EPSON

Epson RC+ 8.0 Option
RC+ API 8.0

Original instructions

© Seiko Epson Corporation 2024-2026

Rev.6
ENM269S9100F

Table of Contents

1. FOREWORD 20

1.1 FOREWORD 21

1.2 TRADEMARKS 21

1.3 TRADEMARK NOTATION IN THIS MANUAL 21

1.4 NOTICE 21

1.5 MANUFACTURER 21

1.6 CONTACT INFORMATION 21

1.7 Before Use 21

2. RC+ API 23

2.1 Features 24

3. Installation 26

3.1 Step by step instructions 27

3.2 What's installed 27

4. Getting Started 28

4.1 Getting Started 29

4.2 Getting started using Visual Basic 29

4.3 Getting started using Visual C# 30

4.4 Getting started using Visual C++ 31

5. Environments 35

5.1 Development Environment 36

5.1.1 Development Startup 36

5.1.2 Spel Class Instance Initialization 36

5.1.3 Spel Class Instance Termination 36

5.1.4 Development Cycle 36

5.2 In Production Facilities 37

5.2.1 Opening Epson RC+ 8.0 at Runtime 37

5.2.2 Using Epson RC+ 8.0 Dialogs and Windows 37

5.2.3 Installation on Target System 37

6. Executing Methods, Programs, Tasks 38

6.1 Executing Methods 39

6.1.1 Using Multiple Threads 39

6.2 Executing SPEL+ Programs 41

6.3 Executing SPEL+ Tasks 42

6.4 Aborting All Tasks 42

7. Events 44

7.1 Overview 45

7.2 System Events 45

7.3 User Events from SPEL+ 45

8. Error Handling 47

8.1 Errors for Spel methods 48

9. Handling Pause and Continue 49

9.1 Pause state 50

9.2 Catching the Pause event 50

9.3 Executing Pause 50

9.4 Continue after pause 51

9.5 Abort after pause 51

10. Handling Emergency Stop 52

10.1 Using system EStop events 53

11. EpsonRC+8.0 Windows and Dialogs 54

11.1 Windows 55

11.2 Dialogs 56

12. Displaying Video 57

12.1 Displaying Video 58

12.2 Using multiple video displays 62

13. Using AsyncMode 63

13.1 Using AsyncMode 64

14. SPELCom_Event 66

14.1 SPELCom_Event 67

15. RCAPINet Reference 68

15.1 Spel Class 69

15.2 Spel Class Properties 70

15.2.1 AsyncMode Property 70

15.2.2 AvoidSingularity Property 71

15.2.3 CommandInCycle Property 72

15.2.4 CommandTask Property 73

15.2.5 DisableMsgDispatch Property 74

15.2.6 ErrorCode Property 75

15.2.7 ErrorOn Property 76

15.2.8 EStopOn Property 77

15.2.9 MotorsOn Property 78

15.2.10 NoProjectSync Property 79

15.2.11 OperationMode Property 80

15.2.12 ParentWindowHandle Property 81

15.2.13 PauseOn Property 82

15.2.14 PowerHigh Property 83

15.2.15 Project Property 84

15.2.16 ProjectBuildComplete Property 85

15.2.17 ProjectOverwriteWarningEnabled Property 86

15.2.18 ResetAbortEnabled Property 87

15.2.19 Robot Property 88

15.2.20 RobotModel Property 89

15.2.21 RobotType Property 90

15.2.22 SafetyOn Property 91

15.2.23 ServerInstance Property 92

15.2.24 SPELVideoControl Property 93

15.2.25 Version Property 94

15.2.26 WarningCode Property 95

15.2.27 WarningOn Property 96

15.3 Spel Class Methods 97

15.3.1 A 97

15.3.1.1 Accel Method 97

15.3.1.2 AccelR Method 98

15.3.1.3 AccelS Method 99

15.3.1.4 Agl Method 100
15.3.1.5 AIO_In Method 101
15.3.1.6 AIO_InW Method 102
15.3.1.7 AIO_Out Method 103
15.3.1.8 AIO_OutW Method 104
15.3.1.9 Arc Method 105
15.3.1.10 Arc3 Method 106
15.3.1.11 Arch Method 107
15.3.1.12 Arm Method 108
15.3.1.13 ArmClr Method 109
15.3.1.14 ArmDef Method 110
15.3.1.15 ArmSet Method 111
15.3.1.16 Atan Method 112
15.3.1.17 Atan2 Method 113
15.3.1.18 ATCLR Method 114
15.3.1.19 AtHome Method 115
15.3.1.20 ATRQ Method 116
15.3.1.21 AvgSpeed Method 117
15.3.1.22 AvgSpeedClear Method 118
15.3.1.23 AxisLocked Method 119
15.3.2 B 120
15.3.2.1 Base Method 120
15.3.2.2 BGo Method 121
15.3.2.3 BMove Method 123
15.3.2.4 Box Method 125
15.3.2.5 BoxClr Method 126
15.3.2.6 BoxDef Method 127
15.3.2.7 Brake Method 128
15.3.2.8 BTst Method 129
15.3.2.9 .BuildProject Method 130
15.3.3 C 131
15.3.3.1 Call Method 131
15.3.3.2 CalPls Method 133
15.3.3.3 ClearPoints Method 135
15.3.3.4 Connect Method 136

15.3.3.5 Continue Method 137
15.3.3.6 Clr Method 138
15.3.3.7 CtReset Method 139
15.3.3.8 Curve Method 140
15.3.3.9 CVMove Method 142
15.3.3.10 CX, CY, CZ, CU, CV, CW, CR, CS, CT Method 143
15.3.4 D 144
15.3.4.1 Delay Method 144
15.3.4.2 DegToRad Method 145
15.3.4.3 Disconnect Method 146
15.3.5 E 147
15.3.5.1 ECP Method 147
15.3.5.2 ECPClr Method 148
15.3.5.3 ECPDef Method 149
15.3.5.4 ECPSet Method 150
15.3.5.5 EnableEvent Method 151
15.3.5.6 ExecuteCommand Method 152
15.3.6 F 153
15.3.6.1 FBusIO_GetBusStatus Method 153
15.3.6.2 FBusIO_GetDeviceStatus Method 154
15.3.6.3 FBusIO_SendMsg Method 155
15.3.6.4 FGGet Method 157
15.3.6.5 FGRun Method 158
15.3.6.6 Find Method 160
15.3.6.7 Fine Method 161
15.3.7 G 162
15.3.7.1 GetAccel Method 162
15.3.7.2 GetArm Method 163
15.3.7.3 GetConnectionInfo Method 164
15.3.7.4 GetControllerInfo Method 165
15.3.7.5 GetCurrentConnectionInfo Method 166
15.3.7.6 GetCurrentUser Method 167
15.3.7.7 GetECP Method 168
15.3.7.8 GetErrorMessage Method 169
15.3.7.9 GetIODef Method 170

15.3.7.10 GetJRange Method 171
15.3.7.11 GetLimitTorque Method 172
15.3.7.12 GetLimZ Method 173
15.3.7.13 GetPoint Method 174
15.3.7.14 GetRealTorque Method 175
15.3.7.15 GetRobotInfo Method 176
15.3.7.16 GetRobotPos Method 177
15.3.7.17 GetSpeed Method 178
15.3.7.18 GetTaskInfo Method 179
15.3.7.19 GetTool Method 180
15.3.7.20 GetVar Method 181
15.3.7.21 Go Method 182
15.3.8 H 184
15.3.8.1 Halt Method 184
15.3.8.2 Here Method 185
15.3.8.3 HideWindow Method 186
15.3.8.4 Home Method 187
15.3.8.5 Hofs Method 188
15.3.8.6 HomeSet Method 190
15.3.8.7 Hordr Method 191
15.3.8.8 Hour Method 192
15.3.9 I 193
15.3.9.1 ImportPoints Method 193
15.3.9.2 In Method 194
15.3.9.3 InBCD Method 195
15.3.9.4 Inertia Method 196
15.3.9.5 Initialize Method 197
15.3.9.6 InReal Method 198
15.3.9.7 InsideBox Method 199
15.3.9.8 InsidePlane Method 200
15.3.9.9 InW Method 201
15.3.9.10 IsOptionActive Method 202
15.3.10 J 203
15.3.10.1 Jrange Method 203
15.3.10.2 JS Method 204

15.3.10.3 JTran Method 205
15.3.10.4 Jump Method 206
15.3.10.5 Jump3 Method 208
15.3.10.6 Jump3CP Method 210
15.3.11 L 212
15.3.11.1 LimitTorque Method 212
15.3.11.2 LimZ Method 213
15.3.11.3 LoadPoints Method 214
15.3.11.4 Local Method 215
15.3.11.5 LocalClr Method 216
15.3.11.6 LocalDef Method 217
15.3.11.7 Login Method 218
15.3.12 M 219
15.3.12.1 MCal Method 219
15.3.12.2 MCalComplete Method 220
15.3.12.3 Mcordr Method 221
15.3.12.4 MemIn Method 222
15.3.12.5 MemInW Method 223
15.3.12.6 MemOff Method 224
15.3.12.7 MemOn Method 225
15.3.12.8 MemOut Method 226
15.3.12.9 MemOutW Method 227
15.3.12.10 MemSw Method 228
15.3.12.11 Move Method 229
15.3.13 O 231
15.3.13.1 Off Method 231
15.3.13.2 OLRate Method 232
15.3.13.3 On Method 233
15.3.13.4 OpBCD Method 234
15.3.13.5 Oport Method 235
15.3.13.6 Out Method 236
15.3.13.7 OutReal Method 237
15.3.13.8 OutW Method 238
15.3.14 P 239
15.3.14.1 PAgl Method 239

15.3.14.2 Pallet Method 240
15.3.14.3 Pass Method 241
15.3.14.4 Pause Method 242
15.3.14.5 PDef Method 243
15.3.14.6 PDel Method 244
15.3.14.7 PeakSpeed Method 245
15.3.14.8 PeakSpeedClear Method 246
15.3.14.9 PF_Abort Method 247
15.3.14.10 PF_Backlight Method 248
15.3.14.11 PF_BacklightBrightness Method 249
15.3.14.12 PF_Name Method 250
15.3.14.13 PF_Number Method 251
15.3.14.14 PF_Start Method 252
15.3.14.15 PF_Stop Method 253
15.3.14.16 PLabel Method 254
15.3.14.17 Plane Method 255
15.3.14.18 PlaneClr Method 256
15.3.14.19 PlaneDef Method 257
15.3.14.20 Pls Method 258
15.3.14.21 PTCLR Method 259
15.3.14.22 PTPBoost Method 260
15.3.14.23 PTPBoostOK Method 261
15.3.14.24 PTran Method 262
15.3.14.25 PTRQ Method 263
15.3.14.26 Pulse Method 264
15.3.15 Q 265
15.3.15.1 Quit Method 265
15.3.16 R 266
15.3.16.1 RadToDeg Method 266
15.3.16.2 RebootController Method 267
15.3.16.3 RebuildProject Method 268
15.3.16.4 Recover Method 269
15.3.16.5 Reset Method 271
15.3.16.6 ResetAbort Method 272
15.3.16.7 Resume Method 273

15.3.16.8 RunDialog Method 274
15.3.17 S 275
15.3.17.1 SavePoints Method 275
15.3.17.2 Sense Method 276
15.3.17.3 SetIODef Method 277
15.3.17.4 SetPoint Method 278
15.3.17.5 SetVar Method 281
15.3.17.6 SFree Method 282
15.3.17.7 ShowWindow Method 283
15.3.17.8 SimGet Method 284
15.3.17.9 SimResetCollision Method 285
15.3.17.10 SimSet Method 286
15.3.17.11 SimSetParent Method 287
15.3.17.12 SimSetPick Method 288
15.3.17.13 SimSetPlace Method 289
15.3.17.14 Shutdown Method 290
15.3.17.15 Slock Method 291
15.3.17.16 Speed Method 292
15.3.17.17 SpeedR Method 293
15.3.17.18 SpeedS Method 294
15.3.17.19 Start Method 295
15.3.17.20 StartBGTask Method 296
15.3.17.21 Start Method 297
15.3.17.22 Stop Method 299
15.3.17.23 Sw Method 300
15.3.18 T 301
15.3.18.1 TargetOK Method 301
15.3.18.2 TasksExecuting Method 302
15.3.18.3 TaskState Method 303
15.3.18.4 TeachPoint Method 304
15.3.18.5 Till Method 305
15.3.18.6 TillOn Method 306
15.3.18.7 TGo Method 307
15.3.18.8 TLClr Method 309
15.3.18.9 TLDef Method 310

15.3.18.10 TLSet Method 311
15.3.18.11 TMove Method 312
15.3.18.12 Tool Method 314
15.3.18.13 TrapStop Method 315
15.3.18.14 TW Method 316
15.3.19 U 317
15.3.19.1 UserHasRight Method 317
15.3.20 V 318
15.3.20.1 VCal Method 318
15.3.20.2 VCalPoints Method 319
15.3.20.3 VCls Method 320
15.3.20.4 VCreateCalibration Method 321
15.3.20.5 VCreateObject Method 322
15.3.20.6 VCreateSequence Method 324
15.3.20.7 VDefArm Method 325
15.3.20.8 VDefGetMotionRange Method 327
15.3.20.9 VDefLocal Method 328
15.3.20.10 VDefSetMotionRange Method 330
15.3.20.11 VDefTool Method 331
15.3.20.12 VDefToolXYZ Method 333
15.3.20.13 VDefToolXYZUVW Method 335
15.3.20.14 VDeleteCalibration Method 336
15.3.20.15 VDeleteObject Method 337
15.3.20.16 VDeleteSequence Method 338
15.3.20.17 VEditWindow Method 339
15.3.20.18 VGet Method 340
15.3.20.19 VGetCameraXYU Method 342
15.3.20.20 VGetEdgeCameraXYU Method 343
15.3.20.21 VGetEdgePixelXYU Method 345
15.3.20.22 VGetEdgeRobotXYU Method 347
15.3.20.23 VGetExtrema Method 349
15.3.20.24 VGetModelWin Method 351
15.3.20.25 VGetPixelToCamera Method 352
15.3.20.26 VGetPixelToRobot Method 353
15.3.20.27 VGetPixelXYU Method 354

15.3.20.28 VGetRobotPlacePos Method 355
15.3.20.29 VGetRobotPlaceTargetPos Method 356
15.3.20.30 VGetRobotXYU Method 357
15.3.20.31 VGetRobotToolXYU Method 358
15.3.20.32 VGetSearchWIn Method 360
15.3.20.33 VGoCenter Method 361
15.3.20.34 VLoad Method 362
15.3.20.35 VLoadModel Method 363
15.3.20.36 VRun Method 364
15.3.20.37 Vsave Method 365
15.3.20.38 VSaveImage Method 366
15.3.20.39 VSaveModel Method 367
15.3.20.40 VSet Method 368
15.3.20.41 VSetModelWin Method 369
15.3.20.42 VSetRobotPlaceTargetPos Method 370
15.3.20.43 VSetSearchWIn Method 371
15.3.20.44 VShowModel Method 372
15.3.20.45 VShowSequence Method 373
15.3.20.46 VStatsReset Method 374
15.3.20.47 VStatsResetAll Method 375
15.3.20.48 VStatsSave Method 376
15.3.20.49 VStatsShow Method 377
15.3.20.50 VTeach Method 378
15.3.20.51 VTrain Method 379
15.3.21 W 382
15.3.21.1 WaitCommandComplete method. 382
15.3.21.2 WaitMem Method 383
15.3.21.3 WaitSw Method 384
15.3.21.4 WaitTaskDone Method 385
15.3.21.5 Weight Method 386
15.3.22 X 387
15.3.22.1 Xqt Method 387
15.3.22.2 XYLim Method 388
15.3.22.3 XYLimClr Method 389
15.3.22.4 XYLimDef Method 390

15.4 Spel Class Events 391
15.4.1 EventReceived Event 391
15.5 SPELVideo Control 395
15.6 SPELVideo Control Properties 396
15.6.1 Camera Property 397
15.6.2 GraphicsEnabled Property 398
15.6.3 VideoEnabled Property 399
15.7 SPELVideo Control Methods 400
15.7.1 LoadImage Method 400
15.8 SPELVideo Control Events 401
15.9 SpelConnectionInfo Class 402
15.10 SpelControllerInfo Class 403
15.11 SpelException Class 404
15.11.1 SpelException Properties 404
15.11.2 SpelException Methods 404
15.12 SpelOptionInfo Class 405
15.13 SpelPoint Class 406
15.13.1 SpelPoint Properties 407
15.13.2 SpelPoint Methods 408
15.14 SpelRobotInfo Class 409
15.15 SpelTaskInfo Class 410
15.16 Enumerations 411
15.16.1 SpelArmDefMode Enumeration 411
15.16.2 SpelArmDefType Enumeration 411
15.16.3 SpelAxis Enumeration 411
15.16.4 SpelBaseAlignment Enumeration 411
15.16.5 SpelCalPlateType Enumeration 411
15.16.6 SpelConnectionType Enumeration 412
15.16.7 SpelDialogs Enumeration 412
15.16.8 SpelElbow Enumeration 412
15.16.9 SpelEvents Enumeration 412
15.16.10 SpelForceCompareType Enumeration 413
15.16.11 SpelForceProps Enumeration 413
15.16.12 SpelHand Enumeration 415
15.16.13 SpelIOLabelTypes Enumeration 415

15.16.14 SpelLocalDefType Enumeration 415
15.16.15 SpelOperationMode Enumeration 416
15.16.16 SpelOptions Enumeration 416
15.16.17 SpelOptionStatus Enumeration 416
15.16.18 SpelRobotPosType Enumeration 416
15.16.19 SpelRobotType Enumeration 417
15.16.20 SpelShutdownMode Enumeration 417
15.16.21 SpelSimObjectType Enumeration 417
15.16.22 SpelSimProps Enumeration 417
15.16.23 SpelStopType Enumeration 418
15.16.24 SpelTaskState Enumeration 418
15.16.25 SpelTaskType Enumeration 419
15.16.26 SpelToolDefType Enumeration 419
15.16.27 SpelToolDefType3D Enumeration 419
15.16.28 SpelUserRights Enumeration 419
15.16.29 SpelVDefShowWarning Enumeration 420
15.16.30 SpelVisionImageSize Enumeration 420
15.16.31 SpelVisionObjectTypes Enumeration 421
15.16.32 SpelVisionProps Enumeration 421
15.16.33 SpelWindows Enumeration 422
15.16.34 SpelWrist Enumeration 422
15.17 Spel Error Numbers and Messages 423

16. 32 Bit and 64 Bit Applications 425

17. Using the LabVIEW VI Library 426

17.1 Overview 427
17.2 Installation 427
17.3 Tool and Control Palettes 429
17.4 Getting started 430
17.5 Working with a Spel+ project 432
17.6 Displaying Video 432
17.7 VI Reference 434
17.7.1 A 435
17.7.1.1 Accel VI 435
17.7.1.2 AccelS VI 436

17.7.1.3 Arc VI 437
17.7.1.4 Arc3 VI 438
17.7.1.5 Arch VI 439
17.7.1.6 Arm VI 440
17.7.1.7 Armset VI 441
17.7.1.8 AtHome VI 442
17.7.1.9 AvoidSing VI 443
17.7.2 B 444
17.7.2.1 BGo VI 444
17.7.2.2 BMove VI 445
17.7.2.3 Box VI 446
17.7.3 C 447
17.7.3.1 Continue VI 447
17.7.4 D 448
17.7.4.1 Delay VI 448
17.7.5 E 449
17.7.5.1 ECP VI 449
17.7.5.2 ECPset VI 450
17.7.5.3 EStopOn VI 451
17.7.6 F 452
17.7.6.1 Find VI 452
17.7.6.2 Fine VI 453
17.7.7 G 454
17.7.7.1 GetArm VI 454
17.7.7.2 GetAvoidSing VI 455
17.7.7.3 GetECP VI 456
17.7.7.4 GetLimZ VI 457
17.7.7.5 GetMotor VI 458
17.7.7.6 GetOprMode VI 459
17.7.7.7 GetPoint VI 460
17.7.7.8 GetPower VI 461
17.7.7.9 GetRobot VI 462
17.7.7.10 GetTool VI 463
17.7.7.11 GetVar VI 464
17.7.7.12 Go VI 465

17.7.8 H 466
17.7.8.1 Halt VI 466
17.7.8.2 Here VI 467
17.7.8.3 HideWindow VI 468
17.7.9 I 469
17.7.9.1 In VI 469
17.7.9.2 InBCD VI 470
17.7.9.3 InsideBox VI 471
17.7.9.4 InsidePlane VI 472
17.7.9.5 InW VI 473
17.7.9.6 Inertia VI 474
17.7.9.7 Initialize VI 475
17.7.10 J 476
17.7.10.1 JRange VI 476
17.7.10.2 JS VI 477
17.7.10.3 JTran VI 478
17.7.10.4 Jump VI 479
17.7.10.5 Jump3 VI 480
17.7.10.6 Jump3CP VI 481
17.7.11 L 482
17.7.11.1 LimZ VI 482
17.7.11.2 LoadPoints VI 483
17.7.12 M 484
17.7.12.1 MemIn VI 484
17.7.12.2 MemInW VI 485
17.7.12.3 MemOut VI 486
17.7.12.4 MemOff VI 487
17.7.12.5 MemOn VI 488
17.7.12.6 MemOut VI 489
17.7.12.7 MemOutW VI 490
17.7.12.8 MemSw VI 491
17.7.12.9 MotorOff VI 492
17.7.12.10 MotorOn VI 493
17.7.12.11 Move VI 494

17.7.13 O 495
17.7.13.1 Off VI 495
17.7.13.2 On VI 496
17.7.13.3 OPort VI 497
17.7.13.4 OprMode VI 498
17.7.13.5 Out VI 499
17.7.13.6 OutW VI 500
17.7.14 P 501
17.7.14.1 Pause VI 501
17.7.14.2 Plane VI 502
17.7.14.3 PowerHigh VI 503
17.7.14.4 PowerLow VI 504
17.7.15 Q 505
17.7.15.1 Quit VI 505
17.7.16 R 506
17.7.16.1 Reset VI 506
17.7.16.2 Resume VI 507
17.7.16.3 Robot VI 508
17.7.16.4 RunDialog VI 509
17.7.17 S 510
17.7.17.1 SafetyOn VI 510
17.7.17.2 SavePoints VI 511
17.7.17.3 Sense VI 512
17.7.17.4 SetPoint VI 513
17.7.17.5 SetVar VI 514
17.7.17.6 SFree VI 515
17.7.17.7 ShowWindow VI 516
17.7.17.8 Shutdown VI 517
17.7.17.9 SLock VI 518
17.7.17.10 Speed VI 519
17.7.17.11 SpeedS VI 520
17.7.17.12 Start VI 521
17.7.17.13 Stop VI 522
17.7.17.14 Sw VI 523

17.7.18 T 524
17.7.18.1 TargetOK VI 524
17.7.18.2 TGo VI 525
17.7.18.3 Till VI 526
17.7.18.4 TillOn VI 527
17.7.18.5 TLSet VI 528
17.7.18.6 TMove VI 529
17.7.18.7 Tool VI 530
17.7.18.8 TW VI 531
17.7.19 V 532
17.7.19.1 VGetBool VI 532
17.7.19.2 VGetDbl VI 533
17.7.19.3 VGetInt VI 534
17.7.19.4 VGetStr VI 535
17.7.19.5 VideoControl VI 536
17.7.19.6 VRun VI 537
17.7.19.7 VSetBool VI 538
17.7.19.8 VSetDbl VI 539
17.7.19.9 VSetInt VI 540
17.7.19.10 VSetStr VI 541
17.7.20 W 542
17.7.20.1 WaitMem VI 542
17.7.20.2 WaitSw VI 543
17.7.20.3 WaitTaskDone VI 544
17.7.20.4 Weight VI 545
17.7.21 X 546
17.7.21.1 Xqt VI 546
17.7.21.2 XYLim VI 547

18. Using LabVIEW with RCAPINt2.dll 549

18.1 Overview 550
18.2 Initialization 550
18.2.1 Add a constructor node for the Spel class 550
18.2.2 Initialize the Spel class instance 550
18.2.3 Connect to Controller and set project 551
18.3 Use Spel properties and methods 551

18.4 Shutdown 551
18.5 Using dialogs and windows 551

19. How to Control Multiple Controllers from One PC 552

19.1 Overview 553
19.1.1 System Requirements 553
19.1.2 Connection of PC and Controllers 554
19.2 Restrictions on controlling multiple Controllers 555
19.2.1 Restrictions on Controller options 555
19.2.2 Restrictions on simulator 555
19.3 Sample Program for connecting multiple Controllers 555
19.3.1 Controller connection setting 555
19.3.2 Project setting 556
19.3.3 Sample program using Visual Basic 556
19.3.4 Sample program using Visual C# 558

1. FOREWORD

1.1 FOREWORD

Thank you for purchasing our robot products.
This manual contains the information necessary for the correct use of the Manipulator.
Please carefully read this manual and other related manuals before installing the robot system.
Keep this manual handy for easy access at all times.

The robot system and its optional parts are shipped to our customers only after being subjected to the strictest quality controls, tests, and inspections to certify its compliance with our high performance standards. Please note that the basic performance of the product will not be exhibited if our robot system is used outside of the usage conditions and product specifications described in the manuals.

This manual describes possible dangers and consequences that we can foresee. Be sure to comply with safety precautions on this manual to use our robot system safety and correctly.

1.2 TRADEMARKS

Microsoft, Windows, and Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brand and product names are trademarks or registered trademarks of the respective holders.

1.3 TRADEMARK NOTATION IN THIS MANUAL

Microsoft® Windows® 10 operating system
Microsoft® Windows® 11 operating system
Throughout this manual, Windows 10 and Windows 11 refer to above respective operating systems. In some cases, Windows refers generically to Windows 10 and Windows 11.

1.4 NOTICE

No part of this manual may be copied or reproduced without authorization.
The contents of this manual are subject to change without notice.
Please notify us if you should find any errors in this manual or if you have any comments regarding its contents.

1.5 MANUFACTURER

SEIKO EPSON CORPORATION

1.6 CONTACT INFORMATION

For detailed contact information, see "SUPPLIER" of the manual below.
"Safety Manual"

1.7 Before Use

Before using this manual, be sure that you understand the following information.

The Installation Folder for Epson RC+ 8.0

You can change the path for the installation folder for Epson RC+ 8.0 anywhere. This manual assumes that Epson RC+ 8.0 is installed in C:\EpsonRC80.

2. RC+ API

The Epson RC+ 8.0 Option RC+ API enables you to use Visual Basic or any other language that supports .NET technology to run your robotic applications. This gives you the power to create sophisticated user interfaces, use databases, and use third party products designed for use with .NET.

A LabVIEW library is also included.

.NET6 and .NET8 can only be used with “Visual Basic”, or Visual C#.

.NET6 and .NET8 are available only in Visual Studio 2022.

2.1 Features

The following features are supported in the RC+ API package:

- A .NET library and LabVIEW library.
- Supports 32 bit and 64 bit applications.
- Properties and methods for controlling multiple robots, I/O, and tasks from multiple controllers.
- Methods for executing vision and force sensing* commands.
  * Force sensing and Force Sensor are different. Methods and properties for force sensing described in API manual are not available for Force Sensor. To use commands for Force Sensor, use Xqt method to execute SPEL function. API does not support Epson RC+ option Force Guide.
- Supports parallel execution of asynchronous commands by multi-threading.
- Several Epson RC+ 8.0 windows and dialogs can be used by your .NET application, including:
  - Robot Manager
  - IO monitor
  - Task manager
  - Simulator
  - Controller Tools dialog

During development, Epson RC+ 8.0 can be run along with Visual Basic. In production facilities, Epson RC+ 8.0 can be run invisibly in the background.

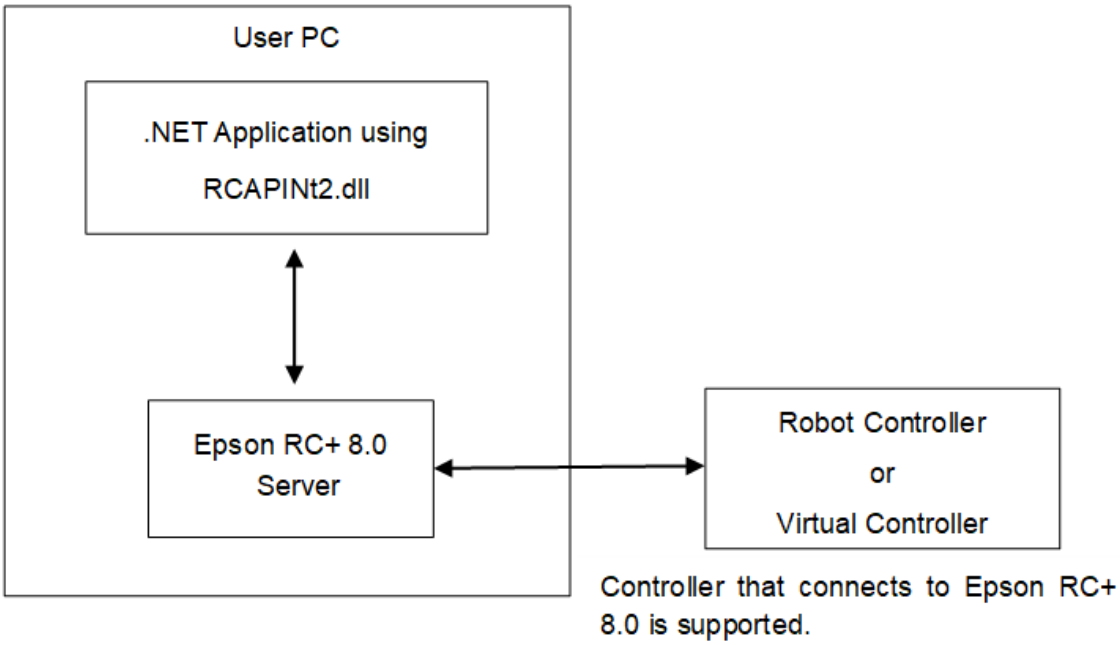
KEY POINTS

RC+ API supports " .NET library" and LabVIEW library. " .NET" 5 is not supported.

RC+ API applications created with Epson RC+ 8.0 or later can only be used with Epson RC+ 8.0.

RC+ applications and projects created with EPSON RC+ 7.0 are also compatible with Epson RC+ 8.0. To use the project created in earlier version, you need to re-build the project on Epson RC+ 8.0.

The figure below shows the basic structure of a system using the RC+ API.



Epson RC+ 8.0 is an out-of-process server for the RCAPINet library.
Each instance of RCAPINet Spel class can start an instance of Epson RC+ 8.0.

3. Installation

Please follow the instructions in this chapter to help ensure proper installation of the RC+ API software.
Before starting, ensure that all Windows applications have been closed.

3.1 Step by step instructions

1. Install either one of the following:
   Visual Studio 2015, 2017, 2019, 2022
   (Includes Enterprise, Professional, Community, and Express Editions)
   LabVIEW 2009 or later
2. Install Epson RC+ 8.0.
3. If you are using LabVIEW, install the LabVIEW VI library.
4. Ensure that the software key has been enabled for RC+ API in the Controllers you will be using.
   Refer to the Epson RC+ 8.0 User's Guide for information on how to enable options in the Controller.

This completes the RC+ API installation.

3.2 What's installed

The directories and files shown in the table below are installed on your PC during installation.

Directories and Files	Description
(Data folder)\API\VS20xx\VB\DEMOS	Visual Basic .NET demonstrations
(Data folder)\API\VS20xx\VCS\DEMOS	Visual C# .NET demonstrations
(Data folder)\API\VS20xx\VC\DEMOS	Visual C++ .NET demonstrations
(Program folder)\API\LabVIEW	LabVIEW VI Library installer
(Data folder)\PROJECTS\API_Demos	Epson RC+ 8.0 projects for demos
(Program folder)\API\Assembly\net462\RCAPINt2.dll	RCAPINet library for .NET Framework (32 bit or 64 bit)
(Program folder)\API\Assembly\net6.0-windows\RCAPINt2.dll	RCAPINet library for .NET6/ .NET8 (32 bit or 64 bit)

4. Getting Started

4.1 Getting Started

This chapter contains information for getting started in the following development environments.

- Visual Basic .NET
- Visual C# .NET
- Visual C++ .NET

Demonstration programs are supplied with the RC+ API. It is recommended that you go through the demonstrations to get more familiar with the product.
See below for details about the location of the directory.
What's installed

LabVIEW users should see below.
Getting started

When you build the demonstration program on Visual C++ 2017 for the first time, the program build may fail. See below when the program build fails.
Getting started using Visual C++

To use .NET application, set .NET Framework version to v4.6.2 and greater or .NET6/ .NET8.

The following explanation is based on operation with Visual Studio 2022.
This manual describes with Visual Studio 2022 but the procedures are the same with any version of them.

4.2 Getting started using Visual Basic

To use RCAPINet in a Visual Basic .NET project, declare a Spel Class instance, as shown in the example below. g_spel can now be used in your project.

1. In Visual Studio .NET, select File | Project.
2. Select the following type for the Visual Basic project.
   - .NET Framework : "Windows Forms App (.NET Framework)"
   - .NET6/ .NET8 : "Windows Forms App"
3. Select [Project]-[Add Reference].
4. In the [Browse] tab, select the "RCAPINt2.dll" file under the "\API" directory.
   - .NET Framework : \Assembly\ net462
   - .NET6/ .NET8 : \Assembly\net6.0-windows
5. From the Project menu, create a new module and add the following code.

```
Module Module1
  Public WithEvents g_spel As RCAPINet.Spel
  Public Sub InitApp()
    g_spel = New RCAPINet.Spel
    With g_spel
      .Initialize
      .Project = "c:\EpsonRC80\projects\API_Demos\Demo1 \demo1.sprj"
    End With
  End Sub

  Public Sub EventReceived( _
          ByVal sender As Object, _
          ByVal e As RCAPINet.SpelEventArgs) _
          Handles g_spel.EventReceived
```

```
    MsgBox("received event " & e.Event)
  End Sub
End Module
```

KEY POINTS

When your application exits, you need to execute Dispose for each Spel class instance. This can be done in your main form's FormClosed event. If Dispose is not executed, the application will not shutdown properly.

g_spel.Dispose()

4.3 Getting started using Visual C#

1. Select menu-[File]-[New]-[Project] in Visual Studio .NET.
2. Select the following type for the Visual C# project.

- .NET Framework : "Windows Forms App (.NET Framework)"
- .NET6/ .NET8 : "Windows Forms App"

3. Select [Project]-[Add Reference].
4. In the [Browse] tab, select the "RCAPINt2.dll" file under the "\API" directory.

- .NET Framework : \Assembly\ net462
- .NET6/ .NET8 : \Assembly\net6.0-windows

5. In the Form1 class, declare a Spel class variable as shown below.

```
private RCAPINet.Spel m_spel;
```

6. In the Form_Load event, add initialization code, as shown below.

```
private void Form1_Load(object sender, EventArgs e)
{
   m_spel = new RCAPINet.Spel();
   m_spel.Initialize();

   m_spel.Project =
"c:\\EpsonRC80\\projects\\API_Demos\\Demo1\\demo1.sprj";

m_spel.EventReceived += new
   RCAPINet.Spel.EventReceivedEventHandler(m_spel_EventReceived);
```

7. Add the event handler, as shown below.

```
public void m_spel_EventReceived(object sender,
   RCAPINet.SpelEventArgs e)
{
}
```

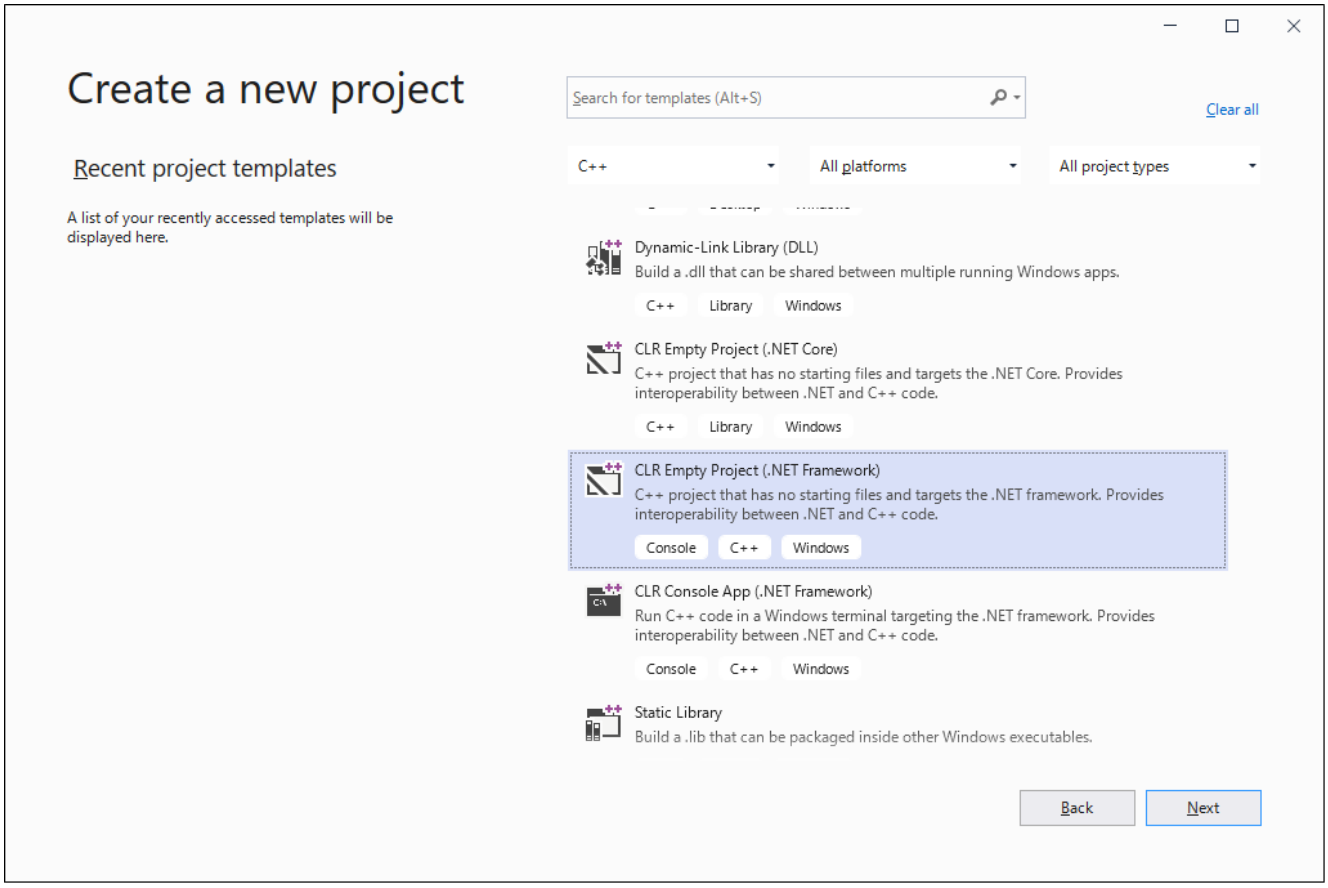
KEY POINTS

When your application exits, you need to execute Dispose for each Spel class instance. This can be done in your main form's FormClosed event. If Dispose is not executed, the application will not shutdown properly.

```
m_spel.Dispose();
```

4.4 Getting started using Visual C++

1. In Visual Studio .NET, select [Create a new project].
2. Select [Visual C++]-[CLR]-[CLR Empty Project (.NET Framework)].

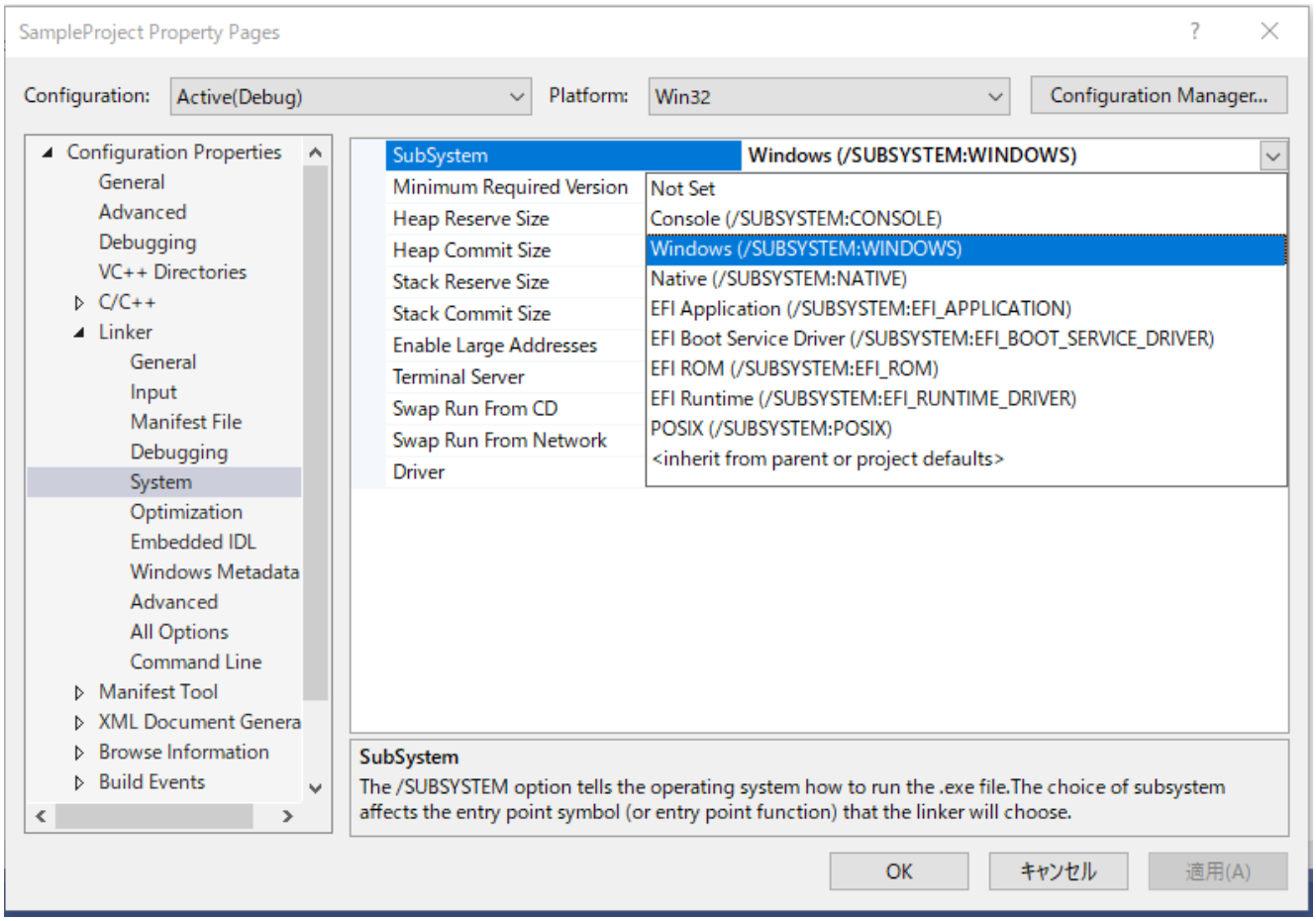


3. Select [Project]-[Add Reference].
4. In the [Browse] tab, select the "RCAPINt2.dll" file under the "\API" directory.
   - .NET Framework : \Assembly\ net462
   - .NET6/ .NET8 : \Assembly\net6.0-windows
5. Select the menu-[Project]-[Add New Item]-[UI]-[Windows Form].
6. Open the cpp file (ex: Form1.cpp) of the added form and add the following source code.

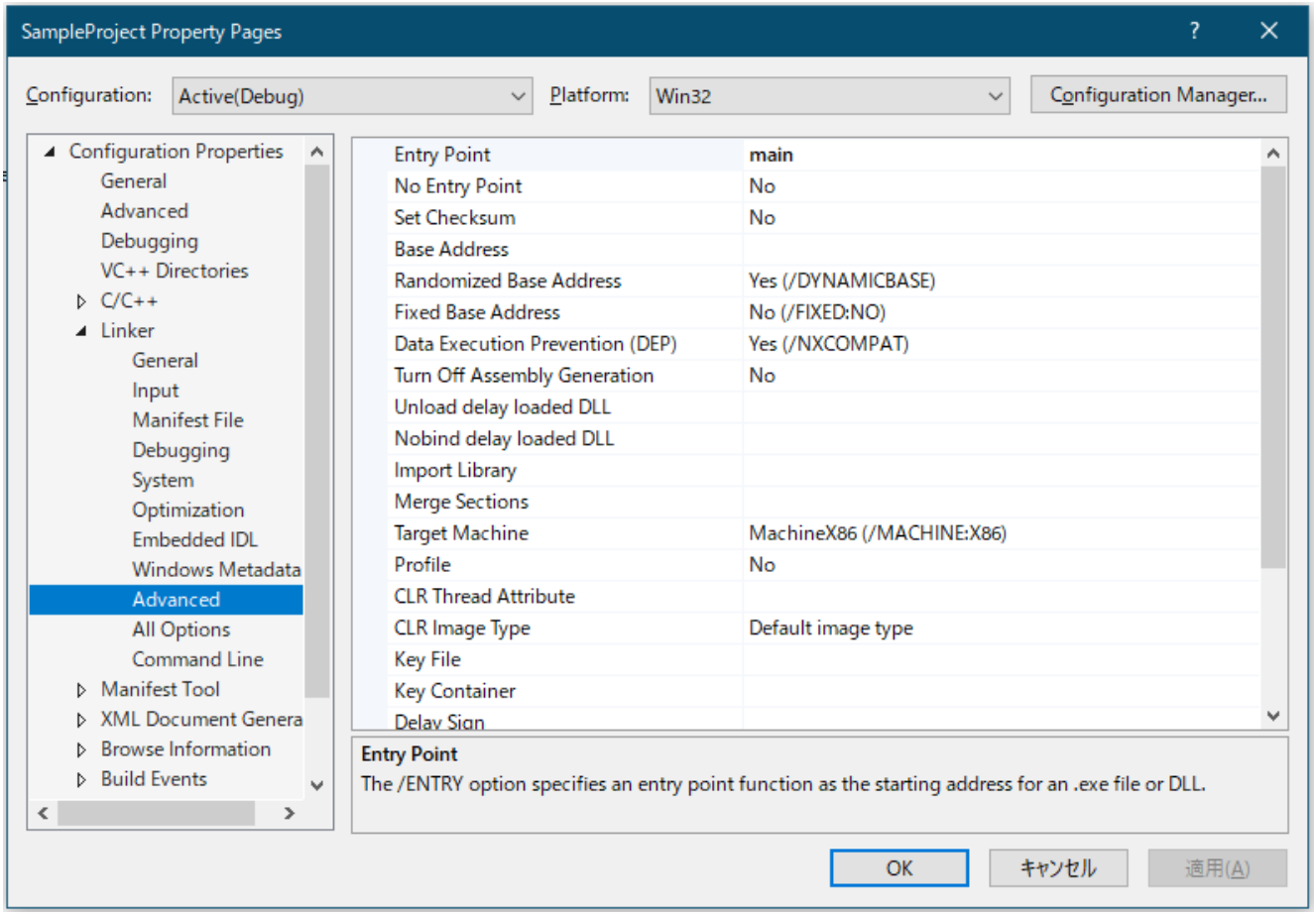
```
#include "Form1.h"

 using namespace SampleProject; // Name of the created project
 void main() {
    Application  EnableVisualStyles();
    Application  SetCompatibleTextRenderingDefault(false);

    Form1 frm; // Name of added form
    Application Run(% frm);
 }
```

7. Select the menu-[Project]-[Project Properties].

8. Select [Configuration Properties]-[Linker]-[System] on Property Pages and select “Windows (/SUBSYSTEM:WINDOWS)” on SubSystem.



9. Select [Configuration Properties]-[Linker]-[Advanced] on Property Pages and enter the function name that is added in the step 6 in “Entry Point”. In here, “main” is entered.



10. Click the [OK] button.

KEY POINTS

After configuring the setting, build the solution once and make sure that no error is occurred. Then, we recommend that you close the solution and reopen it.

11. In the Form1 class, declare a Spel variable as shown below.

```
private:RCAPINet::Spel^ m_spel;
```

12. In the Form_Load event, add initialization code, as shown below.

```
private:System::Void Form1_Load(
   System::Object^ sender, System::EventArgs^ e)
{
   m_spel = gcnew RCAPINet::Spel();
   m_spel->Initialize();
   m_spel->Project =
      "c:\\EpsonRC80\\projects\\ API_Demos\\Demo1\\demo1.sprj";
   m_spel->EventReceived += gcnew
      RCAPINet::Spel::EventReceivedEventHandler(
      this, &Form1::m_spel_EventReceived);
}
```

13. Add the event handler, as shown below.

```
private:System::Void m_spel_EventReceived(
   System::Object^ sender, RCAPINet::SpelEventArgs^ e)
{
   MessageBox::Show(e->Message);
}
```

KEY POINTS

When your application exits, you need to delete each Spel class instance if it was allocated on the heap (using gcnew). This can be done in your main form's FormClosed event. If the Spel class instances are not deleted, then the application will not shutdown properly.

delete m_spel;

5. Environments

5.1 Development Environment

5.1.1 Development Startup

Typically, you would perform these steps to start development:

1. Declare a Spel class variable in a module in your .NET project.
2. Start Epson RC+.
3. Open the desired Epson RC+ project or create a new Epson RC+ project.
4. Build the Epson RC+ project.
5. Add initialization code for the SPEL class instance.
6. Run and debug the .NET project.

5.1.2 Spel Class Instance Initialization

After a new instance of the Spel class has been created, it needs to be initialized. When initialization occurs, the underlying Epson RC+ 8.0 modules are loaded and initialized. Initialization is implicit with the first method call or property access. You can initialize the class by calling the Initialize method.

```
  m_spel.Initialize()
```

5.1.3 Spel Class Instance Termination

When your application exits, you need to execute Dispose for each Spel class instance. This can be done in your main form's FormClosed event. If Dispose is not executed, the application will not shutdown properly.
For Visual Basic and Visual C#, use the Dispose method:

```
m_spel.Dispose()
```

For Visual C++, if your Spel class instance was created on the heap (with gcnew), then use delete:

```
delete m_spel;
```

5.1.4 Development Cycle

Follow these basic steps to edit and run your .NET code:

1. Stop the .NET project.
2. Edit the .NET project
3. Open Epson RC+ 8.0.
4. Make changes in the Epson RC+ 8.0 project.
5. Build the Epson RC+ 8.0 project.
6. Close Epson RC+ 8.0.
7. Switch to Visual Studio.
8. Run the .NET project.

5.2 In Production Facilities

5.2.1 Opening Epson RC+ 8.0 at Runtime

Decide if you want to allow the Epson RC+ 8.0 environment to be opened from your application. This is especially useful for debugging. Set the OperationMode property to Program to put Epson RC+ 8.0 in Program Mode and open the Epson RC+ 8.0 GUI.

5.2.2 Using Epson RC+ 8.0 Dialogs and Windows

At runtime, you can open and hide certain Epson RC+ 8.0 windows from your .NET application. You can also run certain Epson RC+ 8.0 dialogs. See below for more details.
EpsonRC+8.0 Windows and Dialogs

5.2.3 Installation on Target System

You should make an installation program for your .NET project by using a Visual Studio setup project. Then follow these steps to setup a target system for your .NET application:

1. Install Epson RC+ 8.0.
2. Install your Epson RC+ 8.0 project.
3. Install your .NET application.

6. Executing Methods, Programs, Tasks

6.1 Executing Methods

There are several methods in the Spel class. For descriptions of available methods, see below.

Spel Class Methods

When you execute a method, the associated internal functions are called in the Epson RC+ server process, which in turn communicates with the Controller to execute the associated function. There are two types of methods: immediate and asynchronous. For immediate methods, the internal function is executed in the Controller and the reply is returned immediately. Immediate commands include all I/O commands. For asynchronous methods, the associated function is started in the Controller, and then the Spel class instance waits for an event from the Epson RC+ server process indicating that the function has completed. Asynchronous methods include all robot motion commands. While waiting for command completion, the Spel class instance dispatches Windows events, so that the user GUI is still responsive. For example, when the Go method is called, the robot is moving to a point, and the user may want to stop it by clicking a button. You can disable Windows event dispatching during asynchronous methods by setting DisableMsgDispatch to True. You can also wait for asynchronous methods to finish in your program by setting AsyncMode to True.

6.1.1 Using Multiple Threads

You can execute Spel methods in multiple threads in your application. The sections below describe the various scenarios.

One Spel class instance used in multiple threads

You can execute methods with the same Spel class instance in multiple threads, but only one asynchronous command at a time. If you attempt to execute an asynchronous command in one thread while another asynchronous command is already executing in another thread, you will get a “command in cycle” error. You can execute an immediate command in one thread while executing an asynchronous command in another thread.

Separate Spel class instance used in each thread

For each Controller connection, you can have one or more Spel class instances. The first instance for each Controller initializes an Epson RC+ 8.0 server process and connects to the specified Controller. To use one or more additional instances in other threads to communicate with the same Controller, you must specify the ServerInstance property to be the same value. You call Initialize for the first instance before using additional Spel class instances.

VB Example:

```
' Initialize Spel class instance for thread 1
m_spel_1 = New Spel
m_spel_1.ServerInstance = 1
m_spel_1.Initialize()
m_spel_1.Project = "c:\EpsonRC80\Projects\MyProject\MyProject.sprj"
m_spel_1.Connect(1)

' Initialize Spel class instance for thread 2
' This instance uses the same controller as m_spel_1
m_spel_2 = New Spel
m_spel_2.ServerInstance = 1

Thread 1
' Uses instance m_spel_1 for motion
m_spel_1.Robot = 1
Do
m_spel_1.Go(1)
m_spel_1.Go(2)
Loop Until m_stop
Thread 2
' Uses instance m_spel_2 for I/O
Do
m_spel_2.On(1)
m_spel_2.Delay(500)
m_spel_2.Off(1)
```

```
m_spel_2.Delay(500)
Loop Until m_stop
```

C# Example:

```
// Initialize Spel class instance for thread 1
RCAPINet.Spel m_spel_1 = new RCAPINet.Spel();
m_spel_1.ServerInstance = 1;
m_spel_1.Initialize();
m_spel_1.Project = @"c:\EpsonRC80\Projects\MyProject\MyProject.sprj";
m_spel_1.Connect(1);

// Initialize Spel class instance for thread 2
// This instance uses the same controller as m_spel_1
RCAPINet.Spel m_spel_2 = new RCAPINet.Spel();
m_spel_2.ServerInstance = 1;

Thread 1
// Uses instance m_spel_1 for motion
m_spel_1.Robot = 1;
do{
m_spel_1.Go(1);
m_spel_1.Go(2);
}while(\!m_stop);
Thread 2
// Uses instance m_spel_2 for I/O
do{
m_spel_2.On(1);
m_spel_2.Delay(500);
m_spel_2.Off(1);
m_spel_2.Delay(500);
}while(\!m_stop);
```

Using API threads in the Controller

By default, only one API thread is supported in the Controller. In this case, asynchronous methods are executed one at a time in the Controller, even when controlling multiple robots. For most applications that use one robot, or execute robot motion using SPEL+ tasks, this is sufficient, but you can configure the system to use up to 10 API tasks in the Controller to allow parallel processing for your .NET threads, such as when you are controlling more than one robot from the same Controller.

There are two basic steps required to use more than one API task in the Controller.

1. In the Epson RC+ GUI, connect to the Controller, then open [Setup]-[System Configuration]-[Controller]-[Preferences]. Set “Reserved tasks for API” to the desired number of API tasks. Note that the more tasks you reserve for the API, the fewer tasks will be available for your SPEL+ programs. For example, if you reserve 5 API tasks, then there will be 27 tasks (32 – 5) available for SPEL+.
2. In your application, set the CommandTask property to specify which API task you want to execute methods on.

In the simple example below, there is one thread for each robot in the same Controller. The robot motion commands will execute in parallel, since a different CommandTask is used in each thread, and ServerInstance is set to 1 for both Spel instances.

VB Example:

```
' Initialize Spel class instance for thread 1
m_spel_1 = New Spel
m_spel_1.ServerInstance = 1
m_spel_1.CommandTask = 1
m_spel_1.Initialize()
m_spel_1.Project = "c:\EpsonRC80\Projects\MyProject\MyProject.sprj"
m_spel_1.Connect(1)

' Initialize Spel class instance for thread 2
```

```
' This instance uses the same controller as m_spel_1
' And uses the second CommandTask in the controller.
m_spel_2 = New Spel
m_spel_2.ServerInstance = 1
m_spel_2.CommandTask = 2

Thread 1
' Uses instance m_spel_1 for Robot 1 motion
m_spel_1.Robot = 1
Do
m_spel_1.Go(1)
m_spel_1.Go(2)
Loop Until m_stop
Thread 2
' Uses instance m_spel_2 for Robot 2 motion
m_spel_2.Robot = 2
Do
m_spel_2.Go(1)
m_spel_2.Go(2)
Loop Until m_stop
```

C# Example:

```
// Initialize Spel class instance for thread 1
RCAPINet.Spel m_spel_1 = new RCAPINet.Spel();
m_spel_1.ServerInstance = 1;
m_spel_1.CommandTask = 1;
m_spel_1.Initialize();
m_spel_1.Project = @"c:\EpsonRC80\Projects\MyProject\MyProject.sprj";
m_spel_1.Connect(1);

// Initialize Spel class instance for thread 2
// This instance uses the same controller as m_spel_1
// And uses the second CommandTask in the controller.
RCAPINet.Spel m_spel_2 = new RCAPINet.Spel();
m_spel_2.ServerInstance = 1;
m_spel_2.CommandTask = 2;

Thread 1
// Uses instance m_spel_1 for Robot 1 motion
m_spel_1.Robot = 1;
do{
m_spel_1.Go(1);
m_spel_1.Go(2);
}while(\!m_stop);

Thread 2
// Uses instance m_spel_2 for Robot 2 motion
m_spel_2.Robot = 2;
do{
m_spel_2.Go(1);
m_spel_2.Go(2);
}while(\!m_stop);
```

6.2 Executing SPEL+ Programs

A SPEL+ program contains one or more functions, and the program is run by starting the main function of the program. You can run any of the 64 built-in main functions in the current Controller project by using the Start method of the Spel class. The main function(s) that you start must be defined in your SPEL+ code. When you start a main function, all global variables and module variables are cleared to default values.
The table below shows the program numbers and their corresponding function names in the SPEL+ project.

Program Number	SPEL+ Function Name
0	main
1	main1
2	main2
3	main3
...	...
63	main63

Here are examples that start function “main”:
VB Example:

```
Sub btnStart_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnStart.Click

    m_spel.Start(0) ' Starts function main
    btnStart.Enabled = False
    btnStop.Enabled = True
End Sub
```

C# Example:

```
void btnStart_Click(object sender, EventArgs e)
{
    m_spel.Start(0); //Starts function main
    btnStart.Enabled = false;
    btnStop.Enabled  = true;
}
```

6.3 Executing SPEL+ Tasks

You can execute functions in your SPEL+ program as a normal task by using the Xqt method. When you execute a task, global variables are not cleared to default values, as they are when you use the Start method.
To suspend and resume a task, use the Halt and Resume methods.
To quit a task, use the Quit method.
You can also start Controller background tasks using the StartBGTask method.

6.4 Aborting All Tasks

If you are running tasks and want to abort all tasks at once, you can use the Stop method of the Spel class. The Stop method has an optional parameter that allows you to additionally stop all background tasks.

VB Example:

```
Sub btnStop_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnStop.Click
    m_spel.Stop()
    btnStop.Enabled = False
```

```
    btnStart.Enabled = True
End Sub
```

C# Example:

```
void btnStop_Click(object sender, EventArgs e)
{
    m_spel.Stop();
    btnStop.Enabled = false;
    btnStart.Enabled = true;
}
```

7. Events

7.1 Overview

The Spel Class supports two types of events: system events and user events. System events are notifications of system status. User defined events are sent from any SPEL+ task to the .NET application.

7.2 System Events

There are several system events that are sent to the .NET application. Each system event indicates a change in status.
See below for details about the system events.

Spel Class Events

Use the Spel class EnableEvents method to control which system events are sent.

7.3 User Events from SPEL+

You can cause events to occur in your .NET application from your SPEL+ programs. For example, you can inform the .NET application about a continuous cycle loop. This is a better method to use than polling for variable values in the Controller from .NET. To fire an event to .NET from SPEL+, use the SPELCom_Event command in a SPEL+ program statement.
For example:

```
SPELCom_Event 1000, cycNum, lot$, cycTime
```

The SPELCom_Event command is similar to a Print command. You can specify one or more pieces of data to be sent to the .NET application. For details about “SPELCom_Event”, see below.

SPELCom_Event

Before you can receive events, you must declare your Spel class variable using the WithEvents clause.

```
Public WithEvents m_spel As RCAPINet.Spel
```

Catch the event in the EventReceived routine for the Spel class instance. To edit this routine, in the module where the Spel class is declared select “m_spel” from the class name list and EventReceived from the procedure list.

Here is an example of code in the EventReceived routine that updates some labels when an event occurs.

VB Example:

```
Sub m_spel_EventReceived (ByVal sender As Object, _
ByVal e As RCAPINet.SpelEventArgs) _
Handles m_spel.EventReceived
    Dim tokens() As String
    Select Case e.Event
        Case 2000
           tokens = e.Message.Split(New [Char]() {" "c}, _
System.StringSplitOptions.RemoveEmptyEntries)
           lblCycCount.Text = tokens(0)
           lblLotNumber.Text = tokens(1)
           lblCycTime.Text = tokens(2)
    End Select
End Sub
```

C# Example:

```
void m_spel_EventReceived(object sender, SpelEventArgs e)
{
string[] tokens = new string[3];
switch(e.Event)
```

```
   {
   case 2000:
            tokens = e.Message.Split(' ');
            lblCycCount.Text = tokens(0);
            lblLotNumber.Text = tokens(1);
            lblCycTime.Text = tokens(2);
            break;
   default:
            break;
    }
}
```

8. Error Handling

8.1 Errors for Spel methods

When you execute a Spel class method, an exception is thrown if there are any errors.
When an error occurs, the Spel class instance throws it to the calling routine. You should use error handlers in your application to catch this error. In some cases, you will only want to display an error message.

VB Example:

```
Sub btnStart_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnStart.Click

  Try
    m_spel.Start(0)
  Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
  End Try
End Sub
```

You can examine the error number associated with the exception by using the ErrorNumber property of SpelException.

```
  Try
    m_spel.Start(0)
  Catch ex As RCAPINet.SpelException
    MsgBox("SPEL Error: " + ex.ErrorNumber.ToString())
  End Try
```

C# Example:

```
  void btnStart_Click(object sender, EventArgs e)
{
  try{
    m_spel.Start(0);
  }
  catch(SpelException ex){
    MessageBox.Show(ex.Message);
  }
}
```

You can examine the error number associated with the exception by using the ErrorNumber property of SpelException.

```
try {
  m_spel.Start(0);
}
catch(SpelException ex) {
  MessageBox.Show(("SPEL Error: " + ex.ErrorNumber.ToString());
}
```

9. Handling Pause and Continue

9.1 Pause state

When a pause occurs, the Controller and SPEL+ tasks are in the pause state.
The Controller is in the pause state after one of the following occurs while tasks are running:
The Spel class Pause method was executed
A SPEL+ task executed Pause.
The safeguard was opened.

9.2 Catching the Pause event

The Spel class will signal your .NET application that a pause has occurred.
You can catch the Pause event in the EventReceived event for the Spel class.

VB Example:

```
Sub m_spel_EventReceived (ByVal sender As Object, ByVal e As RCAPINet.
SpelEventArgs) Handles m_spel.EventReceived
    Select Case e.Event
        Case RCAPINet.SpelEvents.Pause
            btnPause.Enabled = False
            btnContinue.Enabled = True
    End Select
End Sub
```

C# Example:

```
void m_spel_EventReceived(object sender, SpelEventArgs e)
{
switch(e.Event)
       {
        case SpelEvents.Pause:
            btnPause.Enabled = false;
            btnContinue.Enabled = true;
            break;
        default:
            break;
        }
}
```

9.3 Executing Pause

The following routine shows how to issue a PAUSE from Visual Basic using the Pause method.

VB Example:

```
Sub btnPause_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnPause.Click

    m_spel.Pause()
    btnPause.Enabled = False
    btnContinue.Enabled = True
End Sub
```

C# Example:

```
void btnPause_Click(object sender, EventArgs e)
{
 m_spel.Pause();
 btnPause.Enabled = false;
 btnContinue.Enabled = true;
}
```

9.4 Continue after pause

To continue after a pause has occurred, use the Continue method.

VB Example:

```
Sub btnContinue_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnContinue.Click

    m_spel.Continue()
    btnContinue.Enabled = False
    btnPause.Enabled = True
End Sub
```

C# Example:

```
void btnContinue_Click(object sender, EventArgs e)
{
 m_spel.Continue();
 btnContinue.Enabled = false;
 btnPause.Enabled = true;
}
```

9.5 Abort after pause

You can also execute the Stop method if you don't want to continue after a pause.

VB Example:

```
Sub btnStop_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnStop.Click

    m_spel.Stop()
    btnContinue.Enabled = False
    btnPause.Enabled = False
End Sub
```

C# Example:

```
void btnStop_Click(object sender, EventArgs e)
{
 m_spel.Stop();
 btnContinue.Enabled = true;
 btnPause.Enabled = false;
}
```

10. Handling Emergency Stop

When an Emergency stop occurs, you may want to perform some specific action in your program, such as displaying a dialog, or a message box.
The Spel class issues two standard events for emergency stop status: EStopOn and EStopOff.

10.1 Using system EStop events

You can catch the system EStop events in the EventReceived handler in your Visual Basic application.

```
Imports RCAPINet.Spel

Private Sub m_spel_EventReceived(ByVal sender As Object, ByVal e As SpelEventArgs)
Handles m_spel.EventReceived
    Select Case e.Event
        Case RCAPINet.SpelEvens.EstopOn
            MsgBox "E-Stop detected"
            gEStop = True
            lblEStop.BackColor = Color.Red
            lblEStop.Text = "EStop ON"
        Case RCAPINet.SpelEvents.EstopOff
            gEStop = False
            lblEStop.BackColor = Color.Green
            lblEStop.Text = "EStop OFF"
    End Select
End Sub
```

You can catch the system EStop events in the EventReceived handler in your C# application.

```
private void m_spel_EventReceived(object sender, SpelEventArgs e)
{
switch(e.Event)
       {
        case SpelEvents.EstopOn:
            MessageBox.Show("E-Stop detected");
            gEStop = true;
            lblEStop.BackColor = Color.Red;
            lblEStop.Text = "EStop ON";
        case SpelEvents.EstopOff:
            gEStop = false;
            lblEStop.BackColor = Color.Green;
            lblEStop.Text = "EStop OFF";
        }
}
```

11. EpsonRC+8.0 Windows and Dialogs

You can open certain Epson RC+ 8.0 windows and dialogs from your .NET application using the ShowWindow and RunDialog methods of the Spel class.

11.1 Windows

Windows are non-modal, meaning that they can remain open while other elements of your Visual Basic GUI can be used. You can show and hide Epson RC+ windows from your Visual Basic program.

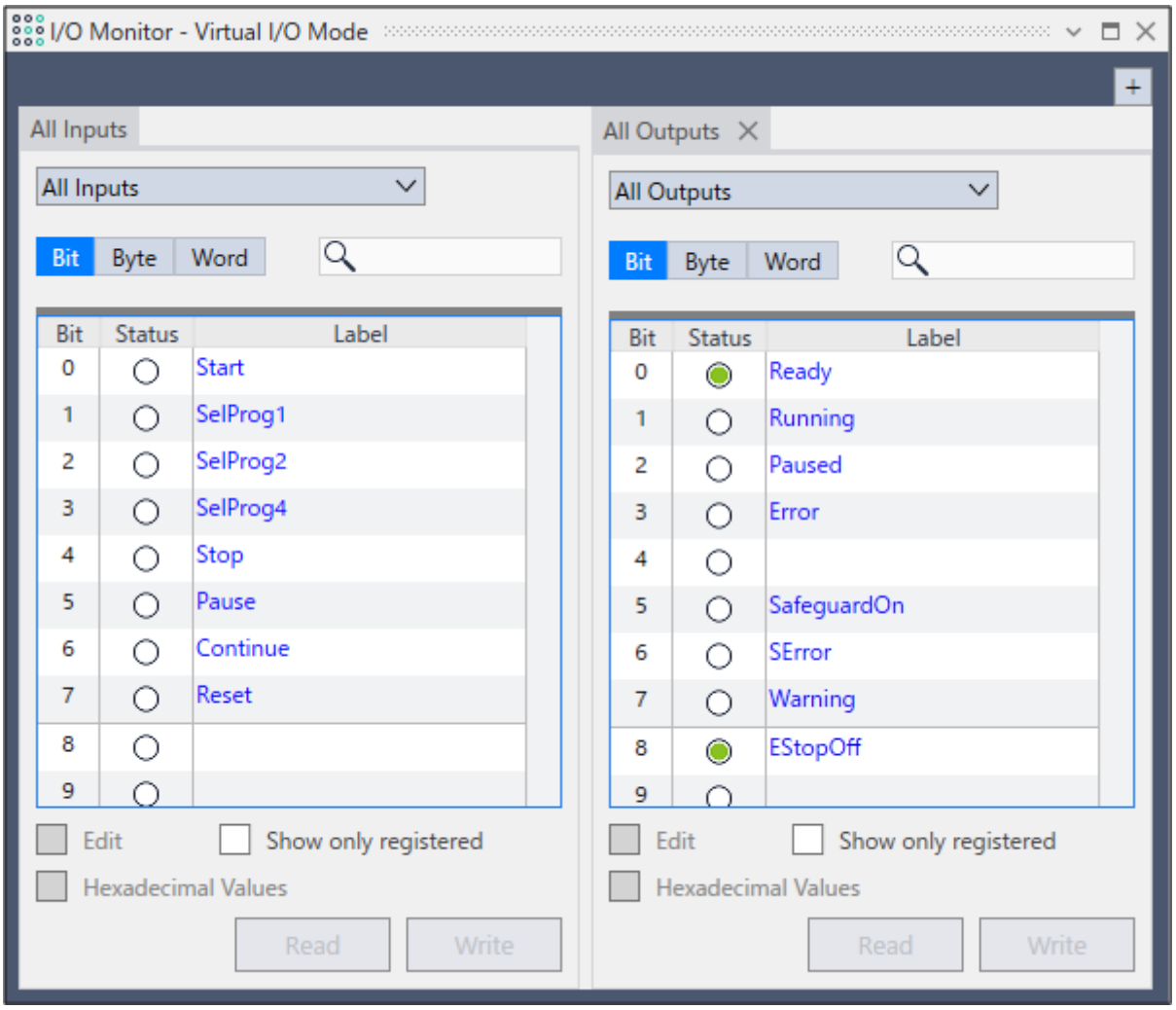
For example, to open and close the I/O Monitor window:

```
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, Me)
m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor)
```

You can show and hide Epson RC+ windows from your C# program.

```
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, this);
m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor);
```

WindowID	Window
RCAPINet.SpelWindows.IOMonitor	IO monitor
RCAPINet.SpelWindows.TaskManager	Task Manager
RCAPINet.SpelWindows.ForceMonitor	Force Monitor
RCAPINet.SpelWindows.Simulator	Simulator



I/O Monitor Window

11.2 Dialogs

If the [Robot Manager] dialog is displayed, other elements of the .NET GUI are also available.

For example, to open the Robot Manager dialog:

```
m_spel.RunDialog(RCAPINet.SpelDialogs.RobotManager)
```

If a dialog other than the [Robot Manager] is displayed, other elements of the .NET GUI are unavailable until you close the dialog. (modal dialog)

If this dialog is displayed, the operator should close it. For safety reasons, the dialog cannot be closed from the program being used.
The following table shows the dialogs that can be opened.

DialogID	Dialog
RCAPINet.SpelDialogs.RobotManager	Robot Manager
RCAPINet.SpelDialogs.ControllerTools	Controller Tools
RCAPINet.SpelDialogs.VisionGuide	Vision Guide
RCAPINet.SpelDialogs.ForceGuide	Force Guide
RCAPINet.SpelDialogs.PartFeeding	Part Feeding

12. Displaying Video

You can display video on a form in your application by using the SPELVideo control. When you run a vision sequence, the graphics can also be displayed on the window.

12.1 Displaying Video

Perform the following steps to create a video display:

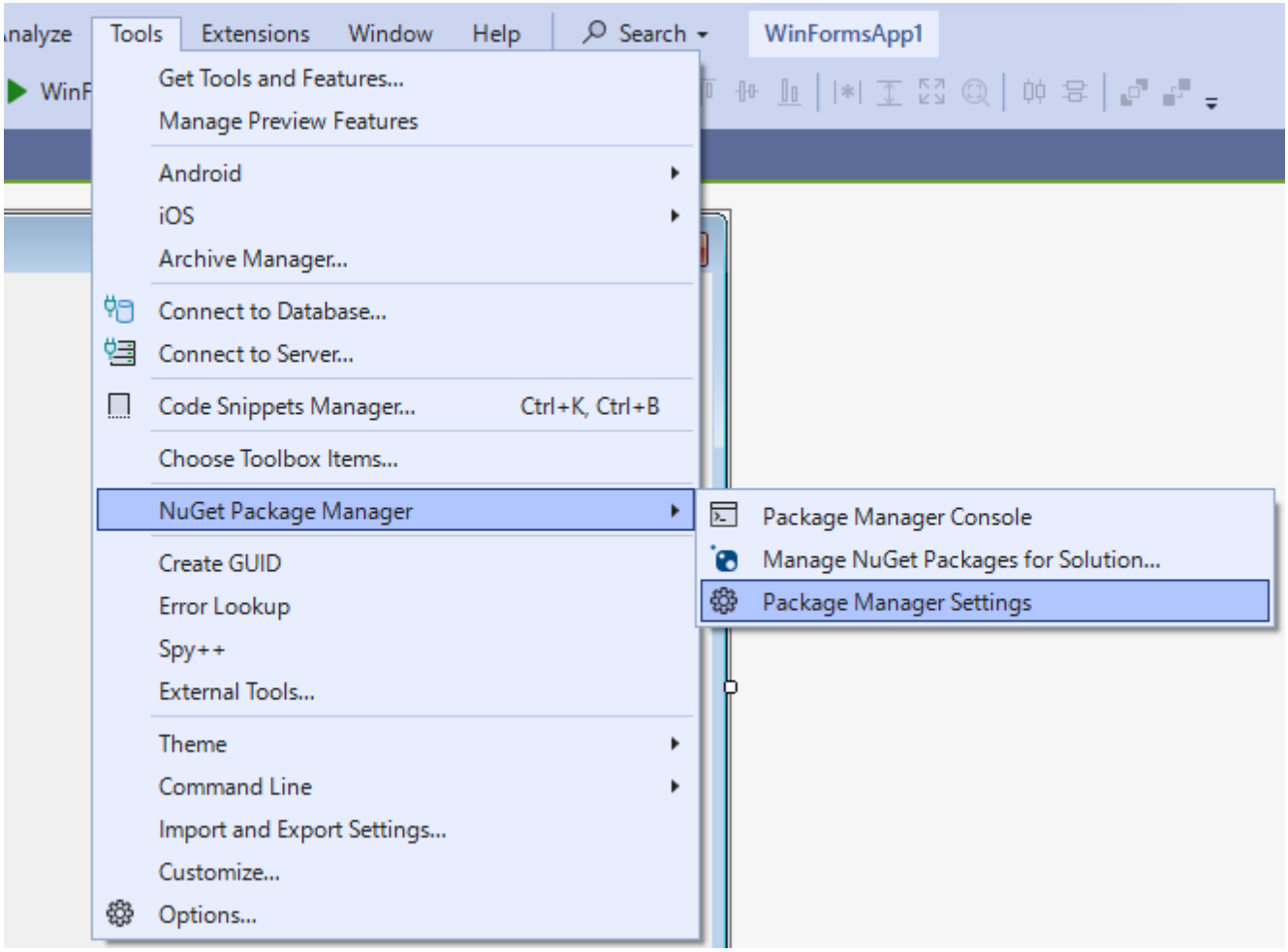
1. Add the SpelVideo control to the project.
   Adding the SPELVideo control differs depending on the .NET Framework, .NET6 and .NET8.

   .NET Framework

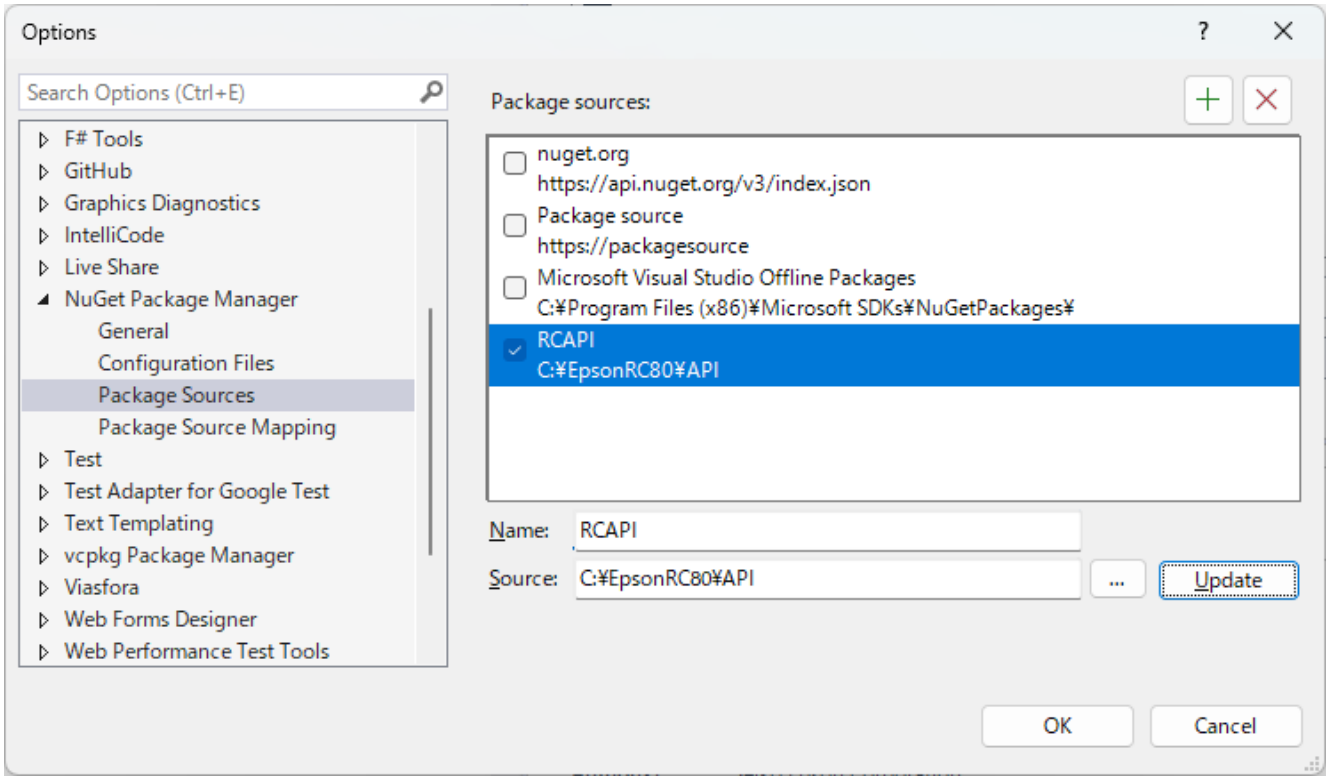
   To add the control to your Visual Studio .NET toolbox, right click on the toolbox and select Choose Items. Select the Browse tab and browse to the \EpsonRC80\Exe directory and select the RCAPINt2.dll file. The SPELVideo control icon will be added to the toolbox.

   .NET6/ .NET8

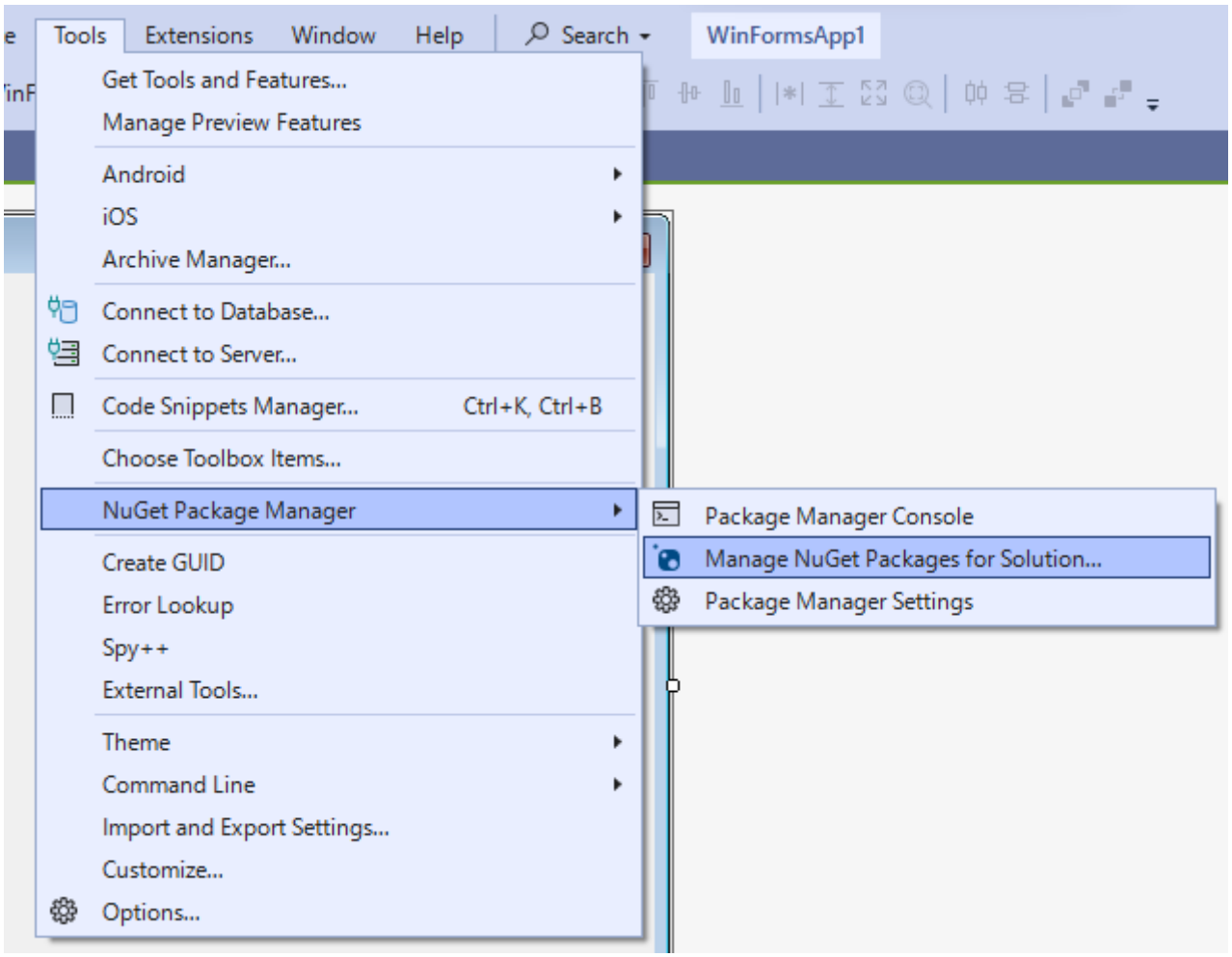
   (1) Add the SpelVideo to the project. Open [Tools]-[NuGet Package Manager]-[Package Manager Settings].



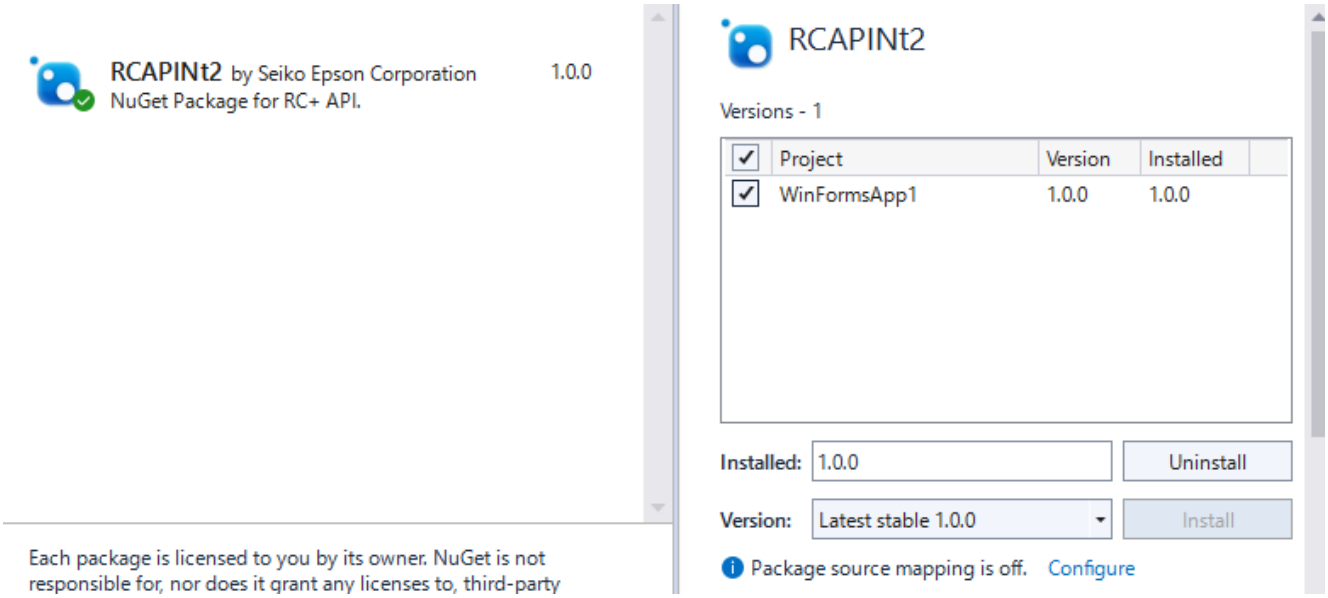
   (2) Select the package source from the tree. Click [+] on the top right corner and create a package source.
   [Name] of the text box: Optional, [Source]: Select the path that has the Epson RC+ installed (If it was installed without specifying a path, select C:\EpsonRC80\API) and click [OK].



(3) Open [Tools]-[NuGet Package Manager]-[Manage NuGet Packages for Solution].



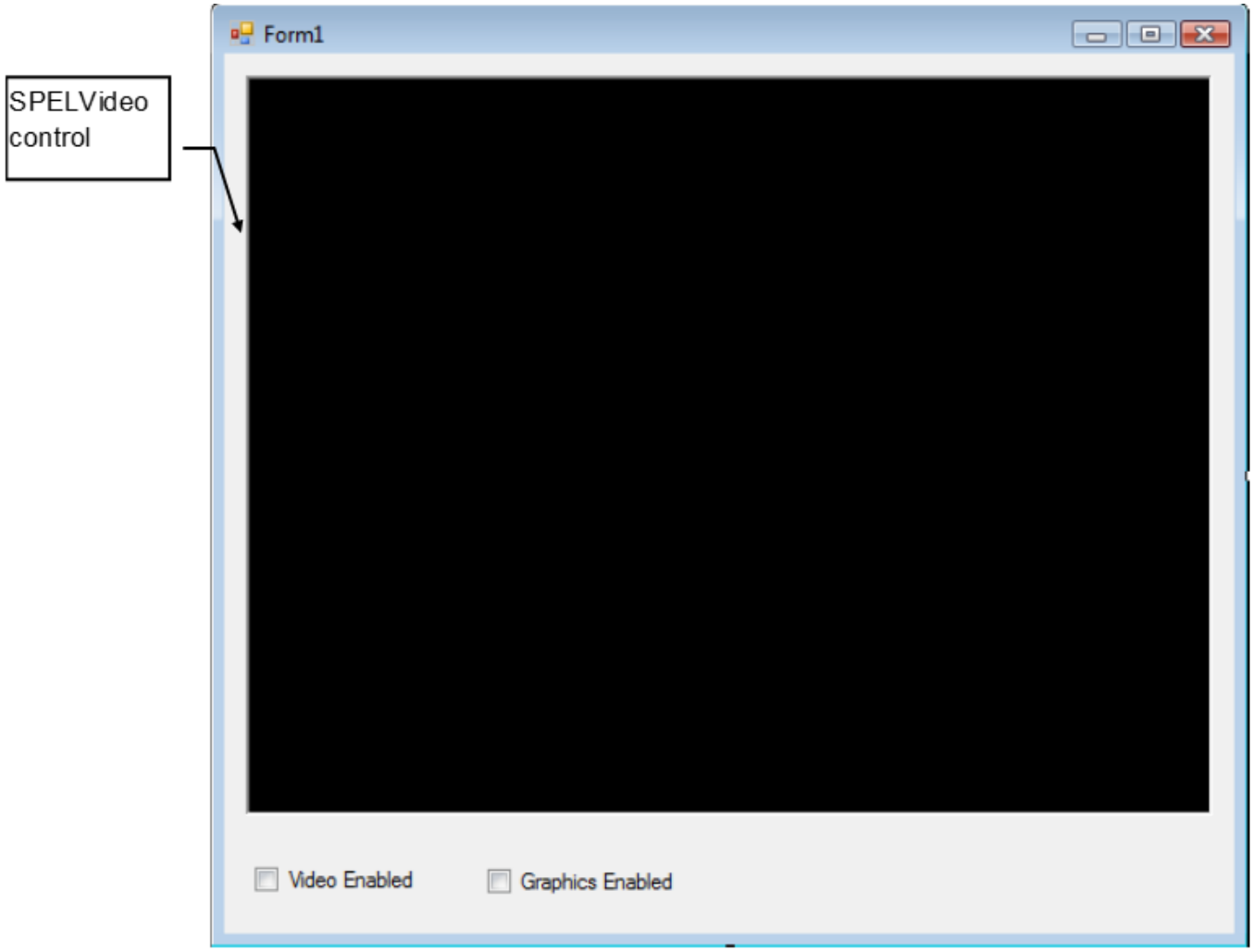
(4) Click [Browse], search [RCAPINt2] and select it. Select the project you want to install and click [Install]. When installation is complete, the SPELVideo control will be added to the tool box.



2. Place a SPELVideo control on the form you want the video to be displayed. The control size can be changed up to the full size.

3. Set the VideoEnabled property to True.

4. Set the GraphicsEnabled property to True if you want to display vision graphics. You must also attach the SPELVideo control to a Spel class instance using the Spel class SpelVideoControl property.



SPELVideo control placed on a form

When the GraphicsEnabled property is True and the control is attached to a Spel class instance, then vision graphics will be displayed whenever the VRun method is executed on the Controller connected to the Spel class instance.

Here is an example showing how to enable video and graphics on a Visual Basic form where a Spel class instance is used and a SPELVideo control have been placed:

```
Private Sub Form_Load(sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    m_spel = New Spel
    m_spel.Initialize()
    m_spel.Project = "c:\EpsonRC80\projects\test\test.sprj"
    SpelVideo1.VideoEnabled = True
    SpelVideo1.GraphicsEnabled = True
    m_spel.SpelVideoControl = SPELVideo1
End Sub
```

Here is an example showing how to enable video and graphics on a C# form where a Spel class instance is used and a SPELVideo control have been placed:

```
private void Form_Load(object sender, EventArgs e)
{
        RCAPINet.Spel m_spel = new RCAPINet.Spel();
        m_spel.Initialize();
        m_spel.Project = @"c:\EpsonRC80\projects\test\test.sprj";
        SpelVideo1.VideoEnabled = True;
        SpelVideo1.GraphicsEnabled = True;
```

```
        m_spel.SpelVideoControl = SPELVideo1;
}
```

12.2 Using multiple video displays

Starting with Epson RC+ 8.0, you can use multiple video displays in your application. For each display, you can select which camera video to display.
To use multiple displays, you must set the SpelVideoControl property for each display.
The example below shows initialization that includes two video displays.

VB Example:

```
Private Sub Form_Load(sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    m_spel = New Spel
    m_spel.Initialize()
    m_spel.Project = "c:\EpsonRC80\projects\test\test.sprj"
    SpelVideo1.VideoEnabled = True
    SpelVideo1.GraphicsEnabled = True
    SpelVideo1.Camera = 1
    SpelVideo2.VideoEnabled = True
    SpelVideo2.GraphicsEnabled = True
    SpelVideo2.Camera = 2
    m_spel.SpelVideoControl = SPELVideo1
    m_spel.SpelVideoControl = SPELVideo2
End Sub
```

C# Example:

```
private void Form_Load)object sender, EventArgs e)
{
        RCAPINet.Spel m_spel = new RCAPINet.Spel();
        m_spel.Initialize();
        m_spel.Project = @"c:\\EpsonRC80\\project\\test\\test.sprj";
        SpelVideo1.VideoEnabled = true;
        SpelVideo1.GraphicsEnabled = true;
        SpelVideo1.Camera = 1;
        SpelVideo2.VideoEnabled = true;
        SpelVideo2.GraphicsEnabled = true;
        SpelViodeo2.Camera = 2;
        m_spel.SpelVdieoControl = SPELVideo1;
        m_spel.SpelVideoControl = SPELVideo2;
}
```

13. Using AsyncMode

AsyncMode allows you to execute Spel methods while other methods are executing. Only the following Spel class methods are allowed to execute asynchronously:

13.1 Using AsyncMode

Arc	Jump3
Arc3	Jump3CP
BGo	MCal
BMove	Move
CVMove	Pass
Go	PTran
Home	Pulse
JTran	TGo
Jump	TMove

To execute a method asynchronously, set the AsyncMode property to True, then execute the method. When the AsyncMode property is true and you execute an asynchronous method, the method will be started and control will return immediately back to the .NET application for further processing. If you execute another asynchronous method while a previous one is executing, SPEL will wait for the first method to complete, then start the next method and return back to .NET.

To wait for an asynchronous method to complete, you can use one of the following:

- Execute the WaitCommandComplete method.
- Set AsyncMode property to False.

If an asynchronous command cannot be started due to an error (e.g. point does not exist), then an exception will occur immediately. However, if an error occurs during a command running asynchronously, the error exception occurs on the next execution of an asynchronous command or execution of WaitCommandComplete, or AsyncMode is set to False. If the exception occurs on the next command, you do not know which statement caused the error (the previous statement or the current statement). If you need to check if an asynchronous command completed successfully before executing another command, then call WaitCommandComplete before the next command. If an error occurred during the previous asynchronous command, a SpelException exception will occur with the error number and message. See the example below.

VB Example:

```
Try
   m_spel.AsyncMode = True
   m_spel.Go(1)
   ' do other things here during motion
   ' When Go(2) executes, an exception occurs if Go(1) had an error during
execution,
   '  so we don't know if error occurred for Go(1) or Go(2)
   m_spel.Go(2)

   m_spel.Go(3)
   ' do other things here during motion
   ' Check if Go(3) was successful
   m_spel.WaitCommandComplete() ' Exception occurs if Go(3) had an error

   m_spel.Go(4)
Catch ex As SpelException
   ' Handle the error exception

End Try
```

C# Example:

```
try {
  m_spel.AyncMode = true;
  m_spel.Go(1);
  //do things here during motion
  //When Go(2) executes, an exception occurs if Go(1) had an error during
execution,
  //so we don’t know if error occurred for Go(1) or Go(2)
  m_spel.Go(2);

  m_spel.Go(3);
  //do things here during motion
  //Check if Go(3) was successful
  m_spel.WaitCommandComplete();
  //Exception occurs if Go(3) had an error

  m_spel.Go(4);
}
catch(RCAPINet.SpelException ex) {
  //Handle the error exception
}
```

14. SPELCom_Event

Generates a user event from a Spel class instance.

14.1 SPELCom_Event

Syntax

SPELCom_Event eventNumber [, msgArg1, msgArg2, msgArg3,...]

Parameters

- Event Number
  An expression or integer from 1000 to 32767.
- msgArg1, msgArg2, msgArg3...
  Optional. Each message argument can be either a number, string literal, or a variable name.

  Description

  This instruction makes it easy to send real time information to an application from a Spel task running in the Controller. For example, you can update parts count, lot number, etc. by sending an event.

SPELCom_Event Use Example

In this example, a SPEL+ task sends cycle data to an application using the RC+ API .

```
Function RunParts
    Integer cycNum
    String lot$
    Double cycTime

    cycNum = 0
    Do
        TmrReset(0)
        …
        …
        cycTime = Tmr(0)
        cycNum = cycNum + 1
        Spelcom_Event 3000, cycNum, lot$, cycTime
        Wait 0.01
    Loop
Fend
```

15. RCAPINet Reference

15.1 Spel Class

Description

This class allows you to execute commands and receive events from Epson RC+ 8.0.

File Name

RCAPINt2.dll (64-bit and 32-bit)

15.2 Spel Class Properties

15.2.1 AsyncMode Property

Description

Sets / returns asynchronous execution mode.

Syntax

Property AsyncMode As Boolean

Default value

False

Return Value

A Boolean value that is True if asynchronous mode is active, False if not.

See Also

Using AsyncMode, **WaitCommandComplete method.**

AsyncMode Example

VB Example:

```
With m_spel
  .AsyncMode = True
  .Jump("pick")
  .Delay(500)
  .On(1)
  .WaitCommandComplete()
End With
```

C# Example:

```
m_spel.AsyncMode = true;
m_spel.Jump("pick");
m_spel.Delay(500);
m_spel.On(1);
m_spel.WaitCommandComplete();
```

15.2.2 AvoidSingularity Property

Description
Sets / returns singularity avoidance mode.

Syntax
Property AvoidSingularity As Boolean

Default value
False

Return Value
A Boolean value that is True if singularity avoidance is active, False if not.

See Also
Go Method, **Jump Method**, **Move Method**

AvoidSingularity Example
VB Example:

```
m_spel.AvoidSingularity = True
```

C# Example:

```
m_spel.AvoidSingularity = true;
```

15.2.3 CommandInCycle Property

Description

Returns whether a method is being executed.

Syntax

ReadOnly Property CommandInCycle As Boolean

Return value

A Boolean value that is True if a method is executing, False if not.

See Also

AsyncMode Property

CommandInCycle Example

VB Example:

```
If m_spel.CommandInCycle Then
    MsgBox "A SPEL command is executing, operation aborted"
End If
```

C# Example:

```
if (m_spel.CommandInCycle)
    MessageBox.Show("SPEL command is executing, operation aborted");
```

15.2.4 CommandTask Property

Description

Specifies the reserved API task to use in the Controller for executing robot commands.

Syntax

Property CommandTask As Integer

Default Value

The default value is 0 (do not use a reserved API task).

Remarks

Use CommandTask when you want to execute Spel robot commands on another thread in the Controller. Normally, CommandTask is used on a multi-robot system. Before using CommandTask, you must first reserve tasks in the Controller to be used by the API from Epson RC+ menu -[Setup]-[System Configuration]-[Controller]-[Preferences]. You can reserve up to 16 API tasks in the Controller.

See Also

ServerInstance Property

CommandTask Example

VB Example:

```
' In Robot1 thread
m_spel.CommandTask = 1
m_spel.Robot = 1

' In Robot2 thread
m_spel.CommandTask = 2
m_spel.Robot = 2

C# Example:
// In Robot1 thread
m_spel.CommandTask = 1;
m_spel.Robot = 1;

// In Robot2 thread
m_spel.CommandTask = 2;
m_spel.Robot = 2;
```

15.2.5 DisableMsgDispatch Property

Description
Sets / returns whether Windows messages should be processed during Spel method execution.

Syntax
DisableMsgDispatch

Type
Boolean

Default Value
False

Remarks
This property should normally not be used. It is intended for special applications that do not want keyboard or mouse processing while a Spel method is executing.

15.2.6 ErrorCode Property

Description

Returns the current Controller error code.

Syntax

ReadOnly Property ErrorCode As Integer

Return Value

Integer value containing the error code.

See Also

ErrorOn Property

ErrorCode Example

VB Example:

```
If m_spel.ErrorOn Then
    lblErrorCode.Text = m_spel.ErrorCode.ToString()
Else
    lblErrorCode.Text = ""
End If

C# Example:
if (m_spel.ErrorOn)
    lblErrorCode.Text = m_spel.ErrorCode.ToString();
else
    lblErrorCode.Text = "";
```

15.2.7 ErrorOn Property

Description

Returns True if a critical error has occurred in the Controller.

Syntax

ReadOnly Property ErrorOn As Boolean

Return value

True if the Controller is in the error state, False if not.

Remarks

When the Controller is in the error state, the ErrorOn property returns True, and you can retrieve the error code by using the ErrorCode property.

See Also

ErrorCode Property

ErrorOn Example

VB Example:

```
If m_spel.ErrorOn Then
    m_spel.Reset
End If

C# Example:
if (m_spel.ErrorOn)
    m_spel.Reset();
```

15.2.8 EStopOn Property

Description

Returns the status of the Controller's emergency stop.

Syntax

ReadOnly Property EStopOn As Boolean

Return Value

True if the emergency stop is active, False if not.

EStopOn Example

VB Example:

```
If m_spel.EStopOn Then
    lblEStop.Text = "Emergency stop is active"
Else
    lblEStop.Text = ""
EndIf
```

C# Example:

```
if (m_spel.EStopOn)
    lblEStop.Text = "Emergency stop is active";
else
    lblEStop.Text = "";
```

15.2.9 MotorsOn Property

Description

Sets and return the status of the motor power on or off for the current robot.

Syntax

Property MotorsOn As Boolean

Default value

False

Return Value

A Boolean value that is True if motors are on, False if not.

See Also

PowerHigh Property, **Reset Method**, **Robot Property**

MotorsOn Example

VB Example:

```
If Not m_spel.MotorsOn Then
   m_spel.MotorsOn = True
End If
```

C# Example:

```
if (\!m_spel.MotorsOn)
   m_spel.MotorsOn = true;
```

15.2.10 NoProjectSync Property

Description
Sets / returns whether the current project in the PC should be synchronized with the Controller project.

Syntax
NoProjectSync

Type
Boolean

Default Value
False

Remarks
When NoProjectSync is set to False (default), then the Spel class ensures that the project on the PC is synchronized with the project on the Controller.
When NoProjectSync is set to True, the Spel class does not check for any project on the PC and does not synchronize the PC project with the Controller. This allows you to run programs in the Controller without any project on the PC.

This property is not persistent. You must set it after creating a Spel class instance if you want to set it to True.

See Also
Start Method

NoProjectSync Examples
VB Example:

```
m_spel.Initialize()
m_spel.NoProjectSync = True
```

C# Example:

```
m_spel.Initialize();
m_spel.NoProjectSync = true;
```

15.2.11 OperationMode Property

Description

Reads or sets the Epson RC+ 8.0 mode of operation.

Syntax

Property OperationMode As SpelOperationMode

Return value

SpelOperationMode value

Remarks

When OperationMode is set to Program, the Epson RC+ 8.0 GUI for the current instance of the Spel class is opened and the Controller operation mode is set to Program. If the user closes the GUI, OperationMode is set to Auto. If OperationMode is set to Auto from Visual Basic, the GUI also closes.

OperationMode Example

VB Example:

```
Sub btnSpelProgramMode_Click _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnHideIOMonitor.Click

  Try
    m_spel.OperationMode = _
                 RCAPINet.SpelOperationMode.Program
    ' If you want to wait for the user to close the RC+ GUI,
    ' you can wait here for OperationMode to change to Auto
    Do
      Application.DoEvents()
      System.Threading.Thread.Sleep(10)
    Loop Until m_spel.OperationMode = _
                  RCAPINet.SpelOperationMode.Auto
  Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
  End Try
End If
```

C# Example:

```
void btnSpelProgramMode_Click(object sender, EventArgs e)
{
  try {
m_spel.OperationMode = RCAPINet.SpelOperationMode.Auto;
// If you want to wait for the user to close the RC+ GUI, you can wait here
    Do {
       Application.DoEvents();
       Systme.Threading.Thread.Sleep(10);
    } while(\!m_spel.OperationMode = RCAPINet.OperationMode.Auto);
  }
  Catch (SpelException ex){
    MessageBox.Show(ex.Message);
  }
}
```

15.2.12 ParentWindowHandle Property

Description

Sets / returns the handle for the parent window used for dialogs and windows.

Syntax

Property ParentWindowHandle As Integer

Return Value

Integer value containing the window handle.

Remarks

Use ParentWindowHandle to specify the parent window from applications that do not have .NET forms.

See Also

ParentWindowHandle Example

VB Example:

```
m_spel.ParentWindowHandle = Me.Handle
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor)
```

C# Example:

```
m_spel.ParentWindowHandle  = (int)this.Handle;
m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor);
```

15.2.13 PauseOn Property

Description

Returns status of the Controller pause state.

Syntax

ReadOnly Property PauseOn As Boolean

Return Value

True if the Controller is in the pause state, False if not.

See Also

Continue Method, **Pause Method**

PauseOn Example

VB Example:

```
If m_spel.PauseOn Then
    btnPause.Enabled = False
    btnContinue.Enabled = True
End If
```

C# Example:

```
if (m_spel.PauseOn){
    btnPause.Enabled = false;
    btnContinue.Enabled = true;
}
```

15.2.14 PowerHigh Property

Description

Sets and returns the power state for the current robot.

Syntax

Property PowerHigh As Boolean

Default Value

False

Return Value

True if the current robot power is high, False if not.

See Also

MotorsOn Property

PowerHigh Example

VB Example:

```
If Not m_spel.PowerHigh Then
    m_spel.PowerHigh = True
End If
```

C# Example:

```
if (\!m_spel.PowerHigh)
    m_spel.PowerHigh = true;
```

15.2.15 Project Property

Description
Sets / returns the current project.

Syntax
Property Project As String

Default Value
Empty string.

Return Value
A string containing the project path and file.

Remarks
When setting the Project, you must supply the full path and name of the Epson RC+ 8.0 project make file. The make file is the project name with a .SPRJ extension.

Project Example
VB Example:

```
m_spel.Project = "c:\EpsonRC80\projects\myapp\myapp.sprj"
```

C# Example:

```
m_spel.Project = @"c:\EpsonRC80\projects\myapp\myapp.sprj";
```

15.2.16 ProjectBuildComplete Property

Description

Returns the status of the current project build.

Syntax

ReadOnly Property ProjectBuildComplete As Boolean

Return Value

True if the project build is complete, False if not.

See Also

.BuildProject Method

ProjectBuildComplete Example

VB Example:

```
If m_spel.ProjectBuildComplete Then
    lblBuild.Text = "Project build is Complete"
Else
    lblBuild.Text = "Project build is not Complete"
End If
```

C# Example:

```
if (m_spel.ProjectBuildComplete)
    lblBuild.Text = "Project build is Complete";
else
    lblBuild.Text = "Project build is not Complete";
```

15.2.17 ProjectOverwriteWarningEnabled Property

Description
Sets / returns whether the project overwrite warning will be displayed.

Syntax
Property ProjectOverwriteWarningEnabled As Boolean

Default Value
True

Return Value
True if the project overwrite warning is enabled, False if not.

See Also
.BuildProject Method

Remarks
By default, when the current project is not the same as the project in the controller, then a project overwrite warning message is displayed when the project is built and sent to the controller. Set ProjectOverwriteWarningEnabled to False when you don't want the overwrite warning message to be displayed. This is useful for when your application needs to switch projects used in the controller.

ProjectOverwriteWarningEnabled Example
VB Example:

```
' Disable the project overwrite warning
m_spel.ProjectOverwriteWarningEnabled = False
m_spel.Project = "c:\EpsonRC80\Projects\Project1\Project1.sprj"
```

C# Example:

```
// Disable the project overwrite warning
m_spel.ProjectOverwriteWarningEnabled = false;
m_spel.Project = @"c:\EpsonRC80\Projects\Project1\Project1.sprj";
```

15.2.18 ResetAbortEnabled Property

Description

Sets / returns whether ResetAbort method should be enabled or not.

Syntax

Property ResetAbortEnabled As Boolean

Default Value

True

Return Value

True if ResetAbort is enabled, False if not.

See Also

ResetAbort Method

ResetAbortEnabled Example

VB Example:

```
' Enable reset abort
m_spel.ResetAbortEnabled = True
```

C# Example:

```
// Enable reset abort
m_spel.ResetAbortEnabled = true;
```

15.2.19 Robot Property

Description
Sets / returns the current robot number.

Syntax
Property Robot As Integer

Default Value
If one or more robots exists, the default value for the first Spel instance is 1, otherwise it is 0. For all other Spel instances, the default value is 0.

Return Value
Integer value that contains the current robot number.

Remarks
On a system using multiple robots, use the Robot property to set the robot for subsequent robot related commands, such as motion commands.

See Also
RobotModel Property, **RobotType Property**

Robot Example
VB Example:

```
m_spel.Robot = 2
If Not m_spel.MotorsOn Then
  m_spel.MotorsOn = True
End If
```

C# Example:

```
m_spel.Robot = 2;
if (\!m_spel.MotorsOn)
  m_spel.MotorsOn = true;
```

15.2.20 RobotModel Property

Description

Returns the model name for the current robot.

Syntax

ReadOnly Property RobotModel As String

Return Value

String that contains the current robot's model name.

See Also

Robot Property, **RobotType Property**

RobotModel Example

VB Example:

```
lblRobotModel.Text = m_spel.RobotModel
```

C# Example:

```
lblRobotModel.Text = m_spel.RobotModel;
```

15.2.21 RobotType Property

Description

Returns the type of the current robot.

Syntax

ReadOnly Property RobotType As SpelRobotType

Return Value

SpelRobotType value

See Also

Robot Property, **RobotModel Property**

RobotType Example

VB Example:

```
Select Case m_spel.RobotType
  Case RCAPINet.SpelRobotType.Scara
    lblRobotType.Text = "Scara"
  Case RCAPINet.SpelRobotType.Cartesian
    lblRobotType.Text = "Cartesian"
End Select
```

C# Example:

```
switch (m_spel.RobotType)
{
    case SpelRobotType.Scara:
         lblRobotType.Text = "Scara";
break;
case SpelRobotType.Cartesian:
         lblRobotType.Text = "Cartesian";
break;
    default:
         break;
}
```

15.2.22 SafetyOn Property

Description
Returns status of the Controller's safeguard input.

Syntax
ReadOnly Property SafetyOn As Boolean

Return Value
True if the safeguard is open, False if not.

Remarks
Use the SafetyOn property to obtain the safeguard status when your application starts, then use the SafeguardOpen and SafeguardClose events to update the status.

SafetyOn Example
VB Example:

```
If m_spel.SafetyOn Then
    lblSafeguard.Text = "Safe guard is active"
Else
    lblSafeguard.Text = ""
End If
```

C# Example:

```
if (m_spel.SafetyOn)
    lblSafeguard.Text = "Safe guard is active";
else
    lblSafeguard.Text = "";
```

15.2.23 ServerInstance Property

Description
Specifies which instance of Epson RC+ server to use.

Syntax
Property ServerInstance As Integer

Default Value
The default value is the next available server instance.

Remarks
The API communicates with an RC+ server process. ServerInstance specifies which server to use. Each server instance corresponds with one controller and one project. By default, when you create a new Spel class instance, the ServerInstance is automatically set to 1.
Sometimes you may want multiple instances of the Spel class for the same Controller, such as for multithreading in your application. In that case, you must set the ServerInstance property to the same value for each Spel class instance using the same controller.
ServerInstance must be between 1 and 10 and it must be set before executing Initialize or any other methods.

See Also
CommandTask Property, **Initialize Method**

ServerInstance Example
VB Example:

```
' Controller 1
spel1 = New Spel
spel1.ServerInstance = 1
spel1.Initialize()
spel1.Connect(1)

' Controller 2
spel2 = New Spel
spel2.ServerInstance = 2
spel2.Initialize()
spel2.Connect(2)
```

C# Example:

```
// Controller 1
RCAPINet.Spel spel1 = new RCAPINet.Spel();
spel1.ServerInstance = 1;
spel1.Initialize();
spel1.Connect(1);

// Controller 2
RCAPINet.Spel spel2 = new RCAPINet.Spel();
spel2.ServerInstance = 2;
spel2.Initialize();
spel2.Connect(2);
```

15.2.24 SPELVideoControl Property

Description

Used to connect a SPELVideo control to the Spel class instance so that video and graphics can be displayed.

Syntax

Property SpelVideoControl As SpelVideo

See Also

GraphicsEnabled Property, **VideoEnabled Property**, **Camera Property**

SpelVideoControl Example

VB Example:

```
m_spel.SpelVideoControl = SpelVideo1
```

C# Example:

```
m_spel.SpelVideoControl = SpelVideo1;
```

15.2.25 Version Property

Description

Returns the current Epson RC+ 8.0 software version.

Syntax

ReadOnly Property Version As String

Return Value

String that contains the current Epson RC+ 8.0 software version.

Version Example

VB Example:

```
' Get version of software
curVer = m_spel.Version
```

C# Example:

```
// Get version of software
curVer = m_spel.Version;
```

15.2.26 WarningCode Property

Description

Returns Controller warning code.

Syntax

ReadOnly Property WarningCode As Integer

Return Value

Integer value that contains the current controller warning code.

See Also

WarningOn Property

WarningCode Example

VB Example:

```
If m_spel.WarningOn Then
    lblWarningCode.Text = m_spel.WarningCode.ToString()
Else
    lblWarningCode.Text = ""
End If
```

C# Example:

```
if (m_spel.WarningOn)
    lblWarningCode.Text = m_spel.WarningCode.ToString();
else
    lblWarningCode.Text = "";
```

15.2.27 WarningOn Property

Description

Returns status of the Controller warning state.

Syntax

ReadOnly Property WarningOn As Boolean

Return Value

True if the Controller is in the warning state, False if not.

See Also

WarningCode Property

WarningOn Example

VB Example:

```
If m_spel.WarningOn Then
    lblWarningStatus.Text = "ON"
Else
    lblWarningStatus.Text = "OFF"
End If
```

C# Example:

```
if (m_spel.WarningOn)
    lblWarningStatus.Text = "ON";
else
    lblWarningStatus.Text = "OFF";
```

15.3 Spel Class Methods

15.3.1 A

15.3.1.1 Accel Method

Description

Sets acceleration and deceleration for point to point motion commands Go, Jump, and Pulse.

Syntax

Sub Accel (PointToPointAccel As Integer, PointToPointDecel As Integer, _[JumpDepartAccel As Integer], [JumpDepartDecel As Integer], _[JumpApproAccel As Integer], [JumpApproDecel As Integer])

Parameters

- PointToPointAccel
  An integer greater than or equal to 1 indicating the percentage of maximum acceleration rate (units: %)
- PointToPointDecel
  An integer greater than or equal to 1 indicating the percentage of maximum deceleration rate (units: %)
- JumpDepartAccel
  Optional. An integer greater than or equal to 1 indicating the depart acceleration for Jump.
- JumpDepartDecel
  Optional. An integer greater than or equal to 1 indicating the depart deceleration for Jump.
- JumpApproAccel
  Optional. An integer greater than or equal to 1 indicating the approach acceleration for Jump.
- JumpApproDecel
  Optional. An integer greater than or equal to 1 indicating the approach deceleration for Jump.

See Also

AccelS Method, **Speed Method**

Accel Example

VB Example:

```
m_spel.Accel(50, 50)
m_spel.Go ("pick")
```

C# Example:

```
m_spel.Accel(50, 50);
m_spel.Go ("pick");
```

15.3.1.2 AccelR Method

Description

Sets acceleration and deceleration for tool rotation motion.

Syntax

Sub AccelR (Accel As Single, [Decel As Single])

Parameters

- Accel
  A real number from 0.1 to 5000 (units: deg/sec2) that defines the tool rotation acceleration when ROT is used in motion commands. If Decel is omitted, this value is used for both the Acceleration and Deceleration rates.
- Decel
  A real number from 0.1 to 5000 (units: deg/sec2) that defines the tool rotation deceleration when ROT is used in motion commands. Optional.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Arc Method, **Arc3 Method**, **BMove Method**, **Jump3CP Method**, **SetPoint Method**, **SpeedR Method**, **TMove Method**

AccelR Example

VB Example:

```
Sub MoveToPlace()
    m_spel.AccelR(100)
    m_spel.Move("place ROT")
End Sub
```

C# Example:

```
void MoveToPlace()
{
        m_spel.AccelR(100);
        m_spel.Move("place ROT");
}
```

15.3.1.3 AccelS Method

Description

Sets acceleration and deceleration for linear interpolar (straight line) motion commands Jump3CP, Move, TMove.

Syntax

Sub AccelS (Accel As Single, Decel As Single,[JumpDepartAccel As Single], [JumpDepartDecel As Single], _[JumpApproAccel As Single], [JumpApproDecel As Single])

Parameters

- Accel
  A real number (units: mm/s2) that specifies the acceleration for linear or CP motion. If Decel is omitted, this value is applied for both the acceleration and deceleration rates.
- Decel
  Optional. A real number (units: mm/sec2) that specifies the deceleration for linear or CP motion.
- JumpDepartAccel
  Optional. A real number (units: mm/sec2) that specifies the depart acceleration for depart operation for Jump or Jump3CP.
- JumpDepartDecel
  Optional. A real number (units: mm/sec2) that specifies the depart deceleration for depart operation for Jump or Jump3CP.
- JumpApproAccel
  Optional. A real number (units: mm/sec2) that specifies the approach acceleration for approach operation for Jump or Jump3CP.
- JumpApproDecel
  Optional. A real number (units: mm/sec2) that specifies the approach deceleration for approach operation for Jump or Jump3CP.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Accel Method, **SpeedS Method**, **SetPoint Method**, **Jump3CP Method**, **Move Method**, **TMove Method**

AccelS Example

VB Example:

```
Sub MoveToPlace()
    m_spel.Base(P1)
    m_spel.Move(pick)
    m_spel.AccelS(500, 300)
    m_spel.Move(place)
End Sub
```

C# Example:

```
void MoveToPlace()
{
    m_spel.AccelS(500);
    m_spel.Move(pick);
    m_spel.AccelS(500, 300);
    m_spel.Move(place);
}
```

15.3.1.4 Agl Method

Description

Returns the joint angle for the selected rotational axis, or position for the selected linear axis.

Syntax

Function Agl (JointNumber As Integer) As Single

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)

Return Value

Returns the joint angle for the selected rotational axis, or position for the selected linear axis.

See Also

Pls Method, **CX, CY, CZ, CU, CV, CW, CR, CS, CT Method**

Agl Example

C# Example:

```
Dim j1Angle As Single
j1Angle = m_spel.Agl(1)
```

C# Example:

```
float j1Angle;
j1Angle = m_spel.Agl(1);
```

15.3.1.5 AIO_In Method

Description

Reads analog value from optional analog I/O input channel.

Syntax

Function AIO_In (Channel As Integer) As Single

Parameters

- Channel
  Specifies the analog I/O channel number.

Return Value

Return the analog input value of the analog I/O channel which specified in channel number in real number. Return value range differs depending on the input range configuration of the analog I/O board.

See Also

AIO_InW Method, **AIO_Out Method**, **AIO_OutW Method**

AIO_In Use Example

VB Examples:

```
Dim val As Single
val = m_spel.AIO_In(2)
```

C# Example:

```
float val;
val = m_spel.AIO_In(2);
```

15.3.1.6 AIO_InW Method

Description

Reads analog value from optional analog I/O input channel.

Syntax

Function AIO_InW (Channel As Integer) As Integer

Parameters

- Channel
  Specifies the analog I/O channel number.

Return Value

Returns the input states (Integer expression from 0 to 65535) of specified analog I/O channel.

The following table shows input voltage (current) and return value of each input channel according to input range configuration of analog I/O board.

Input Data		Input Range Configuration				
Hexadecimal	Decimal	±10.24(V)	±5.12(V)	0-5.12(V)	0-10.24(V)	0-24(mA)
0xFFFF	65535	10.23969	5.11984	5.12000	10.24000	24.00000
0x8001	32769	0.00031	0.00016	2.56008	5.12016	12.00037
0x8000	32768	0.00000	0.00000	2.56000	5.12000	12.00000
0x0000	0	-10.24000	-5.12000	0.00000	0.00000	0.00000

See Also

AIO_In Method, **AIO_Out Method**, **AIO_OutW Method**

AIO_InW Use Example

VB Examples:

```
Dim val As Integer
val = m_spel.AIO_InW(2)
```

C# Example:

```
int val;
val = m_spel.AIO_InW(2);
```

15.3.1.7 AIO_Out Method

Description

Reads or set analog value from the optional analog I/O output channel.

Syntax

Function AIO_Out (Channel As Integer) As Single
Sub AIO_Out (Channel As Integer, Value As Single)

Parameters

- Channel
  Specifies the analog I/O channel number.
- Value
  An expression or number that specifies the real number indicating the voltage [V] or current [mA] you want to output.

Return Value

Returns specified analog I/O channel voltage and current output state in real number. Unit of voltage output is [V] and current output is [mA].
For Function AIO_Out (Channel As Integer) As Single: When outputting the speed information of the robot on specified channel, a return value can be acquired by this method.

Remarks

Output the real value indicating specified voltage [V] or current [mA] to analog output port which specified on channel port. Set the voltage output range of analog output port or selection of voltage and current output by the switch on the board. If setting a value which out of range of analog I/O port, output the border value (maximum and minimum value) which is not out of the range.
For Sub AIO_Out (Channel As Integer, Value As Single): The output setting of this method becomes an error if outputting the speed information by specified channel. Stop the speed information output and execute this method.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AIO_In Method, **AIO_InW Method**, **AIO_OutW Method**, **SetPoint Method**

AIO_Out Use Example

VB Examples:

```
Dim val As Single
val = m_spel.AIO_Out(1)
```

C# Example:

```
float val;
val = m_spel.AIO_Out(1);
```

15.3.1.8 AIO_OutW Method

Description

Reads or set analog value from the optional analog I/O output channel.

Syntax

Function AIO_OutW (Channel As Integer) As Integer
Sub AIO_OutW (Channel As Integer, OutputData As Integer)

Parameters

- Channel
  Specifies the analog I/O channel number.
- OutputData
  An expression or number that specifies the output data (0 to 65535).

Return Value

Returns the output state of specified analog I/O channel in Long integers from 0 to 65535.
The following table shows output voltage (current) and return value of each output channel according to output range configuration of analog I/O board.

Output Data		Output Range Configuration					
Hexadecimal	Decimal	±10(V)	±5(V)	0-5(V)	0-10(V)	4-20(mA)	0-20(mA)
0xFFFF	65535	9.99970	4.99985	5.00000	10.00000	20.00000	20.00000
0x8001	32769	0.00031	0.00015	2.50008	5.00015	12.00024	10.00031
0x8000	32768	0.00000	0.00000	2.50000	5.00000	12.00000	10.00000
0x0000	0	-10.00000	-5.00000	0.00000	0.00000	4.00000	0.00000

This method is available when outputting the speed information of the robot on specified channel.

See Also

AIO_In Method, **AIO_InW Method**, **AIO_Out Method**

AIO_OutW Use Example

VB Examples:

```
Dim val As Integer
val = m_spel.AIO_OutW(1)
```

C# Example:

```
int val;
val = m_spel.AIO_OutW(1);
```

15.3.1.9 Arc Method

Description

Arc moves the arm to the specified point using circular interpolation in the XY plane.

Syntax

Sub Arc (MidPoint As Integer, EndPoint As Integer)
Sub Arc (MidPoint As SpelPoint, EndPoint As SpelPoint)
Sub Arc (MidPoint As String, EndPoint As String)

Parameters

Each syntax has two parameters that specify the mid point and end point of the arc.

- MidPoint
  An integer, SpelPoint or character string indicating the mid point.
- EndPoint
  Specifies the target point using an integer, SpelPoint or character string. When using a character string, you can include ROT, CP, SYNC, Till, Find, and a parallel processing statement.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelR Method, **AccelS Method**, **SpeedR Method**, **SpeedS Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Find Method**, **Till Method**, **SetPoint Method**

Arc Example

VB Example:

```
' Points specified using SpelPoint
Dim midPoint, endPoint As SpelPoint
midPoint = m_spel.GetPoint("P1")
endPoint = m_spel.GetPoint("P2")
m_spel.Arc(midPoint, endPoint)

' Points specified using expressions
m_spel.Arc("P1", "P2")
m_spel.Arc("P1", "P2 CP")

' Using parallel processing
m_spel.Arc("P1", "P2 \!D50; On 1; D90; Off 1\!")
```

C# Example:

```
// Points specified using SpelPoint
SpelPoint midPoint, endPoint;
midPoint = m_spel.GetPoint("P1");
endPoint = m_spel.GetPoint("P2");
m_spel.Arc(midPoint, endPoint);

// Points specified using expressions
m_spel.Arc("P1", "P2");
m_spel.Arc("P1", "P2 CP");

// Using a parallel processing
m_spel.Arc("P1", "P2 \!D50; On 1; D90; Off 1\!");
```

15.3.1.10 Arc3 Method

Description

Arc3 moves the arm to the specified point using circular interpolation in 3 dimensions.

Syntax

Sub Arc3 (MidPoint As Integer, EndPoint As Integer)
Sub Arc3 (MidPoint As SpelPoint, EndPoint As SpelPoint)
Sub Arc3(MidPoint As String, EndPoint As String)

Parameters

Each syntax has two parameters that specify the mid point and end point of the arc.

- MidPoint
  An integer, SpelPoint or character string indicating the mid point.
- EndPoint
  Specifies the target point using an integer, SpelPoint or character string. When using a character string, you can include ROT, CP, SYNC, Till, Find, and a parallel processing statement.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelR Method, **AccelS Method**, **SpeedR Method**, **SpeedS Method**, **Arc Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Find Method**, **Till Method**, **SetPoint Method**

Arc3 Example

VB Example:

```
' Points specified using SpelPoint
Dim midPoint, endPoint As SpelPoint
midPoint = m_spel.GetPoint("P1")
endPoint = m_spel.GetPoint("P2")
m_spel.Arc3(midPoint, endPoint)

' Points specified using expressions
m_spel.Arc3("P1", "P2")
m_spel.Arc3("P1", "P2 CP")

' Using parallel processing
m_spel.Arc3("P1", "P2 \!D50; On 1; D90; Off 1\!")
```

C# Example:

```
// Points specified using SpelPoint
SpelPoint midPoint, endPoint;
midPoint = m_spel.GetPoint("P1");
endPoint = m_spel.GetPoint("P2");
m_spel.Arc3(midPoint, endPoint);

// Points specified using expressions
m_spel.Arc3("P1", "P2");
m_spel.Arc3("P1", "P2 CP");

// Using a parallel processing
m_spel.Arc3("P1", "P2 \!D50; On 1; D90; Off 1\!");
```

15.3.1.11 Arch Method

Description

Defines ARCH parameters (Z height to move before beginning horizontal motion) for use with the JUMP instructions.

Syntax

Sub Arch (ArchNumber As Integer, DepartDist As Integer, ApproDist As Integer)

Parameters

- ArchNumber
  An integer indicating the arch number from 0 to 6.
- DepartDist
  The depart distance in millimeters moved at the beginning of the Jump instruction before starting horizontal motion.
  For the Jump3 and Jump3CP commands: the depart distance in mm moved before starting span motion.
- ApproDist
  The approach distance in millimeters above the target position of the Jump instruction.
  For the Jump3 and Jump3CP commands: the approach distance in mm moved when span motion has completely finished.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Jump Method, **Jump3 Method**, **Jump3CP Method**, **SetPoint Method**

Arch Example

VB Example:

```
Sub SetArchs()
    With m_spel
        .Arch(1, 30, 30)
        .Arch(2, 60, 60)
        .Jump("P1 C1")
        .Jump("P2 C2")
    End With
End Sub
```

C# Example:

```
void SetArchs()
{
    m_spel.Arch(1, 30, 30);
    m_spel.Arch(2, 60, 60);
    m_spel.Jump("P1 C1");
    m_spel.Jump("P2 C2");
}
```

15.3.1.12 Arm Method

Description

Selects the current robot arm.

Syntax

Sub Arm (ArmNumber As Integer)

Parameters

- ArmNumber
  An integer or formula (valid range: 0 to 15) to select up to 16 different arms. Arm 0 is the standard (default) robot arm. Arm(s) 1-15 are auxiliary arms defined by the ArmSet instruction.

See Also

ArmSet Method, **GetArm Method**, **Tool Method**

Arm Example

VB Example:

```
m_spel.Arm(1)
```

C# Example:

```
m_spel.Arm(1);
```

15.3.1.13 ArmClr Method

Description

Clears (undefines) an arm for the current robot.

Syntax

Sub ArmClr (ArmNumber As Integer)

Parameters

- ArmNumber
  An integer or formula (valid range: 1 to 15) that specifies the arm number for which to clear the settings. As Arm 0 is the standard (default) setting, it cannot be cleared. Arm(s) 1-15 are auxiliary arms defined by the ArmSet instruction.

See Also

ArmSet Method, **GetArm Method**, **Tool Method**

ArmClr Example

VB Example:

```
m_spel.ArmClr(1)
```

C# Example:

```
m_spel.ArmClr(1);
```

15.3.1.14 ArmDef Method

Description

Returns whether a robot arm is defined or not.

Syntax

Function ArmDef (ArmNumber As Integer) As Boolean

Parameters

- ArmNumber
  An integer between 0 and 15. Arm 0 is the standard (default) robot arm and is always set. Arm(s) 1-15 are auxiliary arms defined by using the ArmSet method.

Return Value

True if the specified arm is defined, False if not.

See Also

ArmSet Method, **GetArm Method**, **Tool Method**

ArmDef Example

VB Example:

```
x = m_spel.ArmDef(1)
```

C# Example:

```
x = m_spel.ArmDef(1);
```

15.3.1.15 ArmSet Method

Description

Defines an auxiliary robot arm.

Syntax

Sub ArmSet (ArmNumber As Integer, Param1 As Single, Param2 As Single, Param3 As Single, Param4 As Single, Param5 As Single)

Parameters

- ArmNumber
  Integer (valid range: 1 to 15)
- Param1
  (For SCARA Robots) The horizontal distance from the center line of the elbow joint to the center line of the new orientation axis (i.e. the position where the center line of the new auxiliary arm orientation axis is located). (For Cartesian Robots) The positional offset in the X-axis direction (units: mm).
- Param2
  (For SCARA Robots) The offset (in degrees) between the line subtended between the standard elbow joint center line and the center of the orientation axis and the line subtended between the new auxiliary arm elbow joint center line and the center of the orientation axis. (These 2 lines should intersect at the elbow center line and the angle formed is the Param2.) (For Cartesian Robots) Y axis direction position offset from the original Y position specified in mm.
- Param3
  (For SCARA & Cartesian Robots) The Z height offset difference between the new orientation axis center line and the old orientation axis center line.
- Param4
  (For SCARA Robots) The distance from the shoulder center line to the elbow center line of the new auxiliary arm. (For Cartesian Robots) This is a dummy parameter (Specify 0).
- Param5
  (For SCARA & Cartesian Robots) The angular offset (in degrees) between the new arm orientation axis and the old arm orientation axis.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Arm Method, **Tool Method**, **TLSet Method**, **SetPoint Method**

ArmSet Example

VB Example:

```
Sub SetArms()
    With m_spel
        .ArmSet(1, 1.5, 0, 0, 0, 0)
        .ArmSet(2, 3.2, 0, 0, 0, 0)
    End With
End Sub
```

C# Example:

```
void SetArms()
{
    m_spel.ArmSet(1, 1.5, 0, 0, 0, 0);
    m_spel.ArmSet(2, 3.2, 0, 0, 0, 0);
}
```

15.3.1.16 Atan Method

Description

Returns the arc tangent of a numeric expression.

Syntax

Function Atan (number As Double) As Double

Parameters

- number
  Angle tangent (real number)

Return Value

Arc tangent of the specified value

See Also

Atan2 Method

Atan Example

VB Example:

```
Dim angle As Doubleangle = m_spel.Atan(.7)
```

C# Example:

```
double angle;

angle = m_spel.Atan(.7);
```

15.3.1.17 Atan2 Method

Description

Returns the angle of the imaginary line connecting points (0,0) and (X, Y) in radians.

Syntax

Function Atan2 (Dx As Double, Dy as Double) As Double

Parameters

- Dx
  A real number indicating the X coordinate
- Dy
  A real number indicating the Y coordinate

Return Value

A double value containing the angle.

See Also

Atan Method

Atan2 Example

VB Example:

```
Dim angle As Double

angle = m_spel.Atan2(-25, 50)
```

C# Example:

```
double angle;

angle = m_spel.Atan2(-25, 50);
```

15.3.1.18 ATCLR Method

Description

Clears and initializes the average torque for one or more joints.

Syntax

Sub ATCLR ()

See Also

ATRQ Method, **PTCLR Method**, **PTRQ Method**

ATCLR Example

VB Example:

```
m_spel.ATCLR()
```

C# Example:

```
m_spel.ATCLR();
```

15.3.1.19 AtHome Method

Description

Returns True if the current robot is at the home position.

Syntax

Function AtHome () As Boolean

Return Value

True if the current robot is at it's home position, False if not.

See Also

Home Method

AtHome Example

VB Example:

```
If m_spel.AtHome() Then
    lblCurPos.Text = "Robot is at home position"
Else
    lblCurPos.Text = "Robot is not at home position"
End If
```

C# Example:

```
if(m_spel.AtHome())
    lblCurPos.Text = "Robot is at home position";
else
    lblCurPos.Text = "Robot is not at home position";
```

15.3.1.20 ATRQ Method

Description

Returns the average torque for the specified joint.

Syntax

Function ATRQ (JointNumber As Integer) As Single

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)

Return Value

Real value from 0 to 1.

See Also

ATCLR Method, **PTCLR Method**, **PTRQ Method**

ATRQ Example

VB Example:

```
Dim val As Single
Dim i As Integer
For i = 1 To 4
val = m_spel.ATRQ(i)
Next i
```

C# Example:

```
float avgTorque;
for(int i = 1; i <= 4; i++)
    avgTorque = m_spel.ATRQ(i);
```

15.3.1.21 AvgSpeed Method

Description

Returns the average value of the absolute speed values for the specified joints.

Syntax

Function AvgSpeed (JointNumber As Integer) As Single

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)

Return Value

Real value from 0 to 1.

Remarks

This method returns the average value of the absolute speed values for the specified joints. The loading state of the motor can be obtained by this method. The result is a real value from 0 to 1 with 1 being the maximum average speed value.

You must execute this method before AvgSpeedClear method is executed.

This method is time restricted. You must execute thismethod within 60 seconds after AvgSpeedClear method is executed. When this time is exceeded, error 4088 occurs.

When using the virtual controller or conducting dry-run, the average of the absolute speed values is calculated from the commanded speed instead of the actual speed.
This method does not support the PG additional axes.

See Also

AvgSpeedClear Method, **PeakSpeed Method**, **PeakSpeedClear Method**

AvgSpeed Example

VB Example:

```
Dim val As Single
Dim i As Integer
For i = 1 To 4
val = m_spel.AvgSpeed(i)
Next i
```

C# Example:

```
float avgSpeed;
for(int i = 0; i <=4; i++)
    avgSpeed = m_spel.AvgSpeed(i);
```

15.3.1.22 AvgSpeedClear Method

Description

Clears and initializes the average of the absolute speed values for one or more joints.

Syntax

Sub AvgSpeedClear ()

Remarks

This method clears the average of the absolute speed values for the specified joints.
You must execute this method before executing AvgSpeed method.
This method does not support the PG additional axes.

See Also

AvgSpeed Method, **PeakSpeed Method**, **PeakSpeedClear Method**

AvgSpeedClear Example

VB Example:

```
m_spel.AvgSpeedClear()
```

C# Example:

```
m_spel.AvgSpeedClear();
```

15.3.1.23 AxisLocked Method

Description

Returns True when specified axis release free joint state.

Syntax

Function AxisLocked (AxisNumber As Integer) As Boolean

Parameters

- AxisNumber
  A number between 1 and 9 indicating the axis number

Return Value

True if the specified axis is under servo control.

See Also

Slock Method, **SFree Method**

AxisLocked Example

VB Example:

```
If m_spel.AxisLocked(1) Then
    lblAxis1.Text = "Robot axis \#1 is locked"
Else
    lblAxis1.Text = "Robot axis \#1 is free"
End If
```

C# Example:

```
if (m_spel.AxisLocked(1))
    lblAxis1.Text = "Robot axis \#1 is locked";
else
    lblAxis1.Text = "Robot axis \#1 is free";
```

15.3.2 B

15.3.2.1 Base Method

Description

Defines the base coordinate system.

Syntax

Sub Base (OriginPoint As SpelPoint [, XAxisPoint As SpelPoint] [, YAxisPoint As SpelPoint] [, Alignment As SpelBaseAlignment])

Parameters

- OriginPoint
  Point data that specifies the origin and orientation of the base coordinate system
- XAxisPoint
  Optional. SpelPoint on the X-axis of the base coordinate system.
- YAxisPoint
  Optional. SpelPoint on the Y-axis of the base coordinate system.
- Alignment
  Specifies which axis should be priority matched to the base when the parameters XAxisPoint and YAxisPoint are given.
  Optional.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Local Method, **SetPoint Method**

Base Example

VB Example:

```
Dim originPoint As New SpelPoint
originPoint.X = 50
originPoint.Y = 50
m_spel.Base(originPoint)
```

C# Example:

```
SpelPoint originPoint = new SpelPoint();
originPoint.X = 50;
originPoint.Y = 50;
m_spel.Base(originPoint);
```

15.3.2.2 BGo Method

Description

Executes Point to Point relative motion in the selected local coordinate system.

Syntax

Sub BGo (PointNumber As Integer)
Sub BGo (Point As SpelPoint)
Sub BGo (Point As SpelPoint, AttribExpr As String)
Sub BGo (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the BMove motion. This is the final position at the end of the point to point motion.

- PointNumber
  Specifies the target point using a previously taught point number from the current robot point memory stored in the controller.
- Point
  Specifies the target point using the SpelPoint data type.
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain CP, SYNC, Till or Find search expressions, and parallel processing statements.
- PointExpr
  Specifies the target point using a string expression.
  It can contain CP, SYNC, Till or Find search expressions, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Accel Method, **Speed Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Find Method**, **Till Method**, **SetPoint Method**

BGo Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.BGo(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.BGo(pt)

' Using an attribute expression
m_spel.BGo(pt, "Till")

' Points specified using expressions
m_spel.BGo("P0 /L /2")
m_spel.BGo("P1 :Z(-20)")

' Using parallel processing
m_spel.BGo("P1 \!D50; On 1; D90; Off 1\!")
```

```
' Points specified using labels
m_spel.BGo("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.BGo(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P*");
pt.X = 125.5;
m_spel.BGo(pt);

// Using an attribute expression
m_spel.BGo(pt, "Till");

// Points specified using expressions
m_spel.BGo("P0 /L /2");
m_spel.BGo("P1 :Z(-20)");

// Using a parallel processing
m_spel.BGo("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.BGo("pick");
```

15.3.2.3 BMove Method

Description

Executes linear interpolated relative motion in the selected local coordinate system

Syntax

Sub BMove (PointNumber As Integer)
Sub BMove (Point As SpelPoint)
Sub BMove (Point As SpelPoint, AttribExpr As String)
Sub BMove (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the BMove motion. This is the final position at the end of the linear interpolated motion.

- PointNumber
  Specifies the target point using a previously taught point number from the current robot point memory stored in the controller.
- Point
  Specifies the target point using the SpelPoint data type.
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain ROT, CP, SYNC, Till or Find search expressions, and parallel processing statements.
- PointExpr
  Specifies the target point using a string expression.
  It can contain ROT, CP, SYNC, Till or Find search expressions, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelR Method, **AccelS Method**, **SpeedR Method**, **SpeedS Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **TGo Method**, **TMove Method**, **Find Method**, **Till Method**, **SetPoint Method**

Move Example

VB Examples:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.BMove(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.BMove(pt)

' Points specified using expressions
m_spel.BMove("P0 /L /2 ROT")
m_spel.BMove("P1 :Z(-20)")

' Using parallel processing
m_spel.BMove("P1 \!D50; On 1; D90; Off 1\!")

' Points specified using labels
m_spel.BMove("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.BMove(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P*");
pt.X = 125.5;
m_spel.BMove(pt);

// Points specified using expressions
m_spel.BMove("P0 /L /2");

// Using a parallel processing
m_spel.BMove("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.BMove("pick");
```

15.3.2.4 Box Method

Description

Specifies an approach check area defined within a box.

Syntax

Sub Box (AreaNumber As Integer, MinX As Single, MaxX As Single, MinY As Single, MaxY As Single, MinZ As Single, MaxZ As Single)
Sub Box (AreaNumber As Integer, MinX As Single, MaxX As Single, MinY As Single, MaxY As Single, MinZ As Single, MaxZ As Single, PolarityOn As Boolean)

Parameters

- AreaNumber
  An integer from 1 to 15 indicating the number of the set area.
- MinX
  A number or formula that specifies the minimum X coordinate position (real number) of the set area.
- MaxX
  A number or formula that specifies the maximum X coordinate position (real number) of the set area.
- MinY
  A number or formula that specifies the minimum Y coordinate position (real number) of the set area.
- MaxY
  A number or formula that specifies the maximum Y coordinate position (real number) of the set area.
- MinZ
  A number or formula that specifies the minimum Z coordinate position (real number) of the set area.
- MaxZ
  A number or formula that specifies the maximum Z coordinate position (real number) of the set area.
- PolarityOn
  Optional. Sets the remote output logic when the corresponding remote output is used. To set I/O output to on when the end effector is in the box area, use True. To set I/O output to off when the end effector is in the box area, use False.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

BoxClr Method, **BoxDef Method**, **Plane Method**, **SetPoint Method**

Box Example

VB Example:

```
m_spel.Box(1, -5, 5, -10, 10, -20, 20)
```

C# Example:

```
m_spel.Box(1, -5, 5, -10, 10, -20, 20);
```

15.3.2.5 BoxClr Method

Description

Clears the definition of a box (approach check area).

Syntax

Sub BoxClr (BoxNumber As Integer)

Parameters

- BoxNumber
  An integer expression or number (1 to 15) indicating the approach check area number to clear.

See Also

Box Method, **BoxDef Method**

BoxClr Example

VB Example:

```
m_spel.BoxClr(1)
```

C# Example:

```
m_spel.BoxClr(1);
```

15.3.2.6 BoxDef Method

Description

Returns whether Box has been defined or not.

Syntax

Function BoxDef (BoxNumber As Integer) As Boolean

Parameters

- BoxNumber
  An integer expression or number (1 to 15) indicating the approach check area number to return the status for.

Return Value

True if the specified box is defined, False if not.

See Also

Box Method, **BoxClr Method**

BoxDef Example

VB Example:

```
x = m_spel.BoxDef(1)
```

C# Example:

```
x = m_spel.BoxDef(1);
```

15.3.2.7 Brake Method

Description

Reads or sets brake status for specified joint.

Syntax

Sub Brake (JointNumber As Integer, State As Boolean)
Function Brake (JointNumber As Integer) As Boolean

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)
- State
  To apply the brake: On To release the brake: Off

Return Value

0 = Brake Off
1 = Brake On

Remarks

This method is used to turn brakes on or off for one joint of the 6-axis robot (including N series). This method is intended for use by maintenance personnel only.

When this method is executed, the robot control parameter is initialized.

> ⚠ WARNING
>
> Use extreme caution when turning off a brake. Ensure that the joint is properly supported, otherwise the joint can fall and cause damage to the robot and personnel.

> ⚠ Note
>
> Before releasing the brake, be ready to use the emergency stop switch
>
> When the controller is in emergency stop status, the motor brakes are locked. Be aware that the robot arm may fall by its own weight when the brake is turned off with Brake command.

See Also

Reset Method, **SFree Method**, **Slock Method**

Brake Example

VB Example:

```
Dim state As Boolean
state = m_spel.Brake(1)
```

C# Example:

```
bool state;
state = m_spel.Brake(1);
```

15.3.2.8 BTst Method

Description

Returns the status of 1 bit in a number.

Syntax

Function BTst (Number As Integer, BitNumber As Integer) As Boolean

Parameters

- Number
  An expression or number that specifies the number to use for the bit test.
- BitNumber
  A number between 0 and 31 that specifies the bit number to test.

Return Value

True if the specified bit is set, False if not.

See Also

On Method, **Off Method**

BTst Example

VB Example:

```
x = m_spel.BTst(data, 2)
```

C# Example:

```
x = m_spel.BTst(data, 2);
```

15.3.2.9 .BuildProject Method

Description

Builds the Epson RC+ 8.0 project specified by the Project property.

Syntax

Sub BuildProject ()

See Also

Project Property, **ProjectBuildComplete Property**, **ProjectOverwriteWarningEnabled Property**

BuildProject Example

VB Example:

```
With m_spel
    .Project = "c:\EpsonRC80\projects\myproj\myproj.sprj"
    If Not .ProjectBuildComplete() Then
        .BuildProject()
    End If
End With
```

C# Example:

```
m_spel.Project = @"c:\EpsonRC80\projects\myproj\myproj.sprj";
if(\!m_spel.ProjectBuildComplete())
    m_spel.BuildProject();
```

15.3.3 C

15.3.3.1 Call Method

Description

Calls (executes) a SPEL+ function which can optionally return a value.

Syntax

Function Call (FuncName As String [, Parameters As String]) As Object

Parameters

- FuncName
  Function name to call
  Parameters Optional. Specify a list of arguments. Use arguments separated by commas (,).

Return Value

The return value of the SPEL+ function.
The data type matches the data type of the function.

Remarks

Use the Call method to call a SPEL+ function and retrieve the return value. When assigning the result of Call to a variable, ensure that the correct data type is used, otherwise a type mismatch error will occur.
You can also call DLL functions declared in your SPEL+ code from your Visual Basic application.

> ⚠ Note
>
> To stop the function executed by Call Method, use Stop Method.
> With Stop Method, the return value of Call Method becomes null.
> The function executed by Call Method cannot be paused by Pause or Halt Method.
> If you need to pause, use Xqt Method.

See Also

Xqt Method

Call Example

```
' Visual Basic Code
Dim errCode As Integer
errCode = m_spel.Call("GetPart", """Test"",2")

// C# Code
int errCode;
errCode = m_spel.Call("GetPart", """Test"",2");

' SPEL+ function
Function GetPart(Info$ As String, Timeout As Integer) As Integer
    Long errNum
OnErr GoTo GPErr
    Print Info$
    errNum = 0
    Jump P1
    On vacuum
    Wait Sw(vacOn) = On, Timeout
    If TW = True Then
        errNum = VAC_TIMEOUT
    EndIf
```

```
    GetPart = errNum
    Exit Function
GPErr:
    GetPart = Err
Fend
```

15.3.3.2 CalPls Method

Description

Reads or sets the position and orientation pulse values for calibration.

Syntax

Function CalPls (JointNumber As Integer) As Integer
Sub CalPls (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer, J4Pulses As Integer, [J5Pulses As Integer], [J6Pulses As Integer], [J7Pulses As Integer], [J8Pulses As Integer], [J9Pulses As Integer])

Parameters

- J1Pulses - J9Pulses
  Integers indicating the pulse values for Joints 1 to 9. J5Pulses to J9Pulses are optional

Return Value

When parameters are omitted, displays the current CalPls values.

Remarks

Specifies and maintains the correct position pulse value(s) for calibration.

CalPls is intended to be used for maintenance, such as after changing motors or when motor zero position needs to be matched to the corresponding arm mechanical zero position. This matching of motor zero position to corresponding arm mechanical zero position is called calibration. Normally, the calibration position Pulse values match the CalPls pulse values.

However, after performing maintenance operations such as changing motors, these two sets of values no longer match, and therefore calibration becomes necessary.

Calibration may be accomplished by moving the arm to a certain calibration position and then executing Calib. By executing Calib, the calibration position pulse value is changed to the CalPls value (the correct pulse value for the calibration position.)

Hofs values must be determined to execute calibration. To have Hofs values automatically calculated, move the arm to the desired calibration position, and execute Calib. The controller automatically.

> ⚠ Note
>
> CalPls Values Cannot be Changed by cycling power
>
> CalPls values are not initialized by turning main power to the controller off and then on again. The only method to modify the CalPls values is to execute the Calib command.

See Also

Hofs Method

CalPls Example

VB Example:

```
Dim val As Single
Dim i As Integer
For i = 1 To 4
val = m_spel.CalPls(i)
Next i
```

C# Example:

```
float val;
for(int i = 1; i <= 4; i++)
        val = m_spel.CalPls(i);
```

15.3.3.3 ClearPoints Method

Description

Clears the points in memory for the current robot.

Syntax

Sub ClearPoints ()

See Also

LoadPoints Method, **Robot Property**, **SavePoints Method**, **SetPoint Method**

ClearPoints Example

VB Example:

```
With m_spel
    .ClearPoints()
    .SetPoint(1, 100, 200, -20, 0, 0, 0)
    .Jump(1)
End With
```

C# Example:

```
m_spel.ClearPoints();
m_spel.SetPoint(1, 100, 200, -20, 0, 0, 0);
m_spel.Jump(1);
```

15.3.3.4 Connect Method

Description

Connects the Spel class instance with a Controller.

Syntax

Sub Connect (ConnectionName As String)
Sub Connect (ConnectionName As String, ConnectionPassword As String)
Sub Connect (ConnectionNumber As Integer)
Sub Connect (ConnectionNumber As Integer, ConnectionPassword As String)

Parameters

- ConnectionName
  A character string indicating the connection name.
- ConnectionNumber
  An integer indicating the connection number. The connection number is the number of the connection in the RC+ Setup \ [PC to Controller Communications] dialog. Specify a value of -1 to use the last successful connection.
- ConnectionPassword
  A character string indicating the connection password. If the controller requires a password for connection but the password has not been configured in the RC+ Setup \ [PC to Controller Communications] dialog, you must set the password to connect with the controller.

Remarks

When a Spel class instance needs to communicate with the Controller, it automatically connects. If you want to explicitly connect to the Controller, use the Connect method.

See Also

Disconnect Method, **Initialize Method**

> ⚠ CAUTION
>
> When you change the connection number in [PC to Controller Communications] dialog, the parameter value of this method "ConnectionNumber" will be affected. Review the RC+ API program.

Connect Example

VB Example:

```
  Try
    m_spel.Tool(1)
  Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
  End Try
```

C# Example:

```
try{
    m_spel.Tool(1);
}
catch(RCAPINet.SpelException ex){
    MessageBox.Show(ex.Message);
}
```

15.3.3.5 Continue Method

Description

Causes all tasks in the Controller to resume if a pause has occurred.

Syntax

Sub Continue ()

Remarks

Use Continue to resume all tasks that have been paused by the Pause method or by safeguard open.
When the safeguard is open while tasks are running, the robot will decelerate to a stop and the robot motors will be turned off.
After the safeguard has been closed, you can use Continue to resume the cycle.

See Also

Pause Method, **Start Method**, **Stop Method**

Continue Example

VB Example:

```
Sub btnContinue_Click( _
     ByVal sender As System.Object, _
     ByVal e As System.EventArgs) Handles btnContinue.Click

    btnPause.Enabled = True
    btnContinue.Enabled = False
  Try
    m_spel.Continue()
  Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
  End Try
End Sub
```

C# Example:

```
void btnContinue_Click(object sender, EventArgs e)
{
btnPause.Enabled = true;
   btnContinue.Enabled = true;

    try{
                m_spel.Continue();
               }
               catch(RCAPINet.SpelException ex){
                MessageBox.Show(ex.Message);
               }
}
```

15.3.3.6 Clr Method

Description

Returns the counter value of the specified input counter.

Syntax

Function Ctr (BitNumber As Integer) As Integer

Parameters

- BitNumber
  Number of input bits configured as counters. Only 16 counters can be active at the same time.

Return Value

Returns the counter value.(integer from 0 to 65535)

See Also

CtReset Method

Ctr Example

VB Example:

```
lblCounter.Text = m_spel.Ctr(1).ToString()
```

C# Example:

```
lblCounter.Text = m_spel.Ctr(1).ToString();
```

15.3.3.7 CtReset Method

Description

Resets the counter value of the specified input counter. Also defines the input as a counter Input.

Syntax

Sub CtReset (BitNumber As Integer)

Parameters

- BitNumber
  Number of input bits configured as counters. Only 16 counters can be active at the same time.

See Also

Clr Method

CtReset Example

VB Example:

```
m_spel.CtReset(2)
```

C# Example:

```
m_spel.CtReset(2);
```

15.3.3.8 Curve Method

Description

Defines the data and points required to move the arm along a curved path. Many data points can be defined in the path to improve precision of the path.

Syntax

Sub Curve (FileName As String, Closure As Boolean, Mode As Integer, NumOfAxis As Integer, PointList As String)

Parameters

- FileName
  A character string indicating the path and file name where the point data is stored. The extension .crv is appended to the specified file. When the Curve instruction is executed, fileName will be created.
- Closure
  A Boolean expression that specifies whether the end point of the path is connected to the start point.
- Mode
  Specifies whether or not the arm is automatically interpolated in the U-axis tangential direction.

Mode Setting	Tangental Correction
0	No
2	Yes

- NumOfAxes
  Integer expression between 2 and 4 which specifies the axis numbers controlled during the curved motion as follows: 2: Generate a curve in the XY plane with no U-Axis rotation. 3: Generate a curve in the XYZ plane with no U-Axis rotation. (Theta 1, Theta2, and Z) 4: Generate a curve in the XYZ plane with U-Axis rotation. (Controls all 4 Axes)
- pointList
  {point expression | P(start:end) } [, output command] ... This parameter is a series of point data separated by commas. If the point data is complete and listed in ascending or descending order, two point numbers can be combined using a colon and specified like P(1:5). Normally the series of points are separated by commas as shown below: Curve MyFile, O, 0, 4, P1, P2, P3, P4 Or use a colon to specify as shown below: Curve MyFile, O, 0, 4, P(1:4) In the case shown above the user defined a curve using points P1, P2, P3, and P4. output command is optional and is used to control output operation during curve motion. The command can be On or Off for digital outputs or memory outputs. Entering an output command following any point number in the point series causes execution of the output command when the arm reaches the point just before the output command. A maximum of 16 output commands may be included in one Curve statement. In the example below, the "On 2" command is executed just as the arm reaches the point P2, then the arm continues to all points between and including P3 and P10. Curve "MyFile", C, 0, 4, P1, P2, ON 2, P(3:10)

Remarks

Use Curve to define a spline path to be executed with the CVMove method. See the SPEL+ command Curve for more details.

See Also

Curve (SPEL+ Statement), **CVMove Method**

Curve Example

VB Example:

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1, P(4:7)")
m_spel.CVMove("mycurveFile")
```

C# Example:

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1, P(4:7)");
m_spel.CVMove("mycurveFile");
```

15.3.3.9 CVMove Method

Description

Performs the continuous spline path motion defined by the Curve instruction.

Syntax

Sub CVMove (FileName As String [, OptionList As String])

Parameters

- FileName
  A string expression indicating the file name created by the Curve instruction.
- OptionList
  Optional. A character string indicating the Till specification.

Remarks

Use CVMove to execute a path defined with the Curve method. See the SPEL+ command CVMove for more details. If you need to execute CVMove with CP, it is recommended that you execute CVMove from a SPEL+ task rather than from your application. The reason for this is that for CP motion to perform properly, the system needs to know ahead of time where the next motion target is. Since the RC+ API commands are executed one at a time, the system does not know ahead of time where the next target is.

See Also

Curve Method, CVMove(SPEL+)

CVMove Example

VB Example:

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1, P(4:7)")
m_spel.CVMove("mycurveFile", "CP Till Sw(1) = 1")
m_spel.CVMove("mycurveFile")
```

C# Example:

```
m_spel.Curve("mycurveFile", True, 0, 4, "P(1:3), On 1, P(4:7)");
m_spel.CVMove("mycurveFile", "CP Till Sw(1) = 1");
m_spel.CVMove("mycurveFile");
```

15.3.3.10 CX, CY, CZ, CU, CV, CW, CR, CS, CT Method

Description

Retrieves a coordinate value from a point \ - CV and CW are for the 6-axis robot- CS and CT are for the additional axis - CR is for the Joint 7-axis robot

Syntax

Function CX (PointExpr As String) As Single
Function CY (PointExpr As String) As Single
Function CZ (PointExpr As String) As Single
Function CU (PointExpr As String) As Single
Function CV (PointExpr As String) As Single
Function CW (PointExpr As String) As Single
Function CR (PointExpr As String) As Single
Function CS (PointExpr As String) As Single
Function CT (PointExpr As String) As Single

Parameters

- PointExpr
  A string expression specifying the point from which to retrieve the specified coordinate. Any valid point expression can be used, as long as it starts with P0. P* can also be used to retrieve the coordinate from the current position.

Return Value

The specified coordinate value.
Return value of CX, CY, CZ : Real value (mm)
Return value of CU, CV, CW : Real value (deg)
Return value of CR, CS, CT : Real value

See Also

GetPoint Method, **SetPoint Method**

CX, CY, CZ, CU, CV, CW, CR, CS, CT Example

VB Example:

```
Dim x As Single, y As Single
x = m_spel.CX("P1")
y = m_spel.CY("P*")
```

C# Example:

```
float x, y;
x = m_spel.CX("P1");
y = m_spel.CY("P*");
```

15.3.4 D

15.3.4.1 Delay Method

Description

Delays for a specified number of milliseconds.

Syntax

Sub Delay (Milliseconds As Integer)

Parameters

- Milliseconds
  The number of milliseconds to delay.

Delay Example

VB Example:

```
m_spel.Delay(500)
```

C# Example:

```
m_spel.Delay(500);
```

15.3.4.2 DegToRad Method

Description

Converts Degrees into Radians.

Syntax

Function DegToRad (degrees As Double) As Double

Parameters

- degrees
  A real number indicating degrees to convert to radians.

Return Value

A double value containing radians.

See Also

RadToDeg Method

DegToRad Example

VB Example:

```
Dim rad As Double

rad = m_spel.DegToRad(45)
```

C# Example:

```
double rad;

rad = m_spel.DegToRad(45);
```

15.3.4.3 Disconnect Method

Description

Disconnects the Spel class instance from the current connection.

Syntax

Sub Disconnect ()

Remarks

Use Disconnect to disconnect from the current Controller connection.

See Also

Connect Method, **Initialize Method**

Disconnect Example

VB Example:

```
  Try
    m_spel.Disconnect()
  Catch ex As RCAPINet.SpelException
    MsgBox(ex.Message)
  End Try
```

C# Example:

```
try{
    m_spel.Disconnect();
}
catch(RCAPINet.SpelException ex){
MessageBox.Show(ex.Message);}
```

15.3.5 E

15.3.5.1 ECP Method

Description

Selects the current ECP definition.

Syntax

Sub ECP (ECPNumber As Integer)

Parameters

- ECPNumber
  Optional. An integer from 0 to 15 indicating which of the 16 ECP definitions to use with the next motion instruction. ECP 0 disables the ECP selection

See Also

ECPSet Method

ECP Example

VB Example:

```
m_spel.ECP(1)
m_spel.Move("P1 ECP")
```

C# Example:

```
m_spel.ECP(1);
m_spel.Move("P1 ECP");
```

15.3.5.2 ECPClr Method

Description

Clears (undefines) an external control point for the current robot.

Syntax

Sub ECP (ECPNumber As Integer)

Parameters

- ECPNumber
  An integer indicating the ECP number to clear (ECP 0 is the default. It cannot be cleared.)

See Also

ECP Method, **ECPDef Method**

ECP Example

VB Example:

```
m_spel.ECPClr(1)
```

C# Example:

```
m_spel.ECPClr(1);
```

15.3.5.3 ECPDef Method

Description

Returns ECP definition status.

Syntax

Function ECPDef (ECPNumber As Integer) As Boolean

Parameters

- ECPNumber
  An integer indicating the ECP number of the status you want to call.

Return Value

True if the specified ECP is defined, False if not.

See Also

ECP Method, **ECPClr Method**

ECP Example

VB Example:

```
x = m_spel.ECPDef(1)
```

C# Example:

```
x = m_spel.ECPDef(1);
```

15.3.5.4 ECPSet Method

Description

Defines an ECP (external control point).

Syntax

Sub ECPSet (ECPNumber As Integer, Point As SpelPoint)
Sub ECPSet (ECPNumber As Integer, XCoord as Double, YCoord as Double, ZCoord as Double, UCoord as Double [, VCoord As Double] [, WCoord as Double)]

Parameters

- ECPNumber
  An expression or integer expression from 1 to 15 indicating which of the external control points to define.
- Point
  A SpelPoint indicating the point data.
- XCoord
  The external control point X coordinate.
- YCoord
  The external control point Y coordinate.
- ZCoord
  The external control point Z coordinate.
- UCoord
  The external control point U coordinate.
- VCoord
  Optional. The external control point V coordinate.
- WCoord
  Optional. The external control point W coordinate.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

ArmSet Method, **ECP Method**, **GetECP Method**, **TLSet Method**, **SetPoint Method**

ECPSet Example

VB Example:

```
m_spel.ECPSet(1, 100.5, 99.3, 0, 0)
```

C# Example:

```
m_spel.ECPSet(1, 100.5, 99.3, 0, 0);
```

15.3.5.5 EnableEvent Method

Description

Enables certain system events for the EventReceived event.

Syntax

Sub EnableEvent (EventNumber As RCAPINet.SpelEvents, Enabled as Boolean)

Parameters

- EventNumber
  Event number to enable or disable.
- Enabled
  True to enable the event or False to disable it.

See Also

EventReceived Event

EnableEvent Example

VB Example:

```
With m_spel
  .EnableEvent(RCAPINet.SpelEvents.ProjectBuildStatus, True)
  .BuildProject()
End With
```

C# Example:

```
m_spel.EnableEvent(RCAPINet.SpelEvents.ProjectBuildStatus, true);
m_spel.BuildProject();
```

15.3.5.6 ExecuteCommand Method

Description

Sends a command to Epson RC+ 8.0 and waits for it to complete

Syntax

Sub ExecuteCommand (Command As String , [ByRef Reply As String])

Parameters

- Command
  A character string equivalent to the SPEL+command
- Reply
  Returns the required response.

Remarks

Normally, ExecuteCommand is not required. Most operations can be performed by executing Spel methods. However, sometimes it is desirable to execute SPEL+ multi-statements. Multi-statements are one line commands that contain more than one statement separated by semi-colons. Use ExecuteCommand to execute multi-statements.

For example: m_spel.ExecuteCommand("JUMP pick; ON tipvac")
The maximum command line length is 200 characters.

See Also

Pause Method

ExecuteCommand Example

VB Example:

```
m_spel.ExecuteCommand("JUMP P1\!D50; ON 1\!")
```

C# Example:

```
m_spel.ExecuteCommand("JUMP P1\!D50; ON 1\!");
```

15.3.6 F

15.3.6.1 FBusIO_GetBusStatus Method

Description

Returns the status of the specified Fieldbus.

Syntax

Function FBusIO_GetBusStatus (BusNumber As Integer) As Integer

Parameters

- BusNumber
  An integer indicating the Fieldbus system number. This number must be 16, which is the ID of the bus connected to the Fieldbus master board on the PC side of the controller.

Return Value

0 - OK
1 - Disconnected
2 - Power off

Remarks

Note
This method will only work if the Fieldbus Master option is active.

See Also

FBusIO_GetDeviceStatus Method, **FBusIO_SendMsg Method**, **IsOptionActive Method**

FBusIO_GetBusStatus Use Example

VB Examples:

```
Dim val As Integer
val = m_spel.FBusIO_GetBusStatus(16)
```

C# Example:

```
int busStatus;
busStatus = m_spel.FbusIO_GetBusStatus(16);
```

15.3.6.2 FBusIO_GetDeviceStatus Method

Description

Returns the status of the specified Fieldbus device.

Syntax

Function FBusIO_GetDeviceStatus (BusNumber As Integer, DeviceID As Integer) As Integer

Parameters

- BusNumber
  An integer indicating the Fieldbus system number. This number must be 16, which is the ID of the bus connected to the Fieldbus master board on the PC side of the controller.
- DeviceID
  An integer indicating the Fieldbus ID of the device.

Return Value

0 - OK
1 - Disconnected
2 - Power off
3 - Synchronization error. Device is booting, or has incorrect baud rate.

Remarks

Note
This method will only work if the Fieldbus Master option is active.

See Also

FBusIO_GetBusStatus Method, **FBusIO_SendMsg Method**, **IsOptionActive Method**

FBusIO_GetDeviceStatus Use Example

VB Examples:

```
Dim val As Integer
val = m_spel.FBusIO_GetDeviceStatus(16, 10)
```

C# Example:

```
dev
ideStatus = m_spel.FbusIO_GetDeviceStatus(16, 10);
```

15.3.6.3 FBusIO_SendMsg Method

Description

Sends an explicit message to a Fieldbus device and returns the reply.

Syntax

Sub FBusIO_SendMsg (BusNumber As Integer, DeviceID As Integer, MsgParam As Integer, SendData As Byte(), ByRef RecvData As Byte())

Parameters

- BusNumber
  An integer indicating the Fieldbus system number. This number must be 16, which is the ID of the bus connected to the Fieldbus master board on the PC side of the controller.
- DeviceID
  An integer indicating the Fieldbus ID of the device.
- MsgParam
  An integer expression indicating the message parameter. This is not available for DeviceNet.
- SendData
  Specifies the data to send to the device as a byte array. This array must be dimensioned to the number of bytes to send. If there are no bytes to send, specify 0.
- RecvData
  Specifies the data to receive from the device as a byte array. This array will automatically be redimensioned to the number of bytes received.

Remarks

Note
This method will only work if the Fieldbus Master option is active.

See Also

FBusIO_GetBusStatus Method, **FBusIO_GetDeviceStatus Method**, **IsOptionActive Method**

FBusIO_SendMsg Use Example

VB Examples:

```
'Send explicit message to DeviceNet device
Dim recvData() as Byte
Dim sendData(6) as Byte
Array.Clear(sendData, 0, sendData.Length)
sendData(0) = 14 'Command
sendData(1) = 1 'Class
sendData(3) = 1 'Instance
sendData(5) = 7 'Attribute
' MsgParam is 0 for DeviceNet
m_spel.FbusIO_SendMsg(16, 1, 0, sendData, recvData)

' Send explicit message to Profibus device
Dim recvData() As Byte;
m_spel.FbusIO_SendMsg(16, 1, 56, Nothing, recvData);
```

C# Example:

```
// Send explicit message to DeviceNet device
byte[] sendData, recvData;
byte[] sendData = new byte[6];
Array.Clear(sendData, 0, sendData.Length);
sendData[0] = 14; //Command
```

```
sendData[1] = 1; //Class
sendData[3] = 1; //Instance
sendData[5] = 7; //Attribute
//MsgParam is 0 for DeviceNet
m_spel.FbusIO_SendMsg(16, 201, 0, sendData, out recvData);

// Send explicit message to Profibus device
byte[] recvData;
m_spel.FbusIO_SendMsg(16, 1, 56, null, out recvData);
```

15.3.6.4 FGGet Method

Description

Acquires a result of a Force Guide sequence or Force Guide object.

Syntax

Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As Boolean)
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As Double)
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As Integer)s
Sub FGGet (Sequence As String, [Object As String], Property As SpelForceProps, ByRef Result As String)

Parameters

- Sequence
  Force guide sequence name or string variable indicating the Force Guide sequence name
- Object
  Force guide object name or string variable indicating the Force Guide object name. Omitted when acquiring the result of a Force Guide sequence.
- Property
  Result name to acquire the value
- Result
  Variable indicating the returned value. The number and type vary according to the result.

See Also

FGRun Method

FGGet Example

VB Example:

```
Dim val As Integer
m_spel.MotorsOn = True

m_spel.FGRun("Sequence1")
m_spel.FGGet("Sequence1","", SpelForceProps.EndStatus, val)
```

C# Example:

```
int errCode;
m_spel.MotorsOn = true;

m_spel.FGRun("Sequence1");
m_spel.FGGet("Sequence1","", SpelForceProps.EndStatus, val);
```

15.3.6.5 FGRun Method

Description
Executes a Force Guide sequence.

Syntax
Sub FGRun (Sequence As String)

Parameters
- Sequence
  Force guide sequence name or string variable indicating the Force Guide sequence name

Remarks
Executes a specified Force Guide sequence. The Force Guide sequence starts from the position where the FGRun statement was executed. Execute after moving to the assumed start position by the Go statement, Move statement, or other motion commands.

When the specified Force Guide sequence ends, the program proceeds to the next statement.
To acquire the results of sequences executed by FGRun, use FGGet.
When path motion is enabled by the CP parameter or CP statement, the program waits until the robot stops and then executes a Force Guide sequence.

When any of the following conditions is fulfilled at the time of execution start, an error occurs.
The robot specified in the program differs from the robot specified by the RobotNumber property. Specify the correct robot by the Robot statement.
The robot type specified in the program differs from the robot type specified by the RobotType property. Specify the correct robot by the Robot statement.
The tool number specified in the program differs from the tool number specified by the RobotTool property. Specify the correct tool number by the Tool statement.
Motor is in OFF state. Switch to ON state by the Motor statement.
Force control function is currently being executed. Stop force control by the FCEnd statement.
Conveyor tracking is currently being executed. Stop conveyor tracking by the Cnv_AbortTrack statement.
Currently in the torque control mode. Disable the torque control mode by the TC statement.

FGRun, when executed, automatically overwrites the following properties; therefore, it cannot be used together with the following properties:
FM object
AvgForceClear property
PeakForceClear property

This method cannot be executed while the program is running.

See Also
FGGet Method

FGRun Example
VB Example:

```
Dim errCode As Integer
m_spel.MotorsOn = True

m_spel.FGRun("Sequence1")
errCode = m_spel.FGGet("Sequence1", SpelForceProps.EndStatus, val)
```

C# Example:

```
int errCode;
m_spel.MotorsOn = true;

m_spel.FGRun("Sequence1");
errCode = m_spel.FGGet("Sequence1", SpelForceProps.EndStatus, val);
```

15.3.6.6 Find Method

Description

Sets the condition to save the coordinate during motion instructions.

Syntax

Sub Find (Condition As String)

Parameters

- Condition
  Specifies the input state to use as a trigger.

See Also

Go Method, **Jump Method**

Find Example

VB Example:

```
m_spel.Find("Sw(5) = On")
```

C# Example:

```
m_spel.Find("Sw(5) = On");
```

15.3.6.7 Fine Method

Description

Specifies and displays the positioning accuracy for target points.

Syntax

Sub Fine (J1MaxErr As Integer, J2MaxErr As Integer, J3MaxErr As Integer, J4MaxErr As Integer , J5MaxErr As Integer, J6MaxErr As Integer [, J7MaxErr As Integer][, J8MaxErr As Integer] [, J9MaxErr As Integer])

Parameters

- J1MaxErr - J9MaxErr
  Integer number ranging from (0-32767) which represents the allowable positioning error for the each joint. The values for joints 7, 8, and 9 are optional.

See Also

Weight Method

Fine Example

VB Example:

```
m_spel.Fine(1000, 1000, 1000, 1000, 0, 0)
```

C# Example:

```
m_spel.Fine(1000, 1000, 1000, 1000, 0, 0);
```

15.3.7 G

15.3.7.1 GetAccel Method

Description

Returns specified acceleration/deceleration value.

Syntax

Function GetAccel (ParamNumber As Integer) As Integer

Parameters

- ParamNumber
  An integer expression which can have the following values: 1: acceleration setting 2: deceleration setting 3: depart acceleration setting for Jump 4: depart deceleration setting for Jump 5: approach acceleration setting for Jump 6: approach deceleration setting for Jump

Return Value

Integer containing the specified acceleration/deceleration value.

See Also

Accel Method

GetAccel Example

VB Example:

```
Dim x As Integer
x = m_spel.GetAccel(1)
```

C# Example:

```
int x;
x = m_spel.GetAccel(1);
```

15.3.7.2 GetArm Method

Description

Returns the current Arm number for the current robot.

Syntax

Function GetArm () As Integer

Return Value

Integer containing the current arm number.

See Also

Arm Method, **ArmSet Method**, **Robot Property**, **Tool Method**

GetArm Example

VB Example:

```
saveArm = m_spel.GetArm()
m_spel.Arm(2)
```

C# Example:

```
m_spel.Arm(2);
```

15.3.7.3 GetConnectionInfo Method

Description
Returns information about the Controller connections.

Syntax
Function GetConnectionInfo() As SpelConnectionInfo()

Return Value
An array of SpelConnectionInfo.

See Also
GetControllerInfo Method

Return Value
GetConnectionInfo returns an array of SpelConnectionInfo. The connection information is configured in Epson RC+ from the [Setup]-[PC to Controller Communication] dialog.

GetConnectionInfo Example
VB Example:

```
Dim info() As SpelConnectionInfo

info = m_spel.GetConnectionInfo()
```

C# Example:

```
SpelConnectionInfo[] info = m_spel.GetConnectionInfo();
```

15.3.7.4 GetControllerInfo Method

Description

Returns information about the current Controller.

Syntax

Function GetControllerInfo() As SpelControllerInfo

Return Value

A SpelControllerInfo instance.

See Also

GetErrorMessage Method, **GetRobotInfo Method**, **GetTaskInfo Method**

Remarks

GetControllerInfo returns a new instance of the SpelControllerInfo class, which contains Controller information properties.

GetControllerInfo Example

VB Example:

```
Dim info As SpelControllerInfo
Dim msg As String

info = m_spel.GetControllerInfo()
msg = "Project Name: " & info.ProjectName & vbCrLf _
    & "Project ID: " & info.ProjectID
MsgBox(msg)
```

C# Example:

```
SpelControllerInfo info;
string msg;

info = m_spel.GetControllerInfo();
msg = "Project Name:" + info.ProjectName + "\r\n ProjectID :" +
"info.ProjectID";
MessageBox.Show(msg);
```

15.3.7.5 GetCurrentConnectionInfo Method

Description

Returns the information of the current controller connection.

Syntax

Function GetCurrentConnectionInfo () As SpelConnectionInfo

Return Value

SpelConnectionInfo

See Also

GetConnectionInfo Method, **GetControllerInfo Method**

GetCurrentConnectionInfo Example

VB Example:

```
Dim info As SpelConnectionInfo

info = m_spel.GetCurrentConnectionInfo()
```

C# Example:

```
SpelConnectionInfo info;

info = m_spel.GetCurrentConnectionInfo();
```

15.3.7.6 GetCurrentUser Method

Description

Returns the current Epson RC+ 8.0 user.

Syntax

Function GetCurrentUser () As String

Return Value

String variable containing the current user.

See Also

Login Method

GetCurrentUser Example

VB Example:

```
Dim currentUser As String

currentUser = m_spel.GetCurrentUser()
```

C# Example:

```
string currentUser;

currentUser = m_spel.GetCurrentUser();
```

15.3.7.7 GetECP Method

Description

Returns the current ECP number for the current robot.

Syntax

Function GetECP () As Integer

Return Value

Integer containing the current ECP number.

See Also

ECP Method, **ECPSet Method**

GetECP Example

VB Example:

```
saveECP = m_spel.GetECP()
m_spel.ECP(2)
```

C# Example:

```
saveECP = m_spel.GetECP();
m_spel.ECP(2);
```

15.3.7.8 GetErrorMessage Method

Description

Returns the error message for the specified error or warning code.

Syntax

Function GetErrorMessage (ErrorCode As Integer) As String

Parameters

- ErrorCode
  Error code that returns an error message

Return Value

String containing the error message.

See Also

ErrorCode Property

GetErrorMessage Example

VB Example:

```
Dim msg As String

If m_spel.ErrorOn Then
  msg = m_spel.GetErrorMessage(m_spel.ErrorCode)
  MsgBox(msg)
End If
```

C# Example:

```
string msg;

if (m_spel.ErrorOn) {
        msg = m_spel.GetErrorMessage(m_spel.ErrorCode);
        MessageBox.Show(msg);
}
```

15.3.7.9 GetIODef Method

Description

Gets the definition information for an input, output, or memory I/O bit, byte, or word.

Syntax

Sub GetIODef(Type As SpelIOLabelTypes, Index As Integer, ByRef Label as String, ByRef Description As String)

Parameters

- Type
  Specifies the I/O type as shown below: InputBit = 1 InputByte = 2 InputWord = 3 OutputBit = 4 OutputByte = 5 OutputWord = 6 MemoryBit = 7 MemoryByte = 8 MemoryWord = 9 InputReal = 10 OutputReal = 11
- Index
  Specifies the bit or port number.
- Label
  Returns the label.
- Description
  Returns the description.

Return Value

The values are returned in the Label and Description parameters.

Remarks

Use GetIODef to get the labels and descriptions used for all I/O in the current project.

See Also

SetIODef Method

GetIODef Example

VB Example:

```
Dim label As String
Dim desc As String
m_spel.GetIODef(SpelIOLabelTypes.InputBit, 0, label, desc)
```

C# Example:

```
string label, desc;

m_spel.GetIODef(SpelIOLabelTypes.InputBit, 0, out label, out desc);
```

15.3.7.10 GetJRange Method

Description

Gets the permissible working range of the specified joint in pulses.

Syntax

Function GetJRange (JointNumber As Integer, Bound As Integer) As Integer

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)
- Bound
  Specifies one of the following two values as an integer. 1: Specifies lower limit value.2: Specifies upper limit value.

Return Value

Range configuration (integer value, pulses) of the specified joint.

See Also

Jrange Method

GetJRange Example

VB Example:

```
Dim minRange As Integer
Dim maxRange As Integer
val1 = m_spel.GetJRange(1, 1)
val2 = m_spel.GetJRange(1, 2)
```

C# Example:

```
int minRange, maxRange;
minRange = m_spel.GetJRange(1, 1);
maxRange = m_spel.GetJRange(1, 2);
```

15.3.7.11 GetLimitTorque Method

Description

Returns the limit torque for the specified joint for the current robot.

Syntax

Function GetLimitTorque (JointNumber As Integer) As Integer

Parameters

- JointNumber
  An integer indicating the axis number.

Return Value

Integer value between 1 and 9 which represents the limit torque setting for the specified joint.

See Also

GetRealTorque Method, **GetRobotPos Method**, **LimitTorque Method**

GetLimitTorque Example

VB Example:

```
Dim j1LimitTorque As Integer
j1LimitTorque = m_spel.GetLimitTorque(1)
```

C# Example:

```
int j1LimitTorque;
j1LimitTorque = m_spel.GetLimitTorque(1);
```

15.3.7.12 GetLimZ Method

Description

Returns the current LimZ setting.

Syntax

Function GetLimZ () As Single

Return Value

Real value containing the LimZ value.

See Also

LimZ Method, **Jump Method**

GetLimZ Example

VB Example:

```
saveLimZ = m_spel.GetLimZ()
m_spel.LimZ(-22)
```

C# Example:

```
saveLimZ = m_spel.GetLimZ();
m_spel.LimZ(-22);
```

15.3.7.13 GetPoint Method

Description

Retrieves coordinate data for a robot point.

Syntax

Function GetPoint (PointNumber As Integer) As SpelPoint
Function GetPoint (PointName As String) As SpelPoint

Parameters

- PointNumber
  Integer expression indicating a point in the Controller's point memory for the current robot.
- PointName
  Point labels containing modifier characters "Pxxx", "P*" or "*" can be used in the character string.

Return Value

SpelPoint value

See Also

SetPoint Method

GetPoint Example

VB Example:

```
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 25.0
m_spel.Go(pt)
```

C# Example:

```
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 25.0;
m_spel.Go(pt);
```

15.3.7.14 GetRealTorque Method

Description

Returns the torque for the specified joint for the current robot.

Syntax

Function GetRealTorque (JointNumber As Integer) As Double

Parameters

- JointNumber
  An integer indicating the joint number for which you want to acquire the torque value.

Return Value

Double value between 0 and 1 which represents the portion of maximum torque for the current power mode and for the specified joint.

See Also

GetLimitTorque Method, **GetRobotPos Method**

GetRealTorque Example

VB Example:

```
Dim j1Torque As Double
j1Torque = m_spel.GetRealTorque(1)
```

C# Example:

```
double j1Torque;
j1Torque = m_spel.GetRealTorque(1);
```

15.3.7.15 GetRobotInfo Method

Description

Returns robot information.

Syntax

Function GetRobotInfo (RobotNumber As Integer) As SpelRobotInfo

Parameters

- RobotNumber
  Specifies the robot number.

Return Value

SpelRobotInfo

See Also

GetControllerInfo Method, **GetTaskInfo Method**

GetRobotInfo Example

VB Example:

```
Dim info As SpelRobotInfo
Dim msg As String

info = m_spel.GetRobotInfo(1)
msg = "Robot Model: " & info.RobotModel & vbCrLf _
    & "Robot Serial: " & info.RobotSerial
MsgBox(msg)
```

C# Example:

```
SpelRobotInfo info;
string msg;

info = m_spel.GetRobotInfo(1);
msg = "Robot Model: " + info.RobotModel +
      "\r\n Robot Serial: " + info.RobotSerial;
MessageBox.Show(msg);
```

15.3.7.16 GetRobotPos Method

Description

Returns the current robot position.

Syntax

Function GetRobotPos(PosType As SpelRobotPosType, Arm As Integer, Tool As Integer, Local As Integer) As Single()

Parameters

- PosType
  Specifies the position data type.
- Arm
  Integer expression that specifies the robot arm.
- Tool
  Integer expression that specifies the robot tool.
- Local
  Integer expression that specifies local.

Return Value

Single data type array containing 9 elements. The data returned depends on the specified PosType.
World X, Y, Z, U, V, W, R, S, T
Joint J1, J2, J3, J4, J5, J6, J7, J8, J9
Pulse Pls1, Pls2, Pls3, Pls4, Pls5, Pls6, Pls7, Pls8, Pls9

See Also

GetPoint Method

GetRobotPos Example

VB Example:

```
Dim values() As Single
values = m_spel.GetRobotPos(SpelRobotPosType.World, 0, 0, 0)
```

C# Example:

```
float[] values;
values = m_spel.GetRobotPos(SpelRobotPosType.World, 0, 0, 0);
```

15.3.7.17 GetSpeed Method

Description

Returns one of the three speed settings for the current robot.

Syntax

Function GetSpeed (ParamNumber As Integer) As Integer

Parameters

- ParamNumber
  Integer expression which determines one of the following values: 1: PTP motion speed 2: Jump depart speed 3: Jump approach speed

Return Value

Integer expression from 1-100.

See Also

Speed Method

GetSpeed Example

VB Example:

```
Dim ptpSpeed As Integer
ptpSpeed = m_spel.GetSpeed(1)
```

C# Example:

```
ptp
Speed = m_spel.GetSpeed(1);
```

15.3.7.18 GetTaskInfo Method

Description

Returns task information.

Syntax

Function GetTaskInfo (TaskName As String) As SpelTaskInfo
Function GetTaskInfo (TaskNumber As Integer) As SpelTaskInfo

Parameters

- TaskName
  Specifies the task name.
- TaskNumber
  Specifies the task number.

Return Value

SpelTaskInfo

See Also

GetControllerInfo Method, **GetRobotInfo Method**

GetTaskInfo Example

VB Example:

```
Dim info As SpelTaskInfo
Dim msg As String

info = m_spel.GetTaskInfo(1)
msg = "Task Name: " & info.TaskName & vbCrLf _
    & "Task State: " & info.State
MsgBox(msg)
```

C# Example:

```
SpelTaskInfo info;
string msg;

info = m_spel.GetTaskInfo(1);
msg= "Task Name: " + info.TaskName +
     "\r\n Task State: " + info.State;
MessageBox.Show(msg);
```

15.3.7.19 GetTool Method

Description

Returns the current Tool number for the current robot.

Syntax

Function GetTool () As Integer

Return Value

Integer containing the current tool number.

See Also

Arm Method, **TLSet Method**, **Tool Method**

GetTool Example

VB Example:

```
saveTool = m_spel.GetTool()
m_spel.Tool(2)
```

C# Example:

```
saveTool = m_spel.GetTool();
m_spel.Tool(2);
```

15.3.7.20 GetVar Method

Description

Returns the value of a SPEL+ global preserve variable in the Controller.

Syntax

Function GetVar(VarName As String) As Object

Parameters

- VarName
  The name of the SPEL+ global preserve variable. For an array, the entire array can be returned or just one element.

Return Value

You can use GetVar to retrieve values of any global preserve variables in the Controller's current project. Before you can retrieve values, the project must be successfully built.
If you want to retrieve an entire array, then supply the array name in VarName. To retrieve one element of an array, supply the subscript in VarName.

Remarks

You can use GetVar to retrieve values of any global preserve variables in the Controller's current project. Before you can retrieve values, the project must be successfully built.

See Also

SetVar Method

GetVar Example

In the SPEL+ project, the variable is declared:

```
Global Preserve Integer g_myIntVar
Global Preserve Real g_myRealArray(10)
Global Preserve String g_myStringVar$
Function main
   ...
Fend
```

In the Visual Basic project:
Since g_myIntVar is declared as in integer, the Visual Basic variable used to retrieve the value of g_myInVar must be declared as an Integer. For g_myRealArray, the Visual Basic variable must be declared as a Single array.

```
Dim myIntVar As Integer
Dim myRealArray() As Single
Dim myStringVar As String

myIntVar = m_spel.GetVar("g_myIntVar")
myRealArray = m_spel.GetVar("g_myRealArray")
myStringVar = m_spel.GetVar("g_myStringVar$")
```

In the C# project:
Since g_myIntVar is declared as in integer, the C# variable used to retrieve the value of g_myInVar must be declared as an int. For g_myRealArray, the C# variable must be declared as a float array.

```
int myIntVar;
float[] myRealArray;
string myStringVar;

myIntVar = m_spel.GetVar("g_myIntVar");
myRealArray = m_spel.GetVar("g_myRealArray");
myStringVar = m_spel.GetVar("g_myStringVar$");
```

15.3.7.21 Go Method

Description

Moves the arm in a Point to Point fashion from the current position to the specified point or XY position. The GO instruction can move any combination of the robot axes at the same time.

Syntax

Sub Go (PointNumber As Integer)
Sub Go (Point As SpelPoint)
Sub Go (Point As SpelPoint, AttribExpr As String)
Sub Go (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the Go motion. This is the final position at the end of the point to point motion.

- PointNumber
  Specifies the end point using the point number of a previously taught point in the Controller's point memory for the current robot.
- Point
  Specifies the end point using the SpelPoint data type.
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain CP, LJM, SYNC, Till or Find search expressions, and parallel processing statements.
- PointExpr
  Specifies the end point using a string expression.
  It can contain CP, LJM, SYNC, Till or Find search expressions, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Accel Method, **Speed Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Arch Method**, **Find Method**, **Sense Method**, **Till Method**, **SetPoint Method**

Go Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Go(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Go(pt)

' Points specified using expressions
m_spel.Go("P0 /L /2")
m_spel.Go("P1 :Z(-20)")

' Using parallel processing
m_spel.Go("P1 \!D50; On 1; D90; Off 1\!")
```

```
' Points specified using labels
m_spel.Go("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.Go(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.Go(pt);

// Points specified using expressions
m_spel.Go("P0 /L /2");
m_spel.Go("P1 :Z(-20)");

// Using a parallel processing
m_spel.Go("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.Go("pick");
```

15.3.8 H

15.3.8.1 Halt Method

Description

Suspends execution of the specified task.

Syntax

Sub Halt (TaskNumber As Integer)
Sub Halt (TaskName As String)

Parameters

- TaskNumber
  The task number of the task to be paused (1 to 32).
- TaskName
  A string expression indicating the task name to be paused.

> ⚠ Note
>
> The function executed by Call method cannot be paused by Halt method.
> If you need to pause, use Xqt Method.

See Also

Resume Method, **Xqt Method**

Halt Example

VB Example:

```
m_spel.Halt(3)
```

C# Example:

```
m_spel.Halt(3);
```

15.3.8.2 Here Method

Description

Teaches a point at the current position.

Syntax

Sub Here (PointNumber As Integer)
Sub Here (PointName As String)

Parameters

- PointNumber
  An integer expression indicating a point in the point memory for the current robot. Any valid point number can be used, as long as it starts with 0.
- PointName
  A string expression indicating the point label.

See Also

SetPoint Method

Here Example

VB Example:

```
m_spel.Here("P20")
```

C# Example:

```
m_spel.Here("P20");
```

15.3.8.3 HideWindow Method

Description

Hides an Epson RC+ 8.0 window that was previously displayed with ShowWindow.

Syntax

Sub HideWindow (WindowID As SpelWindows)

Parameters

- WindowID
  The ID of the Epson RC+ 8.0 window to hide.

See Also

RunDialog Method, **ShowWindow Method**

HideWindow Example

VB Example:

```
Sub btnHideIOMonitor_Click _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnHideIOMonitor.Click

    m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor)
End Sub
```

C# Example:

```
void btnHideIOMonitor_Click(object sender, EventArgs e)
{
   m_spel.HideWindow(RCAPINet.SpelWindows.IOMonitor);
}
```

15.3.8.4 Home Method

Description

Moves the robot arm to the user defined home position that is set with the HomeSet method.

Syntax

Sub Home ()

See Also

HomeSet Method, **MCal Method**

Home Example

VB Example:

```
With m_spel
    .MotorsOn = True
    .Home()
End With
```

C# Example:

```
m_spel.MotorsOn = true;
m_spel.Home();
```

15.3.8.5 Hofs Method

Description

Reads or sets the offset pulses used for software zero point correction.

Syntax

Function Hofs (JointNumber As Integer) As Integer
Sub Hofs (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer, J4Pulses As Integer, [J5Pulses As Integer], [J6Pulses As Integer], [J7Pulses As Integer], [J8Pulses As Integer] , [J9Pulses As Integer])

Parameters

- J1Pulses - J9Pulses
  Integers indicating the pulse values for Joints 1 to 9. J5Pulses to J9Pulses are optional

Return Value

The offset pulse value (integer value, in pulses).

Remarks

This method displays or sets the home position offset pulses. This method specifies the offset from the encoder 0 point (Z phase) to the mechanical 0 point.)

Although the robot motion control is based on the zero point of the encoder mounted on each joint motor, the encoder zero point may not necessarily match the robot mechanical zero point. To make the encoder position that matches the robot mechanical zero point as the zero point on the software, set the correction pulse amount by this method.

> ⚠ Note
>
> **Hofs Values SHOULD NOT be Changed unless Absolutely Necessary**
>
> The Hofs values are correctly specified prior to delivery. There is a danger that unnecessarily changing the Hofs value may result in position errors and unpredictable motion. Therefore, it is strongly recommended that Hofs values not be changed unless absolutely necessary.
>
> **To Automatically Calculate Hofs Values**
>
> To have Hofs values automatically calculated, move the arm to the desired calibration position, and execute Calib. The controller then automatically calculates Hofs values based on the CalPls pulse values and calibration position pulse values.
>
> **Saving and Restoring Hofs**
>
> Hofs can be saved and restored using the Save and Load commands in the [System Configuration] dialog-[Robot]-[Calibration] from the System Configuration menu.

See Also

CalPls Method, **Home Method**, **Hordr Method**, **MCal Method**

Hofs Example

VB Example:

```
Dim val As Integer
val = m_spel.Hofs(1)
```

C# Example:

```
int val;
val = m_spel.Hofs(1);
```

15.3.8.6 HomeSet Method

Description

Specifies the position used by the Home method.

Syntax

Sub HomeSet (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer, J4Pulses As Integer ,J5Pulses As Integer, J6Pulses As Integer [, J7Pulses As Integer] [, J8Pulses As Integer][, J9Pulses As Integer])

Parameters

- J1Pulses - J9Pulses
  The Home position encoder pulse value for each joint. Joints 7, 8, and 9 are optional.

See Also

Home Method, **MCal Method**

HomeSet Example

VB Example:

```
' Set the home position at the current position
With m_spel
    .HomeSet(.Pls(1), .Pls(2), .Pls(3), .Pls(4), 0, 0)
End With
```

C# Example:

```
//Set the home position at the current position
m_spel.HomeSet(m_spel.Pls(1), m_spel.Pls(2), m_spel.Pls(3),
          m_spel.Pls(4), 0 ,0);
```

15.3.8.7 Hordr Method

Description

Specifies the order of the axes returning to their HOME positions.

Syntax

Sub Hordr (Home1 As Integer, Home2 As Integer, Home3 As Integer, Home4 As Integer, Home5 As Integer, Home6 As Integer [, Home7 As Integer] [, Home8 As Integer] [, Home9 As Integer])

Parameters

- Home 1 - 9
  A bit pattern indicating which axes return to home during each step of the Home process. Any number of axes from 0 to all axes can return home at the 1st step. Home 7 – 9 can be specified when R, S, or T axis is specified.

See Also

Home Method, **HomeSet Method**, **Mcordr Method**

Hordr Example

VB Example:

```
m_spel.Hordr(2, 13, 0, 0, 0, 0)
```

C# Example:

```
m_spel.Hordr(2, 13, 0, 0, 0, 0);
```

15.3.8.8 Hour Method

Description

Returns the accumulated system operating time in hours.

Syntax

Function Hour () As Single

Return Value

Integer expression representing time.

Hour Example

VB Example:

```
Dim hoursRunning As Single
hoursRunning = m_spel.Hour()
```

C# Example:

```
float hoursRunning;
hoursRunning = m_spel.Hour();
```

15.3.9 I

15.3.9.1 ImportPoints Method

Description

Imports a point file into the current project for the current robot.

Syntax

Sub ImportPoints (SourcePath As String, ProjectFileName As String [, RobotNumber As Integer])

Parameters

- SourcePath
  A string expression indicating the file to import into the current project and its specific path. The extension .pts is appended.
- ProjectFileName
  A string expression indicating the specific file to be imported into the project for the currently used robot or the specified robot if RobotNumber is provided. The extension .pts is appended.
- RobotNumber
  Optional. An integer expression indicating the robot for which the points file is to be used. If you specify 0, it will be a points file.

See Also

SavePoints Method

ImportPoints Example

VB Example:

```
With m_spel
    .ImportPoints("c:\mypoints\model1.pts", "robot1.pts")
End With
```

C# Example:

```
m_spel.ImportPoints(@"c:\mypoints\model1.pts", "robot1.pts");
```

15.3.9.2 In Method

Description

Returns the status of the specified input port. Each port contains 8 input bits (one byte).

Syntax

Function In (PortNumber As Integer) As Integer
Function In (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the input port. Each port comprises 8 input bits (one byte).
- Label
  A string expression indicating the input byte label.

Return Value

Integer from 0 to 255 representing the status of the input port.

See Also

InBCD Method, **Out Method**, **OpBCD Method**, **Sw Method**

In Example

VB Example:

```
Dim port1Value As Integer
port1Value = m_spel.In(1)
```

C# Example:

```
int port1Value;
port1Value = m_spel.In(1);
```

15.3.9.3 InBCD Method

Description

Returns the input status of 8 inputs using BCD format. (Binary Coded Decimal)

Syntax

Function InBCD (PortNumber As Integer) As Integer

Function InBCD (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the input port.
- Label
  A string expression indicating the input byte label.

Return Value

Integer from 0 to 9 representing the status of the input port.

See Also

In Method, **Out Method**, **OpBCD Method**, **Sw Method**

InBCD Example

VB Example:

```
Dim port1Value As Integer
port1Value = m_spel.InBCD(1)
```

C# Example:

```
int port1Value;
port1Value = m_spel.InBCD(1);
```

15.3.9.4 Inertia Method

Description

Specifies the load inertia and eccentricity for the current robot.

Syntax

Sub Inertia (LoadInertia As Single, Eccentricity As Single)

Parameters

- LoadInertia
  Double value that specifies total moment of inertia in kgm2 around the center of the end effector joint, including end effector and part.
- Eccentricity
  Double value that specifies eccentricity in mm around the center of the end effector joint, including end effector and part.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Weight Method, **SetPoint Method**

Inertia Example

VB Example:

```
m_spel.Inertia(0.02, 1.0)
```

C# Example:

```
m_spel.Inertia(0.02, 1.0);
```

15.3.9.5 Initialize Method

Description

Initializes the Spel class instance.

Syntax

Sub Initialize ()

Remarks

Normally, the Spel class instance is automatically initialized when the first method has been executed. Initialization can take several seconds as Epson RC+ 8.0 loads into memory. So in some cases, you may want to call initialize first in your application during startup.
Initialize starts RC+ as a server process according to ServerInstance. Each ServerInstance corresponds to one controller and one project. If using the ServerInstance property, it must be set before executing Initialize.

See Also

Connect Method, **Disconnect Method**, **ServerInstance Property**

Initialize Example

VB Example:

```
m_spel.Initialize()
```

C# Example:

```
m_spel.Initialize();
```

15.3.9.6 InReal Method

Description

Returns the input data of 2 words (32 bits) as the floating-point data (IEEE754 compliant) of 32 bits.

Syntax

Function InReal (PortNumber As Integer) As Single

Parameters

- PortNumber
  An integer indicating the input port.

Return Value

Returns the input port status in 32 bits floating-point data (IEEE754 compliant).

See Also

In Method, **InBCD Method**, **InW Method**, **Out Method**, **OutW Method**, **OutReal Method**

InReal Example

VB Example:

```
Dim val As Single
val = m_spel.InReal(32)
```

C# Example:

```
float val;
val = m_spel.InReal(32);
```

15.3.9.7 InsideBox Method

Description

Returns the check status of the approach check area.

Syntax

Function InsideBox (BoxNumber As Integer) As Boolean

Parameters

- BoxNumber
  The number of the approach check area for which to return the detection status (integer between 1 and 15)

Return Value

True if the robot end effector is inside the specified box, False if not.

See Also

Box Method, **InsidePlane Method**

InsideBox Example

VB Example:

```
Dim isInside As Boolean
isInside = m_spel.InsideBox(1)
```

C# Example:

```
bool isInside;
isInside = m_spel.InsideBox(1);
```

15.3.9.8 InsidePlane Method

Description

Returns the check status of the approach check plane.

Syntax

Function InsidePlane (PlaneNumber As Integer) As Boolean

Parameters

- PlaneNumber
  The number of the approach check plane for which to return the detection status (integer between 1 and 15)

Return Value

True if the robot end effector is inside the specified box, False if not.

See Also

InsideBox Method, **Plane Method**

InsidePlane Example

VB Example:

```
Dim isInside As Boolean
isInside = m_spel.InsidePlane(1)
```

C# Example:

```
bool isInside;
isInside = m_spel.InsidePlane(1);
```

15.3.9.9 InW Method

Description

Returns the status of the specified input word port. Each word port contains 16 input bits.

Syntax

Function InW (PortNumber As Integer) As Integer
Function InW (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the input port.
- Label
  A string expression indicating the input word label.

Return Value

Integer value from 0 to 65535 representing the input port

See Also

In Method, **InBCD Method**, **Out Method**, **OpBCD Method**, **Sw Method**

InW Example

VB Example:

```
  Dim data As Integer
  data = m_spel.InW(0)
```

C# Example:

```
int data;
data = m_spel.InW(0);
```

15.3.9.10 IsOptionActive Method

Description

Returns the status of software options.

Syntax

Function IsOptionActive (option As SpelOptions) As Boolean

Parameters

- option
  An integer indicating the option number.

Return Value

False – Disabled
True – Enabled

See Also

GetControllerInfo Method

IsOptionActive Example

VB Example:

```
Dim ret As Boolean
ret = mspel.IsOptionActive(SpelOptions.FieldbusMaster)
```

C# Example:

```
bool ret;
ret = m_spel.IsOptionActive(SpelOptions.FieldbusMaster);
```

15.3.10 J

15.3.10.1 Jrange Method

Description

Defines the permissible working range of the specified robot joint in pulses.

Syntax

Sub JRange (JointNumber As Integer, LowerLimitPulses As Integer, UpperLimitPulses As Integer)

Parameters

- jointNumber
  An integer between 1 and 9 indicating the joint for which JRange will be specified.
- LowerLimitPulses
  Integer number representing the encoder pulse count position for the lower limit range of the specified joint.
- UpperLimitPulses
  An integer indicating the encoder pulse count position for the upper limit range of the specified joint

See Also

XYLim Method

JRange Example

VB Example:

```
m_spel.JRange(1, -30000, 30000)
```

C# Example:

```
m_spel.JRange(1, -30000, 30000);
```

15.3.10.2 JS Method

Description

Jump Sense detects whether the arm stopped prior to completing a JUMP instruction (which used a SENSE input) or if the arm completed the JUMP move.

Syntax

Function JS () As Boolean

Return Value

True if the SENSE input was detected during motion, False if not.

See Also

Jump Method, **Jump3 Method**, **Jump3CP Method**, **Sense Method**, **Till Method**

JS Example

VB Example:

```
With m_spel
    .Sense("Sw(1) = On")
    .Jump("P1 Sense")
    stoppedOnSense = .JS()
End With
```

C# Example:

```
m_spel.Sense("Sw(1) = On");
m_spel.Jump("P1 Sense");
stoppedOnSense = m_spel.JS();
```

15.3.10.3 JTran Method

Description

Executes a relative joint move.

Syntax

Sub JTran (JointNumber As Integer, Distance As Single)

Parameters

- JointNumber
  The specific joint to move.
- Distance
  The distance to move. Units are in degrees for rotary joints and millimeters for linear joints.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

PTran Method, **Pulse Method**, **SetPoint Method**

JTran Example

VB Example:

```
' Move joint 1 45 degrees in the plus direction.
m_spel.JTran(1, 45.0)
```

C# Example:

```
// Move joint 1 45 degrees in the plus direction.
m_spel.JTran(1, 45.0);
```

15.3.10.4 Jump Method

Description

Moves the arm from the current position to the specified point using point to point motion while first moving in a vertical direction up, then horizontally and then finally vertically downward to arrive on the final destination point.

Syntax

Sub Jump (PointNumber As Integer)
Sub Jump (Point As SpelPoint)
Sub Jump (Point As SpelPoint, AttribExpr As String)
Sub Jump (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the Jump motion. This is the final position at the end of the point to point motion.
PointNumber Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot.

- Point
  Specifies the end point using the SpelPoint data type.
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain a C arch number, LimZ, CP, SYNC, Sence, Till or Find search expression, and parallel processing statements.
- PointExpr
  Specifies the end point using a string expression.
  It can contain a C arch number, LimZ, CP, SYNC, Sence, Till or Find search expression, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Accel Method, **Speed Method**, **Executing Methods, Programs, Tasks**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Arch Method**, **Find Method**, **Sense Method**, **Till Method**, **SetPoint Method**

Jump Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Jump(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Jump(pt)

' Points specified using expressions
m_spel.Jump("P0 /L /2")
m_spel.Jump("P1 :Z(-20)")
m_spel.Jump("P1 C0")
m_spel.Jump("P1 C0 LimZ -10")
m_spel.Jump("P1 C0 Sense Sw(0)=On")
```

```
' Using parallel processing
m_spel.Jump("P1 \!D50; On 1; D90; Off 1\!")

' Points specified using labels
m_spel.Jump("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.Jump(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.Jump(pt);

// Points specified using expressions
m_spel.Jump("P0 /L /2");
m_spel.Jump("P1 :Z(-20)");
m_spel.Jump("P1 C0");
m_spel.Jump("P1 C0 LimZ -10");
m_spel.Jump("P1 C0 Sense Sw(0)=On");

// Using parallel processing
m_spel.Jump("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.Jump("pick");
```

15.3.10.5 Jump3 Method

Description

Motion with 3D gate using a combination of two CP motions and one PTP motion. The robot moves to the depart point, then the approach point, and finally the destination point.

Syntax

Sub Jump3 (DepartPoint As Integer, ApproPoint As Integer, DestPoint As Integer)
Sub Jump3 (DepartPoint As SpelPoint, ApproPoint As SpelPoint, DestPoint As SpelPoint)
Sub Jump3 (DepartPoint As String, ApproPoint As String, DestPoint As String)

Parameters

- DepartPoint
  Specifies the departure point above the current position using the point number or the character string indicating a point expression.
- ApproPoint
  Specifies the approach start point above the target coordinates using the point number or the character string indicating a point expression.
- DestPoint
  Specifies the target destination of the motion using the point number or the character string indicating a point expression. It can contain a C arch number, CP, LJM, SYNC, Sence, Till or Find search expression, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Accel Method, **AccelR Method**, **AccelS Method**, **Speed Method**, **SpeedR Method**, **SpeedS Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Arch Method**, **Find Method**, **Sense Method**, **Till Method**, **SetPoint Method**

Jump3 Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Jump3(1, 2, 3)

' Points specified using SpelPoint
Dim pd As SpelPoint
Dim pa As SpelPoint
Dim pt As SpelPoint
pd = m_spel.GetPoint("P*")
pd.Z = 125.5
pa = m_spel.GetPoint("P2")
pa.Z = 125.5
pt = m_spel.GetPoint("P2")
m_spel.Jump3(pd, pa, pt)

' Points specified using expressions
m_spel.Jump3("P1", "P2", "P3 C0")
m_spel.Jump3("P1", "P2", "P3 C0 Sense Sw(0)=On")
m_spel.Jump3("P0 -TLZ(10), P1 -TLZ(10), P1")

' Using parallel processing
m_spel.Jump3("P1", "P2", "P3 \!D50; On 1; D90; Off 1\!")
```

```
' Points specified using labels
m_spel.Jump3("depart", "approach", "place")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.Jump3(1, 2, 3);

// Points specified using SpelPoint
SpelPoint pd, pa, pt;
pd = m_spel.GetPoint("P1");
pd.Z = 125.5;
pa = m_spel.GetPoint("P2");
pa.Z = 125.5;
pt = m_spel.GetPoint("P2");
m_spel.Jump3(pd, pa, pt);

// Points specified using expressions
m_spel.Jump3("P1", "P2", "P3 C0");
m_spel.Jump3("P1", "P2", "P3 C0 Sense Sw(0)=On");
m_spel.Jump3("P0 -TLZ(10), P1 -TLZ(10), P1");

// Using parallel processing
m_spel.Jump3("P1", "P2", "P3 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.Jump3("depart", "approach", "place");
```

15.3.10.6 Jump3CP Method

Description

Motion with 3D gate using a combination of three CP motions.

Syntax

Sub Jump3CP (DepartPoint As Integer, ApproPoint As Integer, DestPoint As Integer)
Sub Jump3CP (DepartPoint As SpelPoint, ApproPoint As SpelPoint, DestPoint As SpelPoint)
Sub Jump3CP (DepartPoint As String, ApproPoint As String, DestPoint As String)

Parameters

- DepartPoint
  Specifies the departure point above the current position using the point number or the character string indicating a point expression.
- ApproPoint
  Specifies the approach start point above the target coordinates using the point number or the character string indicating a point expression.
- DestPoint
  Specifies the target destination of the motion using the point number or the character string indicating a point expression. It can contain a C arch number, ROT, CP, LJM, SYNC, Sence, Till or Find search expression, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelR Method, **AccelS Method**, **SpeedR Method**, **SpeedS Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Arch Method**, **Find Method**, **Sense Method**, **Till Method**, **SetPoint Method**

Jump3CP Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Jump3CP(1, 2, 3)

' Points specified using SpelPoint
Dim pd As SpelPoint
Dim pa As SpelPoint
Dim pt As SpelPoint
pd = m_spel.GetPoint("P*")
pd.Z = 125.5
pa = m_spel.GetPoint("P2")
pa.Z = 125.5
pt = m_spel.GetPoint("P2")
m_spel.Jump3CP(pd, pa, pt)

' Points specified using expressions
m_spel.Jump3CP("P1", "P2", "P3 C0")
m_spel.Jump3CP("P1", "P2", "P3 C0 Sense Sw(0)=On")
m_spel.Jump3CP("P1", "P2", "P3 \!D50; On 1; D90; Off 1\!")

' Using parallel processing
m_spel.Jump3CP("P0 -TLZ(10), P1 -TLZ(10), P1")
```

```
' Points specified using labels
m_spel.Jump3CP("depart", "approch", "place")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.Jump3CP(1, 2, 3);

// Points specified using SpelPoint
SpelPoint pd, pa, pt;
pd = m_spel.GetPoint("P0");
pd.Z = 125.5;
pa = m_spel.GetPoint("P2");
pa.Z = 125.5;
pt = m_spel.GetPoint("P2");
m_spel. Jump3CP(pd, pa, pt);

// Points specified using expressions
m_spel.Jump3CP("P1", "P2", "P3 C0");
m_spel.Jump3CP ("P1", "P2", "P3 C0 Sense Sw(0)=On");
m_spel.Jump3CP("P0 -TLZ(10), P1 -TLZ(10), P1");

// Using parallel processing
m_spel.Jump3CP("P1", "P2", "P3 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.Jump3CP("depart", "approch", "place");
```

15.3.11 L

15.3.11.1 LimitTorque Method

Description

Sets the upper limit torque in high power mode for the current robot.

Syntax

Sub LimitTorque (AllJointsMax As Integer)
Sub LimitTorque (J1Max As Integer, J2Max As Integer, J3Max As Integer, J4Max As Integer, J5Max As Integer, J6Max As Integer)

Parameters

- AllJointsMax
  An integer expression indicating the upper torque limit for all joints in the high power state.
- J1Max - J6Max
  An integer expression indicating the upper torque limit for each joint in the high power state.

Return Value

Integer value between 1 and 9 which represents the limit torque setting for the specified joint.

See Also

GetRealTorque Method, **GetRobotPos Method**

LimitTorque Example

VB Example:

```
Dim j1LimitTorque As Integer
j1LimitTorque = m_spel.LimitTorque(1)
```

C# Example:

```
int j1LimitTorquel
j1LimitTorque = m_spel.LimitTorque(1);
```

15.3.11.2 LimZ Method

Description

Sets the default value of the Z axis height for JUMP commands.

Syntax

Sub LimZ (ZLimit As Single)

Parameters

- ZLimit
  A coordinate value within the movable range of the Z axis.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Jump Method, **SetPoint Method**

LimZ Example

VB Example:

```
saveLimZ = m_spel.GetLimZ()
m_spel.LimZ(-22)
```

C# Example:

```
saveLimZ = m_spel.GetLimZ();
m_spel.LimZ(-22);
```

15.3.11.3 LoadPoints Method

Description

Loads a SPEL+ point file into the Controller's point memory for the current robot.

Syntax

Sub LoadPoints (FileName As String [, Merge As Boolean])

Parameters

- PointFileName
  Point file valid for the current project
- Merge
  Optional. Set this to integrate the current point into the specified point file.

See Also

ImportPoints Method, **SavePoints Method**

LoadPoints Example

VB Example:

```
With m_spel
    .LoadPoints("part1.pts")
End With
```

C# Example:

```
m_spel.LoadPoints("part1.pts");
```

15.3.11.4 Local Method

Description

Defines local coordinate systems.

Syntax

Sub Local (LocalNumber As Integer, OriginPoint As SpelPoint, [XAxisPoint As SpelPoint], [YAxisPoint As SpelPoint])
Sub Local (LocalNumber As Integer, LocalPoint1 As Integer, BasePoint1 As Integer, LocalPoint2 As Integer, BasePoint2 As Integer)
Sub Local (LocalNumber As Integer, LocalPoint1 As String, BasePoint1 As String, LocalPoint2 As String, BasePoint2 As String)

Parameters

- LocalNumbe
  The local coordinate system number. A total of 15 local coordinate systems (integer between 1 and 15) may be defined.
- OriginPoint
  A SpelPoint variable indicating the origin of the local coordinate system.
- XAxisPoint
  Optional. A SpelPoint variable indicating a point on the X axis in the local coordinate system.
- YAxisPoint
  Optional. A SpelPoint variable indicating a point on the Y axis in the local coordinate system.
- LocalPoint1, LocalPoint2
  Specifies the point data in the local coordinate system using an integer or character string.
- BasePoint1,BasePoint2
  Specifies the point data in the base coordinate system using an integer or character string.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Base Method, **SetPoint Method**

Local Example

VB Example:

```
Dim originPoint As New SpelPoint
originPoint.X = 100
originPoint.Y = 50
m_spel.Local(1, originPoint)
```

C# Example:

```
SpelPoint originPoint = new SpelPoint();
originPoint.X = 100;
originPoint.Y = 50;
m_spel.Local(1, originPoint);
```

15.3.11.5 LocalClr Method

Description

Clears a Local defined for the current robot.

Syntax

Sub LocalClr (LocalNumber As Integer)

Parameters

- LocalNumber
  The number of the coordinate system to clear (undefine) amongst the 15 local coordinate systems (integer between 1 and 15).

See Also

Local Method, **LocalDef Method**

LocalClr Example

VB Example:

```
m_spel.LocalClr(1)
```

C# Example:

```
m_spel.LocalClr(1);
```

15.3.11.6 LocalDef Method

Description

Returns local definition status.

Syntax

Function LocalDef (LocalNumber As Integer) As Boolean

Parameters

- LocalNumber
  An integer (1 to 15) indicating the number of the local coordinate system for which to return the status.

Return Value

True if the specified local is defined, False if not.

See Also

Local Method, **LocalClr Method**

LocalDef Example

VB Example:

```
Dim localExists As Boolean
localExists = m_spel.LocalDef(1)
```

C# Example:

```
bool localExists;
localExists = m_spel.LocalDef(1);
```

15.3.11.7 Login Method

Description

Log into Epson RC+ 8.0 as another user.

Syntax

Sub Login (LoginID As String, Password As String)

Parameters

- LoginID
  A string expression indicating the user's login ID.
- Password
  A string expression indicating the user's password.

Remarks

You can utilize Epson RC+ 8.0 security in your application. For example, you can display a menu that allows different users to log into the system. Each type of user can have its own security rights. For more details on security, see Epson RC+ User's Guide.
If security is enabled and you do not execute LogIn, then your Visual Basic application will be logged in as the guest user. Or if Auto LogIn is enabled in Epson RC+ 8.0, your application will automatically be logged in as the current Windows user if such a user has been configured in Epson RC+ 8.0.

See Also

GetCurrentUser Method

Login Example

VB Example:

```
With m_spel
    .Project = "c:\EpsonRC80\projects\myproject\myproject.sprj"
    .LogIn("operator", "oprpass")
End With
```

C# Example:

```
m_spel.Project = @"c:\EpsonRC80\projects\myproject\myproject.sprj";
m_spel.LogIn("operator", "oprpass");
```

15.3.12 M

15.3.12.1 MCal Method

Description

Executes machine calibration for robots with incremental encoders.

Syntax

Sub MCal ()

See Also

MCalComplete Method, **MotorsOn Property**

MCal Example

VB Example:

```
If Not m_spel.MCalComplete() Then
        m_spel.MCal()
End If
```

C# Example:

```
if (\!m_spel.MCalComplete())
        m_spel.MCal();
```

15.3.12.2 MCalComplete Method

Description

Returns True if MCal has been completed successfully.

Syntax

Function MCalComplete () As Boolean

Return Value

True if the MCal has completed, False if not.

See Also

MCal Method

MCalComplete Example

VB Example:

```
If m_spel.MCalComplete() Then
    lblStatus.Text = "MCal Complete"
Else
    lblStatus.Text = "MCal Not Complete"
End If
```

C# Example:

```
if (m_spel.MCalComplete())
    lblStatus.Text = "MCal Complete";
else
    lblStatus.Text = "MCal Not Complete";
```

15.3.12.3 Mcordr Method

Description

Specifies the moving axis order for machine calibration MCal.

Syntax

Sub MCordr (Step1 As Integer, Step2 As Integer, Step3 As Integer, Step4 As Integer, Step5 As Integer, Step6As Integer, [Step7 As Integer], [Step8 As Integer], [Step9 As Integer])

Parameters

- Step 1 - 9
  Bit pattern indicating which axes return to the home position during each step of the MCal process Any number of axes between 0 to all axes can return to the home position at the 1st step. Step 7\ -9 are optional and are used with robots that have more than 7 axes.

See Also

Home Method, **HomeSet Method**, **Hordr Method**, **MCal Method**

Mcordr Example

VB Example:

```
m_spel.Mcordr(2, 13, 0, 0, 0, 0)
```

C# Example:

```
m_spel.Mcordr(2, 13, 0, 0, 0, 0);
```

15.3.12.4 MemIn Method

Description

Returns the status of the specified memory I/O byte port. Each port contains 8 memory I/O bits.

Syntax

Function MemIn (PortNumber As Integer) As Integer
Function MemIn (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the memory I/O port
- Label
  A string expression indicating the memory I/O byte label.

Return Value

Integer expression from 0 to 255 representing the input port status.

See Also

In Method, **InBCD Method**, **MemOut Method**, **MemSw Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemIn Example

VB Example:

```
data = m_spel.MemIn(1)
```

C# Example:

```
data = m_spel.MemIn(1);
```

15.3.12.5 MemInW Method

Description

Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.

Syntax

Function MemInW (PortNumber As Integer) As Integer
Function MemInW (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the memory I/O port.
- Label
  A string expression indicating the memory I/O word label.

Return Value

Integer expression from 0 to 65535 representing the input port status.

See Also

In Method, **InBCD Method**, **MemIn Method**, **MemSw Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemInW Example

VB Example:

```
data = m_spel.MemInW(1)
```

C# Example:

```
data = m_spel.MemInW(1);
```

15.3.12.6 MemOff Method

Description

Turns off the specified bit of the S/W memory I/O.

Syntax

Sub MemOff (BitNumber As Integer)
Sub MemOff (Label As String)

Parameters

- BitNumber
  An integer expression indicating the memory I/O bit
- Label
  A string expression indicating the memory I/O bit label.

See Also

In Method, **InBCD Method**, **MemOut Method**, **MemSw Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemOff Example

VB Example:

```
m_spel.MemOff(500)
```

C# Example:

```
m_spel.MemOff(500);
```

15.3.12.7 MemOn Method

Description

Turns on the specified bit of memory I/O.

Syntax

Sub MemOn (BitNumber As Integer)
Sub MemOn (Label As String)

Parameters

- BitNumber
  An integer expression indicating the memory I/O bit
- Label
  A string expression indicating the memory I/O bit label.

See Also

In Method, **InBCD Method**, **MemOut Method**, **MemSw Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemOn Example

VB Example:

```
m_spel.MemOn(500)
```

C# Example:

```
m_spel.MemOn(500);
```

15.3.12.8 MemOut Method

Description

Simultaneously sets 8 memory I/O bits based on the 8 bit value specified by the user.

Syntax

Sub MemOut (PortNumber As Integer, Value As Integer)
Sub MemOut (Label As String, Value As Integer)

Parameters

- PortNumber
  An integer expression indicating the memory I/O byte
- Label
  A string expression indicating the memory I/O byte label.
- Value
  An integer expression indicating the output pattern of the output group for the specified byte. Valid range: 0 to 255

See Also

In Method, **InBCD Method**, **MemIn Method**, **MemSw Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemOut Example

VB Example:

```
m_spel.MemOut(2, 25)
```

C# Example:

```
m_spel.MemOut(2, 25);
```

15.3.12.9 MemOutW Method

Description

Simultaneously sets 16 memory I/O bits based on the 16 bit value specified by the user.

Syntax

Sub MemOutW (PortNumber As Integer, Value As Integer)
Sub MemOutW (Label As String, Value As Integer)

Parameters

- PortNumber
  An integer expression indicating the memory I/O word.
- Label
  A string expression indicating the memory I/O word label.
- Value
  Specifies the output data (integer from 0 to 65535) using an expression or number.

See Also

In Method, **InBCD Method**, **MemIn Method**, **MemSw Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemOutW Example

VB Example:

```
m_spel.MemOutW(2, 25)
```

C# Example:

```
m_spel.MemOutW(2, 25);
```

15.3.12.10 MemSw Method

Description

Returns the specified memory I/O bit status.

Syntax

Function MemSw (BitNumber As Integer) As Boolean
Function MemSw (Label As String) As Boolean

Parameters

- BitNumber
  An integer expression indicating one memory I/O bit
- Label
  A string expression indicating the memory I/O bit label.

Return Value

True if the specified memory I/O bit is on, False if not.

See Also

In Method, **InBCD Method**, **MemIn Method**, **Sw Method**, **Off Method**, **On Method**, **Oport Method**

MemSw Example

VB Example:

```
If m_spel.MemSw(10) Then
    m_spel.On(2)
End If
```

C# Example:

```
if (m_spel.MemSw(10))
m_spel.On(2);
```

15.3.12.11 Move Method

Description

Moves the arm from the current position to the specified point using linear interpolation (i.e. moving in a straight line).

Syntax

Sub Move (PointNumber As Integer)
Sub Move (Point As SpelPoint)
Sub Move (Point As SpelPoint, AttribExpr As String)
Sub Move (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the Move motion. This is the final position at the end of the linear interpolated motion.
PointNumber
Specifies the target point using a previously taught point number from the current robot point memory stored in the controller

- Point
  Specifies the target point using the SpelPoint data type
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain ROT, ECP, CP, SYNC, Till or Find search expressions, and parallel processing statements.
- PointExpr
  Specifies the target point using a string expression
  It can contain ROT, ECP, CP, SYNC, Till or Find search expressions, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelR Method, **AccelS Method**, **SpeedR Method**, **SpeedS Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **BGo Method**, **BMove Method**, **TGo Method**, **TMove Method**, **Arch Method**, **ECP Method**, **Find Method**, **Till Method**, **SetPoint Method**

Move Example

VB Examples:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.Move(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.Move(pt)

' Points specified using expressions
m_spel.Move("P0 /L /2 ROT")
m_spel.Move("P1 :Z(-20)")

' Using parallel processing
m_spel.Move("P1 \!D50; On 1; D90; Off 1\!")

' Points specified using labels
m_spel.Move("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.Move(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.Move(pt);

// Points specified using expressions
m_spel.Move("P0 /L /2 ROT");
m_spel.Move("P1 :Z(-20)");

// Using a parallel processing
m_spel.Move("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.Move("pick");
```

15.3.13 O

15.3.13.1 Off Method

Description

Turns off the specified output.

Syntax

Sub Off (BitNumber As Integer)
Sub Off (Label As String)

Parameters

- BitNumber
  An integer expression indicating the standard or extended output bits
- Label
  A string expression indicating the output bit label.

See Also

On Method, **Oport Method**, **Out Method**, **OutW Method**

Off Example

VB Example:

```
m_spel.Off(1)
```

C# Example:

```
m_spel.Off(1);
```

15.3.13.2 OLRate Method

Description
Returns overload rating for one joint for the current robot.

Syntax
Function OLRate (JointNumber As Integer) As Single

Parameters
- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)

Return Value
Returns the OLRate for the specified joint. Values are between 0.0 and 2.0.

Remarks
OLRate can be used to check whether a cycle is causing stress on the servo system. Factors such as temperature and current can cause servo errors during applications with high duty cycles. OLRate can help to check if the robot system is close to having a servo error.

During a cycle, run another task to command OLRate. If OLRate exceeds 1.0 for any joint, then a servo error will occur.

Servo errors are more likely to occur with heavy payloads. By using OLRate during a test cycle, you can help insure that the speed and acceleration settings will not cause a servo error during production cycling.

To get valid readings, you must execute this method while the robot is moving.

This method will not be used in proper payloads state.

OLRate Example
VB Example:

```
Dim data As Single
data = m_spel.OLRate(1)
```

C# Example:

```
float data;
data = m_spel.OLRate(1);
```

15.3.13.3 On Method

Description

Turns on the specified output.

Syntax

Sub On (BitNumber As Integer)
Sub On (Label As String)

Parameters

- BitNumber
  An integer expression indicating the standard or extended output bits
- Label
  A string expression indicating the output bit label.

See Also

Off Method, **Oport Method**, **Out Method**, **OutW Method**

On Example

VB Example:

```
m_spel.On(1)
```

C# Example:

```
m_spel.On(1);
```

15.3.13.4 OpBCD Method

Description

Simultaneously sets 8 output bits using BCD (Binary Coded Decimal) format.

Syntax

OpBCD (PortNumber As Integer, Value As Integer)
OpBCD (Label As String, Value As Integer)

Parameters

- PortNumber
  An integer indicating the I/O port Each port comprises 8 output bits (one byte).
- Value
  An integer between 0 and 99 indicating the output pattern for the specified port. The 2nd digit (called the 1's digit) represents the lower 4 output bits of the port and the 1st digit (called the 10's digit) represents the upper 4 output bits of the port.

See Also

Off Method, **Out Method**, **Sw Method**

OpBCD Example

VB Example:

```
m_spel.OpBCD(1, 25)
```

C# Example:

```
m_spel.OpBCD(1, 25);
```

15.3.13.5 Oport Method

Description

Returns the state of the specified output bit.

Syntax

Function Oport (BitNumber As Integer) As Boolean
Function Oport (Label As String) As Boolean

Parameters

- BitNumber
  An integer expression indicating the standard and extended output bits
- Label
  A character string indicating the output bit label.

Return Value

True if the specified output bit is on, False if not.

See Also

Off Method, **On Method**, **OpBCD Method**, **Out Method**, **Sw Method**

Oport Example

VB Example:

```
If m_spel.Oport(1) Then
    m_spel.On(2)
End If
```

C# Example:

```
if(m_spel.Oport(1))
        m_spel.On(2);
```

15.3.13.6 Out Method

Description

Simultaneously reads or sets 8 output bits (one byte).

Syntax

Sub Out (PortNumber As Integer, Value As Integer)
Sub Out (Label As String, Value As Integer)
Function Out (PortNumber As Integer) As Integer
Function Out (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the output port.
- Label
  A string expression indicating the output byte label.
- Value
  An integer between 0 and 255 indicating the output pattern for the output port. If represented in hexadecimal form, the range is from &H0 to &HFF.

Return Value

Integer number between 0-255 containing the port value.

See Also

InBCD Method, **OpBCD Method**, **Oport Method**, **OutW Method**, **Sw Method**

Out Example

VB Example:

```
m_spel.Out(1, 240)
```

C# Example:

```
m_spel.Out(1, 240);
```

15.3.13.7 OutReal Method

Description

Gets or sets the output port status as the 32 bits floating-point data (IEEE754 compliant).

Syntax

Function OutReal (WordPortNumber As Integer) As Single
Sub OutReal (WordPortNumber As Integer, Value As Single)

Parameters

- WordPortNumber
  An integer indicating the output port.
- Value
  A real number indicating the output data.

Return Value

Returns the specified output port status in 32 bits floating-point data (IEEE754 compliant).

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

In Method, **InBCD Method**, **InReal Method**, **InW Method**, **Out Method**, **OutW Method**, **SetPoint Method**

OutReal Example

VB Example:

```
Dim val As Single
val = m_spel.OutReal(32)
```

C# Example:

```
float val;
val = m_spel.OutReal(32);
```

15.3.13.8 OutW Method

Description

Simultaneously reads or sets 16 output bits (one word).

Syntax

Sub OutW (PortNumber As Integer, Value As Integer)
Sub OutW (Label As String, Value As Integer)
Function OutW (PortNumber As Integer) As Integer
Function OutW (Label As String) As Integer

Parameters

- PortNumber
  An integer indicating the output port.
- Label
  A string expression indicating the output word label.
- Value
  An integer between 0 and 65535 indicating the output pattern for the output port. If represented in hexadecimal form, the range is from &H0 to &HFFFF.

Return Value

Integer number between 0-65535 containing the port value.

See Also

InBCD Method, **OpBCD Method**, **Oport Method**, **Out Method**, **Sw Method**

OutW Example

VB Example:

```
m_spel.OutW(1, 240)
```

C# Example:

```
m_spel.OutW(1, 240);
```

15.3.14 P

15.3.14.1 PAgl Method

Description

Returns the joint angle for the selected rotational axis, or position for the selected linear axis, of the specified point.

Syntax

Function PAgl (PointNumber As Integer, JointNumber As Integer) As Single
Function PAgl (Point As SpelPoint, JointNumber As Integer) As Single
Function PAgl (Label As String, JointNumber As Integer) As Single

Parameters

- PointNumber
  An integer expression indicating the point number of a point in the current robot's point memory.
- Point
  A previously initialized SpelPoint
- Label
  A string expression indicating the label of the point that will become the point memory of the current robot.
- JointNumber
  An expression or number indicating the joint number (1 to 9).

Return Value

Single containing the angle for the specified joint in degrees or millimeters.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Agl Method, **Pls Method**, **CX, CY, CZ, CU, CV, CW, CR, CS, CT Method**, **SetPoint Method**

PAgl Example

VB Example:

```
Dim t1Angle As Single
t1Angle = m_spel.PAgl(1, 1)
```

C# Example:

```
float t1Angle;
t1Angle = m_spel.PAgl(1, 1);
```

15.3.14.2 Pallet Method

Description

Defines pallets.

Syntax

Sub Pallet (PalletNumber As Integer, Point1 As String, Point2 As String, Point3 As String [, Point4 As String] , rows As Integer, columns As Integer)

Parameters

- PalletNumber
  The pallet number indicated by an integer between 0 and 15
- Point1
  A point variable which defines the first pallet position.
- Point2
  A point variable which defines the second pallet position.
- Point3
  A point variable which defines the third pallet position.
- Point4
  Optional. A point variable which defines the fourth pallet position.
- Rows
  The number of horizontal points on the palette (integer 1 to 32767)
- Columns
  The number of vertical points on the palette (integer 1 to 32767)

See Also

Jump Method, **Go Method**, **SetPoint Method**

Pallet Example

VB Example:

```
m_spel.Pallet(1, 1, 2, 3, 4, 3, 4)
```

C# Example:

```
m_spel.Pallet(1, 1, 2, 3, 4, 3, 4);
```

15.3.14.3 Pass Method

Description

Specifies the PTP motion to pass a neighborhood of a specified point without stopping motion.

Syntax

Sub Pass(PointNumber As Integer)
Sub Pass(PassExpr As String)

Parameters

- PointNumber
  Specifies a waypoint using the point number of a previously taught point in the Controller's point memory for the current robot.
- PassExpr
  Specifies the end point using a string expression.
  Point specification [, {On | Off | MemOn | MemOff} Bit number [,point specification ...]] [LJM [Orientation flag]]

Point specification Specifies a point number, P(expression), or point label. If the point data is complete and listed in ascending or descending order, two point numbers can be combined using a colon and specified like P(1:5).
Bit number Specifies I/O output bit or memory I/O bit to turn on/off using an integer or output label.
LJM Converts the departure coordinates, approach coordinates, and target coordinates using the LJM function. Optional.
Orientation flag Specifies the orientation flag parameter for the LJM function. Optional.

See Also

Accel Method, **Go Method**, **Jump Method**, **Speed Method**

Pass Example

VB Example:

```
m_spel.Jump(1)
m_spel.Pass(2) 'Move the Arm #2 closer to P2 and execute the following
        command before reaching P2
m_spel.On(2)
m_spel.Pass(3)
m_spel.Pass(4)
m_spel.Off(0)
m_spel.Pass(5)
```

C# Example:

```
m_spel.Jump(1);
m_spel.Pass(2); //Move the Arm #2 closer to P2 and execute the following
        command before reaching P2
m_spel.On(2);
m_spel.Pass(3);
m_spel.Pass(4);
m_spel.Off(0);
m_spel.Pass(5);
```

15.3.14.4 Pause Method

Description

Causes all normal SPEL+ tasks in the Controller to pause. If the robot is moving, it will immediately decelerate to a stop.

Syntax

Sub Pause ()

> ⚠ CAUTION
>
> The function executed by Call Method cannot be paused by Pause Method.
> If you need to pause, use Xqt Method.

See Also

Continue Method, **EventReceived Event**, **Stop Method**

Pause Example

VB Example:

```
Sub btnPause_Click()_
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnPause.Click

    m_spel.Pause()
    btnPause.Enabled = False
    btnContinue.Enabled = True
End Sub
```

C# Example:

```
void btnPause_Click(object sender, EventArgs e)
{
    m_spel.Pause();
    btnPause.Enabled = false;
    btnContinue.Enabled = true;
}
```

15.3.14.5 PDef Method

Description

Returns the definition status of a specified point.

Syntax

Function PDef (PointNumber As Integer) As Boolean

Parameters

- PointNumber
  An integer expression indicating the point number of a point in the current robot's point memory.

Return Value

True if the specified point is defined, False if not.

See Also

PDel Method

PDef Example

VB Example:

```
Dim p1Defined As Boolean
p1Defined = m_spel.PDef(1)
```

C# Example:

```
bool p1Defined;
p1Defined = m_spel.PDef(1);
```

15.3.14.6 PDel Method

Description

Deletes specified position data.

Syntax

Sub PDel (FirstPointNumber As Integer [, LastPointNumber As Integer])

Parameters

- FirstPointNumber
  An integer expression that specifies the first point in the range to delete.
- LastPointNumber
  Optional. An integer expression that specifies the last point in range to delete. If omitted, only the point specified in FirstPointNumber is deleted.

See Also

PDef Method, **LoadPoints Method**, **ClearPoints Method**, **SavePoints Method**

PDel Example

VB Example:

```
m_spel.PDel(1, 10)
m_spel.SavePoints("model1.pts")
```

C# Example:

```
m_spel.PDel(1, 10);
m_spel.SavePoints("model1.pts");
```

15.3.14.7 PeakSpeed Method

Description

Returns the peak speed values for the specified joint.

Syntax

Function PeakSpeed (JointNumber As Integer) As Single

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)

Return Value

Real value from -1 to 1.

Remarks

This method returns the value of the maximum absolute speed for the joint with a sign. The peak speed is a real number from -1 to 1 with 1 being the maximum speed.

Execute PeakSpeedClear method first, and then execute this method to display the peak speed value for the joint.

When using the virtual controller or conducting dry-run, the average of the absolute speed values is calculated from the commanded speed instead of the actual speed.
This method does not support the PG additional axes.

See Also

AvgSpeed Method, **AvgSpeedClear Method**, **PeakSpeedClear Method**

PeakSpeed Example

VB Example:

```
Dim val As Single
val = m_spel.PeakSpeed(1)
```

C# Example:

```
float val;
val = m_spel.PeakSpeed(1);
```

15.3.14.8 PeakSpeedClear Method

Description

Clears and initializes the peak speed for one or more joints.

Syntax

Sub PeakSpeedClear ()

Remarks

This method clears the peak speed values for the specified joints.
You must execute this method before executing PeakSpeed method.

This method does not support the PG additional axes.

See Also

AvgSpeed Method, **AvgSpeedClear Method**, **PeakSpeed Method**

PeakSpeedClear Example

VB Example:

```
m_spel.PeakSpeedClear()
```

C# Example:

```
m_spel.PeakSpeedClear();
```

15.3.14.9 PF_Abort Method

Description

Forces the Part Feeding process to stop for the specified part.

Syntax

Sub PF_Abort (PartID As Integer)

Parameters

- PartID
  An integer indicating the part ID (1 to 16).

Remarks

Immediately aborts the Part Feeding process for the specified part.
Unlike PF_Stop method, this aborts the callback function in progress.
Nothing will occur when using this function when the Part Feeding process has not been started.

PF_Abort Example

VB Examples:

```
m_spel.PF_Abort(1)
```

C# Example:

```
m_spel.PF_Abort(1);
```

15.3.14.10 PF_Backlight Method

Description

Turns the built-in backlight on or off.

Syntax

Sub PF_Backlight (FeederNumber As Integer, State As Boolean)

Parameters

- FeederNumber
  An integer indicating the feeder number.
- State
  Specifies On (True) or Off (False).

Remarks

When the system is controlling vision, the backlight is automatically turned on and off as needed.
Use this method to control the built-in backlight on status when the system is not controlling vision.

PF_Backlight Example

VB Examples:

```
m_spel.PF_Backlight(1, True)
```

C# Example:

```
m_spel.PF_Backlight(1, true);
```

15.3.14.11 PF_BacklightBrightness Method

Description

Sets the brightness for the built-in backlight.

Syntax

Sub PF_Backlightness (FeederNumber As Integer, Brightness As Integer)

Parameters

- FeederNumber
  An integer indicating the feeder number.
- SBrightness
  Specifies the brightness as a value between 0 and 100.

Remarks

Normally, the built-in backlight brightness to be used for a part is set in the Part Feeding Configuration dialog. If changes to brightness are required at runtime, you can use this method to change it.

PF_BacklightBrightness Example

VB Examples:

```
m_spel.PFBacklightness(1, 80)
```

C# Example:

```
m_spel.PF_BacklightBrightness(1, 80);
```

15.3.14.12 PF_Name Method

Description

Gets a part name from a part ID.

Syntax

Function PF_Name (PartID As Integer) As String

Parameters

- PartID
  An integer indicating the part ID (1 to 16).

Return Value

Returns the name of the specified part ID as a character string.

Remarks

If the specified part ID is disabled, “ ” (empty character) is returned.

PF_Name Use Example

VB Examples:

```
Dim part1Name As String
Part1Name = m_spel.PF_Name(1)
```

C# Example:

```
string part1Name;
part1Name = m_spel.PF_Name(1);
```

15.3.14.13 PF_Number Method

Description

Gets a part ID from a part name.

Syntax

Function PF_Number (PartName As String) As Integer

Parameters

- PartName
  A character string indicating the part name.

Return Value

Returns the part ID (integer number from 1 to 16) for the specified part name.

Remarks

Returns −1 if the corresponding part name does not exist.
If multiple parts with the same name exist, the part with the smallest ID will be retrieved.

PF_Number Use Example

VB Examples:

```
Dim part1ID As Integer
Part1ID = m_spel.PF_Number("Part1")
```

C# Example:

```
int part1ID;
part1ID = m_spel.PF_Number("Part1");
```

15.3.14.14 PF_Start Method

Description

Starts the Part Feeding process for a specified part.

Syntax

Sub PF_Start (PartID1 As Integer, [PartID2 As Integer], [PartID3 As Integer], [PartID4 As Integer])

Parameters

- PartID1
  An integer indicating the master part ID (1 to 16).
- PartID2
  An integer indicating the slave part ID (1 to 16). Optional.
- PartID3
  An integer indicating the slave part ID (1 to 16). Optional.
- PartID4
  An integer indicating the slave part ID (1 to 16). Optional.

Remarks

Perform the following before starting this method.
- - Select the robot in use
- - Turn the motors on
- - Run PF_InitLog when outputting a log

Running this method generates a new task and returns control to the caller.

When this occurs, the Status callback function will be run under the following conditions. The Part Feeding process will not start.

Condition	Status callback function Status parameter value
The part ID is invalid	PF_STATUS_BAD_ID
Part parameter settings are invalid (Enabled check box not selected, etc.)	PF_STATUS_BAD_PARAMETER
Feeder calibration not complete	PF_STATUS_CAL_NOT_COMPLETE
An error occurred	PF_STATUS_ERROR

This method cannot be run multiple times at the same time. Once initiated, the processing already executed will continue. An error will not occur.
PF_Start should be run from a normal task. An error will occur when attempting to run PF_Task from a background task.

Caution If a part ID that does not exist is specified in this method, a 7600 error will occur.

PF_Start Use Example

VB Examples:

```
m_spel.PF_Start(1)
```

C# Example:

```
m_spel.PF_Start(1);
```

15.3.14.15 PF_Stop Method

Description

Issues a Part Feeding process end request.
This will wait for running callback functions to finish.
Once complete, the PF_CycleStop callback function will run and the process will stop.

Syntax

Sub PF_Stop (PartID As Integer)

Parameters

- PartID
  An integer indicating the part ID (1 to 16).

Remarks

Stops the Part Feeding process for the specified part.
Unlike the PF_Abort method, PF_Stop will wait for running callback functions to finish.
Once callback functions are complete, the PF_CycleStop callback function will run.
Nothing will occur when using this function when the Part Feeding process has not been started.

PF_Stop Use Example

VB Examples:

```
m_spel.PF_Stop(1)
```

C# Example:

```
m_spel.PF_Stop(1);
```

15.3.14.16 PLabel Method

Description

Gets or sets the point label associated with a point number.

Syntax

Function PLabel (PointNumber As Integer) As String
Sub PLabel (PointNumber As Integer, PointName As String)

Parameters

- PointNumber
  An integer indicating the point number.
- PointName
  A character string that specifies the label to use for the designated pointer data.

Return Value

Returns the corresponding label to the specified point number.

See Also

PDef Method

PLabel Example

VB Example:

```
Dim pt1Label As String
Pt1Label = m_spel.PLabel(1)
```

C# Example:

```
string pt1Label;
pt1Label = m_spel.PLabel(1);
```

15.3.14.17 Plane Method

Description

Defines a Plane.

Syntax

Sub Plane (PlaneNumber As Integer, Point As SpelPoint)
Sub Plane (PlaneNumber As Integer, X As Single, Y As Single, Z As Single, U As Single, V As Single, W As Single)

Parameters

- PlaneNumber
  An integer expression from 1 to 15 indicating which of 15 planes to define.
- Point
  Point data indicating the coordinate data of the approach check plane
- X
  The X coordinate of the point indicating the coordinate data of the approach check plane.
- Y
  The Y coordinate of the point indicating the coordinate data of the approach check plane.
- Z
  The Z coordinate of the point indicating the coordinate data of the approach check plane.
- U
  The U coordinate of the point indicating the coordinate data of the approach check plane.
- V
  The V coordinate of the point indicating the coordinate data of the approach check plane.
- W
  The W coordinate of the point indicating the coordinate data of the approach check plane.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

PlaneClr Method, **PlaneDef Method**, **SetPoint Method**

Plane Example

VB Example:

```
m_spel.Plane(1, -5, 5, -10, 10, -20, 20)
```

C# Example:

```
m_spel.Plane(1, -5, 5, -10, 10, -20, 20);
```

15.3.14.18 PlaneClr Method

Description

Clears (undefines) a Plane.

Syntax

Sub PlaneClr (PlaneNumber As Integer)

Parameters

- PlaneNumber
  An integer indicating which of the 15 approach check planes to clear (1 to 15).

See Also

Plane Method, **PlaneDef Method**

PlaneClr Example

VB Example:

```
m_spel.PlaneClr(1)
```

C# Example:

```
m_spel.PlaneClr(1);
```

15.3.14.19 PlaneDef Method

Description

Returns whether a plane is defined.

Syntax

Function PlaneDef (PlaneNumber As Integer) As Boolean

Parameters

- PlaneNumber
  An integer indicating the approach check plane number (1 to 15).

Return Value

True if the specified plane is defined, False if not.

See Also

Plane Method, **PlaneClr Method**

PlaneDef Example

VB Example:

```
x = m_spel.PlaneDef(1)
```

C# Example:

```
x = m_spel.PlaneDef(1);
```

15.3.14.20 Pls Method

Description

Returns the current encoder pulse count for each axis at the current position.

Syntax

Function Pls (JointNumber As Integer) As Integer

Parameters

- JointNumber
  Specify the axis to acquire the current pulse count for (1 to 9).

Return Value

Integer containing the current pulse count for the specified joint.

See Also

Agl Method, **Pulse Method**

Pls Example

VB Example:

```
j1Pulses = m_spel.Pls(1)
```

C# Example:

```
j1Pulses = m_spel.Pls(1);
```

15.3.14.21 PTCLR Method

Description

Clears and initializes the peak torque for one or more joints.

Syntax

Sub PTCLR ()

Remarks

This method clears the peak torque values for the specified joints.
You must execute this method before executing PTRQ method.

See Also

ATCLR Method, **ATRQ Method**, **PTRQ Method**

PTCLR Example

VB Example:

```
m_spel.PTCLR ()
```

C# Example:

```
m_spel.PTCLR();
```

15.3.14.22 PTPBoost Method

Description

Sets the boost parameters for short distance PTP (point to point) motion.

Syntax

Sub PTPBoost (BoostValue As Integer [, DepartBoost As Integer] [, ApproBoost As Integer])

Parameters

- BoostValue
  Integer from 0 to 100.
- DepartBoost
  Optional. An integer (0 to 100) indicating the depart adjustment setting of the Z coordinate during a jump operation.
- ApproBoost
  Optional. An integer (0 to 100) indicating the approach adjustment setting of the Z coordinate during a jump operation.

See Also

PTPBoostOK Method

PTPBoost Example

VB Example:

```
m_spel.PTPBoost(50)
m_spel.PTPBoost(50, 30, 30)
```

C# Example:

```
m_spel.PTPBoost(50);
m_spel.PTPBoost(50, 30, 30);
```

15.3.14.23 PTPBoostOK Method

Description

Returns whether or not the PTP (Point to Point) motion from a current position to a target position is a small travel distance.

Syntax

Function PTPBoostOK (PointNumber As Integer) As Boolean
Function PTPBoostOK (Point As SpelPoint) As Boolean
Function PTPBoostOK (PointExpr As String) As Boolean

Parameters

Each syntax has one parameter that specifies the target point to check.

- PointNumber
  Specifies the target point using the point number of a previously taught point in the Controller's point memory for the current robot.
- Point
  Specifies the target point using the SpelPoint data type
- PointExpr
  Specifies the target point using a string expression

Return Value

True if PTPBoost will be used, False if not.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

PTPBoost Method, **SetPoint Method**

PTPBoostOK Example

VB Example:

```
If m_spel.PTPBoostOK(1) Then
    m_spel.Go(1)
End If
```

C# Example:

```
if (m_spel.PTPBoostOK(1))
        m_spel.Go(1);
```

15.3.14.24 PTran Method

Description

Executes a relative joint move in pulses.

Syntax

Sub PTran (JointNumber As Integer, Pulses As Integer)

Parameters

- JointNumber
  The joint to move
- Pulses
  Pulse count for movement

See Also

JTran Method, **Pulse Method**

PTran Example

VB Example:

```
' Move joint 1 5000 pulses in the plus direction.
m_spel.PTran(1, 5000)
```

C# Example:

```
//Move joint 1 5000 pulses in the plus direction.
m_spel.PTran(1, 5000);
```

15.3.14.25 PTRQ Method

Description

Returns the peak torque for the specified joint.

Syntax

Function PTRQ (JointNumber As Integer) As Single

Parameters

- JointNumber
  An integer indicating the axis number (range: 1 to number of joints of the robot)

Return Value

Real value from 0 to 1.

See Also

ATCLR Method, **ATRQ Method**, **PTCLR Method**

PTRQ Example

VB Example:

```
Dim peakTorque As Single
peakTorque = m_spel.PTRQ(1)
```

C# Example:

```
float peakTorque;
peakTorque = m_spel.PTRQ(1);
```

15.3.14.26 Pulse Method

Description

Moves the robot arm by Point to Point control to the point specified by the pulse values for all robot joints.

Syntax

Sub Pulse (J1Pulses As Integer, J2Pulses As Integer, J3Pulses As Integer, J4Pulses As Integer [, J5Pulses As Integer] [, J6Pulses As Integer] [, J7Pulses As Integer] [, J8Pulses As Integer] [, J9Pulses As Integer])

Parameters

- J1Pulses - J9Pulses
  An integer expression indicating the pulse values for Joints 1 to 9. J5Pulses to J9Pulses are optional.

KEY POINTS

The pulse values must be within the range specified each joint.

See Also

Go Method, **Move Method**, **Jump Method**

Pulse Example

VB Example:

```
m_spel.Pulse(5000, 1000, 0, 0)
```

C# Example:

```
m_spel.Pulse(5000, 1000, 0, 0);
```

15.3.15 Q

15.3.15.1 Quit Method

Description

Terminates execution of the specified task.

Syntax

Sub Quit (TaskNumber As Integer)
Sub Quit (TaskName As String)

Parameters

- TaskNumber
  The task number for the task to abort (1 to 32).
- TaskName
  A string expression indicating the task name.

> ⚠ Note
>
> To stop the function executed by Call Method, use Stop Method.

See Also

Halt Method, **Resume Method**, **Xqt Method**

Quit Example

VB Example:

```
m_spel.Quit(3)
```

C# Example:

```
m_spel.Quit(3);
```

15.3.16 R

15.3.16.1 RadToDeg Method

Description

Converts Radians into Degrees.

Syntax

Function RadToDeg (Radians As Double) As Double

Parameters

- Radians
  A real number indicating the number of radians to convert to degrees.

Return Value

Double containing the converted value in degrees.

See Also

Executing Methods, Programs, Tasks, **DegToRad Method**

RadToDeg Example

VB Example:

```
Dim deg As Double

deg = m_spel.RadToDeg(1)
```

C# Example:

```
double deg;
deg = m_spel.RadToDeg(1);
```

15.3.16.2 RebootController Method

Description

Reboot the currently connected controller.

Syntax

Sub RebootController (ShowStatusDialog As Boolean)

Parameters

- ShowStatusDialog
  Sets whether to display the status screen until the restart is complete.
  True = show a dialog False = do not show a dialog

Remarks

Use ShowStatusDialog to show a dialog with a progress bar. You can abort the operation from the status dialog or by executing the Abort method.

See Also

Stop Method

RebootController Example

VB Example:

```
m_spel.RebootController(True)
```

C# Example:

```
m_spel.RebootController(true);
```

15.3.16.3 RebuildProject Method

Description

Completely rebuilds the current project specified in the Project property.

Syntax

Sub RebuildProject ()

See Also

.BuildProject Method, **EnableEvent Method**, **EventReceived Event**, **Project Property**, **ProjectBuildComplete Property**

RebuildProject Example

VB Example:

```
With m_spel
    .Project = "c:\EpsonRC80\projects\myproject\myproject.sprj"
    .RebuildProject
End With
```

C# Example:

```
m_spel.Project = @"c:\EpsonRC80\projects\myproject\myproject.sprj";
m_spel.RebuildProject();
```

15.3.16.4 Recover Method

Description

Recover moves the robot back to the position is was in when the safeguard was open.

Syntax

Function Recover () As Boolean

Remarks

The Recover method can be used after the safeguard is closed to turn on the robot motors and slowly move the robot back to the position it was in when the safeguard was open. After Recover has completed successfully, you can execute the Cont method to continue the cycle. If Recover was completed successfully, it will return True. Recover will return False if a pause, abort, or safeguard open occurred during recover motion.

Return Value

True if the recover motion was completed, False if not.

See Also

Continue Method, **Pause Method**

Recover Example

VB Examples:
This example executes a recover, then continue

```
Sub btnCont_Click( _
   ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles btnCont.Click
  Dim sts As Boolean
  Dim answer As Integer

  sts = m_spel.Recover()
If sts = False Then
  Exit Sub
End If
answer = MsgBox("Ready to continue?", vbYesNo)
If answer = vbYes Then
  m_spel.Continue()
EndIF
End sub
```

In the example below, the cycle continues after recover is run. This example shows how the recover action is run as long as the button is held down. If the button is released during the recover motion, a pause is issued and recover is aborted. If the button is held down until recover has completed, then a message is displayed.

```
Sub btnRecover_MouseDown( _
    ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles btnRecover.MouseDown
Dim sts As Boolean

sts = m_spel.Recover()
If sts = True Then
    MsgBox("Recover complete")
EndIf
End Sub

Sub btnRecover_MouseUp( _
    ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles btnRecover.MouseUp
```

```
m_spel.Pause()
End Sub
```

C# Example:
This example executes a recover, then continue

```
void btnCont_Click(object sender, EventArgs e)
{
    bool sts;
    DialogResult answer;

    sts = m_spel.Recover();
    if (sts == true){
            answer = MessageBox.Show("Continue?", "",
            MessageBoxButtons.YesNo);
        If (answer == DialogResult.Yes)
            m_spel.Continue();
    }
}
```

In the example below, the cycle continues after recover is run. This example shows how the recover action is run as long as the button is held down. If the button is released during the recover motion, a pause is issued and recover is aborted. If the button is held down until recover has completed, then a message is displayed.

```
  void btnCont_Click(object sender, EventArgs e)
  {
    bool sts;

    sts = m_spel.Recover();
    if (sts == true)
        MessageBox.Show("Recover complete");
  }

  void btnRecover_MouseUp(oject sender, EventArgs e)
  {
    m_spel.Pause();
  }
```

15.3.16.5 Reset Method

Description

Resets the Controller to the initialized state.

Syntax

Sub Reset ()

See Also

Executing Methods, Programs, Tasks, **ResetAbort Method**

Reset Example

VB Example:

```
m_spel.Reset
```

C# Example:

```
m_spel.Reset();
```

15.3.16.6 ResetAbort Method

Description

Resets the abort flag that is set with the Stop method.

Syntax

Sub ResetAbort ()

Remarks

When the Stop method is executed and no other Spel method is in cycle, then the next Spel method will generate a user abort error. This is done so that no matter when the Stop is issued, the routine that is executing Spel methods will receive the error. Use ResetAbort to clear this condition.

> **Note:**
>
> The ResetAbortEnabled property must be set to True for the ResetAbort feature to work.

See Also

Stop Method, **Reset Method**, **ResetAbortEnabled Property**

ResetAbort Example

VB Example:

```
Sub btnMcal_Click() Handles btnMcal.Click
  m_spel.ResetAbort()
  m_spel.MCal()
End Sub
```

C# Example:

```
void btnMCal_Click(object sender, EventArgs e)
{
 m_spel.ResetAbort();
 m_spel.MCal();
}
```

15.3.16.7 Resume Method

Description

Resumes a task which was suspended by the Halt method.

Syntax

Sub Resume (TaskNumber As Integer)
Sub Resume (TaskName As String)

Parameters

- TaskNumber
  The task number for the aborted task (1 to 32).
- TaskName
  A character string indicating the task name.

See Also

Quit Method, **Xqt Method**

Resume Example

VB Example:

```
m_spel.Resume(2)
```

C# Example:

```
m_spel.Resume(2);
```

15.3.16.8 RunDialog Method

Description

Runs an Epson RC+ 8.0 dialog.

Syntax

Sub RunDialog (DialogID As SpelDialogs [, Parent As Form])

Parameters

- DialogID
  The dialog ID of the Epson RC+ 8.0 to run.
- Parent
  Optional. A .NET form that is the parent of the window.

See Also

ShowWindow Method

RunDialog Example

VB Example:

```
Sub btnRobotManager_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnRobotManager.Click

    m_spel.RunDialog(SpelDialogs.RobotManager)
End Sub
```

C# Example:

```
void btnRobotManager_Click(object sender, EventArgs e)
{
        m_spel.RunDialog(SpelDialogs.RobotManager);
}
```

15.3.17 S

15.3.17.1 SavePoints Method

Description

Save points for the current robot into a file.

Syntax

Sub SavePoints (FileName As String)

Parameters

- PointFileName
  The file name to save the points in the current project.

See Also

LoadPoints Method

SavePoints Example

VB Example:

```
With m_spel
    .SavePoints("part1.pts")
End With
```

C# Example:

```
m_spel.SavePoints("part1.pts");
```

15.3.17.2 Sense Method

Description

Specifies input condition that, if satisfied, completes the Jump in progress by stopping the robot above the target position.

Syntax

Sub Sense (Condition As String) As Boolean

Parameters

- Condition
  For details about the setting the I/O conditions, see “Sense” in the SPEL+ Language Reference manual.

See Also

Jump Method, **JS Method**

Sense Example

VB Example:

```
With m_spel
    .Sense("Sw(1) = On")
    .Jump("P1 SENSE")
    stoppedOnSense = .JS()
End With
```

C# Example:

```
m_spel.Sense("Sw(1) = On");
m_spel.Jump("P1 SENSE");
stoppedOnSense = m_spel.JS();
```

15.3.17.3 SetIODef Method

Description

Sets the I/O label and description for an input, output, or memory I/O bit, byte, or word.

Syntax

Sub SetIODef (Type As SpelLabelTypes, Index As Integer, Label As String, Description As String)

Parameters

- Type Specifies the I/O type as shown below: InputBit = 1 InputByte = 2 InputWord = 3 OutputBit = 4 OutputByte = 5 OutputWord = 6 MemoryBit = 7 MemoryByte = 8 MemoryWord = 9 InputReal = 10 OutputReal = 11
- Index
  Specifies the bit or port number.
- Label
  Specifies a new label.
- Description
  Specifies a new comment.

Remarks

Use SetIODef to define the label and description for any I/O point.

See Also

GetIODef Method

SetIODef Example

VB Example:

```
Dim label, desc As String
label = "StartCycle"
desc = "Starts the robot cycle"
m_spel.SetIODef(SpelLabelTypes.InputBit, 0, label, desc)
```

C# Example:

```
string label, desc;
label = "StartCycle";
desc = "Starts the robot cycle";
m_spel.SetIODef(SpelLabelTypes.InputBit, 0, label, desc);
```

15.3.17.4 SetPoint Method

Description

Sets the coordinate data for a point for the current robot.

Syntax

Sub SetPoint(PointNumber As Integer, Point As SpelPoint)
Sub SetPoint(PointLabel As String, Point As SpelPoint)

Sub SetPoint(PointNumber As Integer, X As Single, Y As Single, Z As Single, U As Single)
Sub SetPoint(PointLabel As String, X As Single, Y As Single, Z As Single, U As Single)

Sub SetPoint(PointNumber As Integer, X As Single, Y As Single, Z As Single, U As Single,Local As Integer, Hand As SpelHand)
Sub SetPoint(PointLabel As String, X As Single, Y As Single, Z As Single, U As Single,Local As Integer, Hand As SpelHand)

Sub SetPoint(PointNumber As Integer, X As Single, Y As Single, Z As Single, U As Single,V As Single, W As Single)
Sub SetPoint(PointLabel As String, X As Single, Y As Single, Z As Single, U As Single,V As Single, W As Single)

Sub SetPoint(PointNumber As Integer, X As Single, Y As Single, Z As Single, U As Single,V As Single, W As Single, Local As Integer, Hand As SpelHand, Elbow As SpelElbow, Wrist As SpelWrist, J4Flag As Integer, J6Flag As Integer)
Sub SetPoint(PointLabel As String, X As Single, Y As Single, Z As Single, U As Single,V As Single, W As Single, Local As Integer, Hand As SpelHand, Elbow As SpelElbow, Wrist As SpelWrist, J4Flag As Integer, J6Flag As Integer)

Sub SetPoint(PointNumber As Integer, X As Single, Y As Single, Z As Single, U As Single,V As Single, W As Single, S As Single, T As Single)
Sub SetPoint(PointLabel As String, X As Single, Y As Single, Z As Single, U As Single,V As Single, W As Single, S As Single, T As Single)

Sub SetPoint(PointNumber As Integer, PointExpr As String)
Sub SetPoint(PointLabel As String, PointExpr As String)

Parameters

- PointNumber
  An integer expression that specifies the point number of a point in the current robot's point memory.
- X
  The X coordinate for the specified point.
- Y
  The Y coordinate for the specified point.
- Z
  The Z coordinate for the specified point.
- U
  The U coordinate for the specified point.
- V
  The V coordinate for the specified point.
- W
  The W coordinate for the specified point.
- S
  The S coordinate for the specified point.
- T
  The T coordinate for the specified point.

- Local
  The Local Number for the specified point. Use 0 when there is no local.
- Hand
  The hand orientation at the specified point.
- Elbow
  The elbow orientation at the specified point.
- Wrist
  The wrist orientation at the specified point.
- PointExpr
  Specifies a point using a string expression.
  Note
  Do not enter integer values to X, Y, Z, U, V, W, S, and T parameters. Use Single variables or directly enter Single type values.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, do one of the following:

1. In Windows, go to [Control Panel] - [Clock and Region] - [Region] - [Formats] - [Additional settings] and change [Decimal symbol] to ".". Then start your .NET application.
2. Before calling the SetPoint method in your .NET application, set the CurrentCulture property of the System.Globalization.CultureInfo class to a CultureInfo instance with the decimal separator set to ".". If necessary, set the CurrentCulture property back to its original setting after the SetPoint method call.

CurrentCulture setting example (C#):

```
// CurrentCulture Save the original settings of the properties
var ciOrg = CultureInfo.CurrentCulture;
try
{
    // Duplicate a new setting from the original and set the decimal separator with
NumberFormat.NumberDecimalSeparator
    var ciNew = (CultureInfo)ciOrg.Clone();
    ciNew.NumberFormat.NumberDecimalSeparator = ".";
    // CurrentCulture Set the duplicated property to the property
    CultureInfo.CurrentCulture = ciNew;
    // Call the SetPoint method
    m_spel.SetPoint(1, pt);
}
finally
{
    // Reset CurrentCulture
    CultureInfo.CurrentCulture = ciOrg;
}
```

See Also

GetPoint Method, **LoadPoints Method**, **SavePoints Method**

SetPoint Example

Example of RC + project SPEL + program to run:

```
Global Integer P_WorkDetect
Global Real offsetX, offsetY, heightZ, offsetU
Function SetOffset
     P_WorkDetect = 5
     offsetX = 0.5
     offsetY = 0.3
     heightZ = 10.0
```

```
     offsetU = 1.2
Fend
```

VB Example:

```
Dim pt As SpelPoint
' Get coordinates of P1
pt = m_spel.GetPoint(1)
' Set it with changes
pt.U = pt.U - 10.5
’ Specify orientation flag
pt.J1Flag = 1
m_spel.SetPoint(1, pt)
’ Run RC + project SPEL + function
m_spel.Call("SetOffset")
’ Set coordinates and orientation using string expressions
m_spel.SetPoint(2, "P(P_WorkDetect) +X(offsetX) +Y(offsetY) :Z(heightZ) +U
(offsetU) /A /J1F1")
```

C# Example:

```
SpelPoint pt;
// Get coordinates of P1
pt = m_spel.GetPoint(1);
// Set it with changes
pt.U = pt.U - 10.5;
// Specify orientation flag
pt.J1Flag = 1;
m_spel.SetPoint(1, pt);
// Run RC + project SPEL + function
m_spel.Call("SetOffset")
// Set coordinates and orientation using string expressions
m_spel.SetPoint(2, "P(P_WorkDetect) +X(offsetX) +Y(offsetY) :Z(heightZ) +U
(offsetU) /A /J1F1");
```

15.3.17.5 SetVar Method

Description

Sets the value of a SPEL+ global preserve variable in the Controller.

Syntax

Sub SetVar (VarName As String, Value As Object)

Parameters

- VarName
  SPEL + backup variable name
- Value
  Value to set

Remarks

You can use SetVar to set the values for single variables and array variables. See the examples below.

See Also

GetVar Method

SetVar Example

VB Example:

```
m_spel.SetVar("g_myIntVar", 123)

Dim i, myArray(10) As Integer
For i = 1 To 10
  myArray(i) = i
Next i
m_spel.SetVar("g_myIntArray", myArray)

m_spel.SetVar("g_myIntArray(1)", myArray(1))
```

C# Example:

```
m_spel.SetVar("g_myIntVar", 123);

int[] myArray = new int[10];
for(int i = 1; i < 10; i++)
        myArray[i] = i;

m_spel.SetVar("g_myIntArray", myArray);

m_spel.SetVar("g_myIntArray[1]", myArray[1]);
```

15.3.17.6 SFree Method

Description

Free joint the specified robot axes.

Syntax

Sub SFree ()
Sub SFree (ParamArray Axes() As Integer)

Parameters

- Axes
  An integer parameter array indicating the robot axis elements to free. You can specify axis numbers from 1 to 9.

See Also

Slock Method

SFree Example

VB Example:

```
' State J1 and J2 to free joint
m_spel.SFree(1, 2)
```

C# Example:

```
// State J1 and J2 to free joint
m_spel.SFree(1, 2);
```

15.3.17.7 ShowWindow Method

Description

Displays an Epson RC+ 8.0 window.

Syntax

Sub ShowWindow (WindowID As SpelWindows [, Parent As Form])

Parameters

- WindowID
  The ID of the Epson RC+ 8.0 window to show.
- Parent
  Optional. A .NET form that is the parent of the window.

Remarks

You can use the Parent parameter to specify the .NET parent form for the window. If you cannot use a .NET parent form, you must omit the Parent parameter and use the ParentWindowHandle property to set the handle of the parent.

See Also

HideWindow Method, **ParentWindowHandle Property**, **RunDialog Method**

ShowWindow Example

VB Example:

```
Sub btnShowIOMonitor_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnShowIOMonitor.Click

    m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, Me)
End Sub
```

C# Example:

```
void btnShowIOMonitor_Click(object sender, EventArgs e)
{
        m_spel.ShowWindow(RCAPINet.SpelWindows.IOMonitor, this);
}
```

15.3.17.8 SimGet Method

Description

Acquire the setting values of each object properties of simulator.

Syntax

Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Boolean)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps, ByRef Value As Boolean)
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Double)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps, ByRef Value As Double)
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Integer)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps, ByRef Value As Integer)
Sub SimGet (Object As String, Property As SpelSimProps, ByRef Value As Boolean)
Sub SimGet (RobotName As String, HandName As String, Property As SpelSimProps, ByRef Value As String)

Parameters

- Object
  String variable that indicates the object name to acquire the property values for.
- RobotName
  Sting variable that indicates the robot name to which the hand specified in “Hand” is mounted.
- HandName
  String variable that indicates the hand name to acquire the property values for.
- Property
  The name of the property to acquire the values for.
- Value
  Variable indicating the returned value.

Remarks

Use this method to acquire the property setting value of each object of the simulator.

For each property details, refer to the following manual.
Epson RC+ 8.0 SPEL+ Language Reference - SimGet

See Also

SimSet Method

SimGet Example

VB Example:

```
Dim posX As Double
m_spel.SimGet("SBox_1", SpelSimProps.PositionX, posX)
```

C# Example:

```
double posX;
SimGet("SBox_1", SpelSimProps.PositionX, out posX);
```

15.3.17.9 SimResetCollision Method

Description

Resets collision detection.

Syntax

Sub SimResetCollision ()

Remarks

If there is no collision between the robot and the object after executing this method, the collision state is reset and the 3D display of simulator is updated. If there is collision between the robot and the object, the collision state will not be reset and the 3D display of simulator will not be updated.

For more details, refer to the following manual.
Epson RC+ 8.0 SPEL+ Language Reference - SimSet

See Also

SimSet Method

SimResetCollision Example

VB Example:

```
m_spel.SimResetCollision ()
```

C# Example:

```
m_spel.SimResetCollision ();
```

15.3.17.10 SimSet Method

Description

Set properties of each object of the simulator. Operate the robot motion, objects, and simulator settings.

Syntax

Sub SimSet (Object As String, Property As SpelSimProps, Value As Boolean)
Sub SimSet (RobotName As String, HandName As String, Property As SpelSimProps, Value As Boolean)
Sub SimSet (Object As String, Property As SpelSimProps, Value As Integer)
Sub SimSet (RobotName As String, HandName As String, Property As SpelSimProps, Value As Integer)
Sub SimSet (Object As String, Property As SpelSimProps, Value As Double)
Sub SimSet (RobotName As String, HandName As String, Property As SpelSimProps, Value As Double)
Sub SimSet (Object As String, Property As SpelSimProps, Value As String)
Sub SimSet (RobotName As String, HandName As String, Property As SpelSimProps, Value As String)

Parameters

- Object
  String variable that indicates the object name to acquire the property values for.
- RobotName
  Sting variable that indicates the robot name to which the hand specified in “Hand” is mounted.
- HandName
  A string variable indicating the hand name for which to set the property value.
- Property
  The name of the property to set the new value for.
- Value
  Formula for new value

Remarks

Use this method to set properties of each object of the simulator. Also, use the command to change the robot motion, objects, and simulator settings.

You cannot specify SpelSimProps.Type for the property.

For details of each property, refer to the following manual.
Epson RC+ 8.0 SPEL+ Language Reference - SimSet

See Also

SimGet Method

SimSet Example

VB Example:

```
m_spel.SimSet ("SBox_1", SpelSimProps.PositionX, 100.0)
```

C# Example:

```
m_spel.SimSet("SBox_1", SpelSimProps.PositionX, 100.0);
```

15.3.17.11 SimSetParent Method

Description

Sets an object operation.

Syntax

Sub SimSetParent (Object As String)
Sub SimSetParent (Object As String, ParentObject As String)

Parameters

- Object
  A string variable indicating the object name to set the parent object for.
- ParentObject
  A string variable indicating the parent object name.

Remarks

Sets the object specified by “ParentObject” as parent object for the object specified by “Object”. “ParentObject” can be omitted. In this case, the object specified by “Object” will be parenet object. For example, if the object specified by “Object” is a child object of another object, the setting as a child object will be canceled.
If the object specified by “Object” is a part or an object set as an arm installation tool, you cannot specify parent object.
For the objects that can be specified as SetParent, refer to the following manual.
Epson RC+ 8.0 SPEL+ Language Reference - SimSet

For camera objects, SetParent can be used only for objects that are set as fixed cameras.

See Also

SimSet Method

SimSetParent Example

VB Example:

```
m_spel.SimSetParent ("SBox_1")
```

C# Example:

```
m_spel.SimSetParent ("SBox_1");
```

15.3.17.12 SimSetPick Method

Description

Picks an object using the specified robot.

Syntax

Sub SimSetPick (RobotName As String, Object As String)
Sub SimSetPick (RobotName As String, Object As String, ToolNumber As Integer)

Parameters

- RobotName
  A string variable indicating the name of the robot to pick.
- Object
  A string variable indicating the picked object name.
- ToolNumber
  Expression indicating the tool number to use when picking

Remarks

The robot specified by RobotName grasps the object specified by Object. Grasped object is registered as the part of the robot. Also, if any tool number is specified by ToolNumber, you can operate grasped motion by using the specified tool number. If the ToolNumber is omitted, use Tool 0 to operate grasped motion.
You cannot grasp the object that is already registered as the part or set as an arm installation tool. Also, you cannot grasp the camera..

For more details, refer to the following manual.
Epson RC+ 8.0 SPEL+ Language Reference - SimSet

See Also

SimGet Method, **SimSet Method**, **SimSetPlace Method**

SimSetPick Example

VB Example:

```
m_spel.SimSetPick ("Robot1", "SBox_1", 1)
```

C# Example:

```
m_spel.SimSetPick ("Robot1", "SBox_1", 1);
```

15.3.17.13 SimSetPlace Method

Description

Places an object using the specified robot.

Syntax

Sub SimSetPlace (RobotName As String, Object As String)

Parameters

- RobotName
  A string variable indicating the name of the robot to place.
- Object
  A string variable indicating the placed object name.

Remarks

The robot specified by RobotName places the object specified by Object. The placed object is deregistered as part of the robot. You cannot place an object that has already been deregistered as a part.

For more details, refer to the following manual and section:
Epson RC+ 8.0 SPEL+ Language Reference - SimSet

See Also

SimGet Method, **SimSet Method**, **SimSetPick Method**

SimSetPlace Example

VB Example:

```
m_spel.SimSetPlace ("Robot1", "SBox_1")
```

C# Example:

```
m_spel.SimSetPlace ("Robot1", "SBox_1");
```

15.3.17.14 Shutdown Method

Description

Shutdown or restart Windows.

Syntax

Sub Shutdown (Mode As SpelShutdownMode)

Parameters

- Mode
  0 = Shutdown Windows. 1 = Restart Windows.

See Also

Reset Method

Shutdown Example

VB Example:

```
' Restart Windows
m_spel.Shutdown(1)
```

C# Example:

```
// Restart Windows
m_spel.Shutdown(1);
```

15.3.17.15 Slock Method

Description

Release the free joint state for the specified robot axes.

Syntax

Sub SLock ()
Sub SLock (ParamArray Axes() As Integer)

Parameters

- Axes
  An integer parameter array indicating the robot axis elements to lock. You can specify axis numbers from 1 to 9.

See Also

SFree Method

SLock Example

VB Example:

```
' Release the free joint state of J1 and J2.
m_spel.SLock(1, 2)
```

C# Example:

```
// Release the free joint state of J1 and J2.
m_spel.SLock(1, 2);
```

15.3.17.16 Speed Method

Description

Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.

Syntax

Sub Speed (PointToPointSpeed As Integer [, JumpDepartSpeed As Integer] [, JumpApproSpeed As Integer])

Parameters

- PointToPointSpeed
  Specifies the arm speed for use with the PTP motion Go, Jump and Pulse.
- JumpDepartSpeed
  An integer between 1 and 100 indicating the depart speed for the Jump instruction.
- JumpApproSpeed
  An integer between 1 and 100 indicating the approach speed for the Jump instruction.

See Also

Accel Method, **Jump Method**, **Go Method**

Speed Example

VB Example:

```
m_spel.Speed(50)
```

C# Example:

```
m_spel.Speed(50);
```

15.3.17.17 SpeedR Method

Description

Specifies the tool rotation speed when ROT is used.

Syntax

Sub SpeedR (RotationSpeed As Single)

Parameters

- RotationSpeed
  Specifies the tool posture change speed as a real number greater than or equal to 0.1 (unit: deg/sec)

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Arc Method, **Arc3 Method**, **BMove Method**, **Jump3CP Method**, **TMove Method**, **SetPoint Method**

SpeedR Example

VB Example:

```
m_spel.SpeedR(100)
```

C# Example:

```
m_spel.SpeedR(100);
```

15.3.17.18 SpeedS Method

Description

Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.

Syntax

Sub SpeedS (LinearSpeed As Single [, JumpDepartSpeed As Single] [, JumpApproSpeed As Single])

Parameters

- LinearSpeed
  Specifies the arm speed to use with the Jump3CP, Move, Arc, and CVMove Continuous Path instructions.
- JumpDepartSpeed
  A real number or formula indicating the depart speed for Jump and Jump3CP. (unit: mm/sec)
- JumpApproSpeed
  A real number or formula defining the approach speed for Jump and Jump3CP. (unit: mm/sec)

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelS Method, **Jump3CP Method**, **Move Method**, **TMove Method**, **SetPoint Method**

SpeedS Example

VB Example:

```
m_spel.SpeedS(500)
```

C# Example:

```
m_spel.SpeedS(500);
```

15.3.17.19 Start Method

Description

Start one SPEL+ program.

Syntax

Sub Start (ProgramNumber As Integer)

Parameters

- ProgramNumber
  The program number to start. The correspondence with the 64 main functions incorporated in SPEL+ is shown in the table below. The range is 0 to 63.

Program Number	SPEL+ Function Name
0	main
1	main1
2	main2
3	main3
4	main4
5	main5
...	...
63	main63

Remarks

When Start is executed, control will return immediately to the calling program. You cannot start a program that is already running. Note that Start causes global variables in the controller to be cleared and default robot points to be loaded.

See Also

Continue Method, **Pause Method**, **Stop Method**, **Xqt Method**

Start Example

VB Example:

```
Sub btnStart_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnStart.Click

    m_spel.Start(0)
End Sub
```

C# Example:

```
void btnStart_Click(object sender, EventArgs e)
{
    m_spel.Start(0);
}
```

15.3.17.20 StartBGTask Method

Description

Start one SPEL+ task as a background task.

Syntax

Sub StartBGTask (FuncName As String)

Parameters

- FuncName
  The name of the function to be executed.

Remarks

Use StartBGTask to start a Spel+ background task in the Controller. Background tasks must be enabled in the Controller.

Note that BGMain automatically starts when the Controller switches to auto mode, so normally StartBGTask is not required. StartBGTask is provided in case you need to stop all tasks, then start background tasks again.

See Also

Call Method, **Start Method**, **Stop Method**, **Xqt Method**

StartBGTask Example

VB Example:

```
' Stop all tasks, including background tasks
m_spel.Stop(SpelStopType.StopAllTasks)
…
m_spel.RebuildProject()

’ Restart the main background task
m_spel.StartBGTask("BGMain")
```

C# Example:

```
// Stop all tasks, including background tasks
m_spel.Stop(SpelStopType.StopAllTasks);
…
m_spel.RebuildProject();

// Restart the main background task
m_spel.StartBGTask("BGMain");
```

15.3.17.21 Start Method

Description

Returns the Controller status.

Syntax

Function Stat (Address As Integer) As Integer

Parameters

- Address
  Specifies the address indicating the status of the Controller. (Integer from 0 to 2)

Return Value

Returns 4 byte value representing the status of the Controller. (See the table below.)

Address	Bit		The status of the Controller while the bit is on
0	0-15	&H1- &H8000	Task 1 to 16 are being executed (Xqt) or Halt state
	16	&H10000	The task is being executed
	17	&H20000	Pause state
	18	&H40000	Error state
	19	&H80000	TEACH mode
	20	&H100000	Emergency stop state
	21	&H200000	Power Low mode
	22	&H400000	The safeguard is open
	23	&H800000	Enable switch is open
	24	&H1000000	Undefined
	25	&H2000000	Undefined
	26	&H4000000	Test mode
	27	&H8000000	T2 mode state
	28-31		Undefined
1	0	&H1	Log of Stop above target position upon satisfaction of condition in Jump...Sense statement. (This log is erased when another Jump statement is executed).
	1	&H2	Log of stop at intermediate travel position upon satisfaction of condition in Go/Jump/Move...Till statement. (This log is erased when another Go/Jump/Move...Till statement is executed)
	2	&H4	Undefined
	3	&H8	The log of motion stop in progress if Trap statement is detected.
	4	&H10	Motor On state
	5	&H20	Home position

Address	Bit		The status of the Controller while the bit is on
	6	&H40	Power Low mode
	7	&H80	Undefined
	8	&H100	Joint #4 is engaged.
	9	&H200	Joint #3 is engaged.
	10	&H400	Joint #2 is engaged.
	11	&H800	Joint #1 is engaged.
	12	&H1000	Joint #6 is engaged.
	13	&H2000	Joint #5 is engaged.
	14	&H4000	T axis is engaged.
	15	&H8000	S axis is engaged.
	16	&H10000	Joint #7 is engaged.
	17-31		Undefined
2	0-15	&H1- &H8000	Task 17 to 32 are being executed (Xqt) or in Halt state

See Also

EStopOn Property, **PauseOn Property**, **SafetyOn Property**

Stat Example

VB Example:

```
Dim ctr_stat As Integer
ctr_stat = m_spel.Stat(0)
```

C# Example:

```
int ctr_stat;
ctr_stat = m_spel.Stat(0);
```

15.3.17.22 Stop Method

Description

Stops all normal SPEL+ tasks running in the Controller and optionally stop all background tasks.

Syntax

Sub Stop ()
Sub Stop (SpelStopType StopType)

Parameters

- StopType
  Specifies whether to stop only normal tasks (StopNormalTasks) or all tasks (StopAllTasks). If omitted, StopNormalTasks is specified.

> ⚠ Note
>
> If the Stop method is executed when ResetAbortEnabled is True, the error 10101 occurs when executing Start or Reset methods.
>
> To release the error, execute ResetAbort method after executing Stop method.

See Also

Continue Method, **Pause Method**, **ResetAbort Method**, **ResetAbortEnabled Property**, **Start Method**, **SpelStopType Enumeration**

Stop Example

VB Example:

```
Sub btnStop_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnStop.Click

    m_spel.Stop()
End Sub
```

C# Example:

```
void btnStop_Click(object sender, EventArgs e)
{
    m_spel.Stop();
}
```

15.3.17.23 Sw Method

Description

Returns the selected input bit status.

Syntax

Function Sw (BitNumber As Integer) As Boolean
Function Sw (Label As String) As Boolean

Parameters

- BitNumber
  An integer or formula indicating the I/O input bits.
- Label
  A string expression indicating the input bit label.

Return Value

True if the specified input bit is on, False if not.

See Also

In Method, **InBCD Method**, **MemSw Method**, **Off Method**, **On Method**, **Oport Method**

Sw Example

VB Example:

```
If m_spel.Sw(1) Then
    m_spel.On(2)
End If
```

C# Example:

```
if (m_spel.Sw(1))
    m_spel.On(2);
```

15.3.18 T

15.3.18.1 TargetOK Method

Description

Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.

Syntax

Function TargetOK (PointNumber As Integer) As Boolean
Function TargetOK (Point As SpelPoint) As Boolean
Function TargetOK (PointExpr As String) As Boolean

Parameters

Each syntax has one parameter that specifies the target point to check.

- PointNumber
  Specifies the end point using the point number of a previously taught point in the Controller's point memory for the current robot.
  Point Specifies the end point using the SpelPoint data type.
- PointExpr
  Specifies the end point using a string expression.

Return Value

True if the target can be moved to from the current position, False if not.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Go Method, **Jump Method**, **Move Method**, **TGo Method**, **TMove Method**, **SetPoint Method**

TargetOK Example

VB Example:

```
If m_spel.TargetOK("P1 /F") Then
    m_spel.Go("P1 /F")
End If
```

C# Example:

```
if (m_spel.TargetOK("P1 /F"))
    m_spel.Go("P1 /F");
```

15.3.18.2 TasksExecuting Method

Description

Returns True if any SPEL+ tasks are executing.

Syntax

Function TasksExecuting () As Boolean

Return Value

True if any SPEL+ tasks are executing, False if not.

See Also

TaskState Method, **Xqt Method**

TasksExecuting Example

VB Example:

```
tasksRunning = m_spel.TasksExecuting()
```

C# Example:

```
tasksRunning = m_spel.TasksExecuting();
```

15.3.18.3 TaskState Method

Description

Returns the status of a task.

Syntax

Function TaskState (TaskNumber As Integer) As SpelTaskState
Function TaskState (TaskName As String) As SpelTaskState

Parameters

- TaskNumber
  Task number for which to return the execution status.
- TaskName
  A string expression indicating the task name.

Return Value

A SpelTaskState value.

See Also

TasksExecuting Method, **Xqt Method**

TaskState Example

VB Example:

```
Dim taskState As SpelTaskState
taskState = m_spel.TaskState(2)
```

C# Example:

```
SpelTaskState taskState;
taskState = m_spel.TaskState(2);
```

15.3.18.4 TeachPoint Method

Description

Runs a dialog that allows an operator to jog and teach one point.

Syntax

Function TeachPoint (PointFile As String, PointNumber As Integer, Prompt As String) As Boolean
Function TeachPoint (PointFile As String, PointName As String, Prompt As String) As Boolean
Function TeachPoint (PointFile As String, PointNumber As Integer, Prompt As String, Parent As Form) As Boolean
Function TeachPoint (PointFile As String, PointName As String, Prompt As String, Parent As Form) As Boolean

Parameters

- PointFile
  A character string indicating the point file name.
- PointNumber
  Point number to be taught
- PointName
  A character string indicating the point label
- Prompt
  A character string indicating the instructional text that is displayed at the bottom of the teach dialog.
- Parent
  Optional. A .NET form that is the parent of the window.

Return Value

Returns True if the operator clicked the Teach button, False if the operator clicked Cancel.

Remarks

Use TeachPoints to allow an operator to teach one robot point in the Controller. When TeachPoints is executed, the point file is loaded in the Controller. When the Teach button is clicked, the point is taught in the Controller and the point file is saved on the Controller.

TeachPoint Example

VB Example:

```
Sub btnTeachPick_Click( _
       ByVal sender As System.Object, _
       ByVal e As System.EventArgs) _
       Handles btnTeachPick.Click

    Dim sts As Boolean
    Dim prompt As String

    prompt = "Jog to Pick position and click Teach"
    sts = m_spel.TeachPoint("points.pts", 1, prompt)
End Sub
```

C# Example:

```
void btnTeachPick_Click(object sender, EventArgs e){
    bool sts;
    string prompt;

    prompt = "Jog to Pick position and click Teach";
    sts = m_spel.TeachPoint("points.pts", 1, prompt);

}
```

15.3.18.5 Till Method

Description

Specifies event condition that, if satisfied, completes the motion command (Jump, Go, Move, etc.) in progress by decelerating and stopping the robot at an intermediate position.

Syntax

Sub Till (Condition As String) As Boolean

Parameters

- Condition
  Specifies the I/O condition. For details see Till Statement in the SPEL+ Language Reference manual.

See Also

Go Method, **Jump Method**, **JS Method**, **Sense Method**, **TillOn Method**

Till Example

VB Example:

```
With m_spel
    .Till("Sw(1) = On")
    .Go("P1 TILL")
End With
```

C# Example:

```
m_spel.Till("Sw(1) = On");
m_spel.Go("P1 TILL");
```

15.3.18.6 TillOn Method

Description

Returns True if a stop has occurred from a till condition during the last Go/Jump/Move statement.

Syntax

Function TillOn () As Boolean

Return Value

True if the robot stopped due to a Till condition, False if not.

Remarks

Use TillOn to check if the Till condition turned on during the last motion command using Till.
TillOn is equivalent to
((Stat(1) And 2) <> 0)

See Also

Jump Method, **Till Method**

TillOn Example

VB Example:

```
If m_spel.TillOn() Then
    m_spel.Jump(2)
End If
```

C# Example:

```
if (m_spel.TillOn())
    m_spel.Jump(2);
```

15.3.18.7 TGo Method

Description

Executes Point to Point relative motion, in the current tool coordinate system.

Syntax

Sub TGo (PointNumber As Integer)
Sub TGo (Point As SpelPoint)
Sub TGo (Point As SpelPoint, AttribExpr As String)
Sub TGo (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the TGo motion. This is the final position at the end of the point to point motion.

- PointNumber
  Specifies the end point using the point number of a previously taught point in the Controller's point memory for the current robot.
- Point
  Specifies the end point using the SpelPoint data type.
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain CP, SYNC, Till or Find search expressions, and parallel processing statements.
- PointExpr
  Specifies the end point using a string expression.
  It can contain CP, SYNC, Till or Find search expressions, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Accel Method, **Speed Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TMove Method**, **Find Method**, **Till Method**, **SetPoint Method**

TGo Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.TGo(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.TGo(pt)

' Points specified using expressions
m_spel.TGo("P0")
m_spel.TGo("XY(0, 0, -20, 0)")

' Using parallel processing
m_spel.TGo("P1 \!D50; On 1; D90; Off 1\!")

' Points specified using labels
m_spel.TGo("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.TGo(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.TGo(pt);

// Points specified using expressions
m_spel.TGo("P0 /L /2");
m_spel.TGo("P1 :Z(-20)");

// Using a parallel processing
m_spel.TGo("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.TGo("pick");
```

15.3.18.8 TLClr Method

Description

Clears (undefines) a tool coordinate system.

Syntax

Sub TLClr (ToolNumber As Integer)

Parameters

- ToolNumber
  An integer indicating which of the tool coordinates to clear (undefine). (Tool 0 is the default tool and cannot be cleared.)

See Also

Tool Method, **TLDef Method**

TLClr Example

VB Example:

```
m_spel. TLClr(1)
```

C# Example:

```
m_spel. TLClr(1);
```

15.3.18.9 TLDef Method

Description

Returns tool definition status.

Syntax

Function TLDef (ToolNumber As Integer) As Boolean

Parameters

- ToolNumber
  An integer indicating which tool status to return.

Return Value

True if the specified tool is defined, False if not.

See Also

Tool Method, **TLClr Method**

TLDef Example

VB Example:

```
m_spel.TLDef(1)
```

C# Example:

```
m_spel.TLDef(1);
```

15.3.18.10 TLSet Method

Description

Defines a tool coordinate system.

Syntax

Sub TLset (ToolNumber As Integer , Point As SpelPoint)

Sub TLset (ToolNumber As Integer, XCoord As Single, YCoord As Single, ZCoord As Single, UCoord As Single, VCoord As Single, WCoord As Single)

Parameters

- ToolNumber
  An integer expression (1 to 15) indicating which of the 15 tools to define. Tool 0 is the default tool. (Tool 0 is the default tool and cannot be modified.)
- Point
  A SpelPoint indicating the point data.
- XCoord
  The tool coordinate system origin X coordinate.
- YCoord
  The tool coordinate system origin Y coordinate.
- ZCoord
  The tool coordinate system origin Z coordinate.
- UCoord
  The tool coordinate system rotation about the Z axis.
- VCoord
  The tool coordinate system rotation about the Y axis.
- WCoord
  The tool coordinate system rotation about the X axis.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Arm Method, **ArmSet Method**, **GetTool Method**, **Tool Method**, **SetPoint Method**

TLSet Example

VB Example:

```
m_spel.TLSet(1, .5, 4.3, 0, 0, 0, 0)
```

C# Example:

```
m_spel.TLSet(1, .5, 4.3, 0, 0, 0, 0);
```

15.3.18.11 TMove Method

Description

Executes linear interpolation relative motion, in the current tool coordinate system

Syntax

Sub TMove (PointNumber As Integer)
Sub TMove (Point As SpelPoint)
Sub TMove (Point As SpelPoint, AttribExpr As String)
Sub TMove (PointExpr As String)

Parameters

Each syntax has one parameter that specifies the end point which the arm travels to during the TMove motion. This is the final position at the end of the linear interpolated motion.

- PointNumber
  Specifies the target point using a previously taught point number from the current robot point memory stored in the controller
- Point
  Specifies the target point using the SpelPoint data type
- AttribExpr
  Specifies the end point attributes using a string expression.
  It can contain ROT, CP, SYNC, Till or Find search expressions, and parallel processing statements.
- PointExpr
  Specifies the target point using a string expression
  It can contain ROT, CP, SYNC, Till or Find search expressions, and parallel processing statements.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

AccelR Method, **AccelS Method**, **SpeedR Method**, **SpeedS Method**, **Arc Method**, **Arc3 Method**, **CVMove Method**, **Go Method**, **Jump Method**, **Jump3 Method**, **Jump3CP Method**, **Move Method**, **BGo Method**, **BMove Method**, **TGo Method**, **Find Method**, **Till Method**, **SetPoint Method**

TMove Example

VB Example:

```
' Points specified using point numbers
m_spel.Tool(1)
m_spel.TMove(100)

' Points specified using SpelPoint
Dim pt As SpelPoint
pt = m_spel.GetPoint("P*")
pt.X = 125.5
m_spel.TMove(pt)

' Points specified using expressions
m_spel.TMove("P0")
m_spel.TMove("XY(0, 0, -20, 0)")

' Using parallel processing
m_spel.TMove("P1 \!D50; On 1; D90; Off 1\!")

' Points specified using labels
m_spel.TMove("pick")
```

C# Example:

```
// Point specified using point number
m_spel.Tool(1);
m_spel.TMove(100);

// Points specified using SpelPoint
SpelPoint pt;
pt = m_spel.GetPoint("P0");
pt.X = 125.5;
m_spel.TMove(pt);

// Points specified using expressions
m_spel.TMove ("P0");
m_spel.TMove ("XY(0, 0, -20, 0)");

// Using a parallel processing
m_spel.TMove("P1 \!D50; On 1; D90; Off 1\!");

// Points specified using labels
m_spel.TMove("pick");
```

15.3.18.12 Tool Method

Description

Selects the current robot tool.

Syntax

Sub Tool (ToolNumber As Integer)

Parameters

- ToolNumber
  Sixteen tool definitions (0 to 15) to be used for the next operation command

See Also

TLSet Method, **Arm Method**, **TGo Method**, **TMove Method**

Tool Example

VB Example:

```
m_spel.Tool(1)
m_spel.TGo(100)
```

C# Example:

```
m_spel.Tool(1);
m_spel.TGo(100);
```

15.3.18.13 TrapStop Method

Description

Returns True if the current robot was stopped by a trap during the previous motion command.

Syntax

Function TrapStop () As Boolean

Return Value

True if the robot was stopped by a trap, False if not.

See Also

EStopOn Property, **ErrorOn Property**

TrapStop Example

VB Example:

```
If m_spel.TrapStop() Then
    MsgBox "Robot stopped by Trap"
End If
```

C# Example:

```
if (m_spel.TrapStop())
    MessageBox.Show("Robot stopped by trap");
```

15.3.18.14 TW Method

Description

Returns the status of the WAIT condition and WAIT timer interval.

Syntax

Function TW () As Boolean

Return Value

True if a timeout occurred, False if not.

See Also

WaitMem Method, **WaitSw Method**

TW Example

VB Example:

```
Const PartPresent = 1
m_spel.WaitSw(PartPresent, True, 5)
If m_spel.TW() Then
    MsgBox "Part present time out occurred"
End If
```

C# Example:

```
const int PartPresent = 1;
m_spel.WaitSw(PartPresent, True, 5);
if (m_spel.TW())
    MessageBox.Show("Part present time out occurred");
```

15.3.19 U

15.3.19.1 UserHasRight Method

Description

Returns whether the currently logged in user has the specified right.

Syntax

Function UserHasRight (SpelUserRights Right) As Boolean

Parameters

- Right
  The right you want to check for the currently logged-in user.

Return Value

True if the user has the specified right, False if not.

See Also

Login Method, **GetCurrentUser Method**

UserHasRight Example

VB Example:

```
Dim hasRight As Boolean
hasRight = m_spel.UserHasRight(SpelUserRights.EditPoints)
```

C# Example:

```
bool hasRight;
hasRight = m_spel.UserHasRight(SpelUserRights.EditPoints);
```

15.3.20 V

15.3.20.1 VCal Method

Description

This command allows you to execute a vision calibration cycle.

Syntax

Sub VCal (CalibName As String)
Sub VCal (CalibName As String, ByRef Status As Integer)
Sub VCal (CalibName As String, Parent As Form)
Sub VCal (CalibName As String, Parent As Form, ByRef Status As Integer)

Parameters

- CalibName
  A string expression indicating the calibration scheme name for the current project.
- Status
  Optional. An integer variable for which to return the calibration status. 0: Unsuccessful, 1: Successful
- Parent
  Optional. A .NET form that is the parent of the window.

Remarks

When you execute the VCal method, the robot will move. You should verify that the operator is ready before executing VCal.

VCal only executes the calibration cycle. It does not allow you to teach points. Use VCalPoints to teach points. Also, you must first set up a calibration in Epson RC+ 8.0. See your Vision Guide manuals for details.

Use the Status parameter to check if the calibration was successful. If the calibration property ShowConfirmation is True, the confirmation dialog box is displayed. When the operator clicks the [OK] button, Status returns 1: Successful.

See Also

VCalPoints Method

VCal Example

VB Example:

```
Dim status As Integer
m_spel.VCal("CAMCAL1", status)
```

C# Example:

```
int status;
m_spel.VCal("CAMCAL1", status);
```

15.3.20.2 VCalPoints Method

Description

This command enables you to teach vision calibration points.

Syntax

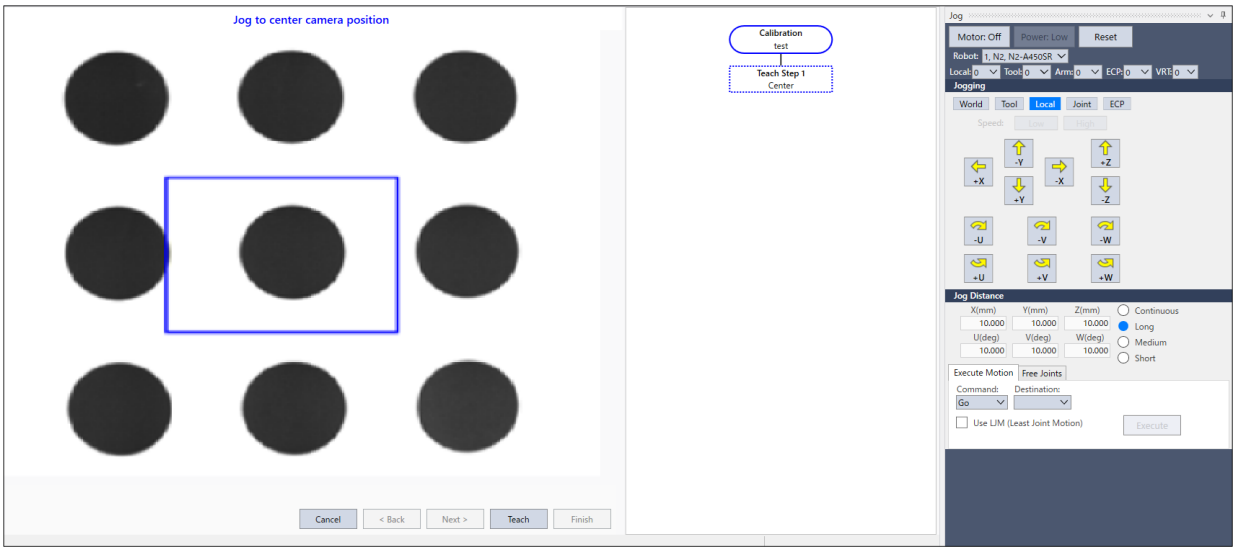
Sub VCalPoints (CalibName As String)
Sub VCalPoints (CalibName As String, ByRef Status As Integer)
Sub VCalPoints (CalibName As String, Parent As Form)
Sub VCalPoints (CalibName As String, Parent As Form, ByRef Status As Integer)

Parameters

- CalibName
  A string expression indicating the calibration scheme name for the current project.
- Status
  Optional. An integer variable that receives the status of the point teaching. 0: Unsuccessful, 1: Successful
- Parent
  Optional. A .NET form that is the parent of the window.

Remarks

When you execute the VCalPoints command, the Teach Calibration Points dialog is opened. When the [Finish] button is clicked, the calibration data is automatically saved.
You must have already created the calibration scheme from Epson RC+ 8.0.



Use the Status parameter to check if the point teaching was successful. When all points are taught and the [Finish] button is clicked, Status returns 1: Successful.

See Also

VCal Method

VCalPoints Example

VB Example:

```
Dim status As Integer
m_spel.VCalPoints("CAMCAL1", status)
```

C# Example:

```
int status;
m_spel.VCalPoints("CAMCAL1", out status);
```

15.3.20.3 VCls Method

Description

Clears vision graphics.

Syntax

Sub VCls ()

Remarks

Use the VCls method to clear the vision screen.

See Also

VRun Method

VCls Example

VB Example:

```
m_spel.VCls()
```

C# Example:

```
m_spel.VCls();
```

15.3.20.4 VCreateCalibration Method

Description

Creates a new vision calibration in the current project.

Syntax

Sub VCreateCalibration (CameraNumber As Integer, CalibName As String)
Sub VCreateCalibration (CameraNumber As Integer, CalibName As String,CopyCalibName As String)

Parameters

- CameraNumber
  An integer expression indicating the number of the camera to calibrate.
- CalibName
  A string expression indicating the vision calibration name to be created.
- CopyCalibName
  Optional. A string expression indicating the vision calibration name to be copied.

See Also

VCreateObject Method, **VCreateSequence Method**, **VDeleteCalibration Method**

VCreateCalibration Example

VB Example:

```
m_spel.VCreateCalibration(1, "mycal")
```

C# Example:

```
m_spel.VCreateCalibration(1, "mycal");
```

15.3.20.5 VCreateObject Method

Description

Creates a vision object in the current project.

Syntax

Sub VCreateObject (Sequence As String, ObjectName As String, ObjectType As SpelVisionObjectTypes)

Parameters

- Sequence
  A string expression indicating the vision sequence name in the current project.
- ObjectName
  A string expression indicating the name of the object to be created in the sequence.
- ObjectType
  A number indicating the vision object type (constants listed below can also be used)

Object Type	Constant	Value
Correlation	Correlation	1
Blob	Blob	2
Edge	Edge	3
Polar	Polar	4
Line	Line	5
Point	Point	6
Frame	Frame	7
ImageOp	ImageOp	8
Ocr	Ocr	9
CodeReader	CodeReader	10
Geometric	Geometric	11
Color Match	ColorMatch	14
Line Finder	LineFinder	15
Arc Finder	ArcFinder	16
Defect Finder	DefectFinder	17
Line Inspector	LineInspector	18
Arc Inspector	ArcInspector	19
Box Finder	BoxFinder	20
Corner Finder	CornerFinder	21
Contour	Contour	22
Text	Text	23
Decision	Decision	26

Object Type	Constant	Value
Coordinates	Coordinates	27

See Also

VCreateSequence Method, VDeleteObject Method, VDeleteSequence Method

VCreateObject Example

VB Example:

```
m_spel.VCreateObject("myseq", "myblob", SpelVisionObjectTypes.Blob)
```

C# Example:

```
m_spel.VCreateObject("myseq", "myblob", SpelVisionObjectTypes.Blob);
```

15.3.20.6 VCreateSequence Method

Description

Creates a new vision sequence in the current project.

Syntax

Sub VCreateSequence (CameraNumber As Integer, SequenceName As String)
Sub VCreateSequence (CameraNumber As Integer, SequenceName As String,
CopySequenceName As String)

Parameters

- CameraNumber
  An integer expression indicating the camera number to use.
- SequenceName
  A string expression indicating the vision sequence name to be created.
- CopySequenceName
  Optional. A string expression indicating the vision calibration name to be copied.

See Also

VCreateObject Method, **VDeleteObject Method**, **VDeleteSequence Method**

VCreateSequence Example

VB Example:

```
m_spel.VCreateSequence(1, "myseq")
```

C# Example:

```
m_spel.VCreateSequence(1, "myseq");
```

15.3.20.7 VDefArm Method

Description

Calculates an arm set value of a mobile camera using a feature point detectable by the vision system.
Note: Robot operates automatically based on the detection results of the target. Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during the arm set.

Syntax

Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double)
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double, Parent As Form)
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, ShowWarning As SpelVDefShowWarning
Sub VDefArm (ArmNumber As Integer, ArmDefType As SpelArmDefType, ArmDefMode As SpelArmDefMode, Sequence As String, Rotation As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, ShowWarning As SpelVDefShowWarning,Parent As Form)

Parameters

- ArmNumber
  An integer expression (1 to 15) indicating the arm number on which to perform arm set.
- ArmDefType
  An integer expression indicating the arm type.
  J2Camera: Calculates a center of mobile J2 camera image.
- ArmDefMode
  An integer expression indicating the arm set mode.
  Rough: A mode to run a rough arm set. Robot will move with setting accuracy of 1 mm as a target. Robot motion will be small.
  Fine: A mode to run a fine arm set. Robot will move largely with arm orientation change and provide arm set with more high accuracy.
- Sequence
  A string expression indicating the vision sequence name for the current project.
- Rotation
  A real number expression that indicates the rotation angle (degrees) when using the coarse tool set.
  Value range: 0 to 45
- TargetTolerance
  A real number expression indicating the pixel distance at which the vision detection result is considered to match the target position.
  Value range: 0 to 3 pixels
- Parent
  Optional. A .NET form that is the parent of the window.
- RobotSpeed
  Optional. Integer variable indicating the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
- RobotAccel
  Optional. Integer variable indicating the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.
- ShowWarning
  Optional. An integer variable that sets whether to display a warning when ArmSetMode is Fine. Always : Always display

DependsOnSpeed : Display when either RobotSpeed or RobotAccel is larger than 5 None : Do not display If omitted, set to “DependsOnSpeed”.

See Also

VDefGetMotionRange Method, **VDefLocal Method**, **VDefSetMotionRange Method**, **VDefTool Method**, **VGoCenter Method**

VDefArm Example

VB Example:

```
m_spel.VDefArm(1, SpelArmDefType.J2Camera, SpelArmDefMode.Rough, "myseq", 5, 1)
```

C# Example:

```
m_spel.VDefArm(1, SpelArmDefType.J2Camera, SpelArmDefMode.Rough, "myseq", 5, 1);
```

15.3.20.8 VDefGetMotionRange Method

Description

Acquires values of the motion range limited by VDefTool, VDefArm, VDefLocal, and VGoCenter.

Syntax

Sub VDefGetMotionRange(ByRef MaxMoveDist As Double, ByRef MaxPoseDiffAngle As Double, ByRef LjmMode As Integer)

Parameters

- MaxMoveDist
  A real number variable indicating the maximum travel distance.
  If 0 is specified, the range is not limited. (0 to 500. Default: 200)
  VDefTool, VDefArm, VDefLocal, and VGoCenter are used to limit the range.
- MaxPoseDiffAngle
  A real number variable indicating the maximum displacement angle (degrees) of the hand orientation (UVW).
  If 0 is specified, the range is not limited.
  It only affects VDefLocal. (0 to 180. Default: 45 degrees)
- LjmMode
  A real number variable indicating the LJM mode.

See Also

VDefTool Method, **VDefArm Method**, **VDefLocal Method**, **VGoCenter Method**, **VDefSetMotionRange Method**

VDefGetMotionRange Example

VB Example:

```
Dim maxMoveDist As Double
Dim maxPoseDiffAngle As Double
Dim ljmMode As Integer
m_spel.VDefGetMotionRange(maxMoveDist, maxPoseDiffAngle, ljmMode)
```

C# Example:

```
double maxMoveDist, maxPoseDiffAngle;
int ljmMode;
m_spel.VDefGetMotionRange(out maxMoveDist, out maxPoseDiffAngle, out ljmMode);
```

15.3.20.9 VDefLocal Method

Description

Detects a calibration plate placed on a work plane by a mobile camera, and defines local coordinates parallel to the work plane. It also detects user's workpiece at the tool end by a fixed camera and defines a local plane which is parallel to a fixed camera sensor.
Note: Robot operates automatically based on the detection results of the target. Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during the local coordinate setting.

Syntax

Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType, CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double, CameraTool As Integer, RefPoint As SpelPoint)
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType, CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double, CameraTool As Integer, RefPoint As SpelPoint, Parent As Form)
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType, CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double, CameraTool As Integer, RefPoint As SpelPoint, RobotSpeed As Integer, RobotAccel As Integer)
Sub VDefLocal(LocalNumber As Integer, LocalDefType As SpelLocalDefType, CalPlateType As SpelCalPlateType, Sequence As String, TargetTolerance As Double, CameraTool As Integer, RefPoint As SpelPoint, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)

Parameters

- LocalNumber
  An integer indicating the tool number to set the local coordinates for (1 to 15).
- LocalDefType
  An integer indicating the local type.
- J5Camera:
  Sets the local coordinates parallel to the calibration plate using the mobile J5 camera.
- J6Camera:
  Sets the local coordinates parallel to the calibration plate using the mobile J6 camera.
- FixedUpwardCamera:
  Sets the local coordinates parallel to the image sensor using a fixed upward-facing camera.
- FixedDownwardCamera:
  Sets the local coordinates parallel to the image sensor using a fixed downward-facing camera.
- CalPlateType
  An integer indicating the calibration plate type.
- Large:
  Large calibration plate
- Medium:
  Medium calibration plate
- Small:
  Small calibration plate
- XSmall:
  Extra small calibration plate
- Sequence
  A string expression indicating the vision sequence name for the current project.
  When using the mobile camera, this is a vision sequence to take a picture of the calibration plate.
  When using the fixed camera, this is a vision sequence to detect a feature point at tool end, such as user's workpiece.

- TargetTolerance
  A real number indicating the threshold for determining the scale match.
- CameraTool
  In the case of a fixed camera, this specifies the tool number that holds a tool offset of the detection target. To perform auto calibration, specify -1.
  Mobile J6 camera: If auto calibration has been executed, specify a tool number of mobile camera. To perform auto calibration, specify -1.
  Mobile J5 camera: Setting of this option is ignored.
- RefPoint Point number which a local plane parallel to a work plane passes.
  This point is used to specify local plane height.
- Parent
  Optional. A .NET form that is the parent of the window.
- RobotSpeed
  Optional. Integer variable indicating the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
- RobotAccel
  Optional. Integer variable indicating the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.

See Also

VDefArm Method, **VDefGetMotionRange Method**, **VDefSetMotionRange Method**, **VDefTool Method**, **VGoCenter Method**

VDefLocal Example

VB Example:

```
Dim p2 = m_spel.GetPoint("P2")
m_spel.VDefLocal(1, SpelLocalDefType.J6Camera, SpelCalPlateType.Large, "myseq",
1.0, 1, p2)
```

C# Example:

```
SpelPoint p2;
p2 = m_spel.GetPoint("P2");
m_spel.VDefLocal(1, SpelLocalDefType.J6Camera,
SpelCalPlateType.Large, "myseq", 1.0, 1, p2);
```

15.3.20.10 VDefSetMotionRange Method

Description

Limits a motion range by VDefTool, VDefArm, VDefLocal, and VGoCenter.

Syntax

Sub VDefSetMotionRange(MaxMoveDist As Double, MaxPoseDiffAngle As Double, LjmMode As Integer)

Parameters

- MaxMoveDist
  A real number indicating the maximum travel distance.
  If 0 is specified, the range is not limited. (0 to 500. Default: 200)
  VDefTool, VDefArm, VDefLocal, and VGoCenter are used to limit the range.
- MaxPoseDiffAngle
  A real number indicating the maximum displacement angle (degrees) of the hand orientation (UVW).
  If 0 is specified, the range is not limited.
  It only affects VDefLocal. (0 to 180. Default: 45 degrees)
- LjmMode
  An integer indicating the LJM mode.

See Also

VDefTool Method, **VDefArm Method**, **VDefLocal Method**, **VGoCenter Method**, **VDefGetMotionRange Method**

VDefSetMotionRange Example

VB Example:

```
m_spel.VDefSetMotionRange(100, 30, 1)
```

C# Example:

```
m_spel.VDefSetMotionRange(100, 30, 1);
```

15.3.20.11 VDefTool Method

Description

Using vision detection, calculates a tool offset value for TPC and mobile camera position.
Note: When the tool type is other than FixedCameraWithCal, the robot operates automatically based on the detection results of the target. Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during the tool set.

Syntax

Sub VDefTool(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, Object As String)
Sub VDefTool(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, Object As String, Parent As Form)
Sub VDefTool(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double)
Sub VDefTool(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, Parent As Form)
Sub VDefTool(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer)
Sub VDefTool(ToolNumber As Integer, ToolDefType As SpelToolDefType, Sequence As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)

Parameters

- ToolNumber
  An integer (1 to 15) indicating the tool number to set.
- ToolDefType
  An integer indicating the tool type. FixedCamera: Tool set using a fixed camera which is not calibrated. J4Camera: Calculates image center of the mobile J4 camera. J6Camera: Calculates image center of the mobile J6 camera. FixedCameraWithCal: Tool set by using the fixed camera which is calibrated.
- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating a vision object in the current project. This parameter is required when ToolDefType is FixedCameraWithCal. When ToolDefType is not FixedCameraWithCal, Object should be an empty string.
- FinalAngle
  A real number indicating an angle (degrees) to rotate the tool or camera tool. Value range: 0, 5 to 180, −5 to −180 If omitted, set to “90”.
- InitAngle
  A real value indicating an angle (degrees) to rotate the tool or camera tool when setting a temporary tool. This value must be smaller than FinalAngle. Value range: −10 to 10 If omitted, set to “5”.
- TargetTolerance
  A real number indicating the pixel distance at which the vision detection result is considered to match the target position. Value range: 0 to 3 pixels. If omitted, set to “1”.
- Parent
  Optional. A .NET form that is the parent of the window.
- RobotSpeed
  Optional. Integer variable indicating the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
- RobotAccel
  Optional. Integer variable indicating the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.

See Also

VDefArm Method, **VDefGetMotionRange Method**, **VDefLocal Method**, **VDefSetMotionRange Method**, **VGoCenter Method**

VDefTool Example

VB Example:

```
m_spel.VDefTool(1, SpelToolDefType.J6Camera, "myseq", 45, 5, 3.0)
m_spel.VDefTool(1, SpelToolDefType.FixedCameraWithCal, "myseq", "myobj")
```

C# Example:

```
m_spel.VDefTool(1, SpelToolDefType.J6Camera, "myseq", 45, 5, 3.0);
m_spel.VDefTool(1, SpelToolDefType.FixedCameraWithCal, "myseq", "myobj");
```

15.3.20.12 VDefToolXYZ Method

Description

VDefToolXYZ uses vision detection to calculate tool XYZ offset values and define a tool.
Note:
The robot operates automatically based on the detection results of the target. Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during the tool set.

Syntax

Sub VDefToolXYZ(ToolNumber As Integer, LocalNumber As Integer, PointNumber1 As Integer, PointNumber2 As Integer, Sequence1 As String,Sequence2 As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer)

Sub VDefToolXYZ(ToolNumber As Integer, LocalNumber As Integer, PointNumber1 As Integer, PointNumber2 As Integer, Sequence1 As String,Sequence2 As String, FinalAngle As Double, InitAngle As Double, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)

Parameters

- ToolNumber
  An integer (1 to 15) indicating the tool number to set.
- LocalNumber
  An integer indicating the local coordinate number to move the robot to. Tool is moved in the XY plane of the specified local coordinate.
- PointNumber1
  An integer indicating the first pose of the point number.
- PointNumber2
  An integer indicating the second pose of the point number.
- Sequence1
  A string expression indicating the vision sequence name for the first pose in the current project.
- Sequence2
  A string expression indicating the vision sequence name for the second pose in the current project.
- FinalAngle
  A real number indicating the rotation angle (degrees) of the tool and camera tool. Value range: 5 to 180, −5 to −180
- InitAngle
  A real number indicating the rotation angle (degrees) display when setting a temporary tool. This value must be smaller than FinalAngle. Value range: 0.01 to 10, -0.01 to -10
- TargetTolerance
  A real number indicating the pixel distance at which the vision detection result is considered to match the target position. Value range: 0.1 to 3.0 pixels
- RobotSpeed
  An integer variable indicating the robot speed (%). Value range: 1 to 100.
- RobotAccel
  An integer variable indicating the robot acceleration (%). Value range: 1 to 99.
- Parent
  Optional. A .NET form that is the parent of the window.

See Also

VDefTool Method, **VDefToolXYZUVW Method**

VDefToolXYZ Example

VB Example:

```
m_spel.VDefToolXYZ(1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ(2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ(3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZUVW(1, 2, 3, SpelToolDefType3D.Bar)
```

C# Example:

```
m_spel.VDefToolXYZ(1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ(2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ(3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZUVW(1, 2, 3, SpelToolDefType3D.Bar);
```

15.3.20.13 VDefToolXYZUVW Method

Description

VDefToolXYZUVW calculates tool UVW offset values using three tool definitions.
Note: The calculated U, V, W tool offset values are set in the specified ToolNumber1. The X, Y, Z offsets for ToolNumber1 are not changed.

Syntax

Sub VDefToolXYZUVW(ToolNumber1 As Integer, ToolNumber2 As Integer, ToolNumber3 As Integer, ToolDefType3D As SpelToolDefType3D)

Parameters

- ToolNumber1
  An integer variable indicating the tool number for the 1st tool definition (1 to 15).
  When ToolDefType3D is Bar, specify the tool that defines the tip of the bar. When ToolDefType3D is Plane, specify the tool that defines the center of the plane.
- ToolNumber2
  An integer variable indicating the tool number for the 1st tool definition (2 to 15).
  When ToolType is bar, specify the tool that defines the middle of the bar. When ToolType is plane, specify the tool that defines a position on the plane other than the center of the plane and other than the position defined by tool definition 3.
- ToolNumber3
  An integer variable indicating the tool number for the 3rd tool definition (1 to 15).
  When ToolType is bar, specify the tool that defines the root of the bar. When ToolType is plane, specify the tool that defines a position other than the center of the plane and other than the position defined by tool definition 2.
- ToolDefType3D
  An integer indicating the tool type in the tool definition. Bar: Bar type Plane: Plane type

See Also

VDefToolXYZ Method

VDefToolXYZUVW Example

VB Example:

```
m_spel.VDefToolXYZ(1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ(2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZ(3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5)
m_spel.VDefToolXYZUVW(1, 2, 3, SpelToolDefType3D.Bar)
```

C# Example:

```
m_spel.VDefToolXYZ(1, 0, 1, 2, "seq01", "seq02", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ(2, 0, 3, 4, "seq03", "seq04", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZ(3, 0, 5, 6, "seq05", "seq06", 5, 30, 1, 5, 5);
m_spel.VDefToolXYZUVW(1, 2, 3, SpelToolDefType3D.Bar);
```

15.3.20.14 VDeleteCalibration Method

Description

Deletes a vision calibration in the current project.

Syntax

Sub VDeleteCalibration (CalibName As String)

Parameters

- CalibName
  A string expression indicating the vision calibration name for the current project.

See Also

VCreateCalibration Method, **VDeleteObject Method**, **VDeleteSequence Method**

VDeleteCalibration Example

VB Example:

```
m_spel.VDeleteCalibration("mycal")
```

C# Example:

```
m_spel.VDeleteCalibration("mycal");
```

15.3.20.15 VDeleteObject Method

Description

Deletes a vision object in the current project.

Syntax

Sub VDeleteObject (Sequence As String, ObjectName As String)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- ObjectName
  A string expression indicating the vision object name for the current project.

See Also

VCreateObject Method, **VCreateSequence Method**, **VDeleteSequence Method**

VDeleteObject Example

VB Example:

```
m_spel.VDeleteObject("myseq", "myobj")
```

C# Example:

```
m_spel.VDeleteObject("myseq", "myobj");
```

15.3.20.16 VDeleteSequence Method

Description

Deletes a vision sequence in the current project.

Syntax

Sub VDeleteSequence (Sequence As String)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.

See Also

VCreateObject Method, **VCreateSequence Method**, **VDeleteObject Method**

VDeleteSequence Example

VB Example:

```
m_spel.VDeleteSequence("myseq")
```

C# Example:

```
m_spel.VDeleteSequence("myseq");
```

15.3.20.17 VEditWindow Method

Description

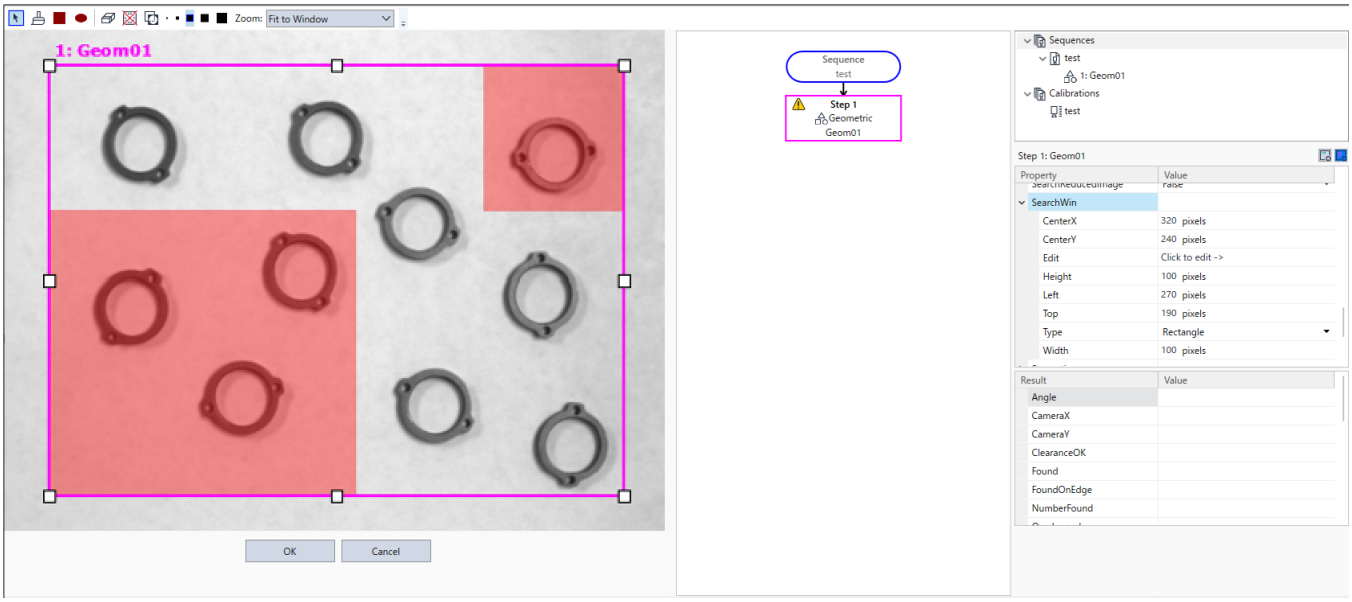
Displays the editor for search window don't care pixels. For more details, see the EditWindow Property in the Vision Guide Properties reference.

Syntax

Sub VEditWindow (Sequence As String, Object As String, Parent As Form)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- ObjectName
  A string expression indicating the vision object name for the current project.



See Also

Vsave Method

VEditWindow Example

VB Example:

```
m_spel.VEditWindow("myseq", "myobj")
```

C# Example:

```
m_spel.VEditWindow("myseq", "myobj");
```

15.3.20.18 VGet Method

Description

Gets the value of a vision sequence or object property or result.

Syntax

Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Integer)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Boolean)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Double)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As Single)
Sub VGet (Sequence As String, PropCode As SpelVisionProps, ByRef Value As String)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, ByRef Value As Integer)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, ByRef Value As Boolean)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, ByRef Value As Color)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, ByRef Value As Double)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, ByRef Value As Single)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, ByRef Value As String)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, Result As Integer, ByRef Value As Integer)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, Result As Integer, ByRef Value As Boolean)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, Result As Integer, ByRef Value As Double)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, Result As Integer, ByRef Value As Single)
Sub VGet (Sequence As String, Object As String, PropCode As SpelVisionProps, Result As Integer, ByRef Value As String)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- PropCode
  A SpelVisionProps value that specifies the property code.
- Result
  An integer expression indicating the result number.
- Value
  A character string indicating the property or result value. The type of character string must match the property or result type.

See Also

VSet Method, **VRun Method**

VGet Example

VB Example:

```
Dim i As Integer
Redim score(10) As Integer

m_spel.VRun("testSeq")
For i = 1 to 10
    m_spel.VGet("testSeq", "corr" & Format$(i, "00"), _
        SpelVisionProps.Score, score(i))
Next i
```

C# Example:

```
int[] score = new int[11];
for(int i = 1; i <= 10; i++)
{
   m_spel.VGet("testSeq", string.Format("Corr 0{0}, i),
           SpelVisionProps.Score, score(i));
}
```

15.3.20.19 VGetCameraXYU Method

Description

Retrieves camera X, Y, and U physical coordinates for any object.

Syntax

Sub VGetCameraXYU (Sequence As String, Object As String, Result As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Result
  An integer expression indicating the result number.
- Found
  A Boolean variable indicating whether the object was found.
- X
  A real number variable indicating the X coordinate in millimeters.
- Y
  A real number variable indicating the Y coordinate in millimeters.
- U
  A real number variable indicating the angle in degrees.

See Also

VGetPixelXYU Method, VGetRobotXYU Method

VGetCameraXYU Example

VB Example:

```
Dim found As Boolean
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGetCameraXYU(seq, blob, 1, found, x, y, u)
```

C# Example:

```
bool found;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGetCameraXYU(seq, blob, 1, out found, out x, out y, out u);
```

15.3.20.20 VGetEdgeCameraXYU Method

Description

Retrieves camera X, Y, and U physical coordinates for each edge of a Line Finder, Arc Finder search.

Syntax

Sub VGetEdgeCameraXYU (Sequence As String, Object As String, EdgeResultIndex As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- EdgeResultIndex
  An integer expression indicating the edge result index.
- Found
  A Boolean variable indicating whether the object was found.
- X
  A real number variable indicating the X coordinate in millimeters.
- Y
  A real number variable indicating the Y coordinate in millimeters.
- U
  A real number variable indicating the angle in degrees.

See Also

VGetEdgePixelXYU Method, **VGetEdgeRobotXYU Method**, **VGetPixelXYU Method**, **VGetRobotXYU Method**

VGetEdgeCameraXYU Example

VB Example:

```
Dim found(10) As Boolean
Dim x(10) As Single, y(10) As Single, u(10) As Single
Dim seq As String, lineFinder As String

seq = "testSeq"
lineFinder = "LineFind01"
m_spel.VRun(seq)
' The NumberOfEdges for the LineFinder is 10
For i = 1 To 10
m_spel.VGetEdgeCameraXYU(seq, lineFinder, i, found(i), x(i),
   y(i), u(i))
Next i
```

C# Example:

```
bool[] found = new bool[11];
float[] x = new float[11];
float[] y = new float[11];
float[] u = new float[11];
string seq, lineFinder;
seq = "testSeq";
lineFinder = "LineFind01";
m_spel.VRun(seq);
// The NumberOfEdges for the LineFinder is 10
for(int i = 1; i <= 10; i++)
```

```
m_spel.VGetEdgeCameraXYU(seq, lineFinder, i, out found[i],
      out x[i], out y[i], out u[i]);
```

15.3.20.21 VGetEdgePixelXYU Method

Description

Retrieves X, Y, and U pixel coordinates for each edge of a Line Finder, Arc Finder search.

Syntax

Sub VGetEdgePixelXYU (Sequence As String, Object As String, EdgeResultIndex As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- EdgeResultIndex
  An integer expression indicating the edge result index.
- Found
  A Boolean variable indicating whether the object was found.
- X
  A real number variable indicating the X coordinate in millimeters.
- Y
  A real number variable indicating the Y coordinate in millimeters.
- U
  A real number variable indicating the angle in degrees.

See Also

VGetEdgeCameraXYU Method, **VGetEdgeRobotXYU Method**, **VGetPixelXYU Method**, **VGetRobotXYU Method**

VGetEdgePixelXYU Example

VB Example:

```
Dim found(10) As Boolean
Dim x(10) As Single, y(10) As Single, u(10) As Single
Dim seq As String, lineFinder As String

seq = "testSeq"
lineFinder = "LineFind01"
m_spel.VRun(seq)
' The NumberOfEdges for the LineFinder is 10
For i = 1 To 10
m_spel.VGetEdgePixelXYU(seq, lineFinder, i, found(i), x(i),
   y(i), u(i))
Next i
```

C# Example:

```
bool[] found = new bool[11];
float[] x = new float[11];
float[] y = new float[11];
float[] u = new float[11];
string seq, lineFinder;
seq = "testSeq";
lineFinder = "LineFind01";
m_spel.VRun(seq);
// The NumberOfEdges for the LineFinder is 10
for(int i = 1; i <= 10; i++)
```

```
m_spel.VGetEdgePixelXYU(seq, lineFinder, i, out found[i],
      out x[i], out y[i], out u[i]);
```

15.3.20.22 VGetEdgeRobotXYU Method

Description

Retrieves robot X, Y, and U physical coordinates for each edge of a Line Finder, Arc Finder search.

Syntax

Sub VGetEdgeRobotXYU (Sequence As String, Object As String, EdgeResultIndex As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- EdgeResultIndex
  An integer expression indicating the edge result index.
- Found
  A Boolean variable indicating whether the object was found.
- X
  A real number variable indicating the X coordinate in millimeters.
- Y
  A real number variable indicating the Y coordinate in millimeters.
- U
  A real number variable indicating the angle in degrees.

See Also

VGetEdgeCameraXYU Method, **VGetEdgePixelXYU Method**, **VGetPixelXYU Method**, **VGetRobotXYU Method**

VGetEdgeRobotXYU Example

VB Example:

```
Dim found(10) As Boolean
Dim x(10) As Single, y(10) As Single, u(10) As Single
Dim seq As String, lineFinder As String

seq = "testSeq"
lineFinder = "LineFind01"
m_spel.VRun(seq)
' The NumberOfEdges for the LineFinder is 10
For i = 1 To 10
m_spel.VGetEdgeRobotXYU(seq, lineFinder, i, found(i), x(i),
   y(i), u(i))
Next i
```

C# Example:

```
bool[] found = new bool[11];
float[] x = new float[11];
float[] y = new float[11];
float[] u = new float[11];
string seq, lineFinder;
seq = "testSeq";
lineFinder = "LineFind01";
m_spel.VRun(seq);
// The NumberOfEdges for the LineFinder is 10
for(int i = 1; i <= 10; i++)
```

```
m_spel.VGetEdgeRobotXYU(seq, lineFinder, i, out found[i],
      out x[i], out y[i], out u[i]);
```

15.3.20.23 VGetExtrema Method

Description

Retrieves extrema coordinates of a blob object.

Syntax

Sub VGetExtrema (Sequence As String, Object As String, Result As Integer, ByRef MinX As Single, ByRef MaxX As Single, ByRef MinY As Single, ByRef MaxY As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Result
  An integer expression indicating the result number.
- MinX
  A real number variable indicating the minimum X coordinate in pixels.
- MaxX
  A real number variable indicating the maximum X coordinate in pixels.
- MinY
  A real number variable indicating the minimum Y coordinate in pixels.
- MaxY
  A real number variable indicating the maximum Y coordinate in pixels.

See Also

VGet Method

VGetExtrema Example

VB Example:

```
Dim xmin As Single, xmax As Single
Dim ymin As Single, ymax As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGet(seq, blob, "found", found)
If found <> 0 Then
    m_spel.VGetExtrema(seq, blob, xmin, xmax, ymin, ymax)
End If
```

C# Example:

```
float xMin, xMax, yMin, yMax;
bool found;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGet(seq, blob, "found", found);

if(found == true)
```

```
    m_spel.VGetExtrema(seq, blob,out xMin,out xMax,out yMin,
        out yMax);
```

15.3.20.24 VGetModelWin Method

Description

Retrieves model window coordinates for objects.

Syntax

Sub VGetModelWin (Sequence As String, Object As String, ByRef Left As Integer, ByRef Top As Integer, ByRef Width As Integer, ByRef Height As Integer)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Left
  An integer variable indicating the left coordinate in pixels.
- Top
  An integer variable indicating the top coordinate in pixels.
- Width
  An integer variable indicating the width in pixels.
- Height
  An integer variable indicating the height in pixels.

See Also

VSetModelWin Method, **VGetSearchWIn Method**, **VSetSearchWIn Method**

VGetModelWin Example

VB Example:

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetModelWin("testSeq", "corr01", left, top, _
       width, height)
    .VSetModelWin("testSeq", "corr01", left + 20, top, _
       width, height)
    .VTeach("testSeq", "corr01")
End With
```

C# Example:

```
int left, top, width, height;

m_spel.VGetModelWin("testSeq", "corr01", out left, out top,
       out width, out height);
m_spel.VSetModelWin("testSeq", "corr01", left + 20, top,
       width, height);
m_spel.VTeach("testSeq", "corr01");
```

15.3.20.25 VGetPixelToCamera Method

Description

Retrieves the camera coordinates for the specified pixel coordinates.

Syntax

Sub VGetPixelToCamera (Calibration As String, PixelX As Single, PixelY As Single, Angle As Single, ByRef CameraX As Single, ByRef CameraY As Single, ByRef CameraU As Single)

Parameters

- Calibration
  A string expression indicating the calibration name for the current project.
- PixelX
  A real number variable indicating the X coordinate in pixels.
- PixelY
  A real number variable indicating the Y coordinate in pixels.
- Angle
  A real number variable indicating the angle in degrees.
- CameraX
  A real number variable indicating the X coordinate in millimeters.
- CameraY
  A real number variable indicating the Y coordinate in millimeters.
- CameraU
  A real number variable indicating the angle in degrees.

See Also

VGetPixelXYU Method, **VGetCameraXYU Method**, **VGetPixelToRobot Method**

VGetPixelToCamera Example

VB Example:

```
Dim x As Single, y As Single, u As Single
Dim cal As String, seq As String

cal = "testCal"
seq = "testSeq"
m_spel.VRun(seq)
m_spel.VGetPixelToCamera(cal, 400, 300, 0, x, y, u)
```

C# Example:

```
float x, y, u
string cal, seq;

cal = "testCal";
seq = "testSeq";
m_spel.VRun(seq);
m_spel.VGetPixelToCamera(cal, 400, 300, 0, out x, out y, out u);
```

15.3.20.26 VGetPixelToRobot Method

Description

Retrieves robot world coordinates for the specified pixel coordinates.

Syntax

Sub VGetPixelToRobot (Calibration As String, PixelX As Single, PixelY As Single, Angle As Single, ByRef RobotX As Single, ByRef RobotY As Single, ByRef RobotU As Single)

Parameters

- Calibration
  A string expression indicating the calibration name for the current project.
- PixelX
  A real number variable indicating the X coordinate in pixels.
- PixelY
  A real number variable indicating the Y coordinate in pixels.
- Angle
  A real number variable indicating the angle in degrees.
- RobotX
  A real number variable indicating the X coordinate in millimeters.
- RobotY
  A real number variable indicating the Y coordinate in millimeters.
- RobotU
  A real number variable indicating the angle in degrees.

See Also

VGetPixelXYU Method, **VGetRobotXYU Method**, **VGetPixelToCamera Method**

VGetPixelToRobot Example

VB Example:

```
Dim x As Single, y As Single, u As Single
Dim cal As String, seq As String

cal = "testCal"
seq = "testSeq"
m_spel.VRun(seq)
m_spel.VGetPixelToRobot(cal, 400, 300, 0, x, y, u)
```

C# Example:

```
float x, y, u
string cal, seq;

cal = "testCal";
seq = "testSeq";
m_spel.VRun(seq);
m_spel.VGetPixelToRobot(cal, 400, 300, 0, out x, out y, out u);
```

15.3.20.27 VGetPixelXYU Method

Description

Retrieves pixel X, Y, and U coordinates for any object.

Syntax

Sub VGetPixelXYU (Sequence As String, Object As String, Result As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Result
  An integer expression indicating the result number.
- Found
  A Boolean expression indicating whether the object was found.
- X
  A real number variable indicating the X coordinate in pixels.
- Y
  A real number variable indicating the Y coordinate in pixels.
- U
  A real number variable indicating the angle in degrees.

See Also

VGetCameraXYU Method, **VGetRobotXYU Method**

VGetPixelXYU Example

VB Example:

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGetPixelXYU(seq, blob, 1, found, x, y, u)
```

C# Example:

```
int found;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGetPixelXYU(seq, blob,1, out found, out x, out y, out u);
```

15.3.20.28 VGetRobotPlacePos Method

Description

Retrieves robot place position.

Syntax

Sub VGetRobotPlacePos (Sequence As String, Object As String, Result As Integer, ByRef Found As Boolean, ByRef PlacePointAs SpelPoint)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Result
  An integer expression indicating the result number.
- Found
  An integer variable indicating the Boolean found status. If found is false, then X, Y, and U are undefined.
- PlacePoint
  A SpelPoint variable indicating the place position.

See Also

VGetRobotPlaceTargetPos Method, VSetRobotPlaceTargetPos Method

VGetRobotPlacePos Example

VB Example:

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String
Dim placePoint As SpelPoint

seq = "testSeq"
blob = "blob01"
' Move part above upward camera
m_spel.Jump("camPos")
m_spel.VRun(seq)
m_spel.VGetRobotPlacePos(seq, blob, 1, found, placePoint)
' Using a SCARA, to use +TLZ for approach
m_spel.Jump(placePoint, "+TLZ(10)")
m_spel.Go(placePoint)
```

C# Example:

```
bool found;
float x, y, u;
string seq, blob;
SpelPoint placePoint = new SpelPoint();
seq = "testSeq";
blob = "blob01";

// Move part above upward camera
m_spel.Jump("camPos");
m_spel.VRun(seq);
m_spel.VGetRobotPlacePos(seq,blob,1,out found, out placePoint);
// Using a SCARA, to use +TLZ for approach
m_spel.Jump(placePoint, "+TLZ(10)");
m_spel.Go(placePoint);
```

15.3.20.29 VGetRobotPlaceTargetPos Method

Description

Retrieves part place position.

Syntax

Sub VGetRobotPlaceTargetPos (Sequence As String, Object As String, ByRef Point As SpelPoint)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Point
  A SpelPoint variable indicating the place position.

See Also

VGetRobotPlacePos Method, **VSetRobotPlaceTargetPos Method**

VGetRobotPlaceTargetPos Example

VB Example:

```
Dim seq As String, blob As String
Dim targetPoint As SpelPoint

seq = "testSeq"
blob = "blob01"
m_spel.VGetRobotPlaceTargetPos(seq, blob, targetPoint)

' Adjust the place position
targetPoint.X = targetPoint.X + 10
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint)
```

C# Example:

```
Spe
lPoint targetPoint = new SpelPoint();

seq = "testSeq";
blob = "blob01";
m_spel.VGetRobotPlaceTargetPos(seq, blob, out targetPoint);

// Adjust the place position
targetPoint.X = targetPoint.X + 10;
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint);
```

15.3.20.30 VGetRobotXYU Method

Description

Retrieves robot world X, Y, and U coordinates for any object.

Syntax

Sub VGetRobotXYU (Sequence As String, Object As String, Result As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Result
  An integer expression indicating the result number.
- Found
  An integer variable indicating the Boolean found status. If found is false, then X, Y, and U are undefined.
- X
  A real number variable indicating the X coordinate in millimeters.
- Y
  A real number variable indicating the Y coordinate in millimeters.
- U
  A real number variable indicating the angle in degrees.

See Also

VGetCameraXYU Method, VGetPixelXYU Method

VGetRobotXYU Example

VB Example:

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
m_spel.VRun(seq)
m_spel.VGetRobotXYU(seq, blob, 1, found, x, y, u)
```

C# Example:

```
bool found;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
m_spel.VRun(seq);
m_spel.VGetRobotXYU(seq, blob, 1, out found, out x, out y, out u);
```

15.3.20.31 VGetRobotToolXYU Method

Description

Retrieves robot world X, Y, and U values for tool definition.

Syntax

Sub VGetRobotToolXYU (Sequence As String, Object As String, Result As Integer, ByRef Found As Boolean, ByRef X As Single, ByRef Y As Single, ByRef U As Single)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Result
  An integer expression indicating the result number.
- Found
  An integer variable indicating the Boolean found status. If found is false, then X, Y, and U are undefined.
- X
  A real number variable indicating the X coordinate in millimeters.
- Y
  A real number variable indicating the Y coordinate in millimeters.
- U
  A real number variable indicating the angle in degrees.

Remarks

Use VGetRobotToolXYU to easily define a tool for a part viewed by an upward camera. This allows you to pick up a part, search for it in the upward camera FOV, define a tool for the part, then place the part.

See Also

VGetCameraXYU Method, **VGetPixelXYU Method**, **VGetRobotXYU Method**

VGetRobotToolXYU Example

VB Example:

```
Dim found As Integer
Dim x As Single, y As Single, u As Single
Dim seq As String, blob As String

seq = "testSeq"
blob = "blob01"
' Move part above upward camera
m_spel.Jump("camPos")
m_spel.VRun(seq)
m_spel.VGetRobotToolXYU(seq, blob, 1, found, x, y, u)
m_spel.TLSet(1, x, y, u)
```

C# Example:

```
bool fnd;
float x, y, u;
string seq, blob;

seq = "testSeq";
blob = "blob01";
// Move part above upward camera
```

```
m_spel.Jump("camPos");
m_spel.VRun(seq);
m_spel.VGetRobotToolXYU(seq, blob, 1, out fnd, out x, out y, out u);
m_spel.TLSet(1, x, y, u);
```

15.3.20.32 VGetSearchWIn Method

Description

Retrieves search window coordinates.

Syntax

Sub VGetSearchWin (Sequence As String, Object As String, ByRef Left As Integer, ByRef Top As Integer, ByRef Width As Integer, ByRef Height As Integer)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Left
  An integer variable indicating the left coordinate in pixels.
- Top
  An integer variable indicating the top coordinate in pixels.
- Width
  An integer variable indicating the width in pixels.
- Height
  An integer variable indicating the height in pixels.

See Also

VGetModelWin Method, **VSetModelWin Method**, **VSetSearchWIn Method**

VGetSearchWin Example

VB Example:

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetSearchWin("testSeq", "corr01", left, top, _
       width, height)
    .VSetSearchWin("testSeq", "corr01", newLeft, top, _
       width, height)
    .VRun("testSeq")
End With
```

C# Example:

```
int left, top, width, height;

m_spel.VGetSearchWin("testSeq", "corr01", out left, out top,
       out width, out height);
m_spel.VSetSearchWin("testSeq", "corr01", newLeft, top,
       width, height);
m_spel.VRun("testSeq");
```

15.3.20.33 VGoCenter Method

Description

Using a feature point that can be detected by the vision system, moves the robot to a position where the feature point is on the center of the camera image.

Syntax

Sub VGoCenter(Sequence As String, LocalNumber As Integer, TargetTolerance As Double)
Sub VGoCenter(Sequence As String, LocalNumber As Integer, TargetTolerance As Double, Parent As Form)
Sub VGoCenter(Sequence As String, LocalNumber As Integer, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer)
Sub VGoCenter(Sequence As String, LocalNumber As Integer, TargetTolerance As Double, RobotSpeed As Integer, RobotAccel As Integer, Parent As Form)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- LocalNumber
  An integer indicating the local coordinate number to move the robot to.
  If -1 is specified, the robot moves in the XY plane of the tool rotation
- TargetTolerance
  A real number indicating the pixel distance at which the vision detection result is considered to match the target position.
  Value range: 0 to 3 pixels
- Form
  Optional. A .NET form that is the parent of the window.
- RobotSpeed
  Optional. Integer variable indicating the robot speed (%). Value range: 0 to 100 If omitted, set to “5”.
- RobotAccel
  Optional. Integer variable indicating the robot acceleration (%). Value range: 0 to 99 If omitted, set to “5”.

See Also

VDefArm Method, **VDefGetMotionRange Method**, **VDefLocal Method**, **VDefSetMotionRange Method**, **VDefTool Method**

VGoCenter Example

VB Example:

```
m_spel.VGoCenter("myseq", 1, 1.0)
```

C# Example:

```
m_spel.VGoCenter("myseq", 1, 1.0);
```

15.3.20.34 VLoad Method

Description

Loads vision properties from the current project.

Syntax

Sub VLoad ()

Remarks

Use the VLoad method when you want to return the vision property settings, models, and fonts back to their original settings when the program was started.

See Also

Vsave Method

VLoad Example

VB Example:

```
m_spel.VLoad()
```

C# Example:

```
m_spel.VLoad();
```

15.3.20.35 VLoadModel Method

Description

Load a vision model from a disk file.

Syntax

Sub VLoadModel (Sequence As String, Object As String, Path As String)

Parameters

- Sequence
  A character string indicating the sequence name in the current project.
- Object
  A character sting indicating the object name. The object must be a Correlation, Geometric, or Polar.
- Path
  The full path name of the file to load the model from, excluding the extension.

Remarks

An error will occur if the model data in the file is the wrong type. For example, if you try to load a polar model into a correlation, an error will occur. If you supply a file extension, it is ignored. There are two files associated with fileName.
For correlation and geometric models, the ModelOrgX and ModelOrgY values are restored along with the model window width and height.
For polar models, the Radius, Thickness, and AngleOffset are restored.

See Also

VSaveModel Method

VLoadModel Example

VB Example:

```
m_spel.VLoadModel("seq01", "corr01", "d:\models\part1")
```

C# Example:

```
m_spel.VLoadModel("seq01", "corr01", @"d:\models\part1");
```

15.3.20.36 VRun Method

Description

Run a vision sequence in the current project.

Syntax

Sub VRun (Sequence As String)

Parameters

- Sequence
  A character string indicating the sequence name in the current project.

Remarks

VRun works with sequences using any type of camera calibration or no calibration.
To display graphics, you need to use a SPELVideo control and set the SpelVideoControl property of the Spel class instance to the SPELVideo control.
After you execute VRun, use VGet to retrieve results.

See Also

VGet Method, **VSet Method**

VRun Example

VB Example:

```
Function FindPart(x As Single, y As Single, angle As Single)As Boolean
  Dim found As Boolean
  Dim x, y, angle As Single
  With m_spel
    .VRun("seq01")
    .VGet("seq01", "corr01", "found", found)
    If found Then
      .VGet("seq01", "corr01", "cameraX", x)
      .VGet("seq01", "corr01", "cameraY", y)
      .VGet("seq01", "corr01", "angle", angle)
      FindPart = True
    End If
  End With
End Function
```

C# Example:

```
bool FindPart(float x, float y, float angle)
{
  bool found;
  m_spel.VRun("seq01");
  m_spel.VGet("seq01", "corr01", "found", found);
  if (found) {
        m_spel.VGet("seq01", "corr01", "cameraX", out x);
       m_spel.VGet("seq01", "corr01", "cameraY", out y);
        m_spel.VGet("seq01", "corr01", "angle", out angle);
}
return found;
}
```

15.3.20.37 Vsave Method

Description

Saves all vision data in the current project.

Syntax

Sub VSave ()

Remarks

Use VSave to make any changes to vision properties permanent.

See Also

VSet Method

VSaveImage Example

VB Examples:

```
With m_spel
    .VSet("seq01", "blob01", "SearchWinLeft", 100)
    .VSet("seq01", "corr01", "Accept", userAccept)
    .VSave()
End With
```

C# Example:

```
m_spel.VSet("seq01", "blob01", "SearchWinLeft", 100);
m_spel.VSet("seq01", "corr01", "Accept", userAccept);
m_spel.VSave();
```

15.3.20.38 VSaveImage Method

Description

Save a vision video window to a PC disk file.

Syntax

Sub VSaveImage (Sequence As String, Path As String)
Sub VSaveImage (Sequence As String, Path As String, WithGraphics As Boolean)

Parameters

- Sequence
  A character string indicating the sequence name in the current project.
- Path
  The full path name of the file to save the image to, including the extension.
- WithGraphics
  A Boolean expression that determines whether to save the sequence result image to an image file.

Remarks

Use VSaveImage to save an image on the Video display to disk. The file extension must be
BMP,
TIF,
or JPG.

See Also

LoadImage Method

VSaveImage Example

VB Examples:

```
Dim found As Boolean
m_spel.VRun("Seq")
m_spel.VGet("Seq", SpelVisionProps.AllFound, found)
If Not found Then
    m_spel.VSaveImage("Seq", "d:\reject.mim")
End If
```

C# Example:

```
bool found;
m_spel.VRun("Seq");
m_spel.VGet("Seq", SpelVisionProps.AllFound, out found);

if (\!found)
    m_spel.VSaveImage("Seq", @"d:\reject.mim");
```

15.3.20.39 VSaveModel Method

Description

Save a vision object search model to a PC disk file.

Syntax

Sub VSaveModel (Sequence As String, Object As String, Path As String)

Parameters

- Sequence
  A character string indicating the sequence name in the current project.
- Object
  A character sting indicating the object name. The object must be a Correlation, Geometric, or Polar.
- Path
  The full path name of the file to save the model to, excluding the extension.

Remarks

When VSaveModel is executed, Epson RC+ 8.0 creates two files (Path + extensions): Path.VOB, Path.MDL
For correlation and geometric models, the ModelOrgX and ModelOrgY values are saved along with the model window. For Polar models, the Radius, Thickness, and AngleOffset are saved.

See Also

VLoadModel Method

VSaveModel Example

VB Example:

```
m_spel.VSaveModel("seq01", "corr01", "d:\models\part1")
```

C# Example:

```
m_spel.VSaveModel("seq01", "corr01", @"d:\models\part1");
```

15.3.20.40 VSet Method

Description

Sets the value of a vision sequence or object property.

Syntax

Sub VSet (Sequence As String, PropCode As SpelVisionProps, Value As Integer)
Sub VSet (Sequence As String, PropCode As SpelVisionProps, Value As Boolean)
Sub VSet (Sequence As String, PropCode As SpelVisionProps, Value As Double)
Sub VSet (Sequence As String, PropCode As SpelVisionProps, Value As Single)
Sub VSet (Sequence As String, PropCode As SpelVisionProps, Value As String)
Sub VSet (Sequence As String, Object As String, PropCode As SpelVisionProps, Value As Integer)
Sub VSet (Sequence As String, Object As String, PropCode As SpelVisionProps, Value As Boolean)
Sub VSet (Sequence As String, Object As String, PropCode As SpelVisionProps, Value As Color)
Sub VSet (Sequence As String, Object As String, PropCode As SpelVisionProps, Value As Double)
Sub VSet (Sequence As String, Object As String, PropCode As SpelVisionProps, Value As Single)
Sub VSet (Sequence As String, Object As String, PropCode As SpelVisionProps, Value As String)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- propName
  A string expression indicating the property name you want to set the value for.
- PropCode
  A SpelVisionProps value that specifies the property code.
- value
  Expression indicating the new value The type of expression must match the property type.

See Also

VGet Method, **VRun Method**

VSet Example

VB Example:

```
m_spel.VSet("seq01", "corr01", SpelVisionProps.Accept, 250)
```

C# Example:

```
m_spel.VSet("seq01", "corr01", SpelVisionProps.Accept, 250);
```

15.3.20.41 VSetModelWin Method

Description

Sets model window coordinates.

Syntax

Sub VSetModelWin (Sequence As String, Object As String, Left As Integer, Top As Integer, Width As Integer, Height As Integer)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Left
  An integer expression indicating the left coordinate in pixels.
- Top
  An integer expression indicating the top coordinate in pixels.
- Width
  An integer expression indicating the width in pixels.
- Height
  An integer expression indicating the height in pixels.

See Also

VGetModelWin Method, **VGetSearchWIn Method**, **VSetSearchWIn Method**

VSetModelWin Example

VB Example:

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
    .VGetModelWin("testSeq", "corr01", left, top, _
       width, height)
    .VSetModelWin("testSeq", "corr01", left + 20, top, _
       width, height)
    .VTeach("testSeq", "corr01")
End With
```

C# Example:

```
int left, top, width, height;

m_spel.VGetModelWin("testSeq", "corr01", out left, out top,
       out width, out height);
m_spel.VSetModelWin("testSeq", "corr01", left + 20, top,
       width, height);
m_spel.VTeach("testSeq", "corr01");
```

15.3.20.42 VSetRobotPlaceTargetPos Method

Description

Sets part place position.

Syntax

Sub VSetRobotPlaceTargetPos (Sequence As String, Object As String, Point As SpelPoint)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Point
  A SpelPoint variable indicating the place position.

See Also

VGetRobotPlacePos Method, VGetRobotPlaceTargetPos Method

VSetRobotPlaceTargetPos Example

VB Example:

```
Dim seq As String, blob As String
Dim targetPoint As SpelPoint

seq = "testSeq"
blob = "blob01"
m_spel.VGetRobotPlaceTargetPos(seq, blob, targetPoint)

' Adjust the place position
targetPoint.X = targetPoint.X + 10
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint)
```

C# Example:

```
string seq, blob;
SpelPoint targetPoint = new SpelPoint();

seq = "testSeq";
blob = "blob01";
m_spel.VGetRobotPlaceTargetPos(seq, blob, out targetPoint);

// Adjust the place position
targetPoint.X = targetPoint.X + 10;
m_spel.VSetRobotPlaceTargetPos(seq, blob, targetPoint);
```

15.3.20.43 VSetSearchWIn Method

Description

Sets search window coordinates.

Syntax

Sub VSetSearchWin (Sequence As String, Object As String, Left As Integer, Top As Integer, Width As Integer, Height As Integer)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- Object
  A string expression indicating the object name for the sequence.
- Left
  An integer expression indicating the left coordinate in pixels.
- Top
  An integer expression indicating the top coordinate in pixels.
- Width
  An integer expression indicating the width in pixels.
- Height
  An integer expression indicating the height in pixels.

See Also

VGetModelWin Method, **VSetModelWin Method**, **VGetSearchWIn Method**

VSetSearchWin Example

VB Example:

```
Dim left As Integer, top As Integer
Dim width As Integer, height As Integer

With m_spel
  .VGetSearchWin("testSeq", "corr01", left, top, _
    width, height)
  .VSetSearchWin("testSeq", "corr01", newLeft, top, _
    width, height)
  .VRun("testSeq")
End With
```

C# Example:

```
int left, top, width, height;

m_spel.VGetSearchWin("testSeq", "corr01", out left, out top,
    out width, out height);
m_spel. .VSetSearchWin("testSeq", "corr01", left + 50,
    top, width, height);
m_spel.VRun("testSeq");
```

15.3.20.44 VShowModel Method

Description

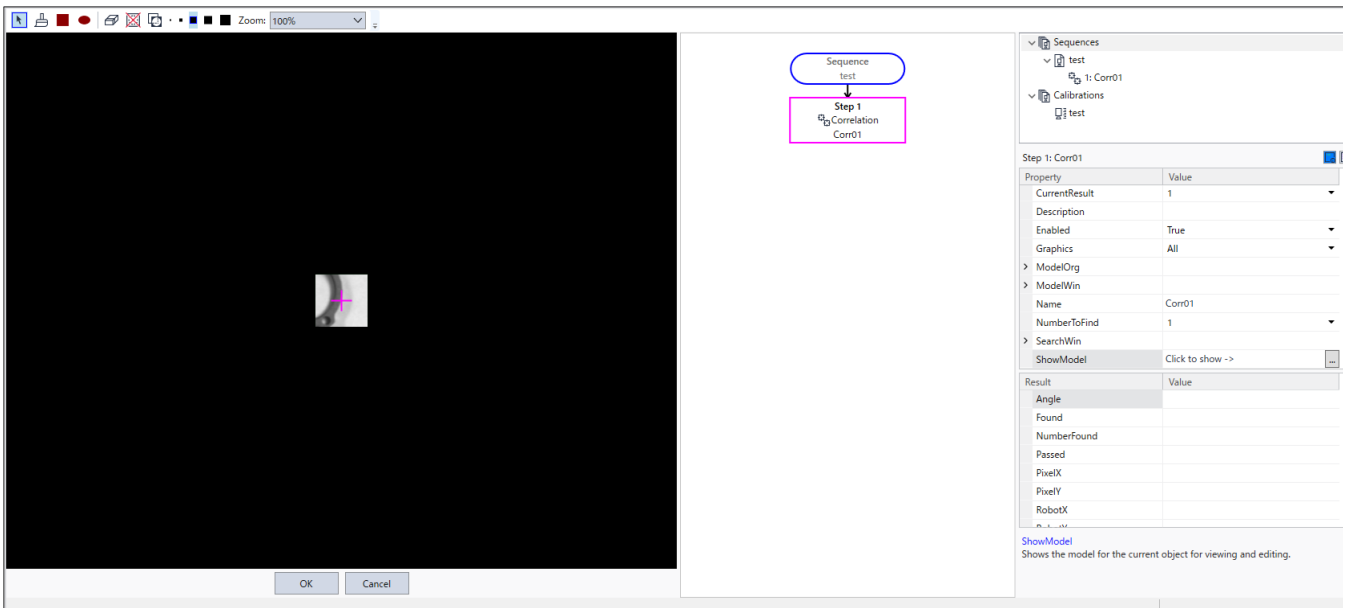
Display the object model. For more details, see the ShowModel Property in the Vision Guide Properties reference.

Syntax

Sub VShowModel (Sequence As String, Object As String, Parent As Form)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.
- ObjectName
  A string expression indicating the vision object name for the current project.



See Also

VShowSequence Method, VTrain Method

VShowModel Example

VB Example:

```
m_spel.VShowModel("myseq", "myobj")
```

C# Example:

```
m_spel.VShowModel("myseq", "myobj");
```

15.3.20.45 VShowSequence Method

Description

Displays all objects in a sequence.

Syntax

Sub VShowSequence (Sequence As String)

Parameters

- Sequence
  A string expression indicating the vision sequence name to create.

Remarks

Use VShowSequence to display the objects in a sequence without running the sequence. The active object color (magenta) is used for all objects so that they can be seen easily.
One use is for when a robot camera is moved over a particular portion of a part being scanned with several sequences. After the robot is positioned, VShowSequence can be called to display the sequence.

See Also

VShowModel Method

VShowSequence Example

VB Example:

```
m_spel.VShowSequence("myseq")
```

C# Example:

```
m_spel.VShowSequence("myseq");
```

15.3.20.46 VStatsReset Method

Description

Resets vision statistics for a specified sequence in the current project.

Syntax

Sub VStatsReset (Sequence As String)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.

Remarks

VStatsReset resets the statistics for the specified sequence in memory only for the current Epson RC+ 8.0 session. You should execute VStatsSave if you want changes to be permanent. Otherwise, if you restart Epson RC+ 8.0, the statistics are restored from disk.

See Also

VStatsResetAll Method, **VStatsShow Method**, **VStatsSave Method**

VStatsReset Example

VB Example:

```
Sub btnResetStats_Click()
    m_spel.VStatsReset("seq01")
End Sub
```

C# Example:

```
void btnResetStats_Click(object sender, EventArgs e)
{
m_spel.VStatsReset("seq01");}
```

15.3.20.47 VStatsResetAll Method

Description

Resets vision statistics for all sequences.

Syntax

Sub VStatsResetAll

Remarks

VStatsResetAll resets the statistics in memory only for the current Epson RC+ 8.0 session. You should execute VStatsSave if you want changes to be permanent.

See Also

VStatsReset Method, **VStatsShow Method**, **VStatsSave Method**

VStatsResetAll Example

VB Example:

```
Sub btnResetStats_Click()
    m_spel.VStatsResetAll()
End Sub
```

C# Example:

```
void btnResetStats_Click(object sender, EventArgs e)
{
    m_spel.VStatsResetAll();
}
```

15.3.20.48 VStatsSave Method

Description

Saves vision statistics for all sequences in the current project.

Syntax

Sub VStatsSave ()

Remarks

VStatsSave must be executed before Epson RC+ 8.0 is shut down if you want to preserve changes made to vision statistics.

See Also

VStatsReset Method, **VStatsResetAll Method**, **VStatsShow Method**

VStatsSave Example

VB Example:

```
Sub btnResetStats_Click()
    m_spel.VStatsSave()
End Sub
```

C# Example:

```
void btnResetStats_Click(object sender, EventArgs e)
{
    m_spel.VStatsSave();
}
```

15.3.20.49 VStatsShow Method

Description

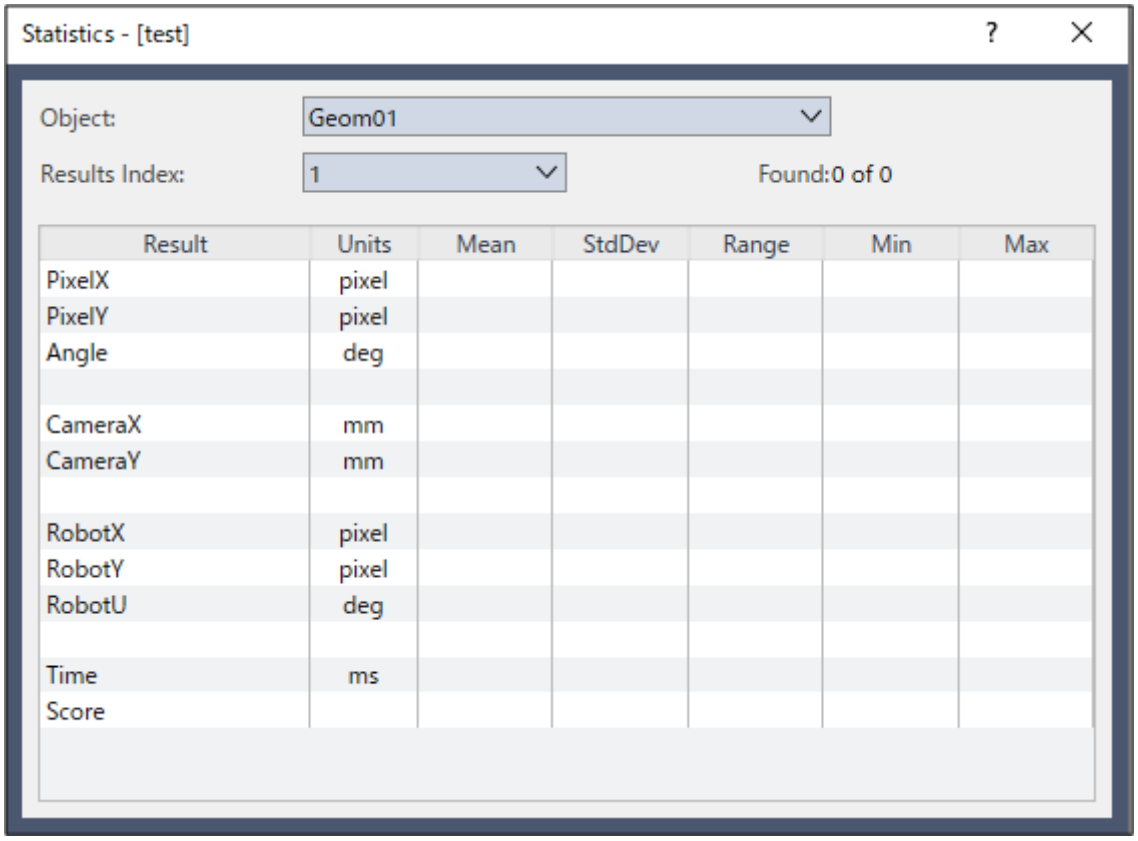
Displays the vision statistics dialog for a specified sequence in the current project.

Syntax

Sub VStatsShow (Sequence As String, Parent As Form)

Parameters

- Sequence
  A string expression indicating the vision sequence name for the current project.



See Also

VStatsReset Method, VStatsResetAll Method, VStatsSave Method

VStatsShow Example

VB Example:

```
Sub btnShowStats_Click()
    m_spel.VStatsShow("seq01")
End Sub
```

C# Example:

```
void btnShowStats_Click(object sender, EventArgs e)
{
    m_spel.VStatsShow("seq01");
}
```

15.3.20.50 VTeach Method

Description

Teach a correlation, geometric, or polar model.

Syntax

Sub VTeach (Sequence As String, Object As String, ByRef Status as Integer)
Sub VTeach (Sequence As String, Object As String, AddSample as Boolean, KeepDontCares As Boolean, ByRef Status as Integer)

Parameters

- Sequence
  The name of a vision sequence in the current project.
- Object
  The object name. Correlation, Geometric, or Polar objects can be taught.
- AddSample
  True to add a sample, False to add as a new model.
- KeepDontCares
  True to use the existing detection mask, False to discard it.
- Status
  Returns the status 0: Unsuccessful, 1: Successful

Remarks

Before you call VTeach, you must ensure that the model window is in the correct position.
For polar objects, the search window and thickness must be set properly. Set the search window location and thickness using VSet.
For correlation and geometric objects, the search window and the model window must be set properly. Set the search and model window locations using VSet for SearchWin and ModelWin. Or you can use the VTrain command so the operator can interactively change the windows.
After teaching the models, you can save them to a PC disk file using the VSaveModel method.

See Also

VTrain Method, **VSaveModel Method**

VTeach Example

VB Example:

```
Dim status As Integer

' First let the operator change the window position
m_spel.VTrain("seq01", "corr01", status)

' Now teach the model
m_spel.VTeach("seq01", "corr01", status)
```

C# Example:

```
int status;

// First let the operator change the window position
m_spel.VTrain("seq01", "corr01", status);

// Now teach the model
m_spel.VTeach("seq01", "corr01", out status);
```

15.3.20.51 VTrain Method

Description

This command allows you to train objects in an entire sequence or individual objects.

Syntax

Function VTrain (Sequence As String [, Object As String] [, Flags as Integer] [, Parent as Form]) As Boolean

Parameters

- Sequence
  The name of a vision sequence in the current project.
- Object
  The object name in Sequence. You can register any type of object. If Object is an empty string, then the entire sequence can be trained.
- Flags
  VTrain dialog configuration. (Optional)1 - Displays the Teach button. 2 - Do not display the model window.
- Parent
  Optional. A .NET form that is the parent of the window.

Return Value

If the operator clicks the OK button, VTrain returns True, otherwise it returns False.

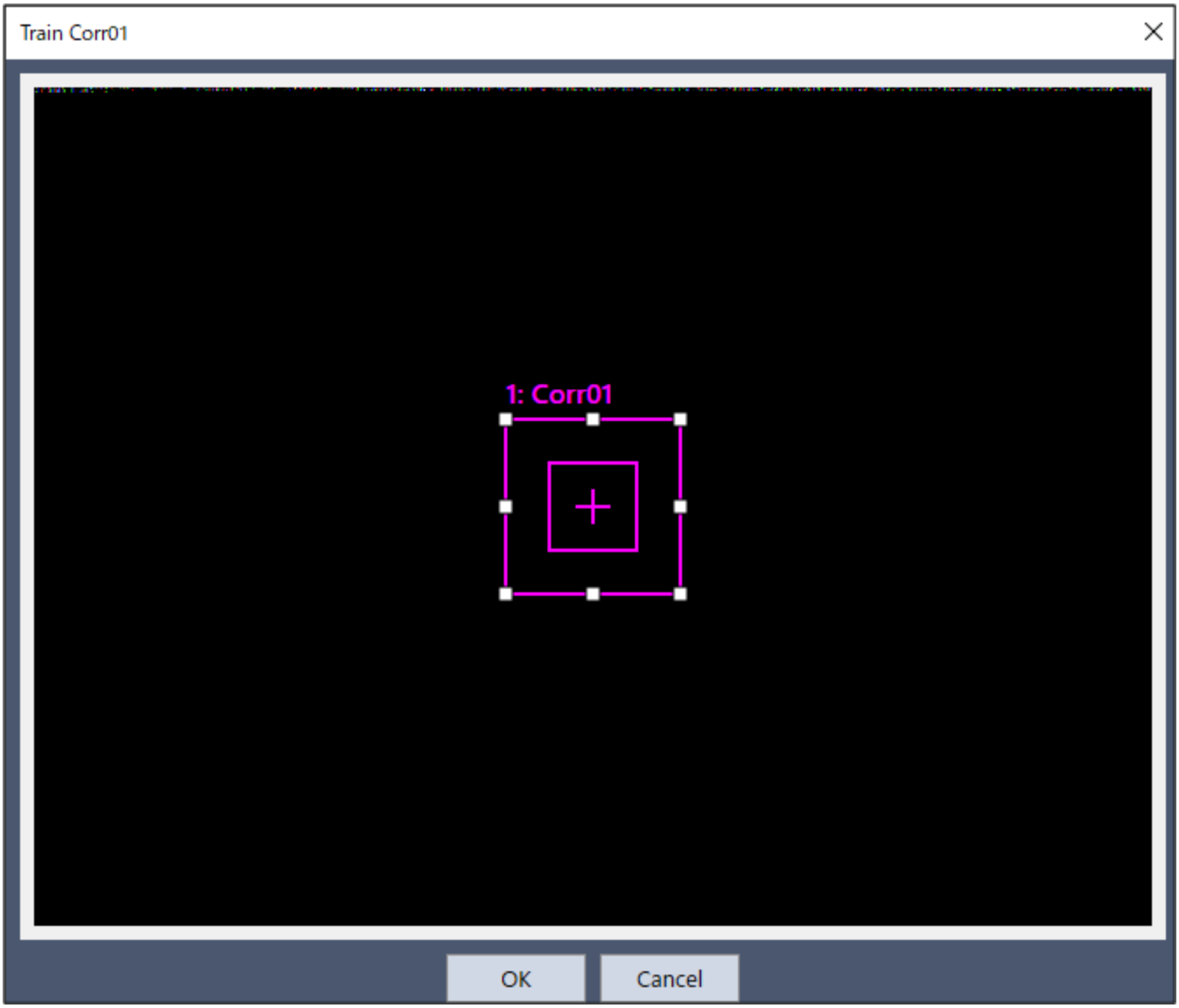
Remarks

When VTrain is executed, a dialog is opened showing a live video image with the specified object displayed. The operator can size/move the search window, and train the model window (for correlation and geometric objects). When the operator is finished, he can click on OK to save the changes, or Cancel to ignore the changes. If OK is clicked, then the new information is automatically saved in the current project.

If flags bit 1 is set, a teach button will be displayed. For Correlation, Geometric, and Polar objects, the model will be taught if the teach button is clicked. You can retrieve the ModelOK property after running VTrain to check if a model was trained. For Blob objects, the button will open the Histogram dialog and the operator can adjust both high and low thresholds and then view the effects of changes.

If flags bit 2 is set, model windows will not be displayed. The operator can only change search windows.

For correlation and geometric objects, you can call VTeach after calling VTrain to teach the model if you are not displaying the teach button.



See Also

VTeach Method, VSaveModel Method

VTrain Example

VB Example:

```
Dim status As Integer
Dim trainOK As Boolean

' First let the operator change the window position
trainOK = m_spel.VTrain("seq01", "corr01")

' Now teach the model
If trainOK Then
    m_spel.VTeach("seq01", "corr01", status)
EndIf
```

C# Example:

```
int status;
bool trainOK;
```

```
// First let the operator change the window position
trainOK = m_spel.VTrain("seq01", "corr01");

// Now teach the model
if (trainOK)
    m_spel.VTeach("seq01", "corr01", out status);
```

15.3.21 W

15.3.21.1 WaitCommandComplete method.

Description

This command waits for a command started with AsyncMode = True to complete.

Syntax

Sub WaitCommandComplete ()

See Also

AsyncMode Property

WaitCommandComplete Example

VB Example:

```
With m_spel
  .AsyncMode = True
  .Jump("pick")
  .Delay(500)
  .On(1)
  .WaitCommandComplete()
End With
```

C# Example:

```
m_spel.AsyncMode = true;
m_spel.Jump("pick");
m_spel.Delay(500);
m_spel.On(1);
m_spel.WaitCommandComplete();
```

15.3.21.2 WaitMem Method

Description

Waits for a memory bit status to change.

Syntax

Sub WaitMem (BitNumber As Integer, Condition As Boolean, Timeout As Single)
Sub WaitMem (Label As String, Condition As Boolean, Timeout As Single)

Parameters

- BitNumber
  An integer expression indicating the memory bit number.
- Label
  A character string indicating the memory bit label.
- Condition
  A Boolean expression indicating the memory bit state.
- Timeout
  Maximum wait time (s)

Remarks

You should always check if a time out occurred by using the TW method. See the example below.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

WaitSw Method, **SetPoint Method**

WaitMem Example

VB Example:

```
' Wait for memory bit 1 to be 1 (True)
' Max time is 5 seconds
m_spel.WaitMem(1, True, 5)
' Did WaitMem time out?
If m_spel.TW() Then
    MsgBox "memory bit time out occurred"
End If
```

C# Example:

```
// Wait for memory bit 1 to be 1 (True)
// Max time is 5 seconds
m_spel.WaitMem(1, True, 5);
// Did WaitMem time out?
if (m_spel.TW())
    MessageBox.Show("memory bit time out occurred”);
```

15.3.21.3 WaitSw Method

Description

Waits for input bit status to change.

Syntax

Sub WaitSw (BitNumber As Integer, Condition As Boolean, Timeout As Single)
Sub WaitSw (Label As String, Condition As Boolean, Timeoutl As Single)

Parameters

- BitNumber
  An integer expression indicating the input bit number.
- Label
  A character string indicating the input bit label.
- Condition
  A Boolean expression indicating the input bit state.
- Timeout
  Maximum wait time (s)

Remarks

You should always check if a time out occurred by using the TW method. See the example below.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

WaitMem Method, **SetPoint Method**

WaitSw Example

VB Example:

```
Const PartPresent = 1
m_spel.WaitSw(PartPresent, True, 5)
If m_spel.TW() Then
    MsgBox "Part present time out occurred"
End If
```

C# Example:

```
const int PartPresent = 1;
m_spel.WaitSw(PartPresent, True, 5);
if (m_spel.TW())
    MessageBox.Show("Part Present time out occurred”);
```

15.3.21.4 WaitTaskDone Method

Description

Waits for a task to finish and returns the status.

Syntax

Function WaitTaskDone (TaskNumber As Integer) As SpelTaskState
Function WaitTaskDone (TaskName As String) As SpelTaskState

Parameters

- TaskNumber
  The task number for which to return the execution status
- TaskName
  A string expression indicating the task name.

Return Value

A SpelTaskState value.

See Also

SpelTaskState Enumeration, **TasksExecuting Method**, **TaskState Method**, **Xqt Method**

WaitTaskDone Example

VB Example:

```
Dim taskState As SpelTaskState
m_spel.Xqt 2, "mytask"
'
' Some processing here
'
taskState = m_spel.WaitTaskDone(2)
```

C# Example:

```
SpelTaskState taskState;
m_spel.Xqt(2, "mytask");
//
// Some processing here
//
taskState = m_spel.WaitTaskDone(2);
```

15.3.21.5 Weight Method

Description

Specifies the weight parameters for the current robot.

Syntax

Sub Weight (PayloadWeight As Single, ArmLength As Single)

Parameters

- PayloadWeight
  The weight that the hand can carry in kilograms.
- ArmLength
  The distance from the center of rotation of the second arm to the center of gravity of the hand in millimeters.
- Axis
  Additional axis (S or T) to which the load is assigned

> ⚠ Note
>
> Do not enter integer values to PayLoadWeight and ArmLength parameters. Use Single variables or directly enter Single type values.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Inertia Method, **Jrange Method**, **Tool Method**, **SetPoint Method**

Weight Example

VB Example:

```
m_spel.Weight(2.0F, 2.5F)
```

C# Example:

```
m_spel.Weight(2.0F, 2.5F);
```

15.3.22 X

15.3.22.1 Xqt Method

Description

Start one SPEL+ task.

Syntax

Sub Xqt (FuncName As String [, TaskType As SpelTaskType])
Sub Xqt (TaskNumber As Integer, FuncName As String [, TaskType As SpelTaskType])

Parameters

- TaskNumber
  Task number of the task to run (1 to 32)
- FuncName
  The name of the function to be executed. It is possible to pass arguments to the function. Arguments must be in parenthesis, separated by commas. For details, see the SPEL+ - Xqt Statement and see the example.
- TaskType
  Optional. Specifies the task type as Normal, NoPause, or NoEmgAbort.

Remarks

When Xqt is executed, control will return immediately to the calling program. Use the Call method to wait for a task to complete, or you can use EventReceived with the task state event to wait for a task to finish.

See Also

Call Method, **EnableEvent Method**, **EventReceived Event**

Xqt Example

VB Example:

```
m_spel.Xqt(2, "conveyor")

' Supply an argument to the RunPart function
m_spel.Xqt(3, "RunPart(3)")

Dim funcToExec As String
funcToExec = "RunPart(" & partNum & ")"
m_spel.Xqt(3, funcCall)
```

C# Example:

```
m_spel.Xqt(2, "conveyor");

// Supply an argument to the RunPart function
m_spel.Xqt(3, "RunPart(3)");

string funcToExec;
funcToExec = string.Format("RunPart({0})", partNum);
m_spel.Xqt(3, funcToExec);
```

15.3.22.2 XYLim Method

Description

Sets the permissible motion range limits for the Manipulator.

Syntax

Sub XYLim (XLowerLimit As Single, XUpperLimit As Single, YLowerLimit As Single, YUpperLimit As Single [, ZLowerLimit As Single] [, ZUpperLimit As Single])

Parameters

- XLowerLimit
  The minimum X coordinate position the robot can move to. (The robot cannot move to a position with an X coordinate less than this minimum X value.)
- XUpperLimit
  The maximum X coordinate position the robot can move to. (The robot cannot move to a position with an X coordinate larger than this maximum X value.)
- YLowerLimit
  The minimum Y coordinate position the robot can move to. (The robot cannot move to a position with a Y coordinate less than this minimum Y value.)
- YUpperLimit
  The maximum Y coordinate position the robot can move to. (The robot cannot move to a position with a Y coordinate larger than this maximum Y value.)
- ZLowerLimit
  The minimum Z coordinate position the robot can move to. (The robot cannot move to a position with a Z coordinate less than this minimum Z value.) Optional.
- ZUpperLimit
  The maximum Z coordinate position the robot can move to. (The robot cannot move to a position with a Z coordinate larger than this maximum Z value.) Optional.

Remarks

XYLim is used to define motion range limits. Many robot systems allow users to define joint limits but the SPEL+ language allows both joint limits and motion range limits to be defined. In effect this allows users to create a work envelope for their application.
The motion range established with XYLim values applies to motion command target positions only, and not to motion paths from starting position to target position. Therefore, the arm may move outside the XYLim range during motion. (i.e. The XYLim range does not affect Pulse.)
To turn off motion range limits, specify 0 for the range limit parameters.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the remarks in the SetPoint method.

See Also

Jrange Method, **SetPoint Method**

XYLim Example

VB Example:

```
m_spel.XYLim(0, 0, 0, 0)
```

C# Example:

```
m_spel.XYLim(0, 0, 0, 0);
```

15.3.22.3 XYLimClr Method

Description

Clears (undefines) the XYLim definition.

Syntax

Sub XYLimClr ()

See Also

XYLim Method, **XYLimDef Method**

XYLimClr Example

VB Example:

```
m_spel.XYLimClr()
```

C# Example:

```
m_spel.XYLimClr();
```

15.3.22.4 XYLimDef Method

Description

Returns whether XYLim has been defined or not.

Syntax

Function XYLimDef () As Boolean

Return Value

True if XYLim is defined, False if not.

See Also

XYLim Method, **XYLimClr Method**

XYLimDef Example

VB Example:

```
Dim xyLimDefined As Boolean
xyLimDefined = m_spel.XYLimDef()
```

C# Example:

```
bool xyLimDefined;
xyLimDefined = m_spel.XYLimDef();
```

15.4 Spel Class Events

15.4.1 EventReceived Event

Description

Occurs when Epson RC+ 8.0 sends a system event or when a program running in SPEL+ sends an event using a SPELCom_Event statement.

Syntax

EventReceived (ByVal sender As Object, ByVal e As RCAPINet.SpelEventArgs)

Parameters

- e.Event
  A number indicating the specified user-defined event.
- e.Message
  A character string indicating the event message.

Remarks

There are several system events that Epson RC+ 8.0 issues. The following table describes them.

System Events

Some events are disabled by default. To use these events you must first enable them using the EnableEvent Method.

Event Number	Event Message	Constant	Description
1	"PAUSE"	SpelEvents.Pause	Occurs when tasks are paused. Enabled by default.
2	"SAFE GUARD OPEN"	SpelEvents.SafeGuardOpen	Occurs when safe guard is open. Enabled by default.
3	"SAFE GUARD CLOSE"	SpelEvents.SafeGuardClose	Occurs when safe guard is closed. Enabled by default.
4	Project build status text	SpelEvents.ProjectBuildStatus	Each build status message is sent during the BuildProject method. CRLFs are added as needed. These messages are the same ones displayed on the Project Build Status window in Epson RC+ 8.0 GUI. This event must be enabled with the EnableEvent method. Disabled by default.
5	"Error xxx!: mmm in task at line yyy"	SpelEvents.Error	Occurs when a task is aborted due to an unhandled error or a system error is generated. Enabled by default.
6	Text from print statement	SpelEvents.Print	Occurs when a Print statement executes from a SPEL+ task. Disabled by default.

Event Number	Event Message	Constant	Description
7	"ESTOP ON"	SpelEvents.EStopOn	Occurs when emergency stop condition changes to ON. Enabled by default.
8	"ESTOP OFF"	SpelEvents.EStopOff	Occurs when emergency stop condition changes to OFF. Enabled by default.
9	"CONT"	SpelEvents.Continue	Occurs after a Cont has been executed. Enabled by default.
10	<Robot #>,"MOTOR ON"	SpelEvents.MotorOn	Occurs when motors go ON for the robot indicated. Disabled by default.
11	<Robot #>,"MOTOR OFF"	SpelEvents.MotorOff	Occurs when motors go OFF for the robot indicated. Disabled by default.
12	<Robot #>,"POWER HIGH"	SpelEvents.PowerHigh	Occurs when power goes HIGH for the robot indicated. Disabled by default.
13	<Robot #>,"POWER LOW"	SpelEvents.PowerLow	Occurs when power goes LOW for the robot indicated. Disabled by default.
14	"TEACH MODE"	SpelEvents.TeachMode	Occurs when teach mode is activated. Enabled by default.
15	"AUTO MODE"	SpelEvents.AutoMode	Occurs when auto mode is activated. Enabled by default.
16	"<TaskID>,<Status>, <FuncName>" Status: "RUN", "HALT", "PAUSE", “FINISHED”, “ABORTED”	SpelEvents.TaskState	Occurs when task state changes. Disabled by default.
17	"SHUTDOWN"	SpelEvents.Shutdown	Occurs when RC+ is shutting down. Disabled by default.
18	"ALL TASKS STOPPED"	SpelEvents.AllTasksStopped	Occurs when all tasks have been stopped. Disabled by default.
19	"DISCONNECTED"	SpelEvents.Disconnected	Occurs when Controller communication has been disconnected from the PC. When enabled, RC+ does not display a message box indicating disconnection. Disabled by default.

Event Number	Event Message	Constant	Description
20	“MOTION STARTED”	SpelEvents.MotionStarted	Occurs when the control command started If enabled, it will be displayed when the following Spel class methods are executed. Arc, Arc3, BGo, BMove, CVMove, Go, Home, JTran, Jump, Jump3, Jump3CP, MCal, Move, Pass, PTran, Pulse, TGo, TMove Disabled by default
21	“MOTION COMPLETE”	SpelEvents.MotionComplete	Occurs when the control command started If enabled, it will be displayed after the execution of the following Spel class methods are completed. Arc, Arc3, BGo, BMove, CVMove, Go, Home, JTran, Jump, Jump3, Jump3CP, MCal, Move, Pass, PTran, Pulse, TGo, TMove Disabled by default

User Events

You can send events from your SPEL+ program to your Visual Basic application using the SPELCom_Event command.

```
Spelcom_Event 3000, cycNum, lotNum, cycTime
```

When this statement executes, the EventReceived routine will be called with the event number and message. See Epson RC+ 8.0 Online Help or 13. SPELCom_Event for details on SPELCom_Event.

Use Example

VB Example:

```
Sub m_spel_EventReceived ( _
         ByVal sender As Object, _
         ByVal e As RCAPINet.SpelEventArgs) _
         Handles m_spel.EventReceived
    Redim tokens(0) As String
    Select Case e.Event
        Case 3000
          tokens = e.Message.Split(New [Char]() {" "c}, _

                   System.StringSplitOptions.RemoveEmptyEntries)
            lblCycCount.Text = tokens(0)
            lblLotNumber.Text = tokens(1)
            lblCycTime.Text = tokens(2)
    End Select
End Sub
```

C# Example:

```
public void m_spel_EventReceived(object sender, SpelEventArgs e)
{
    string[] tokens = new string[3];
    switch ((int) e.Event){
        case 3000:
            tokens = e.Message.Split(' ');
            lblCycCount.Text = tokens(0);
            lblLotNumber.Text = tokens(1);
            lblCycTime.Text = tokens(2);
            break;
```

```
        default:
            break;
    }
}
```

Handling Events

When EventReceived is called from the Spel class instance, the Epson RC+ 8.0 process server will wait for the event handling routine to finish. Therefore, you should never try to execute any RC+ API commands or perform long running processing from within the EventReceived routine. If you want to execute commands based on an event that occurred, set a flag in EventReceived and handle the flag from the main loop of your application, outside of the event handling function.
For example, in your Visual Basic main form Load procedure, you can create an event loop that receives events from SPEL+. In the spel_EventReceived routine, set global flags to indicate which events were received. Then, you can execute an actual event handling from the event loop created in Load procedure.

To display event message

Add a TextBox control to a form. Each time the event is received, you can display the event message in the Text property of the TextBox control.

VB Example:

```
Private Sub m_spel_EventReceived(ByVal sender As Object, _
    ByVal e As SpelEventArgs) Handles m_spel.EventReceived
    txtEvents.AppendText(e.Event & ": " & e.Message & vbCrLf)
End Sub
```

C# Example:

```
private void m_spel_EventReceived(object sender, SpelEventArgs e)
{
   txtEvents.AppendText(e.Event + ": " + e.Message);
}
```

See Also

EnableEvent Method

15.5 SPELVideo Control

Description

This control allows you to display video from Vision system. For details on how to use this control, see Displaying Video.

Displaying Video

File Name

RCAPINt2.dll

15.6 SPELVideo Control Properties

This control supports the properties listed below in addition to standard .NET component properties, such as Left, Top, Width, and Height. See the Visual Basic on-line Help for documentation on the standard properties.

- Camera
- GraphicsEnabled
- VideoEnabled

15.6.1 Camera Property

Description

Sets/gets the camera number to display video from. This is useful when you want to display video during jogging operations, live video monitoring, etc. If you are using the control to display graphics for vision sequences, then when the sequence is run, the camera number for the sequence will be used instead of this property value.

Syntax

Property Camera As Integer

Default Value

0 – any camera is displayed

Return value

Integer value containing the current camera number

See Also

VideoEnabled Property, **GraphicsEnabled Property**

Examples

VB Example:

```
SpelVideo1.Camera = 1
```

C# Example:

```
SpelVideo1.Camera = 1;
```

15.6.2 GraphicsEnabled Property

Description

Sets / returns whether vision graphics are displayed after a sequence is run. In order to see graphics, you must attach the control to a Spel class instance using the SPELVideo Control property. This property can be set "on the fly" so that graphics can be turned on/off while sequences are being run.

Syntax

Property GraphicsEnabled As Boolean

Default Value

False

Return Value

True if vision graphics are displayed, False if not.

See Also

Camera Property, **VideoEnabled Property**

Examples

VB Example:

```
SpelVideo1.GraphicsEnabled = True
```

C# Example:

```
SpelVideo1.GraphicsEnabled = true;
```

15.6.3 VideoEnabled Property

Description
Determines whether video is displayed.

Syntax
Property VideoEnabled As Boolean

Default Value
False

Return Value
True if video is displayed, False if not.

See Also
Camera Property, **GraphicsEnabled Property**

Examples
VB Example:

```
SpelVideo1.VideoEnabled = True
```

C# Example:

```
SpelVideo1.VideoEnabled = true;
```

15.7 SPELVideo Control Methods

15.7.1 LoadImage Method

Description

Loads an image from a file for display.

Syntax

Sub LoadImage (Path As String)

Parameters

- Path
  The full path name of the file to load the image from.

Remarks

Use LoadImage to load a previously saved image for display. The file extension must be BMP, TIF, or JPG.

See Also

VSaveImage Method

LoadImage Example

VB Example:

```
m_spelVideo.LoadImage("c:\RejectImages\reject001.bmp")
```

C# Example:

```
m_spelVideo.LoadImage(@"c:\RejectImages\reject001.bmp");
```

15.8 SPELVideo Control Events

All of the events for this control are standard .NET events. See the Visual Basic on-line Help for details.

15.9 SpelConnectionInfo Class

Member name	Type	Description
ConnectionIPAddress	String	IP address as configured in Epson RC+. * When the type of the connection is USB or virtual Controller, ConnectionIPAddress is blank.
ConnectionNumber	Integer	The number of the connection as configured in Epson RC+.
ConnectionName	String	The name of the connection as configured in Epson RC+.
ConnectionType	SpelConnectionType	The type of the connection as configured in Epson RC+.

Here is an example.
VB Example:

```
Dim connectionInfo() As SpelConnectionInfo
connectionInfo = m_spel.GetConnectionInfo()
```

C# Example:

```
SpelConnectionInfo[] info = m_spel.GetConnectionInfo();
```

15.10 SpelControllerInfo Class

Member name	Type	Description
ProjectName	String	The name of the project in the Controller.
ProjectID	String	The unique project ID of the project in the Controller.
Options	List[SpelOptionInfo]	The list of options in the controller.

Here is an example.
VB Example:

```
Dim info As RCAPINet.SpelControllerInfo
info = m_spel.GetControllerInfo()
Label1.Text = info.ProjectID + " " + info.ProjectName
```

C# Example:

```
SpelControllerInfo info;
info = m_spel.GetControllerInfo();
Label1.text = info.ProjectID + " " + info.ProjectName;
```

15.11 SpelException Class

The SpelException class is derived from the ApplicationException class. It adds an ErrorNumber property and some constructors.
Here is an example, showing how to retrieve the error number and the error message.

VB Example:

```
Try
  m_spel.Go(1)
Catch (ex As RCAPINet.SpelException)
  MsgBox(ex.ErrorNumber & ": " & ex.Message)
End Try
```

C# Example:

```
try{
  m_spel.Go(1);
}
catch(RCAPINet.SpelException ex){
  MessageBox.Show(string.Format("{0}: {1}", ex.ErrorNumber, ex.Message));
}
```

15.11.1 SpelException Properties

ErrorNumber As Integer

15.11.2 SpelException Methods

Sub New ()
The default constructor.

Sub New (Message As String)
The optional constructor that specifies an error message.

Sub New (ErrorNumber As Integer, Message As String)
The optional constructor that specifies the error number and associated message.

Sub New (Message As String, Inner As Exception)
The optional constructor that specifies the error message and inner exception.

Sub New (ErrorNumber As Integer, Message As String, Inner As Exception)
The optional constructor that specifies the error number, error message, and inner exception.

15.12 SpelOptionInfo Class

Member name	Type	Description
Name	String	Option name
Status	SpelOptionStatus	Option setting status

Here is an example.
VB Example:

```
Dim info As SpelControllerInfo
info = m_spel.GetControllerInfo()
Label1.Text = info.Options(1).Name + " " + info.Options(1).Status.ToString()
```

C# Example:

```
SpelControllerInfo info;
info = m_spel.GetControllerInfo();
Label1.Text = info.Options[1].Name + " " + info.Options[1].Status;
```

15.13 SpelPoint Class

The SpelPoint class can be used in several motion methods and also in the GetPoint and SetPoint methods of Spel class.
Here are some Visual Basic examples:

1:

```
Dim pt As New RCAPINet.SpelPoint(25.5, 100.3, -21, 0)
m_spel.Go(pt)
```

2:

```
Dim pt As New RCAPINet.SpelPoint
pt.X = 25.5
pt.Y = 100.3
pt.Z = -21
m_spel.Go(pt)
```

3:

```
Dim pt As New RCAPINet.SpelPoint
pt = m_spel.GetPoint("P*")
pt.Y = 222
m_spel.Go(pt)
```

Here are some Visual C# examples:
1:

```
SpelPoint pt = new SpelPoint(25.5, 100.3,  -21, 0);
m_spel.Go(pt);
```

2:

```
SpelPoint pt = new SpelPoint();
pt.X = 25.5;
pt.Y = 100.3;
pt.Z = -21;
m_spel.Go(pt);
```

3:

```
SpelPoint pt = new SpelPoint();
pt = m_spel.GetPoint("P0");
pt.Y = 222;
m_spel.Go(pt);
```

15.13.1 SpelPoint Properties

VB Example:
X As Single
Y As Single
Z As Single
U As Single
V As Single
W As Single
R As Single
S As Single
T As Single
Hand As SpelHand
Elbow As SpelElbow
Wrist As SpelWrist
Local As Integer
J1Flag As Integer
J2Flag As Integer
J4Flag As Integer
J6Flag As Integer
J1Angle As Single
J4Angle As Single

C# Example:
float x
float y
float z
float u
float v
float w
float r
float s
float t
SpelHand Hand
SpelElbow Elbow
SpelWrist Wrist
int Local
int J1Flag
int J2Flag
int J4Flag
int J6Flag
float J1Angle
float J4Angle

15.13.2 SpelPoint Methods

Sub Clear ()

Clears all point data.

Sub New ()

The default constructor. Creates an empty point (all data is cleared).

Sub New (X As Single, Y As Single, Z As Single, U As Single [, V As Single] [, W As Single])

The optional constructor for a new point that specifies coordinates.

Function ToString([Format As String]) As String

Override for ToString that allows a Format to be specified. This returns the point as defined in SPEL+.

Format can be:

- Empty:
  Returns all the points with all their coordinates and attributes
  "XY" Returns "XY(…)"
  "XYST" Returns "XY(…) :ST(…)"

15.14 SpelRobotInfo Class

Member name	Type	Description
RobotModel	String	Robot model name.
RobotName	String	Robot name.
RobotNumber	Integer	Robot number.
RobotSerial	String	Robot serial number.
RobotType	SpelRobotType	Robot type.

Here is an example.
VB Example:

```
Dim info As SpelRobotInfo
info = m_spel.GetRobotInfo()
Label1.Text = info.RobotNumber + " " + info.RobotModel
```

C# Example:

```
SpelRobotInfo info;
info = m_spel.GetRobotInfo();
Label1.Text = info.RobotNumber + " " + info.RobotModel;
```

15.15 SpelTaskInfo Class

Member name	Type	Description
CPU	Integer	CPU Load factor for SPEL tasks.
ErrorCode	Integer	Error number.
StartTime	DateTime	Start date and time of the task.
TaskName	String	Task name.
TaskNumber	Integer	Task number.
State	SpelTaskState	Task status.

Here is an example.
VB Example:

```
Dim info As SpelTaskInfo
info = m_spel.GetTaskInfo(1)
Label1.Text = info.TaskNumber.ToString() + " " + info.State.ToString()
```

C# Example:

```
SpelTaskInfo info;
info = m_spel.GetTaskInfo(1);
Label1.Text = info.TaskNumber + " " + info.State;
```

15.16 Enumerations

15.16.1 SpelArmDefMode Enumeration

Member name	Value	Description
Rough	1	Define the arm using one posture.
Fine	1	Define the arm using two postures.

15.16.2 SpelArmDefType Enumeration

Member name	Value	Description
J2Camera	1	Define the arm for a J2 mounted camera.

15.16.3 SpelAxis Enumeration

Member name	Value	Description
X	1	X axis.
Y	2	Y axis.
Z	3	Z axis.
U	4	U axis.
V	5	V axis.
W	6	W axis.
R	7	R axis.
S	8	S axis.
T	9	T axis.

15.16.4 SpelBaseAlignment Enumeration

Member name	Value	Description
XAxis	0	Align with X axis.
YAxis	1	Align with Y axis.

15.16.5 SpelCalPlateType Enumeration

Member name	Value	Description
None	0	No calibration plate.
Large	1	Large calibration plate.
Medium	2	Medium calibration plate.

Member name	Value	Description
Small	3	Small calibration plate.
XSmall	4	Extra small calibration plate.

15.16.6 SpelConnectionType Enumeration

Member name	Value	Description
USB	1	USB connection.
Ethernet	2	Ethernet connection.
Virtual	3	Connection to virtual Controller.

15.16.7 SpelDialogs Enumeration

Member name	Value	Description	
RobotManager	1	ID for Tools	Robot Manager dialog
ControllerTools	2	ID for Tools	Controller dialog
VisionGuide	3	ID for Tools	Vision Guide dialog
ForceGuide	4	ID for Tools	Force Guide dialog
PartFeeding	5	ID for Tools	Part Feeding dialog

15.16.8 SpelElbow Enumeration

Member name	Value	Description
Above	1	Elbow orientation is above.
Below	2	Elbow orientation is below.

15.16.9 SpelEvents Enumeration

Member name	Value	Description
Pause	1	ID for pause event.
SafeguardOpen	2	ID for safeguard open event.
SafeguardClose	3	ID for safeguard close event.
ProjectBuildStatus	4	ID for project build status event.
Error	5	ID for error event.
Print	6	ID for print event.
EstopOn	7	ID for emergency stop on event.
EstopOff	8	ID for emergency stop off event.

Member name	Value	Description
Continue	9	ID for continue event.
MotorOn	10	ID for motor on event.
MotorOff	11	ID for motor off event.
PowerHigh	12	ID for power high event.
PowerLow	13	ID for power low event.
TeachMode	14	ID for teach mode event.
AutoMode	15	ID for auto mode event.
TaskState	16	ID for task state event.
Shutdown	17	ID for shutdown event.
AllTasksStopped	18	ID for all tasks stopped event.
Disconnected	19	ID for disconnection event.
MotionStarted	20	ID for control command start event.
MotionComplete	21	ID for control command complete event.

15.16.10 SpelForceCompareType Enumeration

Member name	Value	Description
LessOrEqual	0	Till is triggered when the force is less than or equal to the specified threshold.
GreaterOrEqual	1	Till is triggered when the force is greater than or equal to the specified threshold.

15.16.11 SpelForceProps Enumeration

Member name	Value	Description
EndStatus	28340	End status for a Force Guide sequence or object.
ForceCondOK	28440	Status of force-related end condition of a Force Guide object.
IOCondOK	28590	Status of I/O-related end condition of a Force Guide object.
PosCondOK	28860	Status of achievement of position-related end condition of a Force Guide object.
Time	29070	Execution time for a Force Guide sequence or object.
TimedOut	29080	Reached status to the timeout period of Force Guide object.
LastExecObject	30100	Name of the Force Guide object that was executed at the end.
MeasuredHeight	30500	Measured height of HeightInspect sequence.
FailedStatus	30510	The reason why the force guide sequence failed.
AvgForcesFx	100211	Average value of force in Fx direction during execution of a Force Guide object.
AvgForcesFy	100212	Average value of force in Fy direction during execution of a Force Guide object.

Member name	Value	Description
AvgForcesFz	100213	Average value of force in Fz direction during execution of a Force Guide object.
AvgForcesTx	100214	Average value of force in Tx direction during execution of a Force Guide object.
AvgForcesTy	100215	Average value of force in Ty direction during execution of a Force Guide object.
AvgForcesTz	100216	Average value of for.ce in Tz direction during execution of a Force Guide object
EndForcesFx	102411	Force in Fx direction at end of a Force Guide sequence or Force Guide object.
EndForcesFy	102412	Force in Fy direction at end of a Force Guide sequence or Force Guide object.
EndForcesFz	102413	Force in Fz direction at end of a Force Guide sequence or Force Guide object.
EndForcesTx	102414	Force in Tx direction at end of a Force Guide sequence or Force Guide object.
EndForcesTy	102415	Force in Ty direction at end of a Force Guide sequence or Force Guide object.
EndForcesTz	102416	Force in Tz direction at end of a Force Guide sequence or Force Guide object.
EndPosX	102421	Position at end of a Force Guide object (X coordinate).
EndPosY	102422	Position at end of a Force Guide object (Y coordinate).
EndPosZ	102423	Position at end of a Force Guide object (Z coordinate).
EndPosU	102424	Position at end of a Force Guide object (U coordinate).
EndPosV	102425	Position at end of a Force Guide object (V coordinate).
EndPosW	102426	Position at end of a Force Guide object (W coordinate).
PeakForcesFx	105711	Peak value of force in Fx direction during execution of a Force Guide sequence or object.
PeakForcesFy	105712	Peak value of force in Fy direction during execution of a Force Guide sequence or object.
PeakForcesFz	105713	Peak value of force in Fz direction during execution of a Force Guide sequence or object.
PeakForcesTx	105714	Peak value of force in Tx direction during execution of a Force Guide sequence or object.
PeakForcesTy	105715	Peak value of force in Ty direction during execution of a Force Guide sequence or object.
PeakForcesTz	105716	Peak value of force in Tz direction during execution of a Force Guide sequence or object.
TriggeredForcesFx	109411	Force in Fx direction when force-related end conditions of a Force Guide object are achieved.
TriggeredForcesFy	109412	Force in Fy direction when force-related end conditions of a Force Guide object are achieved.
TriggeredForcesFz	109413	Force in Fz direction when force-related end conditions of a Force Guide object are achieved.
TriggeredForcesTx	109414	Force in Tx direction when force-related end conditions of a Force Guide object are achieved.
TriggeredForcesTy	109415	Force in Ty direction when force-related end conditions of a Force Guide object are achieved.
TriggeredForcesTz	109416	Force in Tz direction when force-related end conditions of a Force Guide object are achieved.
TriggeredPosX	109421	Position for a Force Guide object when force-related end conditions are achieved (X coordinate).

Member name	Value	Description
TriggeredPosY	109422	Position for a Force Guide object when force-related end conditions are achieved (Y coordinate).
TriggeredPosZ	109423	Position for a Force Guide object when force-related end conditions are achieved (Z coordinate).
TriggeredPosU	109424	Position for a Force Guide object when force-related end conditions are achieved (U coordinate).
TriggeredPosV	109425	Position for a Force Guide object when force-related end conditions are achieved (V coordinate).
TriggeredPosW	109426	Position for a Force Guide object when force-related end conditions are achieved (Z coordinate).

15.16.12 SpelHand Enumeration

Member name	Value	Description
Righty	1	Hand orientation is righty.
Lefty	2	Hand orientation is lefty.

15.16.13 SpelIOLabelTypes Enumeration

Member name	Value	Description
InputBit	1	Specifies input bit.
InputByte	2	Specifies input byte.
InputWord	3	Specifies input word.
OutputBit	4	Specifies output bit.
OutputByte	5	Specifies output byte.
OutputWord	6	Specifies output word.
MemoryBit	7	Specifies memory bit.
MemoryByte	8	Specifies memory byte.
MemoryWord	9	Specifies memory word.
InputReal	10	Specifies real number input.
OutputReal	11	Specifies real number output.

15.16.14 SpelLocalDefType Enumeration

Member name	Value	Description
J5Camera	1	Defines the local for a J5 mounted camera.
J6Camera	2	Defines the local for a J6 mounted camera.
FixedUpwardCamera	3	Defines the local by using a fixed upward camera.

Member name	Value	Description
FixedDownwardCamera	4	Defines the local by using a fixed downward camera.

15.16.15 SpelOperationMode Enumeration

Member name	Value	Description
Auto	1	Epson RC+ 8.0 is in auto mode.
Program	2	Epson RC+ 8.0 is in program mode.

15.16.16 SpelOptions Enumeration

Member name	Value	Description
ECP	1	ECP option
API	2	RC+ API option
PCVision	3	PC Vision option
ConveyorTracking	5	Conveyor tracking option
GUIBuilder	6	GUI Builder option
OCR	7	OCR option
FieldbusMaster	8	Fieldbus master option
LegacyForceSensing	9	Force sensing option
PartFeeding	11	Part Feeding option
ThirdPartyForceSensors	13	Third party force sensor option
OPCUA	4	OPCUA option
VRT	10	VRT option
SafetyFunction	17	SafetyFunction option

15.16.17 SpelOptionStatus Enumeration

Member name	Value	Description
Inactive	0	Disable
Active	1	Enable

15.16.18 SpelRobotPosType Enumeration

Member name	Value	Description
World	0	Specifies world coordinates.
Joint	1	Specifies joint coordinates.

Member name	Value	Description
Pulse	2	Specifies pulses.

15.16.19 SpelRobotType Enumeration

Member name	Value	Description
Joint	1	Robot type is joint.
Cartesian	2	Robot type is Cartesian.
Scara	3	Robot type is SCARA.
Cylindrical	4	Robot type is Cylindrical.
SixAxis	5	Robot type is 6-axis.
RS	6	Robot type is SCARA RS series.
N	7	Robot type is N.

15.16.20 SpelShutdownMode Enumeration

Member name	Value	Description
ShutdownWindows	0	Windows will be shutdown.
RebootWindows	1	Windows will be rebooted.

15.16.21 SpelSimObjectType Enumeration

Member name	Value	Description
Unknown	-1	Object type is not set.
Layout	0	Layout object.
Part	1	Parts object.
MountedDevice	3	Arm mount device.

15.16.22 SpelSimProps Enumeration

Member name	Value	Description
PositionX	100	Position of X coordinate system.
PositionY	200	Position of Y coordinate system.
PositionZ	300	Position of Z coordinate system.
RotationX	400	Rotation angle of X axis.
RotationY	500	Rotation angle of Y axis.
RotationZ	600	Rotation angle of Z axis.

Member name	Value	Description
CollisionCheck	700	Sets enable/disable of collision detect.
CollisionCheckSelf	800	Sets enable/disable of self-collision detect of the robot.
Visible	900	State of display/non-display.
Type	1000	Types of objects.
HalfSizeX	1500	Length of Box object in X direction.
HalfSizeY	1600	Length of Box object in Y direction.
HalfSizeZ	1700	Length of Box object in Z direction.
HalfSizeHeight	1800	Height of Plane object.
HalfSizeWidth	1900	Width of Plane object.
PlaneType	2000	Type of Plane object.
Radius	2100	Radius of Sphere or Cylinder object.
Height	2200	Height of Cylinder object.
Name	2300	Names of objects.
Color	2400	Display color of objects.

15.16.23 SpelStopType Enumeration

Member name	Value	Description
StopNormalTasks	0	Stop only normal tasks (not background tasks).
StopAllTasks	1	Stop all tasks, including background tasks.

15.16.24 SpelTaskState Enumeration

Member name	Value	Description
Quit	0	Task is in the quit state.
Run	1	Task is in the run state.
Aborted	2	Task was aborted.
Finished	3	Task was finished.
Breakpoint	4	Task is at a breakpoint.
Halt	5	Task is in the halt state.
Pause	6	Task is in the pause state.
Step	7	Task is being stepped.
Walk	8	Task is being walked.
Error	9	Task is in the error state.

Member name	Value	Description
Waiting	10	Task is in the wait state.

15.16.25 SpelTaskType Enumeration

Member name	Value	Description
Normal	0	Task is a normal task.
NoPause	1	Task is not affected by pause.
NoEmgAbort	2	Task is not affected by emergency stop.

15.16.26 SpelToolDefType Enumeration

Member name	Value	Description
J4Camera	1	Define the tool for a J4 mounted camera.
J6Camera	2	Define the tool for a J6 mounted camera.
FixedCamera	3	Define the tool by using the fixed camera which is not calibrated.
FixedCameraWithCal	4	Define the tool by using the upward camera which is calibrated.

15.16.27 SpelToolDefType3D Enumeration

Member name	Value	Description
Bar	1	Define a 3D tool for a bar.
Plane	2	Define a 3D tool for a plane.

15.16.28 SpelUserRights Enumeration

Member name	Value	Description
All	-1	User has all rights.
None	0	User has no rights.
EditSecurity	1	User can configure security.
SysConfig	2	User can change system configuration.
EditPrograms	4	User can edit programs.
EditPoints	8	User can edit points.
EditVision	16	User can change vision properties.
JogAndTeach	32	User can Jog & Teach.
CommandWindow	64	User can use the command window.
EditRobotParameters	128	User can edit robot parameters.Change robot parameters.

Member name	Value	Description
ConfigureOptions	256	User can configure options.
ViewAudit	512	User can view the security audit log.
EditProject	1024	User can edit the project configuration.
DeleteAudit	2048	User can delete security audit log entries.
TeachPoints	4096	User can teach points.
ChangeOutputs	8192	User can change output status.
ChangeMemIO	16384	User can change memory I/O status.
EditGUIBuilder	32768	User can make changes in GUI Builder.
EditForce	65536	User can make changes in Force Guide and Force Control.
EditPartFeeding	131072	User can make changes in Part Feeding.

15.16.29 SpelVDefShowWarning Enumeration

Member name	Value	Description
None	-1	Do not display a warning.
Always	0	Always display a warning.
DependsOnSpeed	1	Display when either RobotSpeed or RobotAccel is larger than 5.

15.16.30 SpelVisionImageSize Enumeration

Member name	Value	Description
Size320x240	1	320 x 240 image size.
Size640x480	2	640 x 480 image size.
Size800x600	3	800 x 600 image size.
Size1024x768	4	1024 x 768 image size.
Size1280x1024	5	1280 x 1024 image size.
Size1600x1200	6	1600 x 1200 image size.
Size2048x1536	7	2048 x 1536 image size.
Size2560x1920	8	2560 x 1920 image size.
Size3664x2748	9	3664 x 2748 image size.
Size5472x3648	10	5472 x 3648 image size.
Size4024x3036	11	4024 x 3036 image size.
Size2448x2048	12	2448 x 2048 image size.

15.16.31 SpelVisionObjectTypes Enumeration

Member name	Value	Description
Correlation	1	Correlation object.
Blob	2	Blob object.
Edge	3	Edge object.
Polar	4	Polar object.
Line	5	Line object.
Point	6	Point object.
Frame	7	Frame object.
ImageOp	8	ImageOp object.
OCR	9	OCR object
CodeReader	10	CodeReader object.
Geometric	11	Geometric object.
ColorMatch	14	Color Match object.
LineFinder	15	Line Finder object.
ArcFinder	16	Arc Finder object.
DefectFinder	17	Defect Finder object.
LineInspector	18	Line Inspector object.
ArcInspector	19	Arc Inspector object.
BoxFinder	20	Box Finder object
CornerFinder	21	Corner Finder object
Contour	22	Contour object
Text	23	Text object
Decision	26	Decision object
Coordinates	27	Coordinates object
GeometricPlus	28	GeometricPlus object
GeometricFusion	29	GeometricFusion object
GeometricNeural	30	GeometricNeural object

15.16.32 SpelVisionProps Enumeration

This enumeration is for all vision properties and results. Refer to the Vision Guide Reference manual for details.

15.16.33 SpelWindows Enumeration

Member name	Value	Description
IOMonitor	1	ID for the I/O Monitor window.
TaskManager	2	ID for the Task Manager window.
ForceMonitor	3	ID for the Force Monitor window.
Simulator	4	ID for the Simulator window.

15.16.34 SpelWrist Enumeration

Member name	Value	Description
NoFlip	1	Wrist orientation is no flip.
Flip	2	Wrist orientation is flip.

15.17 Spel Error Numbers and Messages

For error numbers and error messages, see the Status Code / Error Code List.

16. 32 Bit and 64 Bit Applications

The RCAPINet library automatically supports 32 bit and 64 bit applications.

17. Using the LabVIEW VI Library

17.1 Overview

The new library of LabVIEW VI has the following features.

- High level interface to Epson RC+ 8.0 using VIs (Virtual Instruments).
- The user no longer needs to deal with the .NET interface to Epson RC+ 8.0 – it is handles automatically.
- Each Spel command is wrapped in an individual VI.
- The VIs are organized in several Tool Palettes.
- Supports both 32 bit and 64 bit LabVIEW applications.
- Supports LabVIEW versions 2009 and greater.

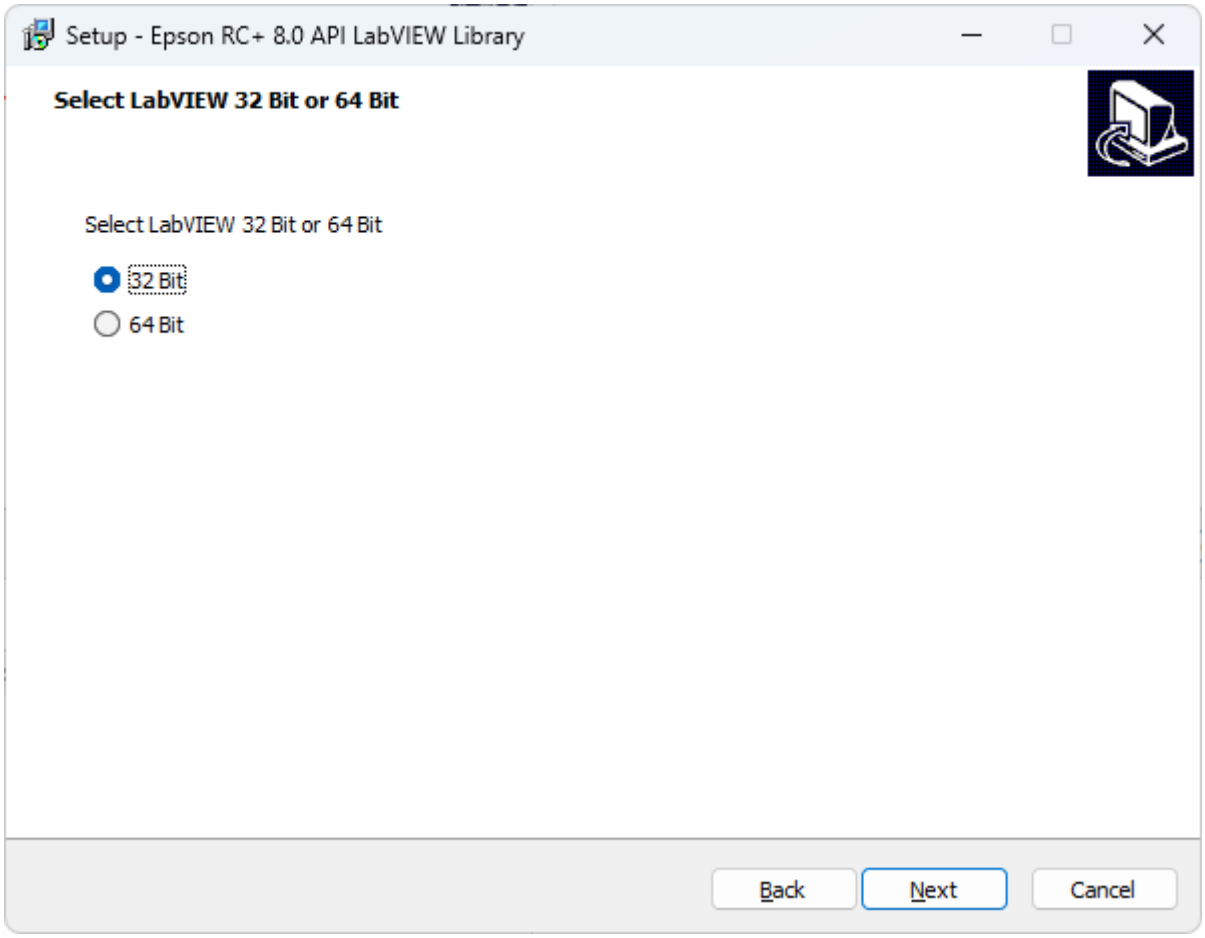
To use the LabVIEW VI library, you must purchase an license of RC+ API software for each Controller that you connect with.

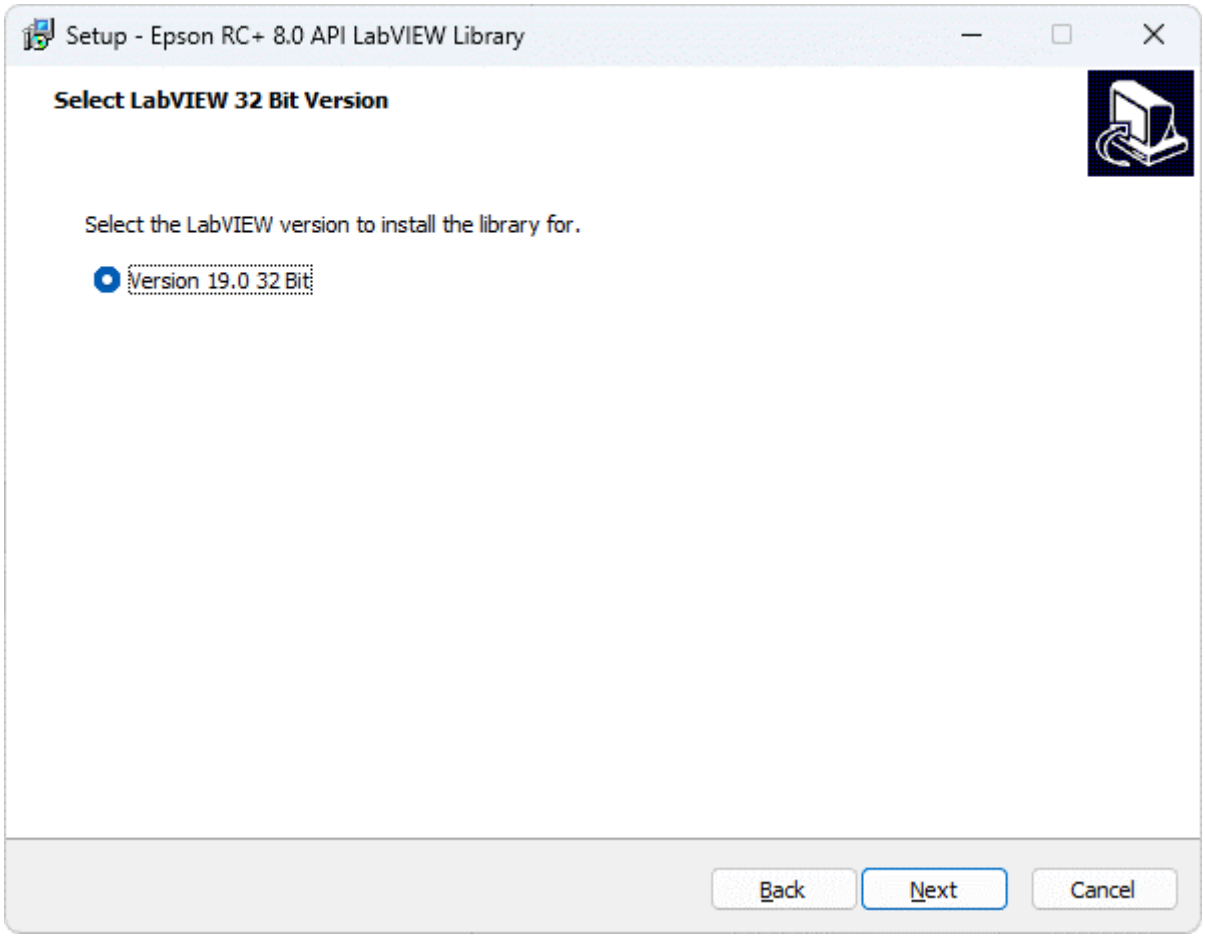
17.2 Installation

To use the Epson RC+ 8.0 LabVIEW VI Library, you must install it using the installer provided in the \EpsonRC80\API\LabVIEW folder on your PC.

1. Install LabVIEW version 2009 or greater.
2. Go to the \EpsonRC80\API\LabVIEW folder on your PC and run the EpsonRC80_vxxx_LabVIEW.exe installer. where xxx is the version number for Epson RC+ 8.0. For example, EpsonRC80_v800_LabVIEW.exe.
3. When you start the installer and select the bit for the LabVIEW application you are using, it will display the version installed on your PC. The latest version is selected by default. Select the version that will use the Epson RC+ 8.0 LabVIEW

VI Library.





4. Click Next, then click Install. The VIs, Controls, and palettes will be installed for the selected version of LabVIEW.

17.3 Tool and Control Palettes

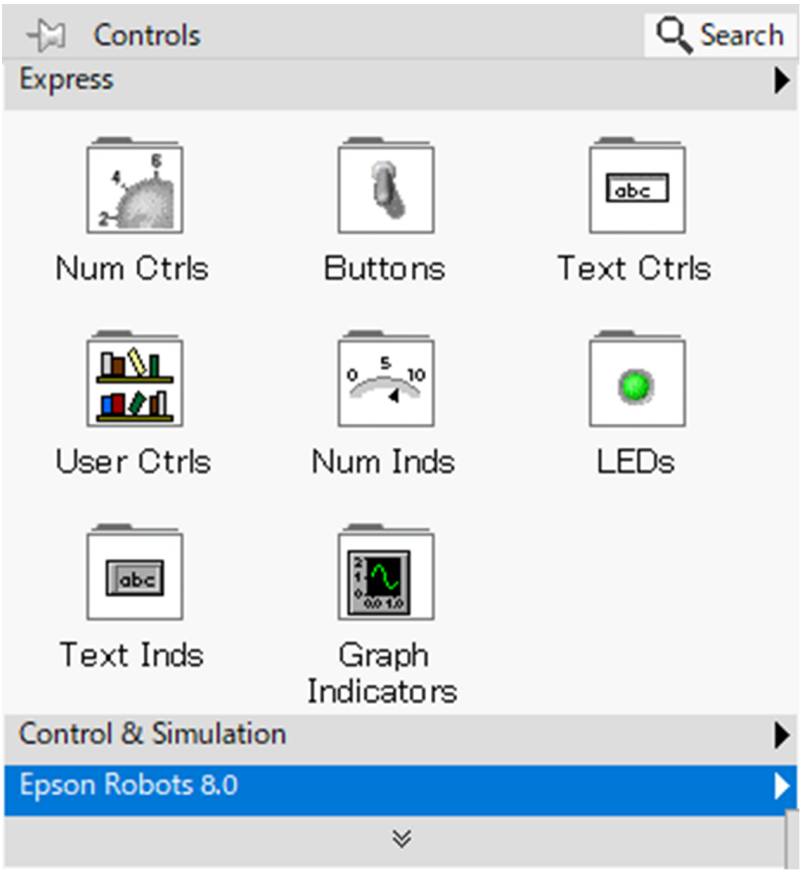
After the Epson RC+ 8.0 LabVIEW VI Library is installed, you can access the VIs and controls available in the library from the [Epson Robots 8.0] tool palette and [Epson Robots 8.0] control palette.

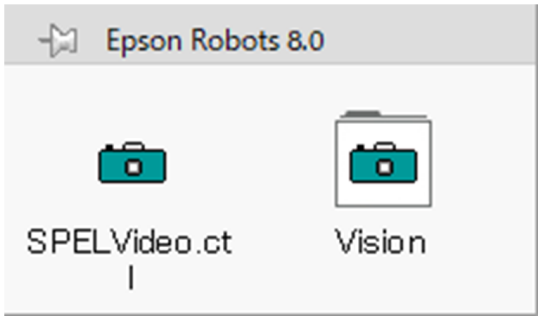
Tool Palette

The tool palette is accessed from the block diagram. Inside the Epson Robots 8.0 tool palette are several sub-palettes, described in the following table:

Palette	Description
System	Used to initialize and shutdown the API.
Robot Settings	Change robot parameters.
Points	Load, save, change robot points.
Motion	Execute robot motion.
Inputs & Outputs	Control and monitor Controller inputs and outputs.
Tasks	Manage tasks in the Robot Controller.
Variables	Read and write variables in the Controller.
Vision	Execute vision commands.
GUI	Display GUI functions.

To access the [Epson Robots 8.0] tool palette, open the block diagram for your VI, then right click on an empty area and select [Epson Robots 8.0] to see the sub-palettes described above.

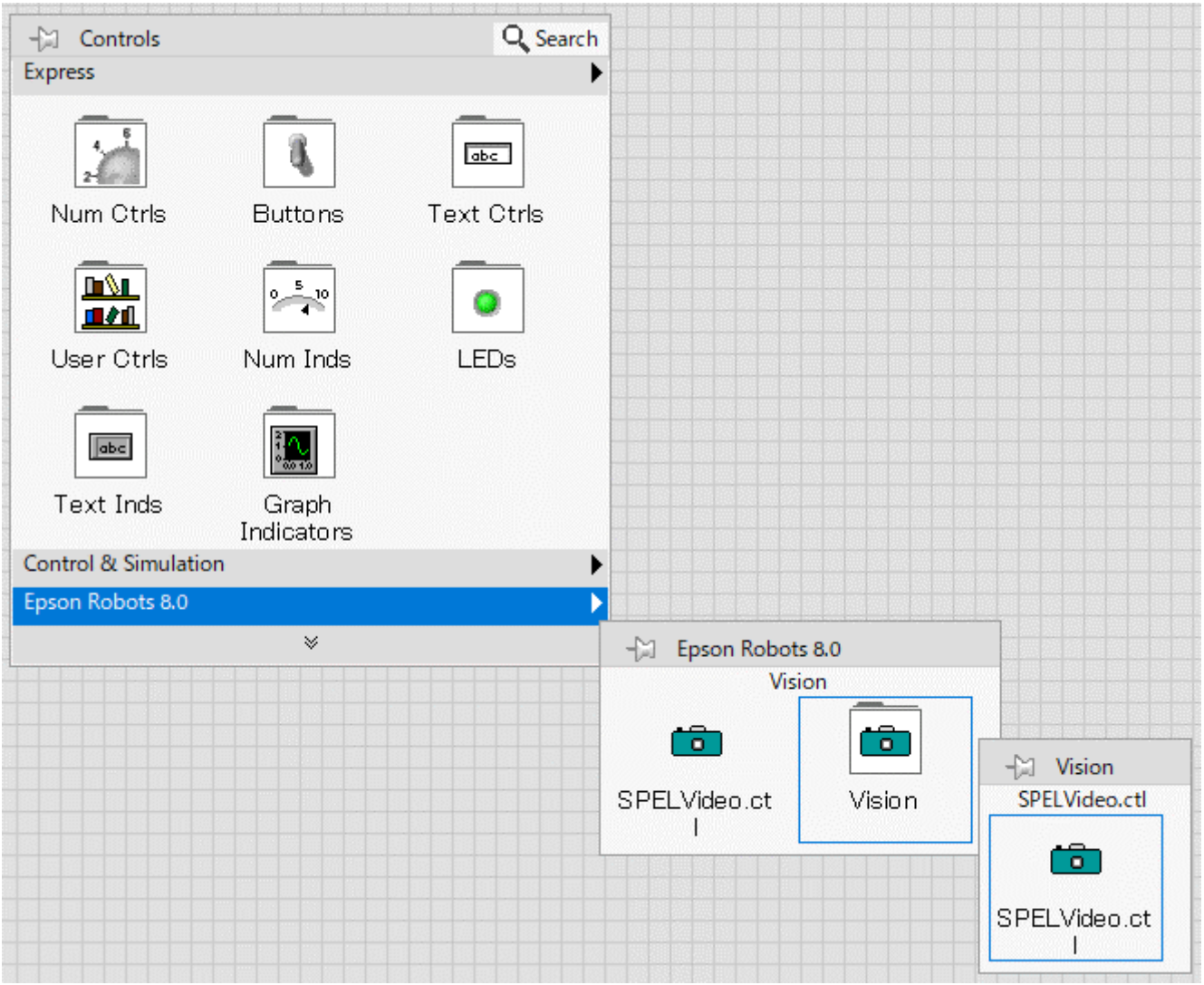




Control Palette

The control palette is accessed from the front panel.

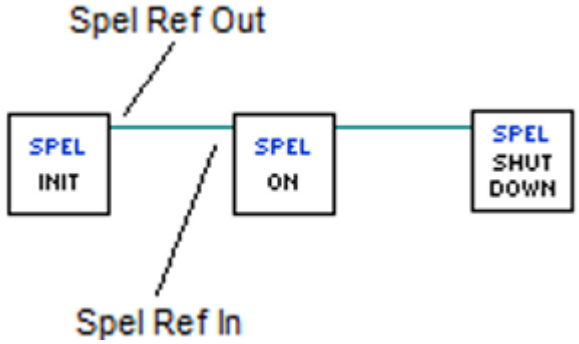
Palette	Description
Vision	Contains the SPELVideo control used to display video.



17.4 Getting started

To use the LabVIEW VI library, your application must first call the Spel Initialize VI for each Controller you want to use. The Initialize VI starts an Epson RC+ 8.0 server process that will connect to the Robot Controller and process the subsequent Spel command VIs. The Initialize VI has a Spel Ref Out output. This must be wired to the next Spel VI Spel Ref In input. Then for each subsequent VI, the Spel Ref Out output from a previous Spel VI must be wired to the Spel Ref In input of the next Spel

VI.
For example, the flow diagram below shows the wires for Spel Ref Out and Spel Ref In between each Spel node.



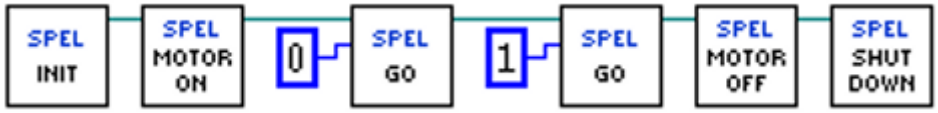
When the application is shutting down, you must call the Spel Shutdown VI. This will disconnect from the Robot Controller and shutdown the associated Epson RC+ 8.0 server process.
Follow these steps to get started. First, you will create two safe robot points from within the Epson RC+ 8.0 GUI for the LabVIEW default Spel+ project. Then you will build a small application in LabVIEW to move the robot between the two points.

1. Ensure that the Epson RC+ 8.0 and the Epson RC+ 8.0 LabVIEW VI Library are installed on your PC. See section for details for installing the LabVIEW VI Library.
   Installation
2. Start Epson RC+ 8.0.
3. From the Project menu, select Open, then navigate to the LabVIEW folder and select the LabVIEW_Default project. Click Open.
4. From the Tools menu, select Robot Manager. Click the Motor On button.
5. Select the Jog & Teach page on the Robot Manager. Jog the robot to some safe position.
6. Click Teach to teach point 0.
7. Jog the robot to another safe position.
8. Select P1 from the Point list, then click Teach to teach point 1.
9. Click the Save button on the main toolbar to save the points.
10. Close Epson RC+ 8.0.
11. Start LabVIEW and create a new VI.
12. Open the block diagram for the new VI.
13. From [Epson Robots 8.0] - [System] tool palette, drag an Init VI onto the block diagram. The Initialize VI is required for each Controller that you interface with.
14. From [Epson Robots 8.0] - [Robot Settings] tool palette, drag the MotorOn VI onto the block diagram. Wire the Spel Ref Out output from the Initialize VI to the Spel Ref In input on the MotorOn VI.
15. From [Epson Robots 8.0] - [Motion] tool palette, drag the Go VI onto the block diagram. Wire the Spel Ref Out output from the MotorOn VI to the Spel Ref In input on the Go VI. Add a constant for the Point Number input and set the value to 0.

16. From [Epson Robots 8.0] - [Motion] tool palette, drag another Go VI onto the block diagram. Wire the Spel Ref Out output from the previous Go VI to the Spel Ref In input on the second Go VI. Add a constant for the Point Number input and set the value to 1.

17. From [Epson Robots 8.0] - [Robot Settings] tool palette, drag the MotorOff VI onto the block diagram. Wire the Spel Ref Out output from the Go VI to the Spel Ref In input on the MotorOff VI.

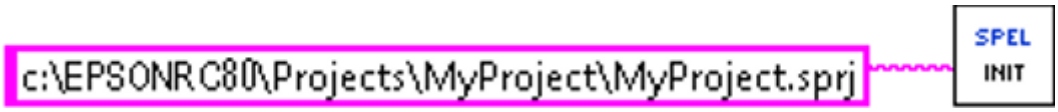
18. From [Epson Robots 8.0] - [System] tool palette, drag the Shutdown VI onto the block diagram. The Shutdown VI must be used for each Init VI.
    The block diagram should look similar to this:



19. Run the application. The robot motors should turn on, then the robot should move to point 0, then move to point 1, and then the robot motors will turn off.

17.5 Working with a Spel+ project

Using a Spel+ project with your LabVIEW application is optional. However, if you will be saving point data, or you want to use point labels and / or I/O labels, tasks, or vision sequences, then you will need to use a Spel+ project.
By default, the project is LabVIEW_Default, located in the \EpsonRC80\Projects\LabVIEW folder.
If desired, you can create your own projects using Epson RC+ 8.0, and then specify which project you want to use with the Initialize VI Project input parameter, as shown below:



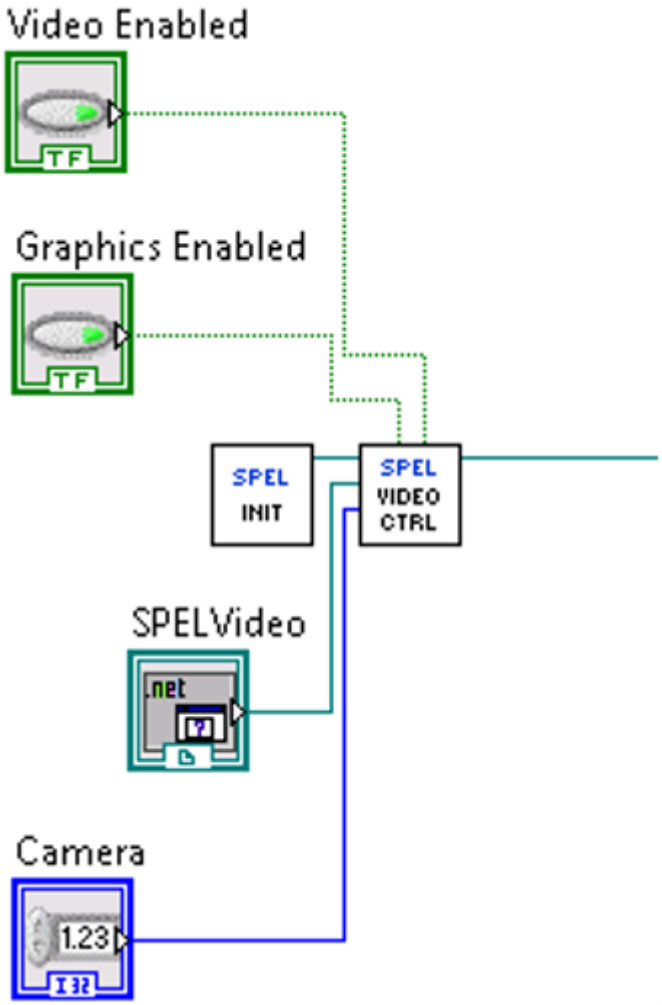
To work with Epson RC+ 8.0 projects, start the Epson RC+ 8.0 application. Use the Project menu to create, open, and edit projects. For more information, see the Epson RC+ User's Guide.

17.6 Displaying Video

You can display video for your Vision Guide sequences using the SPELVideo control and the VideoControl VI.
To display video:

1. Add a SPELVideo control to a front panel.
2. Add a VideoControl.vi to the corresponding block diagram.
3. Wire the output from the SPELVideo control to the SPELVideo Ref In input on the VideoControl VI.
4. Wire the Spel Ref In and Spel Ref Out parameters for the VideoControl VI.
5. Add constants or controls for the Camera, Graphics Enabled, and Video Enabled parameters on the VideoControl VI. Video Enabled must be set to true in order to display video.

The flow diagram below shows the connections for the SPELVideo control and the SPEL VideoControl VI.



When Video Enabled is true, and VRun executes from the VRun VI or in a Controller task, you will see the resulting video, depending on the Camera setting.

By default, the Camera input parameter is zero, which allows video from any camera to be displayed. If you set Camera to a number other than zero, then video will be displayed for sequences using the specified camera.

When Graphics Enabled is true, and VRun executes, then the sequence result graphics are displayed over the video image.

You can only use one SPEL Video control at a time in your application.

17.7 VI Reference

This section contains information for all VIs used in the Epson RC+ 8.0 LabVIEW VI Library.
The following information is provided for each VI:

Tool Palette	The tool palette where the VI is contained.
Description	Brief description of the function.
Inputs	Input parameters
Outputs	Output parameters
Remarks	Additional details.

17.7.1 A

17.7.1.1 Accel VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Sets the point to point acceleration and deceleration for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Accel	Integer value for point to point acceleration.
Decel	Integer value for point to point acceleration.
Depart Accel	Optional. Integer value for Jump depart acceleration.
Depart Decel	Optional. Integer value for Jump depart deceleration.
Appro Accel	Optional. Integer value for Jump approach acceleration.
Appro Decel	Optional. Integer value for Jump approach deceleration.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

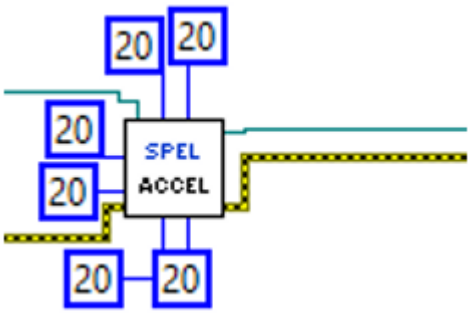
Use Accel to set the point to point acceleration and deceleration values for the current robot. All values can be from 1 to 100%. If Depart Accel is specified, then the remaining inputs must also be specified.

Usually, the maximum value is 100. However, some robots allow setting larger than 100. In general use, Accel setting 100 is the optimum setting that maintains the balance of acceleration and vibration when positioning. However, you may require an operation with high acceleration to shorten the cycle time by decreasing the vibration at positioning. In this case, set the Accel to larger than 100. Except in some operation conditions, the cycle time may not change by setting Accel to larger than 100.

See Also

AccelS VI, **Speed VI**, **SpeedS VI**

Accel Example



17.7.1.2 AccelS VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Sets the linear acceleration and deceleration for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Accel	Double value for linear acceleration.
Decel	Double value for linear acceleration.
Depart Accel	Optional. Double value for Jump depart acceleration.
Depart Decel	Optional. Double value for Jump depart deceleration.
Appro Accel	Optional. Double value for Jump approach acceleration.
Appro Decel	Optional. Double value for Jump approach deceleration.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

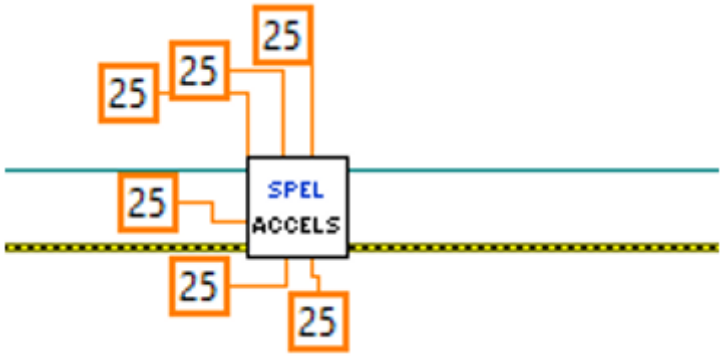
Use AccelS to set the linear acceleration and deceleration values for the current robot. All values are in millimeters / sec2. If Depart Accel is specified, then the remaining inputs must also be specified.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Speed VI**, **SpeedS VI**, **SetPoint VI**

AccelS Example



17.7.1.3 Arc VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Arc moves the arm to the specified point using circular interpolation in the XY plane.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Mid Point Number	Specifies the mid point by using an integer.
Mid Point Expr	Specifies the mid point by using a string expression. If this input is used, then you must also specify the end point with a string expression.
End Point Number	Specifies the end point by using an integer.
End Point Expr	Specifies the end point by using a string expression. You can include ROT, CP, SYNC, Till, Find, and a parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

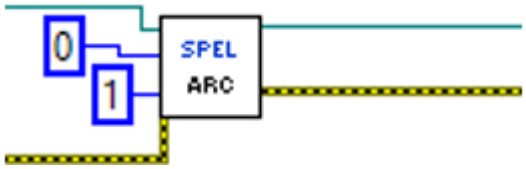
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc3 VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **TMove VI**, **BGo VI**, **Find VI**, **Till VI**, **SetPoint VI**

Arc Example



17.7.1.4 Arc3 VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Arc3 moves the arm to the specified point using circular interpolation in 3 dimensions.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Mid Point Number	Specifies the mid point by using an integer.
Mid Point Expr	Specifies the mid point by using a string expression. If this input is used, then you must also specify the end point with a string expression.
End Point Number	Specifies the end point by using an integer.
End Point Expr	Specifies the end point by using a string expression. You can include ROT, CP, SYNC, Till, Find, and a parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

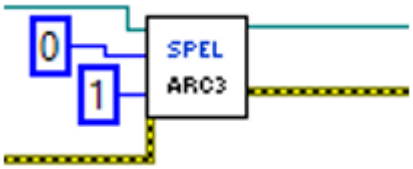
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

AccelS VI, **Arc VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **SpeedS VI**, **TGo VI**, **TMove VI**, **BGo VI**, **Find VI**, **Till VI**, **SetPoint VI**

Arc3 Example



17.7.1.5 Arch VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Defines ARCH parameters (Z height to move before beginning horizontal motion) for use with the JUMP instructions.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Arch Number	The depart distance in millimeters moved at the beginning of the Jump instruction before starting horizontal motion.
Depart Dist	The depart distance in millimeters moved at the beginning of the Jump instruction before starting horizontal motion.
Appro Dist	The approach distance in millimeters above the target position of the Jump instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

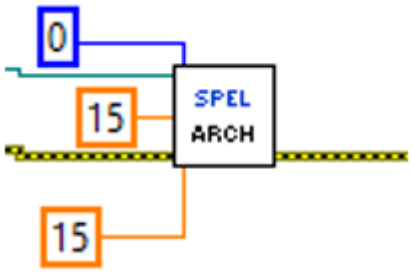
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Jump VI, **Jump3 VI**, **SetPoint VI**

Arch Example



17.7.1.6 Arm VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Selects the current robot arm.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Arm Number	Integer from 0-15. The user may select up to 16 different arms. Arm 0 is the standard (default) robot arm. Arm(s) 1-15 are auxiliary arms defined by the ArmSet instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Armset VI, **GetArm VI**, **Tool VI**

Arm Example

17.7.1.7 Armset VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Defines an auxiliary robot arm.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ArmNumber	Integer number: Valid range from 1-15.
Param1	(For SCARA Robots) The horizontal distance from the center line of the elbow joint to the center line of the new orientation axis. (I.E. the position where the new auxiliary arm's orientation axis center line is located.) (For Cartesian Robots) X axis direction position offset from the original X position specified in mm.
Param2	(For SCARA Robots) The offset (in degrees) between the line formed between the normal Elbow center line and the normal orientation Axis center line and the line formed between the new auxiliary arm elbow center line and the new orientation axis center line. (These 2 lines should intersect at the elbow center line and the angle formed is the Param2.) (For Cartesian Robots) Y axis direction position offset from the original Y position specified in mm.
Param3	(For SCARA & Cartesian Robots) The Z height offset difference between the new orientation axis center and the old orientation axis center. (This is a distance.)
Param4	(For SCARA Robots) The distance from the shoulder center line to the elbow center line of the elbow orientation of the new auxiliary axis. (For Cartesian Robots) This is a dummy parameter (Specify 0)
Param5	(For SCARA & Cartesian Robots) The angular offset (in degrees) for the new orientation axis vs. the old orientation axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

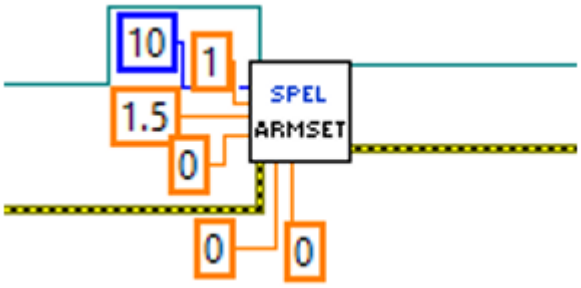
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

SetPoint VI

ArmSet Example



17.7.1.8 AtHome VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns True if the current robot is at the home position.

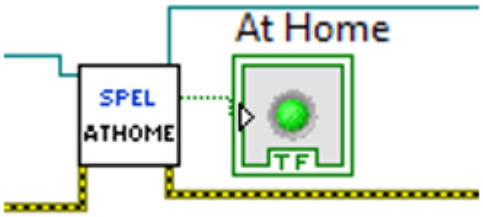
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
At Home	Boolean indicating if the current robot is at the home position.

AtHome Example



17.7.1.9 AvoidSing VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Enables / disables the singularity avoidance feature for Move, Arc, and Arc3 motion methods.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Enable	True enables singularity avoidance and False disables it.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

AvoidSing Example

17.7.2 B

17.7.2.1 BGo VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Executes Point to Point relative motion in the selected local coordinate system.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used. When using a string expression, you can include SYNC, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

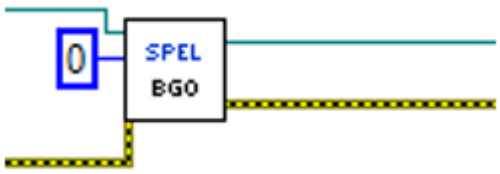
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **TMove VI**, **Find VI**, **Till VI**, **SetPoint VI**

BGo Example



17.7.2.2 BMove VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Executes linear interpolated relative motion in the selected local coordinate system

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used. When using a string expression, you can include ROT, SYNC, Till, Find, and parallel processing statement.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

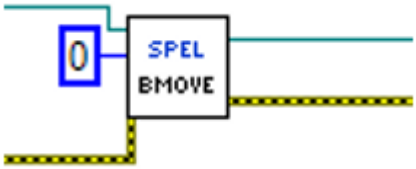
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

AccelS VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **SpeedS VI**, **TGo VI**, **TMove VI**, **Find VI**, **Till VI**, **SetPoint VI**

BMove Example



17.7.2.3 Box VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Specifies an approach check area defined within a box.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
AreaNumber	Integer number from 1-15 representing which of the 15 boxes to define.
Min X	The minimum X coordinate position of the approach check area.
Max X	The maximum X coordinate position of the approach check area.
Min Y	The minimum Y coordinate position of the approach check area.
Max Y	The maximum Y coordinate position of the approach check area.
Min Z	The minimum Z coordinate position of the approach check area.
Max Z	The maximum Z coordinate position of the approach check area.
Polarity On	Set the remote output logic when the corresponding remote output is used. To set I/O output to On when the end effector is in the box area, use True. To set I/O output to Off when the end effector is in the box area, use False.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

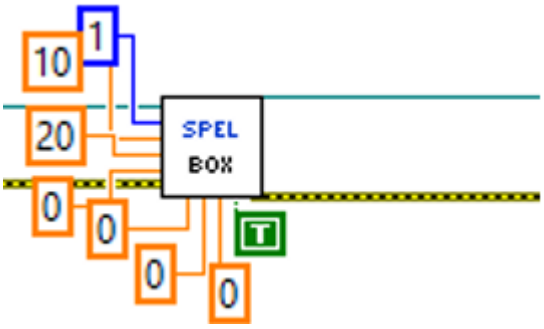
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

XYLim VI, **SetPoint VI**

Box Example



17.7.3 C

17.7.3.1 Continue VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Causes all tasks in the Controller that were paused to resume.

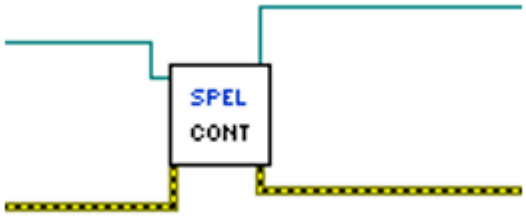
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Continue Example



17.7.4 D

17.7.4.1 Delay VI

Tool Palette

Epson Robots 8.0 | System

Description

Delays processing for the specified number of milliseconds.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Milliseconds	The number of milliseconds to delay.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Delay Example

17.7.5 E

17.7.5.1 ECP VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Selects the current ECP definition.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ECPNumber	Integer number from 0-15 representing which of 16 ECP definitions to use with the next motion instructions.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

ECPset VI, **GetECP VI**

ECP Example

17.7.5.2 ECPset VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Defines an ECP (external control point).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ECPNumber	Integer number from 1-15 representing which of 15 external control points to define.
X	The external control point X coordinate.
Y	The external control point Y coordinate.
Z	The external control point Z coordinate.
U	The external control point U coordinate.
V	The external control point V coordinate.
W	The external control point W coordinate.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

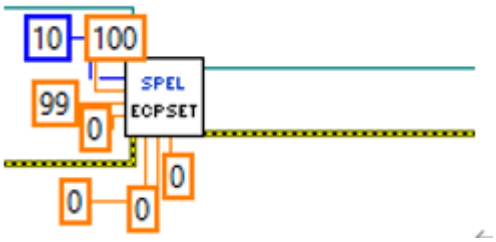
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

ECP VI, **GetECP VI**, **SetPoint VI**

ECPSet Example



17.7.5.3 EStopOn VI

Tool Palette

Epson Robots 8.0 | System

Description

Returns the Emergency Stop status.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

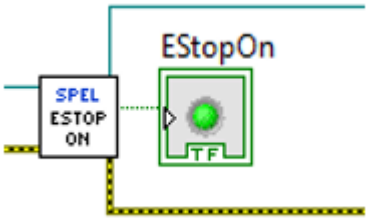
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
EStopOn	True if the status EmergencyStop, otherwise is False

See Also

SafetyOn VI

EStopOn Example



17.7.6 F

17.7.6.1 Find VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Specifies a condition for storing coordinates during motion.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Condition	String expression that contains functions and operators .

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

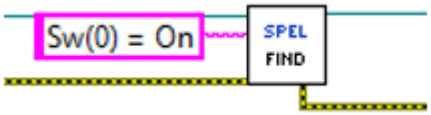
Remarks

Use Find to specify when a position should be stored during motion. When the condition is satisfied, the current position is stored in FindPos.

See Also

FindPos

Find Example



17.7.6.2 Fine VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Specifies and displays the positioning accuracy for target points.

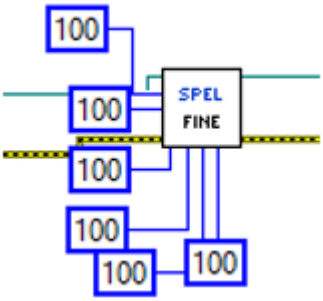
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
J1MaxErr - J9MaxErr	Integer number ranging from (0-32767) which represents the allowable positioning error for the each joint. The values for joints 7, 8, and 9 are optional.

Return Value

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Fine Example



17.7.7 G

17.7.7.1 GetArm VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Returns the current Arm number for the current robot.

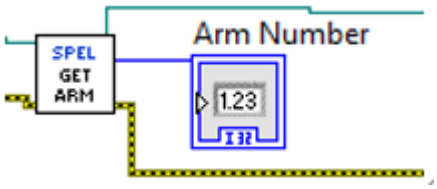
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Arm Number	The current arm number.

GetArm Example



17.7.7.2 GetAvoidSing VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns the state of AvoidSingularity.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

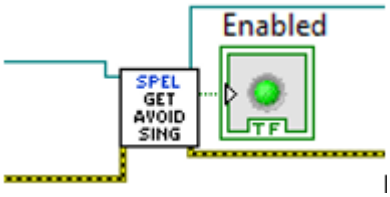
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
EStopOn	True if the status EmergencyStop, otherwise is False

See Also

AvoidSing VI

GetAvoidSing Example



17.7.7.3 GetECP VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Returns the current ECP number for the current robot.

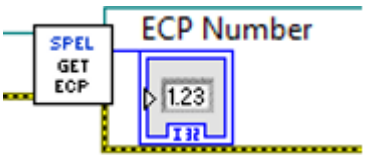
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
ECP Number	The current ECP number.

GetECP Example



17.7.7.4 GetLimZ VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns the current LimZ setting.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

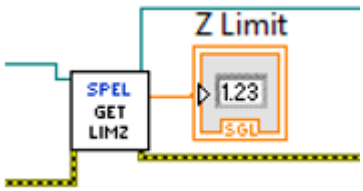
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Z Limit	The current LimZ setting.

See Also

LimZ VI

GetLimZ Example



17.7.7.5 GetMotor VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Returns the motor on status for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

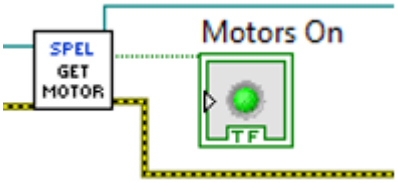
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Motors On	True if motors are on and false if not.

See Also

GetPower VI, **MotorOn VI**, **MotorOff VI**

GetMotor Example



17.7.7.6 GetOprMode VI

Tool Palette

Epson Robots 8.0 | System

Description

Reads the Epson RC+ 8.0 mode of operation.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.		
Error Out	Error condition output for subsequent Spel nodes.		
Operation Mode	The mode of operation for the associated Epson RC+ 8.0 server process.		
	Mode Setting	ID	Description
	Auto	1	Epson RC+ 8.0 is in auto mode.
	Program	2	Epson RC+ 8.0 is in program mode..

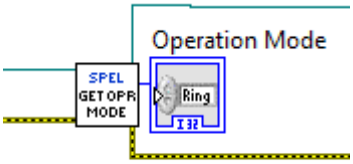
Remarks

When Operation Mode is set to Program, the Epson RC+ 8.0 GUI for the associated server process is opened and the Controller operation mode is set to Program. If the user closes the RC+ GUI, Operation Mode is set to Auto. If Operation Mode is set to Auto, the RC+ GUI also closes.

See Also

OprMode VI

GetOprMode Example



17.7.7.7 GetPoint VI

Tool Palette

Epson Robots 8.0 | Points

Description

Retrieves coordinate data for a robot point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used.

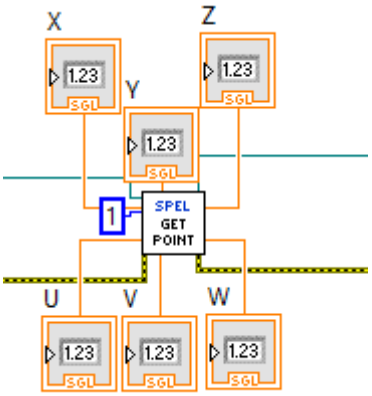
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
X - W	X, Y, Z, U, V, W coordinates of the specified point.

See Also

LoadPoints VI, **SavePoints VI**, **SetPoint VI**

GetPoint Example



17.7.7.8 GetPower VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Returns the power high status for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

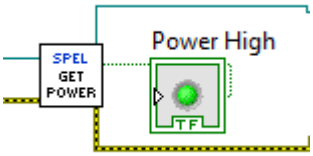
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Power High	True if power is high and false if not.

See Also

PowerHigh VI, **PowerLow VI**

GetPower Example



17.7.7.9 GetRobot VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Returns the current robot number.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

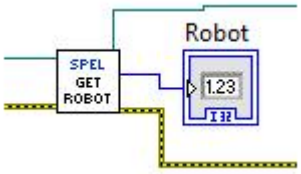
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Robot Number	The current robot number.

See Also

Robot VI

GetRobot Example



17.7.7.10 GetTool VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Returns the current Tool number for the current robot.

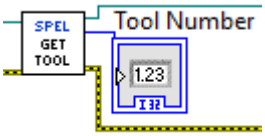
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Tool Number	The current tool number.

GetTool Example



17.7.7.11 GetVar VI

Tool Palette

Epson Robots 8.0 | Variables

Description

Returns the value of a SPEL+ global preserve variable in the Controller.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Var Name	The name of the SPEL+ global preserve variable. For an array, the entire array can be returned or just one element.

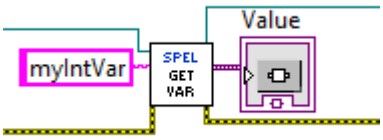
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	A variant containing the value.

See Also

SetVar VI

GetVar Example



17.7.7.12 Go VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Moves the arm in a Point to Point fashion from the current position to the specified point or XY position. The GO instruction can move any combination of the robot axes at the same time.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used. When using a string expression, you can include SYNC, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

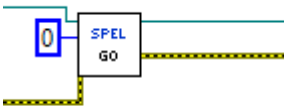
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **TMove VI**, **Find VI**, **Till VI**, **SetPoint VI**

Go Example



17.7.8 H

17.7.8.1 Halt VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Suspends execution of the specified task.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be suspended. The range of the task number is 1 to 32. If Task Name is specified, then Task Number is ignored.
Task Name	Optional. Specifies the name of the task to be suspended. If Task Name is not specified, then the Task Number input will be used.

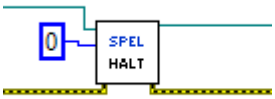
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Quit VI, **Resume VI**

Halt Example



17.7.8.2 Here VI

Tool Palette

Epson Robots 8.0 | Points

Description

Teaches a point at the current position.

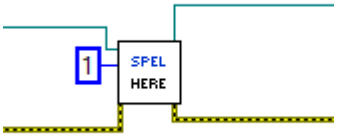
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Name	Optional. Specifies the name of the point. If Point Name is not specified, then the Point Number input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Here Example



17.7.8.3 HideWindow VI

Tool Palette

Epson Robots 8.0 | GUI

Description

Hides an Epson RC+ 8.0 window that was previously displayed with ShowWindow.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Window ID	The ID of the Epson RC+ 8.0 window to show.

Window name	ID	Description
IOMonitor	1	ID for the I/O Monitor window.
TaskManager	2	ID for the Task Manager window.
ForceMonitor	3	ID for the Force Monitor window.
Simulator	4	ID for the Simulator window.

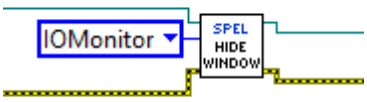
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

RunDialog VI, **ShowWindow VI**

HideWindow Example



17.7.9 I

17.7.9.1 In VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified input port. Each port contains 8 input bits (one byte).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. String containing an input byte label. If Label is specified, then Port Number is ignored.

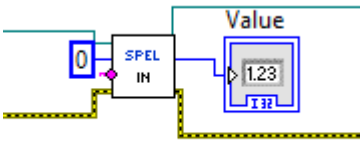
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 255 representing the status of the input port.

See Also

InBCD VI, **InW VI**, **Sw VI**

In Example



17.7.9.2 InBCD VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the input status of 8 inputs using BCD format. (Binary Coded Decimal)

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. String containing an input byte label. If Label is specified, then Port Number is ignored.

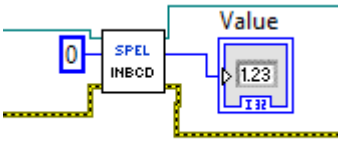
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 9 representing the status of the input port.

See Also

In VI, **InW VI**, **Sw VI**

InBCD Example



17.7.9.3 InsideBox VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns whether the current robot end effector is inside a specified box area.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Area Number	Integer number from 1-15 representing which of the 15 boxes to check.

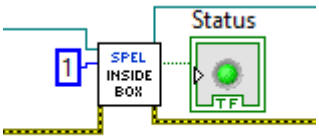
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Status	Boolean that is True if the robot end effector is inside the box.

See Also

Box VI, **InsidePlane VI**, **Plane VI**

InsideBox Example



17.7.9.4 InsidePlane VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns whether the current robot end effector is inside a specified plane.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
AreaNumber	Integer number from 1-15 representing which of the 15 boxes to check.

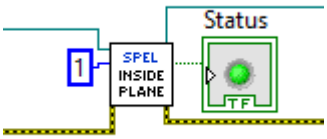
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Status	Boolean that is True if the robot end effector is inside the plane.

See Also

Box VI, **InsideBox VI**, **Plane VI**

InsidePlane Example



17.7.9.5 InW VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified input word port. Each word port contains 16 input bits.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. String containing an input byte label. If Label is specified, then Port Number is ignored.

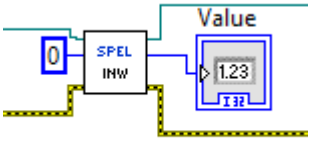
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer expression from 0 to 65535 representing the input port status.

See Also

In VI, **InBCD VI**, **Sw VI**

InW Example



17.7.9.6 Inertia VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Specifies the load inertia and eccentricity for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
LoadInertia	Double value that specifies total moment of inertia in kgm2 around the center of the end effector joint, including end effector and part.
Eccentricity	Double value that specifies eccentricity in mm around the center of the end effector joint, including end effector and part.

Return Value

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Weight VI, **SetPoint VI**

Inertia Example

17.7.9.7 Initialize VI

Tool Palette

Epson Robots 8.0 | System

Description

Initializes the instance of Spel used by the LabVIEW VI library.

Inputs

Server Product Type	Specifies which Epson RC+ product to interface with. Optional.
Connection Number	Specifies which Controller connection to use. Optional.
Project	Specifies the Epson RC+ project to be used. Optional.
ServerInstance	Specifies which instance of Epson RC+ server to use. Optional.
ConnectionPassword	A character string indicating the connection password.
If the controller requires a password for connection but the password has not been configured in the RC+ Setup \ [PC to Controller Communications] dialog, you must set the password to connect with the controller. Optional.	

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

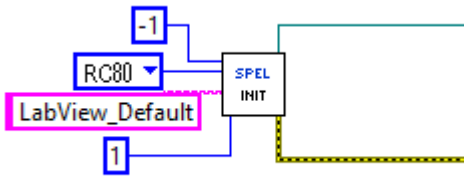
Remarks

The Initialize VI must be called for each instance of the library that will be used.
Server Product Type is used to specify which Epson RC+ product to use. The default is RC80 (Epson RC+ 8.0).
When Connection Number is not specified, then the connection last used in the Epson RC+ 8.0 will be used.
When Project is not specified, the default LabVIEW Epson RC+ 8.0 project will be used. The project must be used in the Epson RC+ product specified with Server Product Type.

See Also

Shutdown VI

Initialize Example



17.7.10 J

17.7.10.1 JRange VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Defines the permissible working range of the specified robot joint in pulses.

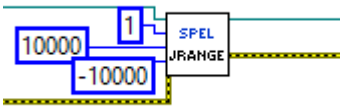
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
JointNumber	Integer number between 1 - 9 representing the joint for which JRange will be specified.
LowerLimitPulses	Integer number representing the encoder pulse count position for the lower limit range of the specified joint.
UpperLimitPulses	Integer number representing the encoder pulse count position for the upper limit range of the specified joint

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

JRange Example



17.7.10.2 JS VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Jump Sense detects whether the arm stopped prior to completing a JUMP instruction (which used a SENSE input) or if the arm completed the JUMP move. JS returns the Jump Sense status.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

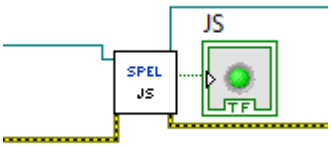
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
JS	True if the SENSE input was detected during motion, False if not.

See Also

Jump VI, **Sense VI**

JS Example



17.7.10.3 JTran VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Executes a relative joint move.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
JointNumber	The specific joint to move.
Distance	The distance to move. Units are in degrees for rotary joints and millimeters for linear joints.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

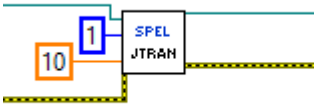
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

SetPoint VI

JTran Example



17.7.10.4 Jump VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Moves the arm from the current position to the specified point using point to point motion while first moving in a vertical direction up, then horizontally and then finally vertically downward to arrive on the final destination point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Specifies the end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used. When using a string expression, you can include SYNC, Sence, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

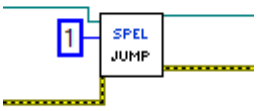
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Go VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **TMove VI**, **Find VI**, **Sense VI**, **Till VI**, **SetPoint VI**

Jump Example



17.7.10.5 Jump3 VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Motion with 3D gate using a combination of two CP motions and one PTP motion. The robot moves to the depart point, then the approach point, and finally the destination point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Depart Point Number	Specifies the depart point by using an integer.
Depart Point Expr	Specifies the depart point by using a string expression. If this input is used, then you must also specify the approach and destination points with string expressions.
Appro Point Number	Specifies the approach point by using an integer.
Appro Point Expr	Specifies the approach point by using a string expression. If this input is used, then you must also specify the depart and destination points with string expressions.
Dest Point Number	Specifies the destination point by using an integer.
Dest Point Expr	Specifies the destination point by using a string expression. If this input is used, then you must also specify the depart and approach points with string expressions.

When using a string expression, you can include ROT, SYNC, Sence, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

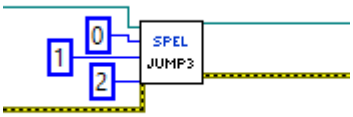
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Go VI**, **Jump VI**, **Move VI**, **Speed VI**, **TGo VI**, **TMove VI**, **Find VI**, **Sense VI**, **Till VI**, **SetPoint VI**

Jump3 Example



17.7.10.6 Jump3CP VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Motion with 3D gate using a combination of three CP motions. The robot moves to the depart point, then the approach point, and finally the destination point.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Depart Point Number	Specifies the depart point by using an integer.
Depart Point Expr	Specifies the depart point by using a string expression. If this input is used, then you must also specify the approach and destination points with string expressions.
Appro Point Number	Specifies the approach point by using an integer.
Appro Point Expr	Specifies the approach point by using a string expression. If this input is used, then you must also specify the depart and destination points with string expressions.
Dest Point Number	Specifies the destination point by using an integer.
Dest Point Expr	Specifies the destination point by using a string expression. If this input is used, then you must also specify the depart and approach points with string expressions.

When using a string expression, you can include ROT, SYNC, Sence, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

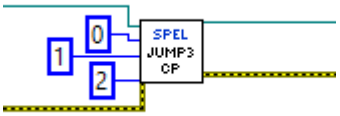
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **TMove VI**, **Find VI**, **Sense VI**, **Till VI**, **SetPoint VI**

Jump3CP Example



17.7.11 L

17.7.11.1 LimZ VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Sets the default value of the Z axis height for JUMP commands.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Z Limit	A coordinate value within the movable range of the Z axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

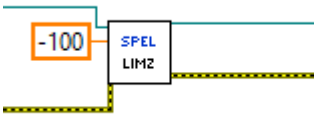
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Jump VI, **Jump3 VI**, **SetPoint VI**

LimZ Example



17.7.11.2 LoadPoints VI

Tool Palette

Epson Robots 8.0 | Points

Description

Loads a SPEL+ point file into the Controller's point memory for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
File Name	A valid point file that is in the current Spel project or was previously saved with the SavePoints VI.

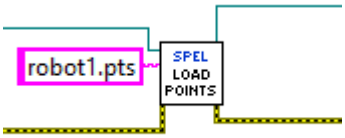
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetPoint VI, **Robot VI**, **SavePoints VI**, **SetPoint VI**

LoadPoints Example



17.7.12 M

17.7.12.1 MemIn VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified memory I/O byte port. Each port contains 8 memory I/O bits.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. String containing an input byte label. If Label is specified, then Port Number is ignored.

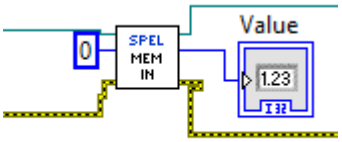
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 255 representing the status of the port.

See Also

MemInW VI, **MemOut VI**, **MemOutW VI**

MemIn Example



17.7.12.2 MemInW VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each word port contains 16 input bits. If Label is not specified, then Port Number is used.
Label	Optional. String containing an input byte label. If Label is specified, then Port Number is ignored.

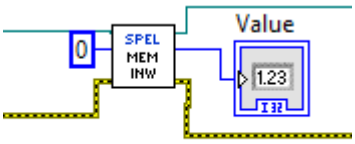
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Integer from 0 to 255 representing the status of the port.

See Also

MemIn VI, **MemOut VI**, **MemOutW VI**

MemInW Example



17.7.12.3 MemOut VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Simultaneously sets 8 memory I/O bits based on the 8 bit value specified by the user.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the input ports. Each port contains 8 input bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. String containing an input byte label. If Label is specified, then Port Number is ignored.
Value	Integer containing the output pattern for the specified byte. Valid values are from 0 - 255.

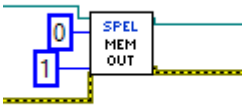
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemIn VI, **MemInW VI**, **MemOutW VI**

MemOut Example



17.7.12.4 MemOff VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Turns off the specified bit of the S/W memory I/O.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the memory I/O bits. If Label is not specified, then Bit Number is used.
Label	Optional. String containing an input bit label. If Label is specified, then Bit Number is ignored.

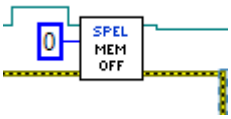
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOn VI, **MemOut VI**, **MemOutW VI**

MemOff Example



17.7.12.5 MemOn VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Turns on the specified bit of memory I/O.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the memory I/O bits. If Label is not specified, then Bit Number is used.
Label	Optional. String containing an input bit label. If Label is specified, then Bit Number is ignored.

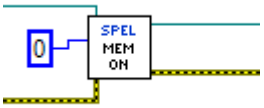
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOff VI, **MemOut VI**, **MemOutW VI**

MemOn Example



17.7.12.6 MemOut VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Simultaneously sets 8 memory I/O bits based on the 8 bit value specified by the user.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer expressing one of the memory I/O ports. Each port contains 8 memory I/O bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. String containing a memory I/O byte label. If Label is specified, then Port Number is ignored.
Value	Integer containing the output pattern for the specified byte. Valid values are from 0 - 255.

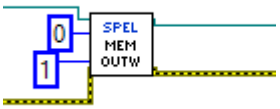
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOn VI, **MemOff VI**, **MemOutW VI**

MemOut Example



17.7.12.7 MemOutW VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Simultaneously sets 16 memory I/O bits based on the 16 bit value specified by the user.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. Integer representing one of the memory I/O ports. Each word port contains 16 input bits. If Label is not specified, then Port Number is used.
Label	Optional. String containing a memory I/O byte label. If Label is specified, then Port Number is ignored.
Value	Integer containing the output pattern for the specified word. Valid values are from 0 - 65535.

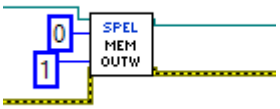
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MemOn VI, **MemOff VI**, **MemOut VI**

MemOutW Example



17.7.12.8 MemSw VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified memory I/O bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the memory I/O bits. If Label is not specified, then Bit Number is used.
Label	Optional. String containing a memory I/O bit label. If Label is specified, then Bit Number is ignored.

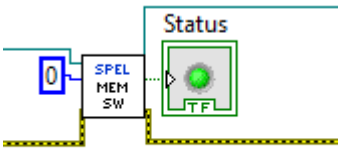
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Boolean that is True with the memory I/O bit is on.

See Also

MemIn VI, **MemInW VI**

MemSW Example



17.7.12.9 MotorOff VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Turns motors off for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

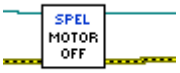
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOn VI, **PowerHigh VI**, **PowerLow VI**, **Robot VI**

MotorOff Example



17.7.12.10 MotorOn VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Turns motors on for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

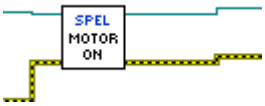
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOff VI, **PowerHigh VI**, **PowerLow VI**, **Robot VI**

MotorOn Example



17.7.12.11 Move VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Moves the arm from the current position to the specified point using linear interpolation (i.e. moving in a straight line).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used.

When using a string expression, you can include ROT, ECP, SYNC, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

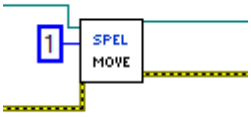
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

AccelS VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **SpeedS VI**, **TGo VI**, **TMove VI**, **ECP VI**, **Find VI**, **Till VI**, **SetPoint VI**

Move Example



17.7.13 O

17.7.13.1 Off VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Turns off the specified output bit.

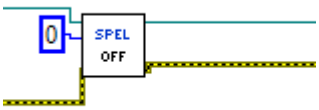
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Integer representing one of the output bits. If Label is not specified, Bit Number is used. Optional.
Label	String containing the input bit label. If Label is specified, Bit Number is ignored. Optional.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Off Example



17.7.13.2 On VI

Tool Palette

EpsonRobots 8.0 | Inputs & outputs

Description

Turns on the specified output bit.

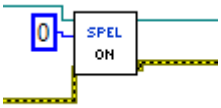
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the output bits. If Label is not specified, then Bit Number is used.
Label	Optional. String containing an input bit label. If Label is specified, then Bit Number is ignored.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

On Example



17.7.13.3 OPort VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified output bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Integer representing one of the output bits. If Label is not specified, Bit Number is used. Optional.
Label	String containing the output bit label. If Label is specified, Bit Number is ignored. Optional.

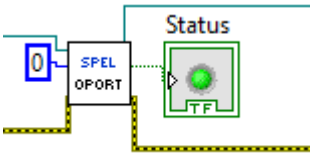
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Boolean that is True with the output bit is on.

See Also

In VI, **InW VI**, **On VI**, **Off VI**, **Out VI**, **Sw VI**

Oport Example



17.7.13.4 OprMode VI

Tool Palette

Epson Robots 8.0 | System

Description

Sets the Epson RC+ 8.0 mode of operation..

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Operation Mode	The mode of operation for the associated Epson RC+ 8.0 server process.

Mode	ID	Description
Auto	1	Epson RC+ 8.0 is in auto mode.
Program	2	Epson RC+ 8.0 is in program mode..

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

When Operation Mode is set to Program, the Epson RC+ 8.0 GUI for the associated server process is opened and the Controller operation mode is set to Program. If the user closes the RC+ GUI, Operation Mode is set to Auto. If Operation Mode is set to Auto, the RC+ GUI also closes.

See Also

GetOprMode VI

OprMode Example

17.7.13.5 Out VI

Tool and Control Palettes

Epson Robots 8.0 | Inputs & Outputs

Description

Sets eight output bits (one byte) at a time.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. An integer indicating the output port. Each port comprises 8 output bits (one byte). If Label is not specified, then Port Number is used.
Label	Optional. A character string indicating the output byte label. If Label is specified, Port Number is ignored.
Value	An integer indicating the specified byte output pattern. Valid range: 0 to 255

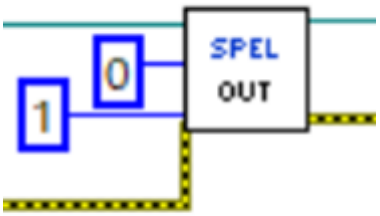
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

On VI, **Off VI**, **OutW VI**

Out Example



17.7.13.6 OutW VI

Tool and Control Palettes

Epson Robots 8.0 | Inputs & Outputs

Description

Sets 16 output bits (one word) at a time.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Port Number	Optional. An integer indicating the output port. Each word port comprises 16 output bits (one word). If Label is not specified, then Port Number is used.
Label	Optional. A character string indicating the output word label. If Label is specified, Port Number is ignored.
Value	An integer indicating the specified word output pattern. Valid range: 0 to 65535

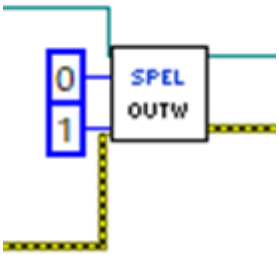
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

On VI, **Off VI**, **Out VI**

Out Example



17.7.14 P

17.7.14.1 Pause VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Causes all normal SPEL+ tasks in the Controller to pause. If the robot is moving, it will immediately decelerate to a stop.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

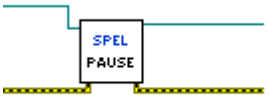
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Continue VI, **Stop VI**

Pause Example



17.7.14.2 Plane VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Defines a Plane.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Plane Number	Integer expression from 1-15 representing which of 15 planes to define.
X	The plane coordinate system origin X coordinate.
Y	The plane coordinate system origin Y coordinate.
Z	The plane coordinate system origin Z coordinate.
U	The plane coordinate system rotation about the Z axis.
V	The plane coordinate system rotation about the Y axis.
W	The plane coordinate system rotation about the X axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

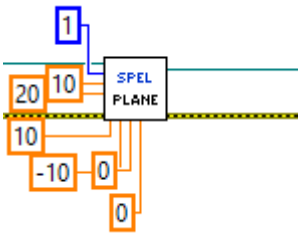
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Box VI, **InsideBox VI**, **InsidePlane VI**, **SetPoint VI**

Plane Example



17.7.14.3 PowerHigh VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Sets motor power to high for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOff VI, **MotorOn VI**, **PowerLow VI**, **Robot VI**

PowerHigh Example

17.7.14.4 PowerLow VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Sets motor power to low for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

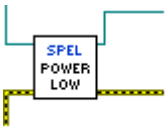
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

MotorOff VI, **MotorOn VI**, **PowerHigh VI**, **Robot VI**

PowerLow Example



17.7.15 Q

17.7.15.1 Quit VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Terminates execution of the specified task.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be terminated. The range of the task number is 1 to 32. If Task Name is specified, then Task Number is ignored.
Task Name	Optional. Specifies the name of the task to be terminated. If Task Name is not specified, then the Task Number input will be used.

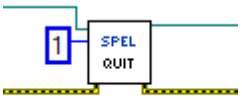
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Halt VI, **Resume VI**

Quit Example



17.7.16 R

17.7.16.1 Reset VI

Tool Palette

Epson Robots 8.0 | System

Description

Resets the Controller to the initialized state.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Reset Example

17.7.16.2 Resume VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Resumes a task which was suspended by the Halt VI.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be resumed. The range of the task number is 1 to 32. If Task Name is specified, then Task Number is ignored.
Task Name	Optional. Specifies the name of the task to be resumed. If Task Name is not specified, then the Task Number input will be used.

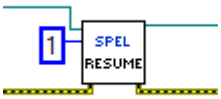
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Halt VI, **Quit VI**

Resume Example



17.7.16.3 Robot VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Selects the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Robot Number	Integer from 1-16.

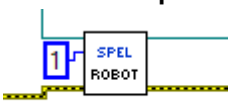
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetRobot VI, **MotorOff VI**, **MotorOn VI**, **PowerHigh VI**, **PowerLow VI**

Robot Example



17.7.16.4 RunDialog VI

Tool Palette

Epson Robots 8.0 | GUI

Description

Runs an Epson RC+ 8.0 dialog.

Inputs

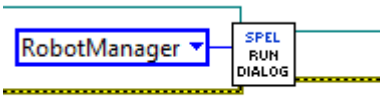
Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Dialog ID	The ID of the Epson RC+ 8.0 dialog to run.

Dialog name	ID	Description	
RobotManager	1	ID for Tools	Robot Manager dialog
ControllerTools	2	ID for Tools	Controller dialog
VisionGuide	3	ID for Tools	Vision Guide dialog

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

RunDialog Example



17.7.17 S

17.7.17.1 SafetyOn VI

Tool Palette

Epson Robots 8.0 | System

Description

Returns the Safety Door open status.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

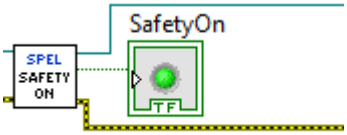
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
SafetyOn	True if the SafetyDoor is Open, otherwise is False

See Also

EStopOn VI

SafetyOn Example



17.7.17.2 SavePoints VI

Tool Palette

Epson Robots 8.0 | Points

Description

Save points for the current robot into a file.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
File Name	The name of a point file that is in the current Spel project or a new file that will be stored in the Controller.

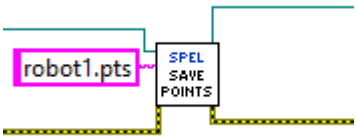
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetPoint VI, **LoadPoints VI**, **Robot VI**, **SetPoint VI**

SavePoints Example



17.7.17.3 Sense VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Specifies input condition that, if satisfied, completes the Jump in progress by stopping the robot above the target position.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Condition	Specifies the I/O condition using a string expression. For details see the Sense Statement in the SPEL+ Language Reference manual.

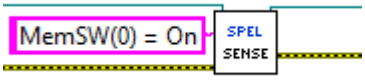
Outputs

sSpel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

JS VI, **Jump VI**

Sense Example



17.7.17.4 SetPoint VI

Tool Palette

Epson Robots 8.0 | Points

Description

Sets the coordinate data for a point for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Name is specified, then Point Number is ignored.
Point Name	Optional. Specifies the point by using a string expression for the point name. If Point Name is not specified, then the Point Number input will be used.
X - W	X, Y, Z, U, V, W coordinates of the specified point.

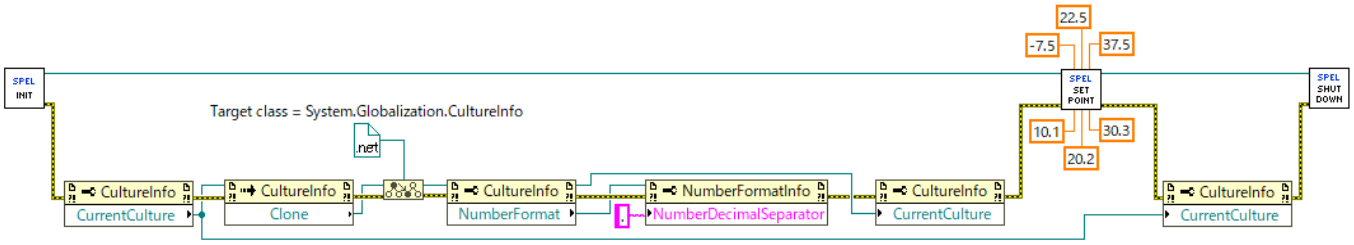
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, refer to the comments of the SetPoint method in RCAPINet Reference - Spel Class Methods.

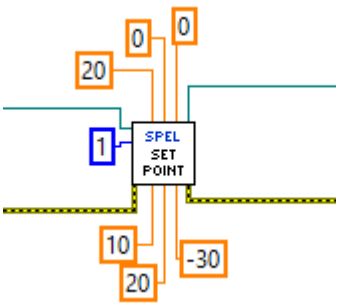
The CurrentCulture property settings can be changed using the LabVIEW Property Node (.NET) and Invoke Node (.NET). Below is an example of how to set the CurrentCulture in C# in the SetPoint method comments in LabVIEW.



See Also

GetPoint VI, LoadPoints VI, Robot VI, SavePoints VI

SetPoint Example



17.7.17.5 SetVar VI

Tool Palette

Epson Robots 8.0 | Variables

Description

Sets the value of a SPEL+ global preserve variable in the Controller.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Var Name	The name of the SPEL+ global preserve variable.
Value	A variant containing the value.

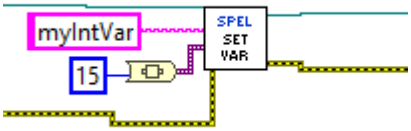
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

GetVar VI

SetVar Example



17.7.17.6 SFree VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Free joint the specified robot axes.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Axes	Optional. Integer array specifying which axes to free. If omitted, all axes are freed.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

If Axes is omitted, then all axes are freed.

See Also

MotorOff VI, **MotorOn VI**, **SLock VI**

SFree Example

17.7.17.7 ShowWindow VI

Tool Palette

Epson Robots 8.0 | GUI

Description

Displays an Epson RC+ 8.0 window.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Window ID	The ID of the Epson RC+ 8.0 window to show.

Window name	ID	Description
IOMonitor	1	ID for the I/O Monitor window.
TaskManager	2	ID for the Task Manager window.
ForceMonitor	3	ID for the Force Monitor window.
Simulator	4	ID for the Simulator window.

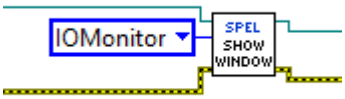
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

HideWindow VI, **RunDialog VI**

ShowWindow Example



17.7.17.8 Shutdown VI

Tool Palette

Epson Robots 8.0 | System

Description

Shuts down the Epson RC+ 8.0 server process that was started when the Initialize VI was called.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Error Out	Error condition output for subsequent Spel nodes.

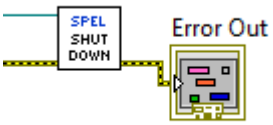
Remarks

The Shutdown VI must be called for each instance of the library. This will shutdown the associated Epson RC+ 8.0 server process.

See Also

Initialize VI

Shutdown Example



17.7.17.9 SLock VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Release the free joint state for the specified robot axes.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Axes	Optional. Integer array specifying which axes to lock. If omitted, all axes are locked.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

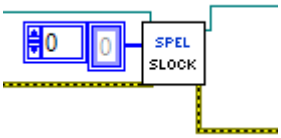
Remarks

If Axes is omitted, then all axes are locked.

See Also

MotorOff VI, **MotorOn VI**, **SFree VI**

SLock Example



17.7.17.10 Speed VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
PointToPoint Speed	Specifies the arm speed for use with the point to point instructions Go, Jump and Pulse.
Depart Speed	Integer number between 1-100 representing the Z axis upward motion speed for the Jump instruction.
Appro Speed	Integer number between 1-100 representing the Z axis downward motion speed for the Jump instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

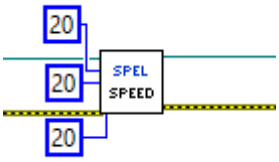
Remarks

Use Speed to set the point to point speed for the current robot. All values can be from 1 to 100%. If Depart Speed is specified, then Appro Speed must also be specified.

See Also

Accel VI, **AccelS VI**, **SpeedS VI**

Speed Example



17.7.17.11 SpeedS VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Linear Speed	Specifies the arm speed for use with the Continuous Path instructions Jump3CP, Move, Arc, and CVMove.
Depart Speed	Double value between 1-5000 representing the Z axis upward motion speed for the Jump3CP instruction.
Appro Speed	Double value between 1-5000 representing the Z axis downward motion speed for the Jump3CP instruction.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

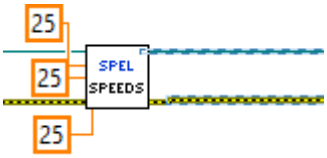
Use Speed to set the linear speed for the current robot in millimeters / sec. If Depart Speed is specified, then Appro Speed must also be specified.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **AccelS VI**, **Speed VI**, **SetPoint VI**

SpeedS Example



17.7.17.12 Start VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Starts a program that will run in the Controller.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ProgramNumber	The program number to start, corresponding to the 64 main functions in SPEL+ as shown in the table below. The range is 0 to 63.

Program Number	SPEL+ Function Name
0	main
1	main1
2	main2
3	main3
4	main4
5	main5
6	main6
...	...
63	main63

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

When Start is executed, control will return immediately to the calling VI. You cannot start a program that is already running. Note that Start causes global variables in the Controller to be cleared and default robot points to be loaded.

See Also

Continue VI, **Pause VI**, **Stop VI**, **Xqt VI**

Start Example

17.7.17.13 Stop VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Stops all normal SPEL+ tasks running in the Controller and optionally stops all background tasks.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Stop Type	Specify StopNormalTasks (default) or StopAllTasks (also stop background tasks). Optional.

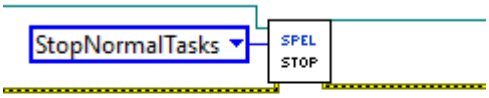
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Continue VI, **Pause VI**, **Start VI**, **Xqt VI**

Stop Example



17.7.17.14 Sw VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the specified input bit.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. Integer representing one of the input bits. If Label is not specified, then Bit Number is used.
Label	Optional. String containing an input bit label. If Label is specified, then Bit Number is ignored.

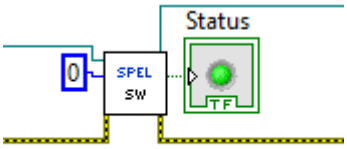
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	Boolean that is True with the output bit is on.

See Also

In VI, **InW VI**, **On VI**, **Off VI**, **OPort VI**, **Out VI**

SW Example



17.7.18 T

17.7.18.1 TargetOK VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Target OK	The robot can move to the target position.
At Home	Boolean indicating if the current robot is at the home position.

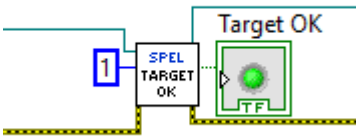
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

BGo VI, **Go VI**, **Jump VI**, **TGo VI**, **SetPoint VI**

TargetOK Example



17.7.18.2 TGo VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Executes Point to Point relative motion, in the current tool coordinate system.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used.

When using a string expression, you can include SYNC, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

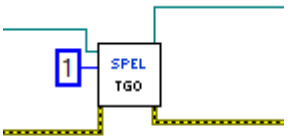
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TMove VI**, **Find VI**, **Till VI**, **SetPoint VI**

TGo Example



17.7.18.3 Till VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Specifies event condition that, if satisfied, completes the motion command (Jump, Go, Move, etc.) in progress by decelerating and stopping the robot at an intermediate position.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Condition	Specifies the I/O condition using a string expression. For details see the Till Statement in the SPEL+ Language Reference manual.

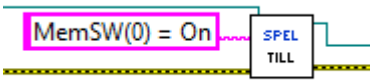
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **TillOn VI**, **TMove VI**

Till Example



17.7.18.4 TillOn VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Returns True if a stop has occurred from a till condition during the last Go/Jump/Move statement.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

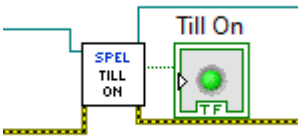
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Till On	True if the till condition was detected during motion. False if not.

See Also

Accel VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **Speed VI**, **TGo VI**, **Till VI**, **TMove VI**

TillOn Example



17.7.18.5 TLSet VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Defines an ECP (external control point).

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
ToolNumber	Integer expression from 1-15 representing which of 15 tools to define. (Tool 0 is the default tool and cannot be modified.)
X	The tool coordinate system origin X coordinate.
Y	The tool coordinate system origin Y coordinate.
Z	The tool coordinate system origin Z coordinate.
U	The tool coordinate system rotation about the Z axis.
V	The tool coordinate system rotation about the Y axis.
W	The tool coordinate system rotation about the X axis.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

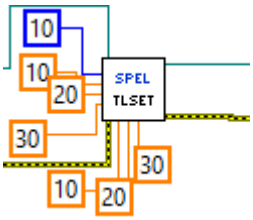
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

SetPoint VI

TLSet Example



17.7.18.6 TMove VI

Tool Palette

Epson Robots 8.0 | Motion

Description

Executes linear interpolation relative motion, in the current tool coordinate system.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Point Number	Optional. Specifies the target end point by using the point number for a previously taught point in the Controller's point memory for the current robot. If Point Expression is specified, then Point Number is ignored.
Point Expression	Optional. Specifies the target end point by using a string expression. If Point Expression is not specified, then the Point Number input will be used.

When using a string expression, you can include ROT, SYNC, Till, Find, and parallel processing statement.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

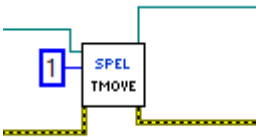
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

AccelS VI, **Arc VI**, **Arc3 VI**, **BGo VI**, **BMove VI**, **Go VI**, **Jump VI**, **Jump3 VI**, **Move VI**, **SpeedS VI**, **TGo VI**, **Find VI**, **Till VI**, **SetPoint VI**

TMove Example



17.7.18.7 Tool VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Selects the current robot tool.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Tool Number	Integer number from 0-15 representing which of 16 tool definitions to use with subsequent motion instructions.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Arm VI, **Armset VI**, **GetTool VI**, **TLSet VI**

Tool Example

17.7.18.8 TW VI

Tool Palette

Epson Robots 8.0 | Inputs & Outputs

Description

Returns the status of the wait condition.

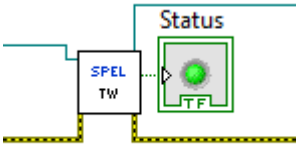
Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Status	Returns the status of the wait condition.

TW Example



17.7.19 V

17.7.19.1 VGetBool VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Retrieves a vision property or result that returns a Boolean value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.|
|Property Code |The property or result code.|
|Result Index |Optional. The index of the result.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The Boolean value.

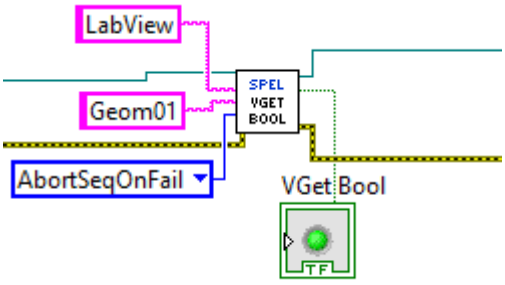
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun VI, **VGetDbl VI**, **VGetInt VI**, **VGetStr VI**, **VSetBool VI**, **VSetDbl VI**, **VSetInt VI**, **VSetStr VI**

VGetBool Example



17.7.19.2 VGetDbl VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Retrieves a vision property or result that returns a double value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.|
|Property Code |The property or result code.|
|Result Index |Optional. The index of the result.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The double value.

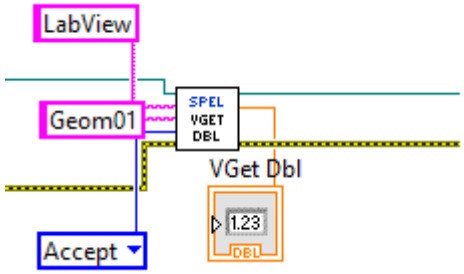
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun VI, **VGetBool VI**, **VGetInt VI**, **VGetStr VI**, **VSetBool VI**, **VSetDbl VI**, **VSetInt VI**, **VSetStr VI**

VGetDbl Example



17.7.19.3 VGetInt VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Retrieves a vision property or result that returns an integer value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.|
|Property Code |The property or result code.|
|Result Index |Optional. The index of the result.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The integer value.

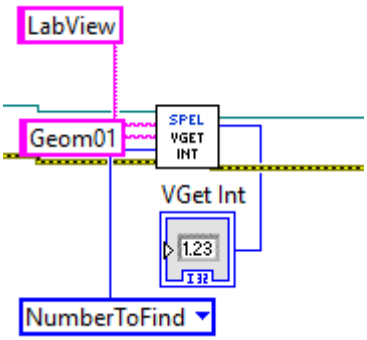
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun VI, **VGetBool VI**, **VGetDbl VI**, **VGetStr VI**, **VSetBool VI**, **VSetDbl VI**, **VSetInt VI**, **VSetStr VI**

VGetInt Example



17.7.19.4 VGetStr VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Retrieves a vision property or result that returns a string value.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when retrieving a property or result for a sequence.|
|Property Code |The property or result code.|
|Result Index |Optional. The index of the result.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Value	The string value.

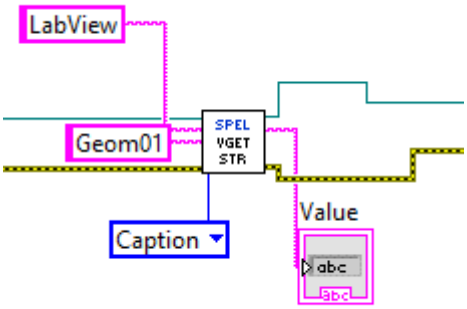
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VRun VI, **VGetBool VI**, **VGetDbl VI**, **VGetInt VI**, **VSetBool VI**, **VSetDbl VI**, **VSetInt VI**, **VSetStr VI**

VGetStr Example



17.7.19.5 VideoControl VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Controls a SPEL Video control.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
VideoRef In	Reference from a SPEL Video control.
Camera	Sets which camera video to display. Default is 0, which displays any camera.
Graphics Enabled	Sets whether graphics should be displayed.
Video Enabled	Sets whether video should be displayed.

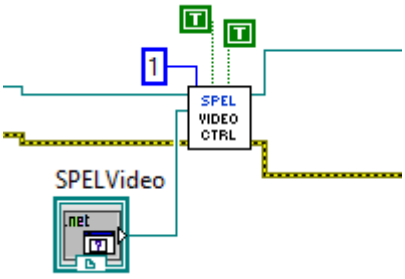
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Displaying Video, **VGet Method**, **VRun VI**

VideoControl Example



17.7.19.6 VRun VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Run a vision sequence in the current project.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

Remarks

Refer to the Vision Guide 8.0 software manual for information on running vision sequences.

See Also

VGetBool VI, **VGetDbl VI**, **VGetInt VI**, **VGetStr VI**, **VSetBool VI**, **VSetDbl VI**, **VSetInt VI**, **VSetStr VI**

VRun Example

17.7.19.7 VSetBool VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Sets the value of a vision property whose data type is Boolean.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when setting a property for a sequence.|
|Property Code |The property code.|
|Value |The new Boolean value of the property.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

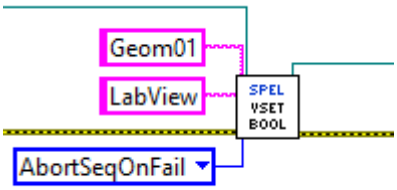
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool VI, **VGetDbl VI**, **VGetInt VI**, **VGetStr VI**, **VRun VI**, **VSetDbl VI**, **VSetInt VI**, **VSetStr VI**

VSetBool Example



17.7.19.8 VSetDbl VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Sets the value of a vision property whose data type is real or double.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when setting a property for a sequence.|
|Property Code |The property code.|
|Value |The new double value of the property.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

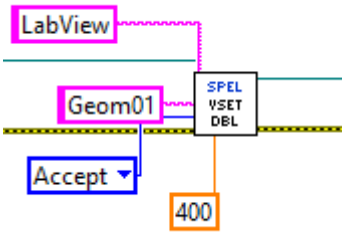
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool VI, **VGetDbl VI**, **VGetInt VI**, **VGetStr VI**, **VRun VI**, **VSetBool VI**, **VSetInt VI**, **VSetStr VI**

VSetDbl Example



17.7.19.9 VSetInt VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Sets the value of a vision property whose data type is integer.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when setting a property for a sequence.|
|Property Code |The property code.|
|Value |The new integer value of the property.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

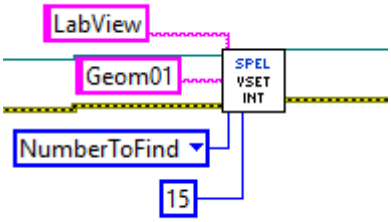
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool VI, **VGetDbl VI**, **VGetInt VI**, **VGetStr VI**, **VRun VI**, **VSetBool VI**, **VSetDbl VI**, **VSetStr VI**

VSetInt Example



17.7.19.10 VSetStr VI

Tool Palette

Epson Robots 8.0 | Vision

Description

Sets the value of a vision property whose data type is string.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Sequence	The name of a vision sequence in the current project.
Object	Optional.

The name of a vision object in the specified sequence. Omit when setting a property for a sequence.|
|Property Code |The property code.|
|Value |The new string value of the property.|

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

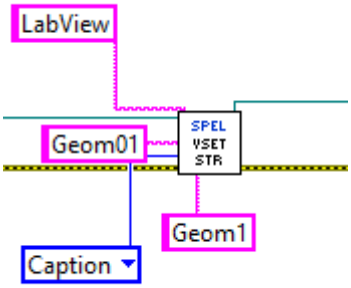
Remarks

See the Vision Guide 8.0 Properties and Results Reference manual for details on Vision Guide properties and results.

See Also

VGetBool VI, **VGetDbl VI**, **VGetInt VI**, **VGetStr VI**, **VRun VI**, **VSetBool VI**, **VSetDbl VI**, **VSetInt VI**

VSetStr Example



17.7.20 W

17.7.20.1 WaitMem VI

Tool and Control Palettes

Epson Robots 8.0 | Inputs & Outputs

Description

Waits for a memory bit status to change.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. An integer indicating the memory I/O bit. If Label is not specified, Bit Number is used.
Label	Optional. A character string indicating the memory bit label. If Label is specified, Bit Number is ignored.
Condition	A Boolean expression indicating the memory bit state.
Timeout	An integer indicating the maximum wait time (s).

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

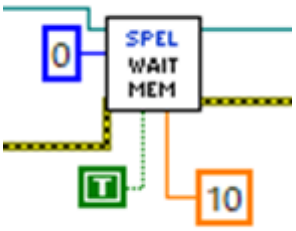
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

WaitSw VI, **SetPoint VI**

WaitMem Example



17.7.20.2 WaitSw VI

Tool and Control Palettes

Epson Robots 8.0 | Inputs & Outputs

Description

Waits for input bit status to change.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Bit Number	Optional. An integer indicating the input bit. If Label is not specified, Bit Number is used.
Label	Optional. A character string indicating the input bit label. If Label is specified, Bit Number is ignored.
Condition	A Boolean expression indicating the input bit state. Timeout An integer representing the maximum wait time in seconds.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

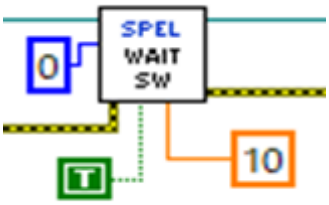
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

WaitMem VI, **SetPoint VI**

WaitSw Example



17.7.20.3 WaitTaskDone VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Waits for a task to finish and returns the status.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to be suspended. The range of the task number is 1 to 32. If Task Name is specified, then Task Number is ignored.
Task Name	Optional. Specifies the name of the task to be suspended. If Task Name is not specified, then the Task Number input will be used.

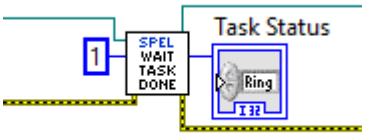
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.
Task State	Indicates the final status of the task (Quit, Aborted, Finished).

See Also

Xqt VI

WaitTaskDone Example



17.7.20.4 Weight VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Specifies the weight parameters for the current robot.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Payload Weight	The weight of the end effector to be carried in Kg units.
Arm Length	The distance from the rotational center of the second arm to the center of the gravity of the end effector in mm units.

Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

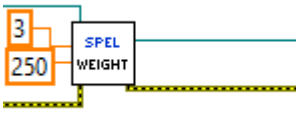
Remarks

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

See Also

Inertia VI, **SetPoint VI**

Weight Example



17.7.21 X

17.7.21.1 Xqt VI

Tool Palette

Epson Robots 8.0 | Tasks

Description

Start one SPEL+ task.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Task Number	Optional. The task number of the task to execute. The range of the task number is 1 to 32. If Task Number is omitted, then a task number will automatically be assigned.
Func Name	The name of the function to be executed.

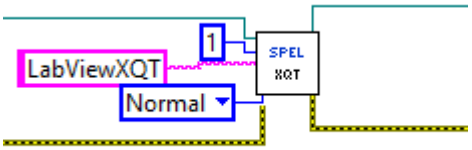
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Halt VI, **Quit VI**, **Resume VI**, **WaitTaskDone VI**

XQT Example



17.7.21.2 XYLim VI

Tool Palette

Epson Robots 8.0 | Robot Settings

Description

Sets the permissible motion range limits for the Manipulator.

Inputs

Spel Ref In	Spel reference from a previous Spel Ref Out.
Error In	Error condition from a previous Spel node.
Min X	The minimum X coordinate position to which the Manipulator may travel. (The Manipulator may not move to a position with the X Coordinate less than min X.)
Max X	The maximum X coordinate position to which the Manipulator may travel. (The Manipulator may not move to a position with the X Coordinate greater than max X.)
Min Y	The minimum Y coordinate position to which the Manipulator may travel. (The Manipulator may not move to a position with the Y Coordinate less than min Y.)
Max Y	The maximum Y coordinate position to which the Manipulator may travel. (The Manipulator may not move to a position with the Y Coordinate greater than max Y.)
Min Z	The minimum Z coordinate position to which the Manipulator may travel. (The Manipulator may not move to a position with the Z Coordinate less than min Z.)
Max Z	The maximum Z coordinate position to which the Manipulator may travel. (The Manipulator may not move to a position with the Z Coordinate greater than max Z.)

Remarks

XYLim is used to define motion range limits. Many robot systems allow users to define joint limits but the SPEL+ language allows both joint limits and motion range limits to be defined. In effect this allows users to create a work envelope for their application. (Keep in mind that joint range limits are also definable with SPEL.)

The motion range established with XYLim values applies to motion command target positions only, and not to motion paths from starting position to target position. Therefore, the arm may move outside the XYLim range during motion. (i.e. The XYLim range does not affect Pulse.)

To turn off motion range limits, specify 0 for the range limit parameters.

If an error occurs and the ErrorNumber property of the SpelException is 3101, see the comments in the SetPoint VI.

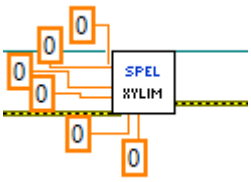
Outputs

Spel Ref Out	Spel reference output for next VI to use.
Error Out	Error condition output for subsequent Spel nodes.

See Also

Box VI, **SetPoint VI**

XYLim Example



18. Using LabVIEW with RCAPINt2.dll

18.1 Overview

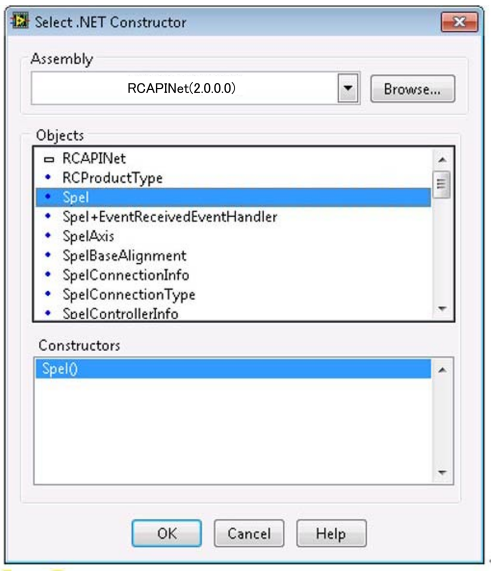
The LabVIEW VI library described in the chapter Using the LabVIEW VI Library is a high level interface that uses the RCAPINt2.dll. Some users may want to interface with RCAPINt2.dll directly instead of using the high level library. This chapter contains information for using LabVIEW with RCAPINt2.dll. The following topics are described.

- Initialization
- Use Spel properties and methods in your application
- Shutdown
- Using dialogs and windows

18.2 Initialization

18.2.1 Add a constructor node for the Spel class

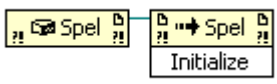
Before you can call methods or use properties from the Spel class, you must create an instance of the Spel class using a Constructor Node. You should use one Spel class instance in your application.
In the Block Diagram view of the VI that will contain the Spel class instance, add a Constructor Node from the [RC+ API] – [.NET palette]. The [Select .NET Constructor] dialog will appear.
Select “RCAPINt2(1.0.0.0)” in the [Assembly] list and select “Spel” in the [Objects] list, as shown below.



Click [OK] to create a constructor node for Spel in the block diagram.

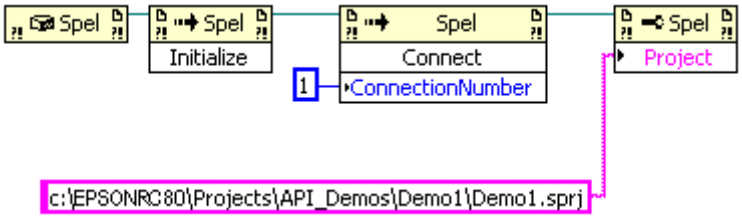
18.2.2 Initialize the Spel class instance

Add an Invoke node for the Spel class Initialize method.
Link the reference output from the Spel node to the reference input of the Initialize node. When Initialize executes, it will configure and start RC+ as a server in the background.



18.2.3 Connect to Controller and set project

Add an Invoke node for the Spel class Connect method.
Set the ConnectionNumber parameter for the Controller connection you want to use.
To view the connection numbers, start Epson RC+ 8.0, then select [Setup]-[PC to Controller Communications].
Add a Property node for the Spel class Project property.
Set the Project parameter to the desired project file.



18.3 Use Spel properties and methods

Add more nodes to use the Spel properties and methods for your application.
You must wire the reference output from the previous node to the reference input of the current node. This allows each property or method to use the Spel class instance you created and initialized in the steps above.
For details about the available properties and methods, see:
RCAPINet Reference

18.4 Shutdown

When you are finished using the Spel class instance, you need to invoke the Dispose method. This will shutdown the Epson RC+ 8.0 server that is associated with the Spel class instance. Normally, you should call Dispose at the end of your application.
When using Epson RC+ as an out-process server:
If your application is aborted without calling Dispose, then the RC+ process continues to run, because LabVIEW (the client process) continues to run. If you start your application again, the RC+ process is restarted if it was running. But if you try to run the RC+ GUI, it will ask if you want to run another instance of RC+. In this case, you can terminate the RC+ process (erc80.exe) from the Windows Task Manager first, then run the Epson RC+ 8.0 GUI.

18.5 Using dialogs and windows

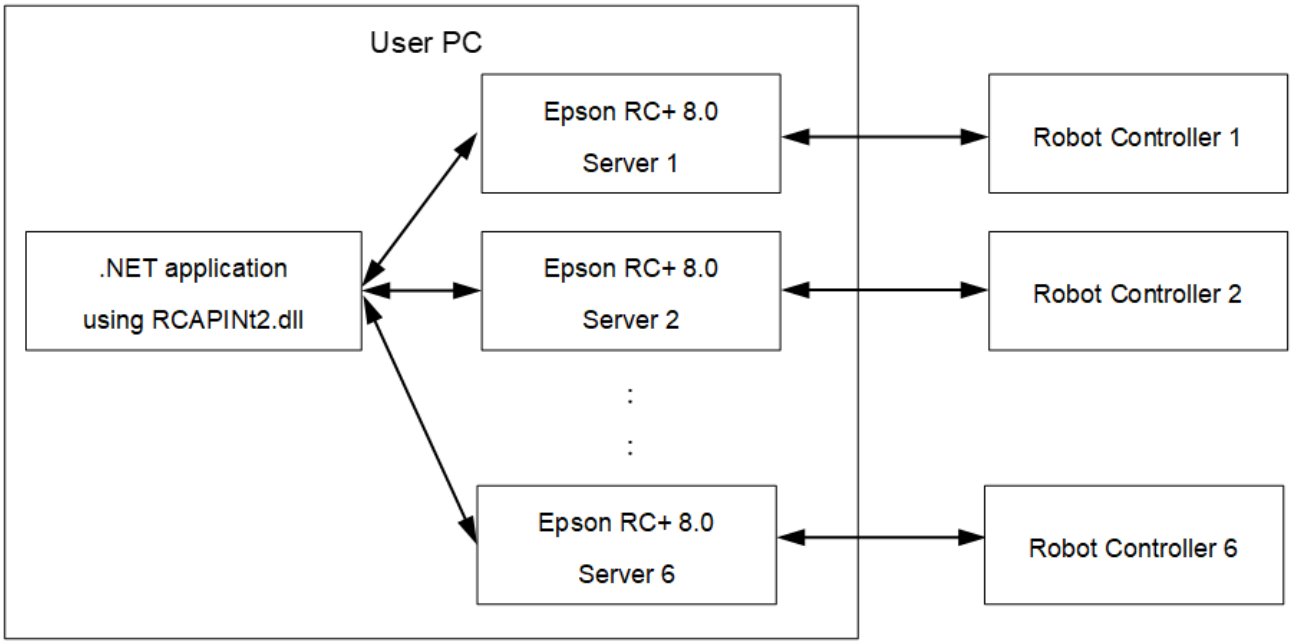
When used with .NET applications, a .NET parent form is normally used as the parent for dialogs and windows that are displayed from the Spel class instance.
But LabVIEW does not use .NET forms, so to display windows and dialogs from LabVIEW, use the ParentWindowHandle property.
Set it to the window handle of your VI. You can call the Windows API FindWindow method to get the window handle.
When using ParentWindowHandle, you must call Spel.ShowWindow without the Parent parameter.

19. How to Control Multiple Controllers from One PC

19.1 Overview

Using the RC+ API, you can control up to six Robot Controllers from one PC.
To control multiple Controllers, an RCAPINet Spel class needs to be instantiated for each Controller.

The figure below shows the basic system configuration diagram for controlling multiple Controllers using the RC+ API.



The application controls the multiple Controllers via the servers (RCAPINet Spel class) prepared for each Controller.

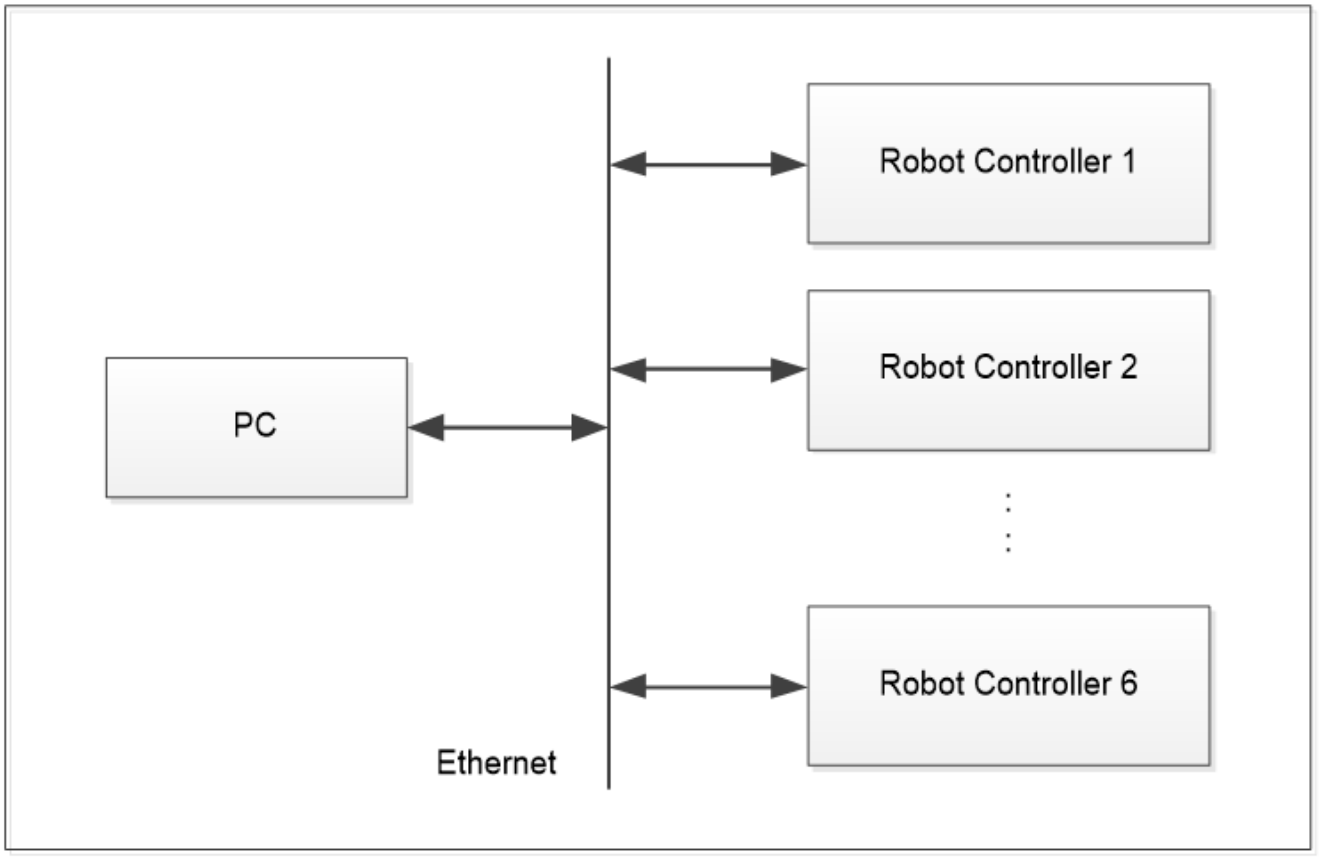
19.1.1 System Requirements

Refer to the following manual for recommended PC performance.
Epson RC+ User's Guide System Requirements

> ⚠ CAUTION
>
> When using a PC that does not satisfy the above requirements, make sure you check operation adequately before using the system in production. If a PC that does not satisfy the above requirements is used, the Controllers may not be controlled reliably.

19.1.2 Connection of PC and Controllers

The connection type for the first Controller can be USB or Ethernet. The connection type for the remaining Controllers must be Ethernet.
The figure below shows the basic configuration diagram of the PC and the multiple Controllers.



Robot Controller supports the Controllers that connect to Epson RC+ 8.0.

KEY POINTS

Controllers that connect to Epson RC+ 8.0 can be connected at the same time.

One virtual Controller can be selected.

One Controller can be connected by the USB communication.

When using the USB communication, connect only one Controller by the USB communication and connect other Controllers by the Ethernet.

CAUTION

If the anti-virus software is installed on the PC, communication with the Controllers may be disconnected abnormally when running a virus scan. To run a virus scan, disconnected the communication with the Controllers beforehand.

19.2 Restrictions on controlling multiple Controllers

Controlling multiple Controllers has restrictions as described in the following sections.

19.2.1 Restrictions on Controller options

Refer to the following manuals for the limitations of each controller option.
“User’s Guide”

19.2.2 Restrictions on simulator

Simulator window display
Epson RC+ 8.0 simulator window can be used from the .NET application. See below for details.
Windows

If the simulator window is opened for each Controller when multiple Controllers are connected, the cycle time may increase by 100 to 200 msec compared to not displaying the simulator windows.
Also, if the program is executed with the simulator windows open, the CPU utilization increases near 100 % and a huge load may be put on the PC.

It is recommended to use the system with the simulator windows closed, except when debugging the program.

Collision detection
To avoid collision with peripherals using the simulator, set 15 mm or more margins around the simulator object to avoid collision detection.
Collision detection in the simulator does not guarantee the accuracy. When applying to the actual system, make sure to set the margins and check operation adequately.
For details on each restriction, refer to Simulator Specifications and Restrictions in Epson RC+ Simulator.

19.3 Sample Program for connecting multiple Controllers

The following sections describe sample programs to connect the PC with Controller 1 and Controller 2 using a .NET application.

KEY POINTS

For details about the available properties and methods, see:

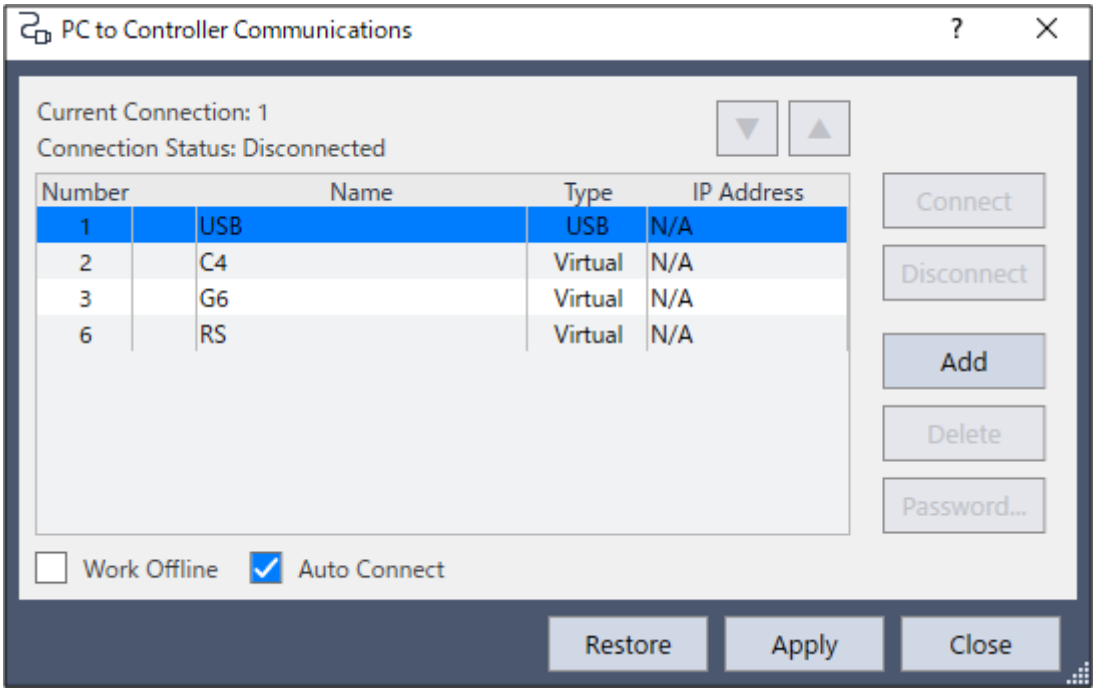
RCAPINet Reference

19.3.1 Controller connection setting

When connecting the multiple Controllers at the same time, specify the connection using the Connect method of Spel class.

```
m_spel.Connect(1)
```

The parameter in the Connect method indicates the connection number. This number is same as the one shown in “Number” in the dialog box below (Epson RC+ 8.0 menu-[Setup]-[PC to Controller Communications]). If a value of -1 is used, it means to use the most recent connection.



19.3.2 Project setting

To connect the multiple controllers, specify the project using the Project property of the Spel class.

```
m_spel.Project = "c:\EpsonRC80\projects\API_Demos\Demo1\Demo1.sprj"
```

19.3.3 Sample program using Visual Basic

1. Select menu-[File]-[New]-[Project] in Visual Studio .NET.
2. Create a Visual Basic project.
3. Select [Project]-[Add Reference].
4. Select the [Browse] tab, refer to the “(Program folder)\API\Assembly\net462” directory, and then select the “RCAPINt2.dll” file.
5. Add two buttons (btnController1, btnController2) to the Form1 class.
6. Add quick events of each button and create the thread to control each Robot Controller.

```
Private trd1 As System.Threading.Thread    ' for Robot Controller 1
Private trd2 As System.Threading.Thread    ' for Robot Controller 2

Private Sub btnController1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnController1.Click
      ' Start thread for Robot Controller 1
      trd1 = New System.Threading.Thread( _
New System.Threading.ThreadStart(AddressOf StartController1))
      trd1.Start()
End Sub
Private Sub StartController1()
    ' Control Robot Controller 1
    Try
       Dim frm1 As New frmDemo1
       frm1.ShowDialog()
       frm1.Dispose()
    Catch ex As Exception
```

```
       MsgBox(ex.Message)
    End Try
End Sub
Private Sub btnController2_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnController2.Click
     ' Start thread for Robot Controller 2
     trd2 = New System.Threading.Thread( _
New System.Threading.ThreadStart(AddressOf StartController2))
     trd2.Start()
End Sub
Private Sub StartController2()
    ' Control Robot Controller 2
    Try
       Dim frm2 As New frmDemo2
       frm2.ShowDialog()
       frm2.Dispose()
    Catch ex As Exception
       MsgBox(ex.Message)
    End Try
End Sub
```

7. Add a form (frmDemo1) for Controller 1.

```
Private WithEvents m_spel1 As New Spel
Private Sub frmDemo1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load

     Try
        m_spel1.ServerInstance=1
        m_spel1.Initialize()
        m_spel1.Connect(5)
        m_spel1.Project = " c:\\EpsonRC80\\Projects\\Demo1\\Demo1.sprj "
     Catch ex As SpelException
        MsgBox(ex.Message)
     End Try
End Sub
Private Sub m_spel1_EventReceived(ByVal sender As Object, ByVal e As
 SpelEventArgs) _Handles m_spel1.EventReceived
    ' for Robot Controller 1
End Sub
Private Sub frmDemo1_FormClosed(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.FormClosedEventArgs) _
Handles MyBase.FormClosed
    m_spel1.Dispose()
End Sub
```

8. Add a form (frmDemo2) for Controller 2.

```
Private WithEvents m_spel2 As New Spel
Private Sub frmDemo2_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load

    Try
       m_spel2.ServerInstance=2
       m_spel2.Initialize()
       m_spel2.Connect(6)
       m_spel2.Project = " c:\\EpsonRC80\\Projects\\Demo2\\Demo2.sprj "
    Catch ex As SpelException
       MsgBox(ex.Message)
    End Try
End Sub

Private Sub m_spel2_EventReceived(ByVal sender As Object, ByVal e As
SpelEventArgs) _Handles m_spel2.EventReceived
    ' for Robot Controller 2
```

```
End Sub
Private Sub frmDemo2_FormClosed(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.FormClosedEventArgs) _
Handles MyBase.FormClosed
    m_spel2.Dispose()
End Sub
```

19.3.4 Sample program using Visual C#

1. Select menu-[File]-[New]-[Project] in Visual Studio .NET.
2. Create a Visual C# project.
3. Select [Project]-[Add Reference].
4. Select the [Browse] tab, refer to the “(Program folder)\API\Assembly\net462” directory, and then select the “RCAPINt2.dll” file.
5. Add two buttons (btnController1, btnController2) to the Form1 class.
6. Add quick events of each button and create the thread to control each Robot Controller.

```
private System.Threading.Thread trd1;  // for robot controller1
private System.Threading.Thread trd2;  // for robot controller2

private void btnController1_Click(object sender, EventArgs e)
{
    // Start thread for robot controller 1
    trd1 = new System.Threading.Thread(new _
                        System.Threading.ThreadStart(StartController1));
    trd1.Start();
}
private void StartController1()
{
    // Control robot controller 1
    try
    {
       frmDemo1 frm1 = new frmDemo1();
       frm1.ShowDialog();
       frm1.Dispose();
    }
    catch (System.Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private void btnController2_Click(object sender, EventArgs e)
{
    // Start thread for robot controller 2
    trd2 = new System.Threading.Thread(new _
System.Threading.ThreadStart(StartController2));
    trd2.Start();
}
private void StartController2()
{
    // Control robot controller 2
    try
    {
       frmDemo2 frm2 = new frmDemo2();
       frm2.ShowDialog();
       frm2.Dispose();
    }
    catch (System.Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

7. Add a form (frmDemo1) for Controller 1.

```
private Spel m_spel1;
private void frmDemo1_Load(object sender, EventArgs e)
{
    m_spel1 = new Spel();
    try
    {
       m_spel1.ServerInstance = 1;
       m_spel1.Initialize();
       m_spel1.Connect(5);
       m_spel1.Project = "c:\\EpsonRC80\\Projects\\Demo1\\Demo1.sprj";
       m_spel1.EventReceived += new _
Spel.EventReceivedEventHandler(m_spel1_EventReceived);
    }
    catch (SpelException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
public void m_spel1_EventReceived(object sender, SpelEventArgs e)
{
    // for robot controller 1
}
private void frmDemo1_FormClosed(object sender, FormClosedEventArgs e)
{
    m_spel1.Dispose();
}
```

8. Add a form (frmDemo2) for Controller 2.

```
private Spel m_spel2;
private void frmDemo2_Load(object sender, EventArgs e)
{
    m_spel2 = new Spel();
    try
    {
       m_spel2.ServerInstance = 2;
       m_spel2.Initialize();
       m_spel2.Connect(6);
       m_spel2.Project = "c:\\EpsonRC80\\Projects\\Demo2\\Demo2.sprj";
       m_spel2.EventReceived += new _
Spel.EventReceivedEventHandler(m_spel2_EventReceived);
    }
    catch (SpelException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
public void m_spel2_EventReceived(object sender, SpelEventArgs e)
{
    // for robot controller 2
}
private void frmDemo2_FormClosed(object sender, FormClosedEventArgs e)
{
    m_spel2.Dispose();
}
```