPosArr | (optional)Array of C axis positions. Each element of the array defines the point where a pulse should be fired. |
|---------|-----------------------------------------------------------------------------------------------------------------|

Comments

Function occupies several channels, depending on the MotionAxes field.

If the parameter is 0, no extra pulses are generated.

The MotionAxes axes mask field is used for trajectory calculation.

The PulseResolution field is used for operation pulse resolution. In case of the default PulseResolution is 0, the pulse resolution should calculate according to **XVEL** parameter and Maximum LCI Frequency.

Command requires "Array and Segment Based Modes" LCI ordering option to use.

Example

5.1.1.7 Tickle

Description

The function initializes the Tickle mode. In this mode the laser control unit generates a signal at constant frequency and with constant width. Usually this mode is used for those types of lasers that require gas ionization during the time period when laser processing is off. By setting this mode the laser will respond faster and more predictably when laser processing resumes. Once this mode is initialized, the laser control unit constantly generates pulses without regard to any other operational modes.

Syntax

```
Tickle (Freq, Width)
```

Return Type

None

Arguments

| | |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Freq | <p>Tickle frequency in Hz.</p> <p>Minimum frequency is 1149Hz, maximum frequency is 100,000Hz</p> <p>Not all frequencies can be configured, so the function automatically rounds the specified frequency to the nearest supported value.</p> |
| Width | <p>Pulse Width in milliseconds</p> <p>Minimal width is 0.000107msec (107nsec), maximal width is 0.0273msec (27.3µsec)</p> <p>Not all width values are supported, so the function automatically rounds the specified width to the nearest supported value.</p> |

5.1.1.8 LaserEnable

Description

The function enables laser pulse generation.

Syntax

```
LaserEnable ()
```

5.1.1.9 LaserDisable

Description

The function disables laser pulse generation.

Syntax

```
LaserDisable ()
```

5.1.1.10 DistanceArrGate

Description

The function adds the gating mode to the system. In this mode laser is to be switched on or off at a predefined position. The position is defined by distance position.

Syntax

```
int DistanceArrGate (arPos, arStates, Size)
```

Return Type

None

Arguments

| | |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| arPos | Array of points. Each element of the array defines the point where a state signal should be changed. |
| arStates | Array of states. Each element of the array defines the state value in the corresponding point from arPos array. The array size should be equal to points array (arPos) |
| Size | Size of the two arrays used for gating definition. |

Comments

The MotionAxes axes mask field is used for trajectory calculation. The PulseResolution field is used for operation pulse resolution. If the default PulseResolution is 0, the pulse resolution should be calculated according to the **XVEL** parameter and Maximum LCI Frequency.

Command requires "Array and Segment Based Modes" LCI ordering option to use.

5.1.1.11 CoordinateArrGate

Description

The function adds the gating mode to the system. In this mode laser is to be switched on or off at a predefined position. The position defined by axes coordinates.

Syntax

```
int CoordinateArrGate (PointsNum, StatesArr[, XPosArr, YPosArr, ZPosArr,
APosArr, BPosArr, CPosArr ])
```

Return Type

First occupied channel index as integer value

Arguments

| | |
|-----------|-----------------------------------------------------------------------------------------------------------------|
| PointsNum | The actual points number |
| StatesArr | States array of integers |
| XPosArr | (optional)Array of X axis positions. Each element of the array defines the point where a pulse should be fired. |
| YPosArr | (optional)Array of Y axis positions.Each element of the array defines the point where a pulse should be fired. |
| ZPosArr | (optional)Array of Z axis positions.Each element of the array defines the point where a pulse should be fired. |
| APosArr | (optional)Array of A axis positions.Each element of the array defines the point where a pulse should be fired. |
| BPosArr | (optional)Array of B axis positions. Each element of the array defines the point where a pulse should be fired. |
| CPosArr | (optional)Array of C axis positions.Each element of the array defines the point where a pulse should be fired. |

Comments

The trajectory calculation is built according to defined Position arrays.

The PulseResolution field is used for operation pulse resolution. If PulseResolution is 0, the pulse resolution should be calculated according to the **XVEL** parameter and Maximum LCI Frequency.

This command is available when the "Array and Segment Based Modes" LCI ordering option is purchased.

5.1.1.12 AddZone

Description

Add the laser activation zone. If the zone is defined, the laser can be activated only in the specified zone.

Syntax

```
int AddZone (MinPos, MaxPos[, MotionAxes] )
```

Return Type

Occupied channel index as integer value

Arguments

| | |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MinPos | Specifies a minimum position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position |
| MaxPos | Specifies a maximum position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position. |
| MotionAxes | [Optional] Mask that defines the axes, which are used for generating pulses along the multi-axis motion trajectory. If parameter is 0 or omitted, the default axes mask is taken from MotionAxes. |

Example

5.1.1.13 SetZone

Description

Change the laser activation zone for specified channel. If zone is defined, the laser can be activated only in the specified zone

Syntax

```
int SetZone (Channel, MinPos, MaxPos)
```

Return Type

None

Arguments

| | |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Channel | Channel or Operation ID where the zone was defined |
| MinPos | Specifies a minimal position of the operation zone along the motion trajectory. The laser pulses start outputting after reaching this position |
| MaxPos | Specifies a maximal position of the operation zone along the motion trajectory. The laser pulses stop outputting after reaching this position. |

Example

5.1.1.14 *SetCondition*

Description

Define laser activation condition in one of the 3 condition registers by a bitwise mask.

Syntax

```
lc.SetCondition (ConditionReg, SetBitMask [,ClearBitMask])
```

Return Type

None

Arguments

| | |
|--------------|--------------------------------------------|
| ConditionReg | Register index (0,1,2) |
| SetBitMask | Bitwise condition mask to set |
| ClearBitMask | (optional) Bitwise condition mask to clear |

Comments

Table 4-19. Condition Mask for Register 0

| Bit # | Signal |
|-------|--------------------|
| 0 | PWM |
| 1 | Tickle |
| 2-17 | For internal usage |
| 18 | In Range 0 |
| 19 | In Range 1 |
| 20 | In Range 2 |
| 21 | In Range 3 |
| 22 | In Range 4 |
| 23 | In Range 5 |
| 24 | In Range 6 |
| 25 | In Range 7 |

Table 4-20. Condition Mask for Register 1

| Bit # | Signal |
|-------|--------------------|
| 0 | PEG Pulse 0 |
| 1 | PEG Pulse 1 |
| 2 | PEG Pulse 2 |
| 3 | PEG Pulse 3 |
| 4 | PEG Pulse 4 |
| 5 | PEG Pulse 5 |
| 6 | PEG Active 0 |
| 7 | PEG Active 1 |
| 8 | PEG Active 2 |
| 9 | PEG Active 3 |
| 10 | PEG Active 4 |
| 11 | PEG Active 5 |
| 12 | PEG State of PEG 0 |
| 13 | PEG State of PEG 1 |
| 14 | PEG State of PEG 2 |
| 15 | PEG State of PEG 3 |
| 16 | PEG State of PEG 4 |
| 17 | PEG State of PEG 5 |

Table 4-21. Condition Mask for Register 2

| Bit # | Signal |
|-------|-------------------------|
| 0 | General Purpose Input 0 |
| 1 | General Purpose Input 1 |
| 2 | General Purpose Input 2 |

| Bit # | |
|-------|--------------------------|
| 3 | General Purpose Input 3 |
| 4 | General Purpose Input 4 |
| 5 | General Purpose Input 5 |
| 6 | General Purpose Input 6 |
| 7 | General Purpose Input 7 |
| 8 | General Purpose Output 0 |
| 9 | General Purpose Output 1 |
| 10 | General Purpose Output 2 |
| 11 | General Purpose Output 3 |
| 12 | General Purpose Output 4 |
| 13 | General Purpose Output 5 |
| 14 | General Purpose Output 6 |
| 15 | General Purpose Output 7 |

Example

```
!Set bit 0 and clear bit 1 in register 1
SetCondition (1, 0x1 , 0x2)
```

5.1.1.15 GetCondition

Description

Return the current laser activation condition mask from the specified register.

Syntax

```
INT GetCondition (ConditionReg)
```

Return Type

Condition mask. See tables in [SetCondition](#) description.

Arguments

| | |
|--------------|------------------------|
| ConditionReg | Register index (0,1,2) |
|--------------|------------------------|

Comments

Example

5.1.1.16 SegmentGate

Description

Activate automatic gating for segment motion.

Syntax

```
int SegmentGate ([InitialState])
```

Arguments

| | |
|--------------|----------------------------------------------------------------------------------------------------------------------------------|
| InitialState | (optional)The gating initial state: 1 or 0 (default). The gating state is set to requested value immediately after function call |
|--------------|----------------------------------------------------------------------------------------------------------------------------------|

Return Type

Channel ID allocated for this operation. Return -1 if failed.

5.1.1.17 SegmentPulse

Description

Activate automatic pulse firing for segment motion.

Syntax

```
int SegmentPulse (Mode, PulseWidth)
```

Return Type

Channel ID allocated for this operation. Return -1 if failed.

Arguments

| | |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mode | This parameter defines the type and occurrence of the synchronization signal: 2 – Pulse on demand. The LINE/ARC1/ARC2 special switch (/p) defines a segment at the beginning of which the pulse occurs. 3 – Pulse is generated automatically at the beginning of every segment (starting from the second wegment) |
| PulseWidth | Pulse width (msec) |

5.1.1.18 SetExtClockSync

Description

Enable or Disable External Laser Clock synchronization.

Syntax

```
SetExtClockSync (Enable [,Polarity, Delay] )
```

| | |
|----------|----------------------------------------------------------------------------------------------------------|
| Enable | Enable or Disable External Laser Clock synchronization mode (1 – Enable, 0 - Disable) |
| Polarity | Optional 1 or 0 1 – Synchronization on rising edge(default) 0 – Synchronization on falling edge |
| Delay | Optional Delay relative to External Laser Clock edge in μ sec. The default is 0. |

5.1.1.19 PowerPWMBurst

Description

The function initializes pulse modulation mode with specified number of pulses

Syntax

```
PowerPWMBurst (Freq, Width, PulseNum, [SynchFlag])
```

Arguments

| | |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Freq | Pulse modulation frequency in Hz, range from 0.035Hz to 1MHz. |
| Width | Pulse Width in milliseconds. Range from 6.67nsec to 28.60sec |
| PulseNum | The number of outgoing PWM pulses |
| SynchFlag | (Optional). If parameter is non-zero, the function works synchronously i.e., waits till the pulse sequence finished, otherwise initializes the pulse sequence without waiting. The default is 1 (works synchronously) |

Comments

The **LaserEnable** function should be called ahead beforehand to enable the sequence of pulses. If the function works asynchronously, the program should check the PWMBurstReady field to ensure that the pulse sequence is finished.

Example

```
lc.LaserEnable ()

!Initialize the 10 pulses with frequency 1 kHz
lc.PowerPWMBurst(1000, 0.5, 10, 0)
```

```
!wait till pulse sequence finished
till lc.PowerBurstReady = 1
```

5.1.1.20 SetSafetyMasks

Description

Enable or disable Safety and Fault inputs.

Syntax

```
SetSafetyMasks (SafetyInput, FaultInput )
```

Arguments

| | |
|-------------|-------------------------------------------------------------------------|
| SafetyInput | The 1 value masks the Safety Input and the system ignores Safety errors |
| FaultInput | The 1 value masks the Fault Input and the system ignores the Faults |

5.1.1.21 Stop

Description

Cancel laser mode or state for specified channel.

Syntax

```
Stop (Channel)
```

Return Type

None

Arguments

| | |
|---------|--------------------------------------------------------------------------------|
| Channel | Channel or Operation ID. If omitted or negative, cancel all active operations. |
|---------|--------------------------------------------------------------------------------|

5.1.1.22 SetMechPlatformAxes

Description

Define the axes configuration for the mechanical platform.

Syntax

```
SetMechPlatformAxes (X_Ax, Y_Ax, Z_Ax , A_Ax, B_Ax, C_Ax)
```

Return Type

None

Arguments

| | |
|------|---------------------------------------------------------|
| X_Ax | Axis index corresponds to X axis of mechanical platform |
| Y_Ax | Axis index corresponds to Y axis of mechanical platform |
| Z_Ax | Axis index corresponds to Z axis of mechanical platform |
| A_Ax | Axis index corresponds to A axis of mechanical platform |
| B_Ax | Axis index corresponds to B axis of mechanical platform |
| C_Ax | Axis index corresponds to C axis of mechanical platform |

Comments

If parameter omitted or -1, axis is not available in the current mechanical platform configuration.

5.1.1.23 SetMotionAxes

Description

Define the axes used for subsequent laser operations.

Syntax

```
SetMotionAxes (Axes_list)
```

Arguments

| | |
|-----------|-----------------------------------------------------|
| Axes_list | Single axis or axis group, valid numbers are: 0...5 |
|-----------|-----------------------------------------------------|

Return Value

None

Comments

The function defines the mechanical platform to use. Valid values for axis designation are as follows:

| Axis | Code |
|------|------|
| X | 0 |
| Y | 1 |
| Z | 2 |
| A | 3 |
| B | 4 |
| C | 5 |

Example

```
!Set platform axes X and Y for subsequent operations.
lcUnit.SetMotionAxes (0,1)
```

5.1.1.24 SetSystemDelay

Description

Change the internal pulse generation delay.

Syntax

```
SetSystemDelay (Channel, TimeDelay[,Relative] )
```

Arguments

| | |
|-----------|--------------------------------------------------------------------------------------------------------------------------------|
| Channel | Channel ID |
| TimeDelay | A new system delay in μsec |
| Relative | If 1, the TimeDelay value is added to the current delay, otherwise the TimeDelay value is used as a new value for system delay |

Return Value

None

5.1.1.25 GetSystemDelay

Description

Return the current system delay in μsec for the specified channel.

Syntax

```
GetSystemDelay (Channel)
```

Arguments

| | |
|---------|------------------------------------------------------------------------------|
| Channel | Channel ID. If omitted or negative cancel defined operation for all channels |
|---------|------------------------------------------------------------------------------|

Return Value

None

5.1.1.26 SetConfigOut

Description

Configure the Digital Output signal

Syntax

```
SetConfigOut (OutInd, Channel, Code)
```

Arguments

| | |
|---------|----------------------------------------------------------------------|
| OutInd | Index of Configurable Output (0..7) or 10 for LPC output |
| Channel | Channel index (0..7) whose signal routed to output |
| Code | Enumerator defines the signal routed to the output. See table below. |

Return Value

None

Comments

| Code | Signal Description |
|------|--------------------|
| 0 | Cancel Routing |
| 2 | P/D Pulse |
| 3 | P/D Direction |
| 4 | A signal of AqB |
| 5 | B signal of AqB |
| 6 | InRange |
| 7 | PEG Pulse |
| 8 | PEG State |
| 9 | PEG Active |

If output index equals 10 (LPC output), channel parameter is irrelevant and the code parameter can accept the following values:

| Code | Signal Description |
|------|-----------------------------------------------------------------------|
| 0 | No signal via LPC |
| 1 | Output PWM via LPC |
| 2 | LPC output use as synchronization pulse signal for 8 bit digital port |
| 3 | Reserved |

Example

```
! Output 2 is configured as the PEG State of channel 1:
lc.SetConfigOut(2, 1, 8)
```

5.1.1.27 AssignChannels

Description

Dedicate channel for a specific operation.

Syntax

```
AssignChannels (OperationsArray)
```

Arguments

| | |
|-----------------|-----------------------------------------------------------------------------------------------------|
| OperationsArray | Operation enumeration array, each member contains the operation type dedicated for specific channel |
|-----------------|-----------------------------------------------------------------------------------------------------|

Return Value

None

Comments

Operation Enumeration

| | |
|---------|------------------------------------------------------------------------------------------------|
| 100 | Fixed Distance Pulse |
| 101 | Distance Array Pulse |
| 102 | Coordinate Array pulse |
| 103 | Distance Array Gating |
| 104 | Segment-based |
| 105 | Coordinate Array Gating |
| 0x10000 | Zone (In Range). InRange code can be combined using an OR operator with another operation code |

Example

5.1.1.28 SetCustomPosCalc

Description

Define the user function intended for position calculation. The firmware calls this function every controller cycle. The function returns the calculated value as a real variable.

Syntax

```
SetCustomPosCalc (Channel, FuncRef)
```

Arguments

| | |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Channel | Channel or Operation ID |
| FuncRef | Reference to the function, which calculates the new position value. The function must match this pattern: <code>real fastcall FuncName()</code> |

Return Value

None

5.1.1.29 *SetCustomVelCalc*

Description

Define the user function intended for custom velocity-based calculation. The firmware calls this function every controller cycle. The function returns the calculated value as a real variable.

Syntax

```
SetCustomVelCalc (Operation, FuncRef)
```

Arguments

| | |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| Operation | Operation ID (Modulation, Analog Power etc.) |
| FuncRef | Reference to the function, which calculates the new value. The function must match this pattern: <code>real fastcall FuncName()</code> |

5.1.1.30 *SetCustomVelVar*

Description

Define the user custom position calculation variable. Every controller cycle the firmware takes the variable as a new calculated value.

Syntax

```
SetCustomVelVar (Operation, VarRef)
```

Arguments

| | |
|-----------|----------------------------------------------|
| Operation | Operation ID (Modulation, Analog Power etc.) |
| VarRef | Reference to the global ACSPL+ variable |

Return Value

None

5.1.2 LCI Structure Fields

| Field Name | Type | Accessibility | Description |
|----------------------------|------|---------------|-------------------------------------------------------------------------------------|
| MotionAxes | int | R/W | Default axes mask used by LCI functions |
| PosResolution | real | R/W | Default Pulse Resolution used by LCI functions |
| Internal PosResolution (8) | real | R/W | Contains the actual Pulse Resolution used by LCI functions, one element per channel |
| PWMDutyCycle | real | R | Current laser duty cycle as percent |
| PWMFrequency | real | R | Current laser frequency in Hz |
| PWMPulseWidth | real | R | PWM pulse width in ms |
| TickleFrequency | real | R | Tickle frequency in Hz |
| TicklePulseWidth | real | R | |
| PWMActive | in | T | Boolean, 1 if modulation mode is active |
| TickleActive | int | R | Boolean, 1 if Tickle mode is active |
| InRange | int | R | Boolean 1/0 |
| LaserEnabled | int | R | Boolean 1/0 |
| OperationMode(8) | int | R | Operation mode specified for channel |
| Positions(8) | int | R | FPOS for specific channel |
| UserPos | real | R | Current Channel Position in user units |
| MultiAxWinSize | real | R/W | Comparison window width for coordinate-based operations |

| Field Name | Type | Accessibility | Description |
|-------------------|------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ExtraPulsesQty | int | R/W | The number of additional pulses generated with ExtraPulsesPeriod after each pulse at each firing position. The parameter is applicable for pulses generated by the following functions: FixedDistPulse, DistanceArrPulse, CoordinateArrPulse. |
| ExtraPulsesPeriod | real | R/W | The period in milliseconds for ExtraPulses additional pulses to be generated after each pulse at each firing position. The parameter is applicable for pulses generated by the following functions: FixedDistPulse, DistanceArrPulse, CoordinateArrPulse |
| PiercePulsesNum | int | R/W | Number of pierce pulses at the beginning |
| PiercePulsesWidth | real | R/W | ms |
| GateOnDelay | real | R/W | Stat on delay in ms in gating mode |
| GateOffDelay | real | R/W | State off delay in ms in gating mode |
| PulseDelay | real | R | Laser firing pulse delay in Distance Array Pulse modes |
| PowerAOutVal | real | R | Analog Output value in % |
| PowerDigOutVal | int | R | Digital Output value |

5.1.2.1 MotionAxes

Description

MotionAxes is a R/W integer field and contains the default axes mask used by LCI functions.

Syntax

```
lcUnit.MotionAxes = AxListAsMask(X,
```

5.1.2.2 PosResolution

Description

PosResolution is a R/W real field and contains the default Pulse Resolution used by LCI functions. If the default Pulse Resolution is 0, the pulse resolution should be calculated according to **XVEL** parameter and Maximum LCI Frequency.

5.1.2.3 InternalPosResolution

Description

InternalPosResolution is a Read-only real array with one element for each LCI channel and containing the actual Pulse Resolution used by LCI functions.

5.1.2.4 PWMDutyCycle

Description

PWMDutyCycle is a Read-Only real field. It presents the current laser duty cycle (%)

Syntax

```
DISP lcUnit.PWMDutyCycle
```

5.1.2.5 PWMFrequency

Description

PWMFrequency is a Read-Only real field. It presents the current laser frequency (Hz)

Syntax

```
DISP lcUnit.PWMFrequency
```

5.1.2.6 PWMPulseWidth

Description

PWMPulseWidth is a Read-Only real field. It presents the current PWM pulse width (ms)

Syntax

```
DISP lcUnit.PWMPulseWidth
```

5.1.2.7 TickleFrequency

Description

TickleFrequency is a Read-Only real field. It presents the current Tickle frequency (Hz)

Syntax

```
DISP lcUnit.TickleFrequency
```

5.1.2.8 TicklePulseWidth

Description

TicklePulseWidth is a Read-Only real field. It returns the current Tickle pulse width in ms.

Syntax

```
DISP lcUnit.TicklePulseWidth
```

5.1.2.9 PWMActive

Description

PWMActive is a Read-Only integer field. It returns a Boolean value: 1 if modulation mode is switched on, otherwise 0.

5.1.2.10 TickleActive

Description

TickleActive is a Read-Only integer field. It returns a Boolean value: 1 if Tickle mode is switched on, otherwise 0.

5.1.2.11 InRange

Description

InRange is a Read-Only integer field. It returns a Boolean value: 1 if the unit is inside of defined range, otherwise 0.

5.1.2.12 LaserEnabled

Description

Laser Enabled is a Read-Only integer field. It returns a Boolean value: 1 if the laser is enabled, otherwise 0.

5.1.2.13 OperationMode

Description

OperationMode is an integer array with one element for each LCI channel. It returns the operation mode defined for a specific channel.

5.1.2.14 Positions

Description

Positions is an integer array with one element for each LCI channel. It returns the pulse counter value for a specific channel.

5.1.2.15 UserPos

Description

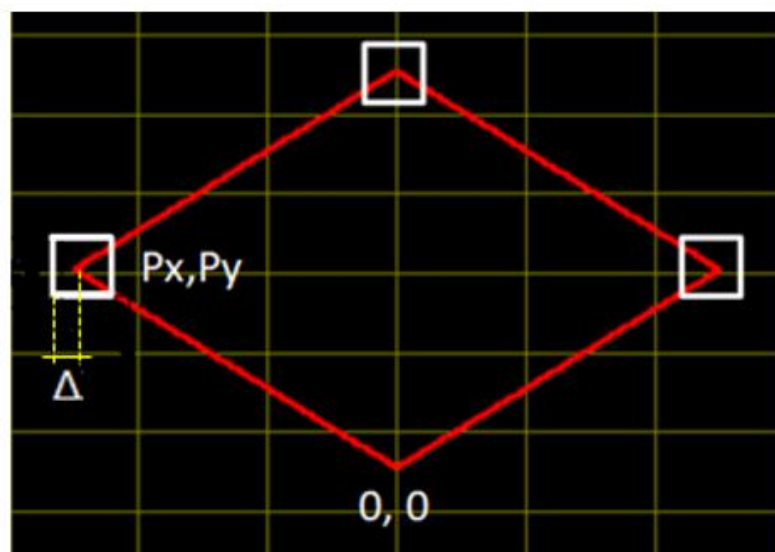
UserPos is an array of real, with one element for each LCI channel. Each element returns the current channel position in user units. UserPos is calculated as follows:

$$\text{UserPos} = \text{Positions} * \text{InternalPosResolution}$$

5.1.2.16 MultiAxWinSize

Description

MultiAxWinSize returns a real value defining the window size in coordinate based mode. The value defines the range (Δ) inside which the pulse and gate signals should be fired. The field is applicable for the **CoordinateArrPulse** and **CoordinateArrGate** functions. The parameter is in user units. The default is 1 resolution count. Δ indicates the window in the following diagram.



5.1.2.17 ExtraPulsesQty

Description

ExtraPulsesQty returns the number of additional pulses generated with ExtraPulsesPeriod after the initial pulse at each firing position. The parameter is applicable for pulses generated by the following functions: **FixedDistPulse**, **DistanceArrPulse**, **CoordinateArrPulse**. The value 0 means no extra pulses are generated (default).

5.1.2.18 ExtraPulsesPeriod

Description

The period in milliseconds for additional pulses defined in ExtraPulses to be generated after each pulse at each firing position. The parameter is applicable for pulses generated by the following functions: **FixedDistPulse**, **DistanceArrPulse**, **CoordinateArrPulse**. The value 0 means no extra pulses are generated (default).

5.1.2.19 *PiercePulsesNum*

Description

The number of pierce pulses generated during a pierce pulse cycle. The parameter is applicable for pulses generated by **FixedDistPulse** function. The value 0 means no pierce pulses are generated (default).

5.1.2.20 *PiercePulsesWidth*

Description

The width of pierce pulses in milliseconds. The parameter is applicable for pulses generated by **FixedDistPulse** function. The 0-value means, no pierce pulses are generated (default).

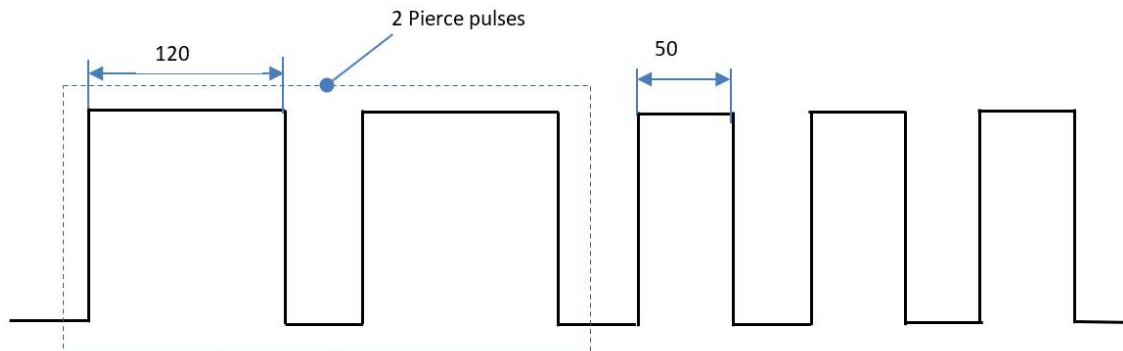


Figure 4-34. 2 Pulses with Pulse Width 120 μ sec.

5.1.2.21 *GateOnDelay*

Description

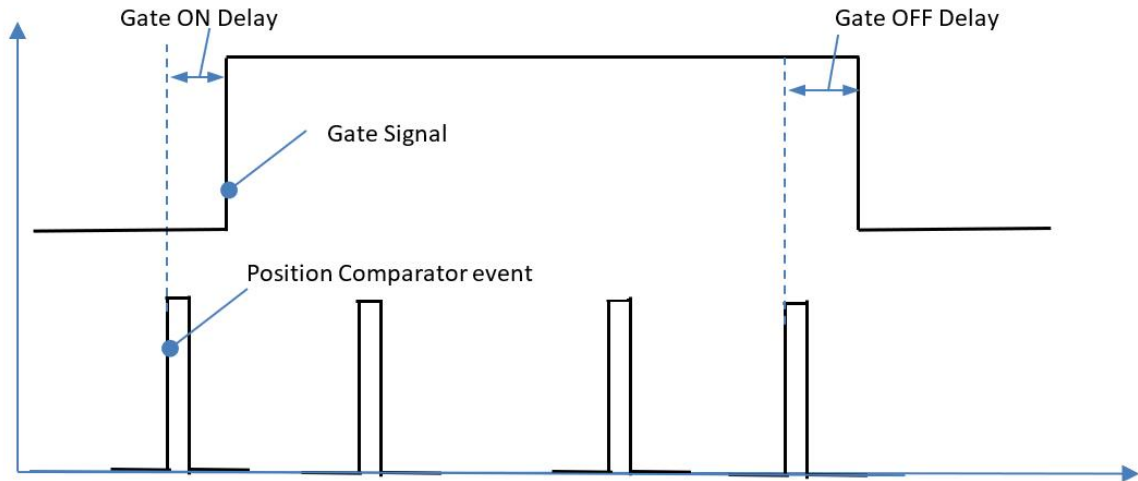
Returns state on delay in milliseconds when in Gating Mode. Default is 0. See [Figure 4-35](#)

5.1.2.22 *GateOffDelay*

Description

Returns state off delay in milliseconds when in Gating Mode. Default is 0.

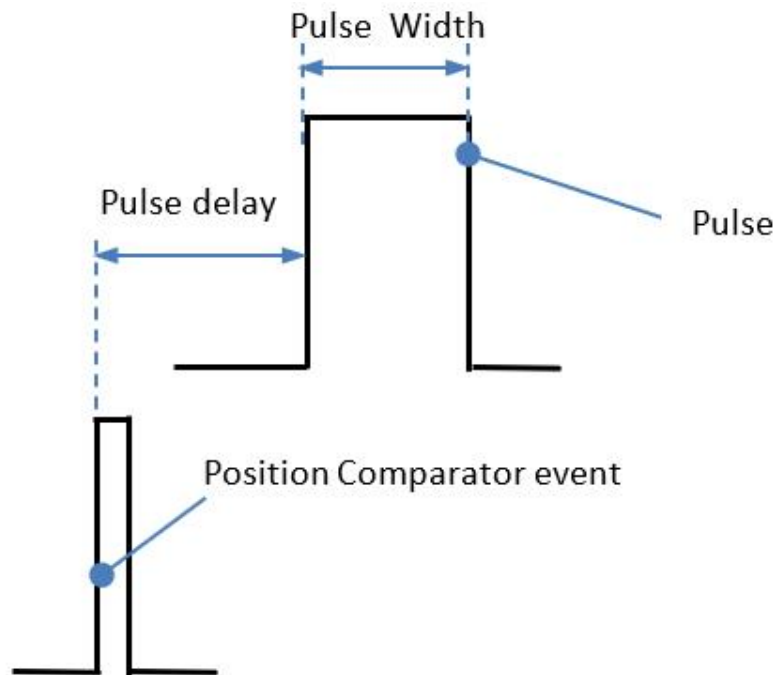
Figure 4-35. Delays in Gating Mode



5.1.2.23 PulseDelay

Description

Laser firing pulse delay in Distance Array Pulse modes. The default is 0.



5.1.2.24 PowerAOutVal

Description

PowerAOutVal is a Real Read-Only real field. It returns the current analog output value. The range is 0 to 100%. The field is updated if the Power Control via Analog Output mode is activated.

5.1.2.25 Faults

Description

Faults is a Read-only Integer Field. It contains a set of bits representing the current LCI error state, according to the following table:

| Bit # | Description |
|-------|-------------------------------------------------------------------------------------------|
| 0 | Laser Fault, fault input is On |
| 1 | Laser safety, safety input is Off |
| 2 | Velocity limit fault, motion axes velocity exceeds the maximum frequency supported by LCI |

5.1.2.26 PWMBurstReady

Description

PWMBurstReady is a read-only Integer field. It is interpreted as a boolean flag: 1 means that the system is ready to perform the next PWM burst.

5.2 Diagnostics and Preventive Maintenance (DPM)

The DPM related STRUCTs and functions answer the need to monitor system performance degradation. The user can implement the required measurements and diagnostics with a minimal effort.

The DPM feature allows the user to identify:

1. Performance degradation
 - > Increase of following error
 - > Increase in settling time
 - > Increase in velocity ripple during constant velocity
2. System degradation (changes that indicate some mechanical or electrical deterioration)
 - > Increase of current consumption
 - > Increase of temperature
3. Part failure (root cause of deterioration)
 - > Coupling break
 - > Cable break / partial break
 - > Motor deterioration due to over temperature
4. Maintenance based on smart measurement
 - > Bearing life measurement for preventive maintenance (like the warning that the car generates about the time for periodic maintenance.)
 - > Moving cable cycles measurement for preventive maintenance
 - > Drive life shortening due to repeated short circuits

This object is supported in version 3.10 and higher.

5.2.1 DPM_Measurement

Description

DPM_Measurement is an ACSPL+ composite data type(STRUCT) with different fields of data types and functions that are used to configure and store the results of the MeasureProcess()/MeasurePeriodically() functions. It is part of the DPM(Diagnostics and Preventive Maintenance) feature. To declare an instance of the DPM_Measurement STRUCT the user will use the reserved word DPM_Measurement followed by the desired struct name.

Syntax

```
DPM_Measurement Struct_name
```

See the *ACSPL+ Commands & Variables Reference Guide* for details about specific commands, functions, and variables.

5.2.1.1 DPM_Measurement Fields

| Variable Type and Name | Description |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| real variable_value | <p>It holds the raw instant measured value of the variable specified by the user. This variable is updated every MPU cycle once the measurement is initiated. Default = 0.</p> <p>Types of variables that can be monitored:</p> <ul style="list-style-type: none"> > Any ACSPL+ standard variable. > User variables that are defined in the D-Buffer as global and static.(Including Structs variables if the Struct is defined as global and static) |
| real variable_abs_value | <p>It holds the absolute value of the variable_value. This variable is updated every MPU cycle once the measurement is initiated.</p> |
| real measured_abs_value | <p>Each MPU cycle this variable is set (or updated) to variable_abs_value * when_to_measure</p> <p>This variable holds the absolute value of the variable_value when when_to_measure is 1, and 0 otherwise.</p> |
| real sampled_value | <p>Holds the latest sample of measured_abs_value.</p> <p>See definition of a sample under sampling_type</p> |

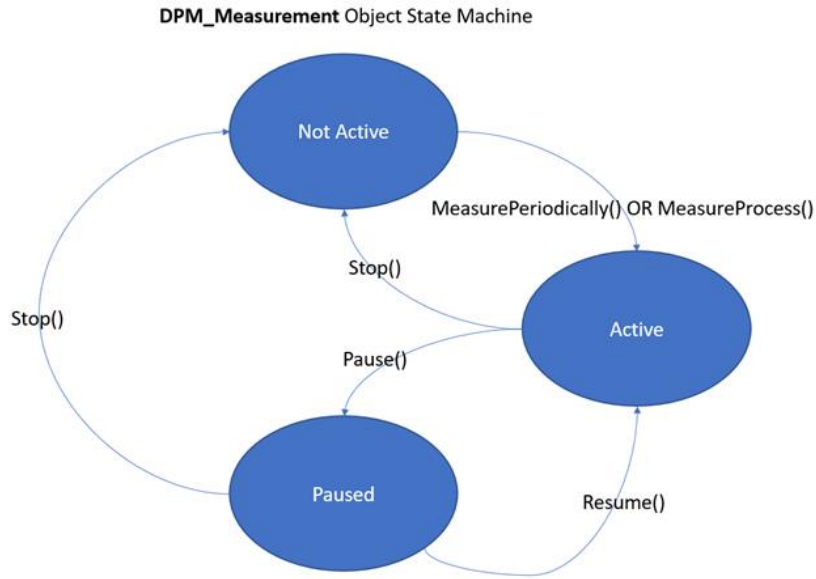
| Variable Type and Name | Description |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| int measurement_status | <p>Holds the current status of the DPM_measurement structure</p> <p>0 – measurement is NOT ACTIVE</p> <p>1 – measurement is PAUSED</p> <p>2 – measurement is ACTIVE</p> |
| int measurement_type | <p>Specifies the current measurement type.</p> <p>0 – None</p> <p>1 – Process measurement</p> <p>2 – Periodical measurement</p> |
| int sample_set_size | <p>Specifies the size of the set of adjacent samples to be used for the calculation of the moving_average_value, std_dev_value, and the rms_value.</p> <p>Values: Ranges between 1 and 512.</p> |
| int when_to_measure | <p>when_to_measure defines the event slots during which the variable_abs_value is measured and assigned to measured_abs_value.</p> <p>Values : 0 or 1.</p> <p>For example: Motor Current during the acceleration phase of a moving axis. For measurements related to motion, the DPM_Motion_Status struct can be used.</p> <p>Types of variables that can be used:</p> <ul style="list-style-type: none"> > Any ACSPL+ integer standard variable. > Any user variable that is defined in the D-Buffer as integer global and static (Including Struct variables if the Struct is defined as global and static). |
| real sampling_time | <p>sampling_time defines the sampling period in milliseconds when using the MeasurePeriodically() function.</p> <p>Minimum value: MPU cycle time.</p> <p>Maximum value: 100,000,000 msec (>24h).</p> <p>MeasurePeriodically() sample will be taken first on the positive edge of when_to_measure, and then every sample_time period (as long as when_to_measure is 1(True)).</p> <p>The resolution of sampling_time is the MPU cycle time.</p> |
| int sampling_type | <p>The sampling_type is used by the MeasureProcess() function, and defines the nature of a sample:</p> |

| Variable Type and Name | Description |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> > When to sample > What is a sample and how it is calculated <p>0 – The measured_abs_value is sampled upon the positive edge of when_to_measure, and the sampled_value is set to this value. As an example, it may be used to measure the settling time of moves.</p> <p>1 – The measured_abs_values are summed up every MPU cycle, and the sampled_value is set to the average value of the summation upon the negative edge of when_to_measure.</p> <p>For example, it may be used to measure current during acceleration. sampled_value presents the average of the absolute current during the acceleration period.</p> <p>This variable is ignored when using MeasurePeriodically(). In a periodic measurement, the sampling_time variable is used. See sampling_time definition for more information.</p> |
| real sample_counter | <p>Specifies the number of samples that have been collected since the beginning of the measurement process.</p> <p>See definition of a sample at sampling_type</p> |
| real peak_value | <p>Holds the highest value of a sample that has been taken since the beginning of the measurement.</p> <p>It can be set to 0 by the user by calling ClearPeak().</p> |
| real moving_average_value | <p>Holds the average value of the samples in the set. The number of samples in the set is specified by sample_set_size.</p> <p>See definition of a sample at sampling_type</p> |
| real std_dev_value | <p>Holds the standard deviation of the samples in the set. The number of samples in the set is specified by sample_set_size.</p> |
| real rms_value | <p>Holds the RMS (Root mean square) of the samples in the set. The number of samples in the set is specified by the sample_set_size.</p> |

5.2.1.2 DPM_Measurement Functions

| Function Name | Arguments | Description |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>MeasureProcess (monitored_variable, when_to_measure, sample_set_size, sampling_type)</p> | <p>real/Int monitored_variable – A variable to be monitored. (see monitored_variable field definition for more details)</p> <p>int when_to_measure – A variable that acts as a flag. when the value of the variable is different than 0, a measurement will take place. (see when_to_measure field definition for more details)</p> <p>int sample_set_size – Defines the samples set size. (see sample_set_size field definition for more details)</p> <p>int sampling_type – Defines the sampling behavior(see sampling_type definition for more details)</p> | <p>This function is used to initiate monitoring and measurements of variables that are related to processes and especially motion processes that are generated by atomic motion commands (PTP, BPTP, JOG).</p> <p>Example: The average current during the acceleration phase of a moving axis.</p> |
| <p>MeasurePeriodically (monitored_variable, when_to_measure, sample_set_size, sampling_time)</p> | <p>real/Int monitored_variable – A variable to be monitored. (see monitored_variable definition for more details)</p> <p>int when_to_measure – A variable that acts as a flag. When the value of the variable is different than 0, a measurement will take place. (see when_to_measure field definition for more details)</p> <p>int sample_set_size – Defines the samples set size. (see sample_set_size field definition for more details)</p> | <p>This function is used to initiate the monitoring and measurements of the supplied variable over time.</p> <p>Example 1: measurement of temperature that is monitored by a sensor that is connected to an analog input</p> <p>Example 2: 2D position error during one “long” arbitrary XY move</p> |

| Function Name | Arguments | Description |
|--------------------|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | real sampling_time – Defines the period between samples(see sampling_time definition for more details) | |
| Stop() | None | Stops the measurement if it is active or paused. It does not affect the measurement if it is not active. |
| Pause() | None | Pauses the measurement if it is active. It does not affect the measurement if it is not active. |
| Resume() | None | Resume previously paused measurement. Does not affect the measurement if it is not paused. |
| ClearAll() | None | Sets variable_value , variable_abs_value , measured_abs_value , sampled_value , sample_counter , peak_value , moving_average_value , std_dev_value , rms_value and when_to_measure to 0. |
| ClearPeak() | None | Sets peak_value to 0. |



5.2.2 DPM_Motion_Status

Description

DPM_Motion_Status is an ACSPL+ composite data type (STRUCT), with different fields of data types and functions that are used to configure and to provide an indication on the phase of the motion for a user-selected axis. It is part of the DPM (Diagnostics and Preventive Maintenance) feature. To declare an instance of the DPM_Motion_Status STRUCT the user uses the reserved word DPM_Motion_Status followed by the desired struct name.

Syntax

```
DPM_Motion_Status struct_name
```

5.2.2.1 DPM_Motion_Status Fields

| Variable Name | Description |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| int during_motion | The variable is set to true (=1) once the motion profile starts and is set automatically to false (=0) once the motion profile is completed. |
| int during_accel | The variable is set to true (=1) once the motion profile reaches the acceleration phase(can be constant acceleration or during jerk) and is set automatically to false (=0) once the acceleration phase is completed. |

| Variable Name | Description |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| int during_decel | The variable is set to true (=1) once the motion profile reaches the deceleration(can be constant deceleration or during jerk) and is set automatically to false (=0) once the deceleration phase is completed. |
| int during_cv | The variable is set to true (=1) once the motion profile reaches the Constant Velocity (CV) phase and is set to false (=0) once deceleration has started. |
| int selected_axis | The variable specifies the selected axis for which its motion phases are monitored. |
| int on_off | When 1, the motion phases are monitored. When 0, the phases are not monitored. |

5.2.2.2 DPM_Motion_Status Functions

| Function | Arguments | Description |
|--------------------------|----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SelectAxis (Axis) | Axis - Valid values are: 0, 1, 2, ... up to the number of axes in the system minus 1. | Axis - Specifies the axis is monitored once MonitorOn() is called. *Can only be used if the on_off variable is OFF (=0)(monitoring is not active). |
| MonitorOn() | None | Provides the user the ability to activate monitoring of the selected_axis. Once the MonitorOn() has been called, the motion phase and settling status of move commands related to the Selected_Axis will be indicated by the STRUCT variables. |

| Function | Arguments | Description |
|---------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MonitorOff() | None | Provides the user the ability to deactivate monitoring of the selected_axis. Move commands that are issued after calling MonitorOff() will not be monitored. |
| ClearAll() | None | Sets during_motion , during_accel , during_decel , and during_cv to 0 |

5.2.3 DPM Example - Adding current measurement during acceleration phase to an existing application

This example demonstrates how measurements can be added to existing applications without modifying existing code. In this example, the current during acceleration of every move of the Y-axis is measured.

To implement the measurement for an existing application:

1. Add code to an AUTOEXEC routine that initiates the measurement.
2. Add a condition monitoring (ON) routine that raises a flag when a threshold is exceeded.

The setting and resetting of **motion_status_Y.MonitorOn() /MonitorOff()** provides a mechanism to control which moves are monitored and measured.

```
D-Buffer:
axisdef Y=1
Global static DPM_Measurement actual_current_during_accel_Y
Global static DPM_Motion_Status motion_status_Y
Global static REAL Y_Accel_Peak_Current_Threshold, Y_Accel_Moving_
Average_Current_Threshold, CURRENT_Y_AMP
Global INT Samplesample_set_size_Y, DRIVE_PEAK_CURRENT_AMP, ADC_RANGE
Global static INT measure_continuously

Buffer 0:
AUTOEXEC:
Y_Accel_Peak_Current_Threshold = 5
Y_Accel_Moving_Average_Current_Threshold = 2.5
measure_continuously = 1
sample_set_size_Y = 20
DRIVE_PEAK_CURRENT_AMP = 10
ADC_RANGE = 32767 ! Current is sampled with a 16 bit ADC

motion_status_Y.SelectAxis(1) ! Set axis 1 as the axis that provides the
motion phases to the the motion_status_Y STRUCT once the motion_status_
Y.MonitorOn() function is called
motion_status_Y.MonitorOn() !Monitoring is activated. The STRUCT will be
updated for each move that follows, and measurement will be made
accordingly
```

```

actual_current_during_accel_Y.Stop() !Stops the measurement if it is
active
!Measurement activation
actual_current_during_accel_Y.MeasureProcess(CURRENT_Y_AMP, motion_
status_Y.during_accel, sample_set_size_Y, measure_continuously)

WHILE(1) !Run every MPU cycle
CURRENT_Y_AMP = DOUT(1) * (DRIVE_PEAK_CURRENT_AMP /ADC_RANGE)!converts
DOUT units to Amperes
END
STOP

Buffer 1:
ENABLE 1
LOOP 10
PTP 1,1000
BPTP 1,3000
PTP 1,4000
PTP 1,0
.
.
.
END
STOP
!acceleration phase peak threshold has been exceeded
ON actual_current_during_accel_Y.peak_value > Y_Accel_Peak_Current_
Threshold
DISP "Y peak current threshold is exceeded"
actual_current_during_accel_Y.ClearPeak()
RET
!acceleration phase moving average threshold has been exceeded
ON actual_current_during_accel_Y.moving_average_value > Y_Accel_Moving_
Average_Current_Threshold
DISP "Y moving average current threshold is exceeded"
RET

```

5.3 Motion Duration

5.3.1 MotionDuration Struct

The MotionDuration struct can be used to calculate motion duration according to motion type and motion parameters.

The MotionDuration struct calculation functions calculate motion duration in seconds. Functions of the struct can be executed in buffer only, and it will wait till the execution is completed.

Struct Fields

| Field Name | Type | Accessibility | Range | Comments |
|---------------------|------|---------------|---------------------------------------------------|---------------------|
| Distance | real | R/W | | user units |
| Velocity | real | R/W | from - 1.79769e+30 8 to 1.79769e+30 8 | Default = 10000 |
| Acceleration | real | R/W | from 2.22507e-308 to 1.79769e+30 8 | Default = 100000 |
| Deceleration | real | R/W | from 2.22507e-308 to 1.79769e+30 8 | Default = 100000 |
| Jerk | real | R/W | from 2.22507e-308 to 1.79769e+30 8 | Default = 2E7 |
| Snap | real | R/W | from 2.22507e-308 to 1.79769e+30 8 | Deafault = 100E7 |
| InitialVelocity | real | R/W | | Default = 0 |
| InitialAcceleration | real | R/W | | Default = 0 |
| FinalVelocity | real | R/W | | Default = 0 |
| FinalAcceleration | real | R/W | | Default = 0 |
| EncoderFactor | real | R/W | from 1e-15 to 1e+15 | Default = 1 |
| T_MotionOverallTime | real | R | | |

| Field Name | Type | Accessibility | Range | Comments |
|--------------------------------------------|------|---------------|-------|----------|
| T31_ JerkBuildupAccelerationBuild up | real | R | | |
| T1_ ConstantJerkAccelerationBuil dup | real | R | | |
| T32_ JerkFinishAccelerationBuildu p | real | R | | |
| T2_ConstantAcceleration | real | R | | |
| T33_ JerkBuildupAccelerationFinis h | real | R | | |
| T3_ ConstantJerkAccelerationFinish | real | R | | |
| T33_ JerkFinishAccelerationFinish | real | R | | |
| T4_ConstantVelocity | real | R | | |
| T35_ JerkBuildupDecelerationBuil dup | real | R | | |
| T5_ ConstantJerkDecelerationBui ldup | real | R | | |
| T36_ JerkFinishDecelerationBuildu p | real | R | | |
| T6_ConstantDeceleration | real | R | | |
| T37_ JerkBuildupDecelerationFinis h | real | R | | |
| T7_ ConstantJerkDecelerationFin | real | R | | |

| Field Name | Type | Accessibility | Range | Comments |
|----------------------------------|------|---------------|-------|----------|
| ish | | | | |
| T38_JerkFinishDecelerationFinish | real | R | | |

Struct Functions

| | |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReadFrom(int AxisNum) | Read Velocity, Acceleration, Deceleration, Jerk, Snap and EncoderFactor from specified axis and write them to appropriate fields in struct. |
| CalculatePTPDuration() | Calculates PTP (3rd order motion profile) motion duration and duration of each motion phase using struct parameters Relevant output members: T_MotionOverallTime, T1_ConstantJerkAccelerationBuildup, T2_ConstantAcceleration, T3_ConstantJerkAccelerationFinish, T4_ConstantVelocity, T5_ConstantJerkDecelerationBuildup, T6_ConstantDeceleration, T7_ConstantJerkDecelerationFinish |
| CalculateBPTPDuration() | Calculates BPTP motion duration using struct parameters Relevant output member: T_MotionOverallTime |
| CalculateSPTPDuration() | Calculates SPTP (4th order motion profile) motion duration and duration of each motion phase using struct parameters Output members: T_MotionOverallTime, T31_JerkBuildupAccelerationBuildup, T1_ConstantJerkAccelerationBuildup, T32_JerkFinishAccelerationBuildup, T2_ConstantAcceleration, T33_JerkBuildupAccelerationFinish, T3_ConstantJerkAccelerationFinish, T33_JerkFinishAccelerationFinish, |

```
T4_ConstantVelocity,  
T35_JerkBuildupDecelerationBuildup,  
T5_ConstantJerkDecelerationBuildup,  
T36_JerkFinishDecelerationBuildup,  
T6_ConstantDeceleration,  
T37_JerkBuildupDecelerationFinish,  
T7_ConstantJerkDecelerationFinish,  
T38_JerkFinishDecelerationFinish
```

Comments

- > Two options exist for initialization of the editable fields of the MotionDuration struct:
 - a. Initialize each writeable field manually, one by one.
 - b. Use the **ReadFrom** function to read the Velocity, Acceleration, Deceleration, Jerk, Snap and EncoderFactor fields from a specified axis. The Distance field will still need to be initialized manually.

Struct members: InitialVelocity, InitialAcceleration, FinalVelocity and FinalAcceleration will not be affected by calling the **ReadFrom** function.

- > Zero value in fields Velocity, Acceleration, Deceleration, Jerk, Snap and EncoderFactor will implicitly be replaced by the default value.
- > There may be calculated results that are not relevant for all types of motion; they will be initialized to their default value.

6. Terminal Commands

Terminal commands are those commands that are specific to the SPiiPlus MMI Application Studio Communication Terminal utility and are not part of the ACSPL+, nor can they be incorporated into ACSPL+ programming. As soon as the command is received through one of the communication channels, it is executed.

This chapter covers all of the available Terminal commands.

6.1 Entering Terminal Commands

Terminal commands are entered in the **Communication Terminal** (the general structure of which is shown in [Figure 5-1](#)).



Figure 5-1. Communication Terminal Window

The **Communication Terminal** window is described in the *SPiiPlus MMI Application Studio User Guide*.



Terminal commands are case sensitive.

Terminal commands are divided into:

- > [Query Commands](#)
- > [Program Management Commands](#)
- > [System Commands](#)

6.2 Query Commands

Query commands are designated by the question mark (?) and are entered through the **Communication Terminal**.

| Command & Syntax | Description | Example |
|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?[ACSPL+ variable] | Returns the current value of a standard variable . | ?TCPIP – Returns the TCP/IP for the Ethernet port N1. |
| ?[ACSPL+ variable ACSPL+ variable]... | Returns the current value of listed standard variables. | ?TCPIP, TCPPORT – Returns the TCP/IP for the Ethernet port N1 and the TCP port number. |
| ?[buffer number]: [local user-defined variable] | Returns the current values of a local user variable or array defined in a program buffer. | ?1:MY_VAR – Returns the values of a local user-defined variable named MY_VAR. |
| ?[array_ variable (index)] | Returns the current value of a specific element in the given array. The brackets enclosing the index are optional. | ?FPOS(0) – Returns the feedback position of 0 axis. ?FPOS0: Returns the feedback position of 0 axis. |
| ?[global user array_ variable [(index)]] | Returns the current values of a global user variable or array. Or the value of an element in the array if index is included. | ?ARRAY1 – Returns the values of a global user-defined array variable named ARRAY1. ?ARRAY1(3) – Returns the value of the fourth element of a global user-defined array variable named ARRAY1. |
| ?[matrix_ variable (row_ index) (col_ index)] | Returns the current value(s) of an element in a two dimensional matrix. The indices may be entered as a range, e.g., (0,4) is the first through fifth, inclusively. | ?MATRIX – Returns all of the values (in tabular format) of the two-dimensional array named MATRIX. ?MATRIX(1)(0) – Returns the value of the second element in the first column of MATRIX. ?MATRIX(0) – Returns all the values of the first row of MATRIX. ?MATRIX(0,2)(0,1) – Returns a range of values, those of the first through third rows in the first and second columns of MATRIX. |
| ?[buffer number] | Returns the current status and information about a program buffer | ?0 – Returns the current status and information about program buffer 0. |

| Command & Syntax | Description | Example |
|--------------------------|-------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| ?# | Returns the current status of all program buffers. | |
| ?\$[axis number] | Returns the current status of the motor for the specified axis. | ?\$16 – Returns the current status of the axis 16 motor. |
| ?\$ | Returns the current status of all motors. | |
| ?VR | Returns the firmware version | |
| ?SN | Returns the controller serial number and hardware version - indicated by a letter. | |
| ??[error_code] | Returns the error code number. If the error code is included in the query, returns error description. | ??3260 : Returns “Motor cannot start because the motor is disabled” |
| ??[variable_name] | Returns a brief description of the variable. | ??CERRA - Returns “Critical Position Error In Accelerating” |

6.2.1 Default Query Formats

All the queries described above produce a variable report in a default format depending on the queried variable. In some cases the default format produces unsatisfactory results. For very large or small real values the output may appear misleading because very large or small values may require more positions than allocated by the default.

The default format for all real variables is similar to the C “%10G” format, where each real value is represented by 10 digits. The controller automatically chooses the position of decimal point and the number of digits right to the decimal point. If required, the controller uses an exponential format, like 3.14E-13.

The default format for all user integer variables and for all standard integer variables, except State flags, and I/O variables, is similar to the C “%10i” format that specifies 10 digits for each integer number.

State and Flag variables are reported in special format. Each State and Flag variable is a collection of bits. Each bit within the variable has its own function. The variable is reported in a format that displays the state of each bit with a short explanation.

The standard I/O arrays IN and OUT are reported in binary format.

6.2.2 Predefined Query Output Formats

The controller provides several predefined formats that can be used instead of the default format for querying variables:

| Format | Description |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?D/ | <p>Decimal format.</p> <p>This format is identical to the default format for integer variables. When applied to a state variable, the format displays the decimal presentation of the variable.</p> <p>C-equivalent: %10i.</p> |
| ?X/ | <p>Hexadecimal format.</p> <p>When applied to an integer variable, this format displays the hexadecimal presentation of the variable.</p> <p>C-equivalent: %08X</p> |
| ?B/ | <p>Binary format.</p> <p>This format is identical to the default format for the IN and OUT variables. When applied to an integer variable, the format displays the binary presentation of the variable.</p> |
| ?E/ | <p>Extended format.</p> <p>This format is useful for very large or small real values, when the default format produces ambiguous results because the default does not provide enough positions to display very large or very small numbers. When applied to a real variable, the format displays each value in 20 positions.</p> <p>C-equivalent: %20G</p> |

Examples

Display the motor state in decimal format:

```
?D/ MST
3      15      15      3      0      0      0      0
X/Y_MST
0      0      0      0      0      0      0      0
```

Display the state of motor x in binary format

```
?B/ MST
00000000,00000000,00000000,00000011
```

Display the status of variable UserReal in extended format

```
?E/UserReal
1.0000000000000001
```

6.2.3 User-Defined Query Output Format

If the default format or any other predefined format is not suitable for the user needs, the user may specify a format using C notation. The specification is placed just before the variable name in curled brackets. The specification applies only to the value of the name that is specified in the command. If an array is queried, the specification applies to each element of the array.

C notation provides an unlimited number of possible formats.

Examples

| Format | Description |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>?{%12.3f}FPOS</code> | The motor feedback values for all axes are displayed in 12 digits, fixed decimal point, 3 digits after the point. The same format applies to all 8 values. |
| <code>?{%8.0f}X_FPOS</code> | 8 digits, no decimal point, no fraction digits. |
| <code>?{XFPOS = %8.0f}X_FPOS</code> | The response will look like XFPOS = 1234. |
| <code>?{%08X} X_MST</code> | 8 digits, hexadecimal format with leading zeros |

6.3 Program Management Commands

Program Management commands are used for:

- > Controlling program execution
- > Viewing and editing program content

The Program Management commands are designated by the pound (#) character.

6.3.1 Program Management Command Arguments

Program Management commands can take two types of arguments:

Buffer Designation

Buffer designation can be specified in two forms:

- > An integer number, between 0 and 16, that addresses a specific buffer (16 addresses the D-buffer)
- > The pound character, #, that addresses all program buffers in one command

Only the following commands can address all buffers:

- > L – List
- > C – Compile
- > S, SR – Stop, Stop and Reset
- > P – Pause
- > F, FI – Find
- > BR – Reset Breakpoints

All other commands operate with one buffer only, and must specify the buffer number.

Line Designation

A line designation can appear in one of three forms, as shown in the following table:

Table 5-1. Line Designation

| Line Qualifier Type | Description |
|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Single number | Specifies only one line. |
| Two numbers separated by comma | Specifies a range of lines. If the second number is larger than a total number of lines in the program, the list range spans the last program line. |
| Label preceded by a slash (/) character | Specifies the line by a designated label (the text following the slash). |
| Label preceded by a slash (/) character, then a comma and number | Specifies a range of lines starting from the line with a designated label. |

Examples

The following are examples of using the **L** command.

List line 4 in buffer 3.

#3L4

```
4: till ^MST(0).#MOVE
```

List lines from 1 to 3 inclusively in buffer 5.

#5L1,3

```
1: movePTP:
2: VEL(0) = 20000
3: ptp 0, 4000
```

List the line that contains the label **MovePTP** in buffer 5.

#5L/MovePTP

```
1: movePTP:
```

List 3 lines, starting from the label **movePTP** in buffer 5.

#5L/MovePTP,3

```
1: movePTP:
2: VEL(0) = 20000
3: ptp 0, 4000
```


6.3.2 Program Buffer Commands

6.3.2.1 Open/Close Buffer (#)

Description

The # (Open/Close Buffer) command is used to open and close a buffer for the purpose of entering code.

Syntax

```
#buffer_number[!][line_number]
```

Arguments

| | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| buffer_number | buffer_number qualifier in the command specifies the buffer, a number between 0 and 16. |
| ! | Optional, if included, opens the buffer for insertion of code. |
| line_number | Optional, if included, the command opens the buffer specified by buffer_number and sets the insert line before the line specified by line_number . line_number can be specified in any of the three forms listed in. |

Comments

The # command opens the buffer and sets the insert line as follows:

- > If the buffer is empty, the insert line is 1.
- > If the buffer already contains a program, the insert line is set after the last line of the program.

Only one buffer can be opened at a time.

To close the buffer, # is entered without the buffer number (since only one buffer is open, the controller knows which buffer to close).

Examples

The following are examples of opening and closing buffers.

```
#0          Open buffer 0
0:00001>    Buffer 0 is empty
#3          Open buffer 3
3:00006>    This indicates that buffer 3 contains 5 lines. The insert
            line is set to 6.
#3          Open buffer 3 for insertion
3:00006>    The command does not specifies a line qualifier. It is
            identical to the command #3.
#3!2       Open buffer 3 and set insertion point prior to line 2.
3:00002>    Insert line is set before line 2
#          Close the buffer
:          The prompt indicates that all buffers are closed
```

When a program buffer is open, ACSPL+ commands are stored in the buffer (and are not executed immediately). The controller checks the syntax of inserted ACSPL+ lines and immediately reports any errors detected.

The following is an example of an editing session:

```
#0                                Open buffer 0
0:00001>                          Buffer 0 is empty
>VEL(0) = 20000                    Enter the program lines sequentially. After each
                                    line, the insert line number is increased by 1.

0:00002>
ptp X, 4000
0:00003>
till ^MST(0).#MOVE
0:00004>

stop
0:00005>
#0L                                List the program
1: VEL(0) = 20000
2: ptp 0, 4000
3: till ^MST
(0).#MOVE
4: stop
0:0005>                          The buffer remains open
#0I1                              Change the insert line number to 1
0:0001>                          Insert line is set before the first line
MovePTP:                          Insert label
0:0002>
#0L                                List the program
1: MovePTP:
2: VEL(0) = 20000
3: ptp 0, 4000
4: till ^MST(0).#MOVE
5: stop
0:0002>                          The buffer remains open
#                                Close the buffer
```

6.3.2.2 D

Description

The **D** (Delete) command deletes the specified lines in the buffer.

Syntax

#buffer_numberDline_number[,line_number]

Arguments

| | |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number qualifier in the command specifies the buffer, a number between 0 and 16. |
| <i>line_number</i> | line_number in the D command is obligatory. line_number can be: <ul style="list-style-type: none"> > A single number, specifying one specific line, or > Two numbers separated by comma that specify a range of lines. If the second number is larger than a total number of lines in the program, the delete range includes the last program line. |

Comments

If a buffer is open, the **D** command that addresses the buffer shifts the insert line to before the first undeleted line.

Example

```
#0L          Open buffer 0
0:00001>    Buffer 0 is empty
#3          Open buffer 3
3:00006>    Buffer 3 contains 5 lines. The insert line is set to 6.
#3I
3:00006>    The command does not specifies a line qualifier. It is
            identical to the command #3

#3I2
3:00002>    Insert line is set before line 2
#0D3       Delete line 3 in buffer 0
#          Close the buffer
:         The prompt indicates that all buffers are closed
```

6.3.2.3 F/IF

Description

The **F/IF** (Find/Find Case-sensitive) commands are used to search for a specific text in a specified buffer or in all buffers.

Syntax

#buffer_number{F|FI}/search_string [,line_number]

Arguments

| | |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number qualifier in the command specifies the buffer, a number between 0 and 16. |
| <i>search_string</i> | The text being sought. |
| <i>line_number</i> | Optional, if included, line_number defines the start line for the search. Otherwise, the search starts from the first line. |

Comments

search_string must be specified as a label, that is, it must be preceded by a slash (/), or as a label and number separated by comma. **search_string** can be any text, such as, a variable name, ACSPL+ command, constant, label or keyword.

The search terminates when the first entry of the specified text is found, or the buffer end is reached. The command reports the line that contains the text, or an error message if the text was not found.

To find the next entry, the user must execute the command again, specifying the new start line number of the reported line plus one.

If the # character is specified instead of the buffer number, the search command addresses all buffers. In this case the command finds the first entry in each buffer.

Examples

The following are examples of using the **F** command.

```
#0F/X           Find "X" in buffer 0
0002           Response: found entry in line 2
#0F/X,3        Find the next entry, starting from line 3
0003           Response: found entry in line 3
#0F/X,5        Find the next entry, starting from line 5
?1078          Response: No more entries
##F/stop       Find stop in all buffers
Buffer #0: no matches
Buffer #1: 4: stop           Entry is found in line 4 of Buffer 1
Buffer #2: no matches
Buffer #3: 4: stop           Entry is found in line 4 of Buffer 3
Buffer #4: 1: stop           Entry is found in line 1 of Buffer 4
Buffer #5: no matches
Buffer #6: no matches
Buffer #7: no matches
Buffer #8: 238: stop        Entry is found in line 238 of Buffer 8
Buffer #9: no matches
Buffer #10: no matches
```

6.3.2.4 L

Description

The **L** (List) command is used for displaying a program listing.

Syntax

#buffer_numberL[line_number]

Arguments

| | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number specifies the buffer, a number between 0 and 16; or you can use the pound (#) to designate all buffers. |
| <i>line_number</i> | Optional, if included, line_number defines a specific line to be listed. Otherwise, the search starts from the first line. line_number can be specified in any of the three forms listed in Line Designation . |

Comments

The listing contains all program lines preceded by line numbers. Each line appears exactly as it was inserted. No automatic formatting is provided.

To address all buffers the # character is used instead of the buffer number, for example, the command **##L** provides a listing of all programs in all buffers. If **line_number** is included and a buffer does not contain the specified line number, only the buffer number is listed.

If the buffer is empty, the list includes the buffer designation followed by the first line (0) which is blank.



It is recommended that you place a remark with a short program description in the first line of each program. This enables using the command **##L1** to get quick information about all loaded programs.

Examples

The following are examples of the **L** command.

Example 1:

Provide a program listing for buffer #3:

#3L

```
1: MovePTP:
2: VEL(0) = 20000
3: ptp X, 4000
4: till ^MST(0).#MOVE
5: stop
```

Example 2:

Provide a program listing for the first line in all buffers:

```
##L1           List the contents of line 1 in all buffers.
Buffer 0      Response
0:
Buffer 1
1: ! Homing of all axes
Buffer 2
1: ! Registration motion
Buffer 3
```

```

1: MovePTP:
Buffer 4
1: ! PLC program
Buffer 5                               Buffer 5 is empty
0:
Buffer 6                               Buffer 6 is empty
0:
Buffer 7                               Buffer 7 is empty
0:
Buffer 8                               Buffer 8 is empty
0:
Buffer 9                               Buffer 9 is empty
0:

```

6.3.3 RESET

Description

The **RESET** command is used to reset the controller to factory default state.

Syntax

#RESET

Comments

The **RESET** command can be issued even if the application is in the Protected mode in which case the password, if included, is not needed.

6.3.4 Listing Program Variables

There are three types of variables:

- > ACSPL+ Variables – variables contained in the ACSPL+ language set
- > SP Variables – variables incorporated in the controller
- > User-Defined Variables – variables that have been declared by the user

For each type of variable there is a Communication Terminal command for listing them.

6.3.4.1 VGR

Description

The **VGR** command lists the categories within which the ACSPL+ variables are grouped. The categories of the ASCPL+ variables are:

- > Axis_State
- > Monitoring
- > Motion
- > Safety_Control
- > Inputs_Outputs
- > Program_Execution_Control
- > System_Configuration

- > Axis_Configuration
- > Communication
- > Commutation
- > Data_Collection
- > Servo_Loop
- > Miscellaneous
- > Obsolete

Syntax

#VGR [group_name]

Arguments

| | |
|-------------------|----------------------------------------|
| <i>group_name</i> | One of the ACSPL+ variable categories. |
|-------------------|----------------------------------------|

Comments

If **group_name** is omitted, the command lists only the categories. If **group_name** is included, the command lists the ACSPL+ sub-categories within the category.



The category must be entered in exactly the same format as given in the list above.

6.3.4.2 VSD

Description

The **VSD** command lists all ACSPL+ variables with a short description.

Syntax

#VSD [group_name]

Arguments

| | |
|-------------------|-------------------------------------------------------------------|
| <i>group_name</i> | One of the ACSPL+ variable categories (see VGR). |
|-------------------|-------------------------------------------------------------------|

Comments

When **group_name** is included in the VSD command, the ACSPL+ variables within the specified category and a brief description of each variable is listed.

6.3.4.3 VS/VSG

Description

The **VS/VSG** commands are used to list the variables that are incorporated in the ACSPL+ language set.

Syntax

#VS

#VSG [group_name]

Arguments

| | |
|-------------------|-------------------------------------------------------------------|
| <i>group_name</i> | One of the ACSPL+ variable categories (see VGR). |
|-------------------|-------------------------------------------------------------------|

Comments

When **group_name** is included in the **VSG** command, the names of the ACSPL+ variables within the specified category are listed.

Example

The following is an example of the **VS** command.

```
#VS          List ACSPL+ variable names
              Response
ACC
AFLAGS
AIN
AOUT
APOS
AST
BAUD
BOFFTIME
BONTIME
MASK
FPOS
FVEL
FVFIL
GACC
GETIME
GJERK
GMOT
GMQU
GMTYPE
```



For brevity, only a portion of the response is shown here.

6.3.4.4 VSF/VSGF

Description

Both the **VSF** and **VSGF** commands, in addition to the global variable names, display the variable type, the number of elements (for arrays only), address of the variable in the controller memory and the step between array elements (for arrays only).

Syntax

#VSF

#VSGF [group_name]

Arguments

| | |
|-------------------|-------------------------------------------------------------------|
| <i>group_name</i> | One of the ACSPL+ variable categories (see VGR). |
|-------------------|-------------------------------------------------------------------|

Comments

When **group_name** is included in the **VSGF** command, the names of the ACSPL+ standard variables within the specified category and their details are listed.

6.3.4.5 VG/VGF

Description

The **VG** command lists all global variable names in the system.

The **VGF** command, in addition to the global variable names, lists the variable type, the number of elements (for arrays only), address of the variable in the controller memory and the step between array elements (for arrays only).

Syntax

#[buffer_no]VG

#[buffer_no]VGF [variable_name]

Arguments

| | |
|----------------------|----------------------------------------------------------------|
| <i>buffer_no</i> | A number ranging from 0 to 16, representing a specific buffer. |
| <i>variable_name</i> | A specific ACSPL+ variable. |

Comments

If **buffer_no** is included, **VG** and **VGF** list all the global variables in the specified buffer.

If **variable_name** specifying an ACSPL+ variable is included, **VGF** lists the details just for the specified variable

6.3.4.6 VL/VLF

Description

The **VL** command lists all local variable names in the system.

The **VLF** command, in addition to the local variable names, lists the variable type, the number of elements (for arrays only), address of the variable in the controller memory and the step between array elements (for arrays only).

Syntax

#[buffer_no]VL

#[buffer_no]VLF [variable_name]

Arguments

| | |
|----------------------|----------------------------------------------------------------|
| buffer_no | A number ranging from 0 to 16, representing a specific buffer. |
| variable_name | A specific ACSPL+ variable. |

Comments

If **buffer_no** is included, **VL** and **VLF** list all the local variables in the specified buffer.

If **variable_name** specifying an ACSPL+ variable is included, **VGF** lists the details just for the specified variable.

6.3.4.7 V/VF

Description

The **V/VF** (List User-Defined Variable Names only/List User-Defined Variables with Description) commands are used to list the user-defined variables that are found in compiled programs.

Syntax

#[buffer_number]V

#[buffer_number]VF

Arguments

| | |
|------------------|----------------------------------------------------------------|
| buffer_no | A number ranging from 0 to 16, representing a specific buffer. |
|------------------|----------------------------------------------------------------|

Comments

If **buffer_no** is not specified, the list includes the user-defined variables in all compiled buffers. The **V** command only displays the names of the user-defined variables. The **VF**, on the other hand, in addition to the variable names, it displays the variable type, the number of elements (for arrays only), address of the variable in the controller memory and the step between array elements (for arrays only).

The list can be saved to a file by clicking **Save** in the **Communication Terminal** window.

Examples

The following are examples of the **V** and **VF** commands.

```
#9V                                     Provide a list of user variables
                                         in buffer 9

                                         Response

ITIME   Global
TS_AMP1 Local
TS_AMP0 Global
#9VF                                         Provide a list of user variables
                                         in buffer 9 with additional information

                                         Response

ITIME   Global   real@00DA0C50
```

```
TS_AMP1 Local    int@00DA1A30
TS_AMP0         Global int@00DA0C80
```

6.3.4.8 VSP

Description

The **VSP** (List Servo Processor Variables) command provides a list of the SP variables that are defined in the program in the specified SP.

Syntax

#VSPservo_number

Arguments

servo_number *servo_number* is number of the Servo Processor.

Comments

Each variable name in the list is accompanied by an SP address of the variable.

The list can be saved to a file by clicking **Save** in the Terminal window.

Example:

```
#VSP0                               List the variables in SP 0
                                     Response
_RMS_SUM @088
A_RMS_SUM @089
X_NOTCH_OUT_PR_H @08A
X_NOTCH_OUT_PR2_H @08B
X_NOTCH_IN_PR @08C
X_NOTCH_IN_PR2 @08D
X_NOTCH_FRC @08E
X_NOTCH_FRC_L @08F
A_NOTCH_OUT_PR_H @091
A_NOTCH_IN_PR @092
```

6.3.4.9 VST/VSGT

Description

Both the **VST** and **VSGT** commands display a list of ACSPL+ variables to which **PROTECT** can be applied.

Syntax

#VST

#VSGT [group_name]

Arguments

group_name One of the ACSPL+ variable categories (see [VGR](#)).

Comments

When **group_name** is included with the **VSGT** command, the ACSPL+ variables within the specified category are listed.

6.3.4.10 VSTF/VSGTF/VSDT

Description

The **VSTF**, **VSGTF**, and **VSDT** commands all list the variable names, the variable type, the number of elements (for arrays only), address of the variable in the controller memory and the step between array elements (for arrays only) of those ACSPL+ variables to which protection can be applied.

Syntax

#VSTF

#VSGTF [group_name]

#VSDT [group_name]

Arguments

| | |
|-------------------|-------------------------------------------------------------------|
| <i>group_name</i> | One of the ACSPL+ variable categories (see VGR). |
|-------------------|-------------------------------------------------------------------|

Comments

When **group_name** is included with the **VSGTF** or **VSDT** command, the ACSPL+ variables within the specified category are listed.

6.3.4.11 VGV

Description

The **VGW** command is used to remove global variables that have been set via **Communication Terminal** or **STATIC** variables defined in the D-Buffer..

Syntax

#VGW [global_var]

Arguments

| | |
|-------------------|---------------------------|
| <i>global_var</i> | Name of a global variable |
|-------------------|---------------------------|

Comments

If **global_var** is included, **VGW** removes only this variable; otherwise it removes all of them.

6.3.4.12 VGS/VGSF

Description

The **VGS** command is used to list all **STATIC** variables currently defined.

The **VGSF** command, in addition to **STATIC** variable names, also lists the type, number of elements (arrays only) and address of the variable in the controller memory.

Syntax

```
#VGS [Static_Var]
#VGSF [Static_Var]
```

Arguments

Static_Var

Optional parameter, check if *Static_Var* is a **STATIC** variable that is currently defined.
Leave empty to list all variables.

6.3.5 Program Handling Commands

These commands are used for compiling, executing, pausing and halting the program.

The commands in this set are:

- > **C** - Compile Program
- > **X** - Execute Program
- > **S/SR** - Stop/Stop & Reset Program
- > **P** - Pause Program

The following diagram shows the program buffer states and the Communication Terminal commands that affect the buffer states:

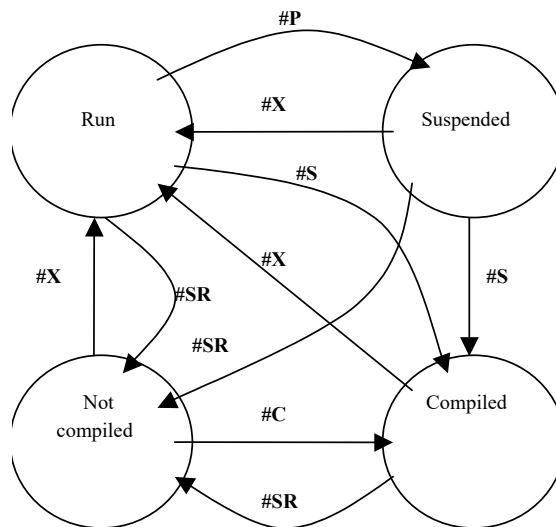


Figure 5-2. Interaction of Program Buffer States

The program buffer enters the Not Compiled state:

- > After any change in the program text.
- > Following execution of the **#S** (Stop) or **#R** (Reset) command.

The program buffer enters the **Compiled** state:

- > Following execution of the **#C** (Compile) command the program buffer is transferred from the **Not Compiled** to the **Compiled** state.

- > The **#S** (Stop) command transfers the program buffer from the Run or Suspended state to Compiled state.
- > Following program termination with an ACSPL+ **STOP/STOPALL** command (or **RET** if an autoroutine was executed).
- > When the program fails due to an error.

The program buffer enters the **Run** state:

- > When the **#X** (Execute) command is issued.
- > When another program executes a **STOP/STOPALL** command, or autoroutine condition is satisfied.

6.3.5.1 C

Description

The **C** (Compile) command compiles a program in the buffer or all programs in all buffers, depending on the buffer qualifier.

The **C** command must not include a line qualifier and is prohibited when the buffer is in the Run or Suspended states.

Syntax

#buffer_numberC

Arguments

| | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number specifies the buffer, a number between 0 and 16; or you can use the pound (#) to designate all buffers. |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------|

Comments

The **C** command is not obligatory in order to execute a program. When the **X** (Execute) command is issued, the controller automatically compiles the program if it was not previously compiled.

However, a separate compilation step is required in the following cases:

- > To check the program correctness without executing it.
- > The program is not intended for direct starting, but contains autoroutines. The autoroutines are ready for execution only after compilation.
- > The program is intended for starting from another program by the **START** command. The program started by the **START** command must be compiled before the **START** command can be executed.

If the program is successfully compiled, the controller prints a short report of how many lines were compiled. If an error was encountered, the controller reports the error code and the line number in which the error was found.

Examples

The following are examples of the **C** command.

```
#0C                               Compile the program in buffer 0
5 lines compiled                Response, the program was compiled successfully
#9C                               Compile the program in buffer 9
```

```

?2026 in line 16      Response, Error 2026 was found in line 16
??2026              Explain error 2026.
Undefined variable name Response
##C                 Compile the programs in all buffers
                    Response
Buffer 0: 5 lines compiled The program was compiled successfully
Buffer 1: 18 lines compiled The program was compiled successfully
Buffer 2: empty           The buffer is empty
Buffer 3: 5 lines compiled The program was compiled successfully
Buffer 4: empty           The buffer is empty
Buffer 5: empty           The buffer is empty
Buffer 6: empty           The buffer is empty
Buffer 7: empty           The buffer is empty
Buffer 8: empty           The buffer is empty
Buffer 9: ?2026 in line 16 Error 2026 was found in line 16

```

6.3.5.2 X

Description

The **X** (Execute) command starts a program in a specific buffer, and can be executed in any program state except the **Run** state.

Syntax

#buffer_numberX[line_number]

Arguments

| | |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number qualifier in the command specifies the buffer, a number between 0 and 16. |
| <i>line_number</i> | Optional, line_number can be a line number or a label. Execution starts from the specified line. If line_number is omitted, the program starts from the first line. |

Comments

buffer_number must specify one buffer only.

If the state of the program is **Not Compiled**, the controller first compiles the program and then starts it. If an error is encountered during compilation, the program does not start.

If the state of the program is **Suspended**, the X command resumes the program execution. In this case the command must not contain **line_number** because upon execution the program resumes from the point where the it was suspended.

Example

```

#1X                 Execute the program in buffer 1
?1                  Query status of buffer 1
Buffer 1: 18 lines, running at line 7 Response

```

6.3.5.3 S/SR

Description

The **S/SR** Commands are used for terminating program execution:

- > **S** - Stop
The **S** command terminates program execution in a buffer or the execution of all programs in all buffers.
- > **SR** - Stop and Reset
The **SR** (Stop and Reset) command terminates program execution in a buffer or the execution of all programs in all buffers, and resets the buffer or all buffers to the **Not Compiled** state. The command provides the de-compile function, which is useful if the program contains autoroutines that are ready to start when the buffer is in the **Compiled** state.

Syntax

#buffer_number{S|SR}

Arguments

buffer_number

buffer_number specifies the buffer, a number between 0 and 16; or you can use the pound (#) to designate all buffers.

Comments

If **buffer_number** is omitted, this will stop, or stop and reset all programs in all buffers, or you can use the # character as the **buffer_number**, for example, **##S**, which will do the same.

Program termination commands must not include **line_numbers**.

The **S** command can be issued in any program state.



The issuance of the **SR** command effectively prevents the activation of autoroutines.

Example

```
#1S           Terminate the program in buffer 1
?1           Query status of buffer 1
Buffer 1: 18 lines, terminated in line 7      Response
#1SR        Reset the program in buffer 1
?1           Query status of buffer 1
Buffer 1: 18 lines, not compiled             Response
##SR       Reset all programs in all buffers
```

6.3.5.4 P

Description

The **P** (Pause) command suspends program execution in a buffer.

Syntax

#buffer_numberP

Arguments

buffer_ number

buffer_number specifies the buffer, a number between 0 and 16; or you can use the pound character, #, to designate all buffers.

Comments

Generally **buffer_number** refers to one buffer only. The # character may be used instead of a buffer number, for example, **##P**, in which case the execution of all programs in all buffers is suspended.

Pause commands must not include **line_numbers**.

Pause commands are allowed in any program state, but in all states other than **Run** the command has no effect.

If the program is in the **Suspended** state, the **X** command resumes execution. The **S/SR** command transfers the buffer to the **Compiled** state. The **S/SR** command transfers the buffer to the **Not Compiled** state.

Example

```
#1P          Suspend the program in buffer 1
?1          Query status of buffer 1
Buffer 1: 18 lines, suspended in line 7      Response
#1SR       Reset the program in buffer 1
?1          Query status of buffer 1
Buffer 1: 18 lines, not compiled            Response
##P       Suspend all programs in all buffers
```

6.3.6 Debug Commands

The following debug commands are supported:

- > **XS** - Execute one program line
- > **XD** - Execute program in debug mode
- > **BS** - Set breakpoint at specified line
- > **BR** - Reset breakpoint

6.3.6.1 XS

Description

The **XS** (Execute one step) command executes one program line.

Syntax

#buffer_numberXSline_number

Arguments

| | |
|----------------------|---------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number specifies the buffer, a number between 0 and 16. |
| <i>line_number</i> | line_number gives the line to be executed, it can be a line number or a label. |

Comments

The **buffer_number** qualifier in the command must specify one buffer only.

After executing the specified **line_number**, the buffer automatically enters the **Suspended** state.

6.3.6.2 XD

Description

The **XD** (Execute in Debug mode) command executes the program up to the next breakpoint (see [BS](#)).

Syntax

#buffer_numberXD

Arguments

| | |
|----------------------|-----------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number specifies the buffer, a number between 0 and 16. |
|----------------------|-----------------------------------------------------------------------|

Comments

The **buffer_number** qualifier in the command must specify one buffer only.

The command is similar to the **X** command. The difference is that the **X** command ignores breakpoints in the program. If the program is started by the **XD** command, it will stop when it reaches a breakpoint. At the breakpoint the program transfers to the **Suspended** state and can be started again by the **X**, **XS**, or **XD** commands.

6.3.6.3 BS

Description

The **BS** (Set Breakpoint) command sets a breakpoint at the specified line.

Syntax

#buffer_numberBSline_number

Arguments

| | |
|----------------------|-----------------------------------------------------------------------------------------------------------|
| <i>buffer_number</i> | buffer_number specifies the buffer, a number between 0 and 16. |
| <i>line_number</i> | line_number specifies the line at which to set the breakpoint, it can be a line number or a label. |

Comments

The **buffer_number** qualifier in the command must specify one buffer only.

Any number of breakpoints can be set in a program. For breakpoints to be active, the program must be started with the **XD** command.



In a program listing, the lines with breakpoints are indicated by an asterisk.

6.3.6.4 BR

Description

The **BR** (Reset Breakpoint) command resets the breakpoint at the specified line or all breakpoints.

Syntax

#buffer_numberBR[line_number]

Arguments

| | |
|----------------------|------------------------------------------------------------------------------------------------------------------------------|
| buffer_number | buffer_number specifies the buffer, a number between 0 and 16; or you can use the pound (#) to designate all buffers. |
| line_number | Optional, if included, the command resets one breakpoint at this line. line_number can be a line number or a label. |

Comments

The **buffer_number** qualifier in the command must specify one buffer only.

If **line_number** is omitted, the command resets all breakpoints in the buffer.

If the buffer qualifier is specified as #, for example, **##BR**, and **line_number** is omitted, the command resets all breakpoints in all buffers.

Example

```
#0L                                     List buffer 0
1: MovePTP:                             Resonse
2: VEL(0) = 20000
3: ptp X, 4000
4: till ^MST(0).#MOVE
5: stop
#0BS3                                    Set breakpoint at line 3
#0L                                     List buffer 0 (note the
                                        asterisk indicating the
                                        breakpoint)
1: MovePTP:                             Response
2: VEL(0) = 20000
3: *ptp X, 4000
4: till ^MST(0).#MOVE
5: stop
#0XD                                     Execute the program in
                                        debug mode in buffer 0
?0                                       Query buffer 0 state
```

```

Buffer 0: 5 lines, suspended in line 3      Response
#0XS                                       Execute line 3 of buffer 0
?0                                         Query buffer 0 state
Buffer 0: 5 lines, suspended in line 4     Response
#0XD                                       Execute the rest of the
?0                                         program in buffer 0
?0                                         Query buffer 0 state
Buffer 0: 5 lines, terminated in line 5    Response

```

6.4 System Commands

System commands provide you with information contained in the system.

6.4.1 SI

Description

The **SI** (System Information) command returns System Information about the SPiiPlus controller including serial number, firmware version, configuration, name and SP programs.

Syntax

#SI

Arguments

None

Example

```

#SI
Network System Name:      140
Controller Firmware Version: 1.95.00.00
Controller Serial Number: NTM000000A
Controller Part Number:  SP+NTM-080000001NNN
Hardware:
  MPU board:              Nexcom EBC220 500MHz
  MPU board ID:           5
  MPU number:             DOM4F00010462
  EtherCAT Master:        N/A
  Master Shift:           Enabled
  Ethernet Adapter:      RealTek RTL8139
    ID:                   3
    IP Address:           10.0.0.140
    MAC Address:          00 10 F3 0D B2 23
  EtherCAT Adapter:      RealTek RTL8139
    ID:                   3
    MAC Address:          00 50 C2 88 91 4A
Axes:
  Dummy:                 none
  DC Brush:              0,1,2,3,4,5,6,7
  DC Brushless:          0,1,2,3,4,5,6,7
  P/D Stepper:           8,9,10,11,12,13,14,15
  Linear drives:         none

```

```

PWM drives:                4,5,6,7
Digital Current Loop:      4,5,6,7
Integrated drives:        4,5,6,7
    Axis (4):              4.0A continuous/5.0A peak
    Axis (5):              4.0A continuous/5.0A peak
    Axis (6):              4.0A continuous/5.0A peak
    Axis (7):              4.0A continuous/5.0A peak
Remote HSSI drives:       0,1,2,3,4,5,6,7
Dual loop:                0,1,2,3,4,5,6,7
Position Event Generation (PEG):
    PEG pulse:            0,1,2,4,5,6
    PEG states:           0,1,2,4,5,6
Options:
    Total Number of Axes: 0
    SIN-COS Encoders:     0
    Input Shaping:        No
    SPiiPlus PLC:         No
    Axes with Customized Servo Algorithms: 0
    Customized Servo Algorithms Mask: 0x0000
    Non-ACS Servo Axes:   0
    Non-ACS Stepper Axes: 0
    Non-ACS I/O Nodes:   0
Network Unit 0:
    ID:                   0
    DIP:                  0
    Part Number:          NT-LT-8
    Vendor ID:            0x00000540
    Product ID:           0x01020000
    Revision:             1
    Serial Number:        16
    HW ID:                0x00000064
    FPGA version:         0x0000001A
Options:
    SIN-COS Encoders:    0
    Motor Type Limitations: None
Axes Assignment:          0,1,2,3,4,5,6,7
Inputs/Outputs Assignment:
    Digital inputs (IN):  0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7
    Digital outputs (OUT): 0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7
    Analog inputs (AIN):  0,1,2,3
    Analog outputs (AOUT): 0,1,2,3
    HSSI channels:        4
    Ext. inputs (EXTIN):  0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
    Ext. outputs (EXTOUT): 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Integrated Component "SPiiPlus DC-LT-8":
    Type:                 Controller (11)
    Address:              0x007
    Subsystems:           2
    Production date:      04/04/11
    HW revision:          A

```

```

S/N:                3N000016
Integrated Component "PSM3U-48V-0.7kW":
  Type:              Power supply (7)
  Address:           0x005
  Subsystems:       1
  Voltage:           48V - 48V
  Power:             700W
  Production date:   08/12/10
  HW revision:      6
  S/N:              66
Integrated Component "DDM3U-4-60V-4/5A":
  Type:              PWM drive (5)
  Address:           0x002
  Subsystems:       4
  Axes:             4,5,6,7
    Drive 0: Axis 4
    Drive 2: Axis 5
    Drive 1: Axis 6
    Drive 3: Axis 7
  Voltage:           60V - 60V
  Nominal current:   4.000000A
  Peak current:      5.000000A
  RMS protection Time: 3476.000000
  Production date:   27/10/10
  HW revision:      10
  S/N:              27
Network Unit 1:
  ID:                2
  DIP:              63
  Part Number:      PDMnt-4-08-08-00-00
  Vendor ID:        0x00000540
  Product ID:       0x02040000
  Revision:         0
  Serial Number:    0
  Options:
    SIN-COS Encoders: 0
    Motor Type Limitations: None
  Axes Assignment:  8,9,10,11
  Inputs/Outputs Assignment:
    Digital inputs (IN): 1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7
    Digital outputs (OUT): 1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7
    Analog inputs (AIN): none
    Analog outputs (AOUT): none
Integrated Component "PDM-4-8-8":
  Type:              Single-Slot Unit (14)
  Address:           0x207
  Subsystems:       1
  Axes:             8,9,10,11
    Drive 0: Axis 8
    Drive 1: Axis 9

```

```

    Drive 2:  Axis 10
    Drive 3:  Axis 11
  Production date:    01/01/10
  HW revision:       0
  S/N:              0
Network Unit 2:
  ID:               3
  DIP:             7
  Vendor ID:       0x00000540
  Product ID:     0x02040000
  Revision:       0
  Serial Number:  0
  Options:
    SIN-COS Encoders: 0
    Motor Type Limitations: None
  Axes Assignment:  12,13,14,15
  Inputs/Outputs Assignment:
    Digital inputs (IN):  2.0,2.1,2.2,2.3,2.4,2.5,2.6,2.7
    Digital outputs (OUT): 2.0,2.1,2.2,2.3,2.4,2.5,2.6,2.7
    Analog inputs (AIN):  none
    Analog outputs (AOUT): none
  Integrated Component "PDM-4-8-8":
    Type:            Single-Slot Unit (14)
    Address:        0x307
    Subsystems:    1
    Axes:          12,13,14,15
      Drive 0:  Axis 12
      Drive 1:  Axis 13
      Drive 2:  Axis 14
      Drive 3:  Axis 15
    Production date:    01/01/10
    HW revision:       0
    S/N:              0
SP0 Program Info:
  Monitor version:1
  Creation Date: Sun Apr 03 08:26:19 2011
  Saving Tool: SPiiPlus NT Servo Application File Generator v.6.83.07.00
  SPiiPlus NT Servo Processor Program.
  Date=      June 14th 2010
  Version=   1.0
  Firmware=  1.0
  ACS Motion Control Ltd.,
  Control and Applications Development,
  Copyright (c) 2010. All Rights Reserved.
SP1 Program Info:
  Monitor version:1
  Creation Date: Sun Apr 03 08:26:19 2011
  Saving Tool: SPiiPlus NT Servo Application File Generator v.6.83.07.00
  SPiiPlus NT Servo Processor Program.

```

```
Date=      June 14th 2010
Version=   1.0
Firmware=  1.0
ACS Motion Control Ltd.,
Control and Applications Development,
Copyright (c) 2010. All Rights Reserved.
SP2 Program Info:
Monitor version:ffffffff
Default Servo Processor Info.
SP3 Program Info:
Monitor version:ffffffff
Default Servo Processor Info.
```

6.4.2 SIR

Description

The **SIR** (System Information Report) command provides information about the controller.

Syntax

#SIR/Section|ALL/Key|ALL/

Arguments

There are, as a minimum, six Sections:

> **Hardware**

This section contains information about the controller's hardware. It has the following keys:

> **Model**

The Model ID for the controller card (hardware prefix). It is a three digit number that can be:

- 001 SPiiPlus DDM-4
- 020 SPiiPlus CM
- 030 SPiiPlus SA
- 040 SPiiPlus 3U-4
- 041 SPiiPlus 3U-8
- 042 SPiiPlus 3U-DDM
- 043 SPiiPlus M
- 044 SPiiPlus M(A)
- 050 SPiiPlus-LF
- 060 SPiiPlus NT

> **FM**

The Firmware version number.

> **Platform**

The controller card type ID, (first two numbers of hardware prefix).

> **SN**

The controller serial number.

> **PN**

The controller part number.

- > **MPU**
A number identifying the controller MPU, which can be:
 - 0) Unknown
 - 1) RTD 686GX-233MHz
 - 2) Sensoray 301-133MHz
 - 3) Netcom CM589/CM585-300MHz
 - 4) Kontron MOPS6-266MHz
 - 5) Nexcom EBC220-500MHz
- > **MPUN**
The MPU serial number.
- > **PAL**
The controller PAL version.
- > **Controller_Version**
Card version for the controller.
- > **SP**
Number of Servo-Processor units in the controller.
- > **Master_Shift**
Enabled - master shift is disabled
Disabled - master shift is enabled
- > **Options**
This section contains information about the controller's options. It has the following keys:
 - > **Total NumberOf Axes**
Maximum number of allowed axes.
 - > **Sin Cos Encoders**
Maximum number of allowed SIN-COS encoders.
 - > **Input Shaping**
Indicates if Input Shaping is allowed or not:
Yes - Input Shaping is allowed
No - Input Shaping is not allowed
 - > **Sin Cos Encoders**
Maximum number of allowed SIN-COS encoders.
 - > **Axes With Customized Servo Algorithms**
Maximum number of allowed axes to be used with customized servo algorithms.
 - > **Customized Servo Algorithms Mask**
A 4 hexadecimal digits (starts with 0x) that serves as a mask of allowed customized servo algorithms.
 - > **Non ACS Servo Axes**
Maximum number of allowed Non-ACS Servo axes that can be used.
 - > **Non ACS Stepper Axes**
Maximum number of allowed Non-ACS Stepper axes that can be used.
 - > **Non ACS IO Nodes**
Maximum number of allowed Non-ACS I/O Nodes that can be used.

> **Network**

This section contains information about the controller's Ethernet channels. It contains the following keys:

> **NIC1**

Code for the type of the first network adapter (the same codes are used for the second NIC if one exists, see NIC2 below) which can be:

- 000> Not present
- 001> NE2000 compatible Ethernet card
- 002> Intel 82559 PCI Ethernet card
- 003> RealTek RTL8139 PCI Ethernet card

> **NIC1_IP**

Number for the first NIC IP address.

> **NIC1_MAC**

12 hexadecimal digits providing the MAC address for the first NIC



This number is fictitious in the Simulator.

> **NIC2**

Code for the type of the second network adapter if it exists. Values are the same as those for **NIC1**, otherwise it is zero.

> **NIC2_IP**

Number for the second NIC IP address, if the second network adapter exists, otherwise it is zero.

> **NIC2_MAC**

12 hexadecimal digits providing the MAC address for the second NIC if it exists, otherwise it is zero. As for **NIC1_MAC**, this information is fictitious in Simulator.

> **Axes_support**

This section contains information about the features that each axis has. It has the following keys:

> **Dummy**

A list of dummy axes numbers, separated by commas.

> **DC_Brush**

All axes that support DC brush motors.

> **DC_Brushless**

All axes that support DC Brushless motors.

> **PD_Stepper**

All axes that support P/D Stepper motors.

> **LDM3U**

All axes that are controlled by an internal LDM3U drive.

> **Digital_Current_Loop**

All axes that support a drive with digital current loop.

- > **PWM**
All axes controlled by an internal PWM drive.
- > **Integrated**
All axes controlled by an Integrated drive (PWM or LDM).
- > **HSSI_Drive**
All axes that support an HSSI drive.
- > **Dual_Loop**
All axes that support Dual Loop control.
- > **PEG_Pulse**
All axes that support the PEG Pulse feature.
- > **PEG_State**
All axes that support the PEG State feature.
- > **UNIT#**
There is a UNIT section for each unit in the system, they are numbered from 0 up to the number of units minus 1, for example, **UNIT0** is the first unit in the system. Each UNIT section contains information about the unit. The keys are as follows:
 - > **ID**
The ID of the unit.
 - > **DIP**
The DIP switch of the unit.
 - > **Network Axes**
The axes indices, separated by commas, of all axes allocated to the unit.
 - > **Digital Inputs**
All digital input variable indices, separated by commas, that are allocated to the unit.
 - > **Digital Outputs**
All digital output variable indices, separated by commas, that are allocated to the unit.
 - > **Analog Inputs**
All analog input variable indices, separated by commas, that are allocated to the unit.
 - > **AnalogOutputs**
All analog output variable indices, separated by commas, that are allocated to the unit.
 - > **HSSIVariables**
All HSSI variable indices (input/output pairs), separated by commas, that are allocated to the unit.
 - > **HSSI Channels**
All HSSI channel indices, separated by commas, for the HSSI variables that are allocated for the unit.



Element N in this list corresponds to element N in the **HSSIRegisters** list.

- > **HSSI Registers**
List of HSSI registers for the HSSI variables, separated by commas, that are allocated for the unit.
- > **AXIS#**
There is an AXIS section for each axis in the system, they are numbered from 0 up to the total number of axes in the system minus 1, for example, **AXIS0** is the first axis in the system. Each AXIS section contains information about the axis. The keys are as follows:
 - > **Current Nominal**
The nominal current, in amperes, of the axis.
 - > **Current Peak**
The peak current, in amperes, of the axis.
 - > **XRM Smax**
Maximum nominal current, in % of peak, of the axis.
 - > **XRMST max**
Maximum time constant for RMS protection.
 - > **Drive Interface.**
Interface numbers for the drive are:
 1. PWM
 2. External ± 10
 3. LDM3U
 4. ED2
 5. Network
 6. Digital LDM3U
 - > **Max Command Current**
Maximum command for the drive.
 - > **Serial Number**
Serial number for the axis drive. If the drive has no serial number, nothing is displayed.

Example 1

```
#SIR/Options/LearningBoost/  
[Options]  
LearningBoost = Yes
```

Example 2

```
#SIR/Hardware/FW/  
[Hardware]  
FW = 3.11.64.00
```

Example 3

```
#SIR/ALL/ALL/  
[Hardware]  
Model = 60  
FW = 1.95.03.00  
SN = 3N000021A  
PN = SP+NT  
MPU = 5  
MPUN = DOMA400088768  
SP = 3  
Master_Shift = Disabled  
[Network]  
NIC1 = 3  
NIC1_IP = -2097151990  
NIC1_MAC = 0010F31A09E3  
NIC2 = 0  
NIC2_IP = 0  
NIC2_MAC = 000000000000  
[Options]  
TotalNumberOfAxes = 32  
SinCosEncoders = 32  
InputShaping = Yes  
AxesWithCustomizedServoAlgorithms = 0  
CustomizedServoAlgorithmsMask = 0x0000  
NonACSServoAxes = 32  
NonACSStepperAxes = 32  
NonACSIONodes = 32  
[Axes_support]  
Dummy = 0  
DC_Brush = 18446744069414584575  
DC_Brushless = 18446744069414584575  
PD_Stepper = 18446744069414584320  
LDM3U = 0  
Digital_Current_Loop = 0  
PWM = 0  
Integrated = 0  
HSSI_Drive = 18446744069414584575  
Dual_Loop = 18446744069414584575  
PEG_Pulse = 18446744069414584439  
PEG_State = 18446744069414584439  
[UNIT0]  
ID = 0  
DIP = 0  
NetworkAxes = 0,1,2,3,4,5,6,7  
DigitalInputs = 0  
DigitalOutputs = 0  
AnalogInputs = 0,1,2,3  
AnalogOutputs = 0,1,2,3
```

```

HSSIVariables = 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
HSSIChannels = 0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3
HSSIRegisters = 0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3
[UNIT1]
ID = 2
DIP = 0
NetworkAxes = none
DigitalInputs = none
DigitalOutputs = none
AnalogInputs = none
AnalogOutputs = none
HSSIVariables = none
HSSIChannels = none
HSSIRegisters = none
[AXIS0]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS1]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS2]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS3]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
  
```

```

SerialNumber =
[AXIS4]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS5]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS6]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS7]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS8]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS9]
CurrentNominal = 0.000000
CurrentPeak = 0.000000

```

```

XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS10]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS11]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS12]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS13]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS14]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
  
```



```

MaxCommandCurrent = 5242
SerialNumber =
[AXIS15]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS16]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS17]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS18]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS19]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS20]
CurrentNominal = 0.000000

```

```

CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS21]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS22]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS23]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS24]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS25]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0

```

```

DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS26]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS27]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS28]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS29]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS30]
CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
[AXIS31]

```

```

CurrentNominal = 0.000000
CurrentPeak = 0.000000
XRMSmax = 50.000000
XRMSTmax = 3230.000000
Voltage = 0
DriveInterface = 2
MaxCommandCurrent = 5242
SerialNumber =
:

```

6.4.3 MEMORY

Description

Retrieves RAM availability.

Syntax

#MEMORY

Arguments

None.

Return Value

Available RAM.

Example

Up to Version 3.11

```

#MEMORY
There is      15 percent(s) of memory in use.
There are 114.83 total MBytes of physical memory.
There are   96.91 free  MBytes of physical memory.

```

From Version 3.12

```

#MEMORY
Memory Allocation for buffers:
for all buffers code: 14832 kB
for all buffers source: 6416 kB
for all buffers local variables: 8704 kB
for global variables: 1024 kB

```

6.4.4 IR

Description

The **IR** (Integrity Report) command activates integrity validation and provides a report of current integrity state. The report displays a list of files. Each list entry displays a file name, expected file size and checksum of the file and actual file size and checksum.



It takes some time to compile the Integrity Report, in order to avoid a Timeout error:

1. Right-click the controller in the Workspace Tree and select **Properties**.
2. Increase the **Connection Timeout** to 10,000 ms.
3. Click **Connect**.

Syntax

#IR

Arguments

None

Comments

If any integrity problem is detected, the command raises fault bit **S_FAULT.#INTGR**.

Example

```
#IR
      Size
      Registered  Actual      Checksum
      Registered  Actual  Registered  Actual
c:\
  SB1218PC.frm    001E2870  001E2870    03672ED6   03672ED6
  SB1218PC.bin    0000812A  0000812A    DDC2E529   DDC2E529
c:\sb4\dsp\
  dsp.###         0004FF18  0004FF18    61F6BA11   61F6BA11
  dsp.##1         0003E11B  0003E11B    783AE65B   783AE65B
  adj0.$$$        0000122B  0000122B    69003B3B   69003B3B
  adj1.$$$        00001A99  00001A99    055D4E11   055D4E11
  adj2.$$$        000019DA  000019DA    BC928A33   BC928A33
  adj3.$$$        00001A9B  00001A9B    4403628F   4403628F
  adj4.$$$        0000197E  0000197E    DE12BDD7   DE12BDD7
  adj5.$$$        00001229  00001229    A974E209   A974E209
  adj6.$$$        00001229  00001229    A977E20C   A977E20C
  adj7.$$$        0000122B  0000122B    61EE382F   61EE382F
c:\sb4\startup\
  par.$$$         000001C9  000001C9    D8A0220A   D8A0220A
  par0.$$$        00000A16  00000A16    B2783961   B2783961
  par1.$$$        00000A13  00000A13    5E2EAD92   5E2EAD92
  par2.$$$        00000A14  00000A14    5A972E6D   5A972E6D
  par3.$$$        00000A13  00000A13    B25EDAD0   B25EDAD0
  par4.$$$        00000A13  00000A13    CD6E01E2   CD6E01E2
  par5.$$$        00000A13  00000A13    F6873205   F6873205
  par6.$$$        00000A15  00000A15    39E27520   39E27520
  par7.$$$        00000A15  00000A15    77CA503D   77CA503D
  par8.$$$        00000A13  00000A13    43A67043   43A67043
  par9.$$$        00000A13  00000A13    67BC915E   67BC915E
  par10.$$$       00000A89  00000A89    454F04F9   454F04F9
  par11.$$$       00000A89  00000A89    5A67281F   5A67281F
```

```

par12.$$$      00000A89    00000A89    6F7F4B45    6F7F4B45
par13.$$$      00000A89    00000A89    84976E6B    84976E6B
par14.$$$      00000A89    00000A89    99AF9191    99AF9191
par15.$$$      00000A89    00000A89    AEC7B4B7    AEC7B4B7
par16.$$$      00000A89    00000A89    C3DFD7DD    C3DFD7DD
par17.$$$      00000A89    00000A89    D8F7FB03    D8F7FB03
par18.$$$      00000A89    00000A89    EE101E29    EE101E29
par19.$$$      00000A89    00000A89    0328414F    0328414F
par20.$$$      00000A89    00000A89    6B641D1C    6B641D1C
par21.$$$      00000A89    00000A89    807C4042    807C4042
par22.$$$      00000A89    00000A89    95946368    95946368
par23.$$$      00000A89    00000A89    AAAC868E    AAAC868E
par24.$$$      00000DF1    00000DF1    26E26061    26E26061
par25.$$$      00000DF1    00000DF1    4B01777C    4B01777C
par26.$$$      00000DF1    00000DF1    6F208E97    6F208E97
par27.$$$      00000DF1    00000DF1    933FA5B2    933FA5B2
par28.$$$      00000DF1    00000DF1    B75EBCCD    B75EBCCD
par29.$$$      00000DF1    00000DF1    DB7DD3E8    DB7DD3E8
par30.$$$      00000DF1    00000DF1    B18A230C    B18A230C
par31.$$$      00000DF1    00000DF1    D5A93A27    D5A93A27
c:\sb4\user\
I              000001A0    000001A0    414453BB    414453BB
V              00000330    00000330    414453BC    414453BC
X_FILE        00000070    00000070    82665362    82665362
PMAP         00000340    00000340    F29BADD0    F29BADD0
ONE          00000FC0    00000FC0    4144573F    4144573F
TWO         00002F00    00002F00    414453A7    414453A7
X_ERR       00000040    00000040    42A2135C    42A2135C
System Integrity is OK

```

6.4.5 U

Description

The **U** (Usage) command is used for monitoring MPU usage. It returns the maximum, average, and minimum values as a percent.

Syntax

#U

Arguments

None

Comments

The controller continuously measures the time taken by real-time tasks. When the **U** command is received, the controller analyzes the measured times during the last 50 controller cycles and calculates minimal, maximal and average time. The results are reported in percents.

6.4.6 TD

Description

The **TD** command returns the names of all user-defined variables and arrays that are in the controller flash memory.

Syntax

#TD

Arguments

None

6.4.7 SC

Description

The **SC** (Safety Control) command reports the current safety system configuration.

The controller response includes the following:

- > active safety groups
- > the configuration of each fault for each motor

Syntax

#SC

Arguments

None

Example

```
#SC
Bit Name      Fault Description      0  1  2  4  5  6  8  9  10 11 12 13 14 15
0 #RL        Hardware Right Limit  K  K  K  K  K  K  K  K  K  K  K  K  K  K
1 #LL        Hardware Left Limit  K  K  K  K  K  K  K  K  K  K  K  K  K  K
2 #NT        Network Error        D  D  D  -  D  D  D  D  D  D  D  D  D  D
4 #HOT       Motor Overheat            -  -  -  -  -  -  -  -  -  -  -  -  -  -
5 #SRL       Software Right Limit    K  K  K  K  K  K  K  K  K  K  K  K  K  K
6 #SLL       Software Left Limit     K  K  K  K  K  K  K  K  K  K  K  K  K  K
7 #ENCNC     Encoder Not Connected        D  D  D  D  D  D  D  D  D  D  D  D  D  D
8 #ENC2NC   Encoder 2 Not Connected      -  -  -  -  -  -  -  -  -  -  -  -  -  -
9 #DRIVE     Drive Fault                  D  D  D  D  D  D  D  D  D  D  D  D  D  D
10 #ENC      Encoder Error                D  D  D  D  D  D  D  D  D  D  D  D  D  D
11 #ENC2     Encoder 2 Error              -  -  -  -  -  -  -  -  -  -  -  -  -  -
12 #PE       Position Error                -  -  -  -  -  -  -  -  -  -  -  -  -  -
13 #CPE      Critical Position Error      D  D  D  D  D  D  D  D  D  D  D  D  D  D
14 #VL       Velocity Limit                K  K  K  K  K  K  K  K  K  K  K  K  K  K
15 #AL       Acceleration Limit         -  -  -  -  -  -  -  -  -  -  -  -  -  -
16 #CL       Overcurrent                  D  D  D  D  D  D  D  D  D  D  D  D  D  D
17 #SP       Servo Processor Alarm        D  D  D  D  D  D  D  D  D  D  D  D  D  D
20 #HSSINC   HSSI Not Connected          -  -  -  -  -  -  -  -  -  -  -  -  -  -
25 #PROG     Program Error                K  K  K  K  K  K  K  K  K  K  K  K  K  K
26 #MEM      Memory Overflow              K  K  K  K  K  K  K  K  K  K  K  K  K  K
27 #TIME     MPU Overuse                   -  -  -  -  -  -  -  -  -  -  -  -  -  -
28 #ES       Hardware Emergency Stop      D  D  D  -  D  D  D  D  D  D  D  D  D  D
29 #INT      Servo Interrupt              D  D  D  D  D  D  D  D  D  D  D  D  D  D
30 #INTGR    File Integrity                -  -  -  -  -  -  -  -  -  -  -  -  -  -
31 #FAILURE  Component Failure            D  D  D  D  D  D  D  D  D  D  D  D  D  D

Legend:
-      Fault Detection Disabled
Blank  No Default Response
K      Kill Motion Response
D      Disable Axis Response
KD     Kill Motion Followed by Disable Axis Response
+      Generalized Fault

Bit Name      Fault Description      16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

| | | | | | | | | | | | | | | | | | |
|----|----------|-------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | #RL | Hardware Right Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 1 | #LL | Hardware Left Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 2 | #NT | Network Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 4 | #HOT | Motor Overheat | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5 | #SRL | Software Right Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 6 | #SLL | Software Left Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 7 | #ENCNC | Encoder Not Connected | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 8 | #ENC2NC | Encoder 2 Not Connected | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 9 | #DRIVE | Drive Fault | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 10 | #ENC | Encoder Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 11 | #ENC2 | Encoder 2 Error | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 12 | #PE | Position Error | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 13 | #CPE | Critical Position Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 14 | #VL | Velocity Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 15 | #AL | Acceleration Limit | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | #CL | Overcurrent | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 17 | #SP | Servo Processor Alarm | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 20 | #HSSINC | HSSI Not Connected | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 25 | #PROG | Program Error | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 26 | #MEM | Memory Overflow | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 27 | #TIME | MPU Overuse | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 28 | #ES | Hardware Emergency Stop | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 29 | #INT | Servo Interrupt | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 30 | #INTGR | File Integrity | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 31 | #FAILURE | Component Failure | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |

Legend:

- Fault Detection Disabled
- Blank No Default Response
- K Kill Motion Response
- D Disable Axis Response
- KD Kill Motion Followed by Disable Axis Response
- + Generalized Fault

| Bit | Name | Fault Description | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|-----|----------|-------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | #RL | Hardware Right Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 1 | #LL | Hardware Left Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 2 | #NT | Network Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 4 | #HOT | Motor Overheat | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5 | #SRL | Software Right Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 6 | #SLL | Software Left Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 7 | #ENCNC | Encoder Not Connected | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 8 | #ENC2NC | Encoder 2 Not Connected | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 9 | #DRIVE | Drive Fault | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 10 | #ENC | Encoder Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 11 | #ENC2 | Encoder 2 Error | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 12 | #PE | Position Error | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 13 | #CPE | Critical Position Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 14 | #VL | Velocity Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 15 | #AL | Acceleration Limit | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | #CL | Overcurrent | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 17 | #SP | Servo Processor Alarm | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 20 | #HSSINC | HSSI Not Connected | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 25 | #PROG | Program Error | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 26 | #MEM | Memory Overflow | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 27 | #TIME | MPU Overuse | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 28 | #ES | Hardware Emergency Stop | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 29 | #INT | Servo Interrupt | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 30 | #INTGR | File Integrity | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 31 | #FAILURE | Component Failure | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |

Legend:

- Fault Detection Disabled
- Blank No Default Response
- K Kill Motion Response
- D Disable Axis Response
- KD Kill Motion Followed by Disable Axis Response
- + Generalized Fault

| Bit | Name | Fault Description | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
|-----|---------|-------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | #RL | Hardware Right Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 1 | #LL | Hardware Left Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 2 | #NT | Network Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 4 | #HOT | Motor Overheat | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5 | #SRL | Software Right Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 6 | #SLL | Software Left Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 7 | #ENCNC | Encoder Not Connected | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 8 | #ENC2NC | Encoder 2 Not Connected | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 9 | #DRIVE | Drive Fault | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 10 | #ENC | Encoder Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 11 | #ENC2 | Encoder 2 Error | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 12 | #PE | Position Error | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | | | | | | | | | | | | | | | | | |
|----|----------|-------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | #CPE | Critical Position Error | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 14 | #VL | Velocity Limit | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 15 | #AL | Acceleration Limit | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | #CL | Overcurrent | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 17 | #SP | Servo Processor Alarm | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 20 | #HSSINC | HSSI Not Connected | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 25 | #PROG | Program Error | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 26 | #MEM | Memory Overflow | K | K | K | K | K | K | K | K | K | K | K | K | K | K | K |
| 27 | #TIME | MPU Overuse | | | | | | | | | | | | | | | |
| 28 | #ES | Hardware Emergency Stop | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 29 | #INT | Servo Interrupt | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |
| 30 | #INTGR | File Integrity | | | | | | | | | | | | | | | |
| 31 | #FAILURE | Component Failure | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |

Legend:

- Fault Detection Disabled
- Blank No Default Response
- K Kill Motion Response
- D Disable Axis Response
- KD Kill Motion Followed by Disable Axis Response
- + Generalized Fault

6.4.8 ETHERCAT

Description

The **ETHERCAT** command is used for obtaining complete information about the connected EtherCAT slaves.

The information it displays is:

- > Slave number
- > Vendor ID
- > Product ID
- > Revision
- > Serial number
- > EtherCAT physical address
- > DC support
- > Mailbox support
- > PdoIndex

Afterwards the list of network variables is listed. Each variable is described with:

- > Name (as in XML)
- > Offset inside the telegram (magic number that is used for mapping)
- > IN or OUT description
- > Data size

Syntax

#ETHERCAT

Arguments

None

Example 1

```
#ETHERCAT
EtherCAT bus scan found 4 nodes:
Node 0:
=====
Name:          Device 1 (SPiiPlus NT-LT-8-New)
Vendor ID:     0x00000540      Product ID:    0x01020000
PHYS ADDR:     0x03E9      Alias:    0x0000
PD IN:         Offset  26.0 Size  118
PD OUT:        Offset  26.0 Size  136
STATE:         OP
Node 1:
=====
Name:          Device 2 (SPiiPlus NT-LT-8-New)
Vendor ID:     0x00000540      Product ID:    0x01020000
PHYS ADDR:     0x03EA      Alias:    0x0000
PD IN:         Offset  162.0 Size  118
PD OUT:        Offset  162.0 Size  136
STATE:         OP
Node 2:
=====
Name:          Device 3 (SPiiPlus PDMnt-4-08-08-00-00-0)
Vendor ID:     0x00000540      Product ID:    0x02040000
PHYS ADDR:     0x03EB      Alias:    0x0000
PD IN:         Offset  298.0 Size   5
PD OUT:        Offset  298.0 Size  56
STATE:         OP
Node 3:
=====
Name:          Device 4 (SPiiPlus PDMnt-4-08-08-00-00-0)
Vendor ID:     0x00000540      Product ID:    0x02040000
PHYS ADDR:     0x03EC      Alias:    0x0000
PD IN:         Offset  354.0 Size   5
PD OUT:        Offset  354.0 Size  56
STATE:         OP

Network variables:
=====
Offset  Size  Dir  Name
  26    32  In   Command response1
  30    32  In   Command response2
  34    32  In   Command response3
  38    32  In   Command response4
  42    32  In   Command response5
  46    32  In   Command response6
  50    32  In   Command response7
  54    32  In   Command response8
  58    32  In   Feedback Position1
  62    32  In   Reference Position1
  66    32  In   Drive status1
```

| | | | |
|-----|----|----|---------------------|
| 70 | 32 | In | GP data 1 |
| 74 | 32 | In | Feedback Position2 |
| 78 | 32 | In | Reference Position2 |
| 82 | 32 | In | Drive status2 |
| 86 | 32 | In | GP data 2 |
| 90 | 32 | In | Feedback Position3 |
| 94 | 32 | In | Reference Position3 |
| 98 | 32 | In | Drive status3 |
| 102 | 32 | In | GP data 3 |
| 106 | 32 | In | Feedback Position4 |
| 110 | 32 | In | Reference Position4 |
| 114 | 32 | In | Drive status4 |
| 118 | 32 | In | GP data 4 |
| 122 | 16 | In | Digital inputs |
| 124 | 16 | In | FPGA status |
| 126 | 16 | In | HSSI 1 |
| 128 | 16 | In | HSSI 2 |
| 130 | 16 | In | HSSI 3 |
| 132 | 16 | In | HSSI 4 |
| 134 | 16 | In | HSSI 5 |
| 136 | 16 | In | HSSI 6 |
| 138 | 16 | In | HSSI 7 |
| 140 | 16 | In | HSSI 8 |
| 142 | 16 | In | Sync Counter |
| 162 | 32 | In | Command response1 |
| 166 | 32 | In | Command response2 |
| 170 | 32 | In | Command response3 |
| 174 | 32 | In | Command response4 |
| 178 | 32 | In | Command response5 |
| 182 | 32 | In | Command response6 |
| 186 | 32 | In | Command response7 |
| 190 | 32 | In | Command response8 |
| 194 | 32 | In | Feedback Position1 |
| 198 | 32 | In | Reference Position1 |
| 202 | 32 | In | Drive status1 |
| 206 | 32 | In | GP data 1 |
| 210 | 32 | In | Feedback Position2 |
| 214 | 32 | In | Reference Position2 |
| 218 | 32 | In | Drive status2 |
| 222 | 32 | In | GP data 2 |
| 226 | 32 | In | Feedback Position3 |
| 230 | 32 | In | Reference Position3 |
| 234 | 32 | In | Drive status3 |
| 238 | 32 | In | GP data 3 |
| 242 | 32 | In | Feedback Position4 |
| 246 | 32 | In | Reference Position4 |
| 250 | 32 | In | Drive status4 |
| 254 | 32 | In | GP data 4 |
| 258 | 16 | In | Digital inputs |
| 260 | 16 | In | FPGA status |

| | | | |
|-----|----|----|----------------|
| 262 | 16 | In | HSSI 1 |
| 264 | 16 | In | HSSI 2 |
| 266 | 16 | In | HSSI 3 |
| 268 | 16 | In | HSSI 4 |
| 270 | 16 | In | HSSI 5 |
| 272 | 16 | In | HSSI 6 |
| 274 | 16 | In | HSSI 7 |
| 276 | 16 | In | HSSI 8 |
| 278 | 16 | In | Sync Counter |
| 298 | 8 | In | LIMITS |
| 299 | 8 | In | DIGITAL_INPUTS |
| 300 | 8 | In | FAULTS |
| 301 | 8 | In | NODE_NUM |
| 302 | 8 | In | WD_COUNTER |
| 354 | 8 | In | LIMITS |
| 355 | 8 | In | DIGITAL_INPUTS |
| 356 | 8 | In | FAULTS |
| 357 | 8 | In | NODE_NUM |
| 358 | 8 | In | WD_COUNTER |
| 422 | 32 | In | DC1 |
| 426 | 32 | In | DC2 |
| 430 | 32 | In | DC3 |
| 434 | 32 | In | DC4 |
| 438 | 32 | In | DC5 |
| 442 | 32 | In | DC6 |
| 446 | 32 | In | DC7 |
| 450 | 32 | In | DC8 |
| 454 | 32 | In | DC9 |
| 458 | 32 | In | DC10 |
| 462 | 32 | In | DC11 |
| 466 | 32 | In | DC12 |
| 470 | 32 | In | DC13 |
| 474 | 32 | In | DC14 |
| 478 | 32 | In | DC15 |
| 482 | 32 | In | DC16 |
| 486 | 32 | In | DC17 |
| 490 | 32 | In | DC18 |
| 494 | 32 | In | DC19 |
| 498 | 32 | In | DC20 |
| 502 | 32 | In | DC21 |
| 506 | 32 | In | DC22 |
| 510 | 32 | In | DC23 |
| 514 | 32 | In | DC24 |
| 518 | 32 | In | DC25 |
| 522 | 32 | In | DC26 |
| 526 | 32 | In | DC27 |
| 530 | 32 | In | DC28 |
| 534 | 32 | In | DC29 |
| 538 | 32 | In | DC30 |
| 542 | 32 | In | DC31 |

```

546      32      In      DC32
550      32      In      DC33
554      32      In      DC34
558      32      In      DC35
562      32      In      DC36
566      32      In      DC37
570      32      In      DC38
574      32      In      DC39
578      32      In      DC40
594      32      In      DC1
598      32      In      DC2
602      32      In      DC3
606      32      In      DC4
610      32      In      DC5
614      32      In      DC6
618      32      In      DC7
622      32      In      DC8
626      32      In      DC9
630      32      In      DC10
634      32      In      DC11
638      32      In      DC12
642      32      In      DC13
646      32      In      DC14
650      32      In      DC15
654      32      In      DC16
658      32      In      DC17
662      32      In      DC18
666      32      In      DC19
670      32      In      DC20
674      32      In      DC21
678      32      In      DC22
682      32      In      DC23
686      32      In      DC24
690      32      In      DC25
694      32      In      DC26
698      32      In      DC27
702      32      In      DC28
706      32      In      DC29
710      32      In      DC30
714      32      In      DC31
718      32      In      DC32
722      32      In      DC33
726      32      In      DC34
730      32      In      DC35
734      32      In      DC36
738      32      In      DC37
742      32      In      DC38
746      32      In      DC39
750      32      In      DC40
  26      16      Out     Command1
  
```

| | | | |
|-----|----|-----|-------------------------|
| 28 | 16 | Out | Command2 |
| 30 | 16 | Out | Command3 |
| 32 | 16 | Out | Command4 |
| 34 | 16 | Out | Command5 |
| 36 | 16 | Out | Command6 |
| 38 | 16 | Out | Command7 |
| 40 | 16 | Out | Command8 |
| 42 | 32 | Out | Command Arg1 |
| 46 | 32 | Out | Command Arg2 |
| 50 | 32 | Out | Command Arg3 |
| 54 | 32 | Out | Command Arg4 |
| 58 | 32 | Out | Command Arg5 |
| 62 | 32 | Out | Command Arg6 |
| 66 | 32 | Out | Command Arg7 |
| 70 | 32 | Out | Command Arg8 |
| 74 | 32 | Out | Direct Command1 |
| 78 | 32 | Out | Reference Acceleration1 |
| 82 | 32 | Out | Reference Velocity1 |
| 86 | 32 | Out | Reference Position1 |
| 90 | 32 | Out | Controller status1 |
| 94 | 32 | Out | Direct Command2 |
| 98 | 32 | Out | Reference Acceleration2 |
| 102 | 32 | Out | Reference Velocity2 |
| 106 | 32 | Out | Reference Position2 |
| 110 | 32 | Out | Controller status2 |
| 114 | 32 | Out | Direct Command3 |
| 118 | 32 | Out | Reference Acceleration3 |
| 122 | 32 | Out | Reference Velocity3 |
| 126 | 32 | Out | Reference Position3 |
| 130 | 32 | Out | Controller status3 |
| 134 | 32 | Out | Direct Command4 |
| 138 | 32 | Out | Reference Acceleration4 |
| 142 | 32 | Out | Reference Velocity4 |
| 146 | 32 | Out | Reference Position4 |
| 150 | 32 | Out | Controller status4 |
| 154 | 16 | Out | analog output1 |
| 156 | 16 | Out | analog output2 |
| 158 | 16 | Out | digital output |
| 160 | 16 | Out | Sync Counter |
| 162 | 16 | Out | Command1 |
| 164 | 16 | Out | Command2 |
| 166 | 16 | Out | Command3 |
| 168 | 16 | Out | Command4 |
| 170 | 16 | Out | Command5 |
| 172 | 16 | Out | Command6 |
| 174 | 16 | Out | Command7 |
| 176 | 16 | Out | Command8 |
| 178 | 32 | Out | Command Arg1 |
| 182 | 32 | Out | Command Arg2 |
| 186 | 32 | Out | Command Arg3 |

| | | | |
|-----|----|-----|-------------------------|
| 190 | 32 | Out | Command Arg4 |
| 194 | 32 | Out | Command Arg5 |
| 198 | 32 | Out | Command Arg6 |
| 202 | 32 | Out | Command Arg7 |
| 206 | 32 | Out | Command Arg8 |
| 210 | 32 | Out | Direct Command1 |
| 214 | 32 | Out | Reference Acceleration1 |
| 218 | 32 | Out | Reference Velocity1 |
| 222 | 32 | Out | Reference Position1 |
| 226 | 32 | Out | Controller status1 |
| 230 | 32 | Out | Direct Command2 |
| 234 | 32 | Out | Reference Acceleration2 |
| 238 | 32 | Out | Reference Velocity2 |
| 242 | 32 | Out | Reference Position2 |
| 246 | 32 | Out | Controller status2 |
| 250 | 32 | Out | Direct Command3 |
| 254 | 32 | Out | Reference Acceleration3 |
| 258 | 32 | Out | Reference Velocity3 |
| 262 | 32 | Out | Reference Position3 |
| 266 | 32 | Out | Controller status3 |
| 270 | 32 | Out | Direct Command4 |
| 274 | 32 | Out | Reference Acceleration4 |
| 278 | 32 | Out | Reference Velocity4 |
| 282 | 32 | Out | Reference Position4 |
| 286 | 32 | Out | Controller status4 |
| 290 | 16 | Out | analog output1 |
| 292 | 16 | Out | analog output2 |
| 294 | 16 | Out | digital output |
| 296 | 16 | Out | Sync Counter |
| 298 | 16 | Out | PULSE_WIDTH |
| 300 | 16 | Out | INTERVAL1_1 |
| 302 | 16 | Out | INTERVAL1_2 |
| 304 | 16 | Out | INTERVAL1_3 |
| 306 | 16 | Out | INTERVAL2_1 |
| 308 | 16 | Out | INTERVAL2_2 |
| 310 | 16 | Out | INTERVAL2_3 |
| 312 | 16 | Out | INTERVAL3_1 |
| 314 | 16 | Out | INTERVAL3_2 |
| 316 | 16 | Out | INTERVAL3_3 |
| 318 | 16 | Out | INTERVAL4_1 |
| 320 | 16 | Out | INTERVAL4_2 |
| 322 | 16 | Out | INTERVAL4_3 |
| 324 | 16 | Out | PULSE_QTY1_1 |
| 326 | 16 | Out | PULSE_QTY1_2 |
| 328 | 16 | Out | PULSE_QTY1_3 |
| 330 | 16 | Out | PULSE_QTY2_1 |
| 332 | 16 | Out | PULSE_QTY2_2 |
| 334 | 16 | Out | PULSE_QTY2_3 |
| 336 | 16 | Out | PULSE_QTY3_1 |
| 338 | 16 | Out | PULSE_QTY3_2 |

| | | | |
|-----|----|-----|----------------|
| 340 | 16 | Out | PULSE_QTY3_3 |
| 342 | 16 | Out | PULSE_QTY4_1 |
| 344 | 16 | Out | PULSE_QTY4_2 |
| 346 | 16 | Out | PULSE_QTY4_3 |
| 348 | 8 | Out | ENABLE |
| 349 | 8 | Out | DIGITAL_OUTPUT |
| 350 | 8 | Out | WD_COUNTER |
| 351 | 16 | Out | SEVEN_SEG |
| 353 | 8 | Out | SPARE |
| 354 | 16 | Out | PULSE_WIDTH |
| 356 | 16 | Out | INTERVAL1_1 |
| 358 | 16 | Out | INTERVAL1_2 |
| 360 | 16 | Out | INTERVAL1_3 |
| 362 | 16 | Out | INTERVAL2_1 |
| 364 | 16 | Out | INTERVAL2_2 |
| 366 | 16 | Out | INTERVAL2_3 |
| 368 | 16 | Out | INTERVAL3_1 |
| 370 | 16 | Out | INTERVAL3_2 |
| 372 | 16 | Out | INTERVAL3_3 |
| 374 | 16 | Out | INTERVAL4_1 |
| 376 | 16 | Out | INTERVAL4_2 |
| 378 | 16 | Out | INTERVAL4_3 |
| 380 | 16 | Out | PULSE_QTY1_1 |
| 382 | 16 | Out | PULSE_QTY1_2 |
| 384 | 16 | Out | PULSE_QTY1_3 |
| 386 | 16 | Out | PULSE_QTY2_1 |
| 388 | 16 | Out | PULSE_QTY2_2 |
| 390 | 16 | Out | PULSE_QTY2_3 |
| 392 | 16 | Out | PULSE_QTY3_1 |
| 394 | 16 | Out | PULSE_QTY3_2 |
| 396 | 16 | Out | PULSE_QTY3_3 |
| 398 | 16 | Out | PULSE_QTY4_1 |
| 400 | 16 | Out | PULSE_QTY4_2 |
| 402 | 16 | Out | PULSE_QTY4_3 |
| 404 | 8 | Out | ENABLE |
| 405 | 8 | Out | DIGITAL_OUTPUT |
| 406 | 8 | Out | WD_COUNTER |
| 407 | 16 | Out | SEVEN_SEG |
| 409 | 8 | Out | SPARE |
| 422 | 32 | Out | REV_DC1 |
| 426 | 32 | Out | REV_DC2 |
| 430 | 32 | Out | REV_DC4 |
| 434 | 32 | Out | REV_DC4 |
| 438 | 32 | Out | REV_DC5 |
| 442 | 32 | Out | REV_DC6 |
| 446 | 32 | Out | REV_DC7 |
| 450 | 32 | Out | REV_DC8 |
| 454 | 32 | Out | REV_DC9 |
| 458 | 32 | Out | REV_DC10 |
| 462 | 32 | Out | REV_DC11 |


```

466      32      Out      REV_DC12
470      32      Out      REV_DC13
474      32      Out      REV_DC14
478      32      Out      REV_DC15
482      32      Out      REV_DC16
486      32      Out      REV_DC17
490      32      Out      REV_DC18
494      32      Out      REV_DC19
498      32      Out      REV_DC20
502      32      Out      REV_DC21
506      32      Out      REV_DC22
510      32      Out      REV_DC23
514      32      Out      REV_DC24
518      32      Out      REV_DC25
522      32      Out      REV_DC26
526      32      Out      REV_DC27
530      32      Out      REV_DC28
534      32      Out      REV_DC29
538      32      Out      REV_DC30
542      32      Out      REV_DC31
546      32      Out      REV_DC32
550      32      Out      REV_DC33
554      32      Out      REV_DC34
558      32      Out      REV_DC35
562      32      Out      REV_DC36
566      32      Out      REV_DC37
570      32      Out      REV_DC38
574      32      Out      REV_DC39
578      32      Out      REV_DC40
594      32      Out      REV_DC1
598      32      Out      REV_DC2
602      32      Out      REV_DC4
606      32      Out      REV_DC4
610      32      Out      REV_DC5
614      32      Out      REV_DC6
618      32      Out      REV_DC7
622      32      Out      REV_DC8
626      32      Out      REV_DC9
630      32      Out      REV_DC10
634      32      Out      REV_DC11
638      32      Out      REV_DC12
642      32      Out      REV_DC13
646      32      Out      REV_DC14
650      32      Out      REV_DC15
654      32      Out      REV_DC16
658      32      Out      REV_DC17
662      32      Out      REV_DC18
666      32      Out      REV_DC19
670      32      Out      REV_DC20
674      32      Out      REV_DC21
  
```

```

678      32   Out   REV_DC22
682      32   Out   REV_DC23
686      32   Out   REV_DC24
690      32   Out   REV_DC25
694      32   Out   REV_DC26
698      32   Out   REV_DC27
702      32   Out   REV_DC28
706      32   Out   REV_DC29
710      32   Out   REV_DC30
714      32   Out   REV_DC31
718      32   Out   REV_DC32
722      32   Out   REV_DC33
726      32   Out   REV_DC34
730      32   Out   REV_DC35
734      32   Out   REV_DC36
738      32   Out   REV_DC37
742      32   Out   REV_DC38
746      32   Out   REV_DC39
750      32   Out   REV_DC40

```

Example 2 from a SPiiPlusES

```

Network variables:
=====
Offset  Size  Dir  PdoIndex  Name
72      32   In   0x1600    Command response1
76      32   In   0x1600    Command response2
80      32   In   0x1600    Command response3
84      32   In   0x1600    Command response4
88      32   In   0x1600    Feedback Position1
92      32   In   0x1600    Reference Position1
96      32   In   0x1600    Drive status1
100     32   In   0x1600    GP data 1A
104     32   In   0x1600    Feedback Position2
108     32   In   0x1600    Reference Position2
112     32   In   0x1600    Drive status2
116     32   In   0x1600    GP data 2A
120     16   In   0x1600    Drive Output1
122     16   In   0x1600    Drive Output2
124     16   In   0x1600    Digital inputs
126     16   In   0x1600    FPGA status

```

6.4.9 ECMAPREP

Description

The **ECMAPREP** command displays a report of all variables mapped using the **ECIN**, **ECOUT**, **ECEXTIN**, and **ECEXTOUT** functions. Bit notation is available for all variables..

Syntax

#ECMAPREP

Example 1

```
#ECMAPREP
Input 1:
=====
Variable I2, at 0x40620334 (integer)
Array Length 0
Data Length 1
EC Offset 22
Limits 0

Input 2:
=====
Variable I0, at 0x40620354 (integer)
Array Length 0
Data Length 1
EC Offset 16
Limits 0

Output 1:
=====
Variable I1, at 0x40620344 (integer)
Array Length 0
Data Length 2
EC Offset 16
Limits 0
Read Only = true
```

Example 2

```
#ECMAPREP
SPiPlusES inputs/outputs 1:
=====
Variable I0, at 0x42A8CFA4 (integer)
PDO Index 0x1A01
Object Dictionary Index 0x6064
Object Dictionary SubIndex 0x00
Data Length 4
Read Only = true
```

6.4.10 CC

Description

The **CC** command returns data on the existing communication channels.

Syntax

#CC

Example

```
#CC
Channel Type      Mode
1      Serial      Command Rate:115200 0N 1
2      Serial      Command Rate:115200 0N 1
6      TCP/IP      ( 701) Peer:None
7      TCP/IP      ( 701) Peer:10.0.0.52
8      TCP/IP      ( 701) Peer:10.0.0.65
9      TCP/IP      ( 701) Peer:10.0.0.58
10     UDP          ( 700) Peer:N/A
36     TCP/IP      ( 701) Peer:None
37     TCP/IP      ( 701) Peer:None
38     TCP/IP      ( 701) Peer:None
39     TCP/IP      ( 701) Peer:None
12     PCI bus     Command
16     TCP/IP      MODBUS Slave Connection:network Peer:None
```

6.4.11 PLC

Description



This command is valid only if SpiiPlus PLC is running in the system.

The **PLC** command provides some very important information about the SpiiPlus PLC co-existence inside the SpiiPlus firmware.

The information it displays is:

- > PLC cycle (in ms):
PLC cycle means what is the frequency that PLC program is executed is. If Maximum is equal to CTIME, the PLC cycle is always identical to Motion Controller realtime tick. The data that is shown is:
- > Avg:
- > Min:
- > Max:
- > PLC Usage consumption (when active):
How much of the controller usage does the PLC execution take when it is running arranged by:
- > Avg:
- > Min:
- > Max:

Where the values shown are percentages.

- > PLC program: (Program status)
The status can be:
- > Running
- > Stopped
- > Not valid

Syntax

#PLC

6.4.12 LOG

Description

The controller supplies a log of important events to the user.

- > The log can keep the last 500 events in memory.
- > There is a command to set the time stamp for the log (setting current time).
- > There is a command to display the log entries (similar to the Communication Terminal #SI command), the user can use a host program to save these.

Initially the log includes:

- > Motor errors
- > Change in an axis FAULT variable
- > System Errors.
- > Program errors (ACSPL+ buffer termination error).
- > Reported I2C drive errors.
- > Reported I2C component errors.
- > EtherCAT errors

The LOG report has the following format

| Error Type | Format |
|--------------------------------------------------|---------------------------------------------------------------------------|
| Motor Errors | TIME axis error, MERR(AXIS) = ERROR, |
| Change in an axis FAULT variable | TIME FAULTS(AXIS): PREVIOUS_VALUE --> CURRENT_VALUE |
| System Errors | TIME system error, S_ERR = ERROR |
| Program errors (ACSPL+ buffer termination error) | TIME program error, PERR(BUFFER) = ERROR, PERL(BUFFER) = LINE |
| Reported I2C drive errors | TIME driver alarm extended error, address:ADDRESS, axis:AXIS, error:ERROR |

| Error Type | Format |
|-------------------------------|----------------------------------------------------------------------|
| Reported I2C component errors | TIME component fault extended error, address:ADDRESS, error:ERROR |
| EtherCAT errors | TIME network error, ECERR = ERROR |

Syntax

#LOG

Example

```
[2011/03/28] [11:15:17.140] FAULTS (57): 0x0 --> 0x20000
[2011/03/28] [11:15:17.140] FAULTS (58): 0x0 --> 0x20000
[2011/03/28] [11:15:17.140] FAULTS (59): 0x0 --> 0x20000
[2011/03/28] [11:15:17.140] FAULTS (60): 0x0 --> 0x20000
[2011/03/28] [11:15:17.140] FAULTS (61): 0x0 --> 0x20000
[2011/03/28] [11:15:17.140] FAULTS (62): 0x0 --> 0x20000
[2011/03/28] [11:15:17.140] FAULTS (63): 0x0 --> 0x20000
[2011/03/28] [11:15:17.403] system error, S_ERR = 5151
[2011/03/28] [11:15:25.502] FAULTS (0): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.526] FAULTS (1): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.527] FAULTS (2): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.527] FAULTS (3): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.527] FAULTS (4): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.527] FAULTS (5): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.527] FAULTS (6): 0x20000 --> 0x20080
[2011/03/28] [11:15:26.527] FAULTS (7): 0x20000 --> 0x20080
[2011/03/28] [11:15:34.352] FAULTS (0): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (1): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (2): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (3): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (4): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (5): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (6): 0x20080 --> 0x80
[2011/03/28] [11:15:34.352] FAULTS (7): 0x20080 --> 0x80
[2011/03/28] [11:17:09.270] program error, PERR(33) = 3232, PERL(33) = 0
[2011/03/28] [11:17:48.000] program error, PERR(33) = 3232, PERL(33) = 0
[2011/03/28] [11:17:57.958] axis error, MERR(3) = 5017
```

6.4.13 LOG_HOST_TICKS

Description

LOG_HOST_TICKS adds a time offset to the controller's system time when reporting events, so that the time of reported events will match the host millisecond counter.

The controller adds the difference between HOST_TICKS (in milliseconds) and controller's time to event time when reporting events. The controller considers its own power-up time as: 1970/1/1 00:00:00.

If the `HOST_TICKS` supplied is the correct number of milliseconds since that date, the controller displays the correct date and time for the GMT time-zone. A time before 1970/1/1 00:00:00 (controller power-up) is displayed as negative milliseconds before this date.

Syntax

#LOG HOST_TICKS

Example

```
const unsigned __int64 SEC_IN_MIN    = 60;
const unsigned __int64 MSEC_IN_SEC  = 1000;
char SETLOGTIME[100] = "#LOG ";
int SETLOGTIMELENGTH = (int)strlen(SETLOGTIME);
struct __timeb64 timebuffer;
double CurrentTime;
// Set time zone from TZ environment variable. If TZ is not set,
// the operating system is queried to obtain the default value
// for the variable.
_tzset();
// GET CURRENT LOCAL TIME
_ftime64_s(&timebuffer);
CurrentTime = (((double)timebuffer.time)
              - (((double)timebuffer.timezone) * SEC_IN_MIN)) * MSEC_IN_SEC
              + timebuffer.millitm;
SETLOGTIMELENGTH = sprintf_s(SETLOGTIME, 100, "#LOG %f\r", CurrentTime);
if (!acsc_Command(Handle, // communication handle
                 SETLOGTIME, // pointer to the buffer that contains
                       // executed controller's command
                 SETLOGTIMELENGTH, // size of this buffer
                 NULL// waiting call
                ))
{
    printf("transaction error: %d\n", acsc_GetLastError());
}
```

6.4.14 LOGP

Description

Presents G-code run-time errors detected during running the program in simulation mode (using `START/s` command).

Syntax

```
#LOGP buffer_number
```

Arguments

| | |
|---------------|------------------------------------------------------------------------------------------|
| buffer_number | The number of buffer that was ran in simulation mode using <code>START/s</code> command. |
|---------------|------------------------------------------------------------------------------------------|

Comments

- > If the program contains calls to sub-routines residing in the D-Buffer, then all run-time errors occurred inside sub-routine will be addressed by the sub-routine calling line number.
- > Buffer recompilation or rerun clears the list of run-time errors detected during the last running process.



Motion generator related run-time errors are not detected, since the simulation takes place only on G-code level.

7. SPiiPlus Error Codes

This chapter contains explanations of the Error Codes that may appear.

The chapter contains only those Error Codes returned by the controller and does not include errors associated with the C Library. Errors that are detected by the C Library are never returned by the controller. A host application that calls a C Library function can receive these codes if a function call failed. For explanations of the C Library errors see the *SPiiPlus C Library Reference Guide*.

The ACSPL+ Error Codes range 1000 to 7099 and are assigned as follows:

ACSPL+ Syntax Errors – numbers 1000 to 1999

ACSPL+ Compilation Errors – numbers 2000 to 2999

ACSPL+ Runtime Errors – numbers 3000 to 3999

Errors – numbers 5000 to 5150

System Errors - numbers 5151 to 5999

EtherCAT Errors - numbers 6000 to 6999

EtherCAT Slave Errors - numbers 7001-7099

G-Code Error Codes - numbers 2000 to 4000, not related to ACSPL+ error codes



Error code values of 7100 through 8999 are reserved for future use and are currently not used.
 Error code values of 9000 and above are reserved for user-defined error codes.

7.1 ACSPL+ Syntax Errors

These appear in response to syntax errors that the controller detects when a program is entered into the buffer and are reported immediately in the prompt that is displayed in response to the command.

The error code values range between 1000 and 1999.

Table 6-1. ACSPL+ Syntax Errors

| Error Code | Error Message | Remarks |
|------------|------------------------------------|----------------------------------------------------------------------------------------|
| 1001 | The program is suspended | The program is being run in the Step mode, and has been suspended at the current step. |
| 1002 | The program was terminated by user | The program has reached a user-set breakpoint. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1020 | Illegal subcommand | A subroutine command that is not recognized has been entered. |
| 1021 | SP command requires axis specification | The required axis designation is missing. |
| 1022 | Illegal command | A program command that is not recognized has been entered. |
| 1023 | Read-only variable cannot be assigned | The command specifies assignment to a read-only variable. |
| 1024 | Set variable cannot be reported | |
| 1025 | Time interval between two characters in command is more than 2 seconds | The compiler recognizes that a follow-on command has not arrived within 2 seconds. The error is relevant when an application is using a call to a SPiiPlus C Library routine, and usually indicates a communication problem. |
| 1026 | Serial Number, Part Number, or Software Options were already specified | An attempt was made to respecify the controller's serial number, part number, or software options. |
| 1027 | Variable requires axis specification | An ACSPL+ variable missing the required axis specification was used. |
| 1028 | Scalar variable cannot accept axis specification | A scalar variable was used with an axis specification. |
| 1029 | Extra characters after the command | A command has been entered with extraneous characters or parameters that are not recognized. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1030 | Too many parameters | The entry contains too many parameters. |
| 1031 | Illegal array in the array command | The array in the command either does not exist, or its size does not match the requirements of the command. |
| 1032 | Illegal data in array | The array contains data in a format that does not match requirements. |
| 1033 | Illegal edit command | The edit command that has been entered cannot be executed. |
| 1034 | Illegal index value | The command includes numerical index specification, but the specified index is not a number. |
| 1035 | Index is out of range | <p>The error is caused by one the following:</p> <ul style="list-style-type: none"> > The specified index value is more than the number of elements in the array > The specified index value is negative > The specified index values are incompatible (first value in the range greater than last). |
| 1036 | Internal error | An internal error has been detected.. |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1037 | Illegal variable name | The command requires specification of an ACSPL+ variable name, but the specified name is not that of an ACSPL+ variable. |
| 1038 | Wrong checksum in the command | The command contains checksum, and the checksum value is wrong. The error is relevant when an application includes SPiiPlus C Library routines, and indicates communication problems. |
| 1039 | Only one motion per axis can be planned in advance | An attempt was made to setup more than one motion for a specified axis. |
| 1040 | Unable to open file | The command specifies an internal file in the flash memory that does not exist. |
| 1041 | Assigned value is out of range | The command attempts to assign an ACSPL+ variable with a value that is out of the range allowed for this variable. |
| 1042 | Operation failed because of exception | |
| 1043 | Program cannot start because the buffer was not compiled | <p>The command attempts to start an un-compiled ACSPL+ program. To compile a program, in the SPiiPlus MMI Application Studio:</p> <ul style="list-style-type: none"> > In the Program Manager, right-click the buffer and select Compile Buffer, or > Use the #nC command in the Communication Terminal, where n is the buffer number. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1044 | Command cannot be executed while the program is running | <p>The command attempts to affect a running ACSPL+ program. Stop the program before executing the command. To stop a program, in the SPiiPlus MMI Application Studio:</p> <ul style="list-style-type: none"> > In the Program Manager, right-click the buffer and select Stop Buffer, or > Use the #nS command in the Communication Terminal, where n is the buffer number. |
| 1045 | Numerical error in standard function | The command includes an ACSPL+ function that caused a numerical error. Check if the arguments of the function fall into the allowed range. |
| 1046 | Write file error | The command has caused writing to the flash memory that failed. Possible reason is a flash memory overflow because of too many stored user files. |
| 1047 | Read file error | The command has caused reading from the flash memory that failed. A reoccurring error points to a serious problem in the controller hardware or firmware. |
| 1048 | More axes than were defined in the motion | The POINT command specifies more axes than were specified in the motion that the command refers to. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1049 | Axis already belongs to a group | An attempt was made to assign an axis to a group when it has already been assigned to another group. |
| 1050 | Conflict with user-defined axis group | The command is incompatible with a user-defined axis group that was defined before. The axes specified in the command must either all belong to one user-defined axis group, or not to intersect with any user-defined axis group. |
| 1051 | Line number is out of range | The command specifies a line number that does not exist in the specified program buffer. |
| 1052 | Buffer number is out of range | The command specifies illegal buffer number. The controller contains 16 program buffers numbered from 0 to 15, plus the D buffer. |
| 1053 | Wrong type | The command addresses an ACSPL+ variable and the type of the variable is different from the type specified in the command. The error is relevant when an application includes a SPiiPlus C Library routine. The error indicates problems in communication. |
| 1054 | This type of motion is valid for single axis only | An attempt was made to send a single-axis motion to more than one axis. |
| 1055 | Command requires line number specification | The command is missing a required line number specification. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1056 | Parameter defining Master has illegal value | The parameter specifying the master has an illegal value (see MASTER). |
| 1057 | Previous superimposed motion is in progress | |
| 1058 | Slave is not synchronized to master | The slave controller has not been synchronized with the master controller. |
| 1059 | Command PTP/V must specify velocity value | The PTP/v command was issued without a value for velocity. |
| 1060 | Illegal memory command | The memory management command is improperly formatted. |
| 1061 | ')' wasn't found | The command contains a non-paired left bracket. |
| 1062 | Command is too long | Command exceeds maximum permitted length |
| 1063 | Variable is not defined in the buffer | The command addresses a variable that is not declared in the specified buffer, or the specified buffer is not compiled. |
| 1064 | Undefined global variable | The command addresses a global variable that is not declared in any buffer, or the buffer that contains the declaration is not compiled. Variable name must be in upper case. |
| 1065 | The command cannot be executed while the current motion is in progress | The command is in conflict with one or more of the currently executed motions. To kill a motion use KILL/KILLALL or KILLALL . |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1066 | Attempt to compile or start empty buffer | |
| 1067 | GO command failed | The controller has no motions waiting for the GO command. |
| 1068 | Referenced axis does not execute a motion (motion was terminated?) | The command refers to a motion, but the specified axis is not involved in any motion. |
| 1069 | This command can only be used with MPTP , PATH or PVSPLINE motion | Commands POINT and MPOINT are compatible only with the MPTP...ENDS , PATH...ENDS , or PVSPLINE...ENDS motions. |
| 1070 | Attempt to add segment after ENDS command | The command attempts to add a segment or a point to the motion that was closed before with an ENDS of the MPTP...ENDS , PATH...ENDS , or PVSPLINE...ENDS commands. |
| 1071 | File name was expected | The command must specify a name of internal file in the flash memory. |
| 1072 | Wrong array size | The command specifies an array, but the motion to which the command refers, or other command arguments require an array of another size. |
| 1073 | Text for search is not specified | The command must specify a text for search operation. |
| 1074 | Only standard or SP variable is allowed in the command | The command requires an ACSPL+ or SP variable name to be specified. |
| 1075 | Name is not a standard or user variable | The specified variable name is not an ACSPL+ or user-defined variable name. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1076 | Undefined label | The command requires a label specification. The program that contains the label specified in the command must be compiled in a buffer. |
| 1077 | Protection violation | The command attempts to assign a protected variable when the controller is in Protected mode. The controller must be put in the Not Protected mode before protected variables can be assigned. To do this, use the SPiiPlus MMI Application Studio Protection Wizard in Development Tools . |
| 1078 | Variable can be changed only while the motor is disabled | The command attempts to assign a value to a variable that can be changed only if the motor is disabled. A DISABLE/DISABLEALL command must be put in before the value can be assigned to the variable. |
| 1079 | Motion cannot start because one or more motors are disabled | Motor(s) are in motion and have to be disabled before the motion can be initiated. A DISABLE/DISABLEALL command must be put in prior to initiating motion. |
| 1080 | Default Connection flag is set for the motor | |
| 1081 | Incompatible suffixes | The command includes command options that cannot be used together. |
| 1082 | Commands BEGIN, END, KILL, GO require axis specification | A GO or KILL/KILLALL command has been entered without an axis specification. |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1083 | Array requires axis specification | An array requiring an axis specification has been entered without an axis specification. |
| 1084 | Illegal array command | Either the array does not exist, or it is not in a compiled buffer. |
| 1085 | Extra number after the command | The command specifies a superfluous numerical argument. |
| 1086 | Variable name must be specified | The command requires a variable name specification. |
| 1087 | Command cannot be executed while the axis is moving | The command specifies one or more axes that are involved in a motion. The command can be executed only after the motion finishes. |
| 1088 | Variable can be queried only in compiled buffer | The command attempts to query a variable in a buffer that was not compiled. |
| 1089 | Label can be referenced only in compiled buffer | The command attempts to reference a label in a buffer that was not compiled. |
| 1090 | This type of motion is not allowed for grouped axis | The slave or track command specifies an axis that was included in a user-defined group. Only a single axis can be specified in the slave or track command. |
| 1091 | Less arguments than required | The motion command specifies fewer coordinate values than required by the axis specification. |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| 1092 | More arguments than required | The motion command specifies more coordinate values than required by the axis specification. |
| 1093 | Bit selector value is greater than 31 or less than 0 | The expression specified in the bit selector yields a value greater than 31 or less than 0. |
| 1094 | Empty line range is specified | The range of lines specified refers to empty lines. |
| 1095 | No master-slave motion is in progress | The command addresses master-slave motion that has not yet started. |
| 1096 | '}' was not found | The command includes a non-paired left curly bracket. |
| 1097 | Previous data collection is in progress | The command attempts to start a data collection while the previous data collection for the same axis is still in progress. |
| 1098 | Stalled motion requires limits specification | The motion command, which includes a command option of stalled motion, requires specification of the motion limits. |
| 1099 | Extra numbers after the command | The motion command includes superfluous numerical arguments. |
| 1100 | Received command has no message ID | The command does not contain the ID of the message to which it refers. |
| 1101 | The program is suspended, the start line number is not allowed | The command attempts to start a program from a specified line when the program is in suspended state. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <p>A program in suspended state can be only resumed from the next line. The line specification is not allowed.</p> <p>The only way to start a suspended program from arbitrary line is to stop the program, and to start it again from the desired line.</p> |
| 1102 | Zero divide | The an illegal divide by zero is attempted by the command. |
| 1103 | Invalid reference | The command contains an invalid reference. |
| 1104 | No ACSPL program is waiting for input | The command is attempting to send input, but no program is open for receipt. |
| 1105 | Format error | The command is incorrectly formatted. |
| 1106 | SP function failed | <p>The function that accesses SP memory cannot be executed.</p> <p>Check the address specified in the function call. The address must fall into the range of 0 to 511.</p> |
| 1107 | Current empty space in the dynamic buffer is not enough for the command | The D-Buffer is full and no further items can be entered. |
| 1108 | Invalid real number | The command includes specification of a real number that cannot be interpreted as a valid real. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1109 | The command is not allowed in SAFE mode | Safe communication mode is a host communication format based on block-by-block transfer of data with delivery confirmation. |
| 1110 | At least one variable must be specified for data collection | Data collection was attempted without specifying at least one variable. |
| 1111 | Too long reply requested | The time of the reply request parameter is too long. |
| 1112 | No matches were found | The search command did not find any match. |
| 1113 | The step in the table is zero or negative | The command reference to a table element is either zero or negative. |
| 1114 | The program finished without a STOP command | The program that runs in a buffer executed the last command and this was not a STOP/STOPALL command. |
| 1115 | Stack underflow (RET without CALL) | The program that runs in a buffer caused stack underflow. This occurs if the program executes the RET command that without the paired CALL command. |
| 1116 | Stack overflow (too many nested CALLs) | The program that runs in a buffer caused stack overflow. This occurs if the program executes too many nested CALL commands. Check that the program does not specify infinite recursion (subroutine calls itself infinitely). |
| 1117 | Attempt to enter autoroutine directly | The program that runs in a buffer comes to the ON command (see ON...RET). |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | ON must never be executed in the normal program flow. |
| 1118 | Illegal axis number | The command specifies an axis by number or expression, and the resulting value is not a legal axis number. Valid axis numbers range between 0 and the number of axes in the system minus one. |
| 1119 | Integer overflow | The integer value is too large. |
| 1120 | The motion must involve the first two axes from the group | The segmented motion must always involve two axes. If a user-defined group contains more than two axes, the segmented motion must involve the first two axes in the group. |
| 1121 | Unknown #-constant | The command specifies an unknown symbolic constant. |
| 1122 | Bit selection is not applicable | Bit selector cannot be applied to real value. |
| 1123 | Illegal bit selector | Value of bit selector must fall in the range of 0 to 31. |
| 1124 | Attempt to enable motor failed | The enable command failed. Additional information can be obtained from the corresponding element of the MERR variable (motor disable code). |
| 1125 | Error in SP program | The command caused unsuccessful loading of an SP program. The file with the SP program contains an error. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1126 | Illegal SP number | The command specifies an illegal SP number. Valid SP numbers are 0, 1, 2, and 3. |
| 1127 | Editing of this buffer is disabled | <p>The command attempts to open the buffer for editing or to change the program in a Protected buffer.</p> <p>Editing a buffer is disabled if the controller is in the Protected mode and bit 1 of the corresponding element of PFLAGS is set to 1.</p> |
| 1128 | Configuration changed. Commands SAVE and HWRES must be executed. | <p>The program has changed the configuration. The configuration has to be saved to the controller flash and the controller restarted.</p> <p>Select the controller in the Workspace Tree and click the Save to Flash button. Then right-click the controller and select Reboot.</p> |
| 1129 | In binary command the name specification must end with / | The command syntax requires a name to be followed by the / (slash) character. |
| 1130 | Segment sequence for the previous motion was not terminated with ENDS command | Commands MPTP...ENDS , MSEG...ENDS , PATH...ENDS , PVSPLINE...ENDS are followed by a sequence of points or segments. The sequence must terminate with ENDS . |
| 1131 | SP program is incompatible with one or more products | SP program interface is unrecognized. |
| 1132 | The file is not a legal user data file | The file designator is not valid. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1133 | Discrepancy in types of the saved and restored arrays | Error detected in array types when comparing the saved and restored versions. |
| 1134 | Discrepancy in sizes of the saved and restored arrays | Error detected in array size when comparing the saved and restored versions. |
| 1135 | Operation failed because of communication overload | Error occurs when there is either a fault in the communication, or too much communication is being conducted. |
| 1136 | Wrong relation between first point, last point and interval | Usually caused by the first point being greater than the last point. |
| 1137 | Illegal analog output number | The command contains an illegal analog output number. |
| 1138 | Incompatible SP and analog output | Conflict between analog output value in the command and that contained in the SP. |
| 1139 | Illegal input | The controller tries to interpret the inserted string as a response to the executed input command, but the string does not follow the required format. |
| 1140 | The function is not supported | An attempt was made to use a function that is not supported in SPiiPlus. |
| 1141 | Timeout | Communication timeout has occurred. This can be corrected by right-clicking the controller in the Workspace Tree, selecting Properties , and increasing the Connection Timeout value. |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1142 | Arguments are inconsistent | Conflict with the command arguments and their values. |
| 1143 | Memory overflow | Usually caused by infinite loops. |
| 1144 | Simulator does not support this command | The command cannot be executed by Simulator. |
| 1145 | The specified DPR address is less than 128 or exceeds the DPR size | SPiiPlus does not contain a dual-port RAM; therefore this is not relevant. |
| 1146 | Collision with other variable in DPR | SPiiPlus does not contain a dual-port RAM; therefore this is not relevant. |
| 1147 | Incomplete command (intrusion of other process?) | The command is not correctly formulated. |
| 1148 | Requested more SINCOS encoder multipliers than installed | The user tried to select more Sin-Cos multipliers than allowed. |
| 1149 | Illegal SP address | Valid SP numbers are 0, 1, 2, and 3. |
| 1150 | Only even numbers are allowed as DPR address | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 1151 | This is a DEMO version of Simulator. 5 minutes of work remains. | The demo version of the Simulator has a limited session length. The Simulator is going to stop after 5 minutes. |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| 1152 | The DEMO version of Simulator has terminated | The demo version of the Simulator has a limited session length. The session time has elapsed. |
| 1153 | Illegal query command | The controller cannot recognize the query command. |
| 1154 | The command can be only used with MSEG motion | The command was incorrectly used - it can only be used with MSEG...ENDS . |
| 1155 | Motion cannot start because the previous motion failed. Use FCLEAR command. | To start the motion enter the FCLEAR command. |
| 1156 | Profile key must be specified as /SECTION/KEY/ | |
| 1157 | Illegal configuration string. Use only characters K,D,+,-. | Illegal characters were used in the configuration string. |
| 1158 | Cannot find matching value. The formula has no root or the root is not single. | |
| 1159 | Axis number is specified more than once | The axis designation parameter is repeated in the command. |
| 1160 | The axis cannot be used in a group (see AFLAGS) | The specified axis has been flagged as not to be grouped using AFLAGS . |
| 1161 | Illegal communication channel | An invalid communication channel has been specified. |
| 1162 | Illegal tag value | The command Tag parameter does not match the intended command. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| 1163 | Illegal configuration parameters | Invalid system configuration parameters have been entered. |
| 1164 | Illegal password | The password that has been entered does not match the required password. |
| 1165 | Attempt to execute optional function which is not installed | |
| 1166 | Flash file operation failed (overlapped file operations?) | |
| 1167 | Wrong start position is specified | |
| 1168 | File not opened | |
| 1169 | D-Buffer cannot be changed while any other buffer is compiled | An attempt was made to make changes to the D-Buffer while other buffers were being compiled. |
| 1171 | The number of axes in coordinated motion is restricted to 4 | |
| 1172 | Illegal category name | |
| 1173 | EtherCAT offset is out of range | |
| 1174 | Can't find EtherCAT network variable | |
| 1175 | EtherCAT Master is not ready | |
| 1176 | Data size should be specified | |
| 1177 | This slave has no mailbox support | |
| 1178 | Invalid CoE SDO parameter | |
| 1179 | EtherCAT slave is in invalid state for this operation | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------|----------------------------------------------------------------------------|
| 1180 | General EtherCAT error | |
| 1181 | EtherCAT Timeout error | |
| 1182 | Can't split non-existing axes group | An axes group that was not previously defined was specified for splitting. |
| 1183 | Attempt to split group with active motion on | Cannot split an axis group while in motion. |
| 1184 | Mapped variable must be defined in D-Buffer | |
| 1185 | Number of connected EtherCAT devices is not covered by SW options | |
| 1186 | Can't initialize SP injection | |
| 1187 | There is an active injection on this SP | |
| 1188 | Can't enable motor, DIP switch settings are incorrect. | |
| 1189 | The command can only be used with XSEG motion | |
| 1190 | PLC option on this device is not enabled | |
| 1191 | PLC is already running | |
| 1192 | General PLC error | |
| 1193 | Wrong array size or type | |
| 1194 | Not allowed Encoder Type | |
| 1195 | Requested more absolute encoder multipliers than installed | |
| 1196 | Requested absolute encoder is not supported | |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| 1197 | The slave is incompatible with current controller configuration | |
| 1198 | Invalid encoder type | See ACSPL+ variable E_TYPE (axis) in the Command and Variable Reference Guide |
| 1199 | Invalid absolute encoder parameter type | See ACSPL+ variable E_TYPE (axis) in the Command and Variable Reference Guide |
| 1200 | JIT and Dynamic modes are not allowed for D-Buffer | |
| 1201 | End-of-Sequence is illegal for this motion | |
| 1202 | JIT and Dynamic modes are not allowed at the same time | |
| 1203 | DEBUG mode is not allowed for JIT and Dynamic buffers | |
| 1204 | JIT and Dynamic modes require the buffer to be empty | |
| 1205 | Illegal zone number | |
| 1206 | Cannot change a constant variable | |
| 1207 | One or more arguments are out of range | |
| 1208 | Function ended without return statement | |
| 1209 | Entered a function not by calling, STOP statement might be missing | |
| 1210 | Incompatible matrix size | |
| 1211 | Too many segments | See MSEG...ENDS . |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1212 | ARC arguments are inconsistent | See MSEG...ENDS , ARC1 and ARC2 . |
| 1213 | Stopper is prohibited for master-slave motion | See STOPPER . |
| 1214 | Adjacent stoppers are prohibited | See STOPPER . |
| 1215 | In cyclic path the first and the last points must coincide | |
| 1216 | Velocity is specified, but the motion command had no V command option | Velocity argument can be specified in POINT , LINE , ARC1 or ARC2 only if the corresponding motion command specifies /v . |
| 1217 | Segment of zero length | |
| 1218 | ARC radius is too small | |
| 1219 | Specified motion delay is out of range | |
| 1220 | Command is incompatible with connect | |
| 1221 | 20 KHz motion generation incompatible with CTIME or number of axes | See SPiiPlus ADK Suite Release Notes section "Controller Cycle Time (CTIME) Support" for Controller CTIME Limitations |
| 1222 | 20 KHz motion limitation reached | |
| 1223 | Time specified is not possible under current motion safety limitations | |
| 1224 | Offset doesn't exist | |
| 1225 | Microblaze monitor is not ready | |
| 1226 | Address couldn't be found | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 1227 | Maximum number of Registers was already added | |
| 1228 | Firmware is not properly installed | |
| 1229 | SLABITS does not match Single Turn and Multi Turn number of bits | |
| 1230 | Non Linear Control Feature is not supported | |
| 1231 | Illegal key | An invalid Key argument has been entered in either the GETCONF or SETCONF function. |
| 1232 | Illegal index | An invalid Index argument has been entered in either the GETCONF or SETCONF function. |
| 1233 | Illegal value | The Value argument of the SETCONF command specifies a value that is not compatible with the Key argument. |
| 1234 | Value in SETCONF must be 0 or 1 | See SETCONF . |
| 1235 | SETCONF function is disabled | |
| 1236 | SETCONF cannot be executed while the motor is enabled | |
| 1238 | The operation can be executed only when the HSSI channel is in command mode | |
| 1239 | HSSI channel cannot switch to command mode because it affects one or more remote drivers. Reset corresponding MFLAGS.#HSSI bits before. | See MFLAGS . |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| 1240 | Operation is temporarily disabled | |
| 1241 | Operation failed | |
| 1242 | The feature is not supported | |
| 1243 | Max number of 20 KHz moving axes reached | |
| 1244 | Illegal number of coordinates specified | |
| 1245 | The specified channel is already being used. Please use the MBCLOSE function to close the connection | |
| 1251 | Operation is temporarily disabled | |
| 1252 | Motor cannot be enabled while a motion is in termination process | |
| 1253 | Unable to work with Dummy motor | The command cannot be executed for a dummy motor. Check flag MFLAGS.#DUMMY (see MFLAGS). |
| 1254 | The operation requires the motor to be enabled | The ENABLE/ENABLE ALL command has to be run. |
| 1255 | The operation requires the motor to be disabled | The DISABLE/DISABLEALL command has to be run. |
| 1256 | The operation is valid only for brushless motor without Hall sensor | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 1257 | The operation failed because the brushless motor is not commutated | Refer to the chapter on the Adjuster Wizard in the <i>SPiiPlus MMI Application Studio User Guide</i> for details on commutation. |
| 1258 | The operation failed because the motor is in open-loop mode | |
| 1259 | Motion cannot start because the motor is defined as dummy | |
| 1260 | Motion cannot start because the motor is disabled | The ENABLE/ENABLE ALL command has to be run. |
| 1261 | Motion cannot start because the brushless motor is not commutated | Refer to the chapter on the Adjuster Wizard in the <i>SPiiPlus MMI Application Studio User Guide</i> for details on commutation. |
| 1262 | Motion cannot start because the motor is in open-loop mode | |
| 1263 | Motion cannot start because the previous motion failed. Use FCLEAR command. | To start the motion enter the FCLEAR command. |
| 1265 | SP program does not support this operation | |
| 1266 | Invalid PEG pulse width | Pulse width set in or is incorrect. |
| 1267 | Maximal number of time-based PEG pulses is 65,535 | Not relevant to PEG operations. |
| 1268 | Invalid period of time-based PEG pulses | Not relevant to PEG operations. |
| 1269 | PEG pulse width must be less than time-based pulse period | Not relevant to operations. |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 1270 | PEG is not supported for the specified axis | See . |
| 1271 | Step does not agree with start/end points | |
| 1272 | Incremental PEG step must correspond to encoder counts within the $(-2^{31}, 2^{31}-1)$ range, zero is excluded | The PEG step in encoder counts must be in the range of -2^{31} to $2^{31}-1$. |
| 1273 | The specified axis does not exist in this controller model | Refers to controllers that support 4 axes only, and the specified axis is above this. |
| 1274 | The specified axis is defined as dummy only | |
| 1275 | Only stepper motor is supported for the specified axis | |
| 1276 | The brushless motor is not supported for the specified axis | |
| 1277 | Dual loop control is not supported for the specified axis | |
| 1278 | Remote HSSI driver is not supported for the specified axis | |
| 1279 | PEG states are not supported for the specified axis | |
| 1280 | The stepper motor is not supported for the specified axis | |
| 1281 | No Hall support for 2-Phase motors | |
| 1282 | Value out of range | |
| 1283 | Number of points for Random PEG exceeds the limit | |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------------------------------|---------|
| 1284 | Axis is not connected to PEG engine | |
| 1285 | NanoMotion Piezoceramic Motor is not supported for the specified axis | |
| 1286 | Fast Sin-Cos Encoder is not supported for the specified axis | |
| 1287 | Laser Modes are not supported for the specified axis | |
| 1288 | Time-based PEG is not supported for the specified axis | |
| 1289 | This function is not supported for the specified axis | |
| 1290 | SPINJECT/SPRT and Fast GPRT cannot be used in parallel | |
| 1291 | P/D interface is required for this function | |
| 1292 | Secondary Feedback is not supported | |
| 1293 | SLABITS does not match Single Turn and Multi Turn number of bits | |
| 1294 | Motion cannot start because the Gantry Complementary brushless motor is not commutated | |
| 1300 | Non Linear Control License is required | |
| 1301 | Another EtherCAT port is already closed | |
| 1302 | Not possible to save network topology configuration when | |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------|---------|
| | more than one line break exists | |
| 1303 | ERROR SDO: Object does not exist in the object dictionary. | |
| 1304 | The value is too high (check E_PAR_D value) | |
| 1305 | The network variable is already mapped to ACSPL+ variable | |
| 1306 | Invalid axis list | |
| 1307 | FoE Protocol is not supported by Slave | |
| 1308 | The Disk is full or the file is too big | |
| 1309 | Function supported only by CiA402 Drive | |
| 1310 | CiA402 Drive is not in OP State. PDO is not enabled. | |
| 1311 | Cannot convert REAL type to INT | |
| 1312 | More than one hold command is not allowed | |
| 1313 | Continue command is already in process | |
| 1314 | Requested Homing Method is not supported | |
| 1316 | Homing in Gantry mode requires additional parameters | |
| 1317 | FoE Error: Access denied | |
| 1318 | Matrix is not invertible | |
| 1319 | PI AME interface is not initialized | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------|---------|
| 1320 | PI AME interface is busy | |
| 1325 | Main FPGA file downloading is in progress | |
| 1326 | Operation aborted, PEG is in process | |
| 1327 | FPGA versions do not match | |
| 1328 | FPGA version is the same. FPGA Upgrade aborted. | |
| 1329 | Invalid value, digital input index should range between 0-99 and bit index should range between 0-31 | |
| 1330 | Invalid value, digital output index should range between 0-99 and bit index should range between 0-31 | |
| 1331 | This output is already mapped as a mechanical brake to a different axis | |
| 1332 | Hardware limit swapping (MFLAGSX.#HLIMSWAP) and limit routing are mutually exclusive | |
| 1333 | File name MAX length is 100 chars | |
| 1335 | Illegal SPATH specification | |
| 1336 | Nurbs motion was not initiated | |
| 1337 | Illegal NURBS/SPATH parameter value | |
| 1338 | Profile has ended or last knots already added | |
| 1339 | Invalid number of axes | |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1340 | Dummy point must also specify a Knot value | |
| 1341 | Illegal dummy point, two points are allowed before coordinates and two after | |
| 1342 | The measurement you are trying to activate is already active. (see DPM_Measurement Stop() function for more details) | |
| 1343 | An invalid sample type has been specified. Type can either be 0 or 1. (see DPM_Measurement sample type for more details) | |
| 1344 | The sampling time specified is invalid, the minimum value is the controller cycle time(?CTIME) and the maximum value is: 100000000msec | |
| 1345 | The sample-set size specified is invalid. Minimum size: 1. Maximum size: 512. | |
| 1346 | The specified "when_to_measure" flag is invalid. (see DPM_Measurement documentation for more details) | |
| 1347 | The measurement you are trying to activate is paused. (see DPM_Measurement Stop()/Resume() functions for more details) | |
| 1348 | Cannot change the selected axis while monitoring is ON. (see DPM_Motion_Status SelectAxis()/MonitorOff() functions for more details) | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1349 | The specified "monitored_variable" is invalid. (see DPM_Measurement documentation for more details) | |
| 1350 | Illegal attempt to copy STRING | |
| 1360 | LCI unit initialization failed | |
| 1361 | Unable to start the LCI operation | |
| 1362 | Trajectory axes are not defined | |
| 1363 | LCI Safety input is off | |
| 1364 | LCI laser fault input is on | |
| 1365 | LCI supported velocity limit exceeded | |
| 1366 | LCI unit not found | |
| 1367 | LCI. Unable to allocate channel for specified operation | |
| 1368 | LCI function timeout | |
| 1375 | The specified IP address is invalid. Please see MBOPEN documentation for more details. | |
| 1376 | The specified server ID is invalid (valid values are between [1 - 247]). Please see MBOPEN documentation for more details. | |
| 1377 | The specified word-order is invalid. (valid values are 0 for big-endian and 1 for little-endian) Please see MBOPEN documentation for more details. | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1378 | Unable to open more than 3 Modbus connections at any given time. Please see MBOPEN documentation for more details. | |
| 1379 | The specified communication handle is invalid. | |
| 1380 | A connection with the specified IP was not found. | |
| 1381 | The specified address is invalid. | |
| 1382 | The specified request frequency is invalid. The minimum value is 5 milliseconds. Please see the documentation for more details. | |
| 1383 | The specified variable length and the specified number of elements do not match. Please see the documentation for more details. | |
| 1384 | The specified suffix combination is invalid. Please see the documentation for more details. | |
| 1385 | Read-Only variables cannot be written. | |
| 1386 | The specified request ID is invalid. | |
| 1387 | The specified variable(or part of it) is already being written to by another Modbus request. Use #MBMAPREP for more details. | |
| 1388 | The specified server address is already being written to by another Modbus request. Use #MBMAPREP for more details. | |
| 1389 | The specified number of | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| | elements is greater than the specified variable length. Please see the documentation for more details. | |
| 1390 | The maximum number of integer values that can be "mapped" in a single request is 16. Please see the documentation for more details. | |
| 1391 | The maximum number of short values that can be "mapped" in a single request is 32. Please see the documentation for more details. | |
| 1392 | The maximum number of single-precision floating-point values that can be "mapped" in a single request is 16. Please see the documentation for more details. | |
| 1393 | The maximum number of double-precision floating-point values that can be "mapped" in a single request is 8. Please see the documentation for more details. | |
| 1394 | The maximum number of coil values that can be "mapped" in a single request is 32. Please see the documentation for more details. | |
| 1395 | The maximum number of discrete input values that can be "mapped" in a single request is 32. Please see the documentation for more details. | |
| 1396 | Failed to establish a connection with the specified server device. | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1397 | The maximum number of mapping requests per server is 32. Please see the documentation for more details. | |
| 1398 | A connection with the specified IP is already open and with different configuration parameters. To change the connection parameters, close and open a new connection. | |
| 1399 | The specified Modbus channel operates in sequential mode. This mode does not support user-defined request frequency. Each request is sent after a response for the previous one has arrived, but no less than 10 milliseconds from the last time it has been sent. Please see the documentation for more details. | |
| 1410 | The minimum value can not be greater than the maximum value. Please see the documentation for more details. | |
| 1411 | G-code: Run-time errors were detected during the program execution in simulation mode. Use the #LOGP <buffer number> command to present the detected errors. | |
| 1412 | This motion type is not supported | |
| 1413 | The supplied correction maps (2-dimensional arrays) should have the same dimensions. Please see the documentation for more details. | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------------------------|---------|
| 1414 | The Specified referenced axes/analog inputs are invalid. Please see the documentation for more details. | |
| 1415 | Internal Error | |
| 1416 | Stage model dll is not loaded | |
| 1417 | Stage model dll's name does not match the stage model implemented by the dll | |
| 1418 | This functionality has not implemented yet | |
| 1419 | TCP and machine axes conflict. Consider to change either TCP axes by changing AXISDEF parameter or machine axes | |

7.2 ACSPL+ Compilation Errors

The error codes in ACSPL+ compilation errors range between 2000 and 2999.

ACSPL+ compilation errors are reported either immediately when the erroneous line is inserted, or subsequently, when the controller attempts to compile an ACSPL+ program. If a program in a buffer undergoes compilation and an error is detected, the error code is stored in the corresponding element of the **PERL** array and the erroneous line number is stored in the corresponding element of the **PERL** array.

Table 6-2. ACSPL+ Compilation Errors

| Error Code | Error Message | Remarks |
|------------|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2002 | Unrecognized command | <p>The program line contains a sequence that is not recognized as a legal command.</p> <p>If the line starts from a standard variable name, check the following:</p> <ul style="list-style-type: none"> > All letters are uppercase > Spelling is correct <p>If the line starts from a user variable name, check the following:</p> <ul style="list-style-type: none"> > The variable is defined before use > The letter case matches the definition > Spelling is correct |
| 2003 | Unexpected delimiter, incomplete command | A command contained in the line is incomplete. |
| 2006 | Unexpected END | The END command is specified without a paired IF , LOOP or WHILE command. |

| Error Code | Error Message | Remarks |
|------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2007 | Two adjacent operators | Two binary operator are specified in the expression without an operand between. |
| 2008 | Left bracket was expected | Left bracket was not found in a position where it is expected. Possible reasons for this error are array name without index or function call without arguments. |
| 2009 | Right bracket was expected | The expression or sub-expression has an unpaired left bracket. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2010 | Comma was expected | <p>Comma character was not found in a position where it is expected.</p> <p>Possible reasons for this error are:</p> <ul style="list-style-type: none"> > Missing comma between the arguments of the command or function > The command or function call has less arguments than required. |
| 2011 | Equals sign was expected | Equal sign in assignment is missing. |
| 2012 | Direct command was expected | |
| 2013 | DO or LOOP without END | |
| 2015 | Integer positive constant is expected | The declaration of array contains a size definition other than positive integer constant. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2016 | Only label or line number is allowed as start point | The start command can contain the start line definition. The start line definition can appear only as a label or a positive line number. |
| 2018 | Scalar variable cannot be indexed or used with axis specification | The command contains a scalar variable followed by index specification or prefixed by axis specification. |
| 2019 | Write-only variable cannot be read | |
| 2020 | Read-only variable cannot be assigned | The command attempts to assign a read-only variable. |
| 2021 | Label was expected | GOTO or CALL must specify a label name. |
| 2022 | Array name was expected | The command or function requires an array name as one of its arguments. The array name must not be indexed. |
| 2023 | Variable name was expected | The declaration specifies an illegal variable name. A variable name must begin with a letter or underscore and can include letters, digits and underscores. |

| Error Code | Error Message | Remarks |
|------------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | <p>A variable name must not include any of the following:</p> <ul style="list-style-type: none"> > Keywords (IF, WAIT, SIN, etc.) or standard variable names (FPOS, MST, etc.) > Standard label (AUTOEXEC). |
| 2024 | Undefined label | GOTO or CALL specify a label name that was not defined in the program. |
| 2025 | Duplicated label | The program contains two identical labels. |
| 2026 | Undefined variable name | The command contains a variable name that was not declared in the program. |
| 2027 | Duplicated variable name | The program declares two variables with identical names. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2030 | Number user array elements can not exceed the XARRSIZE parameter | <p>The value of the XARRSIZE parameter is limited to a maximum of 30,000 elements.</p> <p>The limitation applies also to two-dimensional arrays. The total number of elements in a two-dimensional array is equal to the product of its sizes by each dimension. The total number of elements must not exceed 30,000.</p> |
| 2031 | Global of different type/dimension was defined in other buffer | The program declares a global variable with the name that was already declared in other buffer with different type or dimension. |
| 2033 | Mandatory argument is omitted | The command or function call contains fewer arguments than required. |
| 2034 | More arguments than required | The command or function call contains more arguments than required. |
| 2035 | Wrong argument type | The command or function call contains an argument of incorrect type. |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2036 | Function that does not return a value cannot be used in expression | The expression cannot include a function that has no return value. |
| 2037 | Axis specification was expected | This error refers to a logical check. The command must specify an axis. Axis specification can appear: as an integer designation, like 0, 1, 32, or as an expression that evaluates to an integer within the range of 0 and the number of axes in the system minus 1. |
| 2038 | Axis specification was expected | This error refers to a physical check. The command must specify an axis that physically exists. Axis specification can appear: as an integer designation, like 0, 1, 32, or as an expression that evaluates to an integer within the range of 0 and the number of axes in the system minus 1. |
| 2042 | Array index must be a non-negative integer constant | |
| 2043 | Internal error | |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2044 | Index is out of range | The constant index value is greater than the array size minus one, or is negative. |
| 2045 | Illegal axis value | The axis value specified by a numerical constant is greater than the number of axes in system minus one, or is negative. |
| 2048 | Argument must be specified as + or - sign | The command requires a direction specification that must be presented by character + or -. |
| 2049 | Illegal suffix for this command | The command specifies a command option, which is illegal for this command. |
| 2050 | Name of standard variable was expected | The command requires an argument that specifies one of the ACSPL+ variables. |
| 2051 | Only APOS, RPOS, FPOS and F2POS are allowed in SET command | The SET command must specify assignment to one element of one of the following variables: APOS , RPOS , FPOS , F2POS . The variables must be in upper case. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2052 | Variable name was expected | The command requires an argument that specifies a scalar variable or an element of array. A general expression or a constant is not allowed for this argument. |
| 2053 | Constant argument was expected | The command requires an argument that specifies a numerical constant. |
| 2054 | Illegal buffer number | The command must specify a constant buffer number. The constant must fall in the range of from 0 to 15. |
| 2055 | Assigned value is out of range | The command attempts to assign a variable with a value that falls out of the range allowed for the variable. |
| 2056 | Zero divide | The command attempts to divide by zero. |
| 2057 | Only VEL,ACC,DEC,JERK,KDEC are allowed in IMM command | IMM must specify assignment to one element of one of the following variables: VEL , ACC , DEC , JERK , KDEC . |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2058 | Bit selection cannot be applied to real variable | The bit selector is specified for a real variable. |
| 2059 | ELSE without IF | ELSE is specified without corresponding IF (see IF , ELSEIF , ELSE...END). |
| 2060 | ELSEIF without IF | ELSEIF is specified without corresponding IF (see IF , ELSEIF , ELSE...END). |
| 2061 | LOOP without END | LOOP has no corresponding END (see LOOP...END). |
| 2062 | DO without END | |
| 2063 | IF without END | IF has no corresponding END (see IF , ELSEIF , ELSE...END). |
| 2064 | Memory overflow | The application requires too much memory. Reduce the number and size of the local and global variables used in the application, or use a more powerful model of the controller. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| 2065 | Axis constant or axis expression in brackets expected | The command must include an axis specification in parentheses, for example: FMASK (4).16 , where 4 is the axis specification. |
| 2066 | Too many axis specifiers | The axis specification includes too many axis specifiers. |
| 2067 | An axis is specified more than once | The axis specification includes two identical axis specifiers. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2068 | Sign constant or sign expression in brackets expected | <p>The command must include a sign specification. Each sign in the specification defines a direction of one involved axis.</p> <p>Sign specification can appear:</p> <ul style="list-style-type: none"> > As a sequence of sign characters, like +, +--, --+, -+-- > As a list of expressions enclosed in brackets where (0, 1, 0) is equivalent to +--. |
| 2069 | Too many sign specifiers | The sign specification includes too many sign specifiers. |
| 2070 | Unknown #-constant | The command includes an unknown symbolic constant. |
| 2071 | Local variable is not allowed in this command | The command does not allow the use of local variables. Only ACSPL+ variables and user global variables that have been declared can be used. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| 2072 | WHILE without END | WHILE has no corresponding END (see WHILE...END). |
| 2073 | WAIT,TILL,GOTO,CALL,RET,LOOP,WHILE,ON,INPUT,AXISDEF, ENCREAD are not allowed in immediate command | The specified commands cannot be entered through the SPiiPlus MMI Application Studio Communication Terminal . |
| 2074 | Only RPOS variable is allowed after CONNECT | CONNECT must specify assignment to one element of one of the RPOS variables. |
| 2075 | Only MPOS variable is allowed after MASTER | MASTER must specify assignment to one element of one of the MPOS variables. |
| 2076 | Constant bit selector is greater than 31 or less than 0 | The command includes a constant bit selector. The value of bit selector must fall into the range from 0 to 31. |
| 2077 | Array name requires indexing | The array name must be followed by an index specification. For a two-dimensional array, two index specifiers are required. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2078 | Current empty space in the dynamic buffer is not enough for the command | The dynamic buffer contains too many commands queued for execution. The application must wait for the controller to execute and remove one or more commands from the buffer. |
| 2079 | GOTO,CALL,RET,LOOP,WHILE,IF,ON are not allowed in a dynamic or JIT buffer | Commands GOTO , CALL , LOOP...END , WAIT , IF , ELSEIF , ELSE...END , ON...RET cannot be used in the dynamic buffer. |
| 2080 | Local variable declaration is not allowed in immediate command | Local variables cannot be used in a SPiiPlus MMI Application Studio Communication Terminal command. |
| 2081 | Variable declaration is not allowed in a dynamic or JIT buffer | Variable declaration cannot be executed in a dynamic buffer. |
| 2082 | Illegal string argument | The command requires file name specification. |
| 2083 | Integer overflow | The result of constant integer expression is more than 2147483647 or less than -2147483648. |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Consider using real constants instead of integer. To change constant type to real, add decimal point to the end of the constant. |
| 2084 | Integer constants are allowed in the range from -2147483648 to +2147483647 | Consider using real constants instead of integer. To change constant type to real, add decimal point to the end of the constant. |
| 2085 | Protection violation | The controller is in the Protected mode and the command violates one of the protection rules. |
| 2086 | Protection attribute cannot be changed for this parameter | |
| 2087 | Only constant 0 or 1 is allowed at the right side | The command allows only 0 or 1 to the right of the equals sign. |
| 2088 | No dual-port RAM in this controller | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2089 | Bit selection is not available for DPR variable | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 2090 | Only global variables can be defined in DPR | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 2091 | DPR address must be specified | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2092 | Only even numbers from 128 to 504 are allowed as DPR address | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 2093 | Collision with other variable in DPR | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 2094 | DPR variable is not allowed in this command | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 2095 | Illegal line number | The specified line number must be a positive non-zero integer. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2096 | Only even numbers from 128 to 1016 are allowed as DPR address | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 2097 | Illegal SP number | Valid SP numbers are 0, 1, 2, and 3. |
| 2098 | Illegal SP variable specification | Invalid SP variable was entered. |
| 2099 | Undefined SP variable | Non-SP variable was entered. |
| 2100 | User-defined array was expected | |
| 2101 | Illegal character constant | |
| 2102 | Illegal tag | |
| 2103 | Tag can be specified only for global variables | |
| 2104 | BLOCK cannot be nested | See BLOCK...END . |
| 2105 | BLOCK without END | See BLOCK...END . |
| 2106 | Illegal declaration | |
| 2107 | Axis redefinition | |
| 2108 | Assignment to constant | Value has to be assigned to a variable. |
| 2109 | ALL cannot be used in this command | |
| 2110 | Ambiguous axis specification | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| 2111 | Illegal format of real constant | |
| 2112 | This function cannot be used as argument of CONNECT, TRIGGER functions, or in Autoroutine condition | See CONNECT , TRIGGER or ON...RET (for the autoroutine structure). |
| 2113 | Shared memory variables must be global | |
| 2114 | Shared memory variables can be declared only in the D-Buffer | |
| 2115 | Bit selector must be integer scalar or constant | |
| 2116 | No shared memory support in this controller | |
| 2117 | G-code programs execution requires option | |
| 2118 | CALL,CONNECT,MASTER are not allowed in D-buffer | |
| 2119 | Static/Constant variables can only be declared in the D-Buffer | |
| 2120 | Static variable must be global | |
| 2121 | Static variable of a different type/dimension was already defined. Use #VGV command to deallocate. | |
| 2122 | A Static variable of the same name is already defined | |
| 2123 | A Static variable was expected | |
| 2124 | Dimension mismatch on array initialization. | |
| 2125 | Array Initialization is possible in D-Buffer only | |
| 2126 | Axes X,Y,Z cannot be defined as Rotary axes | |
| 2127 | Closing curly bracket not expected | |
| 2128 | Curly bracket expected | |
| 2129 | It is illegal to define a function inside another function | |
| 2130 | Function parameter definition expected | |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------------|---------|
| 2131 | Wrong function arguments | |
| 2132 | Action not supported | |
| 2133 | Function signature does not conform to previous declaration | |
| 2134 | A parameter by the same name already exists | |
| 2135 | A function by that name already exists | |
| 2136 | A Function was defined but never implemented | |
| 2137 | A void/Composite type function is not applicable | |
| 2138 | A Function should not be defined as Static/Local/Global/Const | |
| 2139 | Array dimensions mismatch | |
| 2140 | Array dimensions must be positive and known at compilation time | |
| 2141 | A variable by that name already exists | |
| 2142 | Function call is not allowed in a DISP statement | |
| 2143 | Only user defined primitive variables can be passed by reference | |
| 2144 | Constant Parameters are not allowed | |
| 2145 | First array dimension should not be specified | |
| 2146 | A function was defined without a return statement | |
| 2147 | Functions variable type expected | |
| 2148 | Structs can only be defined in D-Buffer | |
| 2149 | Struct name was expected | |
| 2150 | Structs can only contain fields and functions | |
| 2151 | Structs can only be defined as global in D_buffer or local otherwise | |


| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------|---------|
| 2152 | Field is not recognised | |
| 2153 | Field selection expected | |
| 2154 | Global/Static not allowed | |
| 2155 | Struct Functions can only accept primitive types | |
| 2156 | Struct already has a property by that name | |
| 2157 | Void type cannot be defined | |
| 2158 | Illegal definition of struct inside struct | |
| 2159 | All struct fields must be defined before functions | |
| 2160 | Illegal array type | |
| 2161 | Name Is Taken | |
| 2162 | Illegal Field/Function defintion | |
| 2163 | Illegal variable qualification | |
| 2164 | A Struct by that name already exists | |
| 2165 | A return value was expected | |
| 2166 | The Type selected was not properly defined | |
| 2167 | Maximum number of structs reached | |
| 2168 | Function name was expected | |
| 2169 | The function has already been implemented | |
| 2170 | Fastcall functions do not allow DISP/TILL/WAIT/Function Call/G-Code operations | |
| 2171 | It is illegal for a name to start with {N/n}{0-9} | |
| 2172 | Only a D-Buffer FASTCALL function can be passed as parameter, all parameters must be INT/REAL | |
| 2173 | Trying to assign to read-only field | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------|---------|
| 2174 | Trying to send a read-only field by reference | |
| 2175 | The function does not conform | |
| 2176 | Default Value for function parameter missing | |
| 2177 | Array default parameter must be 0 | |
| 2178 | Default values can only be defined for INT/REAL/MATRIX/Arrays | |
| 2179 | Default parameters must be specified in declaration only | |
| 2180 | The increment operator can only be applied to an Integer type variable | |
| 2181 | Dimension mismatch on matrix initialization. | |
| 2182 | Illegal matrix operation. | |
| 2183 | The argument matrix must be square. | |
| 2184 | The second argument matrix must be square. | |
| 2185 | The matrices are not of the same size. | |
| 2186 | The first argument matrix columns number does not match the second argument matrix rows number. | |
| 2187 | The result matrix size mismatch. | |
| 2188 | Matrix was expected. | |
| 2189 | Operators concatenation is not allowed. | |
| 2190 | Matrix type requires 2 dimensions. | |
| 2191 | Only ACSPL+ Standard Structs can be defined as static | |
| 2192 | This Struct type must only define static variables | |
| 2193 | Struct fields cannot be initialized | |
| 2194 | Some ACSPL commands cannot be used as function name i.e. "DISP", "CONNECT" ... | |
| 2195 | Connect/Trigger expressions do not support user defined | |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------------------------------|---------|
| | functions | |
| 2196 | Some parameters must have a fixed memory - Static user variable or standard ACSPL variable | |
| 2197 | Real time C function library requires option | |
| 2198 | Real time C function library can only be declared in D-Buffer | |
| 2199 | Real time C function library declarations not allowed in a struct | |
| 2200 | Failed to load library | |
| 2201 | Real time C function library must be FASTCALL | |
| 2202 | Real time C function arguments cannot be structs | |
| 2203 | Real time C function cannot have ACSPL+ body | |
| 2204 | Failed to load function | |
| 2205 | Only FASTCALL functions are applicable in this context | |
| 2206 | Number of real time C function libraries exceeds the limit | |
| 2207 | No return value is expected | |
| 2208 | Illegal SWITCH syntax used | |
| 2209 | Illegal String syntax used | |
| 2210 | Illegal String operation | |
| 2211 | Illegal String definition | |
| 2212 | Function variables definition inside a syntactical strucutre is not allowed | |

7.3 ACSPL+ Runtime Errors

The ACSPL+ runtime error code values range between 3000 and 3999.



Codes from 3000 to 3019, however, do not indicate an error. For example, code 3001 reports that the program is suspended and code 3002 reports that the user has terminated the program.
 Codes from 3020 to 3999 indicate run-time errors.

If an error occurs in immediate execution of ACSPL+ command, the error is indicated immediately in the prompt. If an error occurs when an ACSPL+ program is executed in a buffer, no immediate indication is provided. Instead, the error code and the line number are stored in the corresponding elements of the **PERL** and **PERL** arrays.

Table 6-3. ACSPL+ Runtime Errors

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------|------------------------------------------------------------------|
| 3020 | Illegal subcommand | |
| 3021 | SP command requires axis specification | |
| 3022 | Illegal command | |
| 3023 | Read-only variable cannot be assigned | A command specifies an assignment to a read-only variable. |
| 3024 | Set variable cannot be reported | |
| 3025 | Time interval between two characters in command is more than 2 seconds | |
| 3026 | Serial Number, Part Number, or Software Options were already specified | |
| 3027 | Variable requires axis specification | |
| 3028 | Scalar variable cannot accept axis specification | A scalar variable cannot be prefixed with an axis specification. |
| 3029 | Extra characters after the command | |
| 3030 | Too many parameters | |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3031 | Illegal array in the array command | |
| 3032 | Illegal data in array | |
| 3033 | Illegal edit command | |
| 3034 | Illegal index value | The command specifies an assignment to a read-only variable. |
| 3035 | Index is out of range | <p>The reason of the error is one the following:</p> <ul style="list-style-type: none"> > The specified index value is more or equal to the number of elements in the array > The specified index value is negative > The specified index values are incompatible (first value in the range greater than last). |
| 3036 | Internal error | |
| 3037 | Illegal variable name | The command requires specification of a standard variable name, but the specified name is not a name of an ACSPL+ variable. |
| 3038 | Wrong checksum in the command | |
| 3039 | Only one motion per axis can be planned in advance | |
| 3040 | Unable to open file | The command specifies a file in the flash memory that does not exist. |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3041 | Assigned value is out of range | The command attempts to assign the standard variable with a value that is out of the range allowed for this variable. |
| 3042 | Operation failed because of exception | |
| 3043 | Program cannot start because the buffer was not compiled | <p>The command attempts to start an ACSPL+ program that has not been compiled. To compile a program, in the SPiiPlus MMI Application Studio:</p> <ul style="list-style-type: none"> > In the Program Manager, right-click the buffer and select Compile Buffer, or > Use the #nC command in the Communication Terminal, where n is the buffer number. |
| 3044 | Command cannot be executed while the program is running | <p>The command attempts to affect a running ACSPL+ program. Stop the program before executing the command. To stop a program, in the SPiiPlus MMI Application Studio:</p> <ul style="list-style-type: none"> > In the Program Manager, right-click the buffer and select Stop Buffer, or > Use the #nS command in the Communication Terminal, where n is the buffer number. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3045 | Numerical error in standard function | The command includes an ACSPL+ function that caused a numerical error. Check if the arguments of the function fall into the allowed range. |
| 3046 | Write file error | The command caused a failed write to flash memory. A re-occurring error of this type indicates a serious problem in the controller hardware or firmware. |
| 3047 | Read file error | The command caused a failed read from flash memory. A re-occurring error of this type points to a serious problem in the controller hardware or firmware. |
| 3048 | More axes than were defined in the motion | The POINT command specifies more axes than were specified in the motion that the command refers to. |
| 3049 | Axis already belongs to a group | |
| 3050 | Conflict with user-defined axis group | The command is incompatible with a previously defined user-defined axis group. The axes specified in the command may either belong to one user-defined axis group, or may not intersect with a user-defined axis group. |
| 3051 | Line number is out of range | The command specifies a line number that does not exist in the specified program buffer. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3052 | Buffer number is out of range | The command specifies an illegal buffer number. The controller has program buffers numbered 0 to 15, where 15 is the D Buffer.. |
| 3053 | Wrong type | The command addresses a standard variable of a different type from the variable specified in the command. This error never occurs when the user communicates with the controller through a communication terminal. The error occurs only when an application communicates with the controller using the SPiiPlus C Library. The error indicates a communication problem. |
| 3054 | This type of motion is valid for single axis only | |
| 3055 | Command requires line number specification | The command must contain a line number specification. |
| 3056 | Parameter defining Master has illegal value | |
| 3057 | Previous superimposed motion is in progress | |
| 3058 | Slave is not synchronized to master | |
| 3059 | Command PTP/V must specify velocity value | |
| 3060 | Illegal memory command | The memory management command is improperly formatted. |
| 3061 | ')' wasn't found | The command contains a non-paired left bracket. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 3062 | Command is too long | |
| 3063 | Variable is not defined in the buffer | The command addresses a variable that is not declared in the specified buffer, or the specified buffer is not compiled. |
| 3064 | Undefined global variable | The command addresses a global variable that is not declared in any buffer, or the buffer that contains the declaration is not compiled. |
| 3065 | The command cannot be executed while the current motion is in progress | The command is in conflict with one or more of the currently executed motions. To kill a motion use KILL/KILLALL . |
| 3066 | Attempt to compile or start empty buffer | |
| 3067 | GO command failed | The controller has no motions waiting for GO . |
| 3068 | Referenced axis does not execute a motion (motion was terminated?) | The command specifies an axis, but no motion was specified for this axis. |
| 3069 | This command can only be used with MPTP, PATH or PVSPLINE motion | The command specifies an axis, but the motion specified for the axis is incompatible with the command. |
| 3070 | Attempt to add segment after ENDS command | The command attempts to add a segment or a point to the motion that has already been closed by ENDS . |
| 3071 | File name was expected | The command must specify the name of an internal file in the flash memory. |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3072 | Wrong array size | The command specifies an array, but the motion that the command refers to, or other command arguments require an array of another size. |
| 3073 | Text for search is not specified | The command must specify a text for search operation. |
| 3074 | Only standard or SP variable is allowed in the command | The command requires an ACSPL+ or SP variable name to be specified. |
| 3075 | Name is not a standard or user variable | |
| 3076 | Undefined label | The command requires a label specification. The program that contains the label specified in the command must be compiled in a buffer. |
| 3077 | Protection violation | The command attempts to assign a protected variable when the controller is in protected mode. The controller must be in configuration mode before protected variables can be assigned. |
| 3078 | Variable can be changed only while the motor is disabled | <p>The command attempts to assign a variable that can be changed only if the motor is disabled:</p> <ul style="list-style-type: none"> > If the variable is axis-related, disable the corresponding motor before assigning the variable. > If the variable is not axis-related, disable all motors before assigning the variable. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3079 | Motion cannot start because one or more motors are disabled | The motion command involves one or more motors that are disabled. |
| 3080 | Default Connection flag is set for the motor | The command cannot be executed because the default connection flag is set for the motor. The default connection flag is bit 17, i.e., #DEFCON , of the MFLAGS variable (see MFLAGS). |
| 3081 | Incompatible suffixes | The command includes switches that cannot be used together. |
| 3082 | Commands BEGIN, END, KILL, GO require axis specification | |
| 3083 | Array requires axis specification | |
| 3084 | Illegal array command | |
| 3085 | Extra number after the command | The command specifies a superfluous numerical argument. |
| 3086 | Variable name must be specified | The command requires a variable name specification. |
| 3087 | Command cannot be executed while the axis is moving | The command specifies one or more axes that are involved in a motion. The command can be executed only after the motion finishes. |
| 3088 | Variable can be queried only in compiled buffer | The command attempts to query a variable in a buffer that was not compiled. |
| 3089 | Label can be referenced only in compiled buffer | The command attempts to reference a label in a buffer that was not compiled. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3090 | This type of motion is not allowed for grouped axis | The slave or track command specifies an axis that is included in a user-defined group. Only a single axis can be specified by SLAVE or TRACK . |
| 3091 | Less arguments than required | The motion command specifies less coordinate values than required by the axis specification. |
| 3092 | More arguments than required | The motion command specifies more coordinate values than required by the axis specification. |
| 3093 | Bit selector value is greater than 31 or less than 0 | The expression specified in the bit selector yields a value greater than 31 or less than 0. |
| 3094 | Empty line range is specified | |
| 3095 | No master-slave motion is in progress | |
| 3096 | '}' was not found | The command includes a non-paired left curly bracket. |
| 3097 | Previous data collection is in progress | The command attempts to start a data collection while the previous data collection for the same axis is still in progress. |
| 3098 | Stalled motion requires limits specification | The motion command, which includes a command option of stalled motion, requires specification of the motion limits. |
| 3099 | Extra numbers after the command | The motion command includes superfluous numerical arguments. |
| 3100 | Received command has no message ID | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3101 | The program is suspended, the start line number is not allowed | <p>The command attempts to start a program from a specified line when the program is in suspended state.</p> <p>A program in suspended state can be only resumed from the next line. The line specification is not allowed.</p> <p>The only way to start a suspended program from a specific line is to stop the program, and to start it again from the desired line.</p> |
| 3102 | Zero divide | |
| 3103 | Invalid reference | |
| 3104 | No ACSPL program is waiting for input | |
| 3105 | Format error | The command is incorrectly formatted. |
| 3106 | SP function failed | <p>The function that accesses SP memory cannot be executed.</p> <p>Check the address specified in the function call. The address must fall in the range of 0 to 511.</p> |
| 3107 | Current empty space in the dynamic buffer is not enough for the command | |
| 3108 | Invalid real number | The command includes specification of a real number that cannot be interpreted as a valid real number. |
| 3109 | The command is not allowed in SAFE mode | |
| 3110 | At least one variable must be specified for data collection | |
| 3111 | Too long reply requested | |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3112 | No matches were found | The search command did not find any matches. |
| 3113 | The step in the table is zero or negative | |
| 3114 | The program finished without a STOP command | The program that is running in a buffer executed the last command, and it was not a STOP/STOPALL command. |
| 3115 | Stack underflow (RET without CALL) | The program that runs in a buffer caused a stack underflow. Program execution requires CALL . |
| 3116 | Stack overflow (too many nested CALLs) | The program that is running in a buffer caused stack overflow. This occurs if the program executes too many nested calls. Check that the program does not specify infinite recursion (subroutine is in an infinite loop). |
| 3117 | Attempt to enter autoroutine directly | The program that is running in a buffer comes to the ON command (see ON...RET). ON must never be executed in the normal program flow. |
| 3118 | Illegal axis number | The command specifies an axis by number or expression, and the resulting value is not a legal axis number. Valid axis numbers range between 0 to the number of axes in the system minus one. |
| 3119 | Integer overflow | |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3120 | The motion must involve the first two axes from the group | The segmented motion must always involve two axes. If a user-defined group contains more than two axes, the segmented motion must involve the first two axes in the group. |
| 3121 | Unknown #-constant | The command specifies an unknown symbolic constant. |
| 3122 | Bit selection is not applicable | Bit selector cannot be applied to a real value. |
| 3123 | Illegal bit selector | Value of bit selector must fall into the range 0..32. |
| 3124 | Attempt to enable motor failed | ENABLE/ENABLE ALL failed. Additional information can be obtained from the corresponding element of the MERR variable (motor disable code). |
| 3125 | Error in SP program | The command caused unsuccessful loading of an SP program. The file with the SP program contains an error. |
| 3126 | Illegal SP number | The command specifies an illegal SP number. Legal SP numbers are from 0 to 3. |
| 3127 | Editing of this buffer is disabled | The command attempts to open the buffer for editing or to change the program in the buffer. Editing of a buffer is disabled if bit 1 of the corresponding element of PFLAGS = 1. |
| 3128 | Configuration changed. Commands SAVE and HWRES must be executed. | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3129 | In binary command the name specification must end with / | The command syntax requires a name to be followed by the / (slash) character. |
| 3130 | Segment sequence for the previous motion was not terminated with ENDS command | MPTP , MSEG , PATH are followed by a number of the point or segment commands - segment definition sequence. The segment definition sequence must be closed with ENDS. See MPTP...ENDS , MSEG...ENDS , and PATH...ENDS . |
| 3131 | SP program is incompatible with one or more products | |
| 3132 | The file is not a legal user data file | READ specifies a file in the flash memory that is not a legal user data file. Only files created with WRITE are legal user data files. |
| 3133 | Discrepancy in types of the saved and restored arrays | The array and the user data file specified by READ contain data of different types. READ must specify an array of the same type and dimension as the array that was specified by the WRITE command that created the file. For example, the error occurs if a real array was saved in the file by WRITE , but READ tries to load the file into an integer array. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3134 | Discrepancy in sizes of the saved and restored arrays | The array and the user data file specified by READ are different sizes. READ must specify an array of the same type and dimension as the array that was specified by the WRITE command that created the file. |
| 3135 | Operation failed because of communication overload | |
| 3136 | Wrong relation between first point, last point and interval | The interval value specified by or does not correspond to the start and final points. For example, the error occurs if the final point is less than the start point, but the step is positive. |
| 3137 | Illegal analog output number | The command specifies a number of analog outputs which is not available in the controller. |
| 3138 | Incompatible SP and analog output | The command specifies an analog output number that cannot be accessed in the specified SP. |
| 3139 | Illegal input | The controller tries to interpret the string as a response to the executed input command but the string does not follow the required format. |
| 3140 | The function is not supported | |
| 3141 | Timeout | |
| 3142 | Arguments are inconsistent | Check arguments against the specifications for the or commands. |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3143 | Memory overflow | |
| 3144 | Simulator does not support this command | The command cannot be executed by the simulator. |
| 3145 | The specified DPR address is less than 128 or exceeds the DPR size | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 3146 | Collision with other variable in DPR | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |
| 3147 | Incomplete command (intrusion of other process?) | |
| 3148 | Requested more SINCOS encoder multipliers than installed | |
| 3149 | Illegal SP address | The SP address must fall in the range of from 0 to 512. |
| 3150 | Only even numbers are allowed as DPR address | SPiiPlus does not contain a dual-port RAM. If the program has been imported from an older non-NT version and this error appears, delete any DPR command (such as COPYFROMDPR or COPYTODPR). |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3151 | This is a DEMO version of Simulator. 5 minutes of work remains. | The demo version of the Simulator has a limited session time. The simulator is going to stop after five minutes. |
| 3152 | The DEMO version of Simulator has terminated | The demo version of the Simulator has a limited session time. The session time has elapsed. |
| 3153 | Illegal query command | |
| 3154 | The command can be only used with MSEG motion | Only PROJECTION , LINE , ARC1 , ARC2 , and STOPPER commands apply to MSEG...ENDS . |
| 3155 | Motion cannot start because the previous motion failed. Use FCLEAR command. | In the Strict mode (S_FLAGS.#FCLEAR = 1) the motion cannot start after a fault has occurred. Use the command FCLEAR to clear the result of the fault. |
| 3156 | Profile key must be specified as /SECTION/KEY/ | |
| 3157 | Illegal configuration string. Use only characters K,D,+,-. | |
| 3158 | Cannot find matching value. The formula has no root or the root is not single. | |
| 3159 | Axis number is specified more than once | |
| 3160 | The axis cannot be used in a group (see AFLAGS) | |
| 3161 | Illegal communication channel | |
| 3162 | Illegal tag value | |
| 3163 | Illegal configuration parameters | |
| 3164 | Illegal password | |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------|---------|
| 3165 | Attempt to execute optional function which is not installed | |
| 3166 | Flash file operation failed (overlapped file operations?) | |
| 3167 | Wrong start position is specified | |
| 3168 | File not opened | |
| 3169 | D-Buffer cannot be changed while any other buffer is compiled | |
| 3171 | The number of axes in coordinated motion is restricted to 4 | |
| 3172 | Illegal category name | |
| 3173 | EtherCAT offset is out of range | |
| 3174 | Can't find EtherCAT network variable | |
| 3175 | EtherCAT Master is not ready | |
| 3176 | Data size should be specified | |
| 3177 | This slave has no mailbox support | |
| 3178 | Invalid CoE SDO parameter | |
| 3179 | EtherCAT slave is in invalid state for this operation | |
| 3180 | General EtherCAT error | |
| 3181 | EtherCAT Timeout error | |
| 3182 | Can't split non-existing axes group | |
| 3183 | Attempt to split group with active motion on | |
| 3184 | Mapped variable must be defined in D-Buffer | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------|---------------------------------------------------------------|
| 3185 | Number of connected EtherCAT devices is not covered by SW options | |
| 3186 | Can't initialize SP injection | |
| 3187 | There is an active injection on this SP | |
| 3188 | Can't enable motor, DIP switch settings are incorrect. | |
| 3189 | The command can only be used with XSEG motion | |
| 3190 | PLC option on this device is not enabled | |
| 3191 | PLC is already running | |
| 3192 | General PLC error | |
| 3193 | Wrong array size or type | |
| 3194 | Not allowed Encoder Type | |
| 3195 | Requested more absolute encoder multipliers than installed | |
| 3196 | Requested absolute encoder is not supported | |
| 3197 | The slave is incompatible with current controller configuration | |
| 3198 | Invalid encoder type | |
| 3199 | Invalid absolute encoder parameter type | |
| 3200 | JIT and Dynamic modes are not allowed for D-buffer | JIT and Dynamic modes are not allowed for D-buffer |
| 3201 | End-of-Sequence is illegal for this motion | The ends command cannot be specified for this type of motion. |
| 3202 | JIT and Dynamic modes are not allowed at the same time | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3203 | DEBUG mode is not allowed for JIT and Dynamic buffers | |
| 3204 | JIT and Dynamic modes require the buffer to be empty | |
| 3205 | Illegal zone number | |
| 3206 | Cannot change a constant variable | |
| 3207 | One or more arguments are out of range | |
| 3208 | Function ended without return statement | |
| 3209 | Entered a function not by calling, STOP statement might be missing | |
| 3210 | Incompatible matrix size | |
| 3211 | Too many segments | |
| 3212 | ARC arguments are inconsistent | ARC1 or ARC2 specify inconsistent arguments. The desired arc cannot be calculated. |
| 3213 | Stopper is prohibited for master-slave motion | STOPPER cannot be used with segmented MASTER - SLAVE motion. |
| 3214 | Adjacent stoppers are prohibited | Two adjacent STOPPER commands are not allowed. |
| 3215 | In cyclic path the first and the last points must coincide | In a cyclic segmented motion, the first point of the first segment must coincide with the last point of the last segment. |
| 3216 | Velocity is specified, but the motion command had no V command option | Velocity argument can be specified in POINT , LINE , ARC1 or ARC2 only if the corresponding motion command specifies /v . |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3217 | Segment of zero length | Segments of zero length are not allowed. |
| 3218 | ARC radius is too small | |
| 3219 | Specified motion delay is out of range | |
| 3220 | Command is incompatible with connect | |
| 3221 | 20 KHz motion generation incompatible with CTIME or number of axes | |
| 3222 | 20 KHz motion limitation reached | |
| 3223 | Time specified is not possible under current motion safety limitations | |
| 3224 | Offset doesn't exist | |
| 3225 | Microblaze monitor is not ready | |
| 3226 | Address couldn't be found | |
| 3227 | Maximum number of Registers was already added | |
| 3228 | Firmware is not properly installed | |
| 3229 | SLABITS does not match Single Turn and Multi Turn number of bits | |
| 3230 | Non Linear Control Feature is not supported | |
| 3231 | Illegal key | The Key argument of the GETCONF or SETCONF command specifies a value that is not supported in this controller. |
| 3232 | Illegal index | The Index argument of the GETCONF or SETCONF command specifies a value that is not supported in this controller. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| 3233 | Illegal value | The Value argument of the SETCONF command specifies a value that is not compatible with the Key argument. |
| 3234 | Value in SETCONF must be 0 or 1 | For the specified Key only 0 or 1 are legal in the Value argument. of SETCONF |
| 3235 | SETCONF function is disabled | The specified SETCONF function is disabled in the current controller mode. |
| 3236 | SETCONF cannot be executed while the motor is enabled | The specified SETCONF function cannot be executed while the motor is enabled. |
| 3238 | The operation can be executed only when the HSSI channel is in command mode | |
| 3239 | HSSI channel cannot switch to command mode because it affects one or more remote drivers. Reset corresponding MFLAGS.#HSSI bits before. | |
| 3240 | Operation is temporarily disabled | |
| 3241 | Operation failed | |
| 3242 | The feature is not supported | |
| 3243 | Max number of 20 KHz moving axes reached | |
| 3244 | Illegal number of coordinates specified | |
| 3245 | The specified channel is already being used. Please use the MBCLOSE function to close the connection | |
| 3251 | Operation is temporarily disabled | |
| 3252 | Motor cannot be enabled while a motion is in termination process | |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3253 | Unable to work with Dummy motor | The command cannot be executed for a dummy motor. The addressed motor is configured as a dummy, (see the #DUMMY bit of MFLAGS). |
| 3254 | The operation requires the motor to be enabled | The command can be executed only while the motor is enabled. |
| 3255 | The operation requires the motor to be disabled | The command can be executed only while the motor is disabled. |
| 3256 | The operation is valid only for brushless motor without Hall sensor | The command cannot be executed for any motor type other than brushless, and only for brushless motors that do not have a Hall sensor. |
| 3257 | The operation failed because the brushless motor is not commutated | The command cannot be executed because the defined brushless motor is not commutated. The #BRUSHOK bit of MFLAGS reflects the state of the brushless commutation. To commutate the motor, execute COMMUT or a specific initialization program. |
| 3258 | The operation failed because the motor is in open-loop mode | The command cannot be executed because the motor is in open-loop state. The open-loop mode may be set using the #OPEN bit of MFLAGS . |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3259 | Motion cannot start because the motor is defined as dummy | <p>The motion command cannot be executed because one or more of the motors involved are dummy.</p> <p>Motor dummy state is defined by the #DUMMY bit of MFLAGS.</p> |
| 3260 | Motion cannot start because the motor is disabled | <p>The motion command cannot be executed because one or more of the motors involved are disabled.</p> |
| 3261 | Motion cannot start because the brushless motor is not commutated | <p>The command cannot be executed because the defined brushless motor is not commutated.</p> <p>The #BRUSHOK bit of MFLAGS reflects the state of the brushless commutation.</p> <p>To commutate the motor, execute COMMUT or a specific initialization program.</p> |
| 3262 | Motion cannot start because the motor is in open-loop mode | <p>The motion command cannot be executed because one or more of the motors involved are in open-loop state.</p> <p>Open-loop state is defined by bit MST.#OPEN.</p> <p>The open-loop mode may be set using the #OPEN bit of MFLAGS.</p> |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3263 | Motion cannot start because the previous motion failed. Use FCLEAR command. | <p>The motion command cannot execute because the previous motion for one or more motors failed and the controller is set to the Strict mode.</p> <p>The Strict mode is defined by bit <code>S_FLAGS.#FCLEAR</code>.</p> <p>Use <code>FCLEAR</code> to clear the result of the fault.</p> |
| 3265 | SP program does not support this operation | <p>The command is not supported by the current version of the controller.</p> <p>Specifically, the SP program does not support the operation. Check if the SP program has been changed.</p> |
| 3266 | Invalid PEG pulse width | |
| 3267 | Maximal number of time-based PEG pulses is 65,535 | |
| 3268 | Invalid period of time-based PEG pulses | |
| 3269 | PEG pulse width must be less than time-based pulse period | |
| 3270 | PEG is not supported for the specified axis | |
| 3271 | Step does not agree with start/end points | |
| 3272 | Incremental PEG step must correspond to encoder counts within the $(-2^{31}, 2^{31}-1)$ range, zero is excluded | |
| 3273 | The specified axis does not exist in this controller model | |
| 3274 | The specified axis is defined as dummy only | |
| 3275 | Only stepper motor is supported for the specified axis | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------------------|---------|
| 3276 | The brushless motor is not supported for the specified axis | |
| 3277 | Dual loop control is not supported for the specified axis | |
| 3278 | Remote HSSI driver is not supported for the specified axis | |
| 3279 | PEG states are not supported for the specified axis | |
| 3280 | The stepper motor is not supported for the specified axis | |
| 3281 | No Hall support for 2-Phase motors | |
| 3282 | Value out of range | |
| 3283 | Number of points for Random PEG exceeds the limit | |
| 3284 | Axis is not connected to PEG engine | |
| 3285 | NanoMotion Piezoceramic Motor is not supported for the specified axis | |
| 3286 | Fast Sin-Cos Encoder is not supported for the specified axis | |
| 3287 | Laser Modes are not supported for the specified axis | |
| 3288 | Time-based PEG is not supported for the specified axis | |
| 3289 | This function is not supported for the specified axis | |
| 3290 | SPINJECT/SPRT and Fast GPRT cannot be used in parallel | |
| 3291 | P/D interface is required for this function | |
| 3292 | Secondary Feedback is not supported | |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------------------------|---------|
| 3293 | SLABITS does not match Single Turn and Multi Turn number of bits | |
| 3294 | Motion cannot start because the Gantry Complementary brushless motor is not commutated | |
| 3300 | Non Linear Control License is required | |
| 3301 | Another EtherCAT port is already closed | |
| 3302 | Not possible to save network topology configuration when more than one line break exists | |
| 3303 | ERROR SDO: Object does not exist in the object dictionary. | |
| 3304 | The value is too high (check E_PAR_D value) | |
| 3305 | The network variable is already mapped to ACSPL+ variable | |
| 3306 | Invalid axis list | |
| 3307 | FoE Protocol is not supported by Slave | |
| 3308 | The Disk is full or the file is too big | |
| 3309 | Function supported only by CiA402 Drive | |
| 3310 | CiA402 Drive is not in OP State. PDO is not enabled. | |
| 3311 | Cannot convert REAL type to INT | |
| 3312 | More than one hold command is not allowed | |
| 3313 | Continue command is already in process | |
| 3314 | Requested Homing Method is not supported | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------|---------|
| 3316 | Homing in Gantry mode requires additional parameters | |
| 3317 | FoE Error: Access denied | |
| 3318 | Matrix is not invertible | |
| 3319 | PI AME interface is not initialized | |
| 3320 | PI AME interface is busy | |
| 3325 | Main FPGA file downloading is in progress | |
| 3326 | Operation aborted, PEG is in process | |
| 3327 | FPGA versions do not match | |
| 3328 | FPGA version is the same. FPGA Upgrade aborted. | |
| 3329 | Invalid value, digital input index should range between 0-99 and bit index should range between 0-31 | |
| 3330 | Invalid value, digital output index should range between 0-99 and bit index should range between 0-31 | |
| 3331 | This output is already mapped as a mechanical brake to a different axis | |
| 3332 | Hardware limit swapping (MFLAGSX.#HLIMSWAP) and limit routing are mutually exclusive | |
| 3333 | File name MAX length is 100 chars | |
| 3335 | Illegal SPATH specification | |
| 3336 | Nurbs motion was not initiated | |
| 3337 | Illegal NURBS/SPATH parameter value | |
| 3338 | Profile has ended or last knots already added | |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 3339 | Invalid number of axes | |
| 3340 | Dummy point must also specify a Knot value | |
| 3341 | Illegal dummy point, two points are allowed before coordinates and two after | |
| 3342 | The measurement you are trying to activate is already active. (see DPM_Measurement Stop() function for more details) | |
| 3343 | An invalid sample type has been specified. Type can either be 0 or 1. (see DPM_Measurement sample type for more details) | |
| 3344 | The sampling time specified is invalid, the minimum value is the controller cycle time (?CTIME) and the maximum value is: 100000000msec | |
| 3345 | The sample-set size specified is invalid. Minimum size: 1. Maximum size: 512. | |
| 3346 | The specified "when_to_measure" flag is invalid. (see DPM_Measurement documentation for more details) | |
| 3347 | The measurement you are trying to activate is paused. (see DPM_Measurement Stop()/Resume() functions for more details) | |
| 3348 | Cannot change the selected axis while monitoring is ON. (see DPM_Motion_Status SelectAxis() / MonitorOff() functions for more details) | |
| 3349 | The specified " monitored_variable " is invalid. (see DPM_Measurement documentation for more details) | |
| 3350 | Illegal attempt to copy STRING | |

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 3360 | LCI unit initialization failed | |
| 3361 | Unable to start the LCI operation | |
| 3362 | Trajectory axes are not defined | |
| 3363 | LCI Safety input is off | |
| 3364 | LCI laser fault input is on | |
| 3365 | LCI supported velocity limit exceeded | |
| 3366 | LCI unit not found | |
| 3367 | LCI. Unable to allocate channel for specified operation | |
| 3368 | LCI function timeout | |
| 3375 | The specified IP address is invalid. Please see MBOPEN documentation for more details. | |
| 3376 | The specified server ID is invalid (valid values are between [1 - 247]). Please see MBOPEN documentation for more details. | |
| 3377 | The specified word-order is invalid. (valid values are 0 for big-endian and 1 for little-endian) Please see MBOPEN documentation for more details. | |
| 3378 | Unable to open more than 3 Modbus connections at any given time. Please see MBOPEN documentation for more details. | |
| 3379 | The specified communication handle is invalid. | |
| 3380 | A connection with the specified IP was not found. | |
| 3381 | The specified address is invalid | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------------------------------------|---------|
| 3382 | The specified request frequency is invalid. the minimum value is 5 milliseconds. Please see the documentation for more details. | |
| 3383 | The specified variable length and the specified number of elements do not match. Please see the documentation for more details. | |
| 3384 | The specified suffix combination is invalid. Please see the documentation for more details. | |
| 3385 | Read-Only variables cannot be written. | |
| 3386 | The specified request ID is invalid. | |
| 3387 | The specified variable(or part of it) is already being written to by another Modbus request. Use #MBMAPREP for more details. | |
| 3388 | The specified server address is already being written to by another Modbus request. Use #MBMAPREP for more details. | |
| 3389 | The specified number of elements is greater than the specified variable length. Please see the documentation for more details. | |
| 3390 | The maximum number of integer values that can be "mapped" in a single request is 16. Please see the documentation for more details. | |
| 3391 | The maximum number of short values that can be "mapped" in a single request is 32. Please see the documentation for more details. | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 3392 | The maximum number of single-precision floating-point values that can be "mapped" in a single request is 16. Please see the documentation for more details. | |
| 3393 | The maximum number of double-precision floating-point values that can be "mapped" in a single request is 8. Please see the documentation for more details. | |
| 3394 | The maximum number of coil values that can be "mapped" in a single request is 32. Please see the documentation for more details. | |
| 3395 | The maximum number of discrete input values that can be "mapped" in a single request is 32. Please see the documentation for more details. | |
| 3396 | Failed to establish a connection with the specified server device. | |
| 3397 | The maximum number of mapping requests per server is 32. Please see the documentation for more details. | |
| 3398 | A connection with the specified IP is already open and with different configuration parameters. To change the connection parameters, close and open a new connection. | |
| 3399 | The specified Modbus channel operates in sequential mode. This mode does not support user-defined request frequency. Each request is sent after a response for the previous one has arrived, but no less than 10 milliseconds from the last time it has been sent. Please see the documentation for more details. | |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 3410 | The minimum value can not be greater than the maximum value. Please see the documentation for more details. | |
| 3411 | G-code: Run-time errors were detected during the program execution in simulation mode. Use the #LOGP <buffer number> command to present the detected errors. | |
| 3412 | This motion type is not supported | |
| 3413 | The supplied correction maps (2-dimensional arrays) should have the same dimensions. Please see the documentation for more details. | |
| 3414 | The Specified referenced axes/analog inputs are invalid. Please see the documentation for more details. | |
| 3415 | Internal error | |
| 3416 | Stage model dll is not loaded | |
| 3417 | Stage model dll's name does not match the stage model implemented by the dll | |
| 3418 | This functionality has not implemented yet | |
| 3419 | TCP and machine axes conflict. Consider to change either TCP axes by changing AXISDEF parameter or machine axes | |

7.4 Errors

Motion Termination error code values range between 5000 and 5150.



Codes from 5000 to 5008, however, do not indicate an error. They report normal motion termination.

Codes from 5009 and higher appear when a motion is terminated or a motor is disabled due to a fault detected by the controller.


When a motion terminates abnormally, or a motor is disabled, the error code is stored in the [MERR](#) variable. If there is an initialization fault during startup, the error code is stored in the [S_ERR](#) variable.

Table 6-4. ACSPL+ Motion Termination Errors

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5000 | Motion has not finished | Motion was terminated before reaching final point. |
| 5001 | Motion generation finished | The motion came to the final point and was successfully completed. |
| 5002 | Motion was killed by user | |
| 5003 | Motion was terminated by user | The motion was terminated by HALT before the final point was reached |
| 5004 | Motor was disabled by user | The motion was disabled by DISABLE/DISABLEALL . |
| 5005 | Motion was terminated because a motor was disabled | The motion was disabled by DISABLE/DISABLEALL . |
| 5006 | Motion was killed | The motion was terminated by KILL/KILLALL before the final point was reached |
| 5007 | Motor was disabled due to another motor becoming disabled | If a fault occurs in an axis disabling the motor thereby disabling other motors, code 5007 is stored in the MERR variable for all affected axes. |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5008 | Motion was killed due to another motion being killed | If a fault occurs in an axis killing the motion of a motor thereby killing the motion other motors, code 5008 is stored in the MERR variable for all affected axes. |
| 5009 | Common motion failed | |
| 5010 | Hardware Right Limit | The motion was killed because of a right limit fault. |
| 5011 | Hardware Left Limit | The motion was killed because of a left limit fault. |
| 5012 | Network Error | Check EtherCAT Cables for continuity, check ECERR variable, make sure no loose connections, Run System Viewer and Diagnostics and check Error Registers for possible CRC errors on specific network nodes to identify failure point. |
| 5014 | Motor Overheat | The motor was disabled or the motion failed because of an overheat fault. |
| 5015 | Software Right Limit | The motion is killed because of a software right limit fault. |
| 5016 | Software Left Limit | The motion is killed because of a software left limit fault.. |
| 5017 | Encoder 1 Not Connected | The motor was disabled because of an encoder not connected fault. Note how error is caught in hardware. Difference between A,A- or B,B- differential signal. If A=A- Error is thrown. Check continuity of cable pinouts. |
| 5018 | Encoder 2 Not Connected | The motor was disabled because of an encoder 2 not connected fault. Note how error is caught in hardware. Difference between A,A- or B,B- differential signal. If A=A- Error is thrown. Check continuity of cable pinouts. |
| 5019 | Drive Fault | The motor was disabled because of a drive alarm fault. |
| 5020 | Encoder 1 Error | The motor was disabled or the motion failed because of an encoder error fault. For Quadrature and SinCos encoders the signal monitors the logical |

| Error Code | Error Message | Remarks |
|------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | state transition between A,B signals. For contiuous motion the transition should be {[1,0],[1,1],[0,1],[0,0]}. If the transition ever skips states it indicates count loss. For Absolute encoders check configuration parameters. Problem could stem from signal contiuity or induced noise. Ensure head alignment and tape is clean. |
| 5021 | Encoder 2 Error | The motor was disabled or the motion failed because of an encoder 2 error fault. For Quadrature and SinCos encoders the signal monitors the logical state transition between A,B signals. For contiuous motion the transition should be {[1,0],[1,1],[0,1],[0,0]}. If the transition ever skips states it indicates count loss. For Absolute encoders check configuration parameters. Problem could stem from signal contiuity or induced noise. Ensure head alignment and tape is clean. |
| 5022 | Non-Critical Pos. Error | The motor was disabled or the motion failed because of a position error fault. |
| 5023 | Critical Position Error | The motor was disabled or the motion failed because of a critical position error fault. |
| 5024 | Velocity Limit | The motor was disabled or the motion failed because of a velocity limit fault. See ACSPL+ variable XVEL(axis) |
| 5025 | Acceleration Limit | The motor was disabled or the motion failed because of an acceleration limit fault. See ACSPL+ variable XACC(axis) |
| 5026 | Drive/Motor Overcurrent | The motor was disabled or the motion failed because of an overcurrent fault. Ensure stability of system. User can monitor servo processor variables: SP#(dsp#) Current Command axis axis _ number, or SP#(dsp#) Phase q current axis_number in the scope to establish when current ramps up. Error is triggered exceeding XCURI, XCURV, XRMSM, XRMSD values. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5027 | Servo Processor Alarm | <p>The motor was disabled or the motion failed because of a servo processor alarm fault.</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;">  <p>Activated when an axes does not have a physical drive associated to it.</p> </div> |
| 5028 | Safe Torque Off | The motor was disabled because STO not connected. |
| 5030 | HSSI Not Connected | |
| 5031 | Broken Wire | |
| 5032 | Attempt of motion while a fault is active | |
| 5033 | Attempt of motion in disabled direction | The motion is killed because of an attempt to move to left direction when the Left Limit or Left Software Limit fault is On, or move to right direction when the Right Limit or Right Software Limit fault is On. User can verify proper encoder count direction in "Verification" page in the Adjuster Wizard. |
| 5034 | MPU Overheat | See GETCONF(76,index) in Command and Variable Reference Guide |
| 5035 | Program Error | The motor was disabled or the motion failed because of a program error fault. |
| 5036 | Memory Overflow | The motor was disabled or the motion failed because of a memory overuse fault. |
| 5037 | MPU Overuse | The motor was disabled or the motion failed because of a time overuse fault. For case of XSEG use user should ensure each segment does not execute in less than minimum time given in the SpiiPlus Release notes section: "Controller Cycle Time (CTIME) Support" for the given controller. User can also monitor Usage related variables given in the SpiiPlus Command and Variable reference guide. |
| 5038 | Hardware Emergency Stop | The motor was disabled because of an emergency stop fault. |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5039 | Servo Interrupt | The motor was disabled or the motion failed because of a servo interrupt fault. |
| 5041 | Component Failure | The motor was disabled or the motion failed because of power supply failure or not supplied bus voltage |
| 5042 | Motor was disabled due to hall error | Motor was disabled due to hall error. User can monitor ACSPL variable SLSTHALL or use the hall monitor in "Verification" portion of the Adjuster Wizard to ensure all hall states are read. Check continuity of cable. |
| 5043 | Motion was killed due to external profile error | |
| 5044 | External Network Error | External network error |
| 5045 | Motor was disabled due to SYNC loss | Motor was disabled due to SYNC loss |
| 5046 | Motion was disabled due to GPRT overflow | Motor was disabled due to GPRT overflow |
| 5047 | The axis remains in HOLD state. | |
| 5048 | Motor Overcurrent | Ensure stability of system. User can monitor servo processor variables SP#(dsp#) Current Command axis axis_number , or SP#(dsp#) Phase q current axis_number in the scope to establish when current ramps up. Error is triggered exceeding XCURI, XCURV, XRMSM, XRMDS values. |
| 5049 | Drive Overcurrent | User can monitor servo processor variables: SP#(dsp#) Current Command axis axis_number , or SP#(dsp#) Phase q current axis_number in the scope to establish when current ramps up. Error is triggered exceeding XRMDS value. |
| 5050 | Safe Torque Off | STO Error |
| 5051 | Safe Torque Off: STO_1 24V | STO1 24V source disconnected |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5052 | Safe Torque Off: STO_2 24V | STO2 24V source disconnected |
| 5053 | Safe Torque Off: STO_1 24V and STO_2 24V | STO1 and STO2 24V sources disconnected |
| 5054 | Safe Torque Off: SS1_1 5V | SS1-1 5V source disconnected |
| 5055 | Safe Torque Off: SS1_2 5V | SS1-2 5V source disconnected |
| 5056 | Safe Torque Off: SS1_1 5V and SS1_2 5V | SS1-1 and SS1-2 5V sources disconnected |
| 5057 | Safe Torque Off: SS1 Timing Error | See Timing Error section in STO Application Note for more details. |
| 5060 | Driver Alarm | Drive alarm: No fault |
| 5061 | Driver Alarm: Short circuit | The motor was disabled or the motion failed because of a short circuit condition in the drive. Measure resistance of motor phases to ensure no physical short in motor or cable. Error is triggered by rapid increase in SP#(dsp#) Phase q current axis axis_number |
| 5062 | Driver Alarm: External protection activated | The motor was disabled or the motion failed because of an external protection condition in the drive. |
| 5063 | Driver Alarm: Power supply too low | The motor was disabled or the motion failed because of a power supply too low condition in the drive. |
| 5064 | Driver Alarm: Power supply too high | The motor was disabled or the motion failed because of a power supply too high condition in the drive. |
| 5065 | Driver Alarm: Temperature too high | The motor was disabled or the motion failed because of a power supply too high condition in the drive. |
| 5066 | Driver Alarm: Power supply 24VF1 | The motor was disabled or the motion failed because of a power supply 24VF1 condition in the drive. |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5067 | Driver Alarm: Power supply 24VF2 | The motor was disabled or the motion failed because of a power supply 24VF2 condition in the drive. |
| 5068 | Driver Alarm: Emergency stop | The motor was disabled or the motion failed because of an emergency stop condition in the drive. |
| 5069 | Driver Alarm: Power Down | The motor was disabled or the motion failed because of a power-down condition in the drive. |
| 5070 | Driver Alarm: Phase Loss | |
| 5071 | Driver Alarm: Drive not ready | |
| 5072 | Driver Alarm: Overcurrent | |
| 5073 | Driver Alarm: Reserved | |
| 5074 | Driver Alarm: Dumper fault | |
| 5075 | Driver Alarm: Digital Drive Interface not connected | The motor was disabled or the motion failed because of a Digital Drive Interface Not Connected condition in the drive. Check the cable connections to the drive. |
| 5076 | Driver Alarm: Drive Saturation | |
| 5080 | Component Failure | |
| 5081 | Component Failure: Power supply 0 fault | |
| 5082 | Component Failure: Power supply 1 fault | |
| 5083 | Component Failure: Regeneration module fault | |

| Error Code | Error Message | Remarks |
|------------|------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5084 | Component Failure: Power supply too high | The motor was disabled due to excessive high voltage of the Power Supply. To correct this, either reduce the motor speed. Or, in the case of PSM3U 8 Kw or 10 Kw being installed, connect an External Regeneration (see <i>MC4U Hardware Guide</i>). |
| 5085 | Component Failure: Temperature too high | The motor was disabled, the Power Supply temperature is too high. Check that the unit cooling fans are working, if not, replace the cooling fans. Make sure that the air flow around the unit is clear of any obstructions. |
| 5086 | Component Failure: Unknown error | |
| 5087 | Component Failure: Unknown error | |
| 5088 | Component Failure: Unknown error | |
| 5089 | Component Failure: Power down | Indicates missing 3 phase 230Vac servo supply voltage. Check that the AC input voltage is connected correctly. |
| 5090 | Component Failure: Drive supply phase lost | This applies only to 3-phase AC input power. In the event that one of the AC input supply phases is lost or one of the AC input fuses is blown all axis drivers which are supplied by this power supply are disabled. Make sure that the JP6 jumper is installed in the PSM3U-320V-XXkW board if a three- phase input supply is used. |
| 5091 | Component Failure: Power supply not ready. Try to enable axis again within 10 seconds | This is caused by the inrush power protection detector. It temporarily suspends the power input. Wait for a few seconds for the power to normalize before starting. |
| 5092 | Component Failure: Unknown error | |

| Error Code | Error Message | Remarks |
|------------|-----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5093 | Component Failure: Unknown error | |
| 5094 | Component Failure: Power supply damper not OK | There is a failure in the damper circuit in the Power Supply. There may be a short in the External Regeneration, or Internal Regeneration. |
| 5095 | Component Failure: Unknown error | |
| 5100 | Current bias measured is out of range | <p>At the beginning of the enable process the controller measures the current and calculates an average bias (offset). If the result is greater than the predefined maximum value (2% of maximum output), the controller generates the error and axis remains disabled. 1. The first thing to check is if the drive has STO. If 24V is not connected to STO inputs, the drive will always measure offsets of 100% (in case of our high voltage units). This is because the bootstrap mechanism needed to charge the current loop measurement circuits doesn't work (it requires transistors to be ON). This is the most common case of such fault.</p> <p>2. If that's not the case - Check if motor was moving during the enable process</p> <p>Try to disconnect the motor and see if the results are different</p> <p>3. Check the value of SLBIASA, SLBIASB. If non of the above, and it's more than 5% it usually indicates a problem. Usually it's much less. If you see very big values (>50%) and surely indicates a problem with the drive.</p> <p>If offsets are smaller than 10%, there is still a way to increase the threshold of the protection.</p> <p>SETCONF(207, axis, threshold in %) will allow you to increase the threshold of the "current bias out of range protection". Not recommended to use if offsets are too big,</p> <p>4. Don't disable the automatic measurement in this case. A very big offset indicates that something is wrong with the drive and better not try to work like that.</p> |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5101 | Autocommutation failure (phase error) | Error occurs if COMMUT fails. The error indicates wrong phase sequencing, or incorrect commutation parameters. To avoid this error it is recommended that the initial commutation be performed from the Adjuster. The Adjuster commutation program verifies the phase sequence and commutation parameters. This must be done before COMMUT is executed. |
| 5102 | Autocommutation failure (position error) | Error occurs if COMMUT fails. The error indicates bad or marginal servo tuning, or incorrect commutation parameters. To prevent this error make sure that the axis is properly tuned - use the Adjuster Wizard or FRF Analyzer . |
| 5103 | ENABLE is prohibited while Constant Current is ON | |
| 5104 | ENABLE failed because the motor is moving | ENABLE cannot be executed whenever the motor is in motion (see ENABLE/ENABLE ALL). |
| 5105 | E_TYPE does not match HSSI-HES DIP switch settings | Check the HSSI-HES DIP switch settings. |
| 5106 | EtherCAT node failure | |
| 5107 | The axis uses a licensed servo feature that is not allowed | |
| 5108 | Number of axes that use licensed servo feature exceeds the number allowed | |
| 5109 | Current bias measurements process was not completed | |
| 5110 | Command failed due to transaction timeout | |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------|---------|
| 5111 | ENABLE failed because the system need to be reconfigured | |
| 5121 | Encoder Error: CRC Error | |
| 5122 | Encoder Error: Busy | |
| 5123 | Encoder Error: Encoder Not Ready | |
| 5124 | Encoder Error: Timeout | |
| 5125 | Encoder Error: FPGA File not found | |
| 5126 | Encoder Error: Error Flag | |
| 5127 | Encoder Error: An internal incompatibility was detected. Full system upgrade is required. | |
| 5128 | Encoder Error: Watchdog | |

7.5 Encoder Errors

Table 6-5. Encoder Errors

| Error Code | Error Message | Remarks |
|------------|--------------------------|--------------------------------------------------------------------|
| 5121 | Encoder Error: CRC Error | Result of CRC check of the received value |
| 5122 | Encoder Error: Busy | Indicates that encoder is busy - specifics depend on encoder brand |

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| 5123 | Encoder Error: Encoder Not Ready | Endat: The status register is not completely updated. Not All checks have been performed. Data transmission is not yet completed. |
| 5124 | Encoder Error: Timeout | |
| 5125 | Encoder Error: FPGA File not found | FPGA file is missing, therefore, the Absolute Encoder for the required axis cannot be operated. |
| 5126 | Encoder Error: Error Flag | Error |
| 5127 | Encoder Error: An internal incompatibility was detected. Full system upgrade is required. | |
| 5128 | Encoder error: Watchdog | |

7.6 System Errors

System Errors error code values range between 5151 and 5999. They are generally caused by problems with the firmware of the servo processor (SP) program.

Table 6-6. ACSPL+ System Errors

| Error Code | Error Message |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5151 | Integrity of the firmware or user application is broken |
| 5152 | Initialization problem: Main Interrupt is not detected |
| 5154 | Initialization problem: Main Interrupt period is wrong |
| 5156 | Initialization problem: Main Interrupt Event cannot be created |
| 5158 | Initialization problem: SP initialization failed |
| 5160 | Initialization problem: SP program activation failed |
| 5162 | Initialization problem: SP and MP are not synchronous |
| 5163 | Initialization problem: EthernetIP initialization failed. |
| 5164 | Initialization problem: Improper controller card is detected |
| 5165 | Initialization problem: Improper configuration file is detected |
| 5166 | Initialization problem: SP is not detected by hardware |
| 5167 | Initialization problem: Number of drive modules does not meet the configuration |
| 5168 | Initialization problem: Drive Interface is not connected |
| 5169 | Initialization problem: One or more parameters are out of range for the current controller configuration (See #LOG for details). Default values are assigned |
| 5170 | Initialization problem: One or more EEPROM parameters are different from the current controller configuration. Values from EEPROM are assigned |
| 5171 | Initialization problem: One axis attached to two drives in configuration file |
| 5172 | Initialization problem: Nominal current parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |

| Error Code | Error Message |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5173 | Initialization problem: Peak current parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5174 | Initialization problem: Power parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5175 | Initialization problem: Voltage (min or max) parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5176 | Initialization problem: Type parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5177 | Initialization problem: Number of subsystems parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5178 | Initialization problem: Component was detected by I2C but was not found in configuration file |
| 5179 | Initialization problem: Slave address parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5180 | Initialization problem: Drive number parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5181 | Initialization problem: RMS protection time constant parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5183 | Initialization problem: Number of axes parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5184 | Initialization problem: One of digital inputs parameters in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5185 | Initialization problem: One of digital outputs parameters in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5186 | Initialization problem: One of analog inputs parameters in configuration file is different from the current controller configuration. Value from EEPROM is assigned |

| Error Code | Error Message |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5187 | Initialization problem: One of analog outputs parameters in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5188 | Initialization problem: Number of HSSI channels parameter in configuration file is different from the current controller configuration. Value from EEPROM is assigned |
| 5189 | Initialization problem: Unit ID is wrong. The unit definition is ignored |
| 5190 | Initialization problem: Number of nodes does not meet the configuration |
| 5191 | Initialization problem: Number of axes does not meet controller options |
| 5192 | Initialization problem: Incorrect DIP switch settings |
| 5193 | Initialization problem: Firmware is not properly installed |
| 5194 | Initialization problem: One or more EtherCAT nodes are incompatible with current controller configuration |
| 5195 | Initialization problem: One or more EtherCAT nodes are incompatible with current controller cycle time (CTIME) |
| 5196 | Initialization problem: One or more EtherCAT nodes are incompatible with Ring Topology |
| 5197 | Initialization problem: Cycle time of one or more EtherCAT nodes is different from the controller cycle time (CTIME). System should be reconfigured |
| 5198 | Initialization problem: One or more EtherCAT nodes are incompatible with ENI. System should be reconfigured |
| 5201 | Initialization problem: Wrong range of input parameters. Servo Processor program activation failed |
| 5202 | Initialization problem: File not found. Servo Processor program activation failed |
| 5203 | Initialization problem: Read file error. Servo Processor program activation failed |
| 5204 | Initialization problem: Memory allocation error. Servo Processor program activation failed |
| 5205 | Initialization problem: Checksum error. Servo Processor program activation failed |

| Error Code | Error Message |
|------------|-------------------------------------------------------------------------------------------|
| 5206 | Initialization problem: EtherCAT error. Servo Processor program activation failed |
| 5207 | Initialization problem: EtherCAT timeout. Servo Processor program activation failed |
| 5208 | Initialization problem: Servo Processor program is incompatible with one or more products |

7.7 EtherCAT Errors

EtherCAT errors range from 6000 to 6999 and are latched in the [ECERR](#) variable.

Table 6-7. ACSPL+ EtherCAT Errors

| Error Code | Error Message | Remarks |
|------------|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 6000 | General EtherCAT Error. Check #LOG output for more details. | General EtherCAT Error. Check #LOG output for more details. |
| 6001 | EtherCAT cable not connected | Check that the EtherCAT connections are firmly seated. |
| 6002 | EtherCAT master is in incorrect state | On start up all slaves did not succeed to initialize to full OP state. Can be caused by wrong configuration or a problem in a Slave |
| 6003 | Not all EtherCAT frames can be processed | The Master has detected that at least one frame that was sent has not returned. This implies a hardware problem in the cables or Slaves. |
| 6004 | EtherCAT Slave Error | Slave did not behave according to EtherCAT state machine – internal Slave failure. |
| 6005 | EtherCAT initialization failure | The EtherCAT-related hardware in the Master could not be initialized. Check the EtherCAT hardware. |
| 6006 | EtherCAT cannot complete the operation | The bus scan could not be completed. This implies hardware level problems in the EtherCAT network. |

| Error Code | Error Message | Remarks |
|------------|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6007 | EtherCAT work count error | Every Slave increments the working counter in the telegram. If this error is triggered, it means that a Slave has failed. Possible root cause: cable interruption, Slave reset, Slave hardware failure, |
| 6008 | Not all EtherCAT slaves are in OP state | One or more of the Slaves has changed its state to other than OP, or it may be due to a Slave restart or internal fault that internally forces the Slave to go to PREOP or SAFEOP. |
| 6009 | EtherCAT protocol timeout | The Master has detected that the Slave does not behave as expected for too long, and reports timeout. Implies a Slave hardware problem. Try power down, and system restart. |
| 6010 | Slave initialization failed | The Master cannot initialize a Slave by the configuration file. It can be caused by either wrong configuration, or a hardware problem in the Slave. |
| 6011 | Bus configuration mismatch | The bus topology differs from that in configuration file. |
| 6012 | CoE emergency | A Slave with CoE support has reported an emergency message. |
| 6013 | EtherCAT Master won't enter INIT state | Hardware fault, for example DHD with broken (logic) supply. |
| 6014 | EtherCAT ring topology requires network reconfiguration | System should be reconfigured to use the NetworkBoost feature. |
| 6015 | One or more EtherCAT cables are not connected | One or more EtherCAT cables are not connected; can happen only when using the NetworkBoost feature. |
| 6016 | The actual network configuration doesn't match the last approved configuration. | |

| Error Code | Error Message | Remarks |
|------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| 6017 | Change in configuration was detected. Optional Group was added. It should be approved. | |
| 6018 | EtherCAT Master won't enter PREOP state | EtherCAT Master is in INIT state and won't enter the PREOP state. |
| 6019 | EtherCAT Master won't enter SAFEOP state | EtherCAT Master is in PREOP state and won't enter the SAFEOP state. |
| 6020 | EtherCAT Master won't enter OP state | EtherCAT Master is in SAFEOP state and won't enter the OP state. |
| 6021 | Group ID mismatch between configuration file and actual configuration | |
| 6022 | Change in configuration was detected. Optional Group was removed. It should be approved. | |
| 6023 | EtherCAT cable(s) are crossed. Check proper cables connection from OUT port to IN port. | |
| 6024 | One or more EtherCAT slaves causes frames to be lost. Check proper cables connection from OUT port to IN port. | |

7.8 EtherCAT Slave Errors

EtherCAT Slave errors range from 7000 to 7999 and are latched in the [ECALERR](#) variable.

The error codes are defined according to AL Status Code (ETG 1020).

Table 6-8. EtherCAT Slave Errors

| ACS Error Code | Error Message | Remarks |
|----------------|--------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 7001 | General Error | General Error which is not defined in the following list |
| 7002 | Mailbox No Memory | Local Application runs out of memory (e.g. dynamic memory allocation for emergency messages) |
| 7003 | Invalid device setup | |
| 7004 | Unknown error | If the slave checks if the SII/EEPROM content matches the firmware, e.g. process data description or revision number, and detects a mismatch |
| 7006 | Firmware and EEPROM do not match. Slave needs BOOT-INIT transition | |
| 7007 | Firmware update not successful. Old firmware still running | Error occurred during firmware update |
| 7014 | License Error (HW/SW license invalid or evaluation period expired) | HW/SW license invalid or evaluation period expired |
| 7017 | Invalid requested state change | Requested state change is invalid |
| 7018 | Unknown requested state | Requested state change is unknown |
| 7019 | Bootstrap state is not supported by the slave | |
| 7020 | No valid firmware | |
| 7021 | Invalid Mailbox Configuration in Boot | The mailbox SyncManager configuration is not valid in Bootstrap state |

| ACS Error Code | Error Message | Remarks |
|----------------|-------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| 7022 | Invalid Mailbox Configuration in PREOP | The mailbox SyncManager configuration is not valid in PREOP state |
| 7023 | Invalid Sync Manager Configuration | Invalid sync manager configuration. Possible reason can be invalid PDO size configuration |
| 7024 | No valid inputs available | Slave application cannot provide valid inputs |
| 7025 | No valid outputs available | Slave application cannot provide valid outputs |
| 7026 | Synchronization error | Multiple synchronization errors. Device is not synchronized any more |
| 7027 | Sync Manager Watchdog | No process data received yet or not received within a specified timeout value |
| 7028 | Invalid Sync Manager Types | |
| 7029 | Invalid Sync Manager Output Configuration | SyncManager configuration for output process data is invalid (check ENI) |
| 7030 | Invalid Sync Manager Input Configuration | SyncManager configuration for input process data is invalid (check ENI) |
| 7031 | Invalid Watchdog | Watchdog Settings are invalid (e.g. SyncManger watchdog trigger is enabled but no watchdog timeout is defined) |
| 7032 | Slave needs cold start | Slave device requires a power off – power on reset |
| 7033 | Slave needs INIT | Slave application requests INIT state |
| 7034 | Slave needs PREOP | Slave application requests PREOP state |
| 7035 | Slave needs SAFEOP | Slave application requests SAFEOP state |

| ACS Error Code | Error Message | Remarks |
|----------------|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7036 | Invalid Input Mapping | Input process data mapping do not match to expected mapping |
| 7037 | Invalid Output Mapping | Output process data mapping do not match to expected mapping |
| 7038 | Inconsistent Settings | General settings mismatch |
| 7039 | Free Run Mode is Not Supported | Slave doesn't support Free Run Mode |
| 7040 | SyncMode Not Supported | Slave doesn't support SYNC mode |
| 7041 | Free Run Needs 3-Buffer Mode | FreeRun Mode, sync manager has to run in 3Buffer Mode |
| 7042 | Background watchdog | |
| 7043 | No Valid Inputs and Outputs | |
| 7044 | Fatal Sync Error | Fatal Sync Error: Sync0 or Sync1 are not received any more |
| 7045 | No Sync Error | Sync not received: In SAFEOP the slave waits for the first Sync0/Sync1 events before switching to OP, if these events were not received during the SAFEOP to OP-Timeout time the slave should refuse the state transition to OP with this AL Status Code (SystemTimeOffset too big, no DC event received) |
| 7046 | EtherCAT cycle time smaller than the minimum cycle time supported by the slave | |
| 7048 | Invalid DC SYNC Configuration | Distributed Clocks configuration is invalid due to application requirements |

| ACS Error Code | Error Message | Remarks |
|----------------|--------------------------------------|-------------------------------------------------------------------|
| 7049 | Invalid DC Latch Configuration | DC Latch configuration is invalid due to application requirements |
| 7050 | DC PLL Sync Error | Master not synchronized, at least one DC event received |
| 7051 | Invalid DC IO Error | Multiple synchronization errors. IO is not synchronized any more |
| 7052 | Invalid DC Timeout Error | Multiple synchronization errors. Too much SM Events missed |
| 7053 | DC Invalid Sync Cycle Time | |
| 7054 | DC Sync0 Cycle Time | DC Sync0 Cycle time does not fit to application requirements |
| 7055 | DC Sync1 Cycle Time | DC Sync1 Cycle time does not fit to application requirements |
| 7066 | Mailbox EoE | |
| 7067 | Mailbox CoE | |
| 7068 | Mailbox FoE | |
| 7080 | EEPROM No Access | EEPROM not assigned to PDI |
| 7081 | EEPROM Error | EEPROM access error |
| 7200 | FoE Error: Vendor specific FoE error | |
| 7201 | FoE Error: Not found | |
| 7202 | FoE Error: Access denied | |
| 7203 | FoE Error: Disk full | |
| 7204 | FoE Error: Illegal access | |
| 7205 | FoE Error: Wrong packet number | |

| ACS Error Code | Error Message | Remarks |
|----------------|----------------------------------------------------|---------|
| 7206 | FoE Error: Already exists | |
| 7207 | FoE Error: No user | |
| 7208 | FoE Error: Bootstrap state only | |
| 7209 | FoE Error: Not valid in Bootstrap state | |
| 7210 | FoE Error: No rights | |
| 7211 | FoE Error: Program error | |
| 7212 | FoE Error: Wrong checksum | |
| 7213 | FoE Error: Firmware is incompatible with Hardware | |
| 7214 | FoE Error: No file to read | |
| 7215 | FoE Error: File header does not exist | |
| 7216 | FoE Error: Flash problem | |
| 7217 | FoE Error: File incompatible | |
| 7218 | FoE Error: slave does not support FoE in Bootstrap | |
| 7219 | FoE Error: File is bigger than max file size | |
| | | |

| ACS Error Code | Error Message | Remarks |
|----------------|---------------|---------|
| | | |

7.9 MODBUS Errors

Table 6-9. Modbus Errors

| Error Code | Error Message | Remarks |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 8001 | Function code received in the query is not recognized or allowed by server. | |
| 8002 | Data address of some or all the required entities are not allowed or do not exist in server. | |
| 8003 | A value contained in the query data field is not an allowable value for the server. | |
| 8004 | An unrecoverable error occurred while the server was attempting to perform the requested action. | |
| 8005 | Server has accepted request and is processing it, but a long duration of time is required. | |
| 8006 | Server is engaged in processing a long-duration command. Client should retry later. | |
| 8012 | The received response transaction identifier did not match the expected value. | |
| 8013 | The received response length did not match the expected length. | |
| 8014 | The received response value is invalid. This may be due to the value being out of the allowed range, an attempt to write to a protected variable, etc. | |
| 8015 | No response has been received from the server device for the timeout period(5 seconds). The connection with the server device has been terminated. | |
| 8016 | The connection with the server device has been closed. | |

8. G-Code Error Codes

This chapter contains explanations of the Error Codes that may appear.

- > [G-Code Syntax Errors](#)
- > [G-Code Compilation Errors](#)
- > [G-Code Runtime Errors](#)

8.1 G-Code Syntax Errors

| Code | Meaning |
|------|--------------------------------------------------------------------------------------------------------------------------|
| 1500 | G-code: General G-Code runtime error |
| 1501 | G-code: Incompatible G commands in one statement |
| 1502 | G-code: The value should be positive |
| 1503 | G-code: G2/G3 can be specified in the main plane only |
| 1504 | G-code: An illegal arc with the current radius compensation parameters |
| 1505 | G-code: Segments of radius compensation should be separated by G0 or G1 segment |
| 1506 | G-code: Impossible case in radius compensation |
| 1507 | G-code: Too many S or R addresses |
| 1508 | G-code: Wrong digital output specification in S or R address |
| 1509 | G-code: Unsupported G function |
| 1510 | G-code: Unsupported M function |
| 1511 | G-code: I,J,K, or R specified outside arc definition -or- R specified outside cylindrical interpolation definition(G207) |
| 1512 | G-code: I,J,K and R cannot be specified together |
| 1513 | G-code: Wrong parameters in arc definition |

| Code | Meaning |
|------|----------------------------------------------------------------------------------------------------------|
| 1514 | G-code: Internal error |
| 1515 | G-code: Wrong index in gParamAddr/gParamValue |
| 1516 | G-code: gParamAddr/gParamValue used out of subroutine |
| 1517 | G-code: XSEG address specified outside XSEG Motion Parameterization (No G200) |
| 1518 | G-code: XSEG address specified outside XSEG Segment Parameterization (No G01, G02 or G03) |
| 1519 | G-code: XSEG address specified outside XSEG Motion/Segment Parameterization (No G200/No G01, G02 or G03) |
| 1520 | G-code: XSEG addresses - Cannot use (,Y) together with addresses (,J), (,A) & (,D) |
| 1521 | G-code: XSEG comma-addresses (,g), (,u) and (,h) cannot be specified together |
| 1522 | G-code: Cannot use XSEG motion parameters command G200 with any of G01, G02 or G03 |
| 1523 | G-code: XSEG addresses - Cannot use M61/M62 with addresses (,01) to (,04) |
| 1524 | G-code: XSEG addresses - At least both (,01) and (,02) addresses must be used on the same line |
| 1525 | G-code: XSEG addresses - Cannot use (,M) together with (,F) or (F) |
| 1526 | G-code: XSEG addresses - Cannot use (,M) together with (,L) |
| 1527 | G-code: Incompatible G with custom G208 in one statement |

| Code | Meaning |
|------|-------------------------------------------------------------------------------------------------------------------|
| 1528 | G-code: G208 should define at least two primary axes, according to active trajectory plane (G17/G18/G19) |
| 1529 | G-code: G208 should define minimum of 2 XSEG axes and maximum of 6 XSEG axes |
| 1530 | G-code: Segment processing time (T) option should be specified only with in-plane G01/G02/G03 |
| 1531 | G-code: Cannot use (T) option - Segment processing time, together with (,F) or (F) |
| 1532 | G-code: The value should be above a minimum value (see GSP Reference Guide) |
| 1533 | G-code: Incompatible G with custom G207/G205 in one statement |
| 1534 | G-code: G207 should define one linear axis followed by one rotational axis, and cylinder radius, in one statement |
| 1535 | G-code: G54 should define a WCS number between 1 to 12 |
| 1536 | G-code: GUFAC is out of range (1e-15, 1e15) or zero |
| 1537 | G-code: Radius should be positive |
| 1538 | G-code: Blended motion should not specify feedrate and segment time together |
| 1539 | G-code: Illegal parameter for Blended motion |
| 1540 | G-code: Mandatory parameter for BSEG not specified |
| 1541 | G-code: Bseg parameter must be greater than 0 |
| 1542 | G-code: G84 only allows rotation angle and center parameters to be specified |

| Code | Meaning |
|------|------------------------------------------------------------------------------------------|
| 1543 | G-code: Wrong M-code comand use |
| 1544 | G-code: Unmapped axis name used |
| 1545 | G-code: Tool height compensation is not allowed as second segment of Radius Compensation |
| 1546 | G-code: Incompatible mode to activate Local Coordinate System (G68) |
| 1547 | G-code: Too many G68 parameters |
| 1548 | G-code: Incompatible G code when Local Coordinate System is active |
| 1549 | G-code: Illegal case of segment synchronization |
| 1550 | G-code: Illegal NURBS based motion specification |
| 1551 | G-code: An incompatible address with NURBS/Path_Smoothing was specified |
| 1552 | G-code: An Illegal number of NURBS/Path_Smoothing main axes was specified |
| 1553 | G-code: Path Smoothing motion was not initiated |
| 1554 | G-code: Illegal Address for Path Smoothing |
| 1555 | G-code: Buffer is in Path Smoothing mode, please switch mode to use G06 command |
| 1556 | G-code: The address must contain a value |
| 1557 | G-code: The address must not contain a value |
| 1558 | G-code: SPath arc specification missing |

8.2 G-Code Compilation Errors

| Code | Meaning |
|------|----------------------------------------------------------------|
| 2500 | G-code: General G-Code compilation error |
| 2501 | G-code: Unknown G-Code command |
| 2502 | G-code: Value was expected |
| 2503 | G-code: Illegal syntax used |
| 2504 | G-code: Illegal value |
| 2505 | G-code: Address should not have a value |
| 2506 | G-code: Address value should be global or standard variable |
| 2507 | G-code: The output range specified is too big |
| 2508 | G-code: Wrong digital output specification in S or R address |
| 2509 | G-code: Comment lacks right parenthesis |
| 2510 | G-code: Illegal G specified |
| 2511 | G-code: Illegal M specified |
| 2512 | G-code: ')' expected |
| 2513 | G-code: '[' is not expected |
| 2514 | G-code: Address value should be a global variable |
| 2515 | G-code: All comments must start with '!' |
| 2516 | G-code: Addresses that are not allowed together were specified |
| 2517 | G-code: Illegal Synchronized code specification |

8.3 G-Code Runtime Errors

| Code | Meaning |
|------|--------------------------------------------------|
| 3500 | G-code: General G-Code runtime error |
| 3501 | G-code: Incompatible G commands in one statement |
| 3502 | G-code: The value should be positive |

| | |
|------|--------------------------------------------------------------------------------------------------------------------------|
| | |
| 3503 | G-code: G2/G3 can be specified in the main plane only |
| 3504 | G-code: An illegal arc with the current radius compensation parameters |
| 3505 | G-code: Segments of radius compensation should be separated by G0 or G1 segment |
| 3506 | G-code: Impossible case in radius compensation |
| 3507 | G-code: Too many S or R addresses |
| 3508 | G-code: Wrong digital output specification in S or R address |
| 3509 | G-code: Unsupported G function |
| 3510 | G-code: Unsupported M function |
| 3511 | G-code: I,J,K, or R specified outside arc definition -or- R specified outside cylindrical interpolation definition(G207) |
| 3512 | G-code: I,J,K and R cannot be specified together |
| 3513 | G-code: Wrong parameters in arc definition |
| 3514 | G-code: Internal error |
| 3515 | G-code: Wrong index in gParamAddr/gParamValue |
| 3516 | G-code: gParamAddr/gParamValue used out of subroutine |
| 3517 | G-code: XSEG address specified outside XSEG Motion Parameterization (No G200) |
| 3518 | G-code: XSEG address specified outside XSEG Segment Parameterization (No G01, G02 or G03) |
| 3519 | G-code: XSEG address specified outside XSEG Motion/Segment Parameterization (No G200/No G01, G02 or G03) |
| 3520 | G-code: XSEG addresses - Cannot use (,Y) together with addresses (,J), (,A) & (,D) |
| 3521 | G-code: XSEG comma-addresses (,g), (,u) and (,h) cannot be specified together |
| 3522 | G-code: Cannot use XSEG motion parameters command G200 with any of G01, G02 or G03 |

| | |
|------|-------------------------------------------------------------------------------------------------------------------|
| | |
| 3523 | G-code: XSEG addresses - Cannot use M61/M62 with addresses (,01) to (,04) |
| 3524 | G-code: XSEG addresses - At least both (,01) and (,02) addresses must be used on the same line |
| 3525 | G-code: XSEG addresses - Cannot use (,M) together with (,F) or (F) |
| 3526 | G-code: XSEG addresses - Cannot use (,M) together with (,L) |
| 3527 | G-code: Incompatible G with custom G208 in one statement |
| 3528 | G-code: G208 should define at least two primary axes, according to active trajectory plane (G17/G18/G19) |
| 3529 | G-code: G208 should define minimum of 2 XSEG axes and maximum of 6 XSEG axes |
| 3530 | G-code: Segment processing time (T) option should be specified only with in-plane G01/G02/G03 |
| 3531 | G-code: Cannot use (T) option - Segment processing time, together with (,F) or (F) |
| 3532 | G-code: The value should be above a minimum value (see GSP Reference Guide) |
| 3533 | G-code: Incompatible G with custom G207/G205 in one statement |
| 3534 | G-code: G207 should define one linear axis followed by one rotational axis, and cylinder radius, in one statement |
| 3535 | G-code: G54 should define a WCS number between 1 to 12 |
| 3536 | G-code: GUFAC is out of range (1e-15, 1e15) or zero |
| 3537 | G-code: Radius should be positive |
| 3538 | G-code: Blended motion should not specify feedrate and segment time together |
| 3539 | G-code: Illegal parameter for Blended motion |
| 3540 | G-code: Mandatory parameter for BSEG not specified |
| 3541 | G-code: Bseg parameter must be greater than 0 |
| 3542 | G-code: G84 only allows rotation angle and center parameters to be specified |

| | |
|------|------------------------------------------------------------------------------------------|
| 3543 | G-code: Wrong M-code comand use |
| 3544 | G-code: Unmapped axis name used |
| 3545 | G-code: Tool height compensation is not allowed as second segment of Radius Compensation |
| 3546 | G-code: Incompatible mode to activate Local Coordinate System (G68) |
| 3547 | G-code: Too many G68 parameters |
| 3548 | G-code: Incompatible G code when Local Coordinate System is active |
| 3549 | G-code: Illegal case of segment synchronization |
| 3550 | G-code: Illegal NURBS based motion specification |
| 3551 | G-code: An incompatible address with NURBS/Path_Smoothing was specified |
| 3552 | G-code: An Illegal number of NURBS/Path_Smoothing main axes was specified |
| 3553 | G-code: Path Smoothing motion was not initiated |
| 3554 | G-code: Illegal Address for Path Smoothing |
| 3555 | G-code: Buffer is in Path Smoothing mode, please switch mode to use G06 command |
| 3556 | G-code: The address must contain a value |
| 3557 | G-code: The address must not contain a value |
| 3558 | G-code: SPath arc specification missing |

Appendix A. PEG And MARK Mapping Tables

A.1 ASSIGNPEG Mapping

Mapping PEG engines to encoders

Table A-1. Mapping PEG Engines to Encoders (Servo Processor 0) for SPiiPlusNT/DC-LT/HP/LD

| Bit Code | Encoder 0(X) | Encoder 1(Y) | Encoder 2(A) | Encoder 3(B) |
|---------------|----------------------|----------------------|--------------|--------------|
| 000 (default) | PEGO | PEG1 | PEG2 | no |
| 001 | PEGO | PEG1 | no | PEG2 |
| 010 | PEGO PEG2 | PEG1 | no | no |
| 011 | PEGO | PEG1 PEG2 | no | no |
| 100 | PEGO PEG1 PEG2 | no | no | no |
| 101 | no | PEGO PEG1 PEG2 | no | no |

Table A-2. Mapping PEG Engines to Encoders (Servo Processor 1) for SPiiPlusNT/DC-LT/HP/LD

| Bit Code | Encoder 4(Z) | Encoder 5(T) | Encoder 6(C) | Encoder 7(D) |
|---------------|----------------------|----------------------|--------------|--------------|
| 000 (default) | PEG4 | PEG5 | PEG6 | no |
| 001 | PEG4 | PEG5 | no | PEG6 |
| 010 | PEG4 PEG6 | PEG5 | no | no |
| 011 | PEG4 | PEG5 PEG6 | no | no |
| 100 | PEG4 PEG5 PEG6 | no | no | no |
| 101 | no | PEG4 PEG5 PEG6 | no | no |

**Table A-3. Mapping PEG Engines to Encoders (Servo Processor 0) for SPiiPlus
 CMnt/CMhv/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/UDMhv/UDMnt/UDMpa/UDMpm/UDMpc/UDMcb**

| Bit Code | Encoder 0(X) | Encoder 1(Y) | Encoder 2(A) | Encoder 3(B) |
|---------------|-----------------------------------|-----------------------------------|-------------------------------------------------------------------|---------------------|
| 000 (default) | PEGO | PEG1 | PEG2 ² | |
| 001 | PEGO | PEG1 | no | PEG2 ^{1,2} |
| 010 | PEGO PEG2 ² | PEG1 | no | |
| 011 | PEGO | PEG1 PEG2 ² | no | |
| 100 | PEGO PEG1 PEG2 ² | no | no | |
| 101 | no | PEGO PEG1 PEG2 ² | no | |
| 110 | | | PEGO ^{1,2} PEG1 ^{1,2} PEG2 ^{1,2} | |

| Bit Code | Encoder 0(X) | Encoder 1(Y) | Encoder 2(A) | Encoder 3(B) |
|----------|--------------|--------------|--------------------------------------------|---------------------|
| 111 | | | PEGO ^{1,2} PEG2 ^{1,2} | PEG1 ^{1,2} |

¹ These combinations are **not** supported by UDMpc-x.

² These combinations are **not** supported by UDMnt-x.

Table A-4. Mapping PEG Engines to Encoders (Servo Processor 0) for UDMlc/UDlIt/UDlhp/UDMmc/PDlcl

| Bit Code | Encoder 0 | Encoder 1 | Encoder 2 | Encoder 3 |
|---------------|-----------|-----------|-----------|-------------------|
| 000 (default) | PEGO | no | no | no |
| 001 | no | PEGO | no | no |
| 010 | no | no | PEGO | no |
| 011 | no | no | no | PEGO ¹ |
| 100 | no | no | no | no |
| 101 | no | no | no | no |
| 110 | no | no | no | no |

¹ These combinations are **not** supported by LCM-x.

Table A-5. Mapping PEG Engines to Encoders (Servo Processor 0) for NPMpm/NPMpc-

| Bit Code | Encoder 0 | Encoder 1 | Encoder 2 | Encoder 3 |
|---------------|--------------|--------------|-----------|-----------|
| 000 (default) | PEGO | PEG1 | no | no |
| 001 | PEG1 | PEGO | no | no |
| 010 | PEGO PEG1 | no | no | no |
| 011 | no | PEGO PEG1 | no | no |
| 100 | no | PEG1 | PEGO | no |
| 101 | no | PEG1 | no | PEGO |
| 110 | PEGO | no | PEG1 | no |
| 111 | PEGO | no | no | PEG1 |

Bit Code assignment example

Assume the system includes two Servo Processors:

- > Servo Processor (node) 0: CMba with 4 encoders (0,1,2,3) and 3 PEG engines (0,1,2).
- > Servo Processor (node) 1: NPMpm with 4 encoders (4,5,6) and 2 PEG engines (4,5).

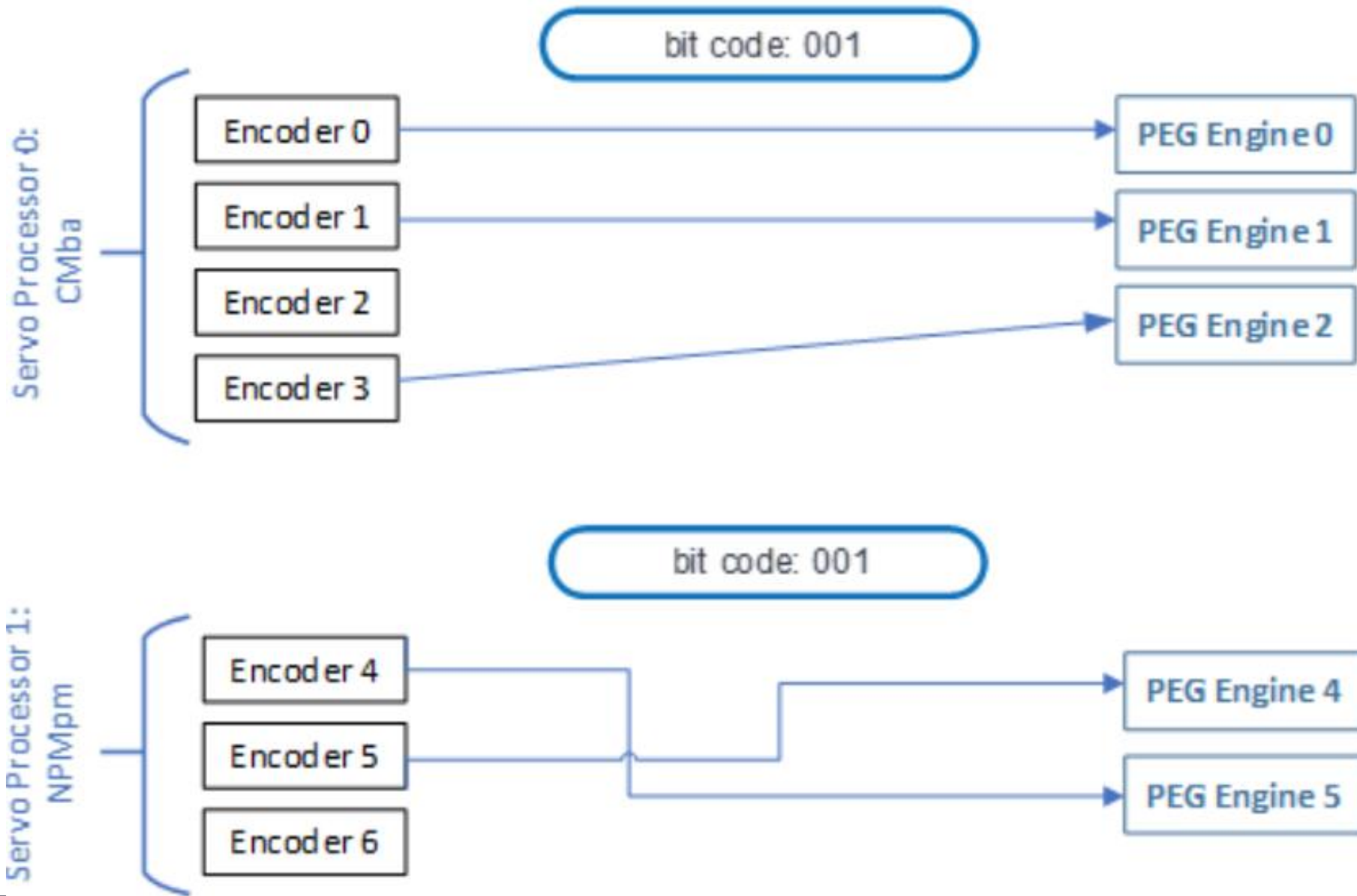
The bit code 001 for an axis associated with Servo Processor 0 (CMba) performs the following assignment:

- > PEG 0 engine is triggered by Encoder 0.

- > PEG 1 engine is triggered by Encoder 1.
- > PEG 2 engine is triggered by Encoder 3.

The bit code 001 for an axis associated with Servo Processor 1 (NPMpm) performs the following assignment:

- > PEG 4 engine is triggered by Encoder 5.
- > PEG 5 engine is triggered by Encoder 4.



General purpose outputs assignment to use as PEG pulse outputs

Table A-6. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) for SPiiPlusNT/DC-LT/HP/LD

| Bit Code | GP Out 0 | GP Out 1 | GP Out 2 | GP Out 3 |
|----------------|------------|------------|------------|----------|
| 0000 (default) | GP Out 0 | GP Out 1 | GP Out 2 | GP Out 3 |
| 0001 | PEGO_PULSE | GP Out 1 | GP Out 2 | GP Out 3 |
| 0010 | GP Out 0 | PEG2_PULSE | GP Out 2 | GP Out 3 |
| 0011 | GP Out 0 | GP Out 1 | PEG1_PULSE | GP Out 3 |
| 0100 | GP Out 0 | GP Out 1 | GP Out 2 | Reserved |
| 0101 | GP Out 0 | PEG2_PULSE | GP Out 2 | Reserved |
| 0110 | PEGO_PULSE | GP Out 1 | PEG1_PULSE | GP Out 3 |
| 0111 | PEGO_PULSE | PEG2_PULSE | PEG1_PULSE | Reserved |
| 1000 - 1111 | Reserved | Reserved | Reserved | Reserved |

Table A-7. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1) for SPiiPlusNT/DC-LT/HP/LD

| Bit Code | GP Out 4 | GP Out 5 | GP Out 6 | GP Out 7 |
|----------------|------------|----------|----------|----------|
| 0000 (default) | GP Out 4 | GP Out 5 | GP Out 6 | GP Out 7 |
| 0001 | PEG4_PULSE | GP Out 5 | GP Out 6 | GP Out 7 |

| Bit Code | GP Out 4 | GP Out 5 | GP Out 6 | GP Out 7 |
|-------------|------------|------------|------------|----------|
| 0010 | GP Out 4 | PEG6_PULSE | GP Out 6 | GP Out 7 |
| 0011 | GP Out 4 | GP Out 5 | PEG5_PULSE | GP Out 7 |
| 0100 | GP Out 4 | GP Out 5 | GP Out 6 | Reserved |
| 0101 | GP Out 4 | PEG6_PULSE | GP Out 6 | Reserved |
| 0110 | PEG4_PULSE | GP Out 5 | PEG5_PULSE | GP Out 7 |
| 0111 | PEG4_PULSE | PEG6_PULSE | PEG5_PULSE | Reserved |
| 1000 - 1111 | Reserved | Reserved | Reserved | Reserved |

Table A-8. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) for SPiiPlus CMnt/UDMpm/CMhv/UDMhv-

| Bit Code | GP Out 0 | GP Out 1 | GP Out 2 | GP Out 3 |
|----------------|------------|------------|------------|----------|
| 0000 (default) | GP Out 0 | GP Out 1 | GP Out 2 | GP Out 3 |
| 0001 | PEG0_PULSE | GP Out 1 | GP Out 2 | GP Out 3 |
| 0010 | GP Out 0 | PEG1_PULSE | GP Out 2 | GP Out 3 |
| 0011 | GP Out 0 | GP Out 1 | PEG2_PULSE | GP Out 3 |
| 0100 | GP Out 0 | GP Out 1 | GP Out 2 | GP Out 3 |

| Bit Code | GP Out 0 | GP Out 1 | GP Out 2 | GP Out 3 |
|-------------|------------|------------|------------|----------|
| 0101 | GP Out 0 | PEG1_PULSE | GP Out 2 | GP Out 3 |
| 0110 | PEGO_PULSE | GP Out 1 | PEG2_PULSE | GP Out 3 |
| 0111 | PEGO_PULSE | PEG1_PULSE | PEG2_PULSE | GP Out 3 |
| 1000 - 1111 | Reserved | Reserved | Reserved | Reserved |

Table A-9. General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) for UDMnt/UDMpa/UDMcb

| Bit Code | GP Out 0 | GP Out 1 |
|----------------|------------|------------|
| 0000 (default) | GP Out 0 | GP Out 1 |
| 0001 | PEGO_PULSE | GP Out 1 |
| 0010 | GP Out 0 | PEG1_PULSE |
| 0011 | PEG1_PULSE | GP Out 1 |
| 0100 | GP Out 0 | PEGO_PULSE |
| 0101 | PEGO_PULSE | PEG1_PULSE |
| 0110 - 1111 | Reserved | Reserved |

Bit Code assignment example

For example, for an axis associated with Servo Processor 0 (SPiiPlusNT / DC-LT / HP / LD), **0110** switches **GP Out 0** to **PEG0_PULSE** and **GP Out 2** to **PEG1_PULSE**.

The same **Bit Code** applied to an axis associated with Servo Processor 1 switches **GP Out 4** to **PEG4_PULSE** and **GP Out 6** to **PEG5_PULSE**.

All other **GP Out** assignments are unchanged.

Table A-10. Engine to Encoder Assignment for IDMxx, ECMxx, and UDMsm/sa/ma

| PEG Engine | Bits | HEX Code | Encoder 0 | Encoder 1 | Encoder 2 | Encoder 3 |
|------------|-------|--------------|-----------|-----------|-----------|-----------|
| 0 | 0..7 | 00 (default) | PEG0 | | | |
| | | 01 | | PEG0 | | |
| | | 02 | | | PEG0 | |
| | | 03 | | | | PEG0 |
| 1 | 8..15 | 00 | PEG1 | | | |
| | | 01 (default) | | PEG1 | | |
| | | 02 | | | PEG1 | |
| | | 03 | | | | PEG1 |

| PEG Engine | Bits | HEX Code | Encoder 0 | Encoder 1 | Encoder 2 | Encoder 3 |
|------------|--------|--------------|-----------|-----------|-----------|-----------|
| 2 | 16..23 | 00 | PEG2 | | | |
| | | 01 | | PEG2 | | |
| | | 02 (default) | | | PEG2 | |
| | | 03 | | | | PEG2 |
| 3 | 24..31 | 00 | PEG3 | | | |
| | | 01 | | PEG3 | | |
| | | 02 | | | PEG3 | |
| | | 03 | | | | PEG3 |

Instructions: the above table is used to build a hexadecimal value for the engines_to_encoders_code argument. Byte x determines that PEG engine x will be triggered by a specific encoder.

Example for IDMsM/ECMsM

```
ASSIGNPEG 0, 0x03020100
```

This code configures the default mapping:

- > PEG engine 0 is triggered by Encoder 0.
- > PEG1 engine 1 is triggered by Encoder 1.
- > PEG engine 2 is triggered by Encoder 2.
- > PEG engine 3 is triggered by Encoder 3.

`ASSIGNPEG 0x02010203`

This code configures the following mapping:

- > PEG engine 0 is triggered by Encoder 3.
- > PEG engine 1 is triggered by Encoder 2.
- > PEG engine 2 is triggered by Encoder 1.
- > PEG engine 3 is triggered by Encoder 2.

A.2 ASSIGNPOUTS Mapping

Mapping of PEG engine outputs to physical outputs

Table A-11. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for SPiiPlusNT/DC-LT/HP/LD

| PEG OUTPUT PIN NAME Bit Code | 0 X_PEG | 1 Y_PEG | 2 Z_PEG | 3 T_PEG | 4 H_DO_1 (HSSI) |
|------------------------------------|-------------|-------------|-------------|-------------|--------------------|
| 000 (default) | PEGO_PULSE | PEG1_PULSE | PEG4_PULSE | PEG5_PULSE | HSSI1_DO |
| 001 | PEG4_STATE0 | PEG1_STATE0 | PEG2_PULSE | PEGO_STATE0 | PEGO_STATE1 |
| 010 | Reserved | Reserved | PEG1_STATE1 | PEG2_STATE0 | PEG2_STATE1 |
| 011 | Reserved | Reserved | Reserved | Reserved | Reserved |
| 100 | Reserved | Reserved | Reserved | Reserved | Reserved |
| 111 | FGP_OUT0 | FGP_OUT1 | FGP_OUT2 | FGP_OUT3 | Reserved |

Table A-12. SPiiPlusNT/DC-LT/HP/LD Mapping of Engine Outputs to Physical Outputs (Servo Processor 1)

| PEG OUTPUT PIN NAME Bit Code | 5 X_STATE0 | 6 X_STATE1 | 7 X_STATE2 | 8 H_DO_0 (HSSI) | 9 H_CON_0 (HSSI) |
|------------------------------------|---------------|---------------|---------------|--------------------|---------------------|
| 000 (default) | PEGO_STATE0 | PEGO_STATE1 | PEGO_STATE2 | HSSIO_DO | HSSIO_CON |
| 001 | PEG1_STATE0 | PEG1_STATE1 | PEG2_STATE0 | PEGO_STATE2 | PEGO_STATE0 |
| 010 | PEG4_PULSE | PEG5_PULSE | PEG6_PULSE | PEG5_STATE0 | PEG2_STATE1 |
| 011 | Reserved | PEG4_STATE0 | PEG4_STATE1 | Reserved | PEG6_STATE1 |
| 100 | Reserved | Reserved | Reserved | Reserved | Reserved |
| 111 | FGP_OUT4 | FGP_OUT5 | FGP_OUT6 | Reserved | Reserved |

Table A-13. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for CMnt/UDMpm/UDMpc/CMhv/UDMhv

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO | 1 PEG1 | 5 STATE0 | 6 STATE1 |
|------------------------------------|----------------------|-------------------|-------------|-------------|
| 000 (default) | PEGO_PULSE | PEG1_PULSE | PEGO_OUT0 | PEGO_OUT1 |
| 001 | Encoder X Phase A | Encoder X Phase B | PEG1_OUT0 | PEG1_OUT1 |

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO | 1 PEG1 | 5 STATE0 | 6 STATE1 |
|------------------------------------|----------------------|-------------------|-------------------|-------------------|
| 010 | Encoder Y Phase A | Encoder Y Phase B | PEG2_OUT0 | PEG2_OUT1 |
| 011 | Reserved | Reserved | Encoder X Phase A | Encoder X Phase B |
| 100 | Reserved | Reserved | Encoder Y Phase A | Encoder Y Phase B |
| 111 | FGP_OUT0 | FGP_OUT1 | Encoder X INDEX | Encoder X INDEX |

Table A-14. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0, OUT 0-4) for CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa

| PEG OUTPUT PIN NAME Bit Code | 0 (0)_PEG_PULSE | 1 (1)_PEG_PULSE | 2 (1)_STATE0 | 3 (1)_STATE1 | 4 (1)_STATE2 |
|------------------------------------|----------------------|--------------------|-----------------|-----------------|-----------------|
| 000 (default) | PEGO_PULSE | PEG1_PULSE | PEG1_STATE0 | PEG1_STATE1 | PEG1_STATE2 |
| 001 | Encoder X Phase A | Encoder X Phase B | PEGO_STATE0 | PEGO_STATE1 | PEGO_STATE2 |
| 010 | Encoder Y Phase A | Encoder Y Phase B | PEG2_STATE0 | PEG2_STATE1 | PEG2_STATE2 |

| PEG OUTPUT PIN NAME Bit Code | 0 (0)_PEG_PULSE | 1 (1)_PEG_PULSE | 2 (1)_STATE0 | 3 (1)_STATE1 | 4 (1)_STATE2 |
|------------------------------------|-------------------------------------------|-------------------------------------------|----------------------|----------------------|--------------------|
| 011 | Encoder A Phase A | Encoder A Phase B | Encoder A Phase A | Encoder A Phase B | Encoder A INDEX |
| 100 | Reserved | PEG2_PULSE | Reserved | Reserved | Reserved |
| 101 | PEG0_PULSE or PEG2_PULSE | PEG2_PULSE or PEG1_PULSE | Reserved | Reserved | Reserved |
| 110 | PEG0_PULSE or PEG1_PULSE or PEG2_PULSE | PEG0_PULSE or PEG1_PULSE or PEG2_PULSE | Reserved | Reserved | Reserved |
| 111 | FGP_OUT0 | FGP_OUT1 | FGP_OUT2 | Reserved | Reserved |

Table A-15. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0, OUT_5-9) for CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa

| PEG OUTPUT PIN NAME Bit Code | 5 (0)_STATE0 | 6 (0)_STATE1 | 7 (0)_STATE2 | 8 (0)_STATE3 | 9 (1)_STATE3 |
|------------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 000 (default) | PEG0_STATE0 | PEG0_STATE1 | PEG0_STATE2 | PEG0_STATE3 | PEG1_STATE3 |
| 001 | PEG1_STATE0 | PEG1_STATE1 | PEG1_STATE2 | PEG1_STATE3 | PEG0_STATE3 |
| 010 | PEG2_STATE0 | PEG2_STATE1 | PEG2_STATE2 | PEG2_STATE3 | PEG2_STATE3 |

| PEG OUTPUT PIN NAME Bit Code | 5 (0)_STATE0 | 6 (0)_STATE1 | 7 (0)_STATE2 | 8 (0)_STATE3 | 9 (1)_STATE3 |
|------------------------------------|----------------------|---------------------|-------------------|----------------------|-----------------|
| 011 | Encoder X Phase A | Encoder X Phase B | Encoder Y Phase A | Encoder Y Phase B | PEG2_STATE0 |
| 100 | Encoder Y Phase A | EncoderY Phase B | Reserved | Encoder Y INDEX | PEG2_STATE1 |
| 101 | Encoder X INDEX | Encoder X INDEX | Reserved | Reserved | Reserved |
| 110 | PEG2_PULSE | Encoder Y INDEX | PEG2_PULSE | PEG2_PULSE | Reserved |
| 111 | Reserved | Reserved | Reserved | Reserved | Reserved |

Table A-16. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for UDMnt/UDMpa/UDMcb

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO | 1 PEG1 |
|------------------------------------|-------------------|-------------------|
| 000 (default) | PEGO_PULSE | PEG1_PULSE |
| 001 | Encoder X Phase A | Encoder X Phase B |
| 010 | Encoder Y Phase A | Encoder Y Phase B |

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO | 1 PEG1 |
|------------------------------------|-------------|-------------|
| 011 | PEG1_STATE0 | PEGO_STATE0 |
| 100 | PEGO_STATE0 | PEG1_STATE0 |
| 101 | Reserved | Reserved |
| 110 | Reserved | Reserved |
| 111 | FGP_OUT0 | FGP_OUT1 |

Table A-17. Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) for UDMlc/UDMmc/UDIlIt/UDIhp/PDIcl

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO |
|------------------------------------|------------|
| 000 (default) | PEGO_PULSE |
| 001 | Reserved |
| 010 | Reserved |
| 011 | Reserved |

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO |
|------------------------------------|-----------|
| 100 | Reserved |
| 101 | Reserved |
| 110 | Reserved |
| 111 | FGP_OUT0 |

Table A-18. NPMpm/NPMpc Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO | 1 PEG1 |
|------------------------------------|-------------|-------------|
| 000 (default) | PEGO_PULSE | PEG1_PULSE |
| 001 | PEGO_STATE0 | PEG1_STATE0 |
| 010 | PEG1_STATE0 | PEGO_STATE0 |
| 011 | PEGO_STATE1 | PEG1_STATE1 |
| 100 | PEG1_STATE1 | PEGO_STATE1 |

| PEG OUTPUT PIN NAME Bit Code | 0 PEGO | 1 PEG1 |
|------------------------------------|-----------|-----------|
| 101 | Reserved | Reserved |
| 110 | Reserved | Reserved |
| 111 | FGP_OUT0 | FGP_OUT1 |

Bit Code: 111

The **Bit Code: 111**, both for Servo Processor 0 and Servo Processor 1, is used for switching the physical output pins to Fast General Purpose Outputs: **FGP_OUT0** to **FGP_OUT6**. The state of the Fast General Purpose Outputs can be read or changed using the ACSPL+ **OUT(x)** variable. The Fast General Purpose Outputs are mapped as follows:

FGP_OUT0 is mapped to bit 16 of the ACSPL+ **OUT(x)** variable

FGP_OUT1 is mapped to bit 17 of the ACSPL+ **OUT(x)** variable

FGP_OUT2 is mapped to bit 18 of the ACSPL+ **OUT(x)** variable

FGP_OUT3 is mapped to bit 19 of the ACSPL+ **OUT(x)** variable

FGP_OUT4 is mapped to bit 20 of the ACSPL+ **OUT(x)** variable

FGP_OUT5 is mapped to bit 21 of the ACSPL+ **OUT(x)** variable

FGP_OUT6 is mapped to bit 22 of the ACSPL+ **OUT(x)** variable

Table A-19. IDMxx/ECMxx/UDMsm/UDMsa/UDMma Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

| PEG OUTPUT PIN NAME Bit Code | 0 OUT_CNFG_0 | 1 OUT_CNFG_1 | 2 OUT_CNFG_2 | 3 OUT_CNFG_3 |
|------------------------------------|-----------------|-----------------|-----------------|-----------------|
| 000 | PEG0_State0 | PEG0_State1 | PEG0_State2 | PEG0_State3 |
| 001 | PEG1_State0 | PEG1_State1 | PEG1_State2 | PEG1_State3 |
| 010 | PEG2_State0 | PEG2_State1 | PEG2_State2 | PEG2_State3 |
| 011 | PEG3_State0 | PEG3_State1 | PEG3_State2 | PEG3_State3 |
| 100 | Reserved | Reserved | Reserved | Reserved |
| 101 | Reserved | Reserved | Reserved | Reserved |
| 110 | Reserved | Reserved | Reserved | Reserved |
| 111(Default) | GP_OUT0 | GP_OUT1 | GP_OUT2 | GP_OUT3 |
| PEG OUTPUT PIN NAME Bit Code | 4 OUT_CNFG_4 | 5 OUT_CNFG_5 | 6 OUT_CNFG_6 | 7 OUT_CNFG_7 |
| 000 | Reserved | Reserved | Reserved | Reserved |
| 001 | Reserved | Reserved | Reserved | Reserved |

| PEG OUTPUT PIN NAME Bit Code | 4 OUT_CNFG_4 | 5 OUT_CNFG_5 | 6 OUT_CNFG_6 | 7 OUT_CNFG_7 |
|------------------------------------|-----------------|-----------------|-----------------|-----------------|
| 010 | Reserved | Reserved | Reserved | Reserved |
| 011 | Reserved | Reserved | Reserved | Reserved |
| 100 | Reserved | Reserved | Reserved | Reserved |
| 101 | Reserved | Reserved | Reserved | Reserved |
| 110 | Reserved | Reserved | Reserved | Reserved |
| 111(Default) | GP_OUT4 | GP_OUT5 | GP_OUT6 | GP_OUT7 |

| PEG OUTPUT PIN NAME Bit Code | 8 PEGO | 9 PEG1 | 10 PEG2 | 11 PEG3 |
|------------------------------------|-------------|-------------|-------------|-------------|
| 000 (Default) | PEGO_Pulse | PEG1_Pulse | PEG2_Pulse | PEG3_Pulse |
| 001 | PEGO_State0 | PEG0_State1 | PEGO_State2 | PEGO_State3 |
| 010 | PEG1_State0 | PEG1_State1 | PEG1_State2 | PEG1_State3 |
| 011 | PEG2_State0 | PEG2_State1 | PEG2_State2 | PEG2_State3 |

| PEG OUTPUT PIN NAME Bit Code | 8 PEGO | 9 PEG1 | 10 PEG2 | 11 PEG3 |
|------------------------------------|-------------|-------------|-------------|-------------|
| 100 | PEG3_State0 | PEG3_State1 | PEG3_State2 | PEG3_State3 |
| 101 | Reserved | Reserved | Reserved | Reserved |
| 110 | Reserved | Reserved | Reserved | Reserved |
| 111 | GP_OUT8 | GP_OUT9 | GP_OUT10 | GP_OUT11 |

A.3 ASSIGNMARK Mapping

Mark Inputs to Encoders Mapping

Table A-20. Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD

| Bit code | Latching of MARK Encoder 0(X) | Latching of MARK Encoder 1(Y) | Latching of MARK Encoder 4(Z) | Latching of MARK Encoder 5(T) | Latching of MARK Encoder 2(A) | Latching of MARK Encoder 3(B) | Latching of MARK Encoder 6(C) | Latching of MARK Encoder 7(D) |
|--------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 00000 (default) | X_MARK1 | Y_MARK1 | Z_MARK1 | T_MARK1 | - | - | - | - |
| 00001 | Y_MARK1 | Z_MARK1 | T_MARK1 | X_MARK1 | - | - | - | - |
| 00010 | Z_MARK1 | T_MARK1 | X_MARK1 | Y_MARK1 | - | - | - | - |
| 00011 | T_MARK1 | X_MARK1 | Y_MARK1 | Z_MARK1 | - | - | - | - |
| 00100 | - | Y_MARK1 | Z_MARK1 | T_MARK1 | X_MARK1 | - | - | - |
| 00101 | X_MARK1 | - | Z_MARK1 | T_MARK1 | - | Y_MARK1 | - | - |
| 00110 | X_MARK1 | Y_MARK1 | - | T_MARK1 | - | - | Z_MARK1 | - |
| 00111 | X_MARK1 | Y_MARK1 | Z_MARK1 | - | - | - | - | T_MARK1 |
| 01000 | - | - | Z_MARK1 | T_MARK1 | X_MARK1 | Y_MARK1 | - | - |
| 01001 | X_MARK1 | Y_MARK1 | - | - | - | - | Z_MARK1 | T_MARK1 |

| Bit code | Latching of MARK Encoder 0(X) | Latching of MARK Encoder 1(Y) | Latching of MARK Encoder 4(Z) | Latching of MARK Encoder 5(T) | Latching of MARK Encoder 2(A) | Latching of MARK Encoder 3(B) | Latching of MARK Encoder 6(C) | Latching of MARK Encoder 7(D) |
|----------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 01010 | X_MARK1 | - | - | - | - | Y_MARK1 | Z_MARK1 | T_MARK1 |
| 01011 | - | Y_MARK1 | - | - | X_MARK1 | - | Z_MARK1 | Z_MARK1 |
| 01100 | - | - | Z_MARK1 | - | X_MARK1 | Y_MARK1 | - | T_MARK1 |
| 01101 | - | - | - | T_MARK1 | X_MARK1 | Y_MARK1 | Z_MARK1 pin | - |
| 01110 | - | - | - | - | X_MARK1 | Y_MARK1 | Z_MARK1 | T_MARK1 |
| 01111 | - | - | - | - | Y_MARK1 | Z_MARK1 | T_MARK1 | X_MARK1 |
| 10000 | - | - | - | - | Z_MARK1 | T_MARK1 | X_MARK1 | Y_MARK1 |
| 10001 | - | - | - | - | T_MARK1 | X_MARK1 | Y_MARK1 | Z_MARK1 |

Example

```
ASSIGNMARK 1, 1, 0x0b00010
```

By using SPiiPlusNT as the first node, entering the command performs the following assignments for these inputs:

- > Latching of Encoder 0(X) occurs once Z_MARK1 physical pin gets an input.
- > Latching of Encoder 1(Y) occurs once T_MARK1 physical pin gets an input.
- > Latching of Encoder 4(Z) occurs once X_MARK1 physical pin gets an input.

- > Latching of Encoder 5(T) occurs once Y_sMARK1 physical pin gets an input.

Table A-21. Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD

| Bit code | Latching of M2ARK Encoder 0(X) | Latching of M2ARK Encoder 1(Y) | Latching of M2ARK Encoder 4(Z) | Latching of M2ARKEncoder 5(T) | Latching of M2ARKEncoder 2(A) | Latching of M2ARKEncoder 3(B) | Latching of M2ARKEncoder 6(C) | Latching of M2ARK Encoder 7(D) |
|--------------------|--------------------------------|--------------------------------|--------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|--------------------------------|
| 00000 (default) | GP IN6 | GP IN7 | GP IN4 | GP IN5 | - | - | - | - |
| 00001 | GP IN7 | GP IN4 | GP IN5 | GP IN6 | - | - | - | - |
| 00010 | GP IN4 | GP IN5 | GP IN6 | GP IN7 | - | - | - | - |
| 00011 | GP IN5 | GP IN6 | GP IN7 | GP IN4 | - | - | - | - |
| 00100 | - | GP IN7 | GP IN4 | GP IN5 | GP IN6 | - | - | - |
| 00101 | GP IN6 | - | GP IN4 | GP IN5 | - | GP IN7 | - | - |
| 00110 | GP IN6 | GP IN7 | - | GP IN5 | - | - | GP IN4 | - |
| 00111 | GP IN6 | GP IN7 | GP IN4 | - | - | - | - | GP IN5 |
| 01000 | - | - | GP IN4 | GP IN5 | GP IN6 | GP IN7 | - | - |
| 01001 | GP IN6 | GP IN7 | - | - | - | - | GP IN4 | GP IN5 |
| 01010 | GP IN6 | - | - | - | - | GP IN7 | GP IN4 | GP IN5 |

| Bit code | Latching of M2ARK Encoder 0(X) | Latching of M2ARK Encoder 1(Y) | Latching of M2ARK Encoder 4(Z) | Latching of M2ARKEncoder 5(T) | Latching of M2ARKEncoder 2(A) | Latching of M2ARKEncoder 3(B) | Latching of M2ARKEncoder 6(C) | Latching of M2ARK Encoder 7(D) |
|----------|--------------------------------|--------------------------------|--------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|--------------------------------|
| 01011 | - | GP IN7 | - | - | GP IN6 | - | GP IN4 | GP IN5 |
| 01100 | - | - | GP IN4 | - | GP IN6 | GP IN7 | - | GP IN5 |
| 01101 | - | - | - | GP IN5 | GP IN6 | GP IN7 | GP IN4 | - |
| 01110 | - | - | - | - | GP IN6 | GP IN7 | GP IN4 | GP IN5 |
| 01111 | - | - | - | - | GP IN7 | GP IN4 | GP IN5 | GP IN6 |
| 10000 | - | - | - | - | GP IN4 | GP IN5 | GP IN6 | GP IN7 |
| 10001 | - | - | - | - | GP IN5 | GP IN6 | GP IN7 | GP IN4 |

Example

```
ASSIGNMARK 1, 2, 0x0b00010
```

By using SPiiPlusNT as the first node, entering the command performs the following assignments for these inputs:

- > Latching of M2ARK of Encoder 0(X) occurs once GP IN4 physical pin gets an input.
- > Latching of M2ARK of of Encoder 1(Y) occurs once GP IN5 physical pin gets an input.
- > Latching of M2ARK of of Encoder 4(Z) occurs once GP IN6 physical pin gets an input.
- > Latching of M2ARK of Encoder 5(T) occurs once GP IN7 physical pin gets an input.

Table A-22. Mark-1 Inputs to Encoders Mapping for with SPiiPlus CMnt/UDMpm-x/UDMpc/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/CMhv/UDMhv

| Bit code | Latching of Mark1 Encoder 0(X) | Latching of Mark1 Encoder 1(Y) | Latching of Mark1 Encoder 2(A) | Latching of Mark1 Encoder 3(B) |
|---------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| 000 (default) | Mark1 of encoder 0(X) pin | Mark1 of encoder 1(Y) pin | Mark1 of encoder 0(X) pin | Mark1 of encoder 1(Y) pin |
| 001 | GP IN6 | Mark1 of encoder 0(X) pin | Mark2 of encoder 0(X) pin | Mark1 of encoder 0(X) pin |
| 010 | - | GP IN4 | GP IN6 | GP IN6 |
| 011 | - | GP IN6 | - | - |

Example

```
ASSIGNMARK 1, 1, 0x0b0001
```

By using CMhp as the first node, entering the command above performs the following assignments for these inputs:

- > Latching of Encoder 0 occurs once IN6 pin (pin 5, J9 connector at CMhp) gets an input.
- > Latching of Encoder 1 occurs once X(0)_MARK1+ physical pin (pin 12, J9 connector at CMhp) gets an input.
- > Latching of Encoder 2 occurs once X(0)_MARK2+ physical pin (pin 13, J9 connector at CMhp) gets an input.
- > Latching of Encoder 3 occurs once X(0)_MARK1+ physical pin (pin 12, J9 connector at CMhp) gets an input.

Table A-23. Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt/UDMpm/UDMpc/CMba/CMhp/CMxa/UDMba/UDMhp/UDMxa/CMhv/UDMhv

| Bit code | Latching of Mark2 Encoder 0(X) | Latching of Mark2 Encoder 1(Y) | Latching of Mark2 Encoder 2(A) | Latching of Mark2 Encoder 3(B) |
|---------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| 000 (default) | Mark2 of axis 0(X) pin | Mark2 of axis 1(Y) pin | GP IN6 | GP IN7 |
| 001 | Mark1 of axis 1(Y) pin | Mark1 of axis 1(Y) pin | Mark1 of axis 1(Y) pin | Mark1 of axis 1(Y) pin |
| 010 | Mark2 of axis 1(Y) pin | GP IN5 | Mark2 of axis 1(Y) pin | - |
| 011 | GP IN7 | GP IN7 | GP IN7 | - |

Example

```
ASSIGNMARK 0x0b010
```

By using CMhp as the first node, entering the command above performs the following assignments for these inputs:

- > Latching of Encoder 0 occurs once Y(1)_MARK2+ physical pin (pin 15, J9 connector at CMhp) gets an input.
- > Latching of Encoder 1 occurs once IN5 pin (pin 23, J9 connector at CMhp) gets an input.
- > Latching of Encoder 2 occurs once Y(1)_MARK2+ physical pin (pin 15, J9 connector at CMhp) gets an input.

Table A-24. IDMxx/ECMxx/UDMsm/UDMsA/UDMma Encoder Mapping

| Byte | 3 | 2 | 1 | 0 |
|-----------------|--------------------------|--------------------------|--------------------------|--------------------------|
| | Latching of Encoder 3(B) | Latching of Encoder 2(A) | Latching of Encoder 1(Y) | Latching of Encoder 0(Y) |
| MARK0 input pin | 0x00 | 0x00 | 0x00 | 0x00 (default) |
| MARK1 input pin | 0x01 | 0x01 | 0x01(default) | 0x01 |
| MARK2 input pin | 0x02 | 0x02(default) | 0x02 | 0x02 |
| MARK3 input pin | 0x03(default) | 0x03 | 0x03 | 0x03 |

The table above allows the user to build a hexadecimal value for the `inputs_to_encoder_bit_code` argument:

`ASSIGNMARK axis, type, 0XAABBCCDD`

Where AA is the code for encoder 3, BB is the code for encoder 2, CC is the code for encoder 1, and DD is the code for encoder 0.

Example 1 (default case)

```
ASSIGNMARK 1, 1, 0x03020100
```

By using UDMsm as the first node, entering the command above performs the following assignments for these inputs:

- > Latching of Encoder 0 occurs once MARK0 physical pin (pin 16, J11 connector) gets an input.
- > Latching of Encoder 1 occurs once MARK1 physical pin (pin 17, J11 connector) gets an input.
- > Latching of Encoder 2 occurs once MARK2 physical pin (pin 18, J11 connector) gets an input.
- > Latching of Encoder 3 occurs once MARK3 physical pin (pin 19, J11 connector) gets an input.

Example 2

```
ASSIGNMARK 1, 1, 0x03010102
```

By using UDMsm as the first node, entering the command above performs the following assignments for these inputs:

- > Latching of Encoder 0 occurs once MARK2 physical pin (pin 18, J11 connector) gets an input.
- > Latching of Encoder 1 occurs once MARK1 physical pin (pin 17, J11 connector) gets an input.
- > Latching of Encoder 2 occurs once MARK1 physical pin (pin 17, J11 connector) gets an input.
- > Latching of Encoder 3 occurs once MARK3 physical pin (pin 19, J11 connector) gets an input.

Smarter



Motion

www.acsmotioncontrol.com

5 HaTnufa St.
Yokneam Illit, 2066717
Israel
Tel: (+972) (4) 654 6440 Fax: (+972) (4) 654 6443

Contact us: sales@acsmotioncontrol.com | www.acsmotioncontrol.com

