

# SPiiPlus Comamnd & Variable

## Reference Guide

July 2017

Document Revision: 2.40

SPiiPlus Command and Variable Reference Guide

Release Date: July 2017

**COPYRIGHT**

© ACS Motion Control Ltd.2017. 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**

ACS Motion Control, SPiiPlus, PEG, MARK, ServoBoost, NetworkBoost and NanoPWN are trademarks of ACS Motion Control Ltd.

Windows and Visual Basic 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.

[www.acsmotioncontrol.com](http://www.acsmotioncontrol.com)

[support@acsmotioncontrol.com](mailto:support@acsmotioncontrol.com)

[sales@acsmotioncontrol.com](mailto: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
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
<b>Bold</b>	Names of GUI objects or commands.
<b>BOLD+ UPPERCASE</b>	ACSPL+ variables and commandss
Monospace + grey background	Code example.
<i>Italic</i>	Names of other documents.
<a href="#">Blue</a>	Web pages, and e-mail addresses.
[ ]	In GUIs indicates optional item(s)
	In GUIs indicates either/or items

## Flagged Text

	<b>Note</b> - includes additional information or programming tips.
	<b>Caution</b> - describes a condition that may result in damage to equipment.
	<b>Warning</b> - describes a condition that may result in serious bodily injury or death.
	<b>Model</b> - highlights a specification, procedure, condition, or statement that depends on the product model.
	<b>Advanced</b> - 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 [www.acsmotioncontrol.com/downloads](http://www.acsmotioncontrol.com/downloads).

Online versions for all ACS software manuals are available to authorized users at [ACS Motion Control Knowledge Center](http://www.acsmotioncontrol.com/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>	.NET Methods, Properties, and Events for Communication with the Controller.
<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

---

<b>1. Introduction .....</b>	<b>28</b>
<b>2. ACSPL+ Commands .....</b>	<b>29</b>
<b>2.1 Axis Management Commands .....</b>	<b>33</b>
<b>2.1.1 BREAK .....</b>	<b>34</b>
<b>2.1.2 COMMUT .....</b>	<b>34</b>
<b>2.1.3 CONNECT .....</b>	<b>35</b>
<b>2.1.4 DEPENDS .....</b>	<b>38</b>
<b>2.1.5 DISABLE .....</b>	<b>38</b>
<b>2.1.6 ENABLE/ENABLEALL .....</b>	<b>40</b>
<b>2.1.7 FCLEAR .....</b>	<b>41</b>
<b>2.1.8 GO .....</b>	<b>42</b>
<b>2.1.9 GROUP .....</b>	<b>43</b>
<b>2.1.10 HALT .....</b>	<b>43</b>
<b>2.1.11 IMM .....</b>	<b>44</b>
<b>2.1.12 KILL .....</b>	<b>45</b>
<b>2.1.13 SAFETYCONF .....</b>	<b>46</b>
<b>2.1.14 SAFETYGROUP .....</b>	<b>48</b>
<b>2.1.15 SET .....</b>	<b>48</b>
<b>2.1.16 SPLIT .....</b>	<b>49</b>
<b>2.2 Interactive Commands .....</b>	<b>50</b>
<b>2.2.1 DISP .....</b>	<b>50</b>
<b>2.2.2 INP .....</b>	<b>55</b>
<b>2.2.3 INTERRUPT .....</b>	<b>56</b>
<b>2.2.4 INTERRUPTEX .....</b>	<b>58</b>
<b>2.2.5 SEND .....</b>	<b>59</b>
<b>2.2.6 TRIGGER .....</b>	<b>61</b>
<b>2.2.7 OUTP .....</b>	<b>62</b>
<b>2.3 PEG and MARK Commands .....</b>	<b>63</b>
<b>2.3.1 ASSIGNMARK .....</b>	<b>64</b>
<b>2.3.2 ASSIGNPEG .....</b>	<b>68</b>
<b>2.3.3 ASSIGNPOUTS .....</b>	<b>76</b>
<b>2.3.4 PEG_I .....</b>	<b>85</b>
<b>2.3.5 PEG_R .....</b>	<b>87</b>

---

2.3.6 STARTPEG .....	91
2.3.7 STOPPEG .....	91
2.4 Miscellaneous Commands .....	92
2.4.1 AXISDEF .....	93
2.4.2 DC .....	94
2.4.3 STOPDC .....	96
2.4.4 READ .....	97
2.4.5 SPDC .....	98
2.4.6 WRITE .....	100
2.4.7 SPINJECT .....	100
2.4.8 STOPINJECT .....	101
2.4.9 SPRT .....	101
2.4.10 SPRTSTOP .....	105
2.5 Motion Commands .....	105
2.5.1 ARC1 .....	106
2.5.2 ARC1 .....	107
2.5.3 ARC2 .....	109
2.5.4 ARC2 .....	111
2.5.5 JOG .....	113
2.5.6 LINE .....	114
2.5.7 LINE .....	115
2.5.8 MASTER .....	116
2.5.9 MPOINT .....	118
2.5.10 MPTP...ENDS .....	122
2.5.11 MSEG...ENDS .....	125
2.5.12 PATH...ENDS .....	127
2.5.13 POINT .....	129
2.5.14 PROJECTION .....	131
2.5.15 PTP .....	134
2.5.16 PVSLINE...ENDS .....	135
2.5.17 SLAVE .....	137
2.5.18 STOPPER .....	138
2.5.19 TRACK .....	140
2.5.20 XSEG...ENDS .....	141
2.6 Program Flow Commands .....	148

---

2.6.1 Assignment Command .....	149
2.6.2 BLOCK...END .....	150
2.6.3 CALL .....	151
2.6.4 GOTO .....	152
2.6.5 IF, ELSEIF, ELSE...END .....	152
2.6.6 INPUT .....	154
2.6.7 LOOP...END .....	155
2.6.8 ON...RET .....	156
2.6.9 TILL .....	157
2.6.10 WAIT .....	158
2.6.11 WHILE...END .....	158
2.7 Program Management Commands .....	159
2.7.1 DISABLEON .....	159
2.7.2 ENABLEON .....	160
2.7.3 PAUSE .....	160
2.7.4 RESUME .....	160
2.7.5 START .....	161
2.7.6 STOP .....	162
2.8 Ethernet/IP ACSPL+ Support Commands .....	162
2.8.1 EIPGETATTR .....	162
2.8.2 EIPGETIND1 .....	164
2.8.3 EIPGETIND2 .....	164
2.8.4 EIPGETTAG .....	165
2.8.5 EIPSETASM .....	165
<b>3. ACSPL+ Variables .....</b>	<b>167</b>
3.1 Axis Configuration Variables .....	181
3.1.1 AFLAGS .....	182
3.1.2 ENTIME .....	183
3.1.3 MFF .....	184
3.1.4 MFLAGS .....	185
3.1.5 SETTLE .....	190
3.1.6 SLPMAX .....	191
3.1.7 SLPMIN .....	192
3.1.8 STEPF .....	192
3.1.9 STEPW .....	193

---

3.1.10 TARGRAD .....	194
3.1.11 Brake Variables .....	195
3.1.11.1 BOFFTIME .....	195
3.1.11.2 BONTIME .....	196
3.1.11.3 VELBRK .....	196
3.1.12 Feedback Variables .....	197
3.1.12.1 E_AOFFS .....	199
3.1.12.2 E_FREQ .....	200
3.1.12.3 E2_FREQ .....	201
3.1.12.4 E_FLAGS .....	201
3.1.12.5 E2_FLAGS .....	203
3.1.12.6 E_PAR_A .....	204
3.1.12.7 E2_PAR_A .....	204
3.1.12.8 E_PAR_B .....	205
3.1.12.9 E2_PAR_B .....	205
3.1.12.10 E_PAR_C .....	206
3.1.12.11 E2_PAR_C .....	206
3.1.12.12 E_PAR_D .....	207
3.1.12.13 E2_PAR_D .....	208
3.1.12.14 E_PAR_E .....	208
3.1.12.15 E2_PAR_E .....	209
3.1.12.16 E_SCMUL .....	209
3.1.12.17 E2_SCMUL .....	210
3.1.12.18 E_TYPE .....	211
3.1.12.19 E2_TYPE .....	212
3.1.12.20 EFAC .....	213
3.1.12.21 E2FAC .....	214
3.1.12.22 EOFFS .....	215
3.1.12.23 E2OFFS .....	216
3.1.12.24 EPOS .....	216
3.1.12.25 FVFIL .....	217
3.1.12.26 F2ACC .....	218
3.1.12.27 RVFIL .....	218
3.1.12.28 SCSOFFS .....	219
3.1.12.29 SCCOFFS .....	220

---

3.1.12.30 SC2COFFS .....	221
3.1.12.31 SC2GAIN .....	221
3.1.12.32 SC2PHASE .....	222
3.1.12.33 SC2SOFFS .....	222
3.1.12.34 SLEBIASA .....	223
3.1.12.35 SLEBIASB .....	224
3.1.12.36 SLEBIASC .....	225
3.1.12.37 SLEBIASD .....	226
3.1.12.38 SLABITS .....	226
3.1.12.39 S2LABITS .....	227
3.1.12.40 SCGAIN .....	227
3.1.12.41 SCPHASE .....	228
3.1.13 Safety Limits Variables .....	229
3.1.13.1 CERRA .....	229
3.1.13.2 CERRI .....	230
3.1.13.3 CERRV .....	231
3.1.13.4 DELV .....	232
3.1.13.5 DELI .....	233
3.1.13.6 ERRA .....	234
3.1.13.7 ERRI .....	234
3.1.13.8 ERRV .....	235
3.1.13.9 SLLIMIT .....	236
3.1.13.10 SLLROUT .....	237
3.1.13.11 SRLIMIT .....	237
3.1.13.12 XACC .....	238
3.1.13.13 XCURI .....	239
3.1.13.14 XCURV .....	240
3.1.13.15 XRMS .....	240
3.1.13.16 XRMST .....	241
3.1.13.17 XSACC .....	242
3.1.13.18 XVEL .....	243
3.2 Axis State Variables .....	243
3.2.1 AST .....	244
3.2.2 IND .....	246
3.2.3 IST .....	246

---

3.2.4 M2ARK .....	247
3.2.5 MARK .....	248
3.2.6 MST .....	248
3.2.7 RMS .....	249
3.2.8 NST .....	250
3.3 Data Collection Variables .....	251
3.3.1 DCN .....	251
3.3.2 DCP .....	252
3.3.3 S_DCN .....	253
3.3.4 S_DCP .....	253
3.3.5 S_ST .....	254
3.4 Input and Output Variables .....	255
3.4.1 AIN .....	255
3.4.2 AOUT .....	256
3.4.3 DOUT .....	256
3.4.4 EXTIN .....	257
3.4.5 EXTOUT .....	257
3.4.6 IN .....	258
3.4.7 OUT .....	259
3.5 Monitoring Variables .....	260
3.5.1 JITTER .....	260
3.5.2 MSSYNC .....	260
3.5.3 USGBUF .....	261
3.5.4 USGTRACE .....	261
3.5.5 SOFTIME .....	262
3.5.6 TIME .....	262
3.5.7 USAGE .....	263
3.6 Motion Variables .....	263
3.6.1 ACC .....	265
3.6.2 APOS .....	265
3.6.3 DAPOS .....	266
3.6.4 DEC .....	267
3.6.5 FACC .....	267
3.6.6 FPOS .....	268
3.6.7 F2POS .....	268

---

3.6.8 FVEL .....	269
3.6.9 F2VEL .....	269
3.6.10 GACC .....	270
3.6.11 GJERK .....	270
3.6.12 GMOT .....	271
3.6.13 GMQU .....	271
3.6.14 GMTYPE .....	272
3.6.15 GPATH .....	273
3.6.16 GPHASE .....	273
3.6.17 GRTIME .....	274
3.6.18 GSEG .....	275
3.6.19 GSFREE .....	276
3.6.20 GVEC .....	276
3.6.21 GVEL .....	277
3.6.22 JERK .....	277
3.6.23 KDEC .....	278
3.6.24 MPOS .....	279
3.6.25 NVEL .....	279
3.6.26 PE .....	280
3.6.27 RACC .....	281
3.6.28 RJERK .....	281
3.6.29 ROFFS .....	281
3.6.30 RPOS .....	282
3.6.31 RVEL .....	283
3.6.32 TPOS .....	283
3.6.33 VEL .....	284
3.7 Program Execution Control Variables .....	285
3.7.1 ONRATE .....	285
3.7.2 PCHARS .....	286
3.7.3 PERL .....	286
3.7.4 PERR .....	287
3.7.5 PEXL .....	287
3.7.6 PFLAGS .....	288
3.7.7 PLINES .....	289
3.7.8 PRATE .....	290

---

3.7.9 PST .....	290
3.8 Safety Control Variables .....	291
3.8.1 AERR .....	292
3.8.2 ECERR .....	293
3.8.3 ECEXTERR .....	293
3.8.4 ECEXTST .....	296
3.8.5 ECST .....	297
3.8.6 FAULT .....	298
3.8.7 FDEF .....	301
3.8.8 FMASK .....	306
3.8.9 MERR .....	308
3.8.10 SAFIN .....	311
3.8.11 SAFINI .....	312
3.8.12 S_ERR .....	313
3.8.13 S_FAULT .....	313
3.8.14 S_FDEF .....	317
3.8.15 S_FMASK .....	319
3.8.16 S_SAFIN .....	320
3.8.17 S_SAFINI .....	321
3.8.18 STODELAY .....	321
3.8.19 SYNC .....	322
3.9 Induction Motor Variables .....	322
3.9.1 SLCFIELD .....	323
3.9.2 SLCSLIP .....	324
3.10 Nanomotion Variables .....	325
3.10.1 SLDZMIN .....	326
3.10.2 SLDZMAX .....	327
3.10.3 SLDZTIME .....	328
3.10.4 SLZFF .....	328
3.10.5 SLFRC .....	329
3.10.6 SLFRCN .....	330
3.10.7 SLHRS .....	330
3.10.8 SLVKPDCF .....	331
3.10.9 SLPKPDCF .....	331
3.10.10 SLVKIDCF .....	332

---

3.11 Servo-Loop Variables .....	333
3.11.1 DCOM .....	333
3.11.2 Servo-Loop Current Variables .....	334
3.11.2.1 SLBIASA .....	334
3.11.2.2 SLBIASB .....	335
3.11.2.3 SLIKI .....	336
3.11.2.4 SLIKP .....	337
3.11.2.5 SLIFILT .....	337
3.11.2.6 SLIOFFS .....	338
3.11.2.7 SLII .....	339
3.11.3 Servo-Loop Velocity Variables .....	339
3.11.3.1 SLVKI .....	340
3.11.3.2 SLVKIIF .....	340
3.11.3.3 SLVKISF .....	341
3.11.3.4 SLVKITF .....	342
3.11.3.5 SLVKP .....	342
3.11.3.6 SLVKPIF .....	343
3.11.3.7 SLVKPSF .....	343
3.11.3.8 SLVKPTF .....	344
3.11.3.9 SLVLI .....	345
3.11.3.10 SLVRAT .....	345
3.11.4 Servo-Loop Velocity Notch Filter Variables .....	346
3.11.4.1 SLVNFRQ .....	346
3.11.4.2 SLVNWID .....	347
3.11.4.3 SLVNATT .....	347
3.11.5 Servo-Loop Velocity Low Pass Filter Variables .....	348
3.11.5.1 SLVSOF .....	348
3.11.5.2 SLVSOFD .....	349
3.11.6 Servo-Loop Velocity Bi-Quad Filter Variables .....	350
3.11.6.1 SLVBODD .....	350
3.11.6.2 SLVBODF .....	351
3.11.6.3 SLVBOND .....	351
3.11.6.4 SLVBONF .....	352
3.11.7 Servo-Loop Position Variables .....	353
3.11.7.1 SLDRA .....	353

---

3.11.7.2 SLDRAIF .....	354
3.11.7.3 SLDRX .....	355
3.11.7.4 SLPKI .....	356
3.11.7.5 SLPKIIF .....	356
3.11.7.6 SLPKISF .....	357
3.11.7.7 SLPKITF .....	357
3.11.7.8 SLPLI .....	358
3.11.7.9 SLPKP .....	359
3.11.7.10 SLPKPIF .....	359
3.11.7.11 SLPKPSF .....	360
3.11.7.12 SLPKPTF .....	361
3.11.8 Servo-Loop Compensations Variables .....	361
3.11.8.1 SLAFF .....	361
3.11.8.2 SLFRCD .....	362
3.11.9 Servo-Loop Miscellaneous Variables .....	363
3.11.9.1 SLCROUT .....	363
3.11.9.2 SLGCAXN .....	365
3.11.9.3 SLPROUT .....	365
3.11.9.4 SLP2ROUT .....	367
3.11.9.5 SLTFWID .....	368
3.11.9.6 SLVROUT .....	369
3.12 Commutation Variables .....	371
3.12.1 SLCHALL .....	371
3.12.2 SLCNP .....	372
3.12.3 SLCPA .....	372
3.12.4 SLCOFFS .....	373
3.12.5 SLCORG .....	374
3.12.6 SLCPRD .....	375
3.12.7 SLHROUT .....	375
3.12.8 SLSTHALL .....	376
3.13 System Configuration Variables .....	376
3.13.1 CFG .....	377
3.13.2 CTIME .....	378
3.13.3 ECHO .....	379
3.13.4 G_01WCS...G_12WCS .....	380

---

3.13.5 GPEXL .....	381
3.13.6 GUFAC .....	381
3.13.7 IENA .....	382
3.13.8 IMASK .....	383
3.13.9 ISENA .....	384
3.13.10 S_FLAGS .....	385
3.13.11 S_SETUP .....	386
3.13.12 XSEGAMAX .....	387
3.13.13 XSEGAMIN .....	388
3.13.14 XSEGRMAX .....	388
3.13.15 XSEGRMIN .....	389
3.14 Communication Variables .....	389
3.14.1 BAUD .....	390
3.14.2 COMMCH .....	390
3.14.3 COMMFL .....	392
3.14.4 CONID .....	393
3.14.5 ECHO .....	393
3.14.6 DISPCH .....	395
3.14.7 GATEWAY .....	396
3.14.8 SUBNET .....	397
3.14.9 TCPIP .....	398
3.14.10 TCPIP2 .....	399
3.14.11 TCPPORT .....	400
3.14.12 UDPPORT .....	401
3.15 Miscellaneous .....	401
3.15.1 FK .....	402
3.15.2 XARRSIZE .....	402
<b>4. ACSPL+ Functions .....</b>	<b>404</b>
<b>4.1 Arithmetical Functions .....</b>	<b>407</b>
4.1.1 ABS .....	408
4.1.2 ACOS .....	408
4.1.3 ASIN .....	409
4.1.4 ATAN .....	409
4.1.5 ATAN2 .....	410
4.1.6 CEIL .....	410

---

4.1.7 COS .....	411
4.1.8 EXP .....	411
4.1.9 FLOOR .....	412
4.1.10 HYPOT .....	412
4.1.11 LDEXP .....	413
4.1.12 LOG .....	413
4.1.13 LOG10 .....	414
4.1.14 POW .....	414
4.1.15 SIGN .....	415
4.1.16 SIN .....	415
4.1.17 SQRT .....	415
4.1.18 TAN .....	416
4.2 Miscellaneous Functions .....	416
4.2.1 GETCONF .....	417
4.2.2 SYSINFO .....	423
4.2.3 GETVAR .....	424
4.2.4 SETCONF .....	425
4.2.5 SETVAR .....	432
4.2.6 STR .....	433
4.2.7 STRTONUM .....	434
4.2.8 NUMTOSTR .....	434
4.2.9 BCOPY .....	436
4.3 Array Processing Functions .....	437
4.3.1 AVG .....	438
4.3.2 COPY .....	438
4.3.3 DSHIFT .....	440
4.3.4 FILL .....	441
4.3.5 MAX .....	442
4.3.6 MAXI .....	442
4.3.7 MIN .....	443
4.3.8 MINI .....	444
4.4 EtherCAT Functions .....	444
4.4.1 COEGETSIZE .....	445
4.4.2 ECCLOSEPORT .....	446
4.4.3 ECCLRREG .....	447

---

4.4.4 ECEXTIN .....	447
4.4.5 ECEXTOUT .....	448
4.4.6 ECGETGRPIND .....	450
4.4.7 ECGETPID .....	450
4.4.8 ECGETMAIN .....	450
4.4.9 ECGETOFFSET .....	451
4.4.10 ECGETOPTGRP .....	451
4.4.11 ECGETRED .....	452
4.4.12 ECGETREG .....	452
4.4.13 ECGETSLAVES .....	454
4.4.14 ECGETSTATE .....	454
4.4.15 ECGETVID .....	454
4.4.16 ECGRPINFO .....	454
4.4.17 ECIN .....	455
4.4.18 ECOUT .....	456
4.4.19 ECREPAIR .....	458
4.4.20 ECRESCAN .....	459
4.4.21 ECSAVECFG .....	459
4.4.22 ECSAVEDCNF .....	460
4.4.23 ECUNMAP .....	460
4.4.24 ECUNMAPIN .....	460
4.4.25 ECUNMAPOUT .....	461
4.5 CoE Functions .....	461
4.5.1 COEREAD/d .....	462
4.5.2 COEWRITE/d .....	463
4.6 Servo Processor Functions .....	464
4.6.1 GETSP .....	464
4.6.2 GETSPA .....	465
4.6.3 GETSPV .....	466
4.6.4 SETSP .....	466
4.6.5 SETSPV .....	466
4.7 Signal Processing Functions .....	466
4.7.1 DEADZONE .....	468
4.7.2 DSIGN .....	470
4.7.3 DSTR .....	471

---

4.7.4 EDGE .....	472
4.7.5 INTGR .....	473
4.7.6 LAG .....	474
4.7.7 Interpolation Functions .....	476
4.7.7.1 Linear interpolation .....	476
4.7.7.2 Spline interpolation .....	476
4.7.7.3 MAP .....	482
4.7.7.4 MAPB .....	484
4.7.7.5 MAPN .....	486
4.7.7.6 MAPNB .....	487
4.7.7.7 MAPNS .....	489
4.7.7.8 MAPS .....	491
4.7.7.9 MAP2 .....	493
4.7.7.10 MAP2B .....	495
4.7.7.11 MAP2N .....	497
4.7.7.12 MAP2NB .....	499
4.7.7.13 MAP2NS .....	502
4.7.7.14 MAP2S .....	504
4.7.8 MATCH .....	506
4.7.9 RAND .....	508
4.7.10 ROLL .....	508
4.7.11 SAT .....	510
5. Terminal Commands .....	512
5.1 Entering Terminal Commands .....	512
5.2 Query Commands .....	512
5.2.1 Default Query Formats .....	514
5.2.2 Predefined Query Output Formats .....	514
5.2.3 User-Defined Query Output Format .....	515
5.3 Program Management Commands .....	516
5.3.1 Program Management Command Arguments .....	516
5.3.2 Program Buffer Commands .....	517
5.3.2.1 Open/Close Buffer (#) .....	517
5.3.2.2 D .....	519
5.3.2.3 F/IF .....	520
5.3.2.4 L .....	521

---

5.3.3 RESET .....	522
5.3.4 Listing Program Variables .....	523
5.3.4.1 VGR .....	523
5.3.4.2 VSD .....	524
5.3.4.3 VS/VSG .....	524
5.3.4.4 VSF/VSGF .....	525
5.3.4.5 VG/VGF .....	525
5.3.4.6 VL/VLF .....	526
5.3.4.7 V/VF .....	526
5.3.4.8 VSP .....	527
5.3.4.9 VST/VSGT .....	528
5.3.4.10 VSTF/VSGTF/VSDT .....	528
5.3.4.11 VGV .....	528
5.3.5 Program Handling Commands .....	529
5.3.5.1 C .....	530
5.3.5.2 X .....	531
5.3.5.3 S/SR .....	531
5.3.5.4 P .....	532
5.3.6 Debug Commands .....	533
5.3.6.1 XS .....	533
5.3.6.2 XD .....	533
5.3.6.3 BS .....	534
5.3.6.4 BR .....	534
5.4 System Commands .....	535
5.4.1 SI .....	535
5.4.2 SIR .....	539
5.4.3 MEMORY .....	551
5.4.4 IR .....	551
5.4.5 U .....	553
5.4.6 TD .....	553
5.4.7 SC .....	554
5.4.8 ETHERCAT .....	557
5.4.9 ECMAPREP .....	566
5.4.10 CC .....	567
5.4.11 PLC .....	568

---

5.4.12 LOG .....	568
5.4.13 LOG HOST_TICKS .....	570
<b>6. SPiiPlus Error Codes .....</b>	<b>572</b>
6.1 ACSPL+ Syntax Errors .....	572
6.2 ACSPL+ Compilation Errors .....	591
6.3 ACSPL+ Runtime Errors .....	600
6.4 Motion Termination Errors .....	612
6.5 System Errors .....	619
6.6 EtherCAT Errors .....	621

# List Of Figures

---

Figure 2-1. CONNECT Using MAP Function .....	37
Figure 2-2. DISABLE and Mechanical Brake Output Process- Positive BONTIME .....	39
Figure 2-3. DISABLE and Mechanical Brake Output Process - Negative BONTIME .....	40
Figure 2-4. ENABLE and Mechanical Brake Output Process - Positive BOFFTIME .....	41
Figure 2-5. ARC1 Coordinate Specification .....	106
Figure 2-6. ARC2 Center Point and Rotation Angle Specification .....	110
Figure 2-7. SLAVE /pt Illustration .....	118
Figure 2-8. Single-Axis Motion Using MPTP .....	124
Figure 2-9. Two-Axis Group Motion Using MPTP/v .....	125
Figure 2-10. Results of Example MSEG .....	127
Figure 2-11. PATH...ENDS Diagram .....	128
Figure 2-12. PROJECTION of the XA Plane .....	132
Figure 2-13. FPOS - PROJECTION Example .....	133
Figure 2-14. PROJECTION Example - Final Result .....	133
Figure 2-15. PVSPLINE Motion Diagram .....	137
Figure 2-16. Use of STOPPER .....	140
Figure 2-17. Corner Processing - Exact Path Option .....	145
Figure 2-18. Corner Processing - Permitted Deviation, Permitted Radius and Corner Smoothing Options .....	146
Figure 4-1. Illustration of COPY Function .....	440
Figure 4-2. Symmetrical Dead Zone Example .....	469
Figure 4-3. Asymmetrical Dead Zone Example .....	470
Figure 4-4. DSIGN Function Example .....	471
Figure 4-5. EDGE Function Example .....	473
Figure 4-6. INTGR Function Example .....	474
Figure 4-7. LAG Function Example .....	476
Figure 4-8. Spline Definition Range .....	477
Figure 4-9. Two-Dimensional Spline Definition Range .....	478
Figure 4-10. 5-Point Catmull-Rom Spline .....	479
Figure 4-11. B-Spline - Approximation of Points .....	480
Figure 4-12. Catmull-Ron Spline Beyond the Definition Range .....	481
Figure 4-13. B-Spline Map .....	482
Figure 4-14. MAP Example on the Scope .....	484
Figure 4-15. MAPB Example on the Scope .....	486

---

Figure 4-16. MAPN Example on the Scope .....	487
Figure 4-17. MAPNB Example on the Scope .....	489
Figure 4-18. MAPNS Example on the Scope .....	491
Figure 4-19. MAPS Example on the Scope .....	492
Figure 4-20. MAP2 Example on the Scope .....	495
Figure 4-21. MAP2B Example on the Scope .....	497
Figure 4-22. MAP2N Example on the Scope .....	499
Figure 4-23. MAP2NB Example on the Scope .....	501
Figure 4-24. MAP2NS Example on the Scope .....	504
Figure 4-25. MAP2S Example on the Scope .....	506
Figure 4-26. ROLL Example on the Scope .....	509
Figure 4-27. SAT Example on the Scope .....	511
Figure 5-1. Communication Terminal Window .....	512
Figure 5-2. Interaction of Program Buffer States .....	529

## List of Tables

---

Table 2-1. The ACSPL+ command set .....	29
Table 2-2. DISP Command Option Escape Sequences .....	51
Table 2-3. Type Characters .....	51
Table 2-4. Channel Designation for TRIGGER .....	61
Table 2-5. Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x ..	64
Table 2-6. Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x ..	66
Table 2-7. Mark-1 and Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x ..	67
Table 2-8. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 0) .....	70
Table 2-9. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 1) .....	70
Table 2-10. SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x/UDMnt-x Mapping PEG Engines to Encoders (Servo Processor 0) .....	71
Table 2-11. UDMlc-x/UDlIt-x/UDlhp-x/UDMmc-x/PDIcl-x/ Mapping PEG Engines to Encoders (Servo Processor 0) .....	72
Table 2-12. NPMpm-x/NPMpc-x Mapping PEG Engines to Encoders (Servo Processor 0) .....	72
Table 2-13. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) .....	73
Table 2-14. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1) .....	74
Table 2-15. SPiiPlus CMnt-x/UDMpm-x/CMhv-x/UDMhv-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) .....	75
Table 2-16. UDMnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0) .....	75
Table 2-17. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	78
Table 2-18. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 1) .....	78
Table 2-19. CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	79
Table 2-20. CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	80
Table 2-21. CMba-x/CMhp-x/UDMba-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	80
Table 2-22. CMba-x/CMhp-x/UDMba-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	81
Table 2-23. UDMnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	82

---

Table 2-24. UDMIc-x/UDMmc-x/UDIlt-x/UDIhp-x/PDIcl-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	82
Table 2-25. NPMpm-x/NPMpc-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0) .....	83
Table 2-26. PEG Output Signal Configuration .....	88
Table 2-27. Commonly Monitored SPDC Variables .....	99
Table 2-28. Matrix Values .....	132
Table 2-29. IF Control Structures .....	152
Table 3-1. Alphabetical Listing of All ACSPL+ Variables .....	168
Table 3-2. AFLAG Bit Description .....	182
Table 3-3. MFLAGS Bit Designators .....	185
Table 3-4. E_FLAGS Bit Description .....	202
Table 3-5. AST Bit Descriptions .....	244
Table 3-6. MST Bit Descriptions. ....	249
Table 3-7. NST Bit Description .....	250
Table 3-8. PFLAGS Bit Description 1 .....	288
Table 3-9. PST Bit Description .....	291
Table 3-10. ECST Bits .....	297
Table 3-11. Axis Fault Bits .....	298
Table 3-12. FDEF Bit Description .....	301
Table 3-13. FMASK Bit Description .....	306
Table 3-14. MERR Return Values .....	309
Table 3-15. SAFINI Valid Bits .....	312
Table 3-16. S_FAULT Fault Bits .....	314
Table 3-17. S_FDEF Bit Description .....	317
Table 3-18. S_FMASK Bit Description .....	319
Table 3-19. SLCROUT Values .....	363
Table 3-20. SLPROUT Values .....	366
Table 3-21. SLVROUT Values .....	369
Table 3-22. ECHO Channel Numbers .....	379
Table 3-23. IENA Bit Description .....	382
Table 3-24. ISENA Bit Description .....	384
Table 3-25. S_FLAGS Bit Description .....	386
Table 3-26. S_SETUP Bit Designators .....	387
Table 3-27. COMMCH Values .....	391
Table 3-28. COMMFL Bit Descriptions .....	392

---

Table 3-29. ECHO Channel Numbers .....	394
Table 3-30. DISPCH Channel Numbers .....	395
Table 4-1. GETCONF Return Values .....	417
Table 4-2. SYSINFO Return Values .....	423
Table 4-3. 16-bit Binary value Template .....	425
Table 4-4. SETCONF Arguments .....	425
Table 4-5. Supported Error Counter Registers .....	452
Table 4-6. MAP Array .....	483
Table 4-7. MAPB Array .....	485
Table 4-8. MAPN Array .....	487
Table 4-9. MAPNB Array .....	488
Table 4-10. MAPNS Array .....	490
Table 4-11. MAPS Array .....	492
Table 4-12. MAP2 .....	494
Table 4-13. MAP2B .....	496
Table 4-14. MAP2B .....	499
Table 4-15. MAP2NB .....	501
Table 4-16. MAP2NS .....	503
Table 4-17. MAP2S .....	506
Table 5-1. Line Designation .....	517
Table 6-1. ACSPL+ Syntax Errors .....	572
Table 6-2. ACSPL+ Compilation Errors .....	591
Table 6-3. ACSPL+ Runtime Errors .....	600
Table 6-4. ACSPL+ Motion Termination Errors .....	612
Table 6-5. ACSPL+ System Errors .....	619
Table 6-6. ACSPL+ EtherCAT Errors .....	621

## ***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.

## 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

This chapter covers the ACSPL+ commands.

Table 2-1. The ACSPL+ command set

Command	Description
ARC1	Adds an arc segment to <a href="#">MSEG...ENDS</a> motion.
ARC1	Adds an arc segment to <a href="#">XSEG...ENDS</a> motion
ARC2	Adds an arc segment to <a href="#">MSEG...ENDS</a> motion.
ARC2	Adds an arc segment to <a href="#">XSEG...ENDS</a> 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 ( <a href="#">RPOS</a> ).
DC	Activates data collection.

Command	Description
DEPENDS	Specifies a logical dependence between a motor and axes.
DISABLE	Shuts off one or more drives. <b>DISABLE ALL</b> provides <b>DISABLE</b> 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/ENABLEALL	Activates one or more drives
ENABLEON	Enables autoroutine activation in a buffer.
FCLEAR	Clears faults.
GO	Starts a motion that was created using the <b>/w</b> 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 ( <b>DEC</b> deceleration).  <b>HALTALL</b> provides <b>HALT</b> operation for all axes.
IF, ELSEIF, ELSE...END	<b>IF</b> 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 <b>INTERRUPT</b> command.
JOG	Creates a jog motion.
KILL	Terminates one or more motions using a second-order deceleration profile and the <b>KDEC</b> deceleration value.  <b>KILL ALL</b> provides <b>KILL</b> operation for all axes.
LINE	Adds a linear segment to <b>MSEG...ENDS</b> motion.
LINE	Adds a linear segment to <b>XSEG...ENDS</b> motion.

Command	Description
<a href="#">LOOP...END</a>	Loop command structure.
<a href="#">MASTER</a>	Defines a formula for calculating <a href="#">MPOS</a> .
<a href="#">MPOINT</a>	Adds a set of points to <a href="#">MPTP...ENDS</a> , <a href="#">PATH...ENDS</a> or <a href="#">PVSPLINE...ENDS</a> motion.
<a href="#">MPTP...ENDS</a>	Creates a multipoint motion.
<a href="#">MSEG...ENDS</a>	Creates a segmented motion.
<a href="#">ON...RET</a>	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.
<a href="#">PATH...ENDS</a>	Creates an arbitrary path motion with linear interpolation between the specified points.
<a href="#">PAUSE</a>	Suspends program execution in a buffer.
<a href="#">PEG_I</a>	Defines Incremental PEG parameters. <a href="#">PEG_I</a> activates the PEG engine.
<a href="#">PEG_R</a>	Defines the Random PEG parameters. <a href="#">PEG_R</a> activates the PEG engine.
<a href="#">POINT</a>	Adds a point to <a href="#">MPTP...ENDS</a> , <a href="#">PATH...ENDS</a> , or <a href="#">PVSPLINE...ENDS</a> motion.
<a href="#">PROJECTION</a>	An expansion command to the <a href="#">MSEG...ENDS</a> 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.
<a href="#">PTP</a>	Creates a point-to-point motion.
<a href="#">PVSPLINE...ENDS</a>	Creates an arbitrary path motion with spline interpolation between the specified points.
<a href="#">READ</a>	Reads an array from a file in the flash memory.
<a href="#">RESUME</a>	Resumes program execution in a buffer.
<a href="#">SAFETYCONF</a>	Configures fault processing for one or more axes.
<a href="#">SAFETYGROUP</a>	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 <a href="#">KILL</a> and <a href="#">DISABLE</a> .

Command	Description
SEND	Same as <a href="#">DISP</a> , but also specifies the communication channel or channels.
SET	Defines the current value of either feedback ( <a href="#">FPOS</a> ), reference ( <a href="#">RPOS</a> ), or axis ( <a href="#">APOS</a> ) position.
SLAVE	Creates a master-slave motion.
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	<b>SPLIT</b> breaks apart an axis group. <b>SPLITALL</b> breaks apart all axis groups. See <a href="#">GROUP</a> .
START	Activates program execution in a buffer.
STARTPEG	Restarts PEG at the current position if <a href="#">STARTPEG</a> has been issued and the <i>last_point</i> has not been reached.
STOP	Terminates program execution in a buffer.
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 <a href="#">MSEG...ENDS</a> 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.

Command	Description
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 ( <a href="#">RPOS</a> ).
DEPENDS	Specifies a logical dependence between a motor and axes.
DISABLE	Shuts off one or more drives. <b>DISABLE ALL</b> provides <b>DISABLE</b> operation for all axes.
ENABLE/ENABLEALL	Activates one or more drives
FCLEAR	Clears faults.
GO	Starts a motion that was created using the <b>/w</b> command option.
GROUP	Defines an axis-group for coordinate multi-axis motion.
HALT	Terminates one or more motions using a third-order deceleration profile ( <a href="#">DEC</a> deceleration). <b>HALTALL</b> provides <b>HALT</b> operation for all axes.
IMM	Provides on the fly change of motion parameters.
KILL	Terminates one or more motions using a second-order deceleration profile and the <a href="#">KDEC</a> deceleration value. <b>KILLALL</b> provides <b>KILL</b> 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 ( <a href="#">FPOS</a> ), reference ( <a href="#">RPOS</a> ), or axis ( <a href="#">APOS</a> ) position.

Command	Description
SPLIT	<b>SPLIT</b> breaks apart an axis group - see <a href="#">GROUP</a> . <b>SPLITALL</b> breaks apart all axis groups.

## 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

### Syntax

**COMMUT** *axis* [,*excitation\_current*] [,*settle\_time*] [,*slope*]

### 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 <b>XRMS</b> .
<i>settle_time</i>	Optional - Determines settling time for the auto commutation process initiated by <b>COMMUT</b> . 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.

### 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\_Commute, acsc\_WaitMotorCommutated

### Example

```
COMMUT 0,80,100,30      !Commute 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/ENABLEALL, DISABLE** and **KILL** 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**

[Example](#)

```

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.
                                !Move to 1300. Each point during the motion
                                !is modified according to the correction
                                !ARRAY in the MAP function.
PTP 0, 1300
                                !End program
STOP

```

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

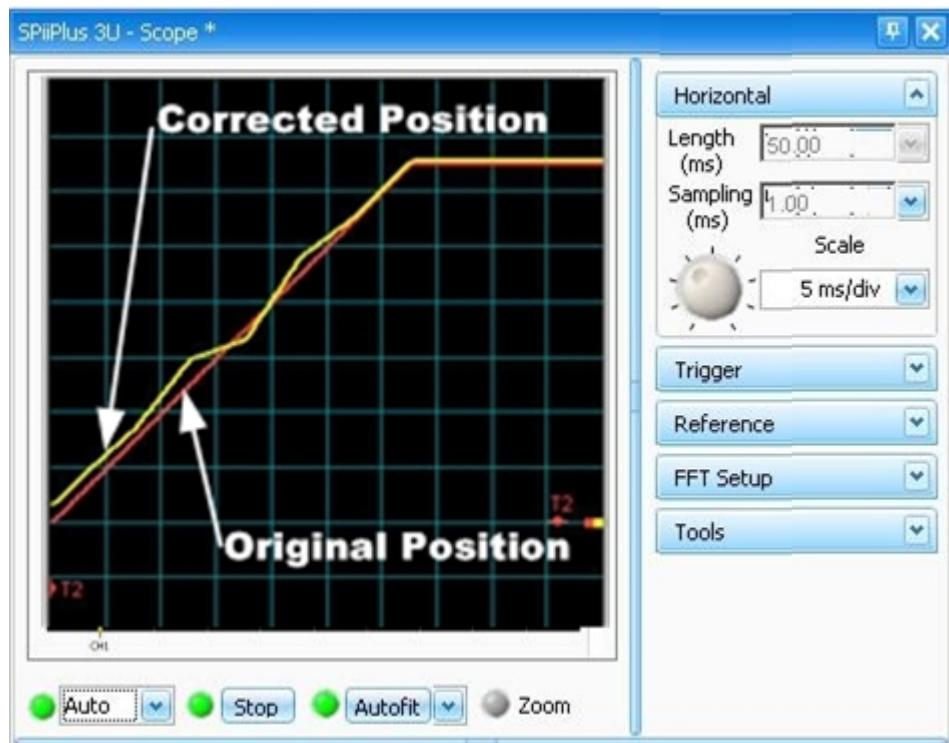


Figure 2-1. CONNECT Using MAP Function

## 2.1.4 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

<i>physical_axis</i>	Physical axis (motor).
<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

- 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](#).

## 2.1.5 DISABLE

### 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 <a href="#">MERR</a> variable. If the parameter is omitted, <a href="#">MERR</a> stores zero after the disable operation.

### Related ACSPL+ Commands

[ENABLE/ENABLEALL](#), [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

### Examples

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.

#### Example1:

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 <b>MERR</b> .
---------------------	--

#### Example 3:

DISABLE ALL	!Disable all motors.
-------------	----------------------

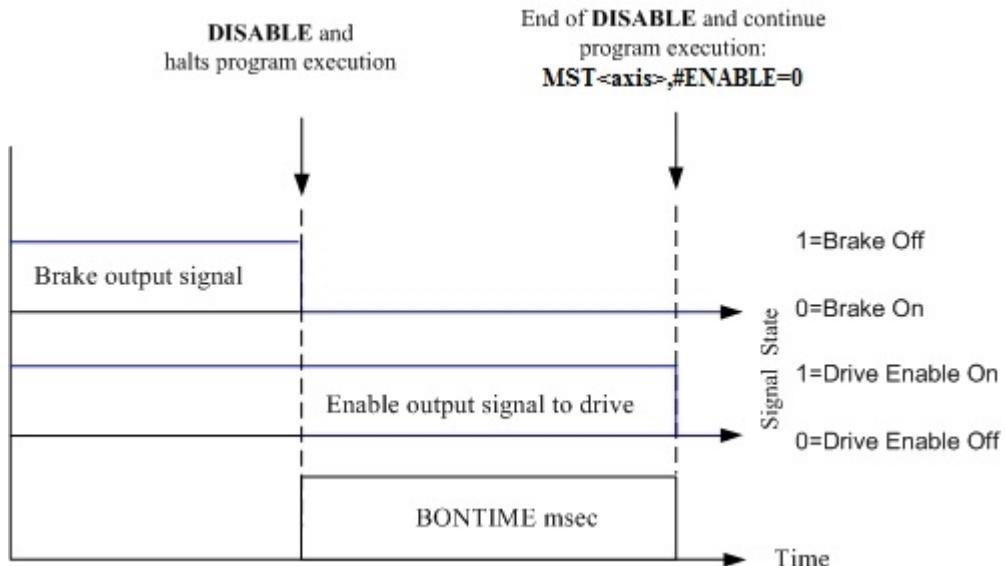


Figure 2-2. DISABLE and Mechanical Brake Output Process- Positive BONTIME

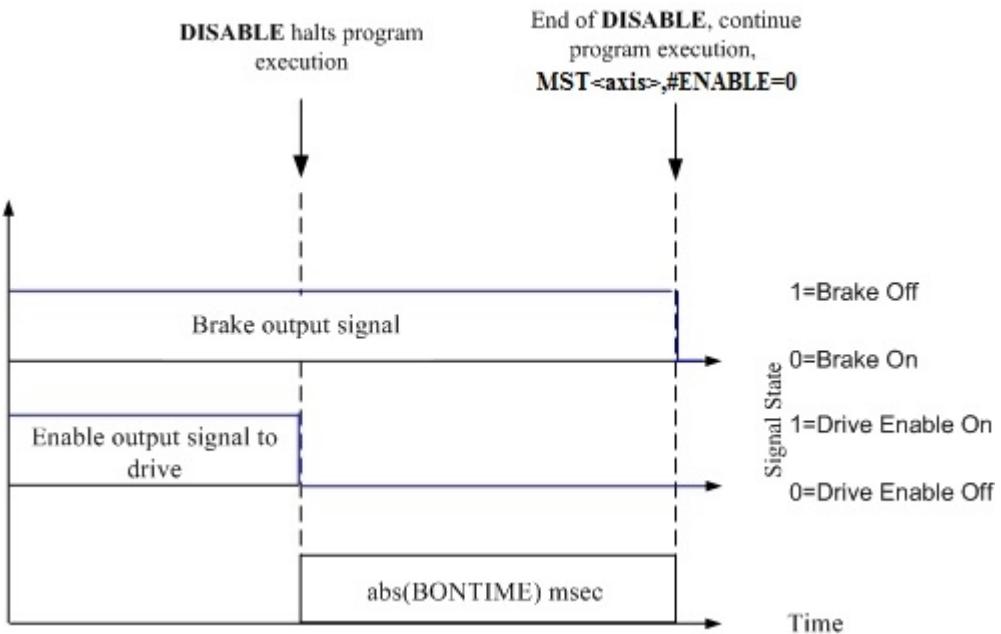


Figure 2-3. DISABLE and Mechanical Brake Output Process - Negative BONTIME

## 2.1.6 ENABLE/ENABLEALL

### Description

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

### Syntax

**ENABLE|ENABLEALL 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

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**

### 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

### Examples

**Example1:**

ENABLE 0

!Enable 0 axis.

### Example 2:

ENABLE (0,1)

!Enable 0 and 1 axes.

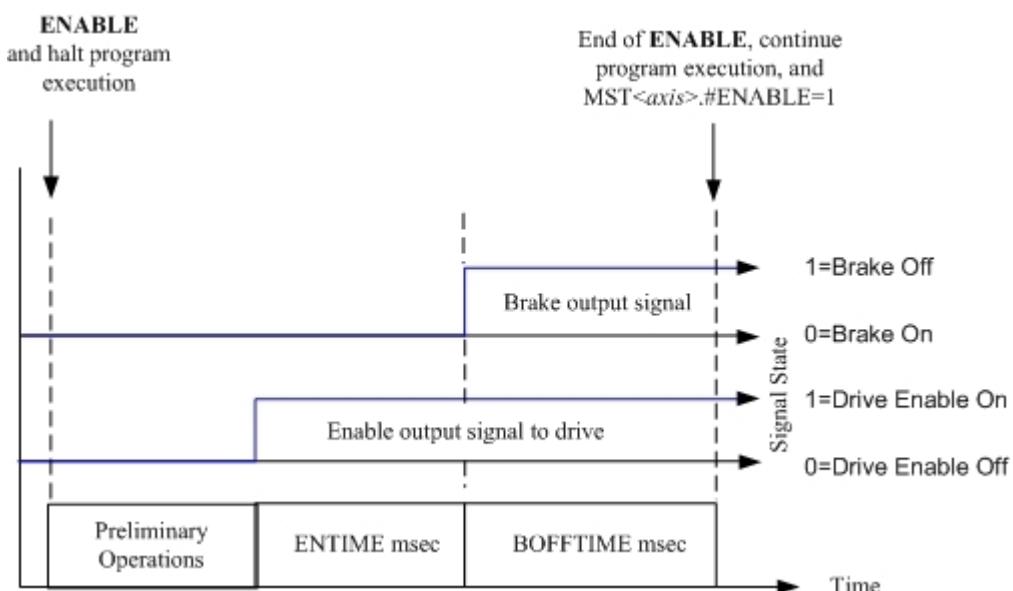


Figure 2-4. ENABLE and Mechanical Brake Output Process - Positive BOFFTIME

### 2.1.7 FCLEAR

#### Description

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

#### Syntax

**FCLEAR** *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

- 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

#### Examples

##### 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.

## 2.1.8 GO

#### Description

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

#### Syntax

**GO** *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

- 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
```

## 2.1.9 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

<b>axis_list</b>	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.
```

## 2.1.10 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

<b>axis_list</b>	List of axes to be halted.
<b>reason</b>	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. If the parameter is specified, its value is stored in the <a href="#">AERR</a> variable. If the parameter is omitted, AERR stores 5003 after HALT.

### Comments

**HALT ALL** terminates the motion of all axes.

### Related ACSPL+ Commands and .NET Library Methods

[KILL](#)

### COM Library Methods

Halt, HaltM

### C Library Functions

acsc\_Halt, acsc\_HaltM

### Examples

#### 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.
```

### 2.1.11 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

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

## 2.1.12 KILL

### Description

Use **KILL** after a safety event to decelerate and stop the axis faster than during normal deceleration and stop.

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

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



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/ENABLEALL](#).

## Related ACSPL+ Commands

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

## Related ACSPL+ Variables

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

## COM Library Methods and .NET Library Methods

Kill, Kill All

## C Library Functions

acsc\_Kill, acsc\_KillAll

## Examples

### 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.)
```

## 2.1.13 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>	A string enclosed in double quotation marks with one or more of the following characters determines the action of <b>SAFETYCONF</b> : <ul style="list-style-type: none"> <li>● K (<b>KILL</b>)</li> <li>● D (<b>DISABLE</b>)</li> <li>● KD (<b>KILL-DISABLE</b>)</li> <li>● + 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)</li> <li>● - applies <b>FMASK&lt;axis&gt;.#fault-name = 0</b> for all axes of the controller</li> </ul>

### 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.

### Related ACSPL+ Commands

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

### Examples

#### 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
                                !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.
------------------------	--

### 2.1.14 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** and **DISABLE**.

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

**Syntax**

**SAFETYGROUP** *axis\_list*

**Arguments**

<b>axis_list</b>	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.
---------------	---

### 2.1.15 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

<b>axis_RPOS</b>	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
<b>value or formula</b>	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

## 2.1.16 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

<b>axis_list</b>	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

### Examples

### 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.

## 2.2 Interactive Commands

The Interactive commands are:

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

### 2.2.1 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

<b>argument</b>	An expression or string where:  <b>Expression:</b> ACSPL+ expression as one or more variables  <b>String:</b> "[text] [escape-sequence] [format-specifier]" String must be enclosed with double quotation marks.
<b>[, argument...]</b>	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 2-2. 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 <b>HH</b> 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:

<b>[width]</b>	Optional specification for the minimum number of characters in the output. If <b>width</b> is smaller than the number of digits of a displayed number, the specified <b>width</b> is ignored, and the displayed number includes all digits.
<b>[.precision]</b>	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.
<b>type</b>	Required character that determines whether the associated argument is interpreted as a character, a string, or a number, described in <a href="#">2.2.1</a> .

**Table 2-3. Type Characters**

Type	Output Format
<b>d</b>	Signed decimal integer
<b>u</b>	Unsigned decimal integer

Type	Output Format
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](#)

### Examples

#### 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 "%1i",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
```

## 2.2.2 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

	Communication channel index:
<i>channel</i>	<ul style="list-style-type: none"> <li>● 1 - serial communication channel COM1</li> <li>● 2 - serial communication channel COM2</li> <li>● 6 - Ethernet network (TCP)</li> <li>● 7 - Ethernet network (TCP)</li> <li>● 8 - Ethernet network (TCP)</li> <li>● 9 - Ethernet network (TCP)</li> <li>● 10 - Ethernet Point-to-Point (UDP)</li> <li>● 12 - PCI bus</li> <li>● 16 - communication channel with <b>Modbus</b> slave</li> <li>● 36 - Ethernet network (TCP)</li> <li>● 37 - Ethernet network (TCP)</li> <li>● 38 - Ethernet network (TCP)</li> <li>● 39 - Ethernet network (TCP)</li> </ul>
<i>array</i>	User-defined integer array to which the data will be stored. If <b>array</b> is omitted, the function purges formerly received characters in the channel.
<i>start_index</i>	The first received character is assigned to the array element with the specified index. If <b>start_index</b> is omitted, the assignment starts from the first element of the array.

<b>number</b>	The number of characters to be collected to the variable array. If <b>number</b> is omitted, the function continues receiving characters until the last element of the array is assigned, or the carriage return character is received.
<b>timeout</b>	The function waits for input not more than the specified number of milliseconds. If <b>timeout</b> is omitted, the waiting time is not limited.

### 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

```

## 2.2.3 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\_ACSPY\_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

#### Examples

#### 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_ACSPLOPROGRAM)
                      !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_ACSPLOPROGRAM(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;
}

```

## 2.2.4 INTERRUPTEX

### Description

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

- It triggers the dedicated call-back ACSC\_INTR\_ACSP<sub>L</sub>\_PROGRAM\_EX (21)
- It receives two mandatory integer 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 100
  - 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*)

### Arguments

<b>32-bit_high_value</b>	Most significant value of a combined 64-bit integer
<b>32-bit_low_value</b>	Least significant value of a combined 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+(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

### Examples

#### Example 1:

```
Enable 0                      ! Enable axis 0
SET FPOS0=0                   ! Set axis 0 position to 0
PTP/e 0,1000                  ! Move to position 1000
INTERRUPTEX (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
INTERRUPTEX (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
INTERRUPTEX (0x00, interrupt_queue)      ! Interrupt description in
                                              ! interrupt_description (1)
Stop
```

## 2.2.5 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>	An integer that defines the communication channel to which the message will be sent, as follows:  -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>	An expression or string where:  <b>Expression:</b> ACSPL+ expression as one or more variables <b>String:</b> "[text] [escape-sequence] [format-specifier]" String must be enclosed with double quotation marks.
<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

## 2.2.6 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

<b>channel</b>	An integer number from 0 to 7 that specifies the trigger's sequential number. The number defines the <b>AST</b> 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.
<b>expression</b>	Specifies the triggering condition. After <b>TRIGGER</b> 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.
<b>timeout</b>	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 2-4. Channel Designation for TRIGGER

Channel	Triggering Bit	Hexadecimal Interrupt Tag (Software Interrupt 10)
0	AST0.11	0x00000001
1	AST1.11	0x00000002

Channel	Triggering Bit	Hexadecimal Interrupt Tag (Software Interrupt 10)
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".
```

## 2.2.7 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

<i>channel</i>	Communication channel index: <ul style="list-style-type: none"> <li>● 1 - serial communication channel COM1.</li> <li>● 2 - serial communication channel COM2.</li> <li>● 6 - Ethernet network (TCP)</li> <li>● 7 - Ethernet network (TCP)</li> <li>● 8 - Ethernet network (TCP)</li> <li>● 9 - Ethernet network (TCP)</li> <li>● 10 - Ethernet Point-to-Point (UDP)</li> <li>● 12 - PCI bus</li> <li>● 16 - communication channel with <b>MODBUS</b> slave</li> <li>● 36 - Ethernet network (TCP)</li> <li>● 37 - Ethernet network (TCP)</li> <li>● 38 - Ethernet network (TCP)</li> <li>● 39 - Ethernet network (TCP)</li> </ul>
<i>variable</i>	User-defined integer array from which the data will be sent.
<i>start_index</i>	The index in the array from which to start. If <b>start_index</b> is omitted, the transmission starts from the first element of the array.
<i>number</i>	The number of characters to be transmitted from the variable array.

### Return Value

The number of entities that have been transmitted.

### Error Conditions

If the function fails, an error is generated.

## 2.3 PEG and MARK Commands

Command	Description
<a href="#">ASSIGNMARK</a>	Assigns MARK inputs pins to encoders.
<a href="#">ASSIGNPEG</a>	Assigns PEG engines to encoders.

Command	Description
<a href="#">ASSIGNPOUTS</a>	Assigns PEG outputs.
<a href="#">PEG_I</a>	Activates incremental PEG, where pulses are generated at predefined, fixed position intervals.
<a href="#">PEG_R</a>	Activates random, table-based PEG where predefined pulses are generated at pre-defined positions.
<a href="#">STARTPEG</a>	Starts PEG motion for specified axis.
<a href="#">STOPPEG</a>	Terminates PEG motion.

### 2.3.1 ASSIGNMARK

#### Description

The **ASSIGNMARK** function is used for Mark inputs-to-encoder assignment. It allows mapping available physical Mark inputs pins to different encoders.

#### Syntax

**ASSIGNMARK** [/i] *axis, mark\_type, inputs\_to\_encoder\_bit\_code*

#### Arguments

<b>Axis</b>	The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>mark_type</b>	1 for Mark-1 2 for Mark-2
<b>inputs_to_encoder_bit_code</b>	Bit code for inputs-to-encoders mapping according to <a href="#">Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x</a> <a href="#">Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x</a> <a href="#">Mark-1 and Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x</a>

#### Comments

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

#### Mark Inputs to Encoders Mapping

Table 2-5. Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x

Mark-1 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x									
Bit code	Encoder 0 (X)	Encoder 1 (Y)	Encoder 4 (Z)	Encoder 5 (T)	Encoder 2 (A)	Encoder 3 (B)	Encoder 6 (C)	Encoder 7 (D)	
0000 0 (default)	0(X)	1(Y)	4(Z)	5(T)	-	-	-	-	
0000 1	1(Y)	4(Z)	5(T)	0(X)	-	-	-	-	
0001 0	4(Z)	5(T)	0(X)	1(Y)	-	-	-	-	
00011	5(T)	0(X)	1(Y)	4(Z)	-	-	-	-	
0010 0	-	1(Y)	4(Z)	5(T)	0(X)	-	-	-	
00101	0(X)	-	4(Z)	5(T)	-	1(Y)	-	-	
00110	0(X)	1(Y)	-	5(T)	-	-	4(Z)	-	
00111	0(X)	1(Y)	4(Z)	-	-	-	-	5(T)	
0100 0	-	-	4(Z)	5(T)	0(X)	1(Y)	-	-	
01001	0(X)	1(Y)	-	-	-	-	4(Z)	5(T)	
01010	0(X)	-	-	-	-	1(Y)	4(Z)	5(T)	
01011	-	1(Y)	-	-	0(X)	-	4(Z)	5(T)	
01100	-	-	4(Z)	-	0(X)	1(Y)	-	5(T)	
01101	-	-	-	5(T)	0(X)	1(Y)	4(Z)	-	
01110	-	-	-	-	0(X)	1(Y)	4(Z)	5(T)	
01111	-	-	-	-	1(Y)	4(Z)	5(T)	0(X)	
1000 0	-	-	-	-	4(Z)	5(T)	0(X)	1(Y)	
10001	-	-	-	-	5(T)	0(X)	1(Y)	4(Z)	



The Bit Code affects all of the inputs in the row, see [Bit code affects](#).

Table 2-6. Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x

Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x								
Bit code	Encoder 0 (X)	Encoder 1 (Y)	Encoder 4 (Z)	Encoder 5 (T)	Encoder 2 (A)	Encoder 3 (B)	Encoder 6 (C)	Encoder 7 (D)
0000 0 (default)	GP IN6	GP IN7	GP IN4	GP IN5	-	-	-	-
0000 1	GP IN7	GP IN4	GP IN5	GP IN6	-	-	-	-
0001 0	GP IN4	GP IN5	GP IN6	GP IN7	-	-	-	-
00011	GP IN5	GP IN6	GP IN7	GP IN4	-	-	-	-
0010 0	-	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
0100 0	-	-	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
01011	-	GP IN7	-	-	GP IN6	-	GP IN4	GP IN5

Mark-2 Inputs to Encoders Mapping for SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x									
Bit code	Encoder 0 (X)	Encoder 1 (Y)	Encoder 4 (Z)	Encoder 5 (T)	Encoder 2 (A)	Encoder 3 (B)	Encoder 6 (C)	Encoder 7 (D)	
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	

### Bit code affects

The Bit code affects all of the inputs in the row. For example, Bit code: 00010 performs the following assignments for these inputs:

- Mark-1 4(Z) - Encoder 0(X)
- Mark-1 5(T) - Encoder 1(Y)
- Mark-1 0(X) - Encoder 4(Z)
- Mark-1 1(Y) - Encoder 5(T)

Table 2-7. Mark-1 and Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x

Mark-1 and Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x									
Bit code	Encoder 0(X)		Encoder 1(Y)		Encoder 2(A)		Encoder 3(B)		
	Mark-1 Input	Mark-2 Input	Mark-1 Input	Mark-2 Input	Mark-1 Input	Mark-2 Input	Mark-1 Input	Mark-2 Input	
000 (deafult)	Mark-1 0 (X)	Mark-2 0 (X)	Mark-1 11 (Y)	Mark-2 11 (Y)	Mark-1 0 (Y)	GP IN6	Mark-1 11 (Y)	GP IN7	

Mark-1 and Mark-2 Inputs to Encoders Mapping for with SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x

Bit code	Encoder 0(X)		Encoder 1(Y)		Encoder 2(A)		Encoder 3(B)	
	Mar k-1 Input	Mar k-2 Input						
001	GP IN6	Mar k-11 (Y)	Mar k-10 (X)	Mar k-11 (Y)	Mar k-20 (X)	Mar k-11 (Y)	Mar k-10 (X)	Mar k-11 (Y)
010	-	Mar k-21 (Y)	GP IN4	GP IN5	GP IN6	Mar k-21 (Y)	GP IN6	-
011	-	GP IN7	GP IN6	GP IN7	-	GP IN7	-	-

### 2.3.2 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. The parameters are different from the original SPiiPlus definitions.

#### Syntax

**ASSIGNPEG** [/f] axis, engines\_to\_encoders\_code, gp\_out\_assign\_code

#### 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.
engines_to_encoders_code	Bit code for engines-to-encoders mapping according to:  <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 0)</a>  <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 1)</a>  <a href="#">SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x/UDMnt-x Mapping PEG Engines to Encoders (Servo Processor 0)</a>  <a href="#">UDMlc-x/UDIlt-x/UDIhp-x/UDMmc-x/PDIcl-x/ Mapping PEG Engines to Encoders (Servo Processor 0)</a>  <a href="#">NPMpm-x/NPMpc-x Mapping PEG Engines to Encoders (Servo Processor 0)</a>

### gp\_out\_assign\_code

General purpose outputs assignment to use as PEG pulse outputs according to:

[SPIIPlusNT/DC-LT/HP/LD-x/SPIIPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs \(Servo Processor 0\)](#)

[SPIIPlusNT/DC-LT/HP/LD-x/SPIIPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs \(Servo Processor 1\)](#)

[SPIIPlus CMnt-x/UDMpm-x/CMhv-x/UDMhv-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs \(Servo Processor 0\)](#)

[UDMnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs \(Servo Processor 0\)](#)



The **axis** parameter actually serves for determining which Servo Processor is 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 switch: /f is included, fast loading of Random PEG arrays is activated.



HSSI devices (HSSI-I016, 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.

### Mapping PEG engines to encoders

**Table 2-8. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 0)**

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 0)				
Bit Code	Encoder 0(X)	Encoder 1(Y)	Encoder 2(A)	Encoder 3(B)
000 (default)	PEG0	PEG1	PEG2	no
001	PEG0	PEG1	no	PEG2
010	PEG0 PEG2	PEG1	no	no
011	PEG0	PEG1 PEG2	no	no
100	PEG0 PEG1 PEG2	no	no	no
101	no	PEG0 PEG1 PEG2	no	no



The **Bit Code** affects all of the connectors in the row, see [Bit Code assignment example](#).

**Table 2-9. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 1)**

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 1)				
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	no	no

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 1)				
Bit Code	Encoder 4(Z)	Encoder 5(T)	Encoder 6(C)	Encoder 7(D)
		PEG6		
100	PEG4 PEG5 PEG6	no	no	no
101	no	PEG4 PEG5 PEG6	no	no



The **Bit Code** affects all of the connectors in the row, see [Bit Code assignment example](#).

Table 2-10. SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x/UDMnt-x Mapping PEG Engines to Encoders (Servo Processor 0)

SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x/UDMnt-x Mapping PEG Engines to Encoders (Servo Processor 0)				
Bit Code	Encoder 0(X)	Encoder 1(Y)	Encoder 2(A)	Encoder 3(B)
000 (default)	PEG0	PEG1	PEG2 <sup>2</sup>	
001	PEG0	PEG1	no	PEG2 <sup>1,2</sup>
010	PEG0 PEG2 <sup>2</sup>	PEG1	no	
011	PEG0	PEG1 PEG2 <sup>2</sup>	no	
100	PEG0 PEG1 PEG2 <sup>2</sup>	no	no	
101	no	PEG0 PEG1 PEG2 <sup>2</sup>	no	

SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMb-a-x/UDMhp-x/CMhv-x/UDMhv-x/UDMnt-x Mapping PEG Engines to Encoders (Servo Processor 0)				
Bit Code	Encoder 0(X)	Encoder 1(Y)	Encoder 2(A)	Encoder 3(B)
110			PEG0 <sup>1,2</sup> PEG1 <sup>1,2</sup> PEG2 <sup>1,2</sup>	
111			PEG0 <sup>1,2</sup> PEG2 <sup>1,2</sup>	PEG1 <sup>1,2</sup>

<sup>1</sup>These combinations are **not** supported by UDMpc-x.

<sup>2</sup>These combinations are **not** supported by UDMnt-x.



The **Bit Code** affects all of the connectors in the row, see [Bit Code assignment example](#).

Table 2-11. UDMlc-x/UDlIt-x/UDlhp-x/UDMmc-x/PDIcl-x/ Mapping PEG Engines to Encoders (Servo Processor 0)

UDMlc-x/UDlIt-x/UDlhp-x/UDMmc-x/PDIcl-x/ Mapping PEG Engines to Encoders (Servo Processor 0)				
Bit Code	Encoder 0(X)	Encoder 1(Y)	Encoder 2(A)	Encoder 3(B)
000 (default)	PEG0	no	no	no
001	no	PEG0	no	no
010	no	no	PEG0	no
011	no	no	no	PEG0 <sup>1</sup>
100	no	no	no	no
101	no	no	no	no
110	no	no	no	no

<sup>1</sup>These combinations are **not** supported by LCM-x.



The **Bit Code** affects all of the connectors in the row, see [Bit Code assignment example](#).

Table 2-12. NPMpm-x/NPMpc-x Mapping PEG Engines to Encoders (Servo Processor 0)

NPMpm-x/NPMpc-x Mapping PEG Engines to Encoders (Servo Processor 0)				
Bit Code	Encoder 0(X)	Encoder 1(Y)	Encoder 2(A)	Encoder 3(B)
000 (default)	PEG0	PEG1	no	no
001	PEG1	PEG0	no	no
010	PEG0 PEG1	no	no	no
011	no	PEG0 PEG1	no	no
100	no	PEG1	PEG0	no
101	no	PEG1	no	PEG0
110	PEG0	no	PEG1	no
111	PEG0	no	no	PEG1

#### Bit Code assignment example

The **Bit Code**: 001 for an axis associated with Servo Processor 0 performs the following assignments:

- PEG0 → Encoder 0
- PEG1 → Encoder 1
- PEG2 → Encoder 3

Or for CMnt-x/UDMpm-x/CMhv-x/UDMhv-x:

- PEG0 → Encoder 0(X)
- PEG1 → Encoder 1(A)

For an axis associated with Servo Processor 1 it performs the following assignments:

- PEG4 → Encoder 4
- PEG5 → Encoder 5
- PEG6 → Encoder 7

#### General purpose outputs assignment to use as PEG pulse outputs

Table 2-13. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)				
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	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	PEG0_PULSE	GP Out 1	PEG1_PULSE	GP Out 3
0111	PEG0_PULSE	PEG2_PULSE	PEG1_PULSE	Reserved
1000 - 1111	Reserved	Reserved	Reserved	Reserved



The **Bit Code** affects the entire row, see [Bit Code assignment example](#).

Table 2-14. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1)

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1)				
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
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

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1)

Bit Code	GP Out 4	GP Out 5	GP Out 6	GP Out 7
0111	PEG4_PULSE	PEG6_PULSE	PEG5_PULSE	Reserved
1000 - 1111	Reserved	Reserved	Reserved	Reserved



The **Bit Code** affects the entire row, see [Bit Code assignment example](#).

Table 2-15. SPiiPlus CMnt-x/UDMpm-x/CMhv-x/UDMhv-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)

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	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
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



The **Bit Code** affects the entire row, see [Bit Code assignment example](#).

Table 2-16. UDMnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)

UDMnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)		
Bit Code	GP Out 0	GP Out 1
0000 (default)	GP Out 0	GP Out 1
0001	PEG0_PULSE	GP Out 1
0010	GP Out 0	PEG1_PULSE
0011	PEG1_PULSE	GP Out 1
0100	GP Out 0	PEG0_PULSE
0101	PEG0_PULSE	PEG1_PULSE
0110 - 1111	Reserved	Reserved

#### Bit Code assignment example

For example, for an axis associated with Servo Processor 0, **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.

#### Comments

**ASSIGNPEG** 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.

The **axis** parameter can be any of the axes controlled by the same servo processor, the result will be the same.

#### Related ACSPL+ Commands

[PEG\\_I](#), [ASSIGNPOUTS](#), [PEG\\_R](#), [STARTPEG](#), [STOPPEG](#)

#### COM Library Methods

None

#### C Library Functions

acsc\_AssignPegNT

### 2.3.3 ASSIGNPOUTS

#### Description

The **ASSIGNPOUTS** function is used for setting Fast General Purpose output pins assignment 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 below 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. It needs to be noted that some of the signals cannot be routed to the physical pins.

## Syntax

**ASSIGNPOUTS** *axis, output\_index, bit\_code*

## Arguments

<b>axis</b>	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.
<b>output_index</b>	0 for <b>OUT_0</b> , 1 for <b>OUT_1</b> , ..., 9 for <b>OUT_9</b>
<b>bit_code</b>	Bit code for engine outputs to physical outputs mapping according to:  <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 1)</a> <a href="#">CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> <a href="#">CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> <a href="#">CMba-x/CMhp-x/UDMb-a-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> <a href="#">CMba-x/CMhp-x/UDMb-a-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a>

UDMnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)  
UDMlc-x/UDMmc-x/UDlIt-x/UDlhp-x/PDlcl-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)  
NPMpm-x/NPMpc-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

### Mapping of PEG engine outputs to physical outputs

Table 2-17. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)					
Bit Code	OUT_4 (HSSI1_DO)	OUT_3 (T_PEG)	OUT_2 (Z_PEG)	OUT_1 (Y_PEG)	OUT_0 (X_PEG)
000 (default)	HSSI1_DO	PEG5_PULSE	PEG4_PULSE	PEG1_PULSE	PEG0_PULSE
001	PEG0_STATE1	PEG0_STATE0	PEG2_PULSE	PEG1_STATE0	PEG4_STATE0
010	PEG2_STATE1	PEG2_STATE0	PEG1_STATE1	Reserved	Reserved
011	Reserved	Reserved	Reserved	Reserved	Reserved
100	Reserved	Reserved	Reserved	Reserved	Reserved
111	Reserved	FGP_OUT3	FGP_OUT2	FGP_OUT1	FGP_OUT0



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

Table 2-18. SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 1)

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 1)					
Bit Code	OUT_9HSSIO_CON	OUT_8HSSIO_DO	OUT_7X_STATE2	OUT_6X_STATE1	OUT_5X_STATE0
000 (default)	HSSIO_CON	HSSIO_DO	PEG0_STATE2	PEG0_STATE1	PEG0_STATE0

SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 1)					
Bit Code	OUT_9HSSIO_CON	OUT_8HSSIO_DO	OUT_7X_STATE2	OUT_6X_STATE1	OUT_5X_STATE0
001	PEG0_STATE0	PEG0_STATE2	PEG2_STATE0	PEG1_STATE1	PEG1_STATE0
010	PEG2_STATE1	PEG5_STATE0	PEG6_PULSE	PEG5_PULSE	PEG4_PULSE
011	PEG6_STATE1	Reserved	PEG4_STATE1	PEG4_STATE0	Reserved
100	Reserved	Reserved	Reserved	Reserved	Reserved
111	Reserved	Reserved	FGP_OUT6	FGP_OUT5	FGP_OUT4



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

Table 2-19. CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)		
Bit Code	OUT_1 (Y_PEG)	OUT_0 (X_PEG)
000 (default)	PEG1_PULSE	PEG0_PULSE
001	Encoder X Phase B	Encoder X Phase A
010	Encoder Y Phase B	Encoder Y Phase A
011	Reserved	Reserved
100	Reserved	Reserved
111	FGP_OUT1	FGP_OUT0



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

**Table 2-20. CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)**

CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)		
Bit Code	OUT_6 (X_STATE1)	OUT_5 (X_STATE0)
000 (default)	PEG0_OUT1	PEG0_OUT0
001	PEG1_OUT1	PEG1_OUT0
010	PEG2_OUT1	PEG2_OUT0
011	Encoder X Phase B	Encoder X Phase A
100	Encoder Y Phase B	Encoder Y Phase A
101	Encoder X INDEX	Encoder X INDEX
110	Encoder Y INDEX	Reserved
111	Reserved	Reserved

**Table 2-21. CMba-x/CMhp-x/UDMbba-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)**

CMba-x/CMhp-x/UDMbba-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)					
Bit Code	OUT_4 (Y_STATE2)	OUT_3 (Y_STATE1)	OUT_2 (Y_STATE0)	OUT_1 (Y_PEG)	OUT_0 (X_PEG)
000 (default)	PEG1_STATE2	PEG1_STATE1	PEG1_STATE0	PEG1_PULSE	PEG0_PULSE
001	PEG0_STATE2	PEG0_STATE1	PEG0_STATE0	Encoder X Phase B	Encoder X Phase A
010	PEG2_STATE2	PEG2_STATE1	PEG2_STATE0	Encoder Y Phase B	Encoder Y Phase A
011	Encoder A INDEX	Encoder A Phase B	Encoder A Phase A	Encoder A Phase B	Encoder A Phase A
100	Reserved	Reserved	Reserved	PEG2_PULSE	Reserved

CMba-x/CMhp-x/UDMb-a-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

Bit Code	OUT_4 (Y_STATE2)	OUT_3 (Y_STATE1)	OUT_2 (Y_STATE0)	OUT_1 (Y_PEG)	OUT_0 (X_PEG)
101	Reserved	Reserved	Reserved	PEG2_PULSE or PEG1_PULSE	PEG0_PULSE or PEG2_PULSE
110	Reserved	Reserved	Reserved	PEG0_PULSE or PEG1_PULSE or PEG2_PULSE	PEG0_PULSE or PEG1_PULSE or PEG2_PULSE
111	Reserved	Reserved	FGP_OUT2	FGP_OUT1	FGP_OUT0



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



For Bit Codes 101 and 110, the OUT\_0 (X\_PEG) and OUT\_1 (Y\_PEG) mappings are supported for default PEG pulse polarity only.

Table 2-22. CMba-x/CMhp-x/UDMb-a-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

Bit Code	OUT_9 (Y_STATE3)	OUT_8 (X_STATE3)	OUT_7 (X_STATE2)	OUT_6 (X_STATE1)	OUT_5 (X_STATE0)
000 (default)	PEG1_STATE3	PEG0_STATE3	PEG0_STATE2	PEG0_STATE1	PEG0_STATE0
001	PEG0_STATE3	PEG1_STATE3	PEG1_STATE2	PEG1_STATE1	PEG1_STATE0
010	PEG2_STATE3	PEG2_STATE3	PEG2_STATE2	PEG2_STATE1	PEG2_STATE0
011	PEG2_STATE0	Encoder Y Phase B	Encoder Y Phase A	Encoder X Phase B	Encoder X Phase A
100	PEG2_STATE1	Encoder Y INDEX	Reserved	Encoder Y Phase B	Encoder Y Phase A

CMba-x/CMhp-x/UDMb-a-x/UDMhp-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)					
Bit Code	OUT_9 (Y_STATE3)	OUT_8 (X_STATE3)	OUT_7 (X_STATE2)	OUT_6 (X_STATE1)	OUT_5 (X_STATE0)
101	Reserved	Reserved	Reserved	Encoder X INDEX	Encoder X INDEX
110	Reserved	PEG2_PULSE	PEG2_PULSE	Encoder Y INDEX	PEG2_PULSE
111	Reserved	Reserved	Reserved	Reserved	Reserved

Table 2-23. UDMnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

UDMnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)		
Bit Code	OUT_1 (X_STATE1)	OUT_0 (X_STATE0)
000 (default)	PEG1_PULSE	PEG0_PULSE
001	Encoder X Phase B	Encoder X Phase A
010	Encoder Y Phase B	Encoder Y Phase A
011	PEG0_STATE0	PEG1_STATE0
100	PEG1_STATE0	PEG0_STATE0
101	Reserved	Reserved
110	Reserved	Reserved
111	FGP_OUT1	FGP_OUT0

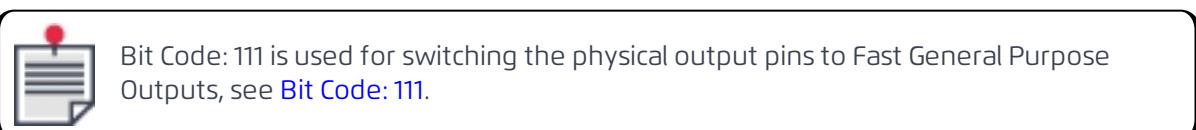


Table 2-24. UDMlc-x/UDMmc-x/UDlIt-x/UDlhp-x/PDlcl-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

UDMlc-x/UDMmc-x/UDlIt-x/UDlhp-x/PDIcl-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)	
Bit Code	OUT_0 (X_PEG)
000 (default)	PEGO_PULSE
001	Reserved
010	Reserved
011	Reserved
100	Reserved
101	Reserved
110	Reserved
111	FGP_OUT0



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

Table 2-25. NPMpm-x/NPMpc-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)

NPMpm-x/NPMpc-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)		
Bit Code	OUT_1 (Y_PEG)	OUT_0 (X_PEG)
000 (default)	PEG1_PULSE	PEGO_PULSE
001	PEG1_STATE0	PEGO_STATE0
010	PEGO_STATE0	PEG1_STATE0
011	PEG1_STATE1	PEGO_STATE1
100	PEGO_STATE1	PEG1_STATE1
101	Reserved	Reserved
110	Reserved	Reserved
111	FGP_OUT1	FGP_OUT0

[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

### 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.

### Examples

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

#### Example 1:

```
ASSIGNPOUTS 0, 2, 0b111
```

This defines the **Z\_PEG** output as **FGP\_OUT2** and maps it to the bit 18 of the ACSPL+ **OUT** variable (see [SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs \(Servo Processor 0\)](#)).

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, 0b111
```

This defines the **X\_STATE2** output as **FGP\_OUT6** and maps it to the bit 22 of the ACSPL+ **OUT** variable (see [SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs \(Servo Processor 1\)](#)).



A separate command should be set for every GP output.

### Related ACSPL+ Commands

[ASSIGNPEG](#), [PEG\\_I](#), [PEG\\_R](#), [STARTPEG](#), [STOPPEG](#)

### COM Library Methods

None

### C Library Functions

acsc\_AssignPegOutputsNT

## 2.3.4 PEG\_I

### Description

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

### Syntax

**PEG\_I** [/awij] *axis, width, first\_point, interval, last\_point*

### Arguments

<b>axis</b>	The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>width</b>	Width of the Pulse.
<b>first_point</b>	First point for the PEG generation.
<b>interval</b>	The distance between PEG events.
<b>last_point</b>	Last point for PEG generation.
<b>time-based-pulses</b>	Optional parameter - number of time-based pulses generated after each encoder-based pulse, the range is from 0 to 65,535.
<b>time-based-period</b>	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).



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.

## Example

In this example, PEG pulses are fired on axis 6 based on axis encoder 7.

```
GLOBAL AXIS6
GLOBAL AXIS7
AXIS6=6 ! Axis assignment
AXIS7=7

assignpeg AXIS6, 0b001, 0b0
! Refer to SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders \(Servo Processor 1\): last column indicates that Encoder 7 is assigned to PEG6.
! 0b0 indicates that the digital outputs are at their default value according
! to SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs \(Servo Processor 1\), thus not used as PEG signals.

assignpouts AXIS6, 7, 0b010
! Axis6 being assigned. Output 7 is used. 0b010 indicates PEG6_PULSE is being
! fired (from SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs \(Servo Processor 1\)).

ST:
peg_i AXIS6,0.5,-100,-200,-10000
TILL AST(AXIS6).#PEGREADY
!Wait till command executes and configuration is set, in order to ensure
!proper PEG engine's execution prior to start of movement
ptp/e AXIS6,-12000
stoppeg AXIS6 ! Prevent PEG pulses' firing on the 'way back'
ptp/e AXIS6,0
goto ST

stop
```

## Related ACSPL+ Commands

[PEG\\_R](#), [ASSIGNPEG](#), [ASSIGNPOUTS](#), [STARTPEG](#), [STOPPEG](#)

## Related ACSPL+ Variables

AST

COM Library Methods

None

C Library Functions

acsc\_PegIncNT, acsc\_WaitPegReady

### 2.3.5 PEG\_R

#### Description

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

#### Syntax

**PEG\_R** [/wi] *axis, width, mode, first\_index, last\_index, POS\_ARRAY, STATE\_ARRAY*

#### Arguments

<b>axis</b>	The axis index, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>width</b>	Width of the Pulse.
<b>mode</b>	<p>PEG state output signals configuration according to <a href="#">PEG Output Signal Configuration</a>.</p> <p>Bits 0-3 relates the PEG State 0 of the specific PEG engine</p> <p>Bits 4-7 relates the PEG State 1 of the specific PEG engine</p> <p>Bits 8-11 relates the PEG State 2 of the specific PEG engine</p> <p>Bits 12-15 relates the PEG State 3 of the specific PEG engine</p> <p>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).</p>
<b>first_index</b>	Index of first entry in the array for PEG generation
<b>last_index</b>	Index of last entry in the array for PEG generation
<b>POS_ARRAY</b>	The Random Event Positions array, maximum of 256/1024 members
<b>STATE_ARRAY</b>	The Outputs States array defining the four PEG output states, maximum of 256/1024 members
<b>time-based-pulses</b>	Optional parameter - number of time-based pulses generated after each encoder-based pulse, the range is from 0 to 65,535.

**time-based-period**

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 2-26. PEG Output Signal Configuration**

PEG Output Signal Configuration			
Bit	Signal	Description	Default Value
0	Pulse polarity	0- Output-0 positive pulse 1- Output-0 negative pulse	'0'
1	State polarity	0- Output-0 positive state 1- Output-0 negative state	'0'
2-3	Output type	00- Output-0 three state 01- Output-0 State 10- Output-0 Pulse 11- Output-0 Pulse&State	'00'
4	Pulse polarity	0- Output-1 positive pulse 1- Output-1 negative pulse	'0'
5	State polarity	0- Output-1 positive state 1- Output-1 negative state	'0'
7-6	Output type	00- Output-1 three state 01- Output-1 State 10- Output-1 Pulse 11- Output-1 Pulse&State	'00'
8	Pulse polarity	0- Output-2 positive pulse 1- Output-2 negative pulse	'0'
9	State polarity	0- Output-2 positive state 1- Output-2 negative state	'0'
11-10	Output type	00- Output-2 three state 01- Output-2 State 10- Output-2 Pulse 11- Output-2 Pulse&State	'00'

PEG Output Signal Configuration			
Bit	Signal	Description	Default Value
12	Pulse polarity	0- Output-3 positive pulse 1- Output-3 negative pulse	'0'
13	State polarity	0- Output-3 positive state 1- Output-3 negative state	'0'
15-14	Output type	00- Output-3 three state 01- Output-3 State 10- Output-3 Pulse 11- Output-3 Pulse&State	"00"

#### 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 included, the PEG pulse output signal is inverted.
- 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.

### Example

In this example PEG pulses are fired on axes 0, 1, and 2 according to encoder of axis 0.

```

GLOBAL AXIS0
GLOBAL AXIS1
GLOBAL AXIS2

GLOBAL REAL ARR(16) !Definition of a 16 point position array
ARR(0)=100;ARR(1)=150;ARR(2)= 200;ARR(3)=250;
ARR(4)=300;ARR(5)=350;ARR(6)=400;ARR(7)= 450;
ARR(8)=500;ARR(9)=550;ARR(10)= 600;ARR(11)=650;
ARR(12)=700;ARR(13)=750;ARR(14)=800;ARR(15)=1000;

AXIS0=0
AXIS1=1
AXIS2=2

assignpeg AXIS0, 0b100, 0b0

assignpouts AXIS0, 0, 0b000 !Assign bit code 000 from SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs \(Servo Processor 0\) to AXIS0,
                                !at Pin0
assignpouts AXIS1, 1, 0b000
assignpouts AXIS2, 2, 0b001

ST:
    peg_r AXIS0,0.5,0x4444,0,15,ARR,STAT !Activate random PEG for axis 0
    peg_r AXIS1,0.5,0x4444,0,15,ARR,STAT
    peg_r AXIS2,0.5,0x4444,0,15,ARR,STAT

    !Wait till command executes and configuration is set, in order to
    !ensure proper PEG engine's execution prior to start of movement
    TILL AST(AXIS0).#PEGREADY
    TILL AST(AXIS1).#PEGREADY
    TILL AST(AXIS2).#PEGREADY

    ptp/e AXIS0,5000
    stoppeg AXIS0 ! Prevent PEG pulses' firing on the 'way back'
    ptp/e AXIS0,0
goto ST
stop

```

### Related ACSPL+ Commands

[PEG\\_I](#), [ASSIGNPEG](#), [ASSIGNPOUTS](#), [STARTPEG](#), [STOPPEG](#)

### Related ACSPL+ Variables

## AST

### COM Library Methods

None

### C Library Functions

acsc\_PegRandomNT, acsc\_WaitPegReady

## 2.3.6 STARTPEG

### Description

The **STARTPEG** command initiates the PEG process on the specified axis. The command is used in both the Incremental and Random PEG modes.

### Syntax

**STARTPEG** *axis*

### Arguments

**axis**

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

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

### Related ACSPL+ Commands

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

### Related ACSPL+ Variables

## AST

### COM Library Methods

None

### C Library Functions

acsc\_StartPegNT

## 2.3.7 STOPPEG

### Description

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

### Syntax

**STOPPEG** *axis*

### Arguments

**axis**

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

## 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.

## Example

```
PEG_I/S 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

## 2.4 Miscellaneous Commands

The Miscellaneous commands are:

Command	Description
<a href="#">AXISDEF</a>	Establishes an axis alias.
<a href="#">DC</a>	Activates data collection.
<a href="#">SPINJECT</a>	Returns the system back to the operational state if one or more slaves underwent a reset or power cycle.
<a href="#">READ</a>	Reads an array from a file in the flash memory.
<a href="#">SPDC</a>	Activates data collection from a Servo Processor variable.
<a href="#">SPINJECT</a>	Initiates the transfer of MPU real-time data to the Servo Processor.
<a href="#">STOPINJECT</a>	Stops the transfer of MPU real-time data to the Servo Processor.
<a href="#">SPRT</a>	Starts a real-time data transfer process from the MPU to a given Servo Processor.
<a href="#">SPRTSTOP</a>	Stops an active real-time data transfer process on the given SP (for cyclic command only).
<a href="#">STOPDC</a>	Terminates data collection.
<a href="#">WRITE</a>	Writes an array to a file in the flash memory.

## 2.4.1 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>	Any string with the following restrictions: <ul style="list-style-type: none"><li>● Only one name can be defined for one axis, that is, different names cannot be used for the same axis.</li><li>● The names must be unique, i.e., two axes cannot be defined with the same name.</li><li>● An axis name must not be the same of any other variable name, label, keyword, etc.</li></ul> A compilation error occurs if one of the above restrictions is not satisfied.
<i>axis</i>	The axis number, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.

### 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

### Examples

#### 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

### Table of index formats.

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

## 2.4.2 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

<b>global array</b>	The name of a global array that stores samples
<b>number of points</b>	Define the number of samples
<b>time-interval</b>	Define the time interval between each sample
<b>variable_1</b>	Define variable/s to be sampled. The number of rows defined in <b>array_name</b> must match the number of variables to be sampled

### Switches

/switch can be one of the following:

/s	Triggers DC with the next motion command. <b>DC</b> synchronized with motion is called Axis Data Collection
/w	Create the synchronous data collection, but do not start until <b>G0</b> . Command option <b>/w</b> can only be used with the <b>/s</b> .
/c	Cyclic data collection

## Comments

1. **DC** can include up to 8 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

## Examples

### 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 SAMPLE_ARRAY. The time 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 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 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
```

## 2.4.3 STOPDC

### Description

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

### Syntax

**STOPDC[/switch]**

### Switch

/switch can be:

/s

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
```

### 2.4.4 READ

#### Description

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

#### Syntax

**READ** *array,filename*

#### Arguments

<b>array</b>	User defined array to which the data will be imported
<b>filename</b>	Non-volatile memory file name

#### Comments

1. The **filename** must not include the file name extension.
2. The **user-array** name must be declared in the buffer where the command is executed.
3. If **READ** is executed from the **Communication Terminal** as a command, **array** must specify the name of a global array.

#### Related ACSPL+ Commands

[WRITE](#)

#### COM Library Methods

Transaction

#### C Library Functions

acsc\_Transaction

### Example

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

## 2.4.5 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 **SET FPOS** command. An offset is added at the MPU level only. The formula (that you can find in our manuals) is:

$$\text{FPOS} = \text{FP} * \text{EFAC} + \text{EOFFS}$$

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

<b>Array</b>	Array name, up to <b>XARRSIZE</b> variable value. By default, <b>Array</b> is assumed to be an integer array, if the <b>/r</b> switch is added, it defines the array as real.
<b>number_of_samples</b>	The number of samples to collect, the maximum value depends on the size of the array.
<b>sampling_period</b>	The time, in millisconds, that each sample is taken.
<b>SP_number</b>	The number of the Servo Processor to be sampled
<b>SP_Address1</b>	The address of the Servo Processor variable in the Servo Processor to sample,
<b>SP_Address2</b>	As an option, you can add another address of an other Servo Processor variable in the Servo Processor to sample,



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 2-27. 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

## Example

```

! ----- 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
```

## 2.4.6 WRITE

### Description

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

### Syntax

**WRITE** *user-array, filename*

### Arguments

<b>user-array</b>	User defined array from which the data will be imported
<b>filename</b>	Non-volatile memory file name

### Comments

1. The nonvolatile memory **filename** must not include an extension.
2. The **user-array** 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** has to specify the name of a global array.
4. If **filename** does not exist, it is created. If the file already exists, it is over written.

### Related ACSPL+ Commands

**READ**

### COM Library Methods

Transaction

### C Library Functions

acsc\_Transaction

## 2.4.7 SPINJECT

### Description

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

### Syntax

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

### Arguments

<b>Array</b>	Data source: 1 or 2 dimensional ACSPL+ array (real or integer).
--------------	---

<b>Nsamples</b>	Number of samples from the source <b>Array</b> 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.
<b>Node</b>	The number of the EtherCAT node as in Servo Processor data collection.
<b>Addr1</b>	Address inside the <b>Servo Processor</b> , it must correspond to a floating or integer variable in <b>Servo Processor</b> program.
<b>Addr2</b>	Address inside the <b>Servo Processor</b> , it must correspond to a floating or integer variable in <b>Servo Processor</b> program. Used only if <b>Array</b> is 2 dimensional.

## Switches

/switch can be one of the following:

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

## Examples

```
INT MYDCOM(1)
SPINJECT/C MYDCOM,1,0,getspe(0,"axes[0].direct_command")
!MYDCOM is RT control of DCOM inside Servo Processor0
real K(2)(30000)
!Fill K array with SIN and COS values
SPINJECT/r K,30000,0,getspe(0,"dummy_double[1]"),getspe(0,"dummy_double
[2]")
```

## 2.4.8 STOPINJECT

### Description

**STOPINJECT** stops active injection process on the given Servo Processor.

### Syntax

**STOPINJECT** *Servo\_Processor*

### Arguments

<b>Servo_Processor</b>	Identifies the Servo Processor upon which the injection process is operating.
------------------------	---

### Example

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

## 2.4.9 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

### Command Options

/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"> <li>● up to 20 values for CTIME <math>\geq 0.50</math></li> <li>● up to 12 values for <math>0.20 \leq CTIME &lt; 0.50</math></li> </ul>
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 <b>Value_Array</b> 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**, **SLIKI**) 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.

### Examples

#### 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).

% Adress of the Servo Processor SLVKP parameter used during motion
Address(0) = getspa(0,"axes[0].params[0].SLVKP")
% Adress of the Servo Processor SLVKP parameter used in idle state
Address(1) = GETSPA(0,"axes[0].params[1].SLVKP")
% Adress 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
SLVPK_value = 50
! Desired SLVKI value
SLVKI_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
STOP

```

### 2.4.10 SPRTSTOP

#### Description

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

#### Syntax

**SPRTSTOP** SP

#### Arguments

<b>SP</b>	Number of the EtherCAT node as in SP data collection.
-----------	---

## 2.5 Motion Commands

The Motion commands are:

Command	Description
ARC1	Adds an arc segment to <a href="#">MSEG...ENDS</a> motion.
ARC1	Adds an arc segment to <a href="#">XSEG...ENDS</a> motion.
ARC2	Adds an arc segment to <a href="#">MSEG...ENDS</a> motion.
ARC2	Adds an arc segment to <a href="#">XSEG...ENDS</a> motion
JOG	Creates a jog motion.
LINE	Adds a linear segment to <a href="#">MSEG...ENDS</a> motion.
LINE	Adds a linear segment to <a href="#">XSEG...ENDS</a> motion.
MASTER	Defines a formula for calculating <a href="#">MPOS</a> .
MPOINT	Adds a set of points to <a href="#">MPTP...ENDS</a> , <a href="#">PATH...ENDS</a> or <a href="#">PVSPLINE...ENDS</a> 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.

Command	Description
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.
STOPPER	Adds a segment separator to MSEG...ENDS motion.
TRACK	Creates tracking motion.
XSEG...ENDS	Creates an extended segment motion.



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

## 2.5.1 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.

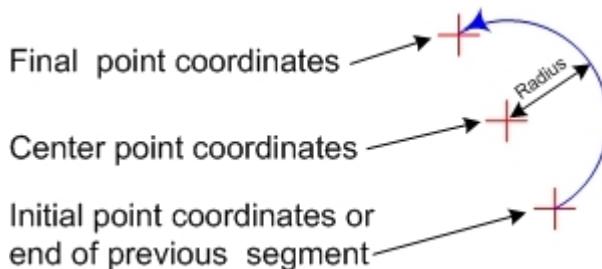


Figure 2-5. ARC1 Coordinate Specification

### Syntax

**ARC1 axis\_list, center-point, final-point, direction[,user-specified velocity]**

### Arguments

<b>axis_list</b>	List of axes involved, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
------------------	--

	The <b>ARC1</b> axis_list can involve two or more axes, see <a href="#">PROJECTION</a> .
	 A minimum of two axes must be specified.
<b>center-point</b>	Center point coordinates.
<b>final point</b>	Last point.
<b>direction</b>	Use + for motion in the direction of increasing encoder counts, or - for motion in the direction decreasing encoder counts.
<b>user-specified velocity</b>	If <b>MSEG</b> command option /v is used, the user-specified velocity must be the last parameter in the <b>ARC1</b> 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](#).

## 2.5.2 ARC1

### Description

This format of **ARC1** is used for extended segment motion and in this form must be initialized with **XSEG...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.

### 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]]]
```

## Arguments

	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 <b>XSEG</b> command.
<b>axis_list</b>	 A minimum of two axes must be specified.
<b>center_point_axis1</b>	Center point position for the first axis
<b>center_point_axis2</b>	Center point position for the second axis
<b>destination_point_axis1</b>	Destination position of the first axis
<b>destination_point_axis2</b>	Destination position of the second axis
<b>destination_point_axis3</b> ... <b>destination_point_axis6</b>	Mandatory only if <b>axis_list</b> 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 <b>axis_list</b> .
<b>direction</b>	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.
<b>velocity</b>	[Optional, only used with /v switch] Defines required velocity for the current and for all subsequent segments. See <a href="#">Switches</a> explanation for details.
<b>end_velocity</b>	[Optional, only used with /f switch] Defines required velocity at the end of the current segment. See <a href="#">Switches</a> explanation for details.
<b>time</b>	[Mandatory with /t switch]. Defines segment processing time.
<b>values</b>	[Optional, only used with /o switch] Defines the values to be written to variables array at the beginning of the current segment execution. <b>values</b> is a one-dimensional user defined array of integer or real type with maximum size of 10 elements.
<b>variables</b>	[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. <b>variables</b> is a one-dimensional user defined array of the same type and size as the <b>values</b> array.
<b>index</b>	[Optional, only used with /o switch] Defines the first element (starting from zero) of the <b>variables</b> array, to which values data will be written. If argument is omitted, <b>values</b> data is written to the <b>variables</b> array starting from the first element (index 0).

<b>masks</b>	<p>[Optional, only used if <b>values</b> and <b>variables</b> are integer]</p> <p>Defines the masks that are applied to <b>values</b> before the values are written to <b>variables</b> array at the beginning of the current segment execution. <b>masks</b> is a one-dimensional user-defined array of integer type and the same size as the <b>values</b> array. The masks are only applied for integer values:</p> <p><b>variables(n) = (variables(n) AND (NOT mask(n))) OR (values(n) AND mask(n))</b></p> <p>If argument is omitted, all values bits are written to <b>variables</b>. If <b>values</b> is a real array, the masks argument should be omitted.</p>
--------------	---

## Switches

The following optional **/switches** may be used singularly or in combination with the **ARC1** command:

<b>/v</b>	<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>
<b>/f</b>	<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>
<b>/o</b>	<p>Synchronize user variables with segment execution. The switch requires additional two or three parameters that specify <b>values</b>, user <b>variable</b> and <b>mask</b>. See details in <a href="#">Arguments</a> for explanation.</p>

### 2.5.3 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.

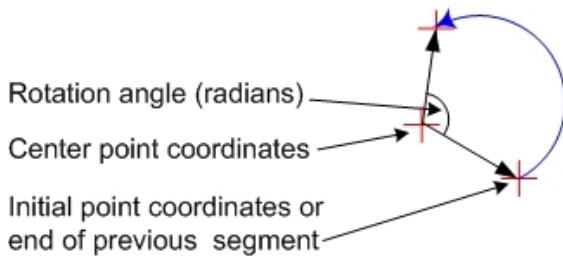


Figure 2-6. ARC2 Center Point and Rotation Angle Specification

## Syntax

**ARC2** *axis\_list, center-point, rotation-angle and direction [user-specified velocity]*

## Arguments

<i>axis_list</i>	List of axes involved, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.  The <b>ARC2 axis_list</b> can involve two or more axes, see <a href="#">PROJECTION</a> .
<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 <b>MSEG</b> command option <b>/v</b> is used, the user-specified velocity must be the last parameter in the <b>ARC2</b> 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.

## 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](#).

## 2.5.4 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]]]
```

### Arguments

<i>axis_list</i>	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 <b>XSEG</b> command.
	 A minimum of two axes must be specified.
<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>	Mandatory only if <b>axis_list</b> 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 <b>axis_list</b> .
<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 <b>/v</b> switch] Defines required velocity for the current and for all subsequent segments. See <a href="#">Switches</a> explanation for details.
<i>end_velocity</i>	[Optional, only used with <b>/f</b> switch] Defines required velocity at the end of the current segment. See <a href="#">Switches</a> explanation for details.
<i>time</i>	[Mandatory with <b>/t</b> switch]. Defines segment processing time.

<b>values</b>	[Optional, only used with <b>/o</b> switch]  Defines the values to be written to variables array at the beginning of the current segment execution. <b>values</b> is a one-dimensional user defined array of integer or real type with maximum size of 10 elements.
<b>variables</b>	[Optional, only used with <b>/o</b> switch]  Defines the user-defined array, which will be written with values data at the beginning of the current segment execution. <b>variables</b> is a one-dimensional user defined array of the same type and size as the <b>values</b> array.
<b>index</b>	[Optional, only used with <b>/o</b> switch]  Defines the first element (starting from zero) of the <b>variables</b> array, to which values data will be written. If argument is omitted, <b>values</b> data is written to the <b>variables</b> array starting from the first element (index 0).
<b>masks</b>	[Optional, only used if <b>values</b> and <b>variables</b> are integer]  Defines the masks that are applied to <b>values</b> before the values are written to <b>variables</b> array at the beginning of the current segment execution. <b>masks</b> is a one-dimensional user-defined array of integer type and the same size as the <b>values</b> array. The masks are only applied for integer values:  $\text{variables}(n) = (\text{variables}(n) \text{ AND } (\text{NOT mask}(n))) \text{ OR } (\text{values}(n) \text{ AND mask}(n))$  If argument is omitted, all values bits are written to <b>variables</b> . If <b>values</b> is a real array, the masks argument should be omitted.

## Switches

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

<b>/v</b>	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.
<b>/f</b>	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.

/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.

## 2.5.5 JOG

### Description

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

JOG motion terminates by:

- [HALT](#) , [KILL](#), [BREAK](#), [DISABLE](#)
- 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

<b>axis_list</b>	List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>user-specified-velocity</b>	Optional parameter used if the <b>/v</b> command option is specified
<b>user-specified-direction</b>	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:

<b>/v</b>	Use the specified velocity instead of the default velocity ( <a href="#">VEL</a> ).
<b>/w</b>	Create the motion, but do not start until <a href="#">GO</a> .

### Related ACSPL+ Commands

[GO](#), [HALT](#) , [KILL](#), [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.
------------------	---

## 2.5.6 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

	List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>axis_list</b>	 A minimum of two axes must be specified.
<b>final position1</b>	First coordinate
<b>final position2</b>	Second coordinate
<b>user specified velocity</b>	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](#).

## 2.5.7 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]]]
```

### Arguments

<i>axis_list</i>	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 <b>XSEG</b> command.   A minimum of two axes must be specified.
<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>	Mandatory if <b>axis_list</b> contains more than 2 axes. Destination position of the rest of axes. Number of destination positions must correspond to the number of axes in <b>axis_list</b> .
<i>velocity</i>	[Optional, only used with <b>/v</b> switch] Defines required velocity for the current and for all subsequent segments. See <b>Switches</b> explanation for details.
<i>end_velocity</i>	[Optional, only used with <b>/f</b> switch] Defines required velocity at the end of the current segment. See <b>Switches</b> explanation for details.
<i>time</i>	[Mandatory with <b>/t</b> switch]. Defines segment processing time.
<i>values</i>	[Optional, only used with <b>/o</b> switch] Defines the values to be written to variables array at the beginning of the current segment execution. <b>values</b> is a one-dimensional user defined array of integer or real type with maximum size of 10 elements .

<b>variables</b>	[Optional, only used with <b>/o</b> switch]  Defines the user-defined array, which will be written with values data at the beginning of the current segment execution. <b>variables</b> is a one-dimensional user defined array of the same type and size as the <b>values</b> array.
<b>index</b>	[Optional, only used with <b>/o</b> switch]  Defines the first element (starting from zero) of the <b>variables</b> array, to which values data will be written. If argument is omitted, <b>values</b> data is written to the <b>variables</b> array starting from the first element (index 0).
<b>masks</b>	[Optional, only used if <b>values</b> and <b>variables</b> are integer]  Defines the masks that are applied to <b>values</b> before the values are written to <b>variables</b> array at the beginning of the current segment execution. <b>masks</b> is a one-dimensional user-defined array of integer type and the same size as the <b>values</b> array. The masks are only applied for integer values:  $\text{variables}(n) = (\text{variables}(n) \text{ AND } (\text{NOT mask}(n))) \text{ OR } (\text{values}(n) \text{ AND mask}(n))$  If argument is omitted, all values bits are written to <b>variables</b> . If <b>values</b> is a real array, the masks argument should be omitted.

## Switches

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

<b>/v</b>	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.
<b>/f</b>	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.
<b>/o</b>	Synchronize user variables with segment execution. The switch requires additional two or three parameters that specify <b>values</b> , user <b>variable</b> and <b>mask</b> . See details in <a href="#">Arguments</a> for explanation.

### 2.5.8 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

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

### Related ACSPL+ Commands

[SLAVE](#), [GO](#), [HALT](#), [KILL](#), [ENABLE/ENABLEALL](#), [DISABLE](#), [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)
          !Initiates the 0 axis motion
SLAVE 0
```

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.

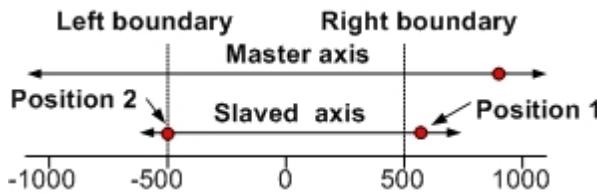


Figure 2-7. SLAVE /pt Illustration

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.

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.

## 2.5.9 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 <b>MPTP</b> , <b>PATH</b> or <b>PVSPLINE</b> .
<i>array (number of rows,number of points)</i>	<p>The <b>MPOINT</b> array must be defined as <b>Real</b>. 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: <b>real array (3)(5)</b>.</p> <p>Each row defines the values for the points for each axes, and depending on command preceding <b>MPOINT</b>, 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 <a href="#">Array Structures</a> 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 command options. These structures are described below.

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

ARRAY (3)(5)					
	Point 1	Point 2	Point 3	Point 4	Point 5
<b>0 Axis</b>	1000	2000	3000	4000	5000
<b>1 Axis</b>	5000	4000	3000	2000	1000
<b>5 Axis</b>	0	2000	4000	6000	8000

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

1. 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
<b>0 Axis</b>	1000	2000	3000	4000	5000
<b>1 Axis</b>	5000	4000	3000	2000	1000
<b>5 Axis</b>	0	2000	4000	6000	8000
<b>Velocity</b>	6000	7000	5000	4000	4000

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

1. 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
<b>0 Axis</b>	1000	2000	3000	4000	5000
<b>1 Axis</b>	5000	4000	3000	2000	1000
<b>5 Axis</b>	0	2000	4000	6000	8000
<b>Time</b>	250	250	250	250	250

Time is specified in milliseconds.

- If **MPOINT** follows **PVSPLINE** without command option **/t**, 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
<b>0 Axis</b>	1000	2000	3000	4000	5000
<b>1 Axis</b>	5000	4000	3000	2000	1000
<b>5 Axis</b>	0	2000	4000	6000	8000
<b>0 Axis VEL</b>	5000	5000	5000	5000	5000
<b>1 Axis VEL</b>	2500	2500	2500	2500	2500
<b>5 Axis VEL</b>	3000	3000	3000	3000	3000

- 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
<b>0 Axis</b>	1000	2000	3000	4000	5000
<b>1 Axis</b>	5000	4000	3000	2000	1000
<b>5 Axis</b>	0	2000	4000	6000	8000

ARRAY (7)(5)					
	Point 1	Point 2	Point 3	Point 4	Point 5
<b>0 Axis VEL</b>	5000	5000	5000	5000	5000
<b>1 Axis VEL</b>	2500	2500	2500	2500	2500
<b>5 Axis VEL</b>	3000	3000	3000	3000	3000
<b>Time</b>	500	500	500	500	500

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

1. 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

```

## 2.5.10 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**, or **BREAK**
- Any fault activation that disables the drive or kills the motion
- **DISABLE** by the user

## Syntax

**MPTP[/switch] axis\_list[,dwell]**

## Arguments

<b>axis_list</b>	List of axes, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>dwell</b>	Dwell is an optional argument, expressed in milliseconds.

## Switches

**/switch** can be:

<b>/w</b>	Create the motion, but do not start until the <b>GO</b> command.
<b>/v</b>	Use the specified velocity instead of the default velocity ( <b>VEL</b> ).
<b>/c</b>	Use the point sequence as a cyclic array
<b>/r</b>	Coordinates of each point are relative to the previous point's coordinates.

## 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**, or **BREAK** to stop cyclic motion.

## Related ACSPL+ Commands

[GO](#), [HALT](#), [KILL](#), [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, **dwell** 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.

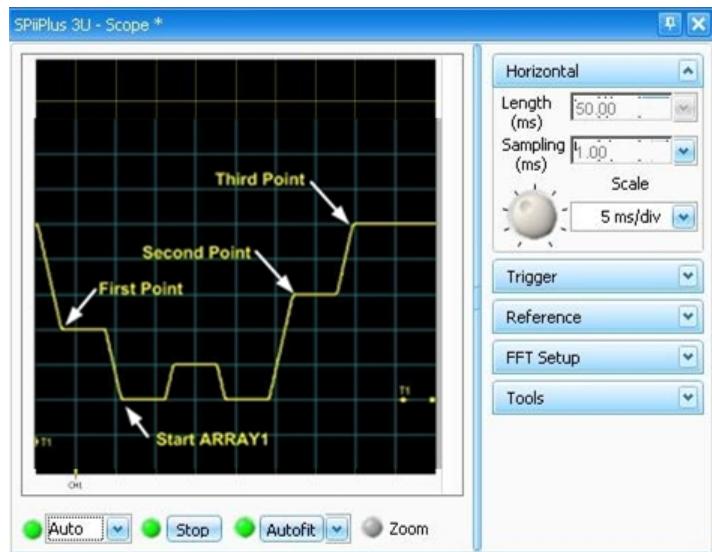


Figure 2-8. Single-Axis Motion Using MPTP

#### 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**.

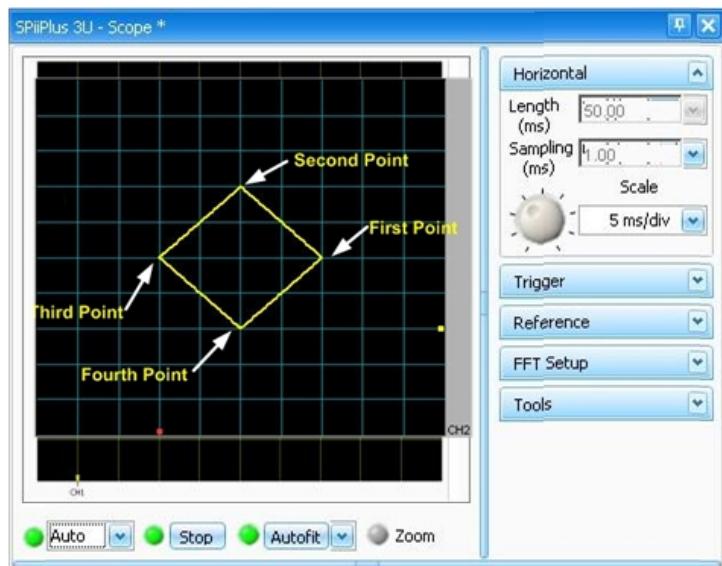


Figure 2-9. Two-Axis Group Motion Using MPTP/v

### 2.5.11 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**, or **BREAK**
- Any fault activation that disables the drive or kills the motion

- [DISABLE](#) 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:

<b>/w</b>	Create the motion, but do not start until <a href="#">GO</a> .
<b>/v</b>	Use the specified velocity instead of the default velocity ( <a href="#">VEL</a> ).
<b>/c</b>	Use the <b>MSEG</b> 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](#), 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](#), [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)
STOP
                                !Ends program

```

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

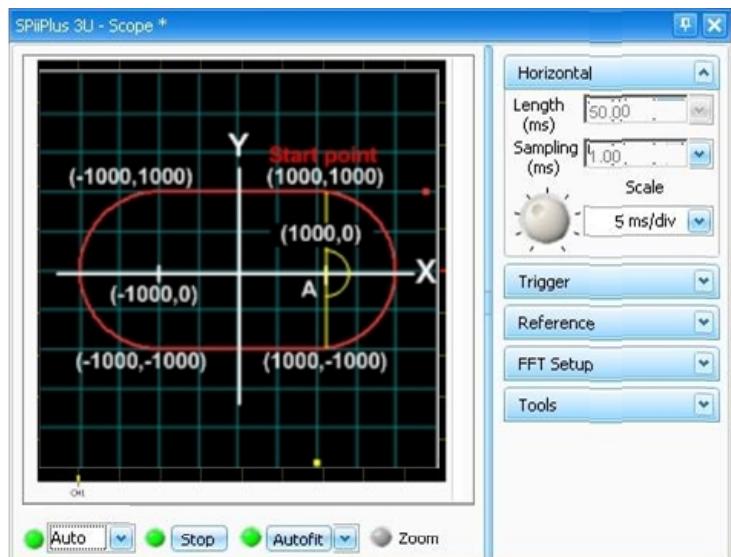


Figure 2-10. Results of Example MSEG

### 2.5.12 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**, or **DISABLE**

#### Syntax

## **PATH[/command options]axis\_list [,time-interval]**

### Arguments

<b>axis_list</b>	Axes list, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>time interval</b>	Optional time interval specification in milliseconds. <b>PATH</b> specified without \t must have the time interval specification as the last argument. The argument defines the time interval between all motion points.

### Command Options

<b>/w</b>	Create the motion, but do not start until the <b>GO</b> command.
<b>/t</b>	Enables a user-defined time interval in a point array
<b>/c</b>	Use the point sequence as a cyclic array. After arriving at the last point, continue from the first point.
<b>/r</b>	Coordinates of each point are relative to the previous point's coordinates.

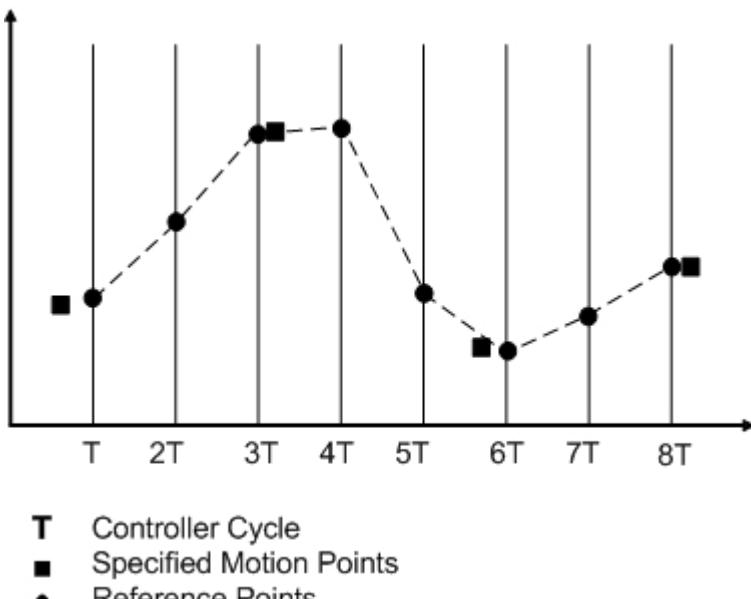


Figure 2-11. PATH...ENDS Diagram

### 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.

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

## 2.5.14 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

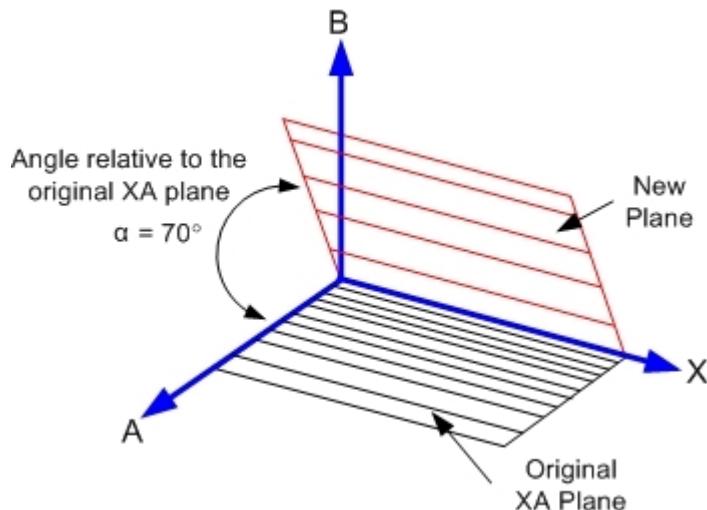
### Example

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 2-28. 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](#)).

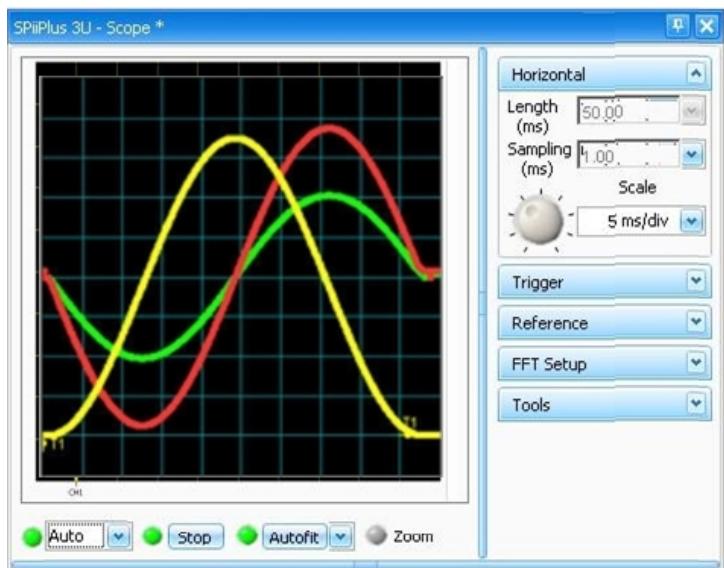


Figure 2-13. FPOS - PROJECTION Example

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

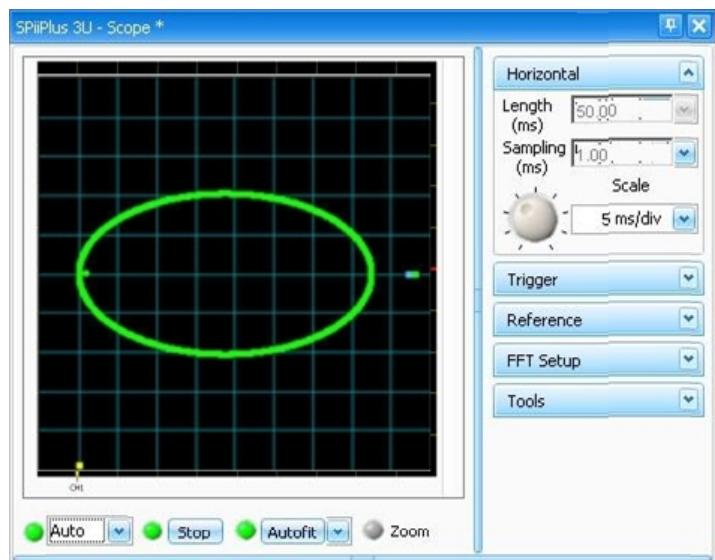


Figure 2-14. PROJECTION Example - Final Result

```

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

## 2.5.15 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**, or **BREAK**.

### Syntax

**PTP[command options]axis\_list,destination-point[,value for v,value for f]**

### Arguments

<b>axis_list</b>	Single axis or axis group, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>destination point</b>	Final destination.
<b>value for v</b>	Optional argument for user-defined velocity.
<b>value for f</b>	Optional argument for user-defined velocity at a destination point

### Command Options

<b>/e</b>	Wait for motion termination before executing next command.
<b>/f</b>	Specify non-zero velocity at each destination point (or points) in a series of PTP motions
<b>/m</b>	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 <b>VEL, ACC, DEC, JERK</b> values.
<b>/r</b>	The destination point is relative to the start point.

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

## 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
```

## 2.5.16 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**, or **DISABLE**

## Syntax

**PVSPLINE**[*command options*]*axis\_list*[,*time-interval*]

## 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.

## Command Options

/r	The point coordinates are relative to the previous point
/w	Create the motion, but wait to start for <b>G0</b>
/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.

## 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
```

```

INT NUMBER_of_POINTS
NUMBER_of_POINTS=3
PVSPLINE/r (0,1),10
POINT (0,1),200,100,1000,2000
MPOINT (0,1), ARRAY_NAME, NUMBER_of_POINTS
ENDS (0,1)
STOP

```

!that are part of the motion.  
!Defines NUMBER\_of\_POINTS as an integer variable.  
!Assigns 3 to the NUMBER\_of\_POINTS variable.  
!Initiates PVSPLINE motion for axis 0 and 1 with  
!relative destination points and a time interval of  
!10 milliseconds between each point.  
!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.  
!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 MPOINT for 0 and 1 axes.  
!Ends program.

Figure 2-15 illustrates a typical **PVSPLINE** motion.

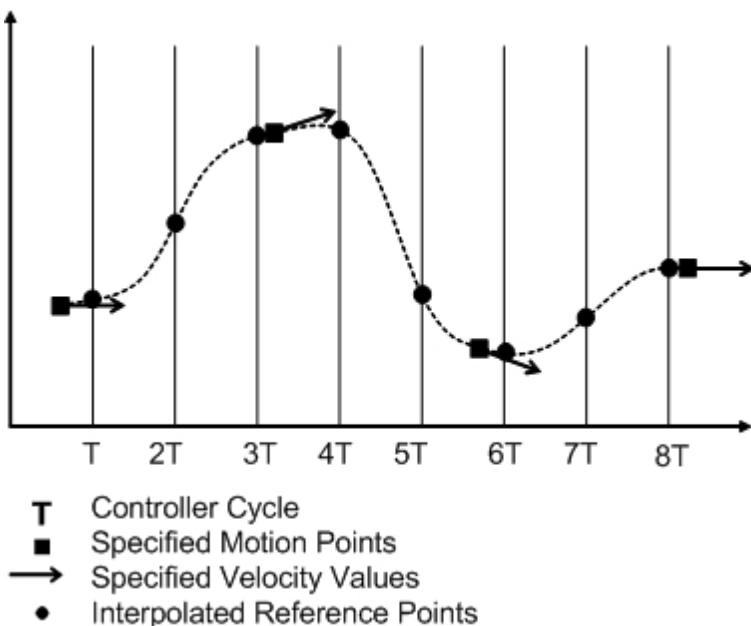


Figure 2-15. PVSPLINE Motion Diagram

### 2.5.17 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](#), [BREAK](#)
- [DISABLE](#) to the **SLAVE** axis.
- Setting the logical dependence between master and slave axes to zero. For example **MASTER MPOS(0)=0**

### Syntax

**SLAVE[/command options] axis[,lower boundary, upper boundary]**

### Arguments

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

### Command Options

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

### Comments

- When **/p** is used, the controller first initiates **velocity lock**. Only after 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](#), [DISABLE](#)

### 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.

### 2.5.18 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

<b>axis_list</b>	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.
```

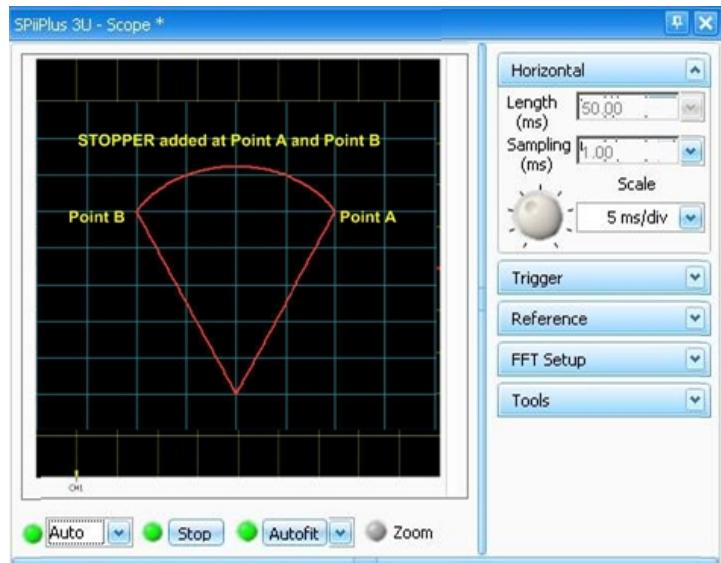


Figure 2-16. Use of STOPPER

### 2.5.19 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**, or **DISABLE**.

#### Syntax

**TRACK [/switch] axis**

#### Arguments

**axis**

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

#### Switch

**/switch** can be:

**/w**

Create the motion, but do not start until **GO**.

#### 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.
```

## 2.5.20 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

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]]
```

Segment commands (ARC1, ARC2, LINE)

#### ENDS

## Arguments

	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.
<i>axis_list</i>	 <p>A minimum of two axes must be specified.</p>
<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... initial_position_axis6</i>	Mandatory only if <b>axis_list</b> contains more than 2 axes. Number of initial positions must correspond to the number of axes in <b>axis_list</b> .
<i>velocity</i>	[Optional, only used with /v switch] Defines required velocity instead of default velocity ( <b>VEL</b> ).
<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 <a href="#">Switches</a> explanation for details.
<i>deviation</i>	[Optional, only used with /g switch] Defines maximal allowed trajectory deviation from the corner point. See <a href="#">Switches</a> explanation for details.
<i>radius</i>	[Optional, only used with /u switch] Defines maximal allowed rounding radius of the additional segment. See <a href="#">Switches</a> 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 <b>AST.#NEWSEG.M</b> 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.

## 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	
<b>/w</b>	Do not start until the <b>GO</b> command.  If the switch is not specified, the motion starts immediately after the first motion segment is defined.
<b>/v</b>	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.
<b>/m</b>	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 ( <b>VEL</b> values) of the involved axes.

### Velocity look-ahead

<b>/f</b>	<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>
<b>/j</b>	<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 <b>j</b> is not specified while switch <b>a</b> is specified, zero value of corner velocity is assumed.</p> <p>If switches <b>j</b>, <b>a</b>, <b>d</b>, and <b>y</b> are not specified, the controller provides automatic calculation of the corner processing.</p>
<b>/a</b>	<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 <b>a</b> is not specified while switch <b>j</b> is specified, the controller accepts default value of 0.01 radians that is about 0.57 degrees.</p> <p>If switches <b>j</b>, <b>a</b>, <b>d</b>, and <b>y</b> are not specified, the controller provides automatic calculation of the corner processing.</p>
<b>/d</b>	<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 <b>j</b> and/or <b>a</b>.</p> <p>If switches <b>j</b>, <b>a</b>, <b>d</b>, and <b>y</b> are not specified, the controller provides automatic calculation of the corner processing.</p>
<b>/y</b>	<p>If the switch is specified the controller provides automatic calculations as described in Enhanced automatic corner and curvature discontinuity points processing (switch <b>/y</b>).</p>

### Geometry look-ahead

<b>/g</b>	<p>Use a corner rounding option with the specified permitted deviation. The switch</p>
-----------	--

	requires additional parameter that specifies maximal allowed deviation of motion trajectory from the corner point. The switch cannot be specified together with switches <b>/u</b> and <b>/h</b>
<b>/u</b>	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 <b>/g</b> and <b>/h</b>
<b>/h</b>	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 segments in a pair is arc, the smoothing is not applied for this corner. The switch cannot be specified together with switches <b>/g</b> and <b>/u</b> .



**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).

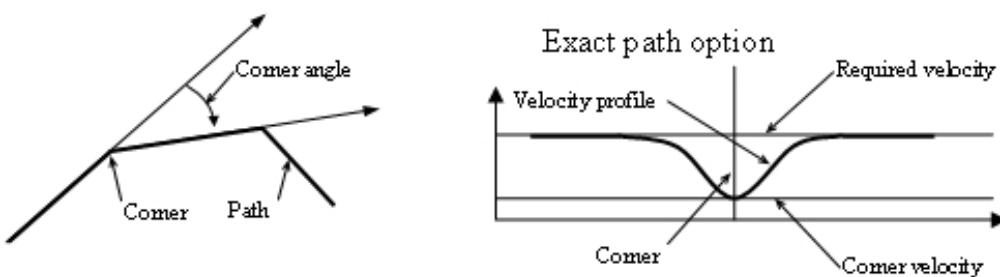


Figure 2-17. Corner Processing - Exact Path Option

**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.

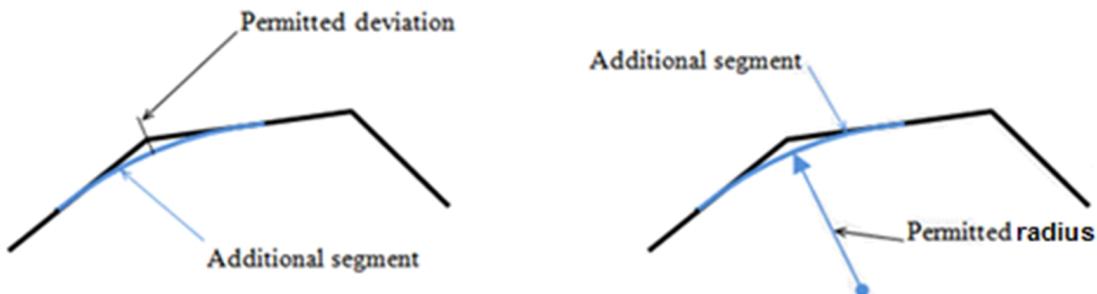


Figure 2-18. Corner Processing - Permitted Deviation, Permitted Radius and Corner Smoothing Options

**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 600 bytes, so if necessary to allocate the buffer for 200 segments, it should be at least  $600 * 200 = 120,000$  bytes. The following declaration defines a 120,000 bytes buffer:

```
real buf(15000)
```



See [XARRSIZE](#) for details 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 point
(0,1),0,0,100,50       !(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 point
(1,2),1000,1000,      !(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

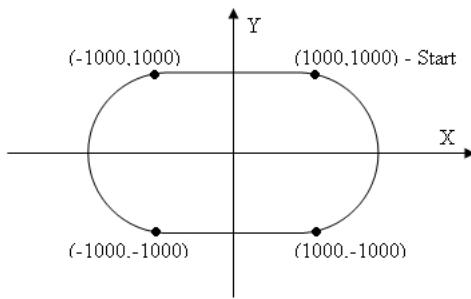
LINE (0,1),-1000,-1000   !Add line segment with final point ( 1000, 1000)

ARC2 (0,1),-1000,0,-3.141529 !Add arc segment with center ( 1000,0) and
rotation                 !angle of -π

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 at this moment. Actual motion starts once the previous motion ends and one or more segments are added. The four segment commands specify the following path:



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 Program Flow Commands

Command	Description
<b>Assignment Command</b>	Assigns values
<b>BLOCK...END</b>	Executes a group of commands in one MPU cycle
<b>CALL</b>	Calls subroutine.
<b>GOTO</b>	Transfers program execution to another point in the program.
<b>IF, ELSEIF, ELSE...END</b>	<b>IF</b> command structure.
<b>INPUT</b>	Suspends program execution pending user input
<b>LOOP...END</b>	Loop command structure.
<b>ON...RET</b>	Defines an autoroutine
<b>TILL</b>	Delays program execution until a specified expression produces a non-zero (true) result.

Command	Description
<code>WAIT</code>	Delays program execution for a specified number of milliseconds.
<code>WHILE...END</code>	While command structure.

## 2.6.1 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"><li>● Name of ACSPL+ standard or user-defined variable</li><li>● An element of an ACSPL+ or user array</li><li>● One bit of integer variable or integer array element (applicable only to those variables having specifically named bits, for example, <code>IMASK</code>)</li></ul>
value	Can be of integer or real type. The <b>value</b> argument can either be: <ul style="list-style-type: none"><li>● A specific value</li><li>● An expression that during runtime calculates to a value</li></ul>

### 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, `ERR1`) 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.
VELO = 1000	!Assign 1000 to axis 0 default velocity - !postfix indexing.
Var1 = FPOS(0)	!Assign value of ACSPL+ variable to user variable.
Var2(0)(5) = 200	!Assign to element of user array.
OUT0.5 = 1	!Assign to digital output 5

## 2.6.2 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

```
END  
ZZ=YY-XX
```

!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 the BLOCK command  
!The result in this example is ZZ=0.

## 2.6.3 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

<b>label</b>	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:  
WHILE (ABS(PE(0)>20)) !Subroutine label name.  
                      !Subroutine command lines.
```

```
DISP "PE is :", PE(0)
END                                !End WHILE loop.
RET                                !Terminate subroutine and return to main program.
```

## 2.6.4 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

<b>label</b>	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
```

## 2.6.5 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 2-29](#) describes the **IF** control structure syntax using these building blocks.

**Table 2-29. IF Control Structures**

### Syntax

Syntax	IF Structure
<b>IF expression command listEND</b>	IF-END
<b>IF expression command listELSE command listEND</b>	IF-ELSE-END
<b>IF expression command listELSEIF sequenceEND</b>	IF-ELSEIF-END
<b>IF expression command listELSEIF sequenceELSE command listEND</b>	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
  
```

STOP

!V0 is equal to or more than 256)  
!Ends program

## 2.6.6 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 argument the array 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)

```

## 2.6.7 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 <b>ARR</b> . !J is increased by 1 for each repetition.
J=J+1	
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.

## 2.6.8 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**.

### 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
DISABLE 0	!FAULT(0).#DRIVE changes from 0 to 1
DISP "0 Axis Drive Alarm"	!Disable drive X
RET	!Display message
	!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

## 2.6.9 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
-----------------	---------------------------------------

## 2.6.10 WAIT

### Description

**WAIT** delays program execution for the specified number of milliseconds.

### Syntax

**WAIT** *wait\_time*

### Arguments

<b>wait_time</b>	Defines how long to delay the program execution. <b>wait_time</b> 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
---------------	--

## 2.6.11 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

<b>condition</b>	Condition controlling the execution of the WHILE <i>command_list</i> .
<b>command_list</b>	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.
OUTO.0=^OUTO.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
```

## 2.7 Program Management Commands

The Program Management commands are:

Command	Description
<b>DISABLEON</b>	Disables autoroutine activation in a buffer.
<b>ENABLEON</b>	Enables autoroutine activation in a buffer.
<b>PAUSE</b>	Suspends program execution in a buffer.
<b>RESUME</b>	Resumes program execution in a buffer.
<b>START</b>	Activates program execution in a buffer.
<b>STOP</b>	Terminates program execution in a buffer. <b>STOPALL</b> terminates all programs.

### 2.7.1 DISABLEON

#### Description

**DISABLEON** disables autoroutine activation in a buffer. The command has the same functionality as **PFLAGS.#NOAUTO=1**.

#### Syntax

**DISABLEON [(buffer-number)]**

#### Arguments

<b>buffer-number</b>	Optional argument, if no buffer number is mentioned, the command disables autoroutines in all buffers.
----------------------	--

### Related ACSPL+ Commands

[ON...RET](#), [ENABLEON](#)

## 2.7.2 *ENABLEON*

### Description

**ENABLEON** enables autoroutine activation in a buffer. The command has the same functionality as [PFLAGS.#NOAUTO=0](#).

### Syntax

**ENABLEON** [*(buffer-number)*]

### Arguments

<b>buffer-number</b>	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](#), [ENABLEON](#)

## 2.7.3 *PAUSE*

### Description

**PAUSE** suspends program execution in a specific buffer.

### Syntax

**PAUSE** *buffer-number*

### Arguments

<b>buffer-number</b>	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
```

## 2.7.4 *RESUME*

### Description

**RESUME** resumes the program execution in a specific buffer after the execution was paused.

### Syntax

**RESUME** *buffer-number*

### Arguments

**buffer-number**

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
----------	---------------------

## 2.7.5 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** *buffer-number, line-number | label*

### Arguments

<b>buffer-number</b>	A number between 0 and 63 specifying the buffer.
<b>line-number   label</b>	The line number of the program within the buffer where execution is to begin, or a label identifying the line number.

### 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.

### Related ACSPL+ Commands

[STOP, 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.
--------------------	---

## 2.7.6 STOP

### Description

**STOP** terminates program execution in the specified buffer. **STOPALL** terminates program execution in all buffers.

### Syntax

**STOP *buffer-number***

### Arguments

<b><i>buffer-number</i></b>	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.
```

## 2.8 Ethernet/IP ACSPL+ Support Commands

The following ACSPL+ functions set assembly configuration and get existing assembly configuration.

For more information refer to *SPiiPlus EtherNet/IP User Guide*.

### 2.8.1 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
 For the class attribute, this parameter should 0. Otherwise, the specific instance should be specified as follows		
<b>Instance</b>	Class Code	Supported Instances
	0x01	1
	0x02	1
	0x06	1
	0xF4	1,2,3
	0xF5	1
	0xF6	1
	0x04	
	0x64	
	0x65	
 Specifies the attribute as follows:		
<b>Attr</b>	Class Code	Class Supported
		Attributes
	0x01	1
	0x02	1
	0x06	1
	0xF4	1,2,3
	0xF5	1

	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.

## 2.8.2 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.

#### 2.8.2.0.0.1 Syntax

```
int eipgetind1(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.

-1 is returned in case of illegal index parameter.

## 2.8.3 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:
-----------------	---

	<p>Instance Instance Name</p> <ul style="list-style-type: none"> <li>● 0x64 (100) Input Integer Assembly</li> <li>● 0x65 (101) Output Integer Assembly</li> <li>● 0x66 (102) Input Real Assembly</li> <li>● 0x67 (103) Output Real Assembly</li> </ul>
<b>element</b>	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.

### 2.8.4 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

	<p>Assembly instance. The following instances are supported:</p> <p>Instance Instance Name</p> <ul style="list-style-type: none"> <li>● 0x64 (100) Input Integer Assembly</li> <li>● 0x65 (101) Output Integer Assembly</li> <li>● 0x66 (102) Input Real Assembly</li> <li>● 0x67 (103) Output Real Assembly</li> </ul>
<b>element</b>	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.

### 2.8.5 EIPSETASM

#### Description

EIPSETASM sets the assembly configuration.

#### Syntax

eipsetasm(int instance, int element,int tag, int first, int second )

#### Arguments

<b>instance</b>	Assembly instance. The following instances are supported:
-----------------	---

	Instance Instance Name
	<ul style="list-style-type: none"><li>● 0x64 (100) Input Integer Assembly</li><li>● 0x65 (101) Output Integer Assembly</li><li>● 0x66 (102) Input Real Assembly</li><li>● 0x67 (103) Output Real Assembly</li></ul>
<b>element</b>	The index of element in the corresponding assembly. Only indexes 0 to 123 are supported.
<b>tag</b>	The tag number of required ACSPL+ standard or user-defined variable. User-defined variables tags start from index 1000.
<b>first</b>	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.
<b>second</b>	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.

### 3. ACSPL+ Variables

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
- Miscellaneous

The ACSPL+ programming variables are:

**Table 3-1. Alphabetical Listing of All ACSPL+ Variables**

Name	Description	Variable Group
ACC	Acceleration	Motion
AERR	Axis Error	Safety Control
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

Name	Description	Variable Group
CTIME	Controller Cycle Time	System Configuration
DAPOS	Delayed Axis Position	Motion
DCN	Axis DC, Number of Samples	Data Collection
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
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

Name	Description	Variable Group
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
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

Name	Description	Variable Group
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
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	Safe ty Control
FDEF	Default Response Mask	Safety Control
FK	Function Key	Miscellaneous
FMASK	Fault Mask	Safety Control
FPOS	Primary Feedback Position	Axis State

Name	Description	Variable Group
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 <sup>st</sup> 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
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

Name	Description	Variable Group
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
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

Name	Description	Variable Group
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	RMS current of axis	Axis State
ROFFS	Reference Offset	Axis State
RPOS	Reference Position	Axis State
RVEL	Reference Velocity	Motion
RVFIL	Reference Velocity Filter	Axis Configuration - Feedback

Name	Description	Variable Group
S_DCN	System DC, Number of Samples	Data Collection
S_DCP	System DC, Period	Data Collection
S_ERR	System Error	Safety Control
SFAULT	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	<b>Bit mask defining various system settings</b>	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
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

Name	Description	Variable Group
<b>SCGAIN</b>	Cosine gain compensation	Axis Configuration - Feedback
<b>SCPHASE</b>	Cosine phase compensation	Axis Configuration - Feedback
<b>SCSOFFS</b>	Sin-Cos Offset (Sine)	Axis Configuration - Feedback
<b>SETTLE</b>	Settling Time	Axis Configuration
<b>S2LABITS</b>	Used for setting the total number of absolute position bits for an absolute encoder connected to a secondary feedback	Axis Configuration - Feedback
<b>SLABITS</b>	Absolute Position Bits	Axis Configuration - Feedback
<b>SLAFF</b>	Acceleration Feed Forward	Servo-Loop - Compensations
<b>SLBIASA</b>	Current Phase A Bias	Servo-Loop - Current
<b>SLBIASB</b>	Current Phase B Bias	Servo-Loop - Current
<b>SLCFIELD</b>	Induction Motor Excitation	Induction Motor
<b>SLCHALL</b>	Hall Shift	Commutation
<b>SLCNP</b>	Number of Motor Poles	Commutation
<b>SLCOFFS</b>	Commutation Offset	Commutation
<b>SLCORG</b>	Commutation Origin	Commutation
<b>SLCPA</b>	Phase Advance	Commutation
<b>SLCPRD</b>	Commutation Period	Commutation

Name	Description	Variable Group
<b>SLCROUT</b>	Commutation Feedback	Servo-Loop - Miscellaneous
<b>SLCSLIP</b>	Induction Motor Slip Factor	Induction Motor
<b>SLDRAIF</b>	DRA frequency	Servo-Loop - Position
<b>SLDRAIF</b>	Provides an Idle Factor to the SLDRA variable.	Servo-Loop - Position
<b>SLDRX</b>	Maximum DRA correction	Servo-Loop - Position
<b>SLDZMAX</b>	Maximum Dead Zone position	Servo-Loop - Nanomotion
<b>SLDZMIN</b>	Minimum Dead Zone position	Servo-Loop - Nanomotion
<b>SLEBIASA</b>	Defines encoder hardware Sine offset	Axis Configuration - Feedback
<b>SLEBIASB</b>	Defines encoder hardware Cosine offset	Axis Configuration - Feedback
<b>SLEBIASC</b>	Defines the Sin-Cos encoder's hardware compensation for the Sine offset in secondary feedback	Axis Configuration - Feedback
<b>SLEBIASD</b>	Defines the Sin-Cos encoder's hardware compensation for the Cosine offset in secondary feedback	Axis Configuration - Feedback
<b>SLFRC</b>	Static Friction	Servo-Loop - Compensations
<b>SLFRCD</b>	Dynamic Friction	Servo-Loop - Compensations
<b>SLHROUT</b>	Sets the Hall state routing	Commutation
<b>SLIFILT</b>	Internal Current filter	Servo-Loop - Current

Name	Description	Variable Group
<b>SLIKI</b>	Integrator Gain	Servo-Loop - Current
<b>SLIKP</b>	Integrator Proportional Gain	Servo-Loop - Current
<b>SLIFILT</b>	Internal Current filter	Servo-Loop - Current
<b>SLILI</b>	Determines output voltage	Servo-Loop - Current
<b>SLOFFS</b>	Current Command Offset	Servo-Loop - Current
<b>SLLIMIT</b>	Soft Left Limit	Safety Limits
<b>SLLROUT</b>	Sets the HW limits routing for the specified axis	Commutation
<b>SLP2ROUT</b>	Sets the feedback routing of the secondary feedback position for the specified axis	Servo Loop - Miscellaneous
<b>SLPKITF</b>	Increases position loop integrator coefficient	Servo-Loop - Position
<b>SLPKP</b>	Proportional Position Gain	Servo-Loop - Position
<b>SLPKPIF</b>	Provides an Idle Factor to the SLPKP variable	Servo-Loop - Position
<b>SLPKPSF</b>	Provides a Settling Factor to the SLPKP variable	Servo-Loop - Position
<b>SLPKPTF</b>	Increases position loop proportional coefficient	Servo-Loop - Position
<b>SLPMAX</b>	Modulo Axis Upper Limit	Axis Configuration
<b>SLPMIN</b>	Modulo Axis Lower Limit	Axis Configuration
<b>SLPROUT</b>	Position Feedback Routing	Servo-Loop - Miscellaneous
<b>SLSTHALL</b>	The hall state of each axis	Commutation

Name	Description	Variable Group
<b>SLTFWID</b>	Determines distance to target at which position and velocity loops gains increase by 50% Servo-Loop - Miscellaneous SLVBODD Bi-Quad filter	Servo-Loop - Miscellaneous
<b>SLVBODD</b>	Bi-Quad filter damping ratio denominator	Servo-Loop - Filter
<b>SLVBODF</b>	Bi-Quad filter algorithm denominator	Servo-Loop - Filter
<b>SLVBOND</b>	Bi-Quad filter damping ratio numerator	Servo-Loop - Filter
<b>SLVBONF</b>	Bi-Quad filter algorithm numerator	Servo-Loop - Filter
<b>SLVKI</b>	Velocity Integrator Coefficient	Servo-Loop - Velocity
<b>SLVKIIF</b>	Provides an Idle Factor to SLVKI variable	Servo-Loop - Velocity
<b>SLVKISF</b>	Provides a Settle Factor to the SLVKI variable	Servo-Loop - Velocity
<b>SLVKITF</b>	Increases velocity loop integrator coefficient	Servo-Loop - Velocity
<b>SLVKP</b>	Proportional Velocity Gain	Servo-Loop - Velocity
<b>SLVKPIF</b>	Provides an Idle Factor to the SLVKP variable	Servo-Loop - Velocity
<b>SLVKPSF</b>	Provides a Settle Factor to the SLVKP variable	Servo-Loop - Velocity
<b>SLVKPTF</b>	Increases the velocity loop proportional coefficient	Servo-Loop - Velocity
<b>SLVLI</b>	Integrator Velocity Limit	Servo-Loop - Velocity
<b>SLVNATT</b>	Notch Filter Attenuation	Servo-Loop - Filter
<b>SLVNFRQ</b>	Notch Filter Frequency	Servo-Loop - Filter

Name	Description	Variable Group
SLVNWID	Notch Filter Width	Servo-Loop - Filter
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 <sup>st</sup> Ethernet	Communication
SYNC	Slave Sync counter	Safety Control
TARGRAD	Target Radius	Axis Configuration
TCPIP	IP Address for 1 <sup>st</sup> Ethernet	Communication
TCPIP2	IP Address for 2 <sup>nd</sup> Ethernet	Communication
TCPPORT	TCP Port Number	Communication
TIME	Elapsed Time	Monitoring
TPOS	Target Position	Motion
UDPPORT	UDP Port Number	Communication

Name	Description	Variable Group
USAGE	MPU Usage	Monitoring
VEL	Velocity	Motion
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
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

Name	Description
MFF	Master Feed forward
MFLAGS	Motor Flags
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

**AFLAG\$axis\_index.bit\_designator = 1|0**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>bit_designator</b>	The meanings of <b>bit_designator</b> are given in <a href="#">3.1.1.meanings of bit_designator</a>

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/ENABLEALL**.

#### Syntax

**ENTIME(axis\_index) = value**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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/ENABLEALL** 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/ENABLEALL](#)

#### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

#### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.1.3 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.1.4 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>bit_designator</b>	The <b>MFLAGS</b> bit designators are given in <a href="#">Table 3-3</a> .

**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"> <li>● In SPiiPlus controllers, excluding Control Modules, the two analog outputs of the axis can be used as General Purpose outputs.</li> <li>● In all SPiiPlus products: all faults of the axis are disabled.</li> <li>● In all SPiiPlus products <a href="#">ENABLE/ENABLEALL</a> is disabled.</li> </ul>
#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 <a href="#">DCOM</a>.</p>
#MICRO	2	<p>0 (default): If MFLAGS.#PHASE2 = 1 – motor operated in full step mode. If MFLAGS.#PHASE2 = 0 – bit is ignored</p>

Name	No.	Description
		1: If MFLAGS.#PHASE2 = 1 – motor operated in micro step mode. If MFLAGS.#PHASE2 = 0 – #MICRO bit should be cleared
#HOME	3	0 (default): Axis homing procedure not done. 1: Axis homing procedure done. This bit can be changed by a user program to 1 ( <b>MFLAGS(</b> <i>axis_index</i> <b>).#HOME</b> = 1) following a successful homing process. The controller clears the bit during power-up, but then neither changes the bit nor verifies its value. Based on this bit the user can implement any logic in his program application.
#STEPPER	4	<b>#STEPPER</b> is applicable only for PDMnt products. 0 (default): Axis is defined as a servo motor. 1: Axis is defined as a pulse-direction stepper motor in open loop.
#ENCLOOP	5	<b>#ENCLOOP</b> is not applicable for NT products. 0 (default): Stepper feedback loop is not active. 1: The axis provides a stepper feedback loop. In this case the <b>FPOS</b> ( <i>axis_index</i> ) 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 <b>#STEPENC</b> (bit 6 = 1). <b>#ENCLOOP</b> is effective only if <b>#STEPPER</b> (bit 4) = 1.
#STEPENC	6	0 (default): Stepper encoder feedback loop is not active. 1: The controller provides an encoder feedback to the pulse-direction stepper. In this case, the <b>FPOS</b> ( <i>axis_index</i> ) variable will indicate the quadrature encoder counts, and not the stepper pulse-direction pulses if <b>#ENCLOOP</b> (bit 5 = 1). <b>#STEPENC</b> is effective only if <b>#STEPPER</b> (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.
#NANO	7	<b>#NANO</b> is applicable only for UDMnt-x (new revision) and UDIhp-x products. 0 (default): Defines the axis as a servo or stepper motor. 1: Defines the axis as a Nanomotion piezo ceramic motor.

Name	No.	Description
#BRUSHL	8	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. 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.
#BRUSHOK	9	<b>#BRUSHOK</b> is applicable only for DC Brushless motors when <b>#BRUSHL.8 =1</b> . 0 (default): DC brushless motor is not commutated. 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.
#PHASE2	10	0 (default): Two phase motor is not selected. 1: Two phase motor is selected.
#DBRAKE	11	0 (default): Dynamic brake is disabled 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 <b>VELBRK</b> . See the <i>SPiiPlus Setup Guide</i> , Dynamic Brake for a complete explanation.
#INVENC	12	0 (default): Encoder counting is non-inverted. 1: The controller inverts the direction of encoder counting. This does not affect the motion direction.
#INVDOUT	13	0 (default): Drive output command/s are not inverted. 1: The controller inverts the drive output command/s. This effectively inverts the direction of the motion and the sign of the feedback.
#NOTCH	14	0 (default): Notch filter is disabled. 1: Notch filter is active
#NOFILT	15	0 (default): The control algorithm includes a second-order filter specified by bandwidth <b>SLVSOF</b> . 1: The control algorithm by-passes the second-order filter.
#BI_QUAD	16	0 (default): First BiQuad Filter is disabled.

Name	No.	Description
		1: First BiQuad Filter is active.
#DEFCON	17	<p>0: <b>CONNECT</b> is allowed. See <a href="#">CONNECT</a></p> <p>1 (default): The controller applies the default connection between <a href="#">APOS</a> and <a href="#">RPOS (RPOS=APOS)</a>. While <b>#DEFCON</b> = 1, the controller does not accept <b>CONNECT</b> for the axis.</p> <p>The bit is reset to 1 every time the controller is restarted.</p>
#FASTSC	18	<p>0 (default): not available</p> <p>1: For working with 5mHz SIN-COS encoders</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>Bit must be set to 1 in the middle of quadrant.</p> </div>
#ENMOD	19	<p>0: Enable time is defined by <a href="#">ENTIME</a>, or when the drive switches its drive alarm signal to the inactive state - whichever comes first. If the signal remains active more than <a href="#">ENTIME</a> milliseconds, <a href="#">ENABLE/ENABLEALL</a> fails.</p> <p>1 (default): The value of <a href="#">ENTIME</a> defines the enable time.</p>
#DUALLOOP	20	<p>0 (default): The control algorithm implements a regular single-loop scheme.</p> <p>1: The control algorithm implements a dual-loop scheme.</p> <p>See <i>SPiiPlus Setup Guide, Appendix B</i> for further details.</p>
#LINEAR	21	<p>0 (default): The axis is defined as rotary.</p> <p>1: The axis is defined as linear.</p>
#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 <a href="#">ENABLE/ENABLEALL</a> and <a href="#">DISABLE</a>, and variables <a href="#">BOFFTIME</a>, <a href="#">BONTIME</a> variables.</p>

Name	No.	Description
#HSSI	24	#HSSI is not currently supported by NT products. 0 (default): Direct drive connection. 1: Remote drive connection via HSSI.
#GANTRY	25	0 (defaults): standard (SISO) PIV control scheme. 1: gantry control (MIMO) scheme.
#BI_QUAD1	26	(default): Second BiQuad Filter is disabled. 1: Second BiQuad Filter is active.
#HALL	27	 <b>#HALL</b> applies only to DC brushless motors when <b>#BRUSHL</b> (bit 8) =1. 0 (default): Commutation is not based on Hall signals. 1: Commutation is based on Hall signals.
#INVHALL	28	 <b>#INVHALL</b> applies only for DC brushless motors when <b>#BRUSHL</b> (bit 8) =1. 0 (default): Motor Hall signals are non-inverted. 1: Motor Hall signals are inverted.
#MODULO	29	0 (default): Axis is defined as non-modulo. 1: Axis is defined as <b>MODULO</b> . In <b>MODULO</b> mode, physically, the motion of the axis is not limited, but each time <b>RPOS</b> goes out of the defined <b>SLPMIN...SLPMAX</b> range, the controller brings <b>RPOS</b> into range by changing the internal offset <b>EOFFS</b> . See <b>SLPMAX</b> and <b>SLPMIN</b> .
#USER1	30	The functionality of this bit can be defined by the user. 0 (default): Functionality not defined 1: User defined functionality
#USER2	31	The functionality of this bit can be defined by the user. 0 (default): Functionality not defined 1: User defined functionality

### 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/ENABLEALL](#), [GETCONF](#)

### Related ACSPL+ Variables

[SLPROUT](#), [APOS](#), [RPOS](#)

### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

### C Library Functions

acsc\_ReadInteger, acsc\_WriteInteger

## 3.1.5 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.1.6 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.1.7 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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



**SLPMIN** values cannot be modified if protection is applied to this variable through [SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection](#)

#### Related ACSPL+ Variables

[SLPMax](#), [EOFFS](#)

#### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

#### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.1.8 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 [SPiiPlus Setup Guide](#), Stepper Drive section for more information.

#### Syntax

**STEPF(axis\_index) = value**

#### Arguments

<b>axis_index</b>	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 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

**3.1.9 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****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  $12.0 \times 10^{-6}$  (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

**3.1.10 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.1.11 Brake Variables

The Brake variables are:

Name	Description
<b>BOFFTIME</b>	Brake Deactivation Time
<b>BONTIME</b>	Brake Activation Time
<b>VELBRK</b>	Braking Velocity

#### 3.1.11.1 BOFFTIME

##### Description

**BOFFTIME** is a real array, with one element for each axis in the system, and is used for specifying the brake deactivation time in milliseconds.

##### Syntax

**BOFFTIME(axis\_index) = value**

##### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

##### Related ACSPL+ Commands

**ENABLE/ENABLEALL** - 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

### 3.11.2 BONTIME

#### Description

**BONTIME** is a real array, with one element for each axis in the system, and is used for specifying the brake activation time in milliseconds.

#### Syntax

**BONTIME(axis\_index) = value**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

#### Related ACSPL+ Commands

[DISABLE](#), [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

### 3.11.3 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	value ranges from 0 to 1.79769e+308, Default = 0.

**Tag**

140

**Comments**

If [MFLAGS\(axis\\_index\).#BRAKE](#) 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\(axis\\_index\).#BRAKE](#)**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadReal, acsc\_WriteReal

**3.1.12 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_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.

Name	Description
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
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
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.

Name	Description
<a href="#">SC2COFFS</a>	Defines the Sin-Cos Cosine offset
<a href="#">SC2GAIN</a>	A Sin-Cos encoder gain compensation variable used to compensate the Cosine signal for an improper amplitude relative to the Sine signal
<a href="#">SC2PHASE</a>	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
<a href="#">SC2SOFFS</a>	Defines Sin-Cos Sine offset
<a href="#">SLEBIASA</a>	Defines firmware sine offset.
<a href="#">SLEBIASB</a>	Defines hardware cosine offset.
<a href="#">SLEBIASC</a>	Defines the Sin-Cos encoder's hardware compensation for the Sine offset
<a href="#">SLEBIASD</a>	Defines the Sin-Cos encoder's hardware compensation for the Cosine offset
<a href="#">SLABITS</a>	Total number of absolute position bits for an absolute encoder
<a href="#">S2LABITS</a>	Sets the total number of absolute position bits for an absolute encoder connected to a secondary feedback
<a href="#">SCGAIN</a>	Feedback gain compensation variable
<a href="#">SCPHEASE</a>	Feedback phase compensation variable

### 3.1.12.1 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:

$$\text{EOFFS} = \text{EOFFS} - \text{E\_AOFFS}.$$

#### Tag

300

#### Accessibility

Read-Write

### Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

### C Library Functions

acsc\_ReadInteger, acsc\_WriteInteger

#### 3.1.12.2 *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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<p>value of each member can be one of:</p> <ul style="list-style-type: none"> <li>● 2 (shown as 2MHz but practically is 2.5MHz, default for an analog Sin-Cos encoder)</li> <li>● 20 (default for a digital encoder)</li> <li>● 60</li> </ul>

##### 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



### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

### C Library Functions

acsc\_ReadInteger, acsc\_WriteInteger

#### 3.1.12.3 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	<p>Value of each member can be one of the following:</p> <ul style="list-style-type: none"> <li>● 2 (shown as 2MHz but practically is 2.5 MHz, default for an analog Sin-Cos encoder)</li> <li>● 20 (default for a digital encoder)</li> <li>● 60</li> </ul>

##### 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

#### 3.1.12.4 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>bit_designator</b>	The meanings of bit_designator are given in <a href="#">Table 3-4</a> .

**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

### 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.

**#UNSIGND** 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 EOFS 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 in 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

***3.1.12.5 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
#ERRLOGIC	0	Encoder Error Logics
#UNSIGNED	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 EOFS 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

**3.1.12.6 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**

<b>axis</b>	The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1.
<b>value</b>	The encoder data transmission actual frequency in MHz ranging from 1.25 to 10.

**Tag**

248

**Accessibility**

Read-Write

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadReal, acsc\_WriteReal

**3.1.12.7 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**

<b>axis</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	The encoder data transmission actual frequency in MHz ranging from 1.25 to 10.

**Tag**

304

**Accessibility**

Read-Write

### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

### C Library Functions

acsc\_ReadInteger, acsc\_WriteInteger

#### 3.1.12.8 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

<b>axis</b>	The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1.
<b>value</b>	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

#### 3.1.12.9 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

<b>axis</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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

***3.1.12.10 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**

<b>axis</b>	The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1.
<b>value</b>	The interval of encoder position reading in microseconds, valid range [0..3]

**Valid Values**

<b>0</b>	50 microseconds
<b>1</b>	100 microseconds
<b>2</b>	200 microseconds
<b>3</b>	400 microseconds

**Tag**

258

**Accessibility**

Read-Write

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadInteger, acsc\_WriteInteger

***3.1.12.11 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

<b>axis</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.1.12.12 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

<b>axis</b>	The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1.
<b>value</b>	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

**3.12.13 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**

<b>axis</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.12.14 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**

<b>axis</b>	The specific axis index. Valid numbers are: 0, 1... up to the number of axes in the system, minus 1.
<b>value</b>	Error bits MASK; minimum value is 0 (default – no error bits). The range is [0,2E_PAES_D-1]

**Tag**

267

**Accessibility**

Read-Write

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadInteger, acsc\_WriteInteger

**3.12.15 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**

<b>axis</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.12.16 E\_SC\_MUL****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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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  $65536 = 216$ . The minimum value of 2 corresponds to a multiplication of  $4 = 22$ .

### 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

### 3.12.17 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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  $65536 = 216$ . The minimum value of 2 corresponds to a multiplication of  $4 = 22$

### Accessibility

Read-Write

### Com Library Methods and .NET Library Methods

ReadVariable, WriteVariable

### C Library Functions

acsc\_ReadInteger, acsc\_WriteInteger

### 3.12.18 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<p><b>value</b> of each element can be one of:</p> <ul style="list-style-type: none"> <li>● 0 - Up-down counter</li> <li>● 1 - Clock direction counter</li> <li>● 2 - Quadrature single-ended encoder</li> <li>● 3 - Quadrature encoder (Default)</li> <li>● 4 - 250 KHz / 500 KHz Sin-Cos encoder multiplier</li> <li>● 5 - HSSI encoder</li> <li>● 9 - Resolver</li> <li>● 10 - EnDat 2.2</li> <li>● 11 - Smart-Abs</li> <li>● 12 - Panasonic</li> <li>● 13 - BiSS-C</li> <li>● 14 - Hiperface</li> <li>● 18 - 5 MHz Sin-Cos encoder multiplier</li> </ul>

#### Tag

29

#### Comments

The most common encoder type is quadrature, which corresponds to the default value 3.

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 2 is supported by the following products:

- UDMlc-x-048 (where x is either 2 or 4)
- UDIlt-x / UDIhp-x (where x is either 2 or 4)

A value of 18 is supported by the following products:

- UDMnt-x (where x is either 1 or 2)
- UDIhp-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

### 3.12.19 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

<b>axis</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	<p>value of each member can be one of the following:</p> <ul style="list-style-type: none"> <li>● 0 – not defined</li> <li>● 1 - Clock direction counter</li> <li>● 2 - Quadrature single-ended encoder</li> <li>● 3 - Quadrature encoder (Default)</li> <li>● 4 - 250 KHz / 500 KHz Sin-Cos encoder multiplier</li> </ul>

- 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\_WritelInteger

**3.1.12.20 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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:

$$\text{FPOS} = \text{FP} * \text{EFAC} + \text{EOFFS}$$

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

### 3.1.12.21 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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](#), [EOFFS](#)

## COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

## C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.1.12.22 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:

$$\text{FPOS} = \text{FP} * \text{EFAC} + \text{EOFFS}$$

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

#### 3.1.12.23 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

#### 3.1.12.24 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

### 3.12.25 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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



**FVFIL** 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

**3.1.12.26 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

**3.1.12.27 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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:

$$\Delta = (\text{RPOS} - \text{RPOS}_{n-1}) * K$$

$$\text{RVEL}_n = \Delta$$

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:

$$\text{RVEL}_n = \Delta * (1 - (\text{RVFIL}/100)) + (\text{RVEL}_{n-1} * (\text{RVFIL}/100))$$

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](#)

## 3.1.12.28 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

**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**

202

**Comments**

The digital range corresponds to  $\pm 0.5$  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

**3.1.12.29 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****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**

203

**Comments**

The digital range corresponds to  $\pm 0.5$  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

**3.1.12.30 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	value designates the offset ranging from -32766 to 32766.

**Tag**

311

**Comments**

The digital range corresponds to  $\pm 0.5$  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

**3.1.12.31 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
-------------------	--

<b>value</b>	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

**3.12.32 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.12.33 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	value designates the offset ranging from -32766 to 32766.

**Tag**

314

**Comments**

The digital range corresponds to  $\pm 0.5$  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

**3.1.12.34 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.1.12.35 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.1.12.36 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.112.37 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0,1,2,... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.112.38 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

#### 3.12.39 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

<b>axis_index</b>	Specific axis, range: 0 up to number of axis in the system minus 1.
<b>value</b>	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

#### 3.12.40 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	value ranges between 0.5 and 1.5; Default: 1.

**Tag**

204

**Comments**

The value of **SCGAIN** 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

**3.12.41 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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`

### 3.1.13 Safety Limits Variables

The Safety Limits variables are:

Name	Description
<code>CERRA</code>	Critical Position Error (Accelerating)
<code>CERRI</code>	Critical Position Error (Idle)
<code>CERRV</code>	Critical Position Error (Velocity)
<code>DELI</code>	Delay on Transition to Idle State
<code>DELV</code>	Delay on Transition to Velocity State
<code>ERRA</code>	Tolerable Error (Accelerating)
<code>ERRI</code>	Tolerable Error (Idle)
<code>ERRV</code>	Tolerable Error (Velocity)
<code>SLLIMIT</code>	Software Left Limit-feedback count down limit
<code>SLLROUT</code>	Sets the HW limits routing for the specified axis
<code>SRLIMIT</code>	Software Right Limit-feedback count up limit
<code>XACC</code>	Over Acceleration fault parameter
<code>XCURI</code>	Maximum idle motor current
<code>XCURV</code>	Maximum drive current during motion
<code>XRMS</code>	Axis RMS over current fault parameter
<code>XRMST</code>	RMS Current Time Constant
<code>XSACC</code>	Maximum slave axis acceleration
<code>XVEL</code>	Over Velocity fault parameter

#### 3.1.13.1 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.1.13.2 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.1.13.3 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.1.13.4 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 SPiiPlus 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

**3.1.13.5 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.1.13.6 **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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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](#), [ERRQV](#), [DELI](#), [DELV](#)

#### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable,

#### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.1.13.7 **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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.1.13.8 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.1.13.9 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.13.10 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

**3.13.11 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges from -1.79769e+308 to 1.79769e+308, Default = 2e+014.

**Tag**

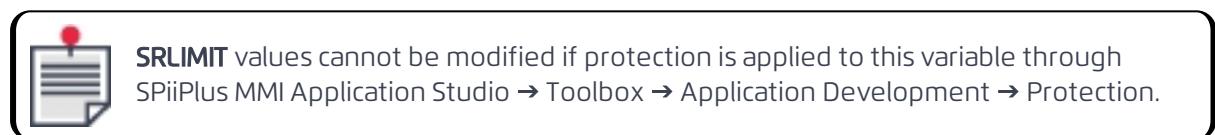
128

**Comments**

If reference position [RPOS](#) is less than this value, a software Right Limit fault results and [FAULT\(axis\\_index\).#SRL](#) is = 1.

**Accessibility**

Read-Write

**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

**3.1.13.12 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.1.13 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.

#### Syntax

**XCURI(axis\_index) = value**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges from 0 to 100, Default = 50.

#### Tag

142

#### Comments

**XCURI** is defined as a percentage of the maximum output voltage. For example, in SPiiPlus PCI-4/8, the maximum output voltage is ±10V. Setting **XCURI** to 50 will limit the drive output to ±5V when the motor is idle.

### 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

**3.1.13.14 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.

**Syntax****XCURV(axis\_index) = value****Arguments**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges from 0 to 100, Default = 50.

**Tag**

143

**Comments**

**XCURV** is defined as a percentage of the maximum output voltage. For example, in SPiiPlus PCI-4/8, the maximum output voltage is  $\pm 10V$ . Setting **XCURV** to 50 will limit the drive output to  $\pm 5V$  when the motor is moving.

**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**, **XCURV****COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable, GetFault

**C Library Functions**

acsc\_ReadReal, acsc\_WriteReal, acsc\_GetFault

**3.1.13.15 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.

**Syntax**

**XRMS(axis\_index) = value****Arguments**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	value ranges from 0 to 100, Default = 50.

**Tag**

144

**Comments**

The SP program calculates **RMS** of the corresponding controller drive output. 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

**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

**3.1.13.16 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.1.13.17 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.1.13.18 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges from 2.22507e-308 to 1.79769e+308, Default = 2e+006.

**Tag**

147

**Comments**

If **RVEL** feedback 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

**3.2 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	RMS current
ROFFS	Reads the offset calculated by the controller in the connect formula.

### 3.2.1 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 <b>DISP</b> 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.
<i>bit_designator</i>	A description of the AST bit designators is given in <a href="#">Table 3-5</a> .

Table 3-5. AST Bit Descriptions

Bit Name	No.	Description
#LEAD	0	1 = axis is leading in a group
#PEG	2	1 = PEG is in progress
#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

Bit Name	No.	Description
#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.
#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 <math>\frac{1}{2}</math> 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+ <a href="#">FCLEAR</a> command.

**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

### 3.2.2 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

### 3.2.3 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>bit_designator</b>	The <b>IST</b> has four bit designators: ● <b>#IND</b> (bit 0) - Primary encoder index

- **#IND2** (bit 1) - Secondary encoder index
- **#MARK** (bit 2) - Mark 1
- **#MARK2** (bit 3) - Mark 2

<b>value</b>	<b>value</b> 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

**3.2.4 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 **MARK2** 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 **#MARK2** = 1.

To resume the latching logic, [IST\(axis\\_index\)](#).#MARK2 must be explicitly cleared with the command [IST\(axis\\_index\).#MARK2=0](#).

### Accessibility

Read-Only

### Related ACSPL+ Variables

[IST](#), [FPOS](#), [F2POS](#), [MARK](#)

### COM Library Methods and .NET Library Methods

ReadVariable, GetIndexState,

### C Library Functions

acsc\_ReadReal, acsc\_GetIndexState

## 3.2.5 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 **#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.

### 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

## 3.2.6 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 3-6](#), for the given axis.

**Table 3-6. 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 <a href="#">TARGRAD</a> and <a href="#">SETTLE</a> )
#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.

#### 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

### 3.2.7 RMS

#### Description

**RMS** is a real array, with one element for each axis in the system, the elements of which store the RMS current for an axis (in %). The value ranges between 0 and 100.

#### Tag

223

#### Accessibility

Read-Only

#### Related ACSPL+ Variables

[XRMS](#)

#### COM Library Methods and .NET Library Methods

ReadVariable

#### C Library Functions

acsc\_ReadReal

### 3.2.8 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

<b>node_index</b>	Designates the specific node, valid numbers are: 0, 1, 2, ... up to the number of nodes in the system minus 1.
<b>bit_designator</b>	The meanings of <b>bit_designator</b> are given in <a href="#">Table 3-7</a> .

Table 3-7. NST Bit Description

Bit Name	Bit No.	Description
#SYNC	0	Sync error
#GPRT	1	GPRT error

#### 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.

**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

## 3.3 Data Collection Variables

The Data Collection variables are:

Name	Description
<b>DCN</b>	Axis Data Collection, Number of Samples
<b>DCP</b>	Axis Data Collection, Period
<b>S_DCN</b>	System Data Collection, Number of Samples
<b>S_DCP</b>	System Data Collection, Period
<b>S_ST</b>	State of System Data Collection

### 3.3.1 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.

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`

## 3.3.2 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`

### 3.3.3 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`

### 3.3.4 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`

## 3.3.5 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`

### 3.4 Input and Output Variables

The Input and Output variables are:

Name	Description
<code>AIN</code>	Analog Inputs
<code>AOUT</code>	Analog Outputs
<code>COMMCH</code>	Returns the last activated communication channel.
<code>DCOM</code>	Drive command - in open loop
<code>DOUT</code>	Drive output.
<code>EXTIN</code>	Extended digital inputs (HSSI)
<code>EXTOUT</code>	Extended digital outputs (HSSI)
<code>IN</code>	General Purpose Digital Inputs
<code>OUT</code>	General Purpose Digital Outputs

#### 3.4.1 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

`AIN(index) = value`

##### Arguments

<code>index</code>	A number between 0 up to the maximum number of analog input signals minus one.
<code>value</code>	<code>value</code> 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

### 3.4.2 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

<b>index</b>	A number between 0 up to the maximum number of analog output signals minus one.
<b>value</b>	<b>value</b> 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

### 3.4.3 DOUT

#### Description

DOUT is an eight member (one for each axis) integer array which stores the output of the velocity loop.

#### Tag

26

## Comments

When operating in the open loop mode ([MFLAGS.1=1](#)), DOUT equals DCOM.

When operating in the closed loop mode ([MFLAGS.1=0](#)), DOUT equals DCOM plus the servo command.

DOUT values range from -32767 to 32768 (16-bit), Default = 0.

*The following ACS products do not support DOUT:*



- UDMpc
- CMnt
- UDMpm
- MC4U
- SPiiPlus SAnt

## Accessibility

Read-Only, [ReadVariable](#)

## COM Library Methods and .NET Library Methods

[ReadVariable](#), [C Library Functions](#), [acsc\\_ReadInteger](#)

### 3.4.4 EXTIN

#### Description

**EXTIN** is an integer array, the size of which is determined by the total number of HSSI input signals in the system, and reads the current state of the HSSI inputs. The number of inputs depends on the number of HSSI modules in the system.

For details about the HSSI, see the *HSSI Modules Hardware Guide*.

#### 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](#)

### 3.4.5 EXTOUT

#### Description

**EXTOUT** is an integer array, the size of which is determined by the total number of HSSI output signals in the system, which can be used for reading or setting the current state of the HSSI outputs. The number of outputs depends on the number of HSSI modules in the system.

For details about the HSSI, see the *HSSI Modules Hardware Guide*.

### Syntax

**EXTOUT(index) = value**

### Arguments

<b>index</b>	A number between 0 and 511.
<b>value</b>	<b>value</b> ranges from -2147483648, 2147483647, Default = 0.

### 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

## 3.4.6 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

<b>port</b>	A number between 0 and the total number of ports in the system minus one.
<b>bit</b>	<b>bit</b> 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

## 3.4.7 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

<b>port</b>	A number between 0 the total number of ports in the system minus one.
<b>bit</b>	<b>bit</b> 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 [ASSIGNPEG](#).

## 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

## 3.5 Monitoring Variables

The Monitoring variables are:

Name	Description
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.
SOFTIME	Parameter which specifies the EtherCAT frame delivery time in microseconds.
TIME	Elapsed Time
USAGE	MPU Usage

### 3.5.1 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

### 3.5.2 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

### 3.5.3 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 90% 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.

### 3.5.4 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

**Tag**

245

**Comments**

The real-time tasks always have the greatest priority. If the usage reaches 90% 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.

***3.5.5 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)

***3.5.6 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

***3.5.7 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 90% 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

***3.6 Motion Variables***

The Motion variables are:

Name	Description
ACC	Default Acceleration
APOS	Axis Position
DEC	Default Deceleration
DAPOS	Delayed Axis Position
FACC	Feedback Acceleration
FPOS	Feedback Position
F2POS	Secondary Feedback Position
FVEL	Feedback Velocity

Name	Description
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
GVEC	Group Vector
GVEL	Group Velocity
JERK	Default jerk
KDEC	Default kill deceleration
MPOS	Master Position
NVEL	Sets a non-zero axis velocity in stepper motor applications
TPOS	Target position for track motion
PE	Position Error
RACC	Reference Acceleration
RJERK	Reference Jerk
RPOS	Reference Position
RVEL	Reference Velocity
VEL	Velocity

### 3.6.1 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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](#)

### 3.6.2 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

**3.6.3 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

#### 3.6.4 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

#### 3.6.5 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

**3.6.6 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](#)**COM Library Methods and .NET Library Methods**

ReadVariable, GetFPosition, SetFPosition

**C Library Functions**

acsc\_ReadReal, acsc\_GetFPosition, acsc\_SetFPosition

**3.6.7 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](#)

## Related ACSPL+ Variables

[IST](#)

## COM Library Methods and .NET Library Methods

ReadVariable

## C Library Functions

acsc\_ReadReal

## 3.6.8 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

## 3.6.9 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

**3.6.10 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

**3.6.11 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

## 3.6.12 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

## 3.6.13 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

**3.6.14 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

**Accessibility**

Read-Only

**COM Library Methods and .NET Library Methods**

ReadVariable

**C Library Functions**

acsc\_ReadInteger

**3.6.15 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

**3.6.16 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

The following values apply only in the case of **MASTER - SLAVE** motion:

- 9 - asynchronous phase of master-slave motion
  - 10 - synchronous phase of master-slave motion
  - 11 - stalled phase of master-slave motion.
- 12 - dwell phase in **JOG** or **MPTP...ENDS** motions, or no defined target point in **PATH...ENDS**, **PVSPLINE...ENDS** or **MPTP...ENDS** motions.



**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

### 3.6.17 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

### 3.6.18 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

### 3.6.19 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

### 3.6.20 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

### 3.6.21 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

### 3.6.22 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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`

## 3.6.23 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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`

### 3.6.24 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`

### 3.6.25 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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** 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

## 3.6.26 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

**3.6.27 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

**3.6.28 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

**3.6.29 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/ENABLEALL`, `DISABLE`, `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/ENABLEALL`, `DISABLE`, `KILL`

### Related ACSPL+ Variables

`MFLAGS(axis_index).#DEFCON` (bit 17 = Default Connection)

### COM Library Methods and .NET Library Methods

ReadVariable

### C Library Functions

`acsc_ReadReal`

## 3.6.30 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

**3.6.31 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, GetRVVelocity

**C Library Functions**

acsc\_ReadReal, acsc\_GetRVVelocity

**3.6.32 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.6.33 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 **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](#), [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`

## 3.7 Program Execution Control Variables

The Program Execution Control variables are:

Name	Description
<a href="#">ONRATE</a>	Autoroutine Rate
<a href="#">PCHARS</a>	Program Size in Characters
<a href="#">PERL</a>	Program Error Line
<a href="#">PERR</a>	Program Error
<a href="#">PEXL</a>	Executed Line
<a href="#">PFLAGS</a>	Program Flags
<a href="#">PLINES</a>	Number of Lines
<a href="#">PRATE</a>	Program Rate
<a href="#">PST</a>	Program State

### 3.7.1 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

<b>buffer_index</b>	<b>buffer index</b> - a number between 0 and the total number of buffers minus one (the highest number is that of the D-Buffer).
<b>value</b>	<b>value</b> 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\_Writelnteger

**3.7.2 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

**3.7.3 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

**3.7.4 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.

**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

**3.7.5 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.

## 3.7.6 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.

### 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 3-8](#)

Table 3-8. 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.
#DYNAMIC	2	0 (default): Buffer works in normal order. 1: Not applicable (obsolete option)



Supported after applying protection, see [Protection Wizard](#) in the *MMI Application Studio User Guide*.

Bit Name	No.	Description
#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)
#NOVIEW	6	0 (default): The program is visible in the buffer. 1: The program in the buffer hidden from being viewed.   Supported after applying protection, see <b>Protection Wizard</b> in the <i>MMI Application Studio User Guide</i> .

**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

**3.7.7 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

### 3.7.8 PRATE

#### Description

**PRATE** is an integer array with one element for each program buffer plus one for the D-Buffer each element of which is used for defining the program execution rate for the given buffer.

#### Syntax

**PRATE(buffer\_index) = value**

#### Arguments

<b>buffer_index</b>	<b>buffer index</b> - a number between 0 and 16 (16 being the D-Buffer).
<b>value</b>	<b>value</b> 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



#### Related ACSPL+ Variables

[ONRATE](#)

[COM Library Methods and .NET Library Methods](#)

ReadVariable, WriteVariable

## C Library Functions

acsc\_ReadInteger, acsc\_WriteInteger

### 3.7.9 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 3-9](#).

**Table 3-9. 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 <b>STEP</b> 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

### 3.8 Safety Control Variables

The Safety Control variables are:

Name	Description
AERR	Axis Error
CERRA	Critical Position Error (Accelerating)
CERRI	Critical Position Error (Idle)
CERRV	Critical Position Error (Velocity)

Name	Description
<b>DELI</b>	Delay on Transition to Idle State
<b>DELV</b>	Delay on Transition to Velocity State
<b>ECEXTST</b>	EtherCAT state of the SPiiPlusES slave
"ECEXTERR" on the next page	SPiiPlusES EtherCAT error code (based on Application Level Error Code).
<b>FAULT</b>	Faults
<b>ECST</b>	Contains Ethernet status
<b>ECERR</b>	Contains Ethernet error code
<b>FDEF</b>	Default Response Mask
<b>FMASK</b>	Fault Mask
<b>MERR</b>	Motor Error
<b>SAFIN</b>	Safety Inputs
<b>SAFINI</b>	Safety Inputs Inversion
<b>S_ERR</b>	System Error
<b>S_FAULT</b>	System Faults
<b>S_FDEF</b>	System Default Response Mask
<b>S_FMASK</b>	System Fault Mask
<b>S_SAFIN</b>	System Safety Inputs
<b>S_SAFINI</b>	System Safety Inputs Inversion
<b>STODELAY</b>	Configures the delay time between the STO fault indication and the default response (disable) to the fault
<b>SYNC</b>	Slave synchronization indicator

### 3.8.1 AERR

#### Description

**AERR** is an integer array, with one element for each axis in the systems, the elements of which store the termination codes.

#### Tag

2

## Comments

When a motion starts, the controller zeroes the **AERR** elements for each axis involved in the motion. When the motion terminates for any reason, the controller stores the termination code in the corresponding **AERR** element. The element remains unchanged until the next motion starts for the corresponding axis.

Single-axis motion stores its termination code in the **AERR** element that corresponds to the axis. Multi-axis motion stores its termination code in the **AERR** element that corresponds to the leading axis of the motion.

## Accessibility

Read-Only

## Related ACSPL+ Variables

[MERR](#)

## COM Library Methods and .NET Library Methods

ReadVariable, GetMotionError

## C Library Functions

acsc\_ReadInteger, acsc\_GetMotionError

## *3.8.2 ECERR*

### Description

**ECERR** is a scalar variable containing an EtherCAT error code. The EtherCAT error codes are given in [Table 6-6](#).

### 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

## *3.8.3 ECEXTERR*

### Description

**ECEXTER** 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 given in the table below.



Error Code	Error Message	Remarks
7001	General Error	Appears for any unspecified EtherCAT error
7002	Mailbox No Memory	Mailbox memory allocation failure
7017	Invalid Control	Invalid requested state change
7018	Unknown Control	Unknown required state
7019	Boot strap not supported	Bootstrap state is not supported by the slave
7021	Invalid Mailbox Configuration BOOT	Invalid mailbox configuration (BOOT state)
7022	Invalid Mailbox Configuration PREOP	Invalid mailbox configuration (PREOP state)
7023	Invalid SM configuration	Invalid sync manager configuration. Possible reason can be invalid PDO size configuration
7024	No Valid Inputs	No valid inputs available
7025	No Valid Outputs	No valid outputs available
7026	SYNC error	Synchronization error
7027	SM Watchdog	Sync Manager Watchdog
7028	SYNC types not compatible	Invalid Sync Manager types
7029	Invalid SM Out Configuration	Invalid SM Output Configuration
7030	Invalid SM IN Configuration	Invalid SM Input Configuration
7031	Invalid Watchdog	Invalid Watchdog configuration

Error Code	Error Message	Remarks
7039	Free Run Not Supported	FreeRun mode is not supported
7040	SYNC Not Supported	SYNC mode is not supported
7041	Free Run Needs 3-Buffer Mode	FreeRun needs SM 3-Buffer Mode
7042	Background watchdog	Background watchdog
7044	FATAL SYNC Error	Fatal Sync Error
7045	No Sync Error	No Sync Error
7046	Cycle Time Too Small	EtherCAT cycle time smaller than the minimum cycle time supported by the slave
7048	DC Invalid Sync Configuration	Invalid DC SYNC Configuration
7049	DC Invalid Latch Configuration	Invalid DC Latch Configuration
7050	DC PLL Sync Error	PLL Error
7051	DC Sync IO Error	DC Sync IO Error
7052	DC Sync Missed Error	DC Sync Timeout Error
7053	DC Invalid Sync Cycle Time	DC Invalid Sync Cycle Time
7054	DC Sync0 Cycle Time	DC Sync0 Cycle Time
7055	DC Sync1 Cycle Time	DC Sync1 Cycle Time
7067	Mailbox CoE	Mailbox CoE
7068	Mailbox FoE	Mailbox FoE
7080	EEPROM No Access	No EEPROM Access
7081	EEPROM Error	EEPROM Error

**Syntax****ECEXTERR****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

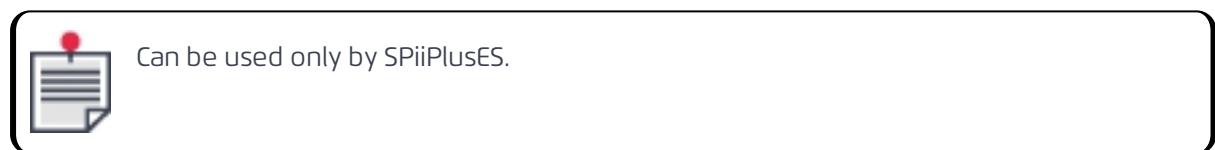
### C Library Functions

Acsc\_ReadInteger

### 3.8.4 ECEXTST

#### Description

**ECEXTST** is a scalar (INT) variable representing the EtherCAT state of the SPiiPlusES slave. The state is reflected in the relevant bits.



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
8	#LNKPORTB	Physical Link Port B

#### Syntax

### ECEXTST

#### Arguments

None

**Comments**

Comments

If the controller is not SPiiPlusES, all bits are 0. If external master is not connected, the SPiiPlusES is in INIT state.

**Tag**

320

**Accessibility**

Read-Only

**Com Library Methods and .NET Library Methods**

ReadVariable

**C Library Functions**

Acsc\_ReadInteger

**3.8.5 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 3-10](#).

**Table 3-10. 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.
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.

Bit	Designator	Description
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 #IN SYNC 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

**3.8.6 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 3-11](#).**Table 3-11. 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.

Bit	Fault	Fault Description
4	#HOT	Motor Overheat. 1 = Motor's temperature sensor indicates overheat.
5	#SRL	Software Right Limit. 1 = Axis reference position ( <a href="#">RPOS</a> ) is greater than the software right limit margin (SRLIMIT).
6	#SLL	Software Left Limit. 1 = Axis reference position ( <a href="#">RPOS</a> ) 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.
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"> <li>● <a href="#">ERRI</a> - Maximum position error while the axis is idle</li> <li>● <a href="#">ERRV</a> - Maximum position error while the axis is moving with constant velocity</li> <li>● <a href="#">ERRA</a> - Maximum position error while the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRV</a></li> </ul>
13	#CPE	>Critical Position Error. 1 = Position error (#PE) exceeds the value of the critical limit.

Bit	Fault	Fault Description
		#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"> <li>● <a href="#">CERRI</a> if the axis is idle (not moving)</li> <li>● <a href="#">CERRV</a> if the axis is moving with constant velocity</li> <li>● <a href="#">CERRA</a> if the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">CERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">CERRV</a></li> </ul>
14	#VL	Velocity Limit. 1 = Absolute value of the reference velocity ( <a href="#">RVEL</a> ) exceeds the limit defined by the <a href="#">XVEL</a> parameter.
15	#AL	Acceleration Limit. 1 = Absolute value of the reference acceleration ( <a href="#">RACC</a> ) exceeds the limit defined by the <a href="#">XACC</a> parameter.
16	#CL	Current Limit. 1 = <a href="#">RMS</a> current calculated in the Servo Processor exceeds the limit value defined by the <a href="#">XRMS</a> 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 [Table 3-16](#)

### 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

## 3.8.7 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>bit_designator</b>	The <b>FDEF</b> bit designators are given in <a href="#">Table 3-12</a> .
<b>value</b>	<b>value</b> ranges from -2147483648 to 2147483647, Default = -1.

Table 3-12. 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.

Bit	Fault	Fault Description	Default Response (FDEF)
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 overheating.	None.
5	#SRL	Software Right Limit 1 = Axis reference position ( <b>RPOS</b> ) 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 ( <b>RPOS</b> ) 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 1 = Secondary encoder (for digital encoder type only) is not connected.	No default response.
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 <b>FCLEAR</b> .

Bit	Fault	Fault Description	Default Response (FDEF)
11	#ENC2	<p>Encoder 2 Error</p> <p>1 = Secondary encoder miscounts.</p>	Same as #ENC.
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"> <li>● <a href="#">ERRI</a> - Maximum position error while the axis is idle</li> <li>● <a href="#">ERRV</a> - Maximum position error while the axis is moving with constant velocity</li> <li>● <a href="#">ERRA</a> - Maximum position error while the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRV</a></li> </ul>	None.
13	#CPE	<p>Critical Position Error</p> <p>1 = Position error (#PE) exceeds the value of the critical limit.</p>	The controller disables the violating axis.

Bit	Fault	Fault Description	Default Response (FDEF)
		<p>#CPE errors occur outside normal range of operation and #CPE &gt; #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"> <li>● <a href="#">CERRI</a> if the axis is idle (not moving)</li> <li>● <a href="#">CERRV</a> if the axis is moving with constant velocity</li> <li>● <a href="#">CERRA</a> if the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">CERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">CERRA</a> to <a href="#">ERRV</a></li> </ul>	
14	#VL	<p>&gt;Velocity Limit</p> <p>1 = Absolute value of the reference velocity (<a href="#">RVEL</a>) exceeds the limit defined by the <a href="#">XVEL</a> parameter.</p>	The controller kills the violating axis.
15	#AL	<p>&gt;Acceleration Limit</p> <p>1 = Absolute value of the reference acceleration (<a href="#">RACC</a>) exceeds the limit defined by the <a href="#">XACC</a> parameter.</p>	The controller kills the violating axis.
16	#CL	Current Limit	The controller disables the violating axis.

Bit	Fault	Fault Description	Default Response (FDEF)
		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.	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 **SPiiPlus 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

### 3.8.8 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 <b>FDEF</b> bit designators are given in <a href="#">Table 3-13</a> .
<i>value</i>	<b>value</b> ranges from -2147483648 to 2147483647, Default=1040414435.

Table 3-13. 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.
5	#SRL	Software Right Limit 1 = Axis reference position ( <b>RPOS</b> ) is greater than the software right limit margin ( <b>SRLIMIT</b> ).
6	#SLL	Software Left Limit 1 = Axis reference position ( <b>RPOS</b> ) is less than the software left limit margin ( <b>SLLIMIT</b> ).
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.

Bit	Fault	Fault Description
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	<p>Position Error 1 = Position error (PE) has occurred.</p> <p>PE is defined by the following variables:</p> <ul style="list-style-type: none"> <li>● <a href="#">ERRI</a> - Maximum position error while the axis is idle</li> <li>● <a href="#">ERRV</a> - Maximum position error while the axis is moving with constant velocity</li> <li>● <a href="#">ERRA</a> - Maximum position error while the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRV</a></li> </ul>
13	#CPE	<p>Critical Position Error 1 = Position error (#PE) exceeds the value of the critical limit.</p> <p>#CPE errors occur outside normal range of operation and #CPE &gt; #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"> <li>● <a href="#">CERRI</a> if the axis is idle (not moving)</li> <li>● <a href="#">CERRQ</a> if the axis is moving with constant velocity</li> <li>● <a href="#">CERRA</a> if the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">CERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">CERRA</a> to <a href="#">ERRV</a></li> </ul>
14	#VL	<p>Velocity Limit 1 = Absolute value of the reference velocity (<a href="#">RVEL</a>) exceeds the limit defined by the <a href="#">XVEL</a> parameter.</p>
15	#AL	<p>Acceleration Limit 1 = Absolute value of the reference acceleration (<a href="#">RACC</a>) exceeds the</p>

Bit	Fault	Fault Description
		limit defined by the <a href="#">XACC</a> parameter.
16	#CL	Current Limit 1 = <b>RMS</b> current calculated in the Servo Processor exceeds the limit value defined by the <a href="#">XRMS</a> 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**

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

**3.8.9 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 3-14](#).

**Table 3-14. MERR Return Values**

Error Code	Description
0	Motor Enabled, in motion, or motion terminated by the user using <a href="#">KILL</a> , <a href="#">HALT</a> , or <a href="#">DISABLE</a> .
5003	Motion was terminated by user
5007	Motor was disabled due to a generalized fault
5008	Motor was killed due to a generalized fault
5010	Motor failed: Hardware Right Limit
5011	Motor failed: Hardware Left Limit
5012	Motor failed: Network error
5014	Motor failed: Motor Overheat
5015	Motor failed: Software Right Limit
5016	Motor failed: Software Left Limit
5017	Motor failed: Encoder Not Connected
5018	Motor failed: Encoder 2 Not Connected
5019	Motor Failed: Drive Fault
5020	Motor Failed: Encoder Alarm
5021	Motor Failed: Encoder 2 Alarm
5022	Motor Failed: Position Error
5023	Motor Failed: Critical Position Error
5024	Motor Failed: Velocity Limit
5025	Motor Failed: Acceleration Limit
5026	Motor Failed: Overcurrent

Error Code	Description
5027	Motor Failed: Servo Processor Alarm
5032	Motor Failed: Attempt of motion while a fault is active
5033	Motor Failed: Attempt of motion in disabled direction
5035	Motor Failed: Program Error
5036	Motor Failed: Memory Overflow
5037	Motor Failed: MPU Overuse
5038	Motor Failed: Hardware Emergency Stop
5039	Motor Failed: Servo Interrupt
5060	(Integrated Control Models only) Drive alarm: No fault
5061	(Integrated models only) Drive alarm: Short Circuit
5062	(Integrated Control Models only) Drive alarm: External Protection Activated
5063	(Integrated Control Models only) Drive alarm: Power supply too low
5064	(Integrated Control Models only) Drive alarm: Power supply too high
5065	(Integrated Control Models only) Drive alarm: Temperature too high
5066	(Integrated Control Models only) Drive alarm: Power supply 24VF1
5067	(Integrated Control Models only) Drive alarm: Power supply 24VF2.
5068	(Integrated Control Models only) Drive alarm: Emergency Stop
5069	(Integrated models only) Drive alarm: Power down
5070	Phase lost.
5071	Drive not ready (power up).
5072	Over-current.
5073	Not in use (reserved).
5074	Damper not ok.

Error Code	Description
5075	(Integrated Control Models only) Drive alarm: Digital drive interface not connected.
5100	Current bias measured is out of range

**Tag**

86

**Comments**

MERR is updated every time the axis motion is terminated. MERR stores the last termination code until either [FCLEAR](#) or [ENABLE/ENABLEALL](#) is executed.

**Accessibility**

Read-Only

**Related ACSPL+ Commands**[FCLEAR](#)**Related ACSPL+ Variables**[AERR, PERR](#)**COM Library Methods and .NET Library Methods**

ReadVariable, GetMotorError

**C Library Functions**

acsc\_ReadInteger, acsc\_GetMotorError

**3.8.10 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.

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

Bit Name	No.	Description
#LL	1	Hardware Left Limit
#HOT	4	Motor Overheat
#DRIVE	9	Drive Fault

## Accessibility

Read-Only



**SAFIN** can be written to when working with the SPiiPlus 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](#)

### 3.8.11 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

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0, 1, 2, .. up to the number of axes in the system minus 1.
<b>bit_designator</b>	The valid <b>SAFINI</b> bits are given in <a href="#">Table 3-15</a> .
<b>value</b>	<b>value</b> 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

Bit Name	No.	Description
#HOT	4	Motor Overheat
#DRIVE	9	Drive Fault

**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

**3.8.12 S\_ERR****Description**

**S\_ERR** is a scalar integer that contains the code of the initialization error set during powerup.

**Tag**

113

**Accessibility**

Read-Only

**Related ACSPL+ Variables**

None

**COM Library Methods and .NET Library Methods**

ReadVariable

**C Library Functions**

acsc\_ReadInteger

**3.8.13 S\_FAULT****Description**

**SFAULT** is a scalar integer variable consisting of a set of bits equating to the occurrence of faults. **SFAULT** has two categories of bits, Axis Faults and System Faults (faults that are not related to any specific axis).

The **SFAULT** bits are described in [Table 3-16](#).

**Table 3-16. SFAULT Fault Bits**

Bit	Fault	Fault Description
<b>Axis Faults</b>		
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 ( <a href="#">RPOS</a> ) is greater than the software right limit margin (SRLIMIT).
6	#SLL	Software Left Limit 1 = Axis reference position ( <a href="#">RPOS</a> ) 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

Bit	Fault	Fault Description
		<p>1 = Position error (PE) has occurred.</p> <p>PE is defined by the following variables:</p> <ul style="list-style-type: none"> <li>● <a href="#">ERRI</a> - Maximum position error while the axis is idle</li> <li>● <a href="#">ERRV</a> - Maximum position error while the axis is moving with constant velocity</li> <li>● <a href="#">ERRA</a> - Maximum position error while the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">ERRV</a></li> </ul>
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 &gt; #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"> <li>● <a href="#">CERRI</a> if the axis is idle (not moving)</li> <li>● <a href="#">CERRV</a> if the axis is moving with constant velocity</li> <li>● <a href="#">CERRA</a> if the axis is accelerating or decelerating</li> <li>● <a href="#">DELI</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">CERRI</a></li> <li>● <a href="#">DELV</a> - Delay on transition from <a href="#">ERRA</a> to <a href="#">CERRV</a></li> </ul>
14	#VL	<p>Velocity Limit.</p> <p>1 = Absolute value of the reference velocity (<a href="#">RVEL</a>) exceeds the limit defined by the <a href="#">XVEL</a> parameter.</p>
15	#AL	<p>Acceleration Limit</p> <p>1 = Absolute value of the reference acceleration (<a href="#">RACC</a>) exceeds the limit defined by the <a href="#">XACC</a> parameter.</p>
16	#CL	<p>Current Limit</p> <p>1 = <b>RMS</b> current calculated in the Servo Processor exceeds the limit value defined by the <a href="#">XRMS</a> 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>

Bit	Fault	Fault Description
20	#HSSINC	HSSI Not Connected 1 = HSSI module is not connected.
<b>System Faults</b>		
22	#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.
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.  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.  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

**Tag**

114

**Comments**

An **SFAULT** bit, such as Left Limit, will be = 1 whenever one or more Left Limit fault bits are = 1. In this manner, **SFAULT** provides an indication of the aggregate state of each **FAULT** bit.

### Accessibility

Read-Only



**SFAULT** 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

## 3.8.14 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 **SFAULT**. **S\_FDEF** is connected to **SFAULT** in the same way that **FDEF** is connected with **FAULT**.

### Syntax

**S\_FDEF**[*bit\_designator*] = *value*

### Arguments

<b>bit_designator</b>	The <b>S_FDEF</b> bits and associated responses are given in <a href="#">Table 3-17</a> .
<b>value</b>	<b>value</b> ranges from -2147483648 to 2147483647, Default = 1.

Table 3-17. **S\_FDEF** Bit Description

Bit	Fault	Fault Description	Default Response ( <b>S_FDEF</b> )
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.

Bit	Fault	Fault Description	Default Response ( <b>S_FDEF</b> )
28	#ES	Hardware Emergency Stop 1 = ES signal is activated.	The controller disables all axes, and sets the offset of each axis to 0.   It does not stop the program buffer s.
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

### 3.8.15 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 <b>S_FMASK</b> bits and associated responses are given in <a href="#">Table 3-18</a> .
<i>value</i>	value ranges from -2147483648 to 2147483647, Default=0.

Table 3-18. **S\_FMASK** Bit Description

Bit	Fault	Fault Description
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.

Bit	Fault	Fault Description
30	#INTGR	File Integrity 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

**3.8.16 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 **SFAULT**, 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

## 3.8.17 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

## 3.8.18 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

<b>axis</b>	The specific axis index. Valid numbers are: 0,1... up to the number of axes in the system, minus 1.
<b>value</b>	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

## 3.8.19 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

## 3.9 Induction Motor Variables

The Induction Motor variables are:

Name	Description
SLCFIELD	Induction Motor Excitation

Name	Description
<a href="#">SLCSLIP</a>	Induction Motor Slip Factor

### 3.9.1 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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:

$$\text{Field} = \frac{\sqrt{2}I_{mag}}{I_{peak}} 100$$

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

### 3.9.2 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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:

$$SLCSLIP = 131 \times f_n [\text{Hz}] \times s_n [\%] \times \frac{I_{peak}}{\sqrt{2} I_n}$$

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

## 3.10 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).

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
<b>SLDZMIN</b>	Parameter which specifies the minimum position of the Dead Zone (when the servo is turned off)
<b>SLDZMAX</b>	Parameter which specifies the maximum position of the Dead Zone (when the servo is turned on).
<b>SLDZTIME</b>	Parameter which specifies the duration (in msec) required for settling after entering the SLDZMIN.
<b>SLZFF</b>	Parameter which specifies the distance from target to stop the velocity Feed Forward.
<b>SLFRC</b>	Parameter which specifies the initial non-zero command to overcome the static friction in positive direction.
<b>SLFRCN</b>	Parameter which specifies the initial non-zero command to overcome the static friction in negative direction.
<b>SLHRS</b>	Parameter which specifies the modulation ratio of the drive command.
<b>SLVKPDCF</b>	Parameter which specifies the multiplication factor of the velocity loop gain (SLVKP) in DC mode.
<b>SLPKPDCF</b>	Parameter which specifies the multiplication factor of the position loop gain (SLPKP) in DC mode.
<b>SLVKIDCF</b>	Parameter which specifies the multiplication factor of the velocity loop integrator (SLVKI) in DC mode.

### 3.10.1 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

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

## 3.10.2 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

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

### 3.10.3 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

<b>axis_index</b>	<i>axis_index</i> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

### 3.10.4 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

<b>axis_index</b>	<i>axis_index</i> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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  $0.1\mu\text{M}$ , set **SLZFF** to 100 - 300 counts (with an equivalent value in user units).
- For an HR8 motor with encoder resolution of  $0.1\mu\text{M}$ , 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

## 3.10.5 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

**SLFRC**(*axis\_index*) = *value*

### Arguments

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

### 3.10.6 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

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

### 3.10.7 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

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

### 3.10.8 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

**axis\_index**

**axis\_index** designates the specific axis: 0 - X, 1 - Y, etc.

**value**

**value** ranges between -1.79769e+308 to 1.79769e+308, Default = 1.

#### Tag

252

#### Accessibility

Read-Write



**SLVKPDCF** 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

### 3.10.9 SLPKPDCF

#### 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**

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

**3.10.10 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**

<b>axis_index</b>	<b>axis_index</b> designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> 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

### 3.11 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 Miscellaneous Variables

#### 3.11.1 DCOM

##### Description

**DCOM** is a real array, with one element for each axis in the system, and is used for defining the direct servo command.

##### Syntax

**DCOM(axis\_index) = value**

##### Arguments

<b>axis_index</b>	axis_index designates the specific axis: 0 - X, 1 - Y, etc.
<b>value</b>	<b>value</b> ranges from -100 to 100, Default = 0.

##### Tag

20

##### Comments

When operating in the open loop mode (**MFLAGS.1=1**), **DCOM** defines a percentage of the maximum drive command, for example, the SPiiPlus PCI 4/8 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** defines an offset.

### 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

## 3.11.2 Servo-Loop Current Variables

The Servo-Loop Current variables are:

Name	Description
<a href="#">SLBIASA</a>	Current phase A bias
<a href="#">SLBIASB</a>	Current phase B bias
<a href="#">SLIKI</a>	Integrator gain
<a href="#">SLIKP</a>	Integrator proportional gain
<a href="#">SLIFILT</a>	Internal current filter
<a href="#">SLIOFFS</a>	Offset to be added to the result of the current loop control
<a href="#">SLILI</a>	Used to limit the drive's output voltage

### 3.11.2.1 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

## COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

## C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.11.2.2 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.11.2.3 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.11.2.4 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges between 0 - 256000, Default = 1000.

#### Tag

171

#### Comments

**SLIKP** is active only in integrated models.

#### Accessibility

Read-Write



#### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

#### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.11.2.5 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.11.2.6 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

**SLIOFFS**(*axis\_index*) = *value*

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

#### 3.11.2.7 *SLILI*

##### Description

**SLILI** 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

**SLILI**(axis\_index) = value

##### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 **SLILI(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

#### 3.11.3 Servo-Loop Velocity Variables

The Servo-Loop Velocity variables are:

Name	Description
<a href="#">SLVKI</a>	Sets velocity integrator coefficient
<a href="#">SLVKIIF</a>	Provides an Idle Factor to the SLVKI variable
<a href="#">SLVKISF</a>	Provides a Settle Factor to the SLVKI variable
<a href="#">SLVKP</a>	Sets the proportional velocity gain.
<a href="#">SLVKPIF</a>	Provides an Idle Factor to the SLVKP variable

Name	Description
<a href="#">SLVKPSF</a>	Provides a Settle Factor to the SLVKP variable
<a href="#">SLVLI</a>	Integrator velocity limit
<a href="#">SLVRAT</a>	Velocity feed forward ratio

### 3.11.3.1 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.11.3.2 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.11.3.3 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.11.3.4 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.11.3.5 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.11.3.6 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.11.3.7 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.11.3.8 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.11.3.9 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges between 0 to 100, Default = 50.

#### Tag

181

#### Comments

**SLVLI** is expressed as a percentage of the maximum value.

#### Accessibility

Read-Write



#### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

#### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.11.3.10 SLVRAT

#### Description

**SLVRAT** is a real array, with one element for each axis in the system, and is used for defining velocity feed forward ratio.

#### Syntax

**SLVRAT(axis\_index) = value**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.11.4 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.

**3.11.4.1 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.11.4.2 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.11.4.3 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges between 0.05 to 20; Default = 5.

#### Tag

184

#### Accessibility

Read-Write



#### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

#### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.11.5 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
<b>SLVSOF</b>	Sets filter bandwidth
<b>SLVSOFD</b>	Sets filter damping

#### 3.11.5.1 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

<b>axis_index</b>	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 0.1 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

**3.11.5.2 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****SLVSOFD**(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 0.3 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

### 3.11.6 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
<a href="#">SLVBODD</a>	Sets the damping ratio denominator for a Bi-Quad filter.
<a href="#">SLVBODF</a>	Sets the denominator value of the Bi-Quad filter.
<a href="#">SLVBOND</a>	Sets the damping ratio numerator for a Bi-Quad filter.
<a href="#">SLVBONF</a>	Sets the numerator value of the Bi-Quad filter.

#### 3.11.6.1 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 2<sup>nd</sup> order Low-pass and Notch filters for the given axis.

##### Syntax

**SLVBODD(axis\_index) = value**

##### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 2<sup>nd</sup> 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



##### COM Library Methods and .NET Library Methods

ReadVariable, WriteVariable

##### C Library Functions

acsc\_ReadReal, acsc\_WriteReal

### 3.11.6.2 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 2<sup>nd</sup> order Low-pass and Notch filters for the given axis.

#### Syntax

**SLVBODF(axis\_index) = value**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 2<sup>nd</sup> 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

### 3.11.6.3 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 2<sup>nd</sup> order Low-pass and Notch filters for the given axis.

#### Syntax

**SLVBOND(axis\_index) = value**

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 2<sup>nd</sup> 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

**3.11.6.4 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 2<sup>nd</sup> order Low-pass and Notch filters for the given axis.

**Syntax****SLVBONF(axis\_index) = value****Arguments**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 2<sup>nd</sup> 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

**3.11.7 Servo-Loop Position Variables**

The Servo-Loop Position variables are:

Name	Description
<b>SLDRA</b>	Defines disturbance rejection.
<b>SLDRAIF</b>	Provides an Idle Factor to the SLDRA.
<b>SLDRX</b>	Defines the maximum DRA correction for a given axis.
<b>SLPKI</b>	Specifies the position loop integrator coefficient
<b>SLPKIIF</b>	Provides the Idle Factor to the SLPKI (Integrator Gain - Position) variable
<b>SLPKISF</b>	Provides the Settle Factor to the SLPKI (Integrator Gain - Position) variable
<b>SLPLI</b>	Defines the limit for the position of the specified axis
<b>SLPKP</b>	Sets the proportional coefficient of the position for the specified axis.
<b>SLPKPIF</b>	Provides an Idle Factor to the SLPKP variable.
<b>SLPKPSF</b>	Provides an Settle Factor to the SLPKP variable.

**3.11.7.1 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadReal, acsc\_WriteReal

**3.11.7.2 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.11.7.3 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

#### 3.11.7.4 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

#### 3.11.7.5 SLPKIIIF

##### Description

**SLPKIIIF** 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

**SLPKIIIF**(*axis\_index*) = *value*

##### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
-------------------	---

<b>value</b>	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

**3.11.7.6 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

**3.11.7.7 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

<b><i>axis_index</i></b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b><i>value</i></b>	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

## 3.11.7.8 SLPI

### Description

**SLPI** 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

**SLPI**(*axis\_index*) = *value*

## Arguments

<b><i>axis_index</i></b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b><i>value</i></b>	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

### 3.11.7.9 SLPKP

#### Description

**SLKP**P 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

**SLKP**(*axis\_index*) = *value*

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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 **SLKP**P is set to zero, or the integrator is very low.

#### Accessibility

Read-Write



**SLKP**P 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

### 3.11.7.10 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 **SLKP** variable.

#### Syntax

**SLPKPIF** *axis\_index* = *value*

#### Arguments

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
-------------------	---

<b>value</b>	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

**3.11.7.11 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.11.7.12 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	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

### 3.11.8 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
<b>SLAFF</b>	Defines acceleration feed forward for a given axis
<b>SLFRCD</b>	Compensation for dynamic friction

### 3.11.8.1 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.11.8.2 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.11.9 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
<b>SLCROUT</b>	Commutation feedback routing , see <a href="#">Commutation Variables</a>
<b>SLGCAXN</b>	Specifies the complementary gantry axis
<b>SLPROUT</b>	Position feedback routing
<b>SLP2ROUT</b>	Sets the feedback routing of the secondary feedback position
<b>SLVROUT</b>	Velocity feedback routing

**3.11.9.1 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	The <b>value</b> values and the feedback sources associated with them are given in <a href="#">Table 3-19</a> . Default = 0.

**Table 3-19. SLCROUT Values**

SLCROUT	FACC (SPonly)
0	According to <a href="#">E_TYPE</a> 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

SLCROUT	FACC (SPonly)
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**

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

### 3.11.9.2 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

### 3.11.9.3 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 <b>value</b> values and the feedback sources associated with them are given in <a href="#">Table 3-20</a> . 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.

**SLPROUT<sup>1</sup>** 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

### 3.11.9.4 SLP2ROUT

**SLP2ROUT** is variable for setting the feedback routing of the secondary feedback position.

#### Description

**SLP2ROUT** 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

**SLP2ROUT(<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 position
004	From analog input 0
005	From channel 0 resolver position
101	From channel 1 quadrature position or Absolute Encoder position

Value	F2POS
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

**3.11.9.5 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

<b><i>axis_index</i></b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b><i>value</i></b>	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

## 3.11.9.6 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

<b><i>axis_index</i></b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b><i>value</i></b>	The <b><i>value</i></b> values and the feedback sources associated with them are given in <a href="#">Table 3-21</a> . Default = 0.

Table 3-21. SLVROUT Values

SLVROUT	FVEL (SP only)
0	According to <a href="#">E_TYPE</a> velocity
001	From channel 0 quadrature velocity Absolute Encoder velocity

SLVROUT	FVEL (SP only)
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

## 3.12 Commutation Variables

The Servo-Loop Commutation variables are:

Name	Description
SLCHALL	Hall Shift
SLCNP	Number of Poles
SLCPA	Phase Advance
SLCOFFS	Commutation Offset
SLCORG	Commutation Origin
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.

### 3.12.1 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

### 3.12.2 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

### 3.12.3 SLCPA

#### Description

**SLCPA** is a real array, with one element for each axis in the system, and defines the servo-loop commutation phase advance in electrical degrees at **XVEL** (maximum allowed velocity for the motor).

#### Syntax

**SLCPA**(*axis\_index*) = *value***Arguments**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges from 0 to 90, Default=0.

**Tag**

157

**Comments**

The actual phase advance is calculated as:

SLCPA\*VEL/XVEL

where **VEL** is the current reference velocity.

**Accessibility**

Read-Write



SLCPA values cannot be modified if protection is applied to this variable through  
SPiiPlus MMI Application Studio → Toolbox → Application Development → Protection

**Related ACSPL+ Variables**[XVEL](#)**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadReal, acsc\_WriteReal

**3.12.4 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

## 3.12.5 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

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> ranges from: <ul style="list-style-type: none"> <li>● -60 to 60, Default=0</li> <li>● 0 to 360, Default=0</li> </ul>

### 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

**3.12.6 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**

<b>axis_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>value</b>	<b>value</b> 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

**3.12.7 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

**3.12.8 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

**3.13 System Configuration Variables**

The System Configuration variables are:

Name	Description
<b>CFG</b>	Configuration Mode
<b>CTIME</b>	Controller Cycle Time
<b>ECHO</b>	I/O Echo Channel
<b>G_01WCS...G_12WCS</b>	Defines one of the 12 work-piece coordinate systems a user can set as part of the CNC setup/configuration
<b>GPEXL</b>	Indicates the GSP program executed block
<b>GUFAC</b>	Holds the value a conversion factor, from 'Common Physical Units' in [mm] to 'Controller Units'
<b>IENA</b>	Interrupt Enable/Disable
<b>IMASK</b>	Interrupt Mask
<b>ISENA</b>	Interrupt-Specific Enable/Disable
<b>S_FLAGS</b>	System Flags
<b>S_SETUP</b>	System Settings
<b>XSEGAMAX</b>	Maximal processing angles
<b>XSEGAMIN</b>	Minimal processing angles
<b>XSEGRMAX</b>	Maximal arc radius difference
<b>XSEGRMIN</b>	Minimal arc radius

### 3.13.1 CFG

#### Description

**CFG** is an integer variable that indicates the application protection configuration mode.

#### Syntax

**CFG = value**

#### Arguments

**value**

**value** can be one of the following:

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

## 3.13.2 CTIME

**CTIME** is a real variable that defines the controller cycle time.

## Syntax

**CTIME** = *value*

## Arguments

**value**    *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*.

The SPiiPlus NTM product line enables setting **CTIME** to the following values:



- SPiiPlus NTM-16: up to 16 axes, CTIME = 0.5ms (default) or 1ms
- SPiiPlus NTM-32: up to 32 axes, CTIME = 1ms only
- SPiiPlus NTM-32H: up to 32 axes, CTIME = 0.5ms (default) or 1ms
- SPiiPlus NTM-64H: up to 64 axes, CTIME = 1ms only

## Accessibility

Read-Write

### **COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

### **C Library Functions**

acsc\_ReadReal, acsc\_WriteReal

### **3.13.3 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 3-22](#).

**Table 3-22. 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)
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)

Channel Number	Channel Name
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

**3.13.4 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 chooses going back to work in the Machine Coordinate System, he can then choose 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

### 3.13.5 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.

#### Syntax

N/A

#### Arguments

N/A

#### Tag

296

#### Accessibility

Read Only

#### COM Library Methods and .NET Library Methods

ReadVariable

#### C Library Functions

acsc\_ReadInteger

### 3.13.6 GUFACE

#### Description

**GUFACE** 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 GUFACE 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

### 3.13.7 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

**bit\_designator**

The meanings of **bit\_designator** are given in [Table 3-23](#).

**Table 3-23. 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
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 <a href="#">INTERRUPT</a> command
23	Enable Digital input interrupt
24	Enable Motion start interrupt

Bit	Interrupt
25	Enable Motion phase interrupt
26	Enable ACSPL+ interrupt by <b>TRIGGER</b> command

**Tag**

69

**Accessibility**

Read-Write

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadInteger, acsc\_WriteInteger

**3.13.8 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>	<b>IMASK</b> has four bit designators: <ul style="list-style-type: none"> <li>● <b>#IND</b> (bit 0) - Primary encoder index</li> <li>● <b>#IND2</b> (bit 1) - Secondary encoder index</li> <li>● <b>#MARK</b> (bit 2) - Mark1</li> <li>● <b>#MARK2</b> (bit 3) - Mark2</li> </ul>
<i>value</i>	<b>value</b> 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

### 3.13.9 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

<b>array_index</b>	Designates the specific axis, valid numbers are: 0, 1, 2, ... up to the number of axes in the system minus 1.
<b>bit_designator</b>	The meanings of <b>bit_designator</b> are given in <a href="#">Table 3-24</a> .

Table 3-24. 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 <a href="#">INTERRUPT</a> command).

Bit	Interrupt
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

## 3.13.10 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

*bit\_designator*

The meanings of *bit\_designator* are given in [Table 3-25](#).

Table 3-25. S\_FLAGS Bit Description

Bit Name	No.	Description	Remarks
#S_FLAGS	1	<p><b>S_FLAGS.1</b> 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) 1: These lines are each allotted a controller cycle.</p>	Changes to <b>S_FLAGS.1</b> take effect only after a program is recompiled.
#FCLEAR	2	<p>0: Sets the controller to regular mode. (Default) 1: Sets the controller to strict mode</p>	<p>In the regular mode the next motion command simply clears the reason for the previous <b>KILL</b>.</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 <b>KILL</b> is non-zero for any involved motor.</p> <p>Motion can continue only after clearing the <b>MERR</b> variable with <b>ENABLE/ENABLEALL</b> or <b>FCLEAR</b>.</p>

**Tag**

116

**Accessibility**

Read-Write

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadInteger, acsc\_WriteInteger

**3.13.11 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 3-26](#).

**Table 3-26. 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.
#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.
#POSSYNC	7	0 - backward compatibility for FPOS/RPOS synchronization 1 (default) - enable FPOS/RPOS synchronization
#ESDBMODE	8	SPiiPlusES Debug Mode

**Tag**

240

**Accessibility**

Read-Write

**COM Library Methods and .NET Library Methods**

ReadVariable, WriteVariable

**C Library Functions**

acsc\_ReadInteger, acsc\_WriteInteger

**3.13.12 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

### 3.13.13 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

### 3.13.14 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

## 3.13.15 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

## 3.14 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
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 <sup>st</sup> Ethernet
TCPIP2	IP Address for 2 <sup>nd</sup> Ethernet

Name	Description
TCPPORT	TCP port identifier
UDPPORT	UDP port identifier

### 3.14.1 BAUD

#### Description

**BAUD** is an integer variable that defines the serial communication rate, given in bits per second.

#### Syntax

**BAUD** = *value*

#### Arguments

<b>value</b>	<b>value</b> can be one of the following:
	<ul style="list-style-type: none"> <li>● 300</li> <li>● 1200</li> <li>● 4800</li> <li>● 9600</li> <li>● 19200</li> <li>● 57600</li> <li>● 115200</li> </ul>

#### 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

### 3.14.2 COMMCH

#### Description

**COMMCH** is an integer that stores a number representing the last activated communication channel.

**Table 3-27. 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

### 3.14.3 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

**bit\_designator**

The **COMMFL** bits and the meanings of their values are given in [Table 3-28](#).

**Table 3-28. 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
#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

### 3.14.4 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

### 3.14.5 ECHO

#### Description

**ECHO** is a 10-member mask integer variable that defines an echo communication channel.

**Syntax****ECHO** = *channel\_number***Arguments**

<i>channel_number</i>	The values of <b>channel_number</b> and their meanings are given in <a href="#">Table 3-29</a> .
-----------------------	--

**Table 3-29. 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)
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

### **3.14.6 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

*channel\_number*

The values of **channel\_number** and their meanings are given in [Table 3-30](#).

**Table 3-30. 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

Channel Number	Description
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



**DISPCH** 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

**3.14.7 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

**3.14.8 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

## 3.14.9 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)

## 3.14.10 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)

### 3.14.11 TCPPORT

#### Description

**TCPPORT** is a scalar integer that stores a number representing a TCP port.

#### Syntax

**TCPPORT** = *Port\_number*

#### Arguments

**Port\_number**

An integer ranging between 0 and 65536.

#### Tag

200

#### Comments

**TCPPORT** 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 **TCPPORT** 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



**TCPPORT** 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

### 3.14.12 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

### 3.15 Miscellaneous

The Miscellaneous variables are:

Name	Description
<a href="#">FK</a>	Function Key
<a href="#">XARRSIZE</a>	Maximum Array Size

### 3.15.1 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

### 3.15.2 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
- Miscellaneous Functions
- EtherCAT Functions
- CoE Functions
- Servo Processor Functions
- Signal Processing Functions

This chapter covers the ACSPL+ functions.

The ACSPL+ Functions, in alphabetical order, are:

Function	Description
<a href="#">ABS</a>	Calculates the absolute value.
<a href="#">ACOS</a>	Calculates the arc cosine.
<a href="#">ASIN</a>	Calculates the arc sine.
<a href="#">ATAN</a>	Calculates the arctangent.
<a href="#">ATAN2</a>	Calculates the arctangent of X/Y.
<a href="#">AVG</a>	Finds the average of all values in an array.
<a href="#">BCOPY</a>	Copies a given number of bytes from a source array into a target array.
<a href="#">CEIL</a>	Calculates the ceiling of a value.
<a href="#">COPY</a>	The function copies one array to another.
<a href="#">COEREAD/d</a>	Read CoE slave Object Directory entry.
<a href="#">COS</a>	Calculates the cosine.
<a href="#">COEWRITE/d</a>	Write into CoE slave Object Directory.
<a href="#">DEADZONE</a>	Implements dead-zone routine.
<a href="#">DSIGN</a>	Implements a dynamic version of the standard <a href="#">SIGN</a> function.
<a href="#">DSTR</a>	Converts a string to an integer array.
<a href="#">ECIN</a>	Copy EtherCAT offset data into ACSPL+ variable
<a href="#">ECGETGRPIND</a>	Returns an array that contains optional groups' indexes that are part of the current configuration

Function	Description
<b>ECGETOPTGRP</b>	Returns number of actually connected optional groups, not including the mandatory group.
<b>ECGETSLAVES</b>	Returns the number of EtherCAT slaves in the network
<b>ECGRPINFO</b>	<ul style="list-style-type: none"> <li>● Fills array with nodes' indexes which are members of a given optional group</li> <li>● Returns the number of members in the group.</li> </ul>
<b>ECOUT</b>	Copy data of ACSPL+ variable into EtherCAT offset
<b>ECSAVECFG</b>	Saves to flash an array of optional groups based on current configuration, based on ecgetgrpind() function.
<b>ECSAVEDCNF</b>	Returns array that contains optional groups' indexes that are part of the last saved configuration
<b>ECUNMAP</b>	Undoes any results from running <b>ECIN</b> and <b>ECOUT</b>
<b>ECUNMAPIN</b>	Undoes any results from running <b>ECIN</b> to a specific offset.
<b>ECUNMAPOUT</b>	Undoes any results from running <b>ECOUT</b> to a specific offset.
<b>EDGE</b>	Returns 1 on positive edge of x.
<b>EXP</b>	Calculates the exponent.
<b>FLOOR</b>	Calculates the floor of a value.
<b>GETCONF</b>	Reads hardware and firmware parameters.
<b>GETSP</b>	Reads a value from the specified SP address.
<b>GETSPA</b>	Retrieves address of the SP variable specified by name.
<b>GETSPV</b>	Reads a value from the specified SP variable name.
<b>HYPOT</b>	Calculates the hypotenuse.
<b>INP</b>	Reads data characters from the specified channel and stores them into integer array.
<b>INTGR</b>	Implements an integrator with <b>DEADZONE</b> and <b>SAT</b> .
<b>LAG</b>	Provides delayed switching on argument change (anti-bouncing effect).
<b>LDEXP</b>	Calculates a value of $x \cdot 2^{\text{exp}}$ .

Function	Description
<b>LOG</b>	Calculates the natural logarithm.
<b>LOG10</b>	Calculates the base-10 logarithm.
<b>MAP</b>	Implements a table-defined function with constant step.
<b>MAPB</b>	One-dimensional uniform b-spline.
<b>MAPN</b>	One-dimensional non-uniform linear interpolation (replaces obsolete <b>MAPBY1</b> and <b>MAPBY2</b> ).
<b>MAPNB</b>	One-dimensional non-uniform B-spline.
<b>MAPNS</b>	One-dimensional non-uniform Catmull-Rom spline.
<b>MAPS</b>	One-dimensional uniform Catmull-Rom spline.
<b>MAP2</b>	Implements a table-defined function with two arguments and constant step along each argument.
<b>MAP2B</b>	Two-dimensional uniform B-spline.
<b>MAP2N</b>	Two-dimensional non-uniform linear interpolation (replaces obsolete <b>MAP2FREE</b> ).
<b>MAP2NB</b>	Two-dimensional non-uniform B-spline.
<b>MAP2NS</b>	Two-dimensional non-uniform Catmull-Rom spline.
<b>MAP2S</b>	Two-dimensional uniform Catmull-Rom spline.
<b>MATCH</b>	Calculates axis position that matches current reference position of the same axis with zero offset.
<b>MAX</b>	Finds the maximum value in an array.
<b>MAXI</b>	Finds the element with maximum value in an array or in a section of an array and returns its index.
<b>MIN</b>	Finds the minimum value in an array.
<b>MINI</b>	Finds the element with minimum value in an array or in a section of an array and returns its index.
<b>NUMTOSTR</b>	Converts a number to an ASCII string.
<b>POW</b>	Calculates x raised to the power of y.

Function	Description
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.
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

Function	Description
COS	Calculates the cosine
EXP	Calculates the exponent
FLOOR	Calculates the floor of a value
HYPOT	Calculates the hypotenuse
LDEXP	Calculates a value of $x \cdot 2^{\text{exp}}$
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

#### 4.1.1 ABS

##### Description

**ABS** calculates the absolute value

##### Syntax

**ABS**(*input*)

##### Arguments

<b>input</b>	<b>input</b> 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 p 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 -p/2 to p/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  $-p/2$  to  $p/2$  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  $-p$  to  $p$  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 <sup>3</sup> **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<sup>input</sup>

### 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  $x \cdot 2^y$ .

### Syntax

**LDEXP**(*X\_input, Y\_input*)

### Arguments

<i>X_input</i>	<i>X_input</i> can be a real number or an expression.
----------------	---

<i>Y_input</i>	<i>Y_input</i> can be a real number or an expression.
----------------	---

### Return Value

Real number - returns the value of  $X_{\text{input}} \cdot 2^{Y_{\text{input}}}$ . 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

<i>input</i>	<i>input</i> 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

<b><i>input</i></b>	<b><i>input</i></b> 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

<b><i>X_input</i></b>	<b><i>X_input</i></b> can be a real number or an expression.
<b><i>Y_input</i></b>	<b><i>Y_input</i></b> can be a real number or an expression.

#### Return Value

Real number - returns the value of  $(X_{\text{input}})^{Y_{\text{input}}}$ .

#### 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

**input**    **input** 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

**input**    **input** 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  $\geq 0$ , 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
```

## 4.2 Miscellaneous Functions

The Miscellaneous functions are:

Function	Description
<b>GETCONF</b>	Reads hardware and firmware parameters.

Function	Description
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 <b>tag</b> .
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.

#### 4.2.1 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	Returns the mask that determines, for each digital input, whether the leading or trailing signal edge triggers an interrupt.

Key	Value - Bit and Explanation
	<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 <b>IN0</b>.</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>
29	<p>Returns the current assignment of the brake output associated with the specified axis.</p> <p>-1: Output follows the corresponding bit of OUT0</p> <p>AANNOO: Brake output supplies the brake control signal where:</p> <ul style="list-style-type: none"> <li>● AA - is the axis designation (00-99)</li> <li>● NN - digital output index (00-99)</li> <li>● OO - specific output (00-99)</li> </ul>
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 <b>IN0</b>.</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 <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 0)</a> or <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping PEG Engines to Encoders (Servo Processor 1)</a> for SPiiPlusNT/DC-LT/HP/LD-x, or <a href="#">SPiiPlus CMnt-x/UDMpm-x/UDMpc-x/CMba-x/CMhp-x/UDMba-x/UDMhp-x/CMhv-x/UDMhv-x/UDMnt-x Mapping PEG Engines to Encoders (Servo Processor 0)</a> 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> <p>Returns the bit code according to <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)</a> or <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 1)</a> for SPiiPlusNT/DC-LT/HP/LD-x, or <a href="#">SPiiPlus CMnt-x/UDMpm-x/CMhv-x/UDMhv-x General Purpose Outputs Assignment for Use as PEG Pulse Outputs (Servo Processor 0)</a> for SPiiPlus CMnt-x-320/UDMpm-x-320, depending on the axis.</p>
73	Used to view the actual output pins assignment for PEG engines.

Key	Value - Bit and Explanation
	Returns the bit code according to <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> or <a href="#">SPiiPlusNT/DC-LT/HP/LD-x/SPiiPlus SAnt-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 1)</a> , for SPiiPlusNT/DC-LT/HP/LD-x, or <a href="#">CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> or <a href="#">CMnt-x/UDMpm-x/UDMpc-x/CMhv-x/UDMhv-x Mapping of Engine Outputs to Physical Outputs (Servo Processor 0)</a> for SPiiPlus CMnt-x-320/UDMpm-x-320, depending on the axis.
74	<p>Returns the saved EtherCAT topology configuration</p> <p>Index = 1: EtherCAT topology mode that is saved on the controller's non-volatile memory:</p> <ul style="list-style-type: none"> <li>0: line topology mode</li> <li>1: ring topology mode</li> <li>2: two lines topology mode</li> </ul> <p>Index =2 : Number of nodes connected to the EtherCAT master's main line that is saved on the controller's non-volatile memory</p> <p>Index =3 : Number of nodes connected to the EtherCAT master's redundant line that is saved on the controller's non-volatile memory.</p>
76	<p>Returns the the System or MPU temperature (in degrees °C):</p> <ul style="list-style-type: none"> <li>0: Current System temperature</li> <li>1: Current MPU temperature</li> </ul>
78	<p>Returns the status if fast loading of Random PEG arrays for the relevant Servo Processor is activated or deactivated:</p> <ul style="list-style-type: none"> <li>0: fast loading of Random PEG arrays is deactivated</li> <li>1: fast loading of Random PEG arrays is activated.</li> </ul> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>Index is the axis of the relevant Servo Processor: 0, 1, 2, ..., up to total number of axes in system minus 1.</p> </div>
79	Returns the maximum <b>USAGE</b> value since power-up or since last call to the <b>SETCONF</b> (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.
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.

Key	Value - Bit and Explanation
	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 <a href="#">MFLAGS.1</a> (Open Loop)  1: Open loop  0: Not open loop
204	Returns the value of <a href="#">MFLAGS.9</a> (Commutation OK).  1: Commutation OK  0: Commutation not OK)
214	Only valid for brushless motors ( <a href="#">MFLAGS.8</a> = 1).  Returns the commutation phase (degrees) at the current point.
216	Only valid for brushless motors ( <a href="#">MFLAGS.8</a> = 1).  Returns the commutation state ( <a href="#">MFLAGS.9</a> ):  0: Commutation is not OK (not initialized)  1: Commutation is OK.
229	Returns the mechanical brake output:  1: Mechanical brake is inactive  0: Mechanical brake is active
246	Upon receipt of a Drive Alarm signal, the controller stores a general Drive Alarm code (5019) in the <a href="#">MERR</a> variable. The extended Drive Fault status code can be obtained by executing <a href="#">GETCONF(246, Axis)</a> .  The following extended Drive Fault statuses are supported (the <a href="#">MERR</a> code appears in brackets) by the DDM3U Motor Drive:  ● Drive Alarm (5060)

Key	Value - Bit and Explanation
	<ul style="list-style-type: none"> <li>● Drive Alarm: Short circuit (5061)</li> <li>● Drive Alarm: External Protection Activated (5062)</li> <li>● Drive Alarm: Power Supply Too Low (5063)</li> <li>● Drive Alarm: Power supply too high (5064)</li> <li>● Drive Alarm: Temperature too high (5065)</li> <li>● Drive Alarm: Power Supply 24VF1 (5066)</li> <li>● Drive Alarm: Power Supply 24VF2 (5067)</li> <li>● Drive Alarm: Emergency Stop (5068)</li> <li>● Drive Alarm: Power down (5069)</li> <li>● Drive Alarm: Phase lost. (5070)</li> <li>● Drive Alarm: Drive not ready (5071)</li> <li>● Drive Alarm: Over current (5072)</li> <li>● Not in use (reserved) (5073)</li> <li>● Drive Alarm: Damper is not ok (5074)</li> <li>● Drive Alarm: Digital Drive Interface not Connected (5075)</li> </ul> <p>Using the <b>GETCONF</b> function, the faults 5064, 5065, 5069, 5071 can be read before the <b>ENABLE</b> command is executed.</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;">  The <b>GETCONF</b> function provides a delay until the extended Drive Fault is received. This behavior differs from the implementation in SPiiPlus PCI-DDM4 and SPiiPlus CM, where the extended Drive Fault status is stored in MERR immediately upon receipt of the Drive Alarm signal.     </div>
253	Returns the STO signal 0: STO1l 1: STO2l
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	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: !!!Original homing procedure here ... TILL IST(AXIS).#IND; !Index was found

Key	Value - Bit and Explanation
	<p>! Move to a middle of a quadrant, close to the index location  <math>\text{PTP}(\text{AXIS}), \text{IND}(\text{AXIS}) + \text{POW}(2, (\text{E\_SCMUL}(\text{AXIS}) - 3)) * \text{EFAC}(\text{AXIS})</math></p> <p>TILL ^MST(AXIS).#MOVE</p> <p>WAIT 1000</p> <p>! Repeatability correction  <math>\text{SET FPOS}(\text{AXIS}) = \text{FPOS}(\text{AXIS}) - \text{IND}(\text{AXIS}) - \text{GETCONF}(265, \text{AXIS})</math></p> <p>PTP(AXIS), 0 ! 0 = index location</p>
301	Returns the size of the segment queue in segmented motions ( <a href="#">MPTP...ENDS</a> , <a href="#">MSEG...ENDS</a> , <a href="#">PATH...ENDS</a> , <a href="#">PVSPINE...ENDS</a> ).
310	Returns an integer value that contains the TCP/IP address currently assigned to the controller. The <b>index</b> argument has to be zero, for example, $\text{GETCONF}(310, 0)$ If a TCP/IP protocol is not configured, or not supported, the return value is zero.
312	<p>Returns the RAM load in percentage, the amount of total physical memory, or the amount of free physical memory as:</p> <p>0 – memory load in percentage  1 – amount of total physical memory  2 – amount of free physical memory</p>

## 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 10.0.0.100

## 4.2.2 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

<b>Int</b>	<b>Value Returned</b>
1	The SPiiPlus model number. For all EtherCAT products, returns 60. A returned value of $\leq 0$ 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

Int	Value Returned
15	Number of data collection channels per DSP.
16	EtherCAT support: 1 - Yes 0 - No

### COM Library Methods

None

### C Library Functions

acsc\_SysInfo

### 4.2.3 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

<b>Tag</b>	The variable Tag number (positive integer)
<b>Index1,</b> <b>Index2</b>	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

DISP GETVAR (1001,1,1)                                         !described in Tag 1001 cell (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.

#### 4.2.4 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

<b>key</b>	Specifies the configured feature.
<b>index</b>	Specifies axis or buffer number.
<b>value</b>	<p>The set of bit states for the defined key. value is set up according to a 16-bit binary template, illustrated in <a href="#">16-bit Binary value Template</a>. The controller strips all leading zeros. The controller understands value in binary, hexadecimal, or decimal format.</p> <p>The following prefixes determine value format:</p> <ul style="list-style-type: none"> <li>OB - binary</li> <li>OH - hexadecimal</li> </ul> <p>A decimal value does not require any prefix.</p>

**Table 4-3. 16-bit Binary value Template**

Bit	3 1	3 0	2 9	2 8	2 7	..	6	5	4	3	2	1	0
Prefix [OB]   [OH] value	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	Sets the mask that determines, for each digital input, whether the leading or trailing signal edge triggers an interrupt.

Key	Index	Value - Bit and Explanation
		<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 <b>IN0</b>.</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>
29	A six digit number: AANNOO, where:	<ul style="list-style-type: none"> <li>● AA - is the axis designation (00-99)</li> <li>● NN - digital output index (00-99)</li> <li>● OO - specific output (00-99)</li> </ul> <p>The following decimal values assign the specified output bit to one of the following functions according to the <i>value</i> argument:</p> <p>2: Output supplies the brake control signal 0: Cancel assignment for the axis, that is, output reverts to digital output</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 <b>IN0</b>.</p> <p>1: The controller generates an interrupt on both edges. 0: The controller generates an interrupt on one edge.</p> <p>After power-up the mask contains 0 in all bits.</p>
79		Clears the value set as the maximum <b>USAGE</b> value.
86		Resets the counter of allowed single EtherCAT frames that were actually lost
203	Axis:	<p>Sets the value of <b>MFLAGS</b>.1 (Open Loop)</p> <p>1: Open loop</p>

Key	Index	Value - Bit and Explanation
	0, 1, 2, ..., up to total number of axes in system minus 1	0: Not open loop
204	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	Sets the value of <b>MFLAGS.9</b> (Commutation OK). 1: Commutation OK 0: Commutation not OK
214	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	Valid only for brushless motors ( <b>MFLAGS.8</b> = 1). <b>SETCONF</b> with the 214 key sets the commutation offset at the current position to the specified <b>value</b> (in degrees). There are, however, two cases that have to be considered when using <b>SETCONF</b> : <ul style="list-style-type: none"><li>● Motor Not Commutated (<b>MFLAGS.9</b>=0) If the motor has not yet been commutated, <b>SETCONF</b> sets the commutation offset at the current position to the value specified by the <b>value</b> argument.</li><li>● Motor Commutated (<b>MFLAGS.9</b>=1) The behavior of <b>SETCONF</b> 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°. <b>SETCONF</b> 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 <b>?GETCONF(214, axis)</b> in the <b>Communication Terminal</b> and subtract 90 from the returned value. From this the user can calculate what the value of the <b>value</b> argument should be to get the proper phase.</li></ul>
216	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	Valid only for brushless motors ( <b>MFLAGS.8</b> = 1). If value = 1: <ul style="list-style-type: none"><li>● <b>MFLAGS.9</b>=0 (Commutation is not OK)</li><li>● <b>MFLAGS</b> bits <b>1, 4, 5, 6</b> are set to zero</li><li>● <b>MFLAGS.8</b> is set to 1</li><li>● <b>DCOM</b> is reset to zero</li><li>● Sets <b>RPOS</b> = <b>FPOS</b></li></ul> If value = 0:

Key	Index	Value - Bit and Explanation
		<ul style="list-style-type: none"> <li>● <b>MFLAGS.9=0</b> (Commutation is not OK)</li> <li>● <b>MFLAGS</b> bits <b>1, 4, 5, 6</b> are set to zero</li> <li>● <b>MFLAGS.8</b> is set to 10</li> <li>● Variable <b>DCOM</b> is reset to zero.</li> </ul>
217	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	Valid only for brushless motors ( <b>MFLAGS.8 = 1</b> ) where the encoder has encountered the index and <b>IND</b> contains a valid value.  <b>value:</b> Not Relevant. <ul style="list-style-type: none"> <li>● Adjusts the commutation offset so that the commutation phase at the last index point is equal to the specified value in degrees.</li> <li>● <b>MFLAGS.9=0</b> (Commutation is not OK)</li> <li>● <b>MFLAGS</b> bits <b>1, 4, 5, 6</b> are set to zero</li> <li>● <b>MFLAGS.8</b> is set to 1</li> <li>● <b>DCOM</b> is reset to zero</li> <li>● Sets <b>RPOS = FPOS</b></li> </ul>
229	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	Mechanical brake output: 1: Deactivates mechanical brake output. 0: Activates mechanical brake output. The motor must be disabled to execute this setting.
230	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	Safety system configuration - 1: Enable safety system configuration 0: Disable safety system configuration  After power-up, the safety system configuration is disabled. The user must enable the safety system configuration for a motor before executing any <b>SETCONF</b> function with from key 231 to key 236.  An index designation of -1 selects all axes.
246	Axis: 0, 1, 2, ..., up to total number of axes in system minus 1	<b>SETCONF(246, Axis, 0)</b> is used to clear the fault status on all axes that relate to the DDM3U Motor Drive that handles the specified axis. <div style="border: 1px solid black; padding: 10px; margin-top: 10px;">  The <b>FCLEAR</b> command does not clear the fault status in the MC4U, <b>SETCONF(246, Axis, 0)</b> has to be used instead.         </div>

Key	Index	Value - Bit and Explanation
249	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><b>SETCONF(249, Axis, 0)</b> disables the automatic current bias measurement for the given axis.</p>
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> <ul style="list-style-type: none"> <li>0: for pairs (0,1) and (2,3)</li> <li>1: for pairs (0,2) and (1,3)</li> </ul> <p>For example:</p> <p>4 axes MC4U system</p> <ul style="list-style-type: none"> <li>Command for setting pairs (0,1) and (2,3) - SETCONF (267,0,0)</li> <li>Command for setting pairs (0,2) and (1,3) - SETCONF (267,0,1)</li> </ul> <p>8-axes MC4U system</p> <ul style="list-style-type: none"> <li>Command for setting pairs (0,1) and (2,3) - SETCONF (267,0,0)</li> <li>Command for setting pairs (0,2) and (1,3) - SETCONF (267,0,1)</li> <li>Command for setting pairs (4,5) and (6,7) - SETCONF (267,4,0)</li> <li>Command for setting pairs (4,6) and (5,7) - SETCONF (267,4,1)</li> </ul> <p>Two 8-axes MC4U systems connected in the network</p> <ul style="list-style-type: none"> <li>Command for setting pairs (0,1) and (2,3) - SETCONF (267,0,0)</li> <li>Command for setting pairs (0,2) and (1,3) - SETCONF (267,0,1)</li> <li>Command for setting pairs (4,5) and (6,7) - SETCONF</li> </ul>

Key	Index	Value - Bit and Explanation
		(267,4,0) Command for setting pairs (4,6) and (5,7) - SETCONF(267,4,1) Command for setting pairs (8,9) and (10,11) - SETCONF(267,8,0) Command for setting pairs (8,10) and (9,11) - SETCONF(267,8,1) Command for setting pairs (12,13) and (14,15) - SETCONF(267,12,0) Command for setting pairs (12,14) and (13,15) - SETCONF(267,12,1)
302	2 1	The following decimal values specify a communication channel for special input:  3: Set the channel to <b>Modbus</b> Master mode. 2: Set the channel to <b>Modbus</b> Slave mode 1: Assigns the channel for special input. 0: Set the channel to regular command processing mode (default channel mode).  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 <b>DISP</b> and <b>SEND</b> commands.  index specifies the channel.
303	2 1	The <b>value</b> sets the baud rate for the specified serial channel, where the baud rate is the decimal value. 115200 (default), 57600, 19200, 9600, 4800, 2400, 1200, 600, 300.  <b>index</b> specifies the channel.
304	2 1	Sets communication options:  Bit 2 - 1: extended stop bit 0: normal stop bit  Bit 3 - 1: check parity 0: no parity  Bit 4 - 1: even parity

Key	Index	Value - Bit and Explanation
		0: odd parity <b>index</b> specifies the channel.
308	19 18 17	When the controller acts as a <b>Modbus</b> Master, the function establishes or closes the <b>Modbus</b> 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. The slaveIP value specifies the <b>Modbus</b> 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). If the specified channel is already open, the function closes the opened channel and then opens new one. If slaveIP is zero, the function closes the TCP connection.
309	Don't Care	Defines the sequence for the two 16-bit <b>Modbus</b> interface registers - 1: Low word will be first, then high word 0: High word will be first, then low word (default configuration)
310	0	Used for providing access to the controller TCP/IP address. Where: 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). If <b>value</b> is zero, <b>SETCONF</b> 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). <b>SETCONF</b> does not change the <b>TCPIP</b> variable. After power-up, the controller is initialized with the TCP/IP address set in the <b>TCPIP</b> variable. There are several limitations when using <b>SETCONF</b> (310): <ul style="list-style-type: none"><li>● If the <b>TCPIP</b> variable stored in the flash is zero, <b>SETCONF</b>(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.</li><li>● If the <b>TCPIP</b> variable stored in the flash is not zero,</li></ul>

Key	Index	Value - Bit and Explanation
		<p><b>SETCONF(310)</b> 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.</p> <ul style="list-style-type: none"> <li>● <b>SETCONF(310)</b> has a long execution time. During this time, communication with the controller is impossible using any communication channel. Use <b>SETCONF(310)</b> only within the controller initialization sequence. Avoid attempts to communicate with the controller and the motor <b>ENABLE</b> command or motion commands while <b>SETCONF(310)</b> is in progress.</li> </ul>

#### Return Value

None

#### COM Library Methods

SetConf

#### C Library Functions

acsc\_SetConf

#### Example

The following example illustrates setting the General Purpose digital output 1 bit 1 as mechanical brake of axis 4:

```
SETCONF(29, 040101,2)
```

## 4.2.5 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

<b>value</b>	A real or integer value assigned to the variable.
<b>Tag</b>	The variable Tag number (positive integer).
<b>Index1, Index2</b>	If the variable is an array, the indexes point to the specific location in the array. If the variable is <b>scalar</b> , 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.

## 4.2.6 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

<b>array_name</b>	Name of user-defined integer array.
<b>start_index</b>	Index in the array from which to start converting.
<b>number</b>	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"

### 4.2.7 STRTONUM

#### Description

This function converts ASCII encoded element string to a number.

#### Syntax

`double STRTONUM(Source, Type, From, N)`

#### Arguments

<b>Source</b>	An integer array
<b>Type</b>	Designates the format of the converted number, it can be: <ul style="list-style-type: none"> <li>● 0 - decimal</li> <li>● 1- hex</li> <li>● 2 - floating point</li> </ul>
<b>From</b>	Index of the first element in the array that may contain the number
<b>N</b>	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

```

### 4.2.8 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

<b>Number</b>	The number to be converted
<b>Target</b>	An integer array used in which to put the results
<b>Type</b>	Designates the format of the number to be converted, it can be: <ul style="list-style-type: none"> <li>● 0 - decimal</li> <li>● 1- hex</li> <li>● 2 - floating point</li> </ul>
<b>From</b>	Index of the first element in the array that may contain the number
<b>N</b>	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

```

## 4.2.9 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

<b>Source_array</b>	Array containing input data
<b>Target_array</b>	Array containing output data
<b>CopyBytes</b>	Number specifying how many bytes will each copy operation
<b>S_SkipBytes</b>	Number specifying how many bytes will be skipped in the source array following each copy operation
<b>T_SkipBytes</b>	Number specifying how many bytes will be skipped in the target array following each copy operation
<b>From</b>	Index of the first element in the source array that may be used
<b>N</b>	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 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.

### 4.3.1 AVG

#### Description

**AVG** finds the average of all values in an array.

#### Syntax

**AVG(array\_name)**

#### Arguments

<b>array_name</b>	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
```

### 4.3.2 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

<b>source</b>	The name of an array that has been declared in the program from which the data is to be copied.
<b>destination</b>	The name of an array that has been declared in the program to which the data is to be copied.
<b>from_source_row</b>	The index of the first row of the source to begin copying.
<b>to_source_row</b>	The index of the last row of the source to end copying.
<b>from_source_col</b>	The index of the first column of the source to begin copying. Used only for matrix type arrays, otherwise it can be omitted.
<b>to_source_col</b>	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
                                                       ! 4 5 6
                                                       ! 7 8 9
                                                       ! Destination:
                                                       ! 0 0 0
                                                       ! 1 2 3
                                                       ! 4 5 6
                                                       ! Ends program
STOP

```

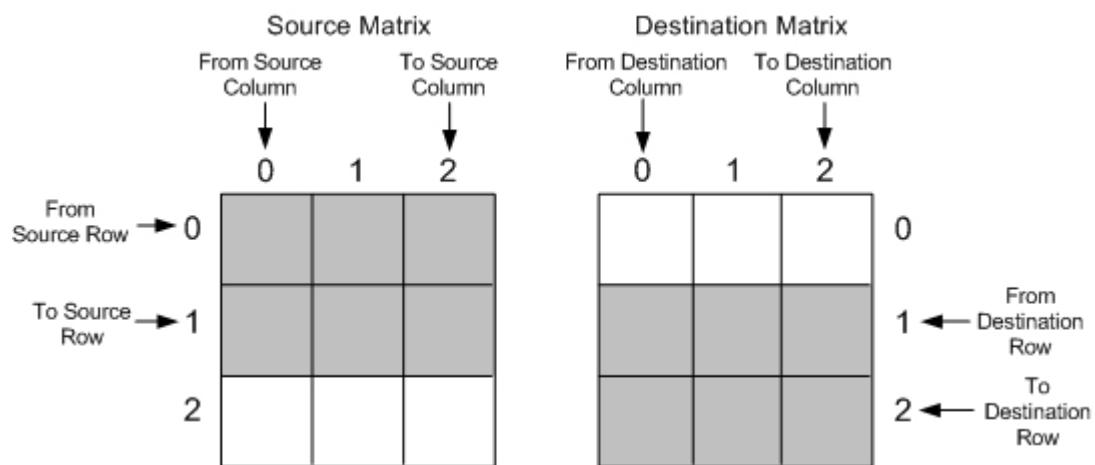


Figure 4-1. Illustration of COPY Function

### 4.3.3 DSHIFT

#### Description

DSHIFT shifts all elements of the array to one position left.

#### Syntax

`real DSHIFT(array, value, index)`

#### Arguments

<b>array</b>	An array of real with the maximal size of 10,000 elements
<b>value</b>	The real value to be inserted to the array. The Value is inserted to position of the element with the index Index.
<b>index</b>	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

```

### 4.3.4 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

<b>real value</b>	Any real number.
<b>destination</b>	The name of an array that has been declared in the program to which the value is to be copied.
<b>from_array_row</b>	The index of the first row of the destination array to begin filling.
<b>to_array_row</b>	The index of the last row of the destination array to end filling.
<b>from_array_column</b>	The index of the first column of the destination array to begin filling. Used only for matrix type arrays, otherwise it can be omitted.
<b>to_array_column</b>	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
```

## 4.3.5 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

<b>array_name</b>	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
```

## 4.3.6 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

<b>array_name</b>	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
```

## 4.3.7 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

<b>array_name</b>	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
```

### 4.3.8 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

<b>array_name</b>	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 EtherCAT Functions

EtherCAT functions include:

<b>COEGETSIZE</b>	Returns the size, in bits, of a specific entry in the object dictionary of a specific slave.
<b>ECCLOSEPORT</b>	The function closes the specified port of specified EtherCAT node.
<b>ECCLRREG</b>	Clears the contents of error counters registers.
<b>ECEXTIN</b>	Used for mapping input variables (TxPDO) to non-ACS EtherCAT network.
<b>ECEXTOUT</b>	Used for mapping output variables (RxPDO) from non-ACS EtherCAT network to the SPiiPlusES.
<b>ECGETGRPIND</b>	The function returns an array that contains optional groups' indexes that are part of the current configuration (including mandatory group, which is 0).
<b>ECGETPID</b>	Returns product ID of the node.

<b>ECGETMAIN</b>	The function returns the number of EtherCAT slaves connected to the main EtherCAT line.
<b>ECGETOFFSET</b>	The function returns offset of the specified variable of the specific EtherCAT node.
<b>ECGETOPTGRP</b>	The function returns number of actually connected optional groups, not including the mandatory group.
<b>ECGETRED</b>	The function returns the number of EtherCAT slaves connected to the redundant EtherCAT line.
<b>ECGETREG</b>	Gets the contents of ESCs error counters registers.
<b>ECGETSLAVES</b>	Returns the number of slaves in an EtherCAT network.
<b>ECGETSTATE</b>	Returns the state of the node.
<b>ECGETVID</b>	Returns vendor ID of the node.
<b>ECGRPINFO</b>	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.
<b>ECIN</b>	Copies the EtherCAT network input variable to an ACSPL+ variable.
<b>ECOUT</b>	Copies ACSPL+ variable to EtherCAT network output variable.
<b>ECRESCAN</b>	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.
<b>ECREPAIR</b>	Returns the system back to the operational state if one or more slaves underwent a reset or power cycle.
<b>ECSAVECFG</b>	Saves the current network topology configuration into the non-volatile memory.
<b>ECSAVEDCNF</b>	The function returns array that contains optional groups' indexes that are part of the last saved configuration (including mandatory group, which is 0).
<b>ECUNMAP</b>	Resets mapping of <b>ECIN</b> and <b>ECOUT</b> .
<b>ECUNMAPIN</b>	Resets mapping of <b>ECIN</b> of a specified offset.
<b>ECUNMAPOUT</b>	Resets mapping of <b>ECOUT</b> of a specified offset.

#### 4.4.1 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

<b>Slave</b>	An integer representing slave number, starting from 0
<b>Index</b>	An integer representing index in the object dictionary
<b>Subindex</b>	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.

## 4.4.2 ECCLOSEPORT

### Description

The function closes the specified port of specified EtherCAT node.

### Syntax

Int **ECCLOSEPORT**(*name, index*)

## Arguments

<b>Index</b>	EtherCAT node index.
<b>Port</b>	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

```
ECCLOSEPORT(<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.

#### 4.4.3 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)**.

#### 4.4.4 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
```

## 4.4.5 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
```

#### 4.4.6 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.
```

#### 4.4.7 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.

#### 4.4.8 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

#### 4.4.9 ECGETOFFSET

##### Description

The function returns offset of the specified variable of the specific EtherCAT node.

##### Syntax

Int ECGETOFFSET(*name, index, InOut(optional), occurrence(optional)*)

##### Arguments

<b>Name</b>	Name of variable as shown in System Setup.
<b>Index</b>	EtherCAT node index.
<b>InOut (optional)</b>	Can be Output (=0) or Input (=1). By default, the variable is being searched in Inputs and, if not found, in Outputs.
<b>Occurrence (optional)</b>	By default the value is 0.

##### 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)
!Returns offset of network variable (PDO) "Channel 1 Data", slave 0,
Input, second occurrence
```

#### 4.4.10 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
```

#### 4.4.11 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.

#### 4.4.12 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**

<b>Index</b>	EtherCAT slave index.
<b>Offset</b>	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"> <li>● Invalid Frame Counter: 0x300/2/4/6</li> <li>● RX Error Counter: 0x301/3/5/7</li> </ul>
0x302	Port Error Counter (CRC B)	

Offset	Name	Description
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)

#### 4.4.13 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.

#### 4.4.14 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.

#### 4.4.15 ECGETVID

##### Description

ECGETVID returns the vendor ID of the node.

##### Syntax

**ECGETVID(index)**

##### Arguments

**Index** EtherCAT slave index, starting from 0.

##### Return Value

The ACS vendor ID of the node.

#### 4.4.16 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

<b>group_index</b>	Optional Group Index, starting from 0 (0 is the mandatory group)
<b>nodes_array</b>	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}
```

### 4.4.17 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(int offset, Varname)**

#### Arguments

<b>offset</b>	Internal EtherCAT offset of network variable derived from the SPiiPlus MMI Application Studio <b>Communication Terminal ETHERCAT</b> command.
<b>Varname</b>	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.

#### 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
```

#### 4.4.18 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[/r] (int offset, Varname)**

### Arguments

<b>offset</b>	Internal EtherCAT offset of network variable derived from the SPiiPlus MMI Application Studio <b>Communication Terminal</b> <b>ETHERCAT</b> command.
<b>Varname</b>	Valid name of ACSPL+ variable, global or standard.

### Command Options

<b>/r</b>	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.

### 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

```

## 4.4.19 ECREPAIR

### Description

**ECPREPAIR** 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

**ECPREPAIR**

### Comments

**ECPREPAIR** 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

**ECPREPAIR** can take a long time to complete; therefore it is recommended calling **ECPREPAIR** from a Program Buffer. It is possible to execute the **ECPREPAIR** 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 ECPREPAIR 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 **ECPREPAIR** operation was successful.

EtherCAT slaves that were operational before **ECPREPAIR** activation will keep their feedback and commutation valid.

#### 4.4.20 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**.

#### 4.4.21 ECSAVECFG

##### Description

The command saves to flash an array of optional groups based on current configuration, based on egcgetgrpind() 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:

ecsavectfg

#### 4.4.22 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)
ecsavectfg(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.
```

#### 4.4.23 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

#### 4.4.24 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.

**4.4.25 ECUNMAPOUT****Description**

This function is used to reset all previous mapping defined by **ECOUT** to a specific offset.

**Syntax****ECUNMAPOUT(ECOffset)****Arguments****ECOffset**

An integer providing the offset to which a variable has been mapped by **ECOUT**.

**Return Value**

None

**Example**

Assuming previous execution of **ECOUT(48,I0)** and **ECOUT(50,I1)**, executing **ECUNMAPOUT(48)** will unmap only I0.

**4.5 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
<b>COEREAD/d</b>	Read CoE slave Object Directory entry.

Function	Description
<a href="#">COEWRITE/d</a>	Write into CoE slave Object Directory.
<a href="#">COEGETSIZE</a>	Returns the size, in bits, of a specific entry in the object dictionary of a specific slave.

### 4.5.1 COEREAD/d

#### Description

This function is used to read Object Dictionary entry from CoE slave.

This function with the "/d" suffix 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

#### Syntax

**COEREAD/d[/size] (int slave,int Index,int Subindex)**

#### Arguments

<b>size</b>	1/2/4 number of bytes in the Object Dictionary /f for floating (32 bit) or /d for double (64 bit)
<b>slave</b>	Slave number (which can be obtained by running the SPiiPlus MMI Application Studio <b>Communication Terminal</b> <a href="#">ETHERCAT</a> command). -1 means "SPiiPlusES"
<b>Index</b>	Index in the Object Dictionary.
<b>Subindex</b>	Sub-index in the Object Dictionary.

#### Return Value

Returns the value stored in the variable in the specified **Index** of the Object Dictionary.

#### 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.

## 4.5.2 COEWRITE/d

### Description

This function is used to write a value into the CoE slave Object Dictionary.

This function with the "/d" suffix is an extension to the existing coeread function, and provides the capability of reading a double type (64 bit).

### Syntax

**COEWRITE/d[/size] (int slave,int Index,int Subindex,double Value)**

### Arguments

<b>size</b>	1, 2 or 4 - the number of bytes in the Object Dictionary /f for floating.(32 bit) or /d for double (64 bit)
<b>slave</b>	Slave number (which can be obtained by running the SPiiPlus MMI Application Studio <b>Communication Terminal</b> <b>ETHERCAT</b> command)
<b>Index</b>	Index in the Object Dictionary.
<b>Subindex</b>	Sub-index in the Object Dictionary.
<b>Value</b>	The value to be written.

### 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.

## 4.6 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 SPiiPlus 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

### 4.6.1 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

<b>SP</b>	The SP number in the system. Use getconf(260, axis), where axis is axis in the system to get the SP number.
<b>Address</b>	Address within the SP. Get the address by using getspa(SPnumber, varname) 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
```

## 4.6.2 GETSPA

#### Description

**GETSPA** retrieves the SP address of the specified SP variable.

#### Syntax

**int GETSPA(*SP\_number, "SP\_variable*)**

#### Arguments

<b>SP_number</b>	The SP number in the system.
<b>SP_variable</b>	String representing the name of an SP variable. In order to avoid unexpected results, place <b>SP_variable</b> 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.
```

### 4.6.3 GETSPV



This function is obsolete and is replaced by [GETSP](#).

### 4.6.4 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

```
real SETSP(int SP_number, int Address, value)
```

#### Arguments

<b>SP_number</b>	The SP number in the system.
<b>Address</b>	Address within the SP. Look up addresses by using <b>getspa</b> function
<b>value</b>	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)                          !axes[0].params[0].SLVKP in SP 0 to  
1000.
```

### 4.6.5 SETSPV



This function is obsolete and is replaced by [SETSP](#).

## 4.7 Signal Processing Functions

The Signal Processing functions are:

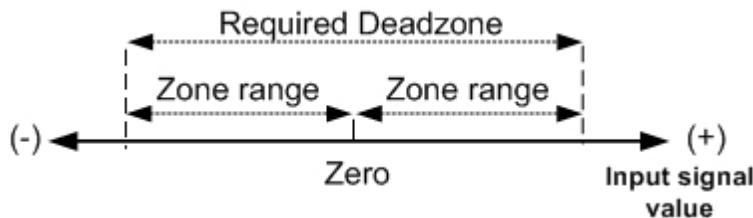
Function	Description
<b>COPY</b>	The function copies one array to another.
<b>DEADZONE</b>	Implements dead-zone routine
<b>DSIGN</b>	Implements a dynamic version of the standard <b>SIGN</b> function.
<b>DSTR</b>	Converts a string to an integer array.
<b>EDGE</b>	Returns 1 on positive edge of x
<b>INP</b>	Reads data characters from the specified channel and stores them into integer array
<b>INTGR</b>	Implements an integrator with <b>DEADZONE</b> and <b>SAT</b> .
<b>LAG</b>	Provides delayed switching on argument change (anti-bouncing effect)
<b>MAP</b>	Implements a table-defined function with constant step
<b>MAPB</b>	Implements a one-dimensional uniform B-spline interpolation
<b>MAPN</b>	One-dimensional non-uniform linear interpolation (replaces the obsolete <b>MAPBY1</b> and <b>MAPBY2</b> functions)
<b>MAPS</b>	One-dimensional non-uniform linear interpolation (replaces the obsolete <b>MAPBY1</b> and <b>MAPBY2</b> functions)
<b>MAPNB</b>	One-dimensional non-uniform b-spline
<b>MAPNS</b>	One-dimensional non-uniform Catmull-Rom spline
<b>MAPS</b>	One-dimensional uniform Catmull-Rom spline
<b>MAP2</b>	Implements a table-defined function with two arguments and constant step along each argument.
<b>MAP2B</b>	Two-dimensional uniform b-spline
<b>MAP2N</b>	Two-dimensional non-uniform linear interpolation (replaces the obsolete <b>MAP2FREE</b> function)
<b>MAP2NB</b>	Two-dimensional non-uniform b-spline
<b>MAP2NS</b>	Two-dimensional non-uniform Catmull-Rom spline
<b>MAP2S</b>	Two-dimensional uniform Catmull-Rom spline

Function	Description
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

### 4.7.1 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 <a href="#">Example 1</a> .
<i>zone2</i>	Any real number or integer expression, required for an asymmetrical deadzone requires two values. See <a href="#">Example 2</a> .

#### Return Value

**DEADZONE** returns a **real number** as follows:

If  $-\text{zone} < \text{input\_signal} < \text{zone}$ , return value = 0

If  $\text{input\_signal} < -\text{zone}$ , return value =  $\text{input\_signal} + \text{zone}$

If:  $\text{input\_signal} > \text{zone}$ , return value =  $\text{input\_signal} - \text{zone}$

#### 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 4-2](#), input\_signal ranges from -20: +20, and creates a symmetrical dead zone from -10: +10.

```
return_value = DEADZONE(input_signal,10)
```

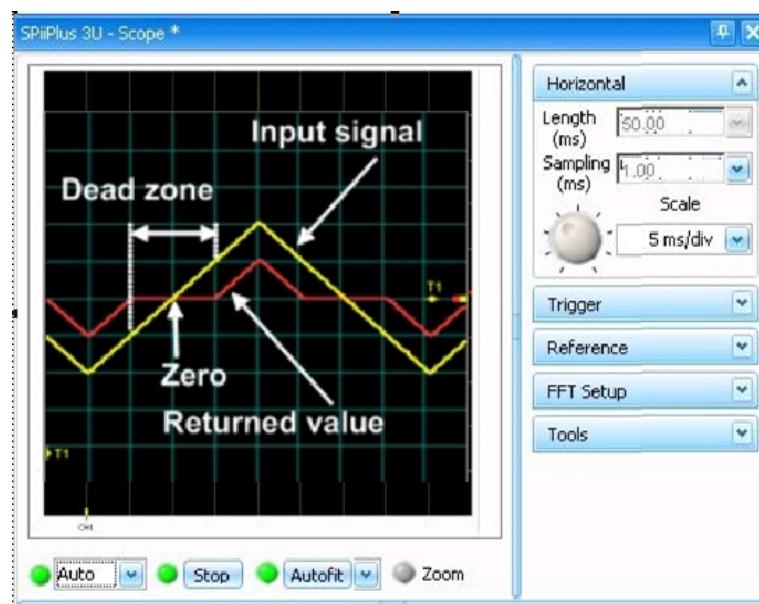


Figure 4-2. Symmetrical Dead Zone Example

### Example 2

#### Asymmetrical Dead Zone

In this example, illustrated in [Figure 4-3](#), input\_signal ranges from -20: +20, and creates an asymmetrical dead zone from -5 – +15.

```
return_value = DEADZONE(input_signal,-5,10)
```

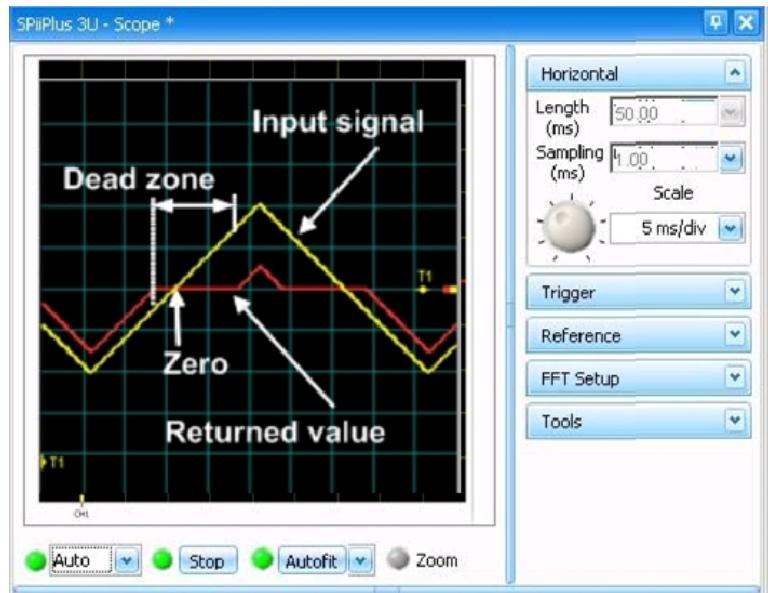


Figure 4-3. Asymmetrical Dead Zone Example

### 4.7.2 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

<b>X</b>	Input variable name declared as real or real expression.
<b>Delay_Time</b>	Real number that determines on zero crossing of X, continues the previous return value for defined delay time in msec.
<b>Ramp_Time</b>	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 4-4 illustrates the **DSIGN** function.

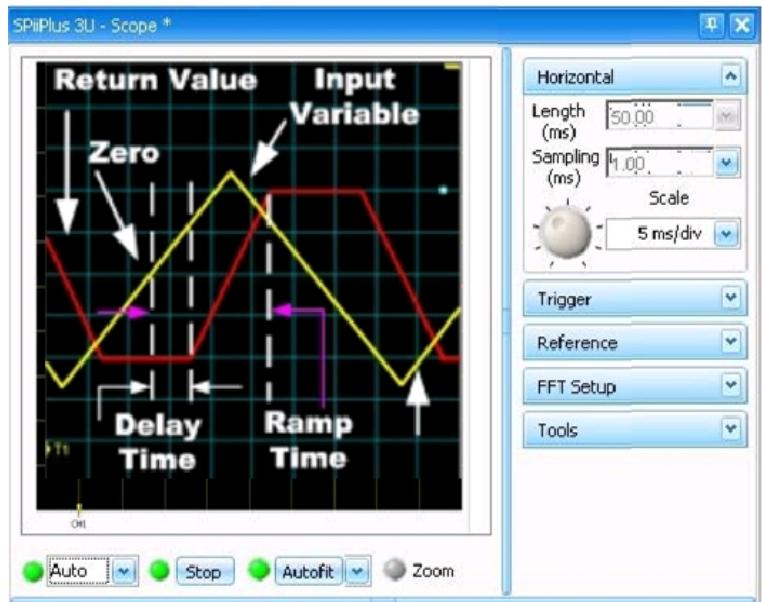


Figure 4-4. DSIGN Function Example

### 4.7.3 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 **number** is omitted, the function assigns all characters of the string. If **number** 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 then 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
```

## 4.7.4 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

<b>X</b>	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)                ! Keeps XX between -5 to 5.
IF (XX>5) | (XX<-5) XI=-XI   ! Ends WHILE
END !Ends IF END              ! Ends program
STOP
```

Figure 4-5 illustrates the **EDGE** function.

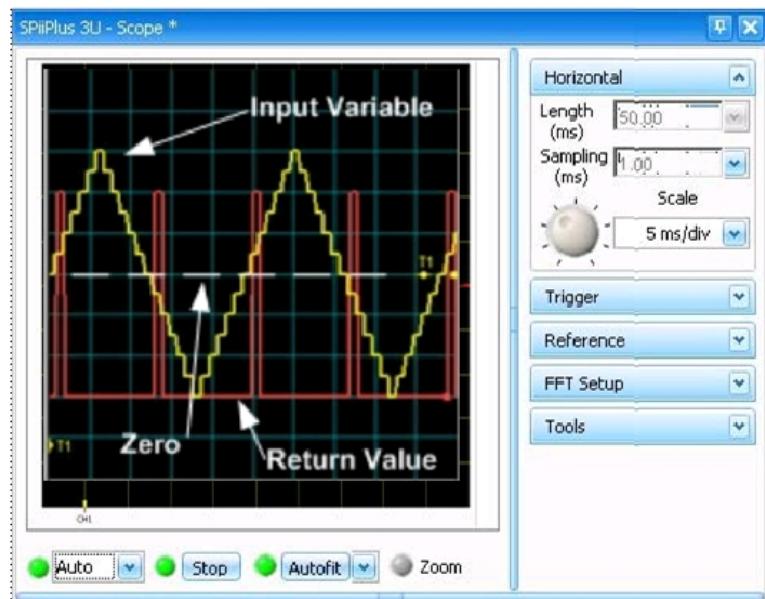


Figure 4-5. EDGE Function Example

### 4.7.5 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 <b>Deadzone</b> value falls into the range of <b>-Deadzone</b> to <b>+Deadzone</b> , <b>INTGR</b> 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 4-6 illustrates the **INTGR** function.

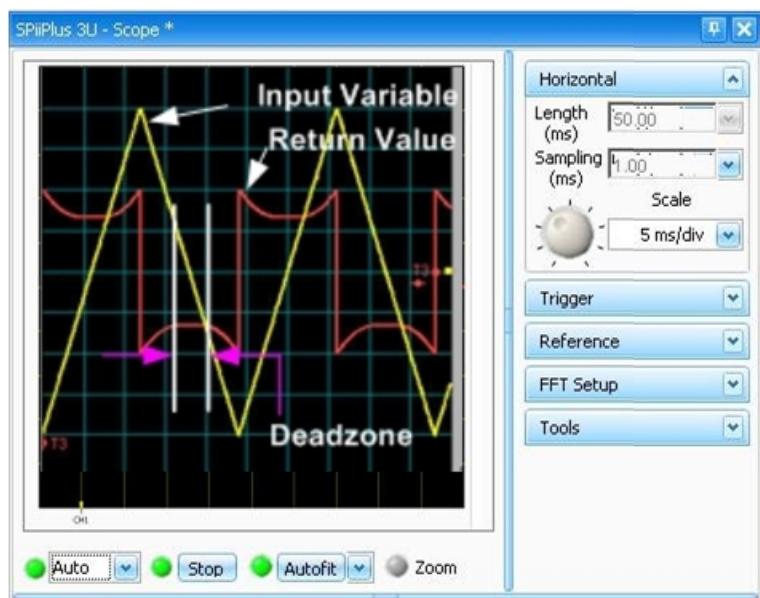


Figure 4-6. INTGR Function Example

## 4.7.6 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 <b>LAG</b> accepts any real or integer argument for <b>X</b> , the typical use implies a logical <b>X</b> argument, for example, an integer variable or expression that supplies only 0 or 1. If <b>X</b> supplies values other than 0 or 1, <b>LAG</b> treats any non-zero value as 1.
<b>P_delay</b>	A real number representing the delay on the positive edge in msec.
<b>down_delay</b>	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
XT=TIME; XX=0
              ! Define three integers
              ! XT equals the time elapsed from
              ! startup. XX equal zero.

WHILE 1
IF TIME-XT>100; XX=^XX; XT=TIME; END
              !Run an infinite routine
              !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
STOP
              !Ends WHILE
              !Ends program

```

[Figure 4-7](#) illustrates the **LAG** function.

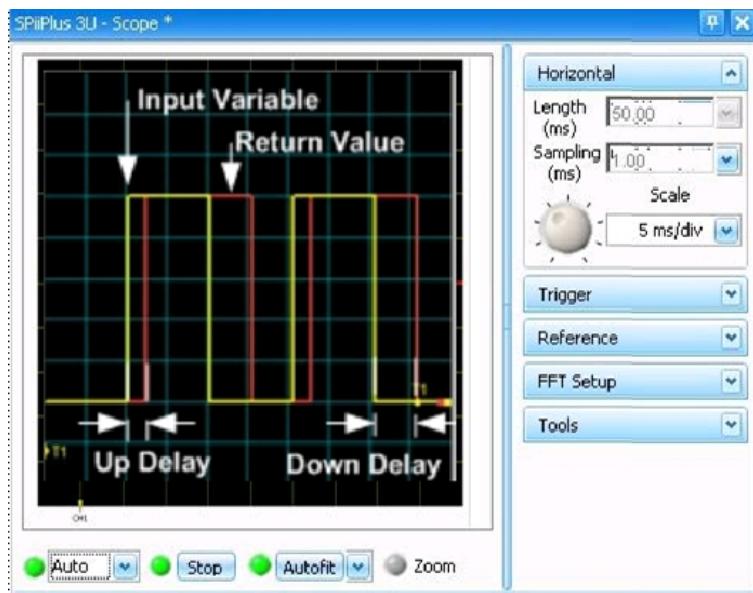


Figure 4-7. LAG Function Example

#### 4.7.7 Interpolation Functions

There are a number of interpolation methods between points among them are Linear and Spline.

##### 4.7.7.1 Linear interpolation

Linear interpolation of function  $y(x)$  between distant points  $a$  and  $b$  are calculated as follows:

$$y(x_i) = \frac{y(b) - y(a)}{b - a}x_i + y(a) - a \quad x_i \in [a, b]$$

##### 4.7.7.2 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:  $t_i$  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 3<sup>rd</sup> 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.

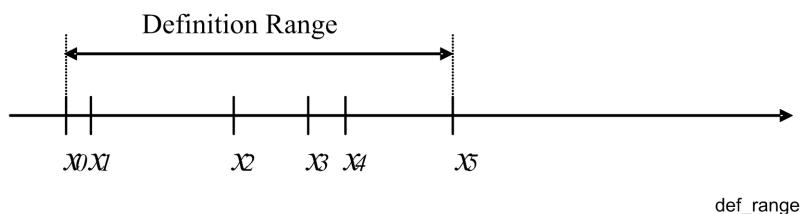
The SPiiPlus controller also supports two-dimensional splines. In this case, the definition table is a two-dimensional matrix. Knot points are defined for two arguments  $x$  and  $y$ , and the matrix contains corresponding  $p$  values. Knot values divide the XY plane into rectangular cells. The matrix 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 4-8](#) illustrates definition range of a function defined with six non-uniform knots:



**Figure 4-8. Spline Definition Range**

In a two-dimensional case, definition range is a rectangular area, as illustrated in [Figure 4-9](#):

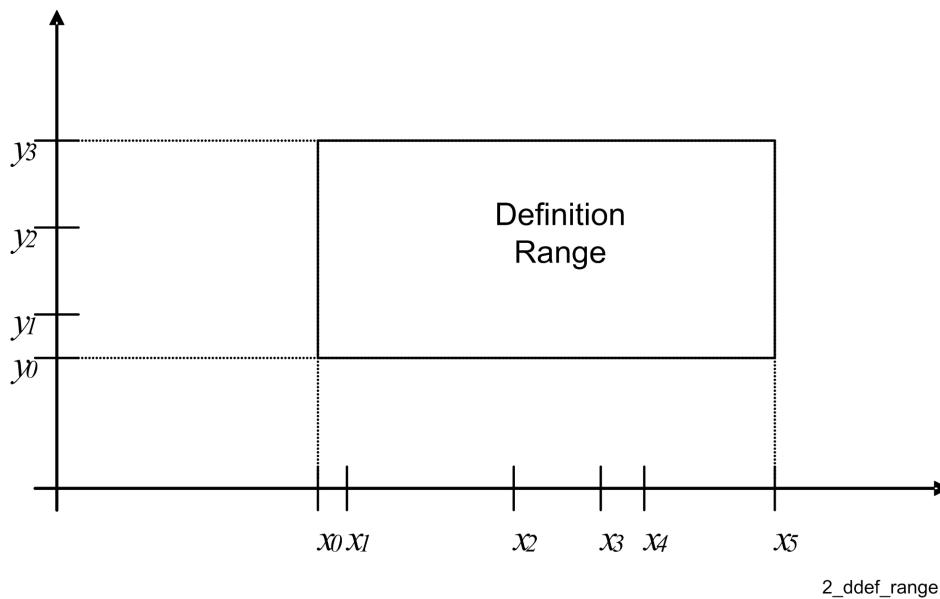


Figure 4-9. Two-Dimensional Spline Definition Range

#### 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 3<sup>rd</sup> 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 4-10 illustrates a Catmull-Rom spline that interpolates a 5-component definition table.

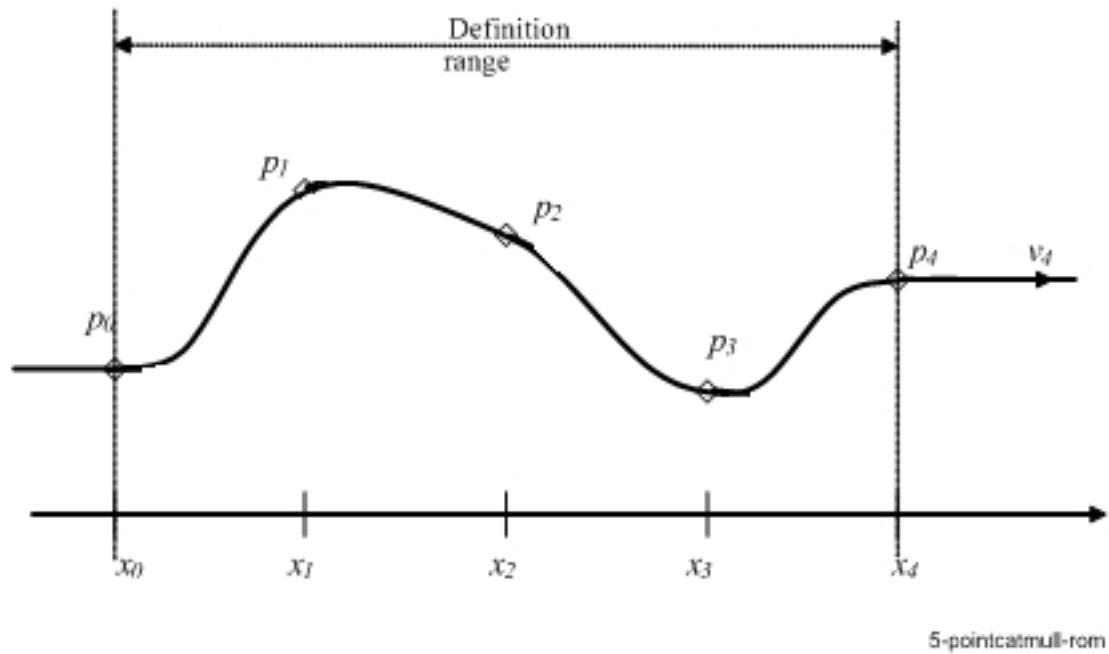


Figure 4-10. 5-Point Catmull-Rom Spline

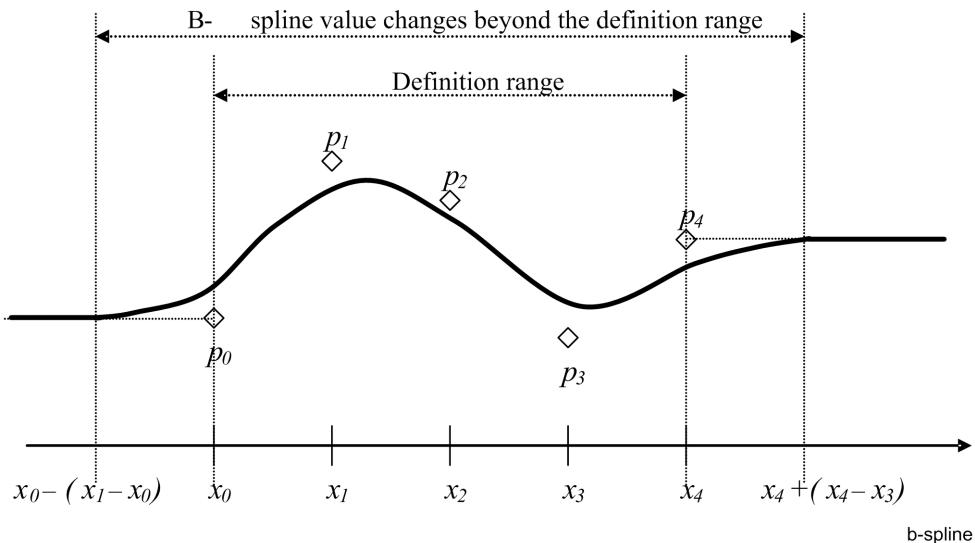
Features of the Catmull-Rom spline:

- 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 4-11](#).



**Figure 4-11. 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.

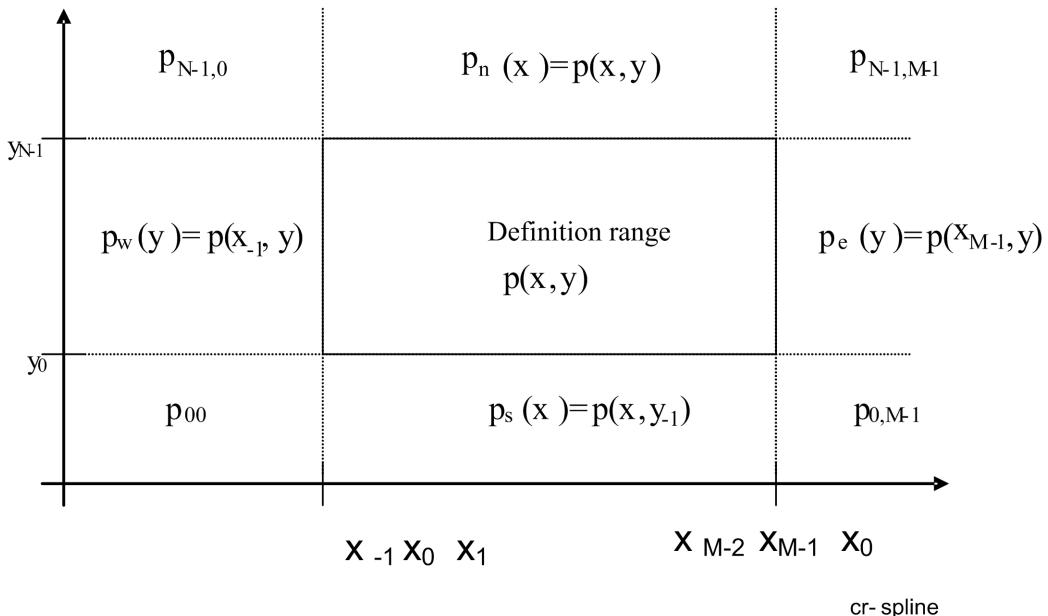
A two-dimensional spline approximates a definition table that provides  $N \times M$  control points  $p_{00}, p_{01}, \dots, p_{0,M-1}, p_{10}, p_{11}, \dots, p_{N-1,M-1}$  on the grid defined by knots  $x_0, x_1, x_2 \dots x_{M-1}$  and  $y_0, y_1, y_2 \dots y_{N-1}$ .

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 4-12](#) illustrates the Catmull-Rom spline beyond the definition range:



**Figure 4-12. Catmull-Rom 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:

$$x_{-1} = x_0 - (x_1 - x_0)$$

$$x_M = x_{M-1} + (x_{M-1} - x_{M-2})$$

$$y_{-1} = y_0 - (y_1 - y_0)$$

$$x_N = x_{N-1} + (x_{N-1} - x_{N-2})$$

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:

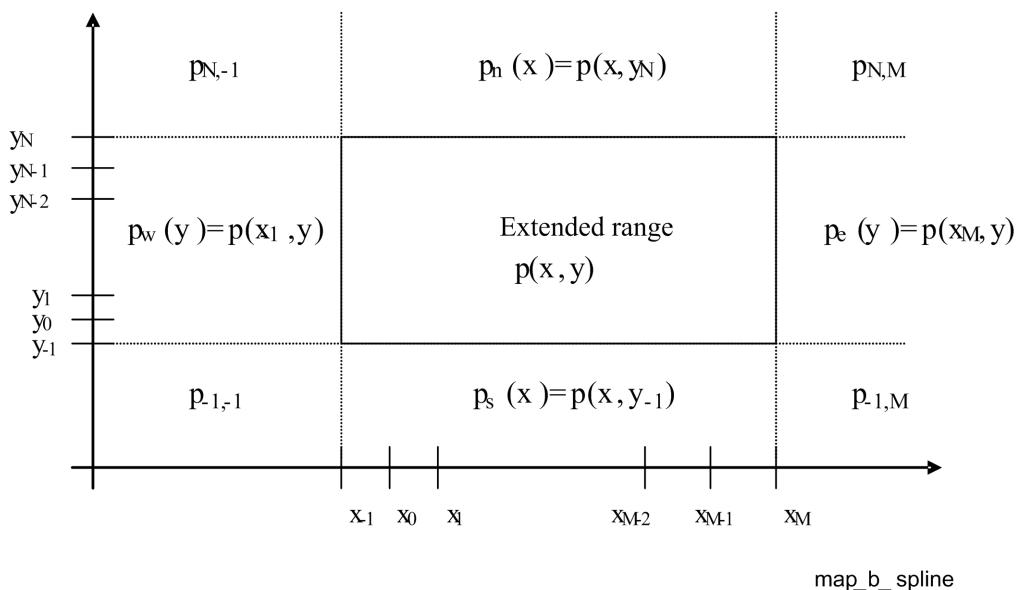


Figure 4-13. B-Spline Map

#### 4.7.7.3 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

<b>X</b>	Real variable name or real expression.
<b>array</b>	The name of a real one-dimensional array that specifies points
<b>base</b>	A real number representing the value of <b>XX</b> that corresponds to the first point in array.
<b>step</b>	A real number representing the value of <b>XX</b> 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 4-6 shows the XX and YY values.

Table 4-6. 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 4-14 illustrates this MAP example on the Scope.

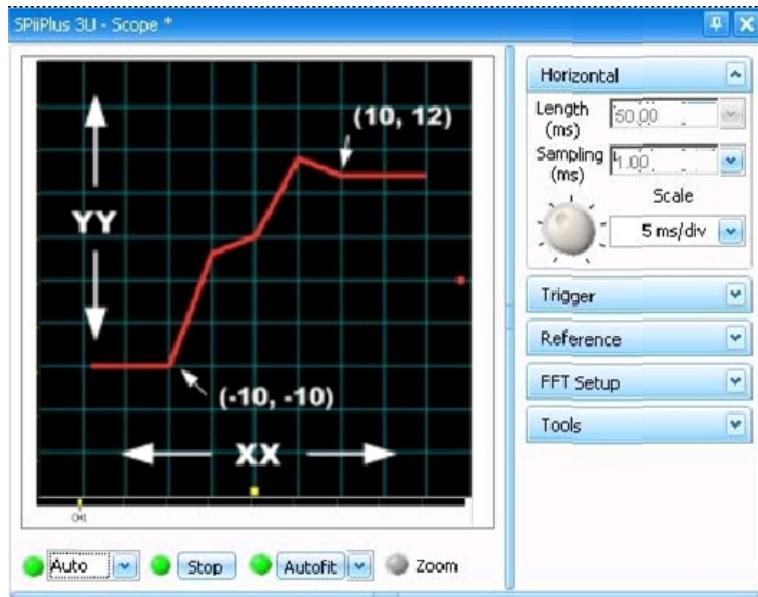


Figure 4-14. MAP Example on the Scope

#### 4.7.7.4 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

<b>XX</b>	Real variable name or real expression.
<b>array</b>	The name of a real one-dimensional array that specifies points
<b>base</b>	A real number representing the value of <b>XX</b> that corresponds to the first point in <b>array</b> .
<b>step</b>	A real number representing the value of <b>XX</b> that defines fixed intervals between the <b>array</b> 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+XI;YY=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
STOP
                                         !Ends mapping
                                         !Ends program

```

Table 4-7 shows the XX and YY values.

Table 4-7. MAPB 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	-7.83	-2.45	1.16	6.16	12.33	12.33	12	12

Figure 4-15 illustrates this **MAPB** example on the Scope.

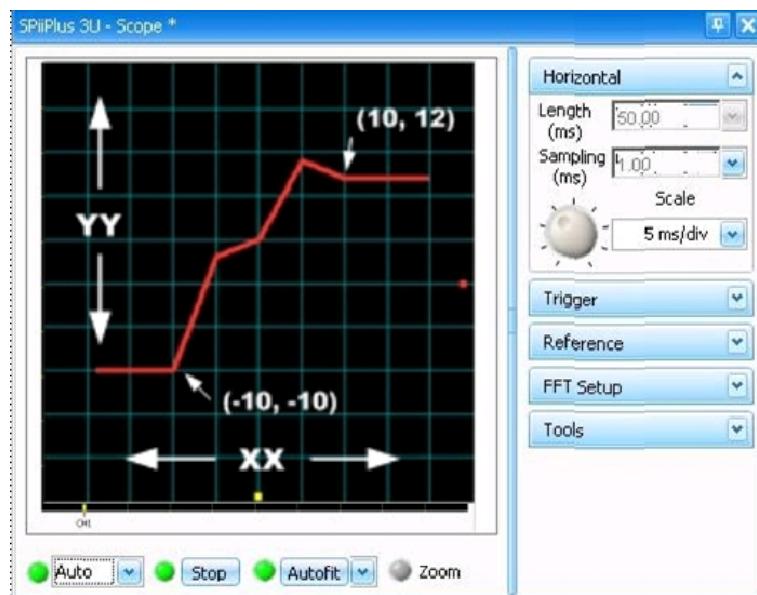


Figure 4-15. MAPB Example on the Scope

#### 4.7.7.5 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

<b>XX</b>	Real variable name or real expression.
<b>X_table</b>	The name of a real one-dimensional array that specifies the values of <b>XX</b> that correspond to the point array.
<b>Y_table</b>	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 4-8 shows the XX and YY values.

Table 4-8. 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 4-16 illustrates this **MAPN** example on the Scope.

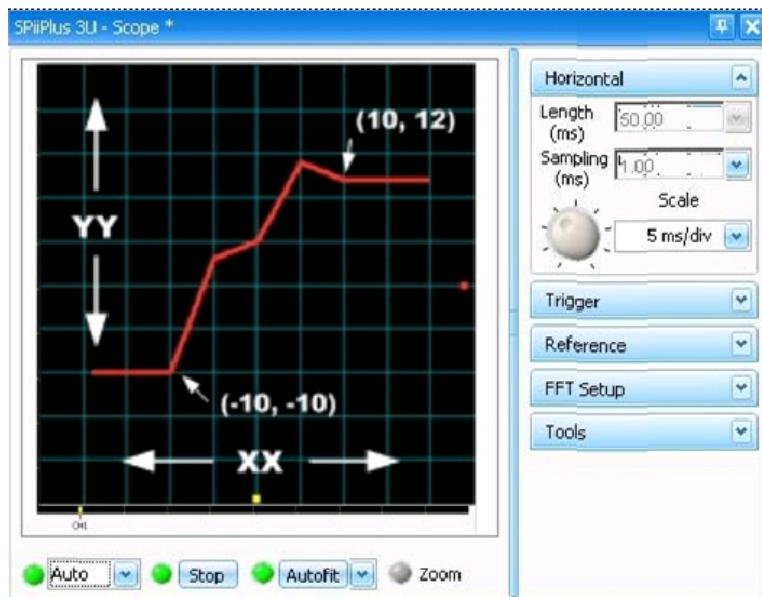


Figure 4-16. MAPN Example on the Scope

#### 4.7.7.6 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

<b>XX</b>	Real variable name or real expression.
<b>X_table</b>	The name of a real one-dimensional array that specifies the values of <b>XX</b> that correspond to the point array.
<b>Y_table</b>	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
STOP

```

Table 4-9 shows the XX and YY values.

Table 4-9. MAPNB 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	-8.02	-4.65	-0.41	7.64	9.23	5	4.124	4

Figure 4-17 illustrates this **MAPNB** example on the Scope.

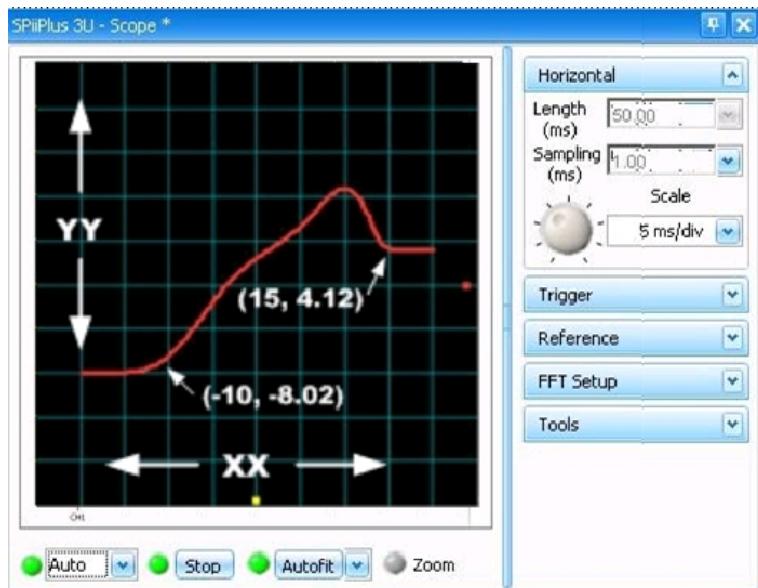


Figure 4-17. MAPNB Example on the Scope

#### 4.7.7.7 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

<b>XX</b>	Real variable name or real expression.
<b>X_table</b>	The name of a real one-dimensional array that specifies the values of <b>XX</b> that correspond to the point array.
<b>Y_table</b>	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
STOP                         !Ends mapping
                            !Ends program

```

Table 4-10 shows the XX and YY values.

**Table 4-10. 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 4-18 illustrates this **MAPNS** example on the Scope.

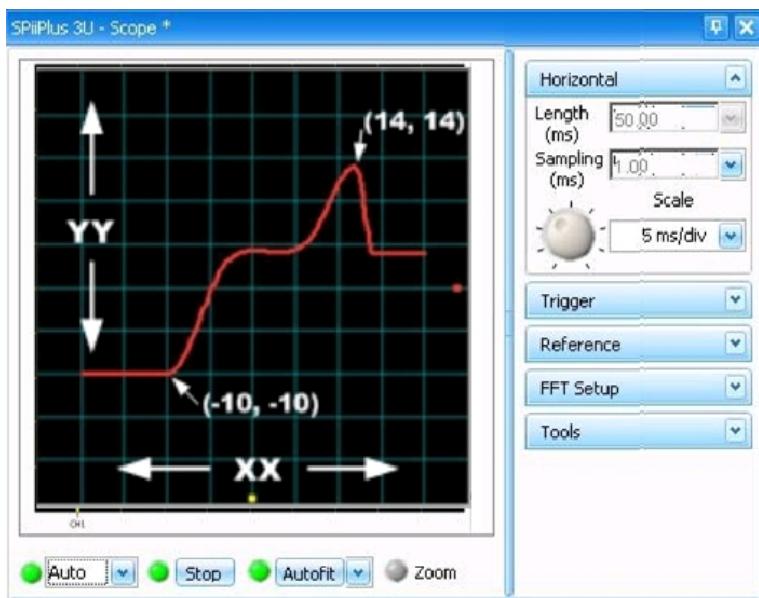


Figure 4-18. MAPNS Example on the Scope

#### 4.7.7.8 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

<b>XX</b>	Real variable name or real expression.
<b>array</b>	The name of a real one-dimensional array that specifies points
<b>base</b>	A real number representing the value of <b>XX</b> that corresponds to the first point in array.
<b>step</b>	A real number representing the value of <b>XX</b> 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
STOP                               !Ends mapping
                                    !Ends program

```

Table 4-11 shows the XX and YY values.

Table 4-11. 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 4-19 illustrates this **MAPS** example on the Scope.

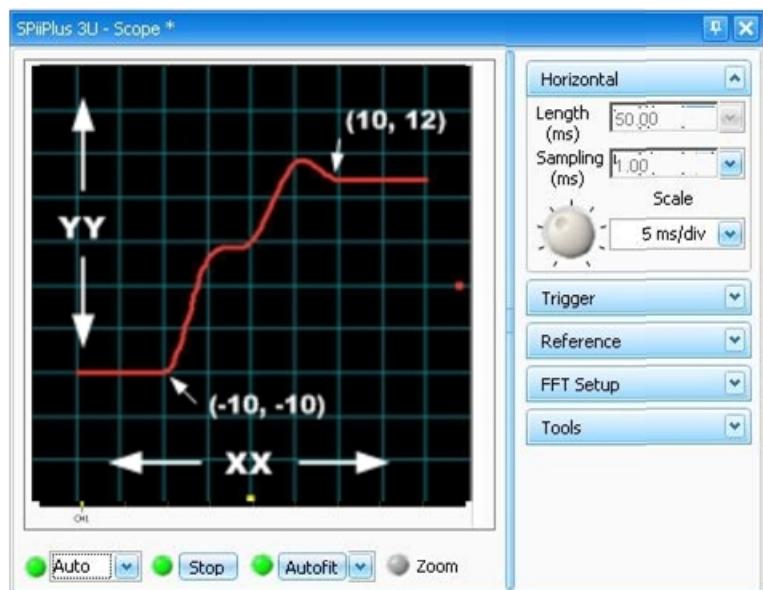


Figure 4-19. MAPS Example on the Scope

#### 4.7.7.9 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

<b>XX</b>	Real variable name or real expression.
<b>ZZ</b>	Real variable name or real expression.
<b>table</b>	The name of a real two-dimensional array that specifies points
<b>baseX</b>	A real number representing the value of <b>XX</b> that corresponds to the first point in the array.
<b>stepX</b>	A real number representing the value of <b>XX</b> that defines fixed intervals between the array points.
<b>baseY</b>	A real number representing the value of <b>ZZ</b> that corresponds to the first point in the array.
<b>stepY</b>	A real number representing the value of <b>ZZ</b> 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 4-12 shows the YY values as a function of XX and ZZ arguments.

Table 4-12. MAP2

		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 4-20 illustrates this **MAP2** example on the Scope.

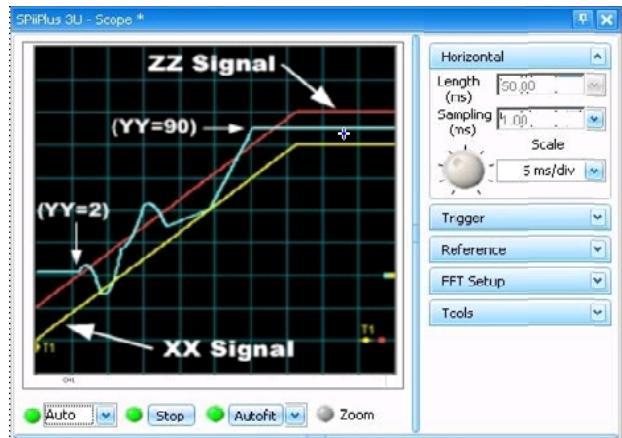


Figure 4-20. MAP2 Example on the Scope

#### 4.7.7.10 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

<b>XX</b>	Real variable name or real expression.
<b>ZZ</b>	Real variable name or real expression.
<b>table</b>	The name of a real two-dimensional array that specifies points
<b>baseX</b>	A real number representing the value of <b>XX</b> that corresponds to the first point in the array.
<b>stepX</b>	A real number representing the value of <b>XX</b> that defines fixed intervals between the array points.
<b>baseY</b>	A real number representing the value of <b>ZZ</b> that corresponds to the first point in the array.
<b>stepY</b>	A real number representing the value of <b>ZZ</b> 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
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
END
STOP
        !Ends MAP2B.
        !Ends program

```

Table 4-13 shows the YY values as a function of XX and ZZ arguments.

**Table 4-13. MAP2B**

		XX							
		-30	-10	-5	0	5	10	50	
ZZ	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 4-21 illustrates this **MAP2B** example on the Scope.

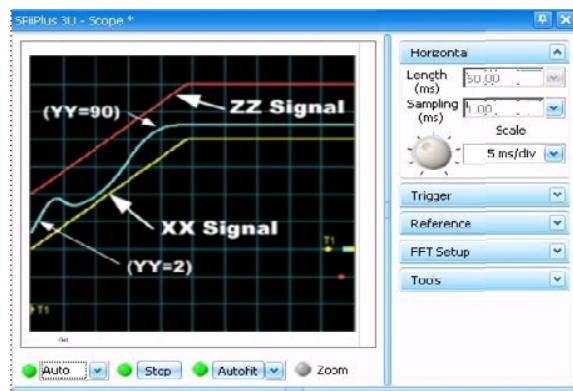


Figure 4-21. MAP2B Example on the Scope

#### 4.7.7.11 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

<b>XX</b>	Real variable name or real expression.
<b>ZZ</b>	Real variable name or real expression.
<b>table</b>	The name of a real two-dimensional array that specifies points
<b>X_table</b>	A one-dimensional real array that specifies the values of <b>XX</b> that corresponds the point array.
<b>Y_table</b>	A one-dimensional real array that specifies the values of <b>ZZ</b> 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 4-14](#) shows the YY values as a function of XX and ZZ arguments.

Table 4-14. 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 4-22 illustrates this **MAP2N** example on the Scope.

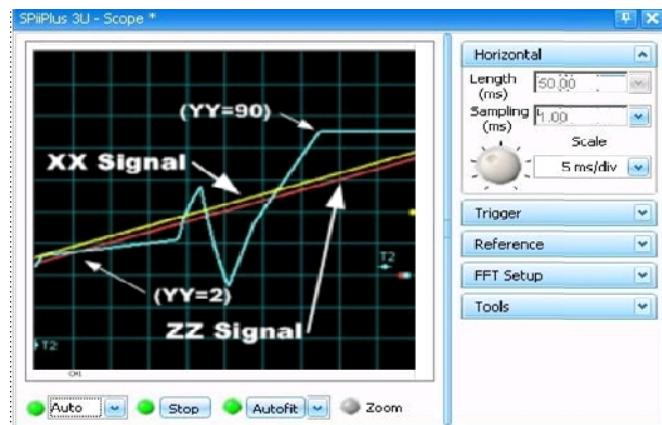


Figure 4-22. MAP2N Example on the Scope

#### 4.7.7.12 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

**XX**

Real variable name or real expression.

<b>ZZ</b>	Real variable name or real expression.
<b>table</b>	The name of a real two-dimensional array that specifies points
<b>X_table</b>	A one-dimensional real array that specifies the values of <b>XX</b> that corresponds the point array.
<b>Y_table</b>	A one-dimensional real array that specifies the values of <b>ZZ</b> 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
ZZ=ZZ+XI
YY=MAP2NB (XX,ZZ,TABLE,XTABLE,YTABLE)
DISP XX,ZZ,YY
END
STOP

```

! Set conditions  
 ! 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.  
 ! Displays values of XX, ZZ and YY.  
 ! Ends MAP2N.  
 ! Ends program

Table 4-15 shows the YY values as a function of XX and ZZ arguments.

Table 4-15. 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.

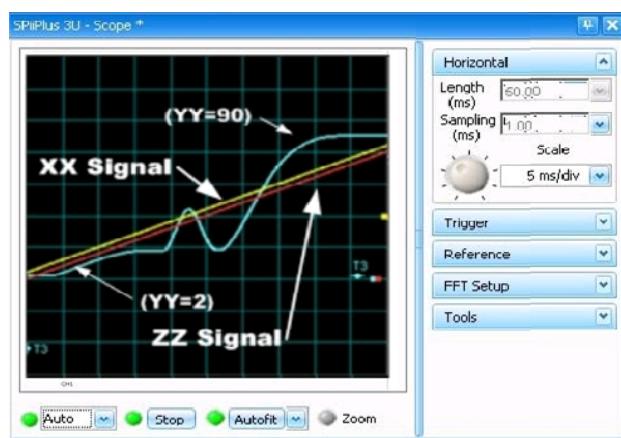


Figure 4-23. MAP2NB Example on the Scope

#### 4.7.7.13 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

<b>XX</b>	Real variable name or real expression.
<b>ZZ</b>	Real variable name or real expression.
<b>table</b>	The name of a real two-dimensional array that specifies points
<b>X_table</b>	A one-dimensional real array that specifies the values of <b>XX</b> that corresponds the point array.
<b>Y_table</b>	A one-dimensional real array that specifies the values of <b>ZZ</b> 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  

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  

END  

STOP  

!Displays values of XX, ZZ and YY.  

!Ends MAP2N.  

!Ends program

```

Table 4-16 shows the YY values as a function of XX and ZZ arguments.

**Table 4-16. 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	

<b>Key:</b>	XX and ZZ values
	Interpolated YY values outside the boundaries of array "Table"
	"Table" array values

Figure 4-24 illustrates this **MAP2NS** example on the Scope.

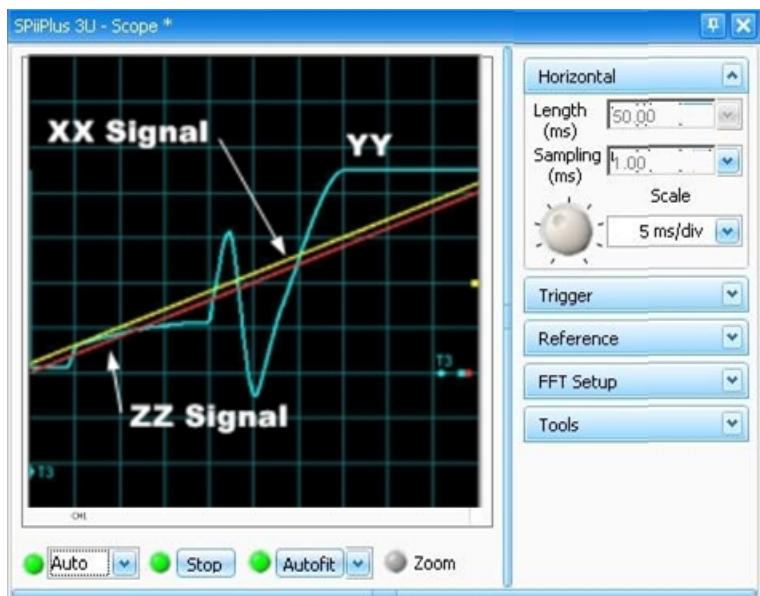


Figure 4-24. MAP2NS Example on the Scope

#### 4.7.7.14 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

<b>XX</b>	Real variable name or real expression.
<b>ZZ</b>	Real variable name or real expression.
<b>table</b>	The name of a real two-dimensional array that specifies points
<b>baseX</b>	A real number representing the value of <b>XX</b> that corresponds to the first point in the array.
<b>stepX</b>	A real number representing the value of <b>XX</b> that defines the fixed intervals between the array points.
<b>baseY</b>	A real number representing the value of <b>ZZ</b> that corresponds to the first point in the array.
<b>stepY</b>	A real number representing the value of <b>ZZ</b> 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 4-17 shows the YY values as a function of XX and ZZ arguments.

Table 4-17. 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 4-25 illustrates this **MAP2S** example on the Scope.

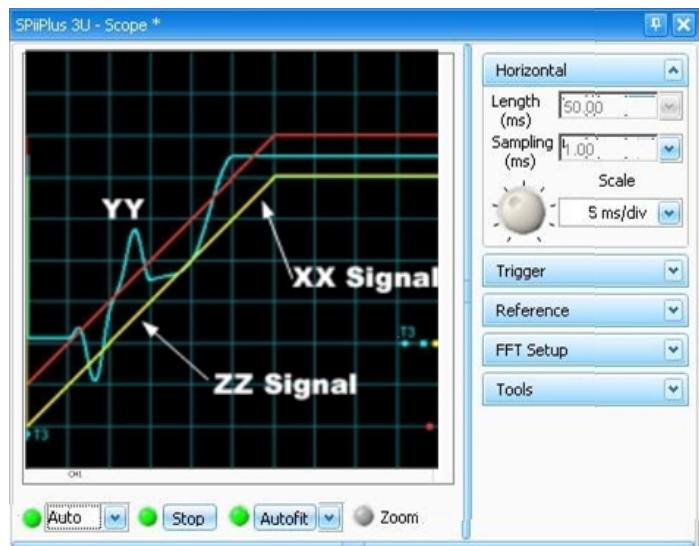


Figure 4-25. MAP2S Example on the Scope

#### 4.7.8 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.

<b>from</b>	The start point to begin searching for matching values of <b>APOS</b> .
<b>to</b>	The end point to stop searching for matching values of <b>APOS</b> .

## 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

## 4.7.9 RAND

### Description

**RAND** generates a random number

### Syntax

real **RAND(min, max[,seed])**

### Arguments

<b>min</b>	Lower boundary of the interval from which randomized number will be selected.
<b>max</b>	Upper boundary of the interval from which randomized number will be selected.
<b>seed</b>	Sets the random sequence generator. The number range is limited only by the controller register size - 32bit. If omitted, the controller uses ACSPL+ <b>TIME</b> 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

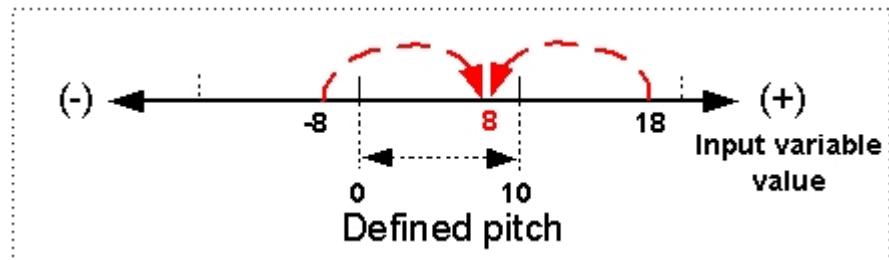
### Example

```
RAND (-45,45,TIME)
```

## 4.7.10 ROLL

### Description

The **ROLL** function returns a result rolled-over to within a defined Pitch (range), as illustrated below.

**Syntax****ROLL(X, Pitch)****Arguments**

<b>X</b>	Variable name declared as real or a real expression.
<b>Pitch</b>	A range from 0 to <b>Pitch</b> (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

 $|X|/\text{Pitch} - \text{FLOOR}(|X|/\text{Pitch}) * \text{Pitch}$ , 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.
```

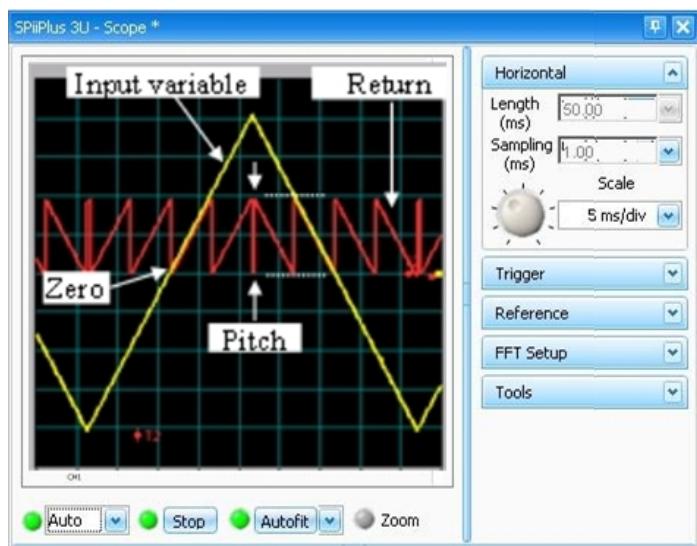
Figure 4-26 illustrates this **ROLL** example on the Scope.

Figure 4-26. ROLL Example on the Scope

### 4.7.11 SAT

#### Description

SAT returns a result of defined saturation range

#### Syntax

real **SAT(X, Min, Max)**

#### Arguments

<b>X</b>	Variable name declared as real or a real expression.
<b>Min</b>	Low saturation value. Must be a real number.
<b>Max</b>	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 4-27 illustrates this **SAT** example on the Scope.

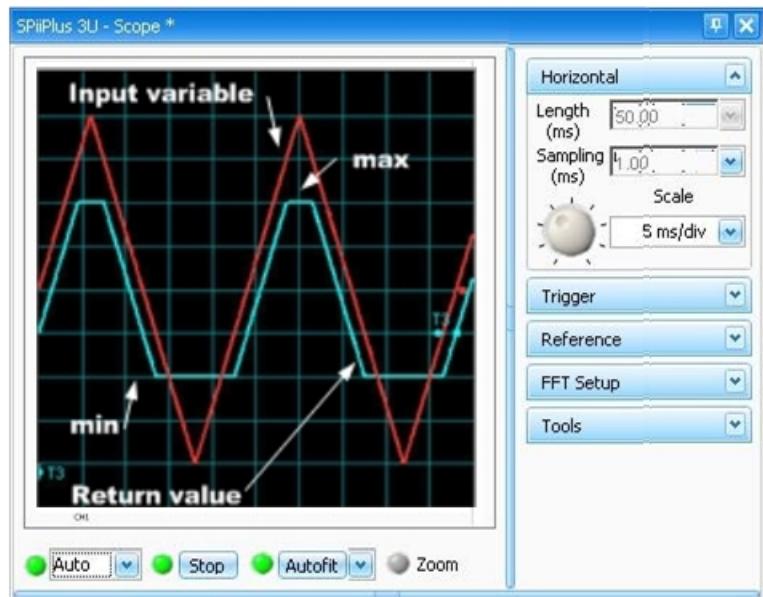


Figure 4-27. SAT Example on the Scope

## 5. 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.

### 5.1 Entering Terminal Commands

Terminal commands are entered in the **Communication Terminal** (the general structure of which is shown in [Figure 5-1](#)).

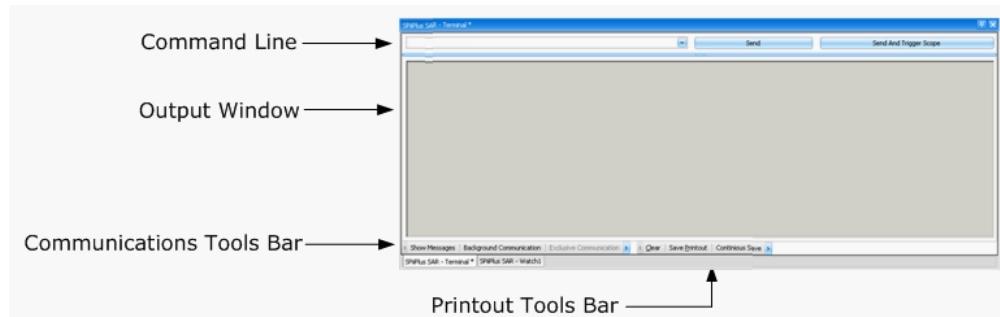
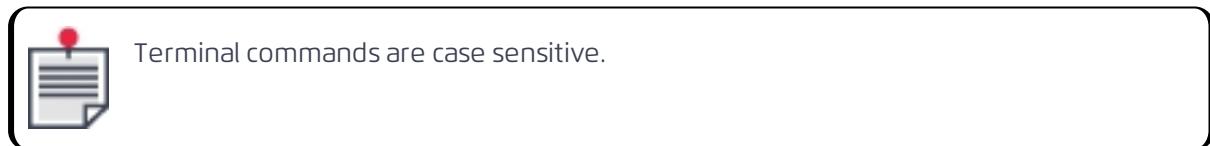


Figure 5-1. Communication Terminal Window

The **Communication Terminal** window is described in the *SPiiPlus MMI Application Studio User Guide*.



Terminal commands are divided into:

- [Query Commands](#)
- [Program Management Commands](#)
- [System Commands](#)

### 5.2 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 .	?TCPPIP – Returns the TCP/IP for the Ethernet port N1.

Command & Syntax	Description	Example
<code>?[ACSP<sup>L</sup>+ variable ACSP<sup>L</sup>+ variable]...</code>	Returns the current value of listed standard variables.	<b>?TCPPIP, TCPPORT</b> – Returns the TCP/IP for the Ethernet port N1 and the TCP port number.
<code>?[buffer number]: [local user-defined variable]</code>	Returns the current values of a local user variable or array defined in a program buffer.	<b>?1:MY_VAR</b> – Returns the values of a local user-defined variable named MY_VAR.
<code>?[array_variable (index)]</code>	Returns the current value of a specific element in the given array. The brackets enclosing the index are optional.	<b>?FPOS(0)</b> – Returns the feedback position of 0 axis. <b>?FPOS0:</b> Returns the feedback position of 0 axis.
<code>?[global user array_variable [(index)]]</code>	Returns the current values of a global user variable or array. Or the value of an element in the array if index is included.	<b>?ARRAY1</b> – Returns the values of a global user-defined array variable named ARRAY1. <b>?ARRAY1(3)</b> – Returns the value of the fourth element of a global user-defined array variable named ARRAY1.
<code>?[matrix_variable (row_index) (col_index)]</code>	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.	<b>?MATRIX</b> – Returns all of the values (in tabular format) of the two-dimensional array named MATRIX. <b>?MATRIX(1)(0)</b> – Returns the value of the second element in the first column of MATRIX. <b>?MATRIX(0)</b> – Returns all the values of the first row of MATRIX. <b>?MATRIX(0,2)(0,1)</b> – Returns a range of values, those of the first through third rows in the first and second columns of MATRIX.
<code>?[buffer number]</code>	Returns the current status and information about a program buffer	<b>?0</b> – Returns the current status and information about program buffer 0.
<code>?#</code>	Returns the current status of all program buffers.	

Command & Syntax	Description	Example
<code>?\$[axis number]</code>	Returns the current status of the motor for the specified axis.	<code>?\$16</code> – Returns the current status of the axis 16 motor.
<code>?\$</code>	Returns the current status of all motors.	
<code>?VR</code>	Returns the firmware version	
<code>?SN</code>	Returns the controller serial number and hardware version - indicated by a letter.	
<code>??[error_code]</code>	Returns the error code number. If the error code is included in the query, returns error description.	<code>??3260</code> : Returns "Motor cannot start because the motor is disabled"
<code>??[variable_name]</code>	Returns a brief description of the variable.	<code>??CERRA</code> - Returns "Critical Position Error In Accelerating"

### 5.2.1 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.

### 5.2.2 Predefined Query Output Formats

The controller provides several predefined formats that can be used instead of the default format for querying variables:

Format	Description
<code>?D/</code>	Decimal format.

Format	Description
	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. C-equivalent: %10i.
?X/	Hexadecimal format. When applied to an integer variable, this format displays the hexadecimal presentation of the variable. C-equivalent: %08X
?B/	Binary format. This format is identical to the default format for the <b>IN</b> and <b>OUT</b> variables. When applied to an integer variable, the format displays the binary presentation of the variable.
?E/	Extended format. 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. C-equivalent: %20G

## 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
```

### 5.2.3 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
? {%-12.3f}FPOS	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.
?{%-8.0f}X_FPOS	8 digits, no decimal point, no fraction digits.
?{XFPOS = %8.0f}X_FPOS	The response will look like XFPOS = 1234.
?{%-08X} X_MST	8 digits, hexadecimal format with leading zeros

## 5.3 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.

### 5.3.1 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
```

## 5.3.2 Program Buffer Commands

### 5.3.2.1 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[I][line_number]
```

### Arguments

<b>buffer_number</b>	<b>buffer_number</b> qualifier in the command specifies the buffer, a number between 0 and 16.
<b>I</b>	Optional, if included, opens the buffer for insertion of code.
<b>line_number</b>	Optional, if included, the command opens the buffer specified by <b>buffer_number</b> and sets the insert line before the line specified by <b>line_number</b> . <b>line_number</b> 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.
#3I2        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

```

### 5.3.2.2 D

#### Description

The **D** (Delete) command deletes the specified lines in the buffer.

#### Syntax

**#buffer\_numberDline\_number[,line\_number]**

#### Arguments

<b>buffer_number</b>	<b>buffer_number</b> qualifier in the command specifies the buffer, a number between 0 and 16.
<b>line_number</b>	<p><b>line_number</b> in the <b>D</b> command is obligatory. <b>line_number</b> can be:</p> <ul style="list-style-type: none"> <li>● A single number, specifying one specific line, or</li> <li>● 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.</li> </ul>

#### 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
#
:          Close the buffer
The prompt indicates that all buffers are closed
```

### 5.3.2.3 F/IF

#### Description

The **F/FI** (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

<b>buffer_number</b>	<b>buffer_number</b> qualifier in the command specifies the buffer, a number between 0 and 16.
<b>search_string</b>	The text being sought.
<b>line_number</b>	Optional, if included, <b>line_number</b> 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 <b>stop</b> 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	

### 5.3.2.4 L

#### Description

The **L** (List) command is used for displaying a program listing.

#### Syntax

**#buffer\_numberL[line\_number]**

#### Arguments

<b>buffer_number</b>	<b>buffer_number</b> specifies the buffer, a number between 0 and 16; or you can use the pound (#) to designate all buffers.
<b>line_number</b>	Optional, if included, <b>line_number</b> defines a specific line to be listed. Otherwise, the search starts from the first line. <b>line_number</b> can be specified in any of the three forms listed in <a href="#">Table 5-1</a> .

#### 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:
```

## 5.3.3 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.

## 5.3.4 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.

### 5.3.4.1 VGR

#### Description

The **VGR** command lists the categories within which the ACSPL+ variables are grouped. The categories of the ACSPL+ 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

**group\_name**

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+ variables within the category.



The category must be entered in exactly the same format as given in the list above.

### 5.3.4.2 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 <a href="#">VGR</a> ).
-------------------	---

#### 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.

### 5.3.4.3 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 <a href="#">VGR</a> ).
-------------------	---

#### 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
AERR
AFLAGS
AIN
AOUT
APOS
AST
BAUD
```

```
BOFFTIME
BONTIME
MASK
FPOS
FVEL
FVFIL
GACC
GETIME
GJERK
GMOT
GQU
GMTYPE
```



For brevity, only a portion of the response is shown here.

#### 5.3.4.4 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

**group\_name**

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.

#### 5.3.4.5 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

<b>buffer_no</b>	A number ranging from 0 to 16, representing a specific buffer.
<b>variable_name</b>	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

### 5.3.4.6 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

<b>buffer_no</b>	A number ranging from 0 to 16, representing a specific buffer.
<b>variable_name</b>	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.

### 5.3.4.7 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

<b>buffer_no</b>	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	

### 5.3.4.8 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

<b>servo_number</b>	<b>servo_number</b> 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
```

### 5.3.4.9 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

<i>group_name</i>	One of the ACSPL+ variable categories (see <a href="#">VGR</a> ).
-------------------	---

#### Comments

When **group\_name** is included with the **VSGT** command, the ACSPL+ variables within the specified category are listed.

### 5.3.4.10 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 PROTECT can be applied.

#### Syntax

#VSTF

#VSGTF [group\_name]

#VSDT [group\_name]

#### Arguments

<i>group_name</i>	One of the ACSPL+ variable categories (see <a href="#">VGR</a> ).
-------------------	---

#### Comments

When **group\_name** is included with the **VSGTF** or **VSDT** command, the ACSPL+ variables within the specified category are listed.

### 5.3.4.11 VGV

#### Description

The **VGV** command is used to remove global variables that have been set via **Communication Terminal**.

#### Syntax

#VGV [global\_var]

### Arguments

*global\_var* Name of a global variable

### Comments

If **global\_var** is included, **VGV** removes only this variable; otherwise it removes all of them.

### 5.3.5 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:

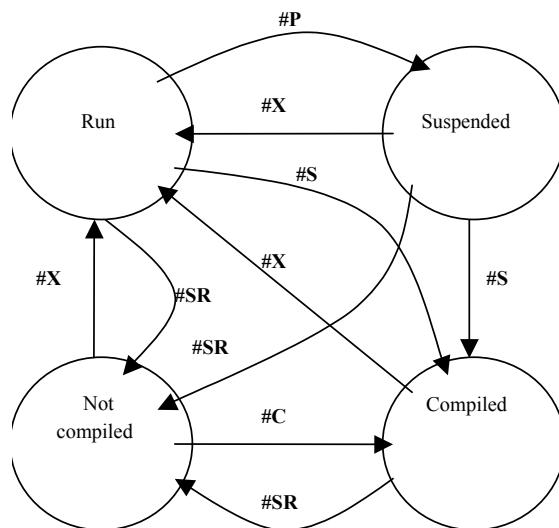


Figure 5-2. Interaction of Program Buffer States

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** 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** command, or autoroutine condition is satisfied.

### 5.3.5.1 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

<b>buffer_number</b>	<b>buffer_number</b> 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.

<b>#0C</b>	Compile the program in buffer 0
<b>5 lines compiled</b>	Response, the program was compiled successfully
<b>#9C</b>	Compile the program in buffer 9
<b>?2026 in line 16</b>	Response, Error 2026 was found in line 16
<b>??2026</b>	Explain error 2026.
<b>Undefined variable name</b>	Response
<b>##C</b>	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

### 5.3.5.2 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>	<b>buffer_number</b> qualifier in the command specifies the buffer, a number between 0 and 16.
<i>line_number</i>	Optional, <b>line_number</b> can be a line number or a label. Execution starts from the specified line. If <b>line_number</b> 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 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

### 5.3.5.3 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**

<b>buffer_number</b>	<b>buffer_number</b> 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

**5.3.5.4 P**

**Description**

The **P** (Pause) command suspends program execution in a buffer.

**Syntax**

#buffer\_numberP

**Arguments**

<b>buffer_number</b>	<b>buffer_number</b> 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

## 5.3.6 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

### 5.3.6.1 XS

#### Description

The **XS** (Execute one step) command executes one program line.

#### Syntax

**#buffer\_numberXSline\_number**

#### Arguments

<b>buffer_number</b>	<b>buffer_number</b> specifies the buffer, a number between 0 and 16.
<b>line_number</b>	<b>line_number</b> 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.

### 5.3.6.2 XD

#### Description

The **XD** (Execute in Debug mode) command executes the program up to the next breakpoint (see [BS](#)).

#### Syntax

#buffer\_numberXD

#### Arguments

*buffer\_number*

**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.

### 5.3.6.3 BS

#### Description

The **BS** (Set Breakpoint) command sets a breakpoint at the specified line.

#### Syntax

#buffer\_numberBSline\_number

#### Arguments

*buffer\_number*

**buffer\_number** specifies the buffer, a number between 0 and 16.

*line\_number*

**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.

### 5.3.6.4 BR

#### Description

The **BR** (Reset Breakpoint) command resets the breakpoint at the specified line or all breakpoints.

#### Syntax

#buffer\_numberBR[line\_number]

#### Arguments

<b>buffer_number</b>	<b>buffer_number</b> specifies the buffer, a number between 0 and 16; or you can use the pound (#) to designate all buffers.
<b>line_number</b>	Optional, if included, the command resets one breakpoint at this line. <b>line_number</b> 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:                                Response
2: VEL(0) = 20000
3: ptp X, 4000
4: till ^MST(0).#MOVE
5: stop
#0BS3
#0L                                         Set breakpoint at line 3
                                             List buffer 0 (note the
                                             asterisk indicating the
                                             breakpoint)
                                             Response

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
                                             Query buffer 0 state
                                             Response
                                             Execute line 3 of buffer 0
                                             Query buffer 0 state
                                             Response
                                             Execute the rest of the
                                             program in buffer 0
                                             Query buffer 0 state
                                             Response

?0
Buffer 0: 5 lines, suspended in line 3
#0XS
?0
Buffer 0: 5 lines, suspended in line 4
#0XD

?0
Buffer 0: 5 lines, terminated in line 5
```

## 5.4 System Commands

System commands provide you with information contained in the system.

### 5.4.1 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: NTM00000A
Controller Part Number: SP+NTM-08000001NNN
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:fffffff
Default Servo Processor Info.
SP3 Program Info:
Monitor version:fffffff
Default Servo Processor Info.
```

## 5.4.2 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:

000 SPiiPlus PCI  
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.
- **HSSI Variables**  
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.

### Comments

All section and key names are case sensitive. For example, if only the Firmware version number is needed, the command would be entered as:

#SIR/Hardware/fw/

### Example

```
#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  
NonACSS stepperAxes = 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 =
:
:
```

### 5.4.3 MEMORY

#### Description

Retrieves RAM availability.

#### Syntax

**#MEMORY**

#### Arguments

None.

#### Return Value

Available RAM.

#### Example

```
#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.
```

### 5.4.4 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 **SFAULT.#INTGR**.

## Example

```
#IR
      Size      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

```

## 5.4.5 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.

## 5.4.6 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

## 5.4.7 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

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	



25	#PROG	Program Error	K	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	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	D
29	#INT	Servo Interrupt	D	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	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	K
1	#LL	Hardware Left Limit	K	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	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	K
6	#SLL	Software Left Limit	K	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	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	D
10	#ENC	Encoder Error	D	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	D
14	#VL	Velocity Limit	K	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	D
17	#SP	Servo Processor Alarm	D	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	K
26	#MEM	Memory Overflow	K	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	D
29	#INT	Servo Interrupt	D	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	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																			

### 5.4.8 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
- PdolIndex

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
```

## 5.4.9 ECMAPREP

### Description

The **ECMAPREP** command displays a report of all variables mapped using the **ECIN**, **ECOUT**, **ECEXTIN**, and **ECEXTOUT** functions.

### 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

```

## 5.4.10 CC

### Description

The **CC** command returns data on the existing communication channels.

### Syntax

**#CC**

### Example

```

#CC
Channel Type      Mode
 1   Serial    Command  Rate:115200 ON 1
 2   Serial    Command  Rate:115200 ON 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

### 5.4.11 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

### 5.4.12 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
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
```

#### **5.4.13 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. 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 6011 and are broken down as follows:

[ACSPL+ Syntax Errors](#) – numbers 1000 to 1999

[ACSPL+ Compilation Errors](#) – numbers 2000 to 2999

[ACSPL+ Runtime Errors](#) – numbers 3000 to 3999

[Motion Termination Errors](#) – numbers 5000 to 5150

[System Errors](#) - numbers 5151 to 5999

[EtherCAT Errors](#) - numbers 6000 to 6999



Error code values of 7000 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.

### 6.1 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.
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.

Error Code	Error Message	Remarks
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.
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"> <li>● The specified index value is more than the number of elements in the array</li> <li>● The specified index value is negative</li> <li>● The specified index values are incompatible</li> </ul>

Error Code	Error Message	Remarks
		(first value in the range greater than last).
1036	Internal error	An internal error has been detected..
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"> <li>● In the <b>Program Manager</b>, right-click the buffer and select <b>Compile Buffer</b>, or</li> <li>● Use the <b>#nC</b> command in the <b>Communication Terminal</b>, where <b>n</b> is the buffer number.</li> </ul>
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"> <li>● In the <b>Program Manager</b>, right-click the buffer and select <b>Stop Buffer</b>, or</li> <li>● Use the <b>#nS</b> command in the <b>Communication Terminal</b>, where <b>n</b> is the buffer number.</li> </ul>

Error Code	Error Message	Remarks
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 <a href="#">POINT</a> command specifies more axes than were specified in the motion that the command refers to.
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.
1056	Parameter MA defines illegal master value	The parameter specifying the master has an illegal value (see <a href="#">MASTER</a> ).

Error Code	Error Message	Remarks
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 PTPV must specify velocity value	The <a href="#">PTP/v</a> 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.
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	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 <a href="#">KILL</a> or <a href="#">KILLALL</a> .
1067	GO command failed	The controller has no motions waiting for the <a href="#">GO</a> 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 be used only with MPTP, PATH or PVSPINE motion	Commands <a href="#">POINT</a> and <a href="#">MPOINT</a> are compatible only with the <a href="#">MPTP...ENDS</a> , <a href="#">PATH...ENDS</a> , or <a href="#">PVSPINE...ENDS</a> 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 <a href="#">ENDS</a> of the <a href="#">MPTP...ENDS</a> , <a href="#">PATH...ENDS</a> , or <a href="#">PVSPINE...ENDS</a> commands.

Error Code	Error Message	Remarks
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 ASCPL+ 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.
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 <b>Protection Wizard in Development Tools</b> .
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 <b>DISABLE</b> 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 <b>DISABLE</b> 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 <b>GO</b> or <b>KILL</b> 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.
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.

Error Code	Error Message	Remarks
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	<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 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.

Error Code	Error Message	Remarks
1108	Invalid real number	The command includes specification of a real number that cannot be interpreted as a valid real.
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 STOP command	The program that runs in a buffer executed the last command and this was not a <b>STOP</b> command.
1115	Stack underflow (RET without CALL)	The program that runs in a buffer caused stack underflow. This occurs if the program executes the <b>RET</b> command that without the paired <b>CALL</b> 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 <b>CALL</b> 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 <b>ON</b> command (see <b>ON...RET</b> ). <b>ON</b> 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.

Error Code	Error Message	Remarks
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 <a href="#">MERR</a> 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.
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	The command attempts to open the buffer for editing or to change the program in a Protected buffer. Editing a buffer is disabled if the controller is in the Protected mode and bit 1 of the corresponding element of <a href="#">PFLAGS</a> is set to 1.
1128	Configuration changed. Commands SAVE and HWRES must be executed.	The program has changed the configuration. The configuration has to be saved to the controller flash and the controller restarted. Select the controller in the Workspace Tree and click the <b>Save to Flash</b> button. Then right-click the controller and select <b>Reboot</b> .
1129	In binary command the name specification must end with /	The command syntax requires a name to be followed by the / (slash) character.

Error Code	Error Message	Remarks
1130	Segment sequence for the previous motion were not terminated with ends command.	Commands <b>MPTP...ENDS</b> , <b>MSEG...ENDS</b> , <b>PATH...ENDS</b> , <b>PVSPLINE...ENDS</b> are followed by a sequence of points or segments. The sequence must terminate with <b>ENDS</b> .
1131	SP program with unknown interface: NORMAL is assumed	SP program interface is unrecognized.
1132	The file is not a legal user data file	The file designator is not valid.
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 <b>Properties</b> , and increasing the <b>Connection Timeout</b> 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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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.
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	This command can be used only with MSEG motion	The command was incorrectly used - it can only be used with <b>MSEG...ENDS</b> .

Error Code	Error Message	Remarks
1155	Motion cannot start because the previous motion failed. Use FCLEAR command.	To start the motion enter the <a href="#">FCLEAR</a> 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 <a href="#">AFLAGS</a> )	The specified axis has been flagged as not to be grouped using <a href="#">AFLAGS</a> .
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.
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	

Error Code	Error Message	Remarks
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	
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.
1201	End-of-Sequence is illegal for this motion	

Error Code	Error Message	Remarks
1211	Too many segments	See <a href="#">MSEG...ENDS</a> .
1212	ARC arguments are inconsistent	See <a href="#">MSEG...ENDS</a> , <a href="#">ARC1</a> and <a href="#">ARC2</a> .
1213	Stopper is prohibited for master-slave motion	See <a href="#">STOPPER</a> .
1214	Adjacent stoppers are prohibited	See <a href="#">STOPPER</a> .
1215	In cyclic path the first and the last points must coincide	
1216	Velocity is specified, but the motion command had no V switch	Velocity argument can be specified in <a href="#">POINT</a> , <a href="#">LINE</a> , <a href="#">ARC1</a> or <a href="#">ARC2</a> only if the corresponding motion command specifies <a href="#">/v</a> .
1217	Segment of zero length	
1221	End-of-Sequence is illegal for this motion	
1231	Illegal key	An invalid <b>Key</b> argument has been entered in either the <a href="#">GETCONF</a> or <a href="#">SETCONF</a> function.
1232	Invalid index	An invalid <b>Index</b> argument has been entered in either the <a href="#">GETCONF</a> or <a href="#">SETCONF</a> function.
1233	Invalid value	The <b>Value</b> argument of the <a href="#">SETCONF</a> command specifies a value that is not compatible with the <b>Key</b> argument.
1234	Value in SETCONF must be 0 or 1	See <a href="#">SETCONF</a> .
1235	SETCONF function is disabled	
1236	SETCONF cannot be executed while the motor is enabled	

Error Code	Error Message	Remarks
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 <a href="#">MFLAGS</a> .
1240	Operation is temporarily disabled	
1241	Operation failed	
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 <b>MFLAGS.#DUMMY</b> (see <a href="#">MFLAGS</a> ).
1254	The operation requires the motor to be enabled	The <a href="#">ENABLE/ENABLEALL</a> command has to be run.
1255	The operation requires the motor to be disabled	The <a href="#">DISABLE</a> 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 <b>Adjuster Wizard</b> 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 <a href="#">ENABLE/ENABLEALL</a> command has to be run.
1261	Motion cannot start because the brushless motor is not commutated	Refer to the chapter on the <b>Adjuster Wizard</b> 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 <a href="#">FCLEAR</a> command.
1265	SP program does not support this operation	
1266	Invalid PEG pulse width	Pulse width set in <a href="#">PEG_I</a> or <a href="#">PEG_R</a> is incorrect.
1267	Maximal number of time-based PEG pulses is 127	Not relevant to PEG operations.
1268	Maximal period of time-based PEG pulses is 0.012 millisecond	Not relevant to PEG operations.

Error Code	Error Message	Remarks
1269	PEG pulse width must be less than time-based pulse period	Not relevant to operations.
1270	PEG is not supported for the specified axis	See <a href="#">ASSIGNPEG</a> .
1271	Step does not agree with start/end points	
1272	Incremental PEG step must be specified in encoder counts within the (-2 <sup>31</sup> , 2 <sup>31</sup> -1) range, zero is excluded.	The PEG step in encoder counts must be in the range of -2 <sup>31</sup> to 2 <sup>31</sup> -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	

Error Code	Error Message	Remarks
1281	No Hall support for 2-Phase motors	
1282	Value out of range	
1283	Random PEG is limited to 256 points per axis at a time	See <a href="#">PEG_R</a> .
1284	Axis is not connected to PEG engine	See <a href="#">ASSIGNPEG</a> .

## 6.2 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"> <li>● All letters are uppercase</li> <li>● Spelling is correct</li> </ul> <p>If the line starts from a user variable name, check the following:</p> <ul style="list-style-type: none"> <li>● The variable is defined before use</li> <li>● The letter case matches the definition</li> <li>● Spelling is correct</li> </ul>
2003	Unexpected delimiter, incomplete command	A command contained in the line is incomplete.
2006	Unexpected END	The <b>END</b> command is specified without a paired <b>IF</b> , <b>LOOP</b> or <b>WHILE</b> command.
2007	Two adjacent operators	Two binary operator are specified in the expression without an operand between.
2008	Left bracket was expected	<p>Left bracket was not found in a position where it is expected.</p> <p>Possible reasons for this error are array name without index or function call without arguments.</p>
2009	Right bracket was expected	The expression or sub-expression has an unpaired left bracket.
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>

Error Code	Error Message	Remarks
		<ul style="list-style-type: none"> <li>● Missing comma between the arguments of the command or function</li> <li>● The command or function call has less arguments than required.</li> </ul>
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.
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	<a href="#">GOTO</a> or <a href="#">CALL</a> 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	<p>The declaration specifies an illegal variable name. A variable name must begin with a letter or underscore and can include letters, digits and underscores. A variable name must not include any of the following:</p> <ul style="list-style-type: none"> <li>● Keywords (<a href="#">IF</a>, <a href="#">WAIT</a>, <a href="#">SIN</a>, etc.) or standard variable names (<a href="#">FPOS</a>, <a href="#">MST</a>, etc.)</li> <li>● Standard label (<a href="#">AUTOEXEC</a>).</li> </ul>

Error Code	Error Message	Remarks
2024	Undefined label	<a href="#">GOTO</a> or <a href="#">CALL</a> 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.
2030	Number user array elements can not exceed the XARRSIZE parameter	The value of the <a href="#">XARRSIZE</a> parameter is limited to a maximum of 30,000 elements. 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.
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.
2036	Function that not returns 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.

Error Code	Error Message	Remarks
		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.
2043	Internal error	
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 <b>SET</b> command must specify assignment to one element of one of the following variables: <b>APOS, RPOS, FPOS, F2POS</b> . The variables must be in upper case.
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.

Error Code	Error Message	Remarks
2057	Only VEL, ACC, DEC, JERK, KDEC are allowed in IMM command	<b>IMM</b> must specify assignment to one element of one of the following variables: <a href="#">VEL</a> , <a href="#">ACC</a> , <a href="#">DEC</a> , <a href="#">JERK</a> , <a href="#">KDEC</a> .
2058	Bit selection cannot be applied to real variable	The bit selector is specified for a real variable.
2059	ELSE without IF	<b>ELSE</b> is specified without corresponding <b>IF</b> (see <a href="#">IF</a> , <a href="#">ELSEIF</a> , <a href="#">ELSE...END</a> ).
2060	ELSEIF without IF	<b>ELSEIF</b> is specified without corresponding <b>IF</b> (see <a href="#">IF</a> , <a href="#">ELSEIF</a> , <a href="#">ELSE...END</a> ).
2061	LOOP without END	<b>LOOP</b> has no corresponding <b>END</b> (see <a href="#">LOOP...END</a> ).
2063	IF without END	<b>IF</b> has no corresponding <b>END</b> (see <a href="#">IF</a> , <a href="#">ELSEIF</a> , <a href="#">ELSE...END</a> ).
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.
2065	Axis constant or axis expression in brackets expected	The command must include an axis specification in parentheses, for example: <b>FMASK(4).16</b> , 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.
2068	Sign constant or sign expression in brackets expected	The command must include a sign specification. Each sign in the specification defines a direction of one involved axis. Sign specification can appear: <ul style="list-style-type: none"> <li>● As a sequence of sign characters, like +, +-+, -+-</li> <li>● As a list of expressions enclosed in brackets where (0, 1, 0) is equivalent to +-+.</li> </ul>
2069	Too many sign specifiers	The sign specification includes too many sign specifiers.

Error Code	Error Message	Remarks
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.
2072	WHILE without END	<b>WHILE</b> has no corresponding <b>END</b> (see <a href="#">WHILE...END</a> ).
2073	WAIT, TILL, GOTO, CALL, LOOP..END, WHILE..END, INPUT are not allowed in immediate command	The specified commands cannot be entered through the SPiiPlus MMI Application Studio <b>Communication Terminal</b> .
2074	Only RPOS is allowed after CONNECT	<b>CONNECT</b> must specify assignment to one element of one of the <b>RPOS</b> variables.
2075	Only MPOS variable is allowed after MASTER	<b>MASTER</b> must specify assignment to one element of one of the <b>MPOS</b> 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.
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 dynamic buffer	Commands <b>GOTO</b> , <b>CALL</b> , <b>LOOP...END</b> , <b>WAIT</b> , <b>IF</b> , <b>ELSEIF</b> , <b>ELSE...END</b> , <b>ON...RET</b> cannot be used in the dynamic buffer.
2080	Local variable declaration is not allowed in a immediate command.	Local variables cannot be used in a SPiiPlus MMI Application Studio <b>Communication Terminal</b> command.

Error Code	Error Message	Remarks
2081	Variable declaration is not allowed in dynamic buffer	Variable declaration cannot be executed in dynamic buffer.
2082	Illegal file name	The command requires file name specification.
2083	Integer overflow	The result of constant integer expression is more than 2147483647 or less than -2147483648. 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.
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).

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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
2095	Illegal line number	The specified line number must be a positive non-zero integer.
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <a href="#">BLOCK...END</a> .
2105	BLOCK without END	See <a href="#">BLOCK...END</a> .

Error Code	Error Message	Remarks
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	
2111	Illegal format of real constant	
2112	This function cannot be used as argument of CONNECT, TRIGGER or in autoroutine condition	See <a href="#">CONNECT</a> , <a href="#">TRIGGER</a> or <a href="#">ON...RET</a> (for the autoroutine structure).
2113	Shared memory variables must be global	
2114	Shared memory variables can be declared only in the D-Buffer	

### 6.3 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 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
3023	Read-only variable cannot be assigned	A command specifies an assignment to a read-only variable.
3028	Scalar variable cannot accept axis specification	A scalar variable cannot be prefixed with an axis specification.
3034	Illegal index value	The command specifies an assignment to a read-only variable.
3035	Index is out of range	The reason of the error is one the following: <ul style="list-style-type: none"> <li>● The specified index value is more or equal to the number of elements in the array</li> <li>● The specified index value is negative</li> <li>● The specified index values are incompatible (first value in the range greater than last).</li> </ul>
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.
3040	Unable to open file	The command specifies a file in the flash memory that does not exist.
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.

Error Code	Error Message	Remarks
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"> <li>● In the <b>Program Manager</b>, right-click the buffer and select <b>Compile Buffer</b>, or</li> <li>● Use the <b>#nC</b> command in the <b>Communication Terminal</b>, where <b>n</b> is the buffer number.</li> </ul>
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"> <li>● In the <b>Program Manager</b>, right-click the buffer and select <b>Stop Buffer</b>, or</li> <li>● Use the <b>#nS</b> command in the <b>Communication Terminal</b>, where <b>n</b> is the buffer number.</li> </ul>
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 <b>POINT</b> command specifies more axes than were specified in the motion that the command refers to.
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.
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..

Error Code	Error Message	Remarks
3053	Wrong type	<p>The command addresses a standard variable of a different type from the variable specified in the command.</p> <p>This error never occurs when the user communicates with the controller through a communication terminal.</p> <p>The error occurs only when an application communicates with the controller using the SPiiPlus C Library. The error indicates a communication problem.</p>
3055	Command requires line number specification	The command must contain a line number specification.
3060	Illegal memory command	The memory management command is improperly formatted.
3061	')' wasn't found	The command contains a non-paired left bracket.
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	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 <a href="#">KILL</a> .
3067	GO command failed	The controller has no motions waiting for <a href="#">GO</a> .
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 be used only with MPTP, PATH, or PVSPLINE motion	The command specifies an axis, but the motion specified for the axis is incompatible with the command.

Error Code	Error Message	Remarks
3070	Attempt to add segment after <b>ENDS</b> command	The command attempts to add a segment or a point to the motion that has already been closed by <b>ENDS</b> .
3071	File name was expected	The command must specify the name of an internal file in the flash memory.
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.
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	The command attempts to assign a variable that can be changed only if the motor is disabled: <ul style="list-style-type: none"> <li>● If the variable is axis-related, disable the corresponding motor before assigning the variable.</li> <li>● If the variable is not axis-related, disable all motors before assigning the variable.</li> </ul>
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., <b>#DEFCON</b> , of the <b>MFLAGS</b> variable (see <a href="#">MFLAGS</a> ).

Error Code	Error Message	Remarks
3081	Incompatible suffixes	The command includes command options that cannot be used together.
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.
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 <a href="#">SLAVE</a> or <a href="#">TRACK</a> .
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.
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.

Error Code	Error Message	Remarks
3099	Extra numbers after the command	The motion command includes superfluous numerical arguments.
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>
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>
3108	Invalid real number	The command includes specification of a real number that cannot be interpreted as a valid real number.
3112	No matches were found	The search command did not find any matches.
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 <b>STOP</b> command.
3115	Stack underflow (RET without CALL)	<p>The program that runs in a buffer caused a stack underflow.</p> <p>Program execution requires <b>CALL</b>.</p>
3116	Stack overflow (too many nested CALLs)	<p>The program that is running in a buffer caused stack overflow.</p> <p>This occurs if the program executes two many nested calls. Check that the program does not specify infinite recursion (subroutine is in an infinite loop).</p>
3117	Attempt to enter autoroutine directly	<p>The program that is running in a buffer comes to the <b>ON</b> command (see <b>ON...RET</b>).</p> <p><b>ON</b> must never be executed in the normal program flow.</p>
3118	Illegal axis number	<p>The command specifies an axis by number or expression, and the resulting value is not a legal axis number.</p> <p>Valid axis numbers range between 0 to the number of axes in the system minus one.</p>

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	<b>ENABLE/ENABLEALL</b> failed. Additional information can be obtained from the corresponding element of the <b>MERR</b> 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 <b>PFLAGS</b> = 1.
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	<b>MPTP, MSEG, PATH</b> are followed by a number of the point or segment commands - segment definition sequence. The segment definition sequence must be closed with ENDS. See <b>MPTP...ENDS, MSEG...ENDS</b> , and <b>PATH...ENDS</b> .
3132	The file is not a legal user data file	<b>READ</b> specifies a file in the flash memory that is not a legal user data file. Only files created with <b>WRITE</b> are legal user data files.

Error Code	Error Message	Remarks
3133	Discrepancy in types of the saved and restored arrays	The array and the user data file specified by <b>READ</b> contain data of different types.  <b>READ</b> must specify an array of the same type and dimension as the array that was specified by the <b>WRITE</b> command that created the file.  For example, the error occurs if a real array was saved in the file by <b>WRITE</b> , but <b>READ</b> tries to load the file into an integer array.
3134	Discrepancy in sizes of the saved and restored arrays	The array and the user data file specified by <b>READ</b> are different sizes.  <b>READ</b> must specify an array of the same type and dimension as the array that was specified by the <b>WRITE</b> command that created the file.
3136	Wrong relation between first point, last point and interval	The interval value specified by <b>PEG_I</b> or <b>PEG_R</b> 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.
3142	Arguments are inconsistent	Check arguments against the specifications for the <b>PEG_I</b> or <b>PEG_R</b> commands.
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).

Error Code	Error Message	Remarks
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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 <b>COPYFROMDPR</b> or <b>COPYTODPR</b> ).
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.
3154	This command can be used only with MSEG motion	Only <b>PROJECTION</b> , <b>LINE</b> , <b>ARC1</b> , <b>ARC2</b> , and <b>STOPPER</b> commands apply to <b>MSEG...ENDS</b> .
3155	Motion cannot start because the previous motion failed. Use FCLEAR command.	In the Strict mode ( <b>S_FLAGS.#FCLEAR = 1</b> ) the motion cannot start after a fault has occurred. Use the command <b>FCLEAR</b> to clear the result of the fault.
3201	End-of-Sequence is illegal for this motion	The ends command cannot be specified for this type of motion.
3212	ARC arguments are inconsistent	<b>ARC1</b> or <b>ARC2</b> specify inconsistent arguments. The desired arc cannot be calculated.
3213	Stopper is prohibited for master-slave motion	<b>STOPPER</b> cannot be used with segmented <b>MASTER - SLAVE</b> motion.
3214	Adjacent stoppers are prohibited	Two adjacent <b>STOPPER</b> commands are not allowed.

Error Code	Error Message	Remarks
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 <a href="#">POINT</a> , <a href="#">LINE</a> , <a href="#">ARC1</a> or <a href="#">ARC2</a> only if the corresponding motion command specifies /v.
3217	Segment of zero length	Segments of zero length are not allowed.
3231	Illegal key	The <b>Key</b> argument of the <a href="#">GETCONF</a> or <a href="#">SETCONF</a> command specifies a value that is not supported in this controller.
3232	Illegal index	The <b>Index</b> argument of the <a href="#">GETCONF</a> or <a href="#">SETCONF</a> command specifies a value that is not supported in this controller.
3233	Illegal value	The <b>Value</b> argument of the <a href="#">SETCONF</a> command specifies a value that is not compatible with the <b>Key</b> argument.
3234	Value in SETCONF must be 0 or 1	For the specified <b>Key</b> only 0 or 1 are legal in the <b>Value</b> argument of <a href="#">SETCONF</a>
3235	SETCONF function is disabled	The specified <a href="#">SETCONF</a> function is disabled in the current controller mode.
3236	SETCONF cannot be executed while the motor is enabled	The specified <a href="#">SETCONF</a> function cannot be executed while the motor is enabled.
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 <b>#DUMMY</b> bit of <a href="#">MFLAGS</a> ).
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.

Error Code	Error Message	Remarks
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 <b>#BRUSHOK</b> bit of <b>MFLAGS</b> reflects the state of the brushless commutation. To commutate the motor, execute <b>COMMUT</b> 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 <b>#OPEN</b> bit of <b>MFLAGS</b> .
3259	Motion cannot start because the motor is defined as dummy	The motion command cannot be executed because one or more of the motors involved are dummy. Motor dummy state is defined by the <b>#DUMMY</b> bit of <b>MFLAGS</b> .
3260	Motion cannot start because the motor is disabled	The motion command cannot be executed because one or more of the motors involved are disabled.
3261	Motion cannot start because the brushless motor is not commutated	The command cannot be executed because the defined brushless motor is not commutated. The <b>#BRUSHOK</b> bit of <b>MFLAGS</b> reflects the state of the brushless commutation. To commutate the motor, execute <b>COMMUT</b> or a specific initialization program.
3262	Motion cannot start because the motor is in open-loop mode	The motion command cannot be executed because one or more of the motors involved are in open-loop state. Open-loop state is defined by bit <b>MST.#OPEN</b> . The open-loop mode may be set using the <b>#OPEN</b> bit of <b>MFLAGS</b> .

Error Code	Error Message	Remarks
3263	Motion cannot start because the previous motion failed. Use FCLEAR command.	The motion command cannot execute because the previous motion for one or more motors failed and the controller is set to the Strict mode. The Strict mode is defined by bit <a href="#">S_FLAGS.#FCLEAR</a> . Use <a href="#">FCLEAR</a> to clear the result of the fault.
3265	SP program does not support this operation	The command is not supported by the current version of the controller. Specifically, the SP program does not support the operation. Check if the SP program has been changed.

## 6.4 Motion Termination 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.
5003	Motion was terminated by user	The motion was terminated by <a href="#">HALT</a> before the final point was reached
5004	Motor was disabled by user	The motion was disabled by <a href="#">DISABLE</a> .
5005	Motion was terminated because a motor was disabled	The motion was disabled by <a href="#">DISABLE</a> .
5006	Motion was killed	The motion was terminated by <a href="#">KILL</a> 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 <a href="#">MERR</a> 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 <a href="#">MERR</a> variable for all affected axes.
5010	Right Limit	The motion was killed because of a right limit fault.
5011	Left Limit	The motion was killed because of a left limit fault.
5014	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 Not Connected	The motor was disabled because of an encoder not connected fault.
5018	Encoder 2 Not Connected	The motor was disabled because of an encoder 2 not connected fault.
5019	Drive Alarm	The motor was disabled because of a drive alarm fault.
5020	Encoder Error	The motor was disabled or the motion failed because of an encoder error fault.
5021	Encoder 2 Error	The motor was disabled or the motion failed because of an encoder 2 error fault.
5022	Position 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.
5025	Acceleration Limit	The motor was disabled or the motion failed because of an acceleration limit fault.
5026	Overcurrent	The motor was disabled or the motion failed because of an overcurrent fault.

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>  <p>Activated when an axes does not have a physical drive associated to it.</p>
5028	Failed to enable motor	The motor was disabled because STO not connected.
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.
5035	Program Error	The motor was disabled or the motion failed because of a program error fault.
5036	Memory Overuse	The motor was disabled or the motion failed because of a memory overuse fault.
5037	Time Overuse	The motor was disabled or the motion failed because of a time overuse fault.
5038	Emergency Stop	The motor was disabled because of an emergency stop fault.
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
5044	External Network Error	External network error
5060	Driver Alarm	Drive alarm: No fault
5061	Drive Alarm: Short Circuit	The motor was disabled or the motion failed because of a short circuit condition in the drive.

Error Code	Error Message	Remarks
5062	Drive Alarm: External Protection Activated	The motor was disabled or the motion failed because of an external protection condition in the drive.
5063	Drive 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	Drive 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	Drive Alarm: Temperature Too High	The motor was disabled or the motion failed because of a power supply too high condition in the drive.
5066	Drive Alarm: Power Supply 24VF1	The motor was disabled or the motion failed because of a power supply 24VF1 condition in the drive.
5067	Drive Alarm: Power Supply 24VF2	The motor was disabled or the motion failed because of a power supply 24VF2 condition in the drive.
5068	Drive Alarm: Emergency Stop	The motor was disabled or the motion failed because of an emergency stop condition in the drive.
5069	Drive alarm: Power-down	The motor was disabled or the motion failed because of a power-down condition in the drive.
5070	Phase lost.	
5071	Driver not ready (power up).	
5072	Over-current.	
5073	Not in use (reserved).	
5074	Damper is not ok.	
5075	Drive 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.

Error Code	Error Message	Remarks
5080	Component Failure	
5081	Component Failure: Unknown error	
5082	Component Failure: Unknown error	
5083	Component Failure: Unknown error	
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.

Error Code	Error Message	Remarks
		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	
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	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.  If the unit has STO installed, check drive supply voltage.
5101	Autocommutation failure (phase error)	Error occurs if <b>COMMUT</b> 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 <b>COMMUT</b> is executed.
5102	Autocommutation failure (position error)	Error occurs if <b>COMMUT</b> 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 <b>Adjuster Wizard</b> or <b>FRF Analyzer</b> .

Error Code	Error Message	Remarks
5103	ENABLE is prohibited while Constant Current is ON	
5104	ENABLE failed because the motor is moving	<b>ENABLE</b> cannot be executed whenever the motor is in motion (see <a href="#">ENABLE/ENABLEALL</a> ).
5105	E_TYPE does not match HSSI-HES DIP switch settings	Check the HSSI-HES DIP switch settings.
5106	EtherCAT node failure	

## 6.5 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 6-5. 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
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. 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
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

Error Code	Error Message
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
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 the configuration

Error Code	Error Message
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
5206	Initialization problem: EtherCAT error. Servo Processor program activation failed
5207	Initialization problem: EtherCAT timeout. Servo Processor program activation failed

## 6.6 EtherCAT Errors

EtherCAT errors range from 6000 to 6999 and are latched in the [ECERR](#) variable.

Table 6-6. ACSPL+ EtherCAT Errors

Error Code	Error Message	Remarks
6000	General EtherCAT Error	Appears for any unspecified EtherCAT error. In general, it rarely appears.
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.

Error Code	Error Message	Remarks
6006	EtherCAT cannot complete the operation	The bus scan could not be completed. This implies hardware level problems in the EtherCAT network.
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 operational	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 Slave 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.

Error Code	Error Message	Remarks
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.

# Smarter Motion

1 Hataasia St  
Ramat Gabriel Industrial Park  
Migdal Ha'Emek 2307037 Israel  
Tel: (+972) (4) 654 6440 Fax: (+972) (4) 654 6443

Contact us: [sales@acsmotioncontrol.com](mailto:sales@acsmotioncontrol.com) | [www.acsmotioncontrol.com](http://www.acsmotioncontrol.com)

