

Rexroth VisualMotion 9 Multi-Axis Motion Control using GPP and GMP Firmware

R911292840
Edition 01

Functional Description



Title	VisualMotion 9 Multi-Axis Motion Control using GPP and GMP Firmware
Type of Documentation	Functional Description
Document Typecode	DOK-VISMOT-VM*-09VRS**-FK01-EN-P
Internal File Reference	Document Number, 120-2300-B316-01/EN Part of Box Set, 20-09V-EN (Material No.:293201)
Purpose of Documentation	This document is a functional description for the PPC-R multiple axes motion control using GPP09 and GMP09 firmwares. The information provided in this document is intended to support the VisualMotion 9 system.

Record of Revisions	Description	Release Date	Notes
	DOK-VISMOT-VM*-09VRS**-FK01-EN-P	07/2003	Initial Release



Copyright © 2003 Bosch Rexroth AG
Copying this document, giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34-1).


Validity The specified data is for product description purposes only and may not be deemed to be guaranteed unless expressly confirmed in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2 • D-97816 Lohr a. Main
Tel.: +49 (0)93 52/40-0 • Fax: +49 (0)93 52/40-48 85 • Telex: 68 94 21
Bosch Rexroth Corporation • Electric Drives and Controls
5150 Prairie Stone Parkway • Hoffman Estates, IL 60192 • USA
Tel.: 847-645-3600 • Fax: 847-645-6201
<http://www.boschrexroth.com/>
Dept. ESG4 (DPJ)

Note This document has been printed on chlorine-free bleached paper.

Table of Contents

1	VisualMotion 9 Overview	1-1
1.1	System Overview	1-1
1.2	GPP 9 System Overview.....	1-1
	GPP 9 System Components.....	1-2
	GPP 9 PLC Support.....	1-2
	GPP 9 Interface Support.....	1-2
	Drive I/O Support	1-3
1.3	GMP 9 System Overview.....	1-3
	GMP 9 Firmware Features.....	1-4
	GMP 9 System Components	1-4
	GMP 9 Interface Support	1-4
	Drive I/O Support	1-4
2	VisualMotion Toolkit Menu Commands	2-1
2.1	Overview	2-1
	Programming Modes.....	2-1
	Toolbar Buttons.....	2-2
	Project Navigator	2-3
2.2	The <u>F</u> ile Menu	2-6
	File ⇒ New.....	2-7
	Open... 	2-10
	Close.....	2-10
	Save Program	2-10
	Save As...	2-10
	Save All Project Data...	2-10
	Save Project As...	2-10
	Online / Offline ... F8	2-11
	Import Project Component...	2-12
	Export Project Component.....	2-14
	Remove Project Components.....	2-15
	Synchronize Project Components	2-16
	Print Project Data... 	2-17
	Sample Programs	2-18
	Recent Programs	2-18
	Recent Projects.....	2-18
	Exit	2-18
2.3	The <u>E</u> dit Menu.....	2-19
	Windows Editing Features	2-19
	Clear Icon Flow	2-20

	Browse	2-20
	Find, Find Next, Replace	2-24
	VM Data	2-26
	Labels.....	2-38
2.4	The <u>V</u> iew Menu	2-43
	Task.....	2-44
	Subroutines	2-44
	Event Functions	2-46
	Zoom Out F6.....	2-47
	Icon Palette	2-48
	Icon Comments.....	2-49
	Icon Captions	2-49
	Function Comment.....	2-49
2.5	Insert Menu	2-49
	Subroutine.....	2-49
	Event Function	2-50
2.6	The Build Menu	2-51
	Save, Compile, Download 	2-51
	Compile Program	2-51
	Program Management	2-52
2.7	The Commission Menu	2-55
	Drive Overview.....	2-55
	I/O Setup	2-57
	I/O Mapper	2-58
	Fieldbus Mapper	2-58
	PLS (Programmable Limit Switch).....	2-59
	Position Monitoring Group	2-60
	Coordinated Motion.....	2-66
	Archive	2-70
	Transfer.....	2-75
2.8	The Data Menu	2-87
	Parameters	2-87
	Registers	2-102
	Variables	2-107
	Events	2-108
	Points	2-110
	CAM Indexer	2-112
	ELS	2-113
	PID	2-116
	Registration	2-123
	Zones	2-125
2.9	The <u>D</u> iagnostics Menu	2-126
	System	2-127
	Tasks.....	2-132
	Oscilloscope.....	2-133
	Show Program Flow F7	2-143

Toggle Breakpoint.....	2-144
2.10 The Tools Menu	2-146
CAM Builder	2-146
Jogging.....	2-153
Registered Tools.....	2-157
Control Selection.....	2-157
Control Settings	2-159
Options.....	2-167
2.11 The Window Menu	2-168
2.12 The Help Menu.....	2-169
Getting Started.....	2-169
Search.....	2-169
Registered Help	2-170
About VisualMotion	2-170
3 Icon Programming	3-1
3.1 Overview	3-1
Working with VisualMotion Toolkit's Icon Palettes.....	3-1
3.2 VisualMotion Toolkit Icons	3-7
Start1 and Finish1 Icons	3-7
Initialization Icon Palette	3-8
Single Axis Icon Palette	3-9
Coordinated Motion Icon Palette	3-10
ELS Icon Palette	3-11
Utility Icon Palette	3-12
Accel	3-13
Axis2	3-14
Branch.....	3-20
Calc2.....	3-22
CAM	3-31
CAMAdj.....	3-32
CamBld2	3-33
CAM Indexer	3-37
Circle	3-47
Command	3-50
CoordArt (Coordinated Articulation).....	3-51
ELSAAdj1	3-61
ELSGrp1	3-62
ELSMstr1	3-73
ELSMode	3-79
Event2	3-80
Finish1.....	3-87
Go1	3-88
Home.....	3-88
I_O (I/O Setup).....	3-89
Join.....	3-90

Joint.....	3-90
Line	3-91
Move2	3-91
Msg1	3-93
Param.....	3-94
ParamBit	3-96
Path.....	3-98
PID1	3-100
Position	3-104
PrmBit	3-104
PrmInt.....	3-105
Ratio.....	3-107
Reg (Register Transfer)	3-108
Scissors.....	3-108
Start1.....	3-109
Stop1	3-113
Sub1	3-114
Veloc (Velocity)	3-117
VM (Virtual Master)	3-117
Wait1	3-122

4 Registers 4-1

4.1 Register 001: System Control	4-2
Bit 1: Parameter Mode/ nRun Mode	4-2
Bit 3: nEmergency Stop	4-2
Bit 5: Clear All Errors	4-2
Bit 6: Pendant Live Man.....	4-2
Bit 7: Rebuild Double Ring.....	4-3
Bit 8-12: Activate Program and Binary Program Select.....	4-3
Bit 14: Pendant Enable	4-3
Bits 15-16: Pendant Access Level	4-3
4.2 Registers 002-005: Task Control	4-4
Bit 1: Mode: Automatic/ nManual.....	4-4
Bit 2: Override Automatic Start	4-4
Bit 4: Single Step Select	4-5
Bit 6: Cycle Start/Resume.....	4-5
Bit 7: nTask Stop.....	4-5
Bit 9: Task Event Trigger	4-6
Bit 11: Breakpoint Enable	4-6
Bit 12: Sequencer Single Step	4-6
Bit 13: Step Sequence Function	4-6
Bit 15: Coordinate Fast Stop.....	4-6
4.3 Cycle Control Considerations.....	4-7
Cycle Stop in User Program	4-7
System Parameter Mode	4-7
System Shutdown Errors	4-7

Programmed End of Task	4-7
4.4 Register 006: System Diagnostic Code	4-7
4.5 Registers 007-010: Task Jog Control	4-8
Single-Axis and Velocity Mode Jogging.....	4-8
Coordinated Jogging	4-8
Bit 1: Mode: Continuous/nStep	4-9
Bit 2 and 3: Coordinated Jog Forward and Reverse	4-9
Bits 4 and 5: Jog Type	4-9
Bit 6: Distance/Speed (Large/nSmall, Fast/nSlow).....	4-9
Bits 9-14: Jog Coordinate and Joint.....	4-9
4.6 Registers 011-018, 209-240: Axis Control.....	4-10
Bit 1: Disable Axis	4-10
Bit 2 and 3: Jog Forward (bit 2) and Reverse (bit 3).....	4-11
Bit 4: Synchronized Jog	4-11
4.7 Register 019: Fieldbus/PLC Status.....	4-11
Bit 1 and 2: Fieldbus Initialization OK; LSB and MSB	4-11
Bit 4: Fieldbus Slave Ready, LSB.....	4-12
Bit 5: Non Cyclic Ready	4-12
Bit 14: Register Data Valid.....	4-12
Bit 15: Cyclic Data Valid.....	4-12
4.8 Register 020: Fieldbus/PLC Diagnostics	4-13
Bits 13- 16: Fieldbus/PLC Type	4-13
4.9 Register 021: System Status.....	4-14
Bit 1: Parameter Mode/ Initializing.....	4-14
Bit 4: Service Channel Ready.....	4-14
Bit 5: Error	4-14
Bit 6: Error Active	4-14
Bit 7: Warning Active.....	4-14
Bits 9-12: Active Program	4-15
Bit 13: Teach Pendant Password Active.....	4-15
Bit 14: Teach Pendant Connected.....	4-15
4.10 Registers 022-025: Task Status.....	4-15
Bit 1: Mode: Automatic/Manual.....	4-15
Bit 2: Coordinated Running.....	4-15
Bit 4: Single Stepping.....	4-16
Bit 5: Task Error	4-16
Bit 6: Task Running.....	4-16
Bit 8: Coordinate In Position	4-16
Bit 11: Breakpoint Reached	4-16
Bit 15: Coordinated Fast Stop.....	4-16
4.11 Register 027: Initialization Task Control	4-16
4.12 Register 028: Initialization Task Status.....	4-16
4.13 Registers 031-038, 309-340: Axis Status	4-17
Bit 1: Multiplex Channel Enabled.....	4-17
Bits 2 and 3: Jogging Forward (Bit 2) and Reverse (Bit 3)	4-17
Bit 4: Phase Adjusted.....	4-17




Bit 5: ELS Enabled.....	4-17
Bit 6: ELS Secondary Mode.....	4-18
Bit 7: Single Axis in Position	4-18
Bit 8: Axis Aligned (Control cam axes only).....	4-18
Bit 10: Axis Stopped / Axis Present on Ring.....	4-18
Bit 11: Axis Halted.....	4-18
Bit 12: Class 3 Status.....	4-18
Bit 13: Class 2 Warning	4-18
Bit 14: Drive Shutdown Error	4-19
Bit 15, 16: Ready to Operate	4-19
4.14 Registers 040: Link Ring Status.....	4-19
Bit 4: Link Error	4-19
Bit 5: Error in Primary Optic Ring.....	4-19
Bit 6: Error in Secondary Optic Ring.....	4-20
Bit 7: Redundancy Loss	4-20
4.15 Registers 041: Link Ring Data 1	4-20
Bit 1-16: Node 1-16 Data Valid	4-20
4.16 Registers 042: Link Ring Data 2	4-21
Bit 17-32: Node 17-32 Data Valid	4-21
4.17 Registers 050: Ethernet Status	4-21
Bit 1: Card Present.....	4-21
Bit 9: Request Received	4-21
Bit 10: Response Pending	4-22
Bit 11: Response Done	4-22
Bit 12: Response Sent	4-22
Bit 16: Invalid Protocol	4-22
4.18 Register 051: Standard Message Count.....	4-22
4.19 Register 052: Cyphered Message Count.....	4-22
4.20 Register 053: Invalid Protocol Count	4-22
4.21 Register 054: SIS Message Count.....	4-23
4.22 Registers 086: PMG Control.....	4-23
Bit 1-8: PMG#_ENABLE	4-23
Bit 9-16: PMG#_CALC_OFFSET	4-23
4.23 Registers 087: PMG Status.....	4-24
Bit 1-8: PMG#_ENABLED.....	4-24
Bit 9-16: PMG#_ERROR	4-24
4.24 Registers 088 and 089: Task A Extend Event Control	4-24
Register Operation.....	4-24
4.25 Registers 090 and 091: Latch and Unlatch.....	4-25
4.26 Registers 092-094: Mask Pendant Key Functionality	4-25
4.27 Registers 095-097: BTC06 Teach Pendant Status.....	4-25
4.28 Registers 098: Pendant Control - Task A, B	4-26
4.29 Registers 099: Pendant Control - Task C-D	4-27
4.30 Register 140 ELS Master Control	4-27
Bits 7- 12: Set ELS Master (1-6) Reference Position	4-28
Bit 15: Capture Slip Monitoring	4-28

Bit 16: Enable Slip Monitoring.....	4-28
4.31 Register 141 ELS Master Status.....	4-29
Bit 1-6: Master at Standstill.....	4-29
Bits 7- 8: ELS Master (1-6) Position Referenced.....	4-29
Bit 14: Monitoring ERROR Active	4-29
Bit 15: Lead Encoder	4-30
Bit 16: Enable Slip Monitoring.....	4-30
4.32 Registers 150 and 151: Virtual Master 1 & 2 Control	4-30
Bit 1: Virtual Master 1or 2 Fast Stop.....	4-30
Bit 2: Virtual Master Home Position	4-30
Bit 3: Virtual Master Go Command	4-31
Bit 4: Virtual Master Mode of Operation.....	4-31
Bit 5: Virtual Master Absolute/Relative Mode	4-32
Bit 6: Virtual Master Relative Trigger Mode.....	4-32
4.33 Registers 152-159: ELS Groups 1-8 Control	4-33
Bit 1: Group Lock Off/ Lock on Cams	4-33
Bit 2: Group Master Relative Phase Adjust	4-33
Bit 3: Group Slave Relative Phase Adjust	4-33
Bit 4: Group Master Input Select.....	4-34
Bit 5: Group Forcing.....	4-34
Bit 6: Group Local Mode Select	4-34
Bit 7: Group Jog Continuous/Incremental.....	4-34
Bit 8: Group Jog Absolute	4-35
Bit 9: Group Jog in Positive Direction	4-35
Bit 10: Group Jog in Negative Direction.....	4-35
Bit 11: Group Master Absolute Phase Adjust	4-35
Bit 12: Group Slave Absolute Phase Adjust	4-35
4.34 Register 197: Coordinated Articulation Synchronized Mode Control.....	4-35
Bits 1-6: Synchronized Mode.....	4-36
Bit 7-12: Positioning Mode	4-36
4.35 Register 198: Coordinated Articulation Local Mode Control.....	4-37
Bits 1-6: Normal or Immediate Local Mode	4-37
Bits 7-12: Immediate Mode.....	4-38
4.36 Registers 241 and 242: Virtual Master 1 & 2 Status	4-38
Bit 1: Virtual Master 1or 2 Fast Stop Active	4-38
Bit 2: Virtual Master Home Position	4-38
Bit 4: Virtual Master Mode of Operation Active.....	4-38
Bit 5: Virtual Master Absolute/Relative Mode Active	4-39
Bit 7: Virtual Master at Zero Velocity.....	4-39
Bit 8: Virtual Master in Position.....	4-39
4.37 Registers 243 - 250: ELS Groups 1-8 Status.....	4-39
Bit 1: Group Lock Off/ Lock On Cams Status	4-39
Bit 2: Group Master Phase Adjust Status	4-40
Bit 3: Group Slave Phase Adjust Status	4-40
Bit 4: Group Master Input Select Status	4-40
Bit 5: Group Forcing Status.....	4-40

Bit 6: Group Local Mode Select	4-40
Bit 8: Group Jog Position Status	4-40
Bit 9: Group Motion Status	4-41
Bit 11: Group Master Absolute Phase Adjust Status	4-41
Bit 12: Group Slave Absolute Phase Adjust Status	4-41
4.38 Register 288: Coordinated Articulation Synchronized Mode Status	4-41
Bits 1-6: Synchronized Mode	4-41
Bit 7-12: Positioning Mode	4-42
4.39 Register 289: Coordinated Articulation Local Mode Status	4-42
Bits 1-6: Local Mode At Target Position	4-42

5 Parameters	5-1
5.1 Overview	5-1
5.2 Parameter Identification	5-2
Parameter Attributes	5-2
Class "C" Parameters	5-2
Class "T" Parameters	5-2
Class "A" Parameters	5-3
Class "S" and "P" Parameters	5-3
5.3 Parameter Transfer	5-3
5.4 SERCOS Drive Telegram Utility	5-4
AT (Drive Telegram)	5-4
MDT (Master Data Telegram)	5-5
Displaying the Contents of the AT and MDT	5-6
Configuring the AT and MDT	5-7
Multiplex (MUX) Channel (DKC 2.3 only)	5-9
Telegram Options	5-9
5.5 Parameter Types	5-11
Control Parameters - Class C	5-11
Task Parameters - Class T	5-20
Axis Parameters - Class A	5-22
5.6 Control Parameters – Class C	5-25
System Setup (C-0-0001 through C-0-0035)	5-25
Jogging and Display Parameters (C-0-0042 through C-0-0056)	5-37
Program Management (C-0-0090 through C-0-0099)	5-43
System Status (C-0-0100 through C-0-0127)	5-46
Control Processor Usage Status (C-0-0200 through C-0-0203)	5-55
Link Ring Parameters (C-0-0300 through C-0-0303)	5-56
Ethernet Parameters (C-0-0400 through C-0-0408)	5-59
Initialization Task Parameters (C-0-0522 through C-0-0537)	5-65
BTC06 Teach Pendant (C-0-0801 through C-0-0814)	5-68
Internal System Monitoring (C-0-0990)	5-77
System Memory Parameters (C-0-0994 and C-0-0996)	5-79
System Parameter Lists (C-0-2000 through C-0-2021)	5-80
Oscilloscope Parameters (C-0-2501 through C-0-2523)	5-86
Fieldbus/PLC Interface Parameters (C-0-2600 through C-0-2653)	5-96

Option Card PLS Interface Parameters (C-0-2901 through C-0-2943)	5-110
I/O Mapper Parameters (C-0-3000 through C-0-3005)	5-120
CAM Table Parameters (C-0-3100 through C-0-3140)	5-123
Position Monitoring Group Parameters	5-125
5.7 Task Parameters - Class T	5-130
Task Setup (T-0-0001 and T-0-0002)	5-130
Coordinated Motion (T-0-0005 through T-0-0026)	5-132
Coordinated Motion Status (T-0-0100 through T-0-0113)	5-138
Task Status (T-0-0120 through T-0-0200)	5-141
Task Parameter Lists (T-0-2000 and T-0-2001)	5-146
5.8 Axis Parameters – Class A	5-147
Axis Setup (A-0-0001 through A-0-0038)	5-147
Axis Status (A-0-0100 through A-0-0145)	5-164
Electronic Line Shafting (A-0-0150 through A-0-0164)	5-170
Axis Feedback Capture (Registration) (A-0-0170 through A-0-0174)	5-177
Optional SERCOS Data (A-0-0180 through A-0-0196)	5-178
Multiplexing Parameters (A-0-0200 through A-0-0203) (DKC 2.3 only)	5-182
Axis Parameter Lists (A-0-2000 and A-0-2001)	5-184
6 VisualMotion's I/O System	6-1
6.1 Overview	6-1
6.2 I/O Configuration Tool	6-1
RECO02 Error Detection	6-2
6.3 Menu Selection	6-3
The File Menu	6-3
The Edit Menu	6-3
The View Menu	6-7
The Tools Menu	6-8
The Help Menu	6-9
I/O Configuration in Project Mode	6-10
I/O Configuration in Service Mode	6-12
7 I/O Mapper	7-1
7.1 Overview	7-1
7.2 Specifications	7-2
Ladder Window	7-2
Boolean Equations	7-3
Memory Usage	7-4
7.3 Menu Selection	7-4
The File Menu	7-4
The Edit Menu	7-5
The View Menu	7-5
The Tools Menu	7-6
The Window Menu	7-7
The Help Menu	7-7
7.4 Additional Menu Selections	7-8

Delete row	7-8
Undo.....	7-8
Insert row	7-8
Cross Reference	7-8
Check rungs and convert to Boolean strings	7-9
7.5 Toolbar Buttons.....	7-9
Ladder Logic Toolbar Buttons.....	7-9
Forcing Icon Toolbar Buttons.....	7-10
7.6 Input Logic Functions	7-13
Register and Bit	7-13
Contact Setup	7-13
Contact Selection for Functions.....	7-14
7.7 Output Logic Functions	7-14
Coil Relay.....	7-14
Binary Shift Register (BSR) 	7-17
Timer (TON) 	7-18
Counter (UDC) 	7-19
7.8 I/O Mapper in Project Mode	7-20
Switching to Online Mode	7-20
Importing an I/O Mapper into a Project.....	7-20
7.9 I/O Mapper in Service Mode	7-21
7.10 Import Default I/O Mapper.....	7-21
Default I/O Mapper in Project Mode (Offline).....	7-22
Default I/O Mapper in Service Mode.....	7-22

8 Programmable Limit Switch Functionality 8-1

8.1 PLS Description	8-1
PLS Object.....	8-1
Control PLS.....	8-2
Drive PLS.....	8-3
Option Card PLS.....	8-5
VisualMotion PLS Tool.....	8-7
PLS Tool Communication Modes	8-7
8.2 Configure a Control PLS	8-9
Switch Configuration for a Control PLS	8-9
PLS Master Configuration for a Control PLS.....	8-10
PLS Register Assignment for a Control PLS	8-12
8.3 Configure a Drive PLS	8-14
Switch Configuration for a Drive PLS	8-15
PLS Master Configuration for a Drive PLS	8-16
PLS Register Assignment for a Drive PLS	8-16
8.4 Configure an Option Card PLS	8-18
Switch Configuration for an Option Card PLS	8-19
PLS Master(s) Configuration for an Option Card PLS.....	8-20
PLS Register Assignment for an Option Card PLS	8-22

PLS Outputs.....	8-24
Output Configuration.....	8-25
8.5 Editing PLS Configurations.....	8-31
Edit PLS Configuration in Project Mode.....	8-31
Edit PLS Configuration in Service Mode.....	8-34
Editing with a Right Mouse Click.....	8-35
8.6 Saving and Downloading PLS Configurations.....	8-36
Saving a PLS Configuration.....	8-36
Downloading a PLS to the Control.....	8-37
8.7 Uploading PLS Configurations.....	8-38
Upload in Project Mode.....	8-38
Upload in Service Mode.....	8-38
8.8 Monitoring a PLS Status.....	8-38
Monitor in Project Mode.....	8-39
Monitor in Service Mode.....	8-39
PLS Message.....	8-40
8.9 Access PLS Data via the Calc Icon.....	8-40
Control PLS.....	8-41
Drive PLS.....	8-41
Option Card PLS.....	8-42
9 PPC-P11.1 Control Functionality.....	9-1
9.1 Overview.....	9-1
Fieldbus and I/O Support.....	9-1
Bosch Rexroth Interfaces.....	9-2
PCI Bus Memory.....	9-2
9.2 DPR Interface.....	9-3
DPR Parameters.....	9-4
Shared DPR Memory Map.....	9-6
9.3 Register Channel.....	9-7
Register Channel Transmission.....	9-7
Register Channel Configuration.....	9-7
9.4 Cyclic Channel.....	9-8
Accessing Drive Parameters.....	9-8
Cyclic Channel Transmission.....	9-8
Cyclic Channel Configuration.....	9-8
Handshaking.....	9-9
9.5 Non-Cyclic Channel.....	9-11
Protocol Identification.....	9-11
Non-Cyclic Channel Transmission.....	9-11
Operating Procedure of the Non-Cyclic Channel.....	9-12
9.6 PPC-P11.1 Diagnostic Information.....	9-13
Error_Code.....	9-13
Diagnostic_Text.....	9-13
MC_Mode.....	9-13
9.7 Status and Control Registers.....	9-13

PLC_Stat.....	9-13
PLC_Cmd	9-14
PLC_Count.....	9-14
PLC_Clock	9-14
MC_Stat	9-14
MC_Response	9-15
MC_Count	9-15
Out_PLC	9-15
Out_MC.....	9-15
In_PLC	9-15
In_MC.....	9-16
INT_REQ_OS	9-16
INT_REQ_RTOS	9-16
INT_RES_OS.....	9-16
INT_RES_RTOS.....	9-16
PLC_Ident	9-17
MC_Ident.....	9-17
PLC_Phase.....	9-17
MC_Phase	9-17
PLC_Result.....	9-17
9.8 Operation Registers	9-18
HINT - Host Interrupt Control and Status Register	9-18
LINT - Local Interrupt Control and Status Register	9-18
Interrupt Mailboxes	9-19
9.9 Interrupt Handshaking.....	9-21
PCI Interrupt to the PPC-P11.1.....	9-21
Receiving Interrupt Response from PPC-P11.1	9-22
Enabling the PCI Interrupts.....	9-23
9.10 System Initialization	9-24
PLC to PPC-P11.1 Communication.....	9-24
PLC / PPC-P11.1 Initialization Sequence.....	9-24
PLC / PPC-P11.1 Program Download or Shutdown Sequence	9-25

10 Coordinated Motion 10-1

10.1 Standard Coordinated Motion	10-1
Associated Task Parameters.....	10-1
Normal Case Kinematics	10-2
Special Case Kinematics	10-2
10.2 Coordinated Articulation.....	10-3
Differences from Normal Coordinated Motion	10-3
Applications using Coordinated Articulation	10-3
Block Functionality	10-3
System Considerations	10-5
Control and Status Registers.....	10-6
Coordinated Articulation Program Variables.....	10-9
Initializing and Synchronizing Coordinated Articulation.....	10-12

10.3 Kinematic Library	10-13
Standard Coordinated Motion Kinematics	10-13
Coordinated Articulation Kinematics	10-18
11 Link Ring Functionality	11-1
11.1 Link Ring Interface Overview	11-1
Link Ring Master	11-1
Link Ring Slave	11-1
Passive Participant (Repeater)	11-1
Link Ring Types	11-1
11.2 Hardware Setup	11-2
Single Ring	11-2
Double Ring	11-4
Setting SERCOS Baud Rate in Drive	11-5
11.3 Software Setup	11-5
Registers, Parameters, and Bits	11-5
Control Settings for SERCOS	11-7
Control Settings for Link Ring	11-8
Programming Icons	11-9
11.4 Fault Tolerance in Double Ring	11-12
Fault in Primary Ring	11-12
Fault in Secondary Ring	11-12
Fault in Primary and Secondary Rings	11-13
Clearing A Redundancy Loss	11-14
12 Error Reaction	12-1
12.1 Overview	12-1
Error Types	12-2
Error Levels	12-3
Configurable Error Reaction	12-5
Error Diagnostics and Logging	12-5
12.2 Task Association and Setup	12-7
Task Error Reaction Setup	12-7
12.3 Motion Type Error Reaction	12-10
Special Case Error Reactions for the ELS system	12-11
12.4 Drive Error Reaction	12-12
Power Supply Error Reaction	12-12
Disable Axis Command (Ab)	12-12
Drive Halt Setup (AH)	12-13
13 Communication Protocols	13-1
13.1 Protocol Overview	13-1
13.2 Communication Protocols	13-2
ASCII Protocol	13-2
SIS Protocol	13-3
Communication Parameters	13-3

13.3	Direct ACSII Communication	13-4
	VisualMotion ASCII Protocol.....	13-4
	Parameter Class and Subclass.....	13-7
	User Program Variable Class and Subclass.....	13-13
	PID Class and Subclass	13-14
	Point Tables Class and Subclass	13-15
	Event Tables Class and Subclass	13-18
	Program Communication Class and Subclass	13-20
	Functions Class and Subclass.....	13-32
	I/O Registers Class and Subclass	13-34
	Sequencer Data Class and Subclass	13-37
	Control PLS Class and Subclass.....	13-41
	Zones Class and Subclass	13-44
13.4	SIS Communication	13-46
	SIS Protocol	13-46
	Telegrams	13-46
	Telegram Header	13-49
	User Data Header	13-50
	GPP-specific Command / Response Parameter Telegrams	13-54
14	VisualMotion BTC06 Teach Pendant Interface	14-1
14.1	Overview	14-1
14.2	BTC06 Teach Pendant Screens	14-1
	Menu Map	14-2
14.3	BTC06 Teach Pendant Setup	14-4
14.4	BTC06 Keyboard Operation.....	14-6
	Keyboard Map.....	14-8
	Cursor Control and Editing.....	14-9
	Number or Letter Selection	14-9
	Jogging Control.....	14-9
	Task Control.....	14-10
	Teach Control.....	14-10
14.5	F1 Program Menu	14-11
	Sequencer Editing (F4)	14-12
14.6	F2 Table Edit Menu.....	14-14
	Absolute Table Menu (F1)	14-14
	Relative Table Menu (F2)	14-15
	Event Table Menu (F3)	14-16
	Integer Table Menu (F4)	14-17
	Floating Table Menu (F5).....	14-17
	Global Integer Table Menu (F6).....	14-18
	Global Floating Table Menu (F7)	14-18
14.7	F3 Jog Menu	14-19
	Jog Systems	14-20
	Jog Method	14-20
	Teaching Points	14-20

Jog Fine Adjustments	14-21
14.8 F4 Control Menu	14-21
Control Menu: Auto Run/Hold Mode.....	14-22
Control Menu: Auto Step Mode.....	14-23
Control Menu: Manual Mode.....	14-24
14.9 F5 Register I/O Menu.....	14-24
14.10 F6 Parameter Menu	14-25
F1 - Card Parameter Screen.....	14-26
F2 - Axis Parameter Screen.....	14-26
F3 - Task Parameter Screen.....	14-27
F4 - Drive Parameter Screen	14-28
14.11 F6 Security Menu	14-29
14.12 F8 Diagnostics Menu	14-30
14.13 Error Screen.....	14-31
15 Textual Language Programming	15-1
15.1 Overview	15-1
15.2 Directives.....	15-1
15.3 VisualMotion Textual Instructions	15-2
Instruction Format.....	15-2
AXIS/EVENT.....	15-4
AXIS/HOME	15-5
AXIS/INITIALIZE	15-6
AXIS/MOVE (Single Axis, Non-Coordinated).....	15-7
AXIS/RATIO	15-8
AXIS/SPINDLE (Continuous Velocity Mode	15-9
AXIS/START	15-10
AXIS/STOP	15-11
AXIS/WAIT (Axis Wait For In-Position).....	15-12
CALL (retval = CALL).....	15-13
CAM/ACTIVATE	15-14
CAM/ADJUST	15-15
CAM/BUILD.....	15-16
CAM/INDEX	15-18
CAM/STATUS.....	15-19
CAPTURE/ENABLE.....	15-20
CAPTURE/SETUP	15-21
COORD_ARTICULATION	15-22
DATA/SIZE.....	15-23
DEFINE (Label a Variable)	15-24
DELAY (Suspend Task Execution).....	15-25
ELS/ADJUST (Adjust ELS Axis)	15-25
ELS/GROUPM	15-27
ELS/GROUPS.....	15-28
ELS/MODE (Set ELS Axis Mode).....	15-29
ELS/MASTER	15-30

EQU (Equate)	15-30
EVENT/DONE (Signal Event Completed)	15-31
EVENT/END (Mark End of Event)	15-32
EVENT/START (Start of Event function)	15-32
EVENT/TRIGGER (Trigger a Task Event).....	15-32
EVENT/WAIT (Pause Task for Event Done Signal)	15-33
FUNCTION/ARG.....	15-34
FUNCTION/END.....	15-35
FUNCTION/START.....	15-35
GOSUB (Go To Subroutine)	15-36
GOTO (Go To Mark)	15-37
IF (If-Else-Endif Conditional Branch)	15-38
KINEMATIC (Use a Kinematic Definition for a Task)	15-40
LOCAL/VARIABLE.....	15-41
MESSAGE/DIAG (Task Diagnostic Message Definition)	15-42
MESSAGE/STATUS (Task Status Message Definition).....	15-43
MOVE/CIRCLE (Coordinated Move with Circular Interpolation)	15-44
MOVE/JOINT (Coordinated Move Joint Point to Point).....	15-45
MOVE/LINE (Coordinated Move with Straight Line Interpolation).....	15-46
PARAMETER/BIT (Initialize Parameter Bit)	15-47
PARAMETER/GET (Load Parameter to a Variable)	15-48
PARAMETER/INIT (Initialize a Parameter)	15-49
PARAMETER/SET (Set a Parameter).....	15-50
PATH/ABORT (Aborts Coordinated Motion).....	15-51
PATH/POSITION (Get Current Path Absolute Position)	15-52
PATH/RESUME (Resume Coordinated Motion).....	15-53
PATH/STOP (Halt Coordinated Motion)	15-53
PATH/WAIT (Pause Program for Motion).....	15-54
PID/CONFIG	15-55
PLC/CLEAR (Clear I/O Register Bit)	15-57
PLC/READ (Read I/O Register(s))	15-57
PLC/SET (Set I/O Register Bit).....	15-58
PLC/TEST (Test I/O Register Bit).....	15-59
PLC/WAIT (Pause Program for I/O)	15-60
PLC/WRITE (Write to I/O Register(s))	15-60
PLS/INIT.....	15-61
REGISTRATION	15-63
RETURN (Return From Subroutine).....	15-64
ROBOT/ORIGIN	15-64
ROBOT/TOOL	15-65
ROTARY/EVENT	15-65
SEQUENCER	15-66
SEQ/LIST	15-67
SEQ/STEP	15-68
TASK/AXES (Task Axes Definition).....	15-69
TASK/END (Mark the End of a Task)	15-70

TASK/START (Define the Start of a Task(s))	15-71
VAR/INIT	15-72
VIRTUAL/MASTER.....	15-73
16 Index	16-1
17 Service & Support	17-1
17.1 Helpdesk	17-1
17.2 Service-Hotline	17-1
17.3 Internet	17-1
17.4 Vor der Kontaktaufnahme... - Before contacting us.....	17-1
17.5 Kundenbetreuungsstellen - Sales & Service Facilities	17-2

1 VisualMotion 9 Overview

1.1 System Overview

VisualMotion is a programmable multi-axis motion control system capable of controlling up to 40 intelligent digital drives from Bosch Rexroth. The PC software used for programming and commissioning is VisualMotion Toolkit.

VisualMotion 9 supports the following hardware form factors and firmware versions:

- PPC-R using GPP 9 firmware (RECO-version)
- PPC-P11.1 using GMP 9 firmware (PCI-version)

1.2 GPP 9 System Overview

The PPC-R is a stand-alone multi-axis motion control. It has the RECO02 form factor, a form factor used by Bosch Rexroth for motion controls, PLCs and I/O modules. These devices share the RECO02 back-plane bus for data exchange.

It is recommended to use the VisualMotion motion control with Bosch Rexroth's DIAX04 and/or ECODRIVE03 digital servo drives. The communication between control and digital servo drives is performed using the SERCOS interface, the international standard for real-time communication for digital servo drives.

VisualMotion can provide multi-axis coordinated or non-coordinated motion control with tightly integrated logic control functions. The GPP 9 firmware supports a variety of applications, from general motion control to sophisticated multiple master electronic line shafting (ELS) and robotics.

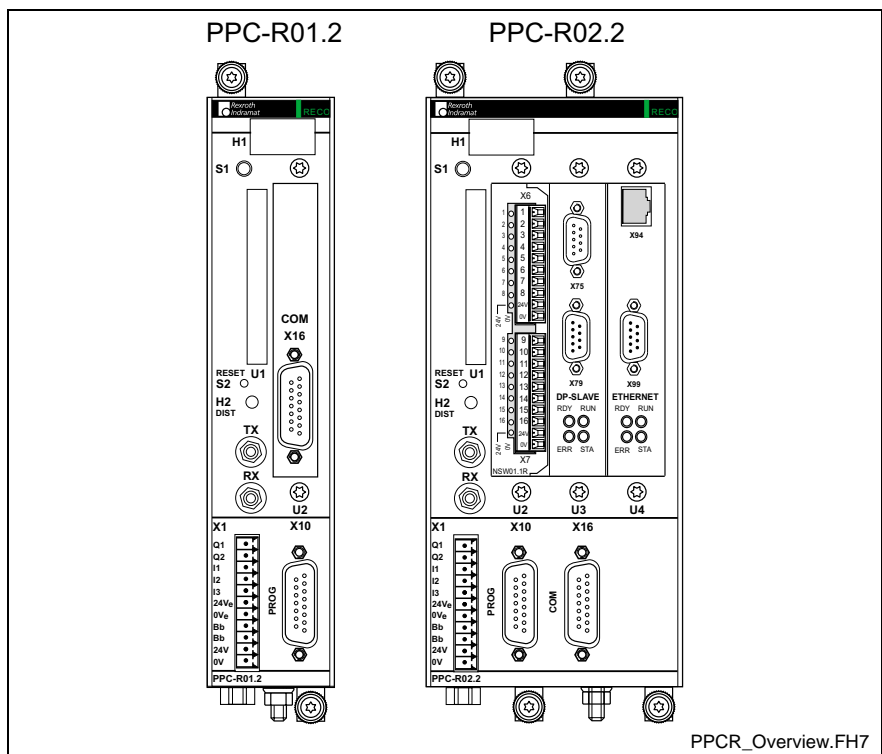


Fig. 1-1: PPC-R Motion Control

GPP 9 System Components

The VisualMotion 9 system is comprised of the following components:

- PPC-R control using GPP 9 firmware
- RECO02 I/O modules (Local and SERCOS)
- VisualMotion Toolkit (VMT) Windows program for motion control programming, parametrization, system diagnostics and motion control management. VMT also includes DDE and OPC servers. These servers support communication between Windows programs and the control.
- Up to 40 intelligent digital drives can be connected to one control over the SERCOS ring
 - DIAX04 (using SSE03 or ELS05 firmware) drives and motors
 - ECODRIVE03 (using SMT02, SGP01, SGP03 or SGP20 firmware) drives and motors
 - ECODRIVE C (using MPG01 firmware) drives and motors
- HMI interfaces (BTC06, BTV04, BTV05, BTV06)

GPP 9 PLC Support

The Bosch Rexroth MTS-R is a PLC unit that interfaces with the VisualMotion control (PPC-R) and is available preconfigured in two sizes.

- MTS-R01.1 with one expansion slot
- MTS-R02.1 with three expansion slot

Note: The expansion slot(s) on the MTS-R can be configured with fieldbus master interface or serial interface cards.

GPP 9 Interface Support

VisualMotion 9 supports the following interfaces:

Fieldbus Interfaces

- Profibus-DP slave interface (32 words) ^{Note 1.}
- Interbus slave interface (16 words) ^{Note 1.}
- DeviceNet, ControlNet or EtherNet/IP slave interface (32 words)

Note: When using EtherNet/IP in a VisualMotion 9 system, no other fieldbus interface card (i.e., Profibus, DeviceNet, ControlNet, Interbus) or the MTS-R PLC interface can be installed.

EtherNet/IP uses firmware FMC-ETH01*-PHT-02VRS-NN.

Note 1: The word size in parenthesis indicates the maximum number of words allowed in the cyclic telegram for both the Input and Output directions.

Additional Interfaces:

- Option Card **P**rogrammable **L**imit **S**witch (16 or 32 outputs)
- Link Ring for Master/Slave interfacing of VisualMotion controls
- Ethernet Interface

Drive I/O Support

Bosch Rexroth digital drives support the following I/O devices:

- DEA0x.2M (x = 4, 5 or 6) I/O cards for DIAX04 digital drives
- EMD I/O module using the EcoX interface for DKC22.3 digital drives using SGP20 firmware

1.3 GMP 9 System Overview

The PPC-P11.1 is a PC-based stand-alone multi-axis motion control. In a typical configuration, the host PC contains a Logic Controller (SoftPLC) which handles the system logic, fieldbus and Ethernet communication.

Just like the PPC-R, the PPC-P supports Bosch Rexroth DIAX04 and ECODRIVE03 digital servo drives. Communication between the control and digital servo drives is performed via the SERCOS interface.



ppc_pci.tif

Fig. 1-2: PPC-P Motion Control

GMP 9 Firmware Features

All firmware functionality supported in GPP 9 will also be supported in GMP 9 with the following restriction:

- VisualMotion fieldbus slave interfaces are not supported
- Ethernet interface is not supported

GMP 9 System Components

The VisualMotion GMP 9 system is composed of the following components:

- PPC-P control using GMP firmware
- Optional SERCOS RECO02 I/O modules
- VisualMotion Toolkit (VMT) Windows program for motion control programming, parametrization, system diagnostics and motion control management. VMT also includes DDE and OPC servers. These servers are the communication protocol between Windows programs and the control.
- Up to 40 intelligent digital drives can be connected to one control over the SERCOS ring
 - DIAX04 (using SSE03 or ELS05 firmware) drives and motors
 - ECODRIVE03 (SMT02, SGP01, SGP03 and SGP20 firmware) drives and motors
 - ECODRIVE C (using MPG01 firmware) drives and motors
- HMI interfaces (BTC06, BTV04, BTV05, BTV06)

Note: When using VisualMotion's I/O Setup tool to assign registers to physical outputs, the location (either input or output registers) will determine which device is the "master" of the particular set of physical outputs. If they are mapped to the PPC output section, then the PPC will have control of the outputs. If they are mapped to the PPC input section, then the SoftPLC will have control over the physical outputs.

GMP 9 Interface Support

VisualMotion GMP 9 supports the following interfaces:

- Optional **P**rogrammable **L**imit **S**witch Card with 16 or 32 outputs
- Link Ring for Master/Slave interfacing of VisualMotion controls

Drive I/O Support

Bosch Rexroth digital drives support the following I/O devices:

- DEA0x.2M (x = 4, 5 or 6) I/O cards for DIAX04 digital drives
- EMD I/O module using the EcoX interface for DKC22.3 digital drives using SGP20 firmware

2 VisualMotion Toolkit Menu Commands

2.1 Overview

Access to VisualMotion programs and system data is provided via menu selections and toolbar buttons. Main menu selections are available in project and service modes. Some menu items available under each main menu selection are dependent upon the current programming mode.

Programming Modes

The following table defines the main programming terms.



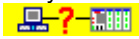



Term	Description
Project	VisualMotion data stored on a PC and not on the control. This data is a set of pre-defined file formats which, when combined together, make a snapshot of the data on the control. This data can be updated and edited using VisualMotion Toolkit 9 in offline or online modes.
Offline mode 	Programming mode used to create and manage project data when no communication between VisualMotion Toolkit 9 and the control exist. All data management will be directed only to the project files.
Online mode	Programming mode used for communicating between VisualMotion Toolkit 9 and the control for viewing, monitoring and editing of data. Synchronized and unsynchronized online programming modes are supported. The source of all data is the control. However, data will be saved to both the control and project in order to maintain synchronization.
Online Synchronized 	All project components contained in the offline project are updated on the control. Any change to project data has been downloaded to the control.
Online Unsynchronized 	Project components contained in the offline project are not updated on the control. However, the user can access control data, such as parameters, without downloading changes to the control. When editing online, the entire project and all tools display the unsynchronized icon as an indication that one or more components belonging to the project has been changed.
Synch. Project Component 	Synchronizes project components (when online or online unsynchronized) with the data on the control. This toolbar button is located to the left of the online toolbar button () and becomes visible after project data is modified and saved online.
Service mode 	Programming mode used to view and edit data on the control's memory. The project's data is not affected. Programming environment is similar to that of VisualMotion Toolkit 8.

Table 2-1: VisualMotion 9 Programming Modes

Toolbar Buttons

Some of the more frequently used menu commands are also displayed in the toolbar as buttons. VisualMotion Toolkit 9 displays menu items and toolbar icons based on the current programming mode. The following figure illustrates the menu items and toolbar buttons displayed for each programming mode.

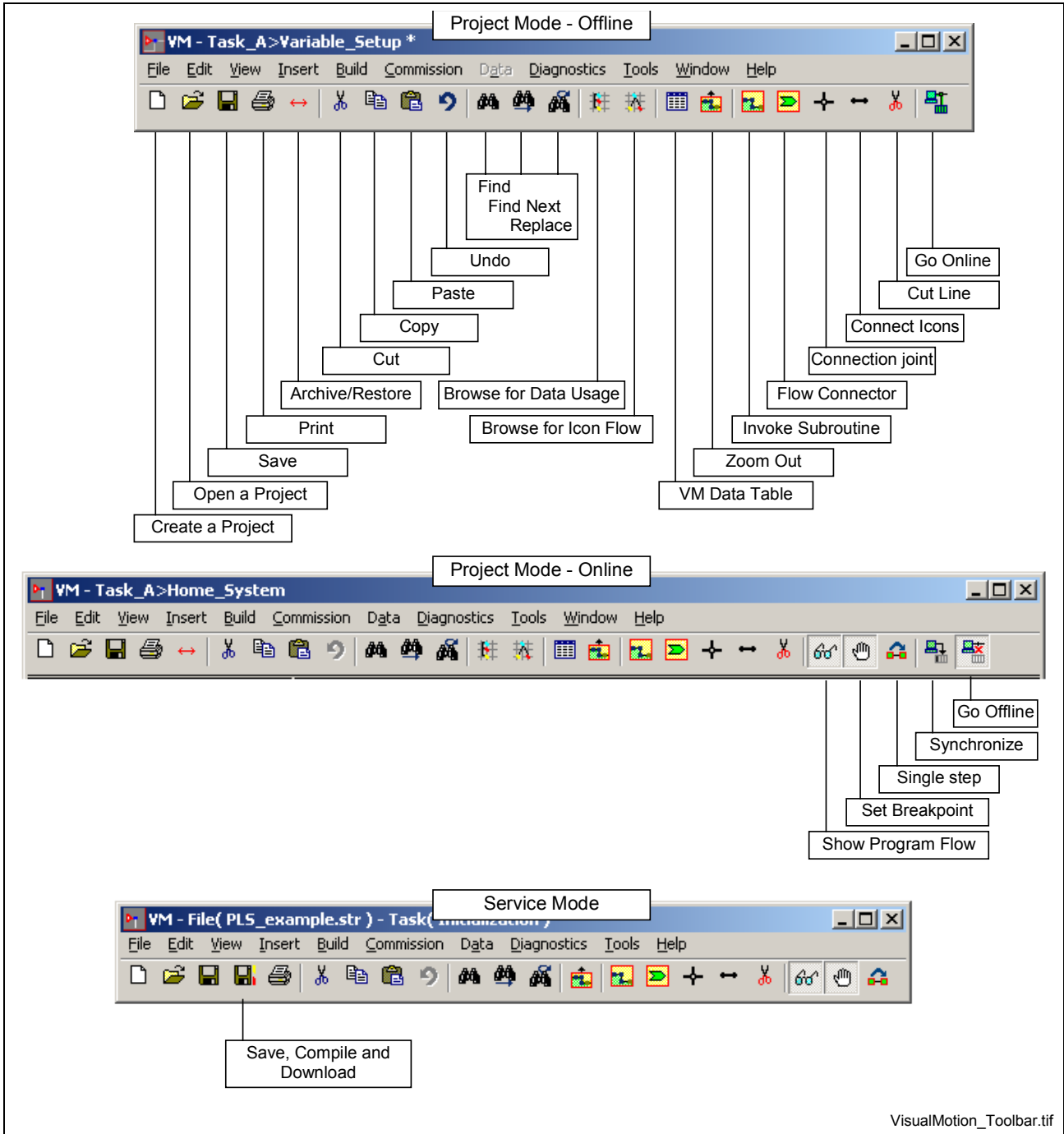


Fig. 2-1: VisualMotion Program Menu and Icon Button Bar

Project Navigator

VisualMotion Toolkit 9 includes a tool for navigating between Program Tasks, Initialization Subroutines, Subroutines and Event Functions. Clicking on any of the items, the user can easily switch between all tasks, subroutines and events available in the current program.

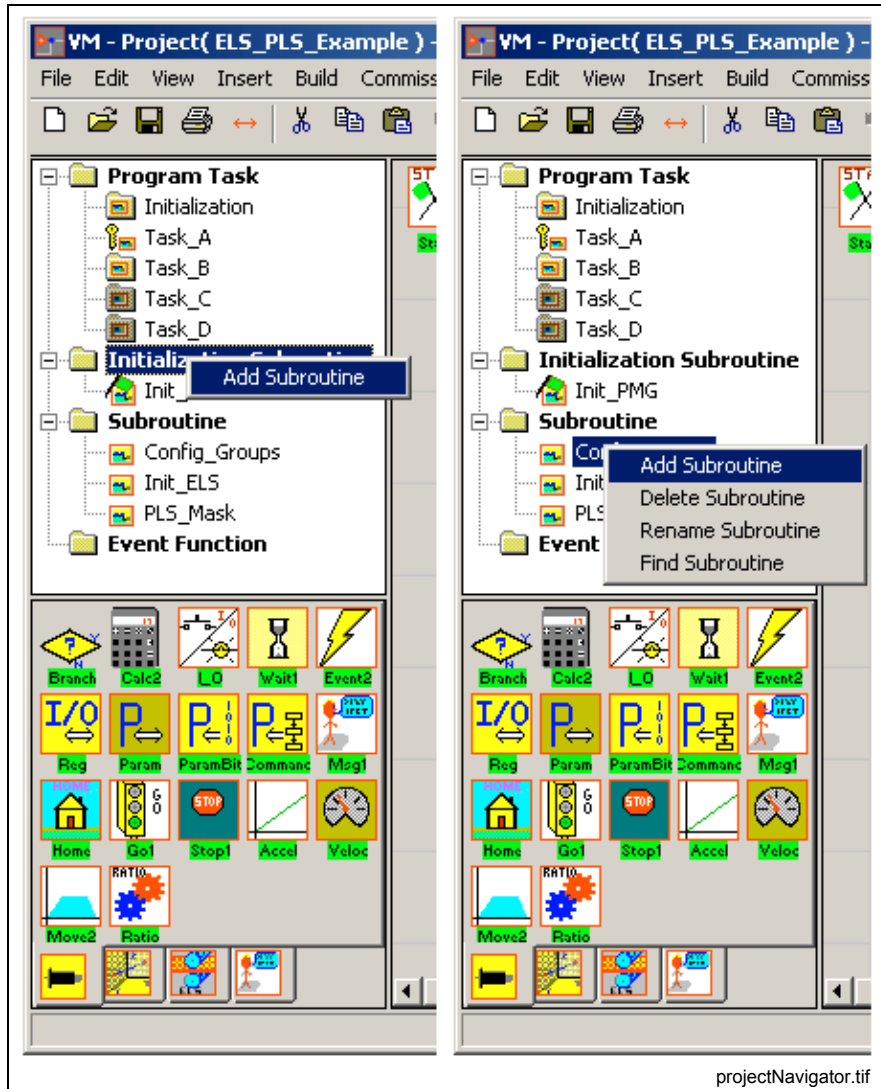


Fig. 2-2: Project Navigator

Navigator Features

From the Project Navigator tree structure, the user can make additions, delete, rename or find subroutines or events within the current project. Right clicking over a main item name (i.e., Subroutines) allows the user to **Add a Subroutine** or **Add an Event Function**.

Right clicking over a sub item name (i.e., Config_Groups) allows the user to perform any of the following functions:

- Add Subroutine (Add Event Function)
- Delete Subroutine (Delete Event Function)
- Rename Subroutine (Rename Event Function)
- Find Subroutine (Find Event Function)

Note: Adding a subroutine or event function from the Project Navigator assigns a name to the function and opens a new icon flow, displaying a start and finish icons. The actual assignment of the newly created subroutine or event function is performed by selecting and placing the appropriate icon into the program flow.

Note: The Find feature of the Project Navigator allows the user to find every instance of the subroutine or event in the current project.

Current Instruction Display

Selecting the **Show Program Flow (Status)** toolbar button, switches VisualMotion Toolkit's programming environment to status mode. While in status mode, program instructions currently executing are displayed in the top portion of the *Project Navigator* window. The icon palettes are not displayed while *Show Program Flow* is active.

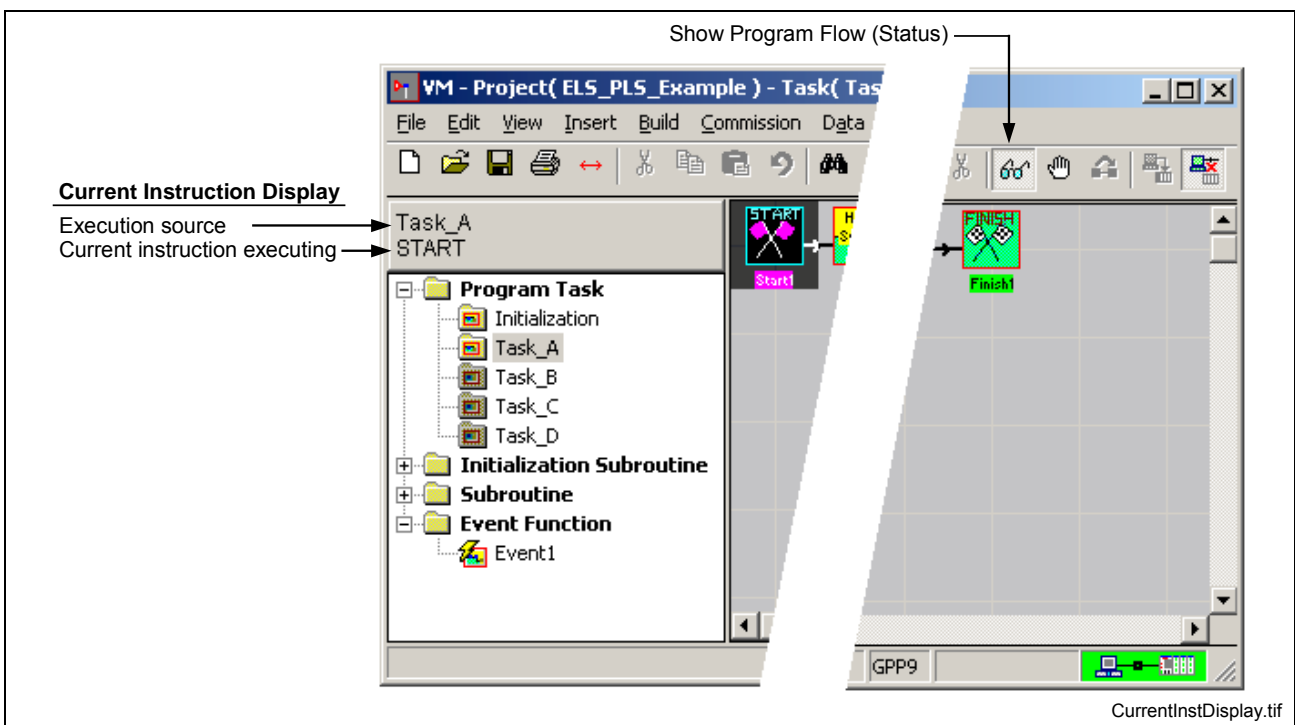


Fig. 2-3: Current Instruction Display

The current instruction is displayed in the *Project Navigator* whether or not a program is running. Once the control is initialized, the *Initialization* task runs to completion and program flow stops on the *Start* icon of each task (A-D), if programmed. This occurs before program execution is initialized by the user. Once the program is running, the location and current instruction execution displays change based on the current instruction (icon). The name displayed for the current instruction might not always match the expected name of the icon. The name displayed is that of the compiled instruction for that icon.

Displaying Task Function Arguments and Local Variables

Function arguments can be defined in the **Start** icon of any subroutine (initialization and standard). Local variables can be defined in the **Start** icon of any task, subroutine or event function. When a task, subroutine or event function is executing and the **Show Program Flow** toolbar button is set, any function argument and/or local variable are displayed below the **Current Instruction Display** section. The values of function arguments and local variables may change quickly while a program is running. However, the function arguments and/or local variables displayed can be helpful when using a breakpoint in the desired task or subroutine. The user can monitor the values for function arguments and/or local variables by single stepping through the task or subroutine. Refer to Set Breakpoint using VisualMotion Toolbar Buttons on page 2-144 for details.

Note: When no values are passed to the function arguments and/or local variables, the values display question marks.

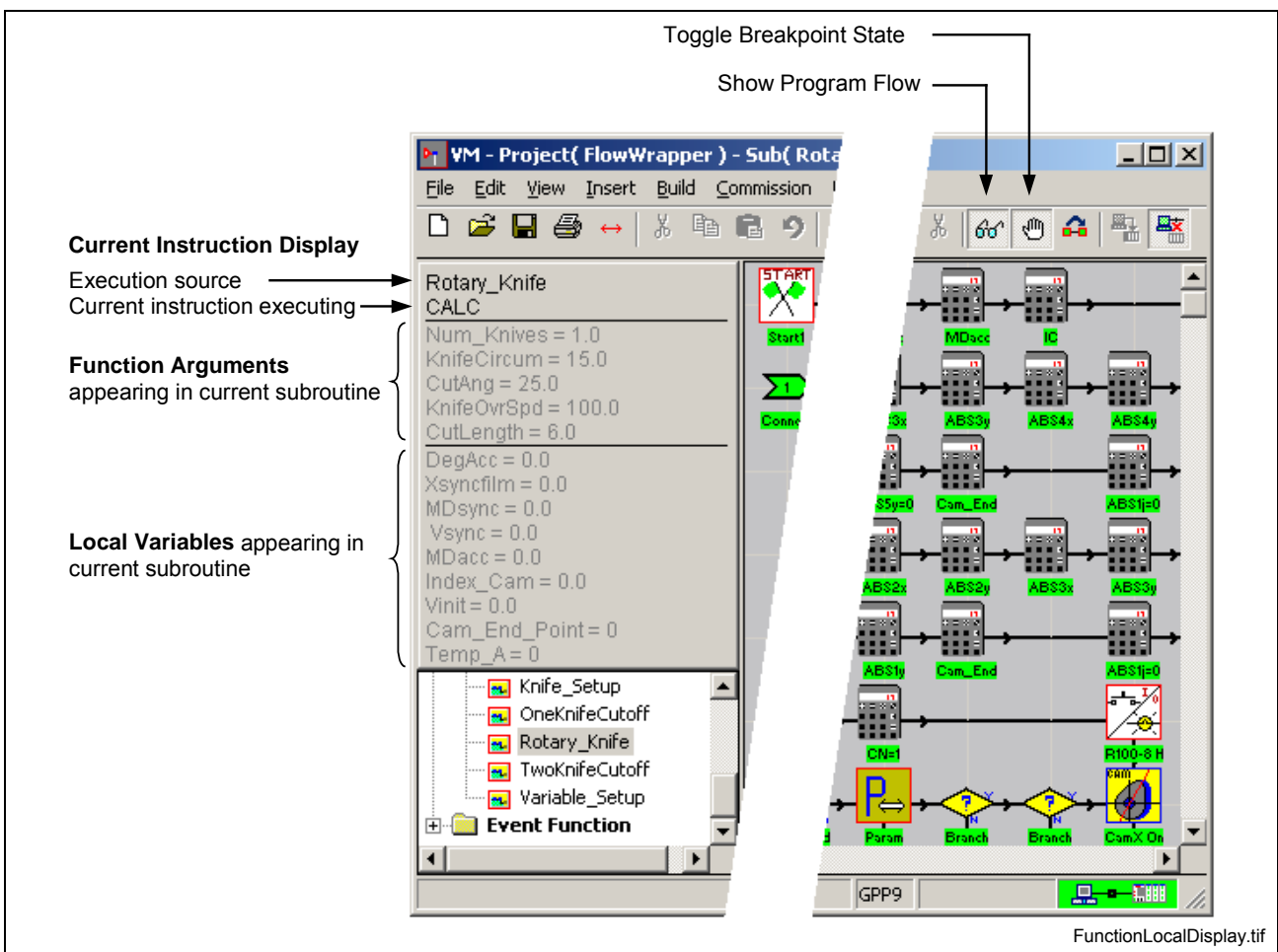


Fig. 2-4: Displaying Function Argument and Local Variables

2.2 The File Menu

The file menu displays different selections based on the mode of communication (offline/online/service).

The **F**ile menu commands are as follow:

- standard Windows functions (New, Open and Close)
- saving VisualMotion programs and projects
- switching between online and offline programming modes
- importing and exporting of project components
- printing VisualMotion programs/elements (variables, etc.)
- accessing sample programs and quick launching of recently open programs or projects

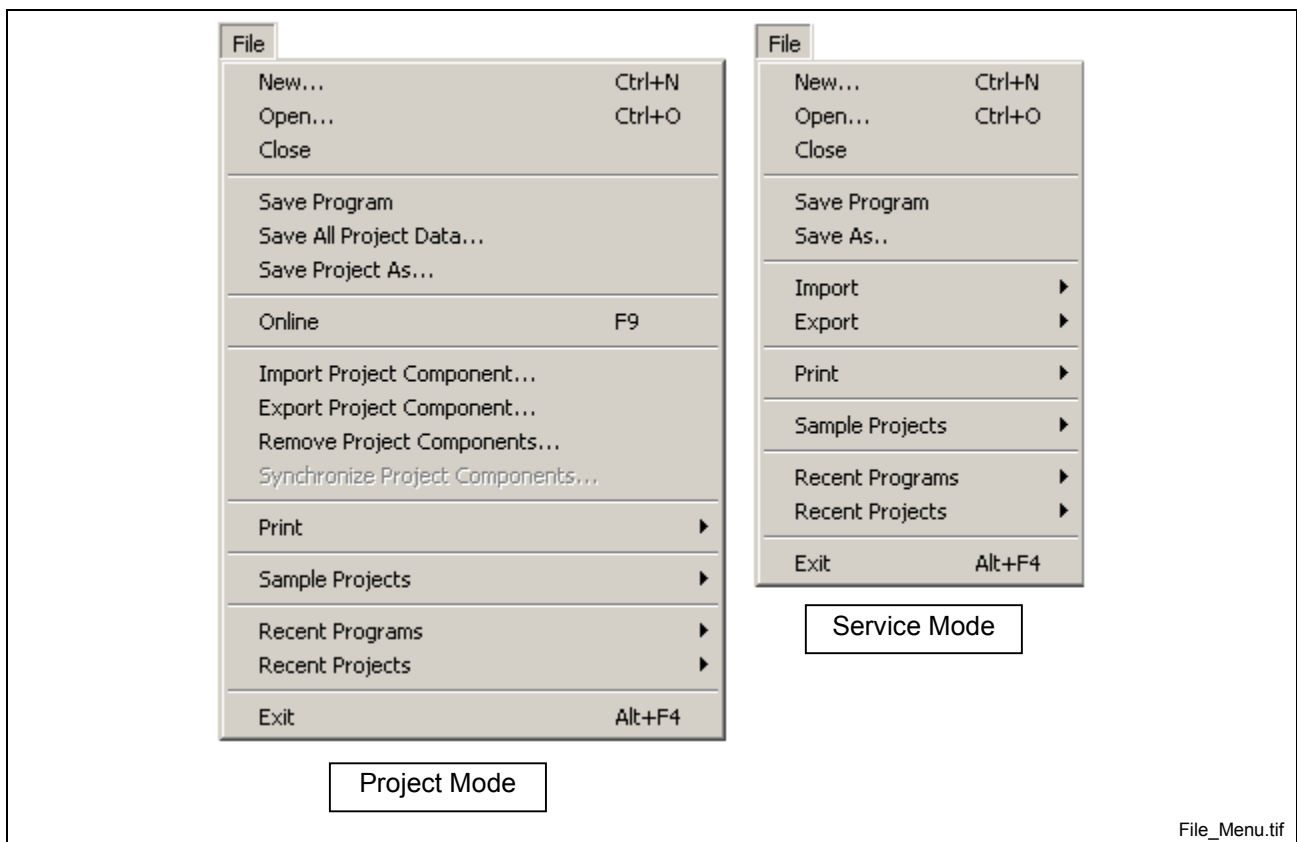


Fig. 2-5: The File Menu

File ⇒ New...

Selecting **File ⇒ New** opens the "What do you want to do" window in Fig. 2-6. From this window, the user can select a radio button for one of the following choices.

- Create a new project
- Create a new project from program and data on the control
- Create a new project from an existing *.str program
- Open existing project
- Open existing icon program
- View and edit control data in service mode

Note: If the Cancel button is pressed, VisualMotion Toolkit will be launched in service mode.

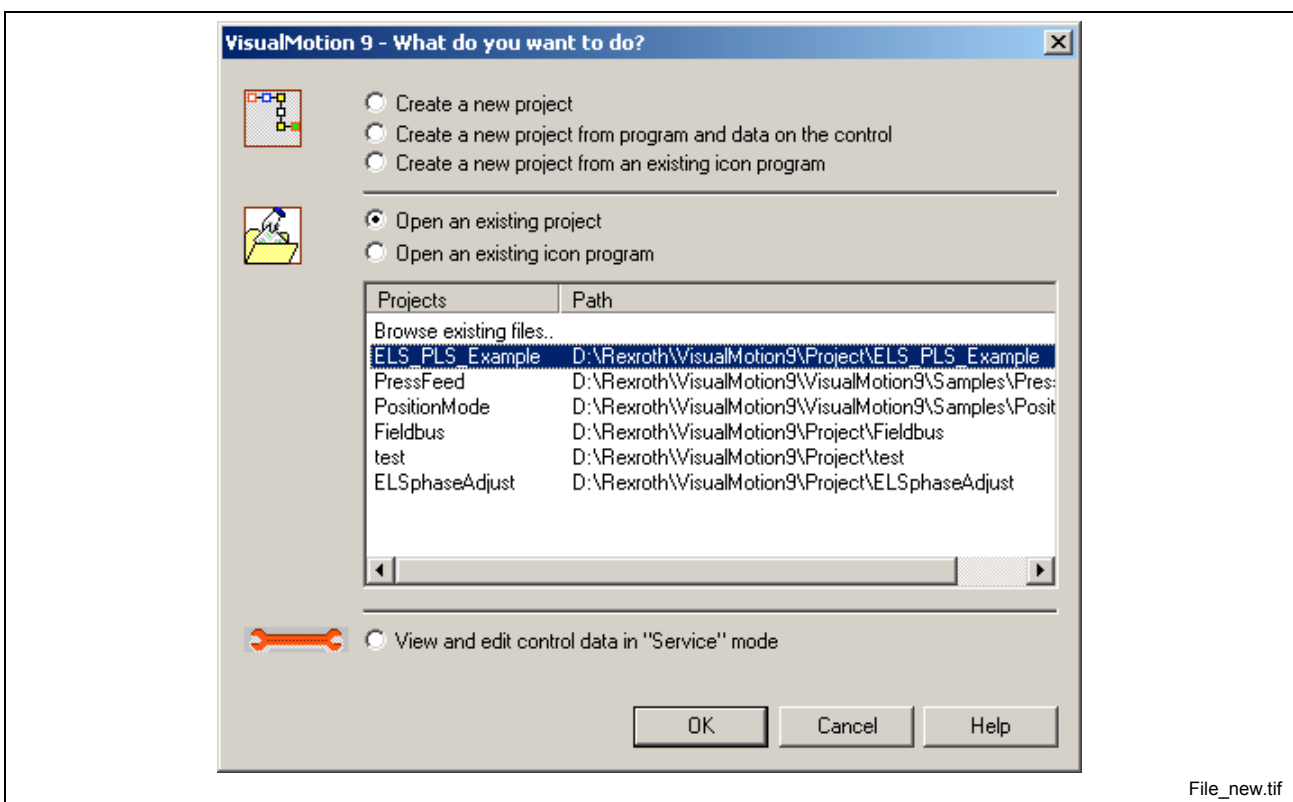


Fig. 2-6: New VM Program Development Window

Create a New Project

VisualMotion Toolkit 9 introduces a project structure that organizes all data relevant to an icon program under a specific folder. Any new program, regardless of the target firmware, that is created using VisualMotion Toolkit 9 is created and structured as a project.

When creating a new program, VisualMotion Toolkit 9 creates a project folder, named by the entry in the **Project name** field, where the icon program and all relevant data is stored. The project folder is created under the **Project Location** hard drive designation. The default project location is "\\Rexroth\VisualMotion9\project". All modifications made to any component of the project will be maintained in the project folder structure.

Note: The available icons and functionality are based on the selected **Target Firmware**. For example, if GPP8 is selected, the project will not contain an initialization task and only those icons available at the selected firmware level will be available.

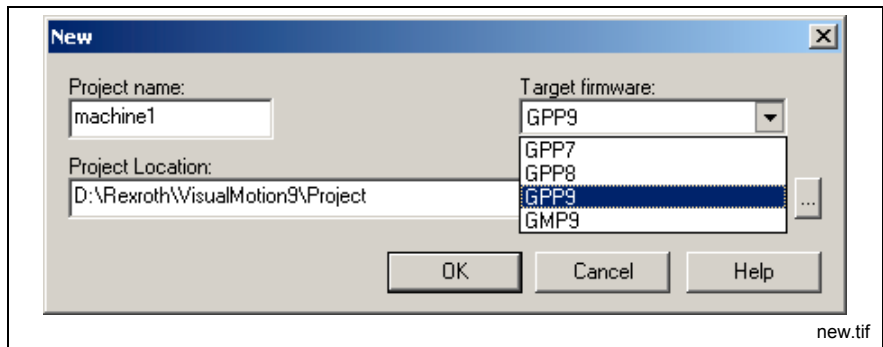


Fig. 2-7: New Project

For example:

If the name "machine1" is entered as the project name, the main folder will be named "machine1". When an icon program is created and downloaded to the control, it will receive the project name (machine1.str).

Project File (vmj)

VisualMotion Toolkit 9 creates a project file and saves it under the "machine1" project folder, i.e., "machine1.vmj". The project file contains project specific information about the assigned drives in an icon program used to initialize DriveTop, serial settings, etc. This file allows the user to copy or move the main project folder to any location while maintaining all required settings. Data is saved to the file when the OK button is clicked in the "Axis" or "ELS Group" icons, or at compile time. Data saved to the file includes the SERCOS address, name and mode of operation of drives. Drives added to the icons and then later removed will remain in the file until the file is rebuilt at compile time.

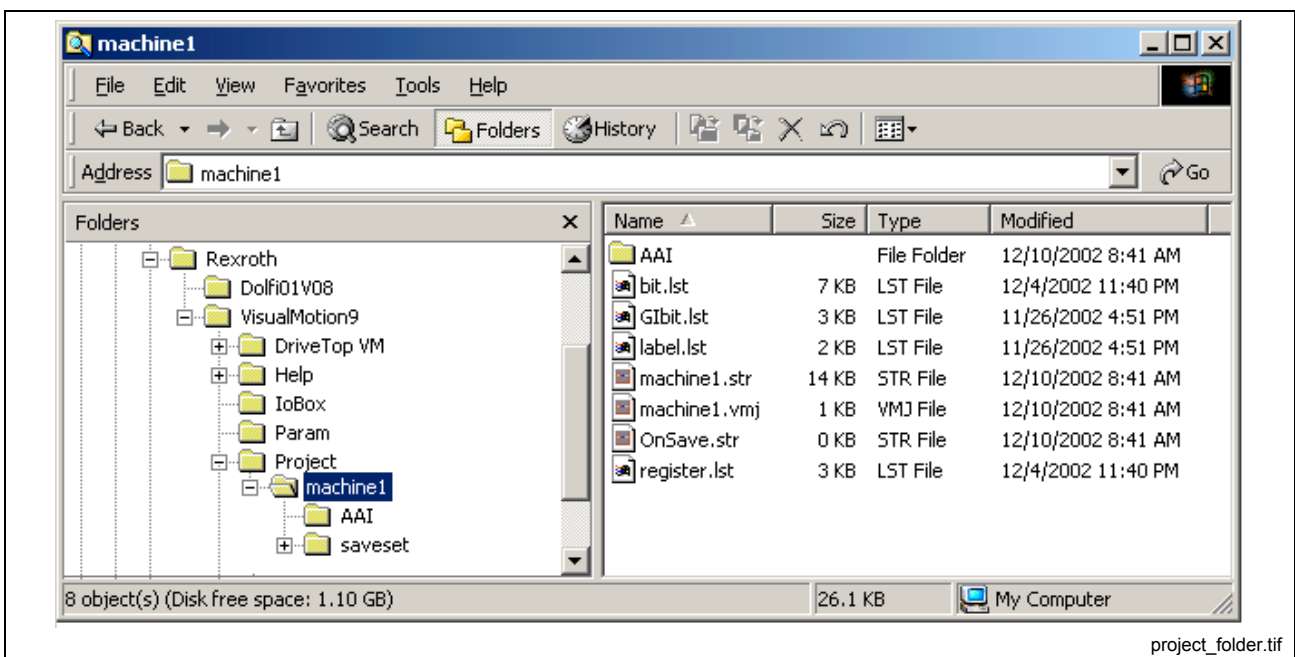


Fig. 2-8: Project Folder

Create a New Project from Program and Data on the Control

A new VisualMotion project is created from the source data retrieved from the control in online mode. When selected, the user selects the communication type (serial or Ethernet) and enters a new project folder name where the source data of the currently active program is stored. The source data consists of the following:

Note: This command can only be performed if an icon program was previously archived to the control's memory from the *Synchronize Project Data* window. Refer to *Synchronize Project Data* on page 2-11.

- Icon (*.str) program
- Project (*.vmj) file
- I/O Mapper
- System variables (Floats, Integers, Global Floats and Global Integers)
- Event data
- PLS data
- Points data
- Zone data
- System parameters

Create a New Project from an Existing Icon Program

This menu command is used to open an existing program (*.str) file for creating a new project. A new sub-folder is created containing a new project (*.vmj) and source (*.str) file.

Open Existing Project

This menu command is used to open a VisualMotion project (*.vmj) from a listing of up to 8 previously opened projects or browse to locate a project file on the hard drive.

The project's name and entire path is displayed. The number of displayed project files can be specified, up to a maximum of 8, from the *VisualMotion Options* window by selecting **Tools** ⇒ **Options...**

Open Existing Icon Program

This menu command is used to open a VisualMotion icon program (*.str) from a listing of up to 8 previously opened programs or browse to locate an icon file on the hard drive. VisualMotion 9 will open in service mode.

The program's name and entire path is displayed. The number of displayed program files can be specified, up to a maximum of 8, from the *VisualMotion Options* window by selecting **Tools** ⇒ **Options...**

Note: When opening older firmware source (*.str) files, the icon and functionality used with the older program will become available in VisualMotion 9.

View and Edit Control Data in Service Mode

This menu command is used to edit data in service mode. All tools and utilities available in service mode function the same as they do in VisualMotion 8. The intent of this mode is to provide access to a machine

in the field when no project data is available. From service mode, the user can perform backup and restore operations as performed with previous versions of VisualMotion Toolkit.

Open...

This menu command is used to open any of the following project or program files:

- Project file (*.vmj)
- Icon Motion File (*.str)
- Textual Motion Files (*.mtn)
- Embedded Icon Files (*.exb, *.exc)

Note: Embedded icon files require that they were compiled using the *Include compressed source* option available in VisualMotion 8.

Close

This menu command closes an existing project or icon program and switches VisualMotion Toolkit to service mode.

Save Program

An icon program (*.str), belonging to a project, can be modified and saved back to the project folder while in offline or service mode. When the project file (*.vmj) is opened and VisualMotion Toolkit is switched to online mode, all modifications are automatically detected and the user is asked to Save, Compile and Download changes.

Note: This menu command is only available in offline or service modes.

Save As...

This menu command is used to save programs and data when working in service mode.

Note: This command is only visible in the File menu when working in service mode.

Save All Project Data...

This menu command saves the current project data to the appropriate project folder location when modifications are made in offline mode. Any tools and utilities that have been modified and are currently opened will also be saved.

Save Project As...

This menu command copies project files from the source project folder to location specified in the **Project Location** field. The target firmware type can also be specified before saving the project. Only files relevant to the project are copied.

Note: This command is only available from a project in offline mode.

Online / Offline ... F8

This menu command switches VisualMotion Toolkit from offline to online mode. This feature requires a valid project program to be saved, compiled and downloaded to the control. If VisualMotion detects changes to the data in a project, the user is requested to Save, Compile and Download before switching the online mode. If VisualMotion Toolkit is currently online, the menu selection reads, "*Offline ...F8*".

Synchronize Project Data

The *Synchronize Project Data* window is displayed when a project is switched to online mode and data in the current project has changed from the last time it was downloaded.

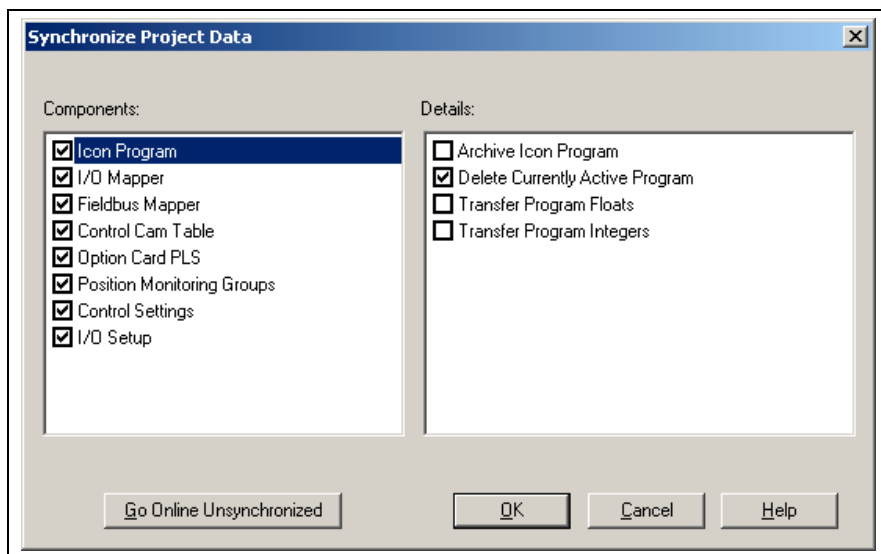


Fig. 2-9: Synchronize Project Data Window

Options

The **Options** section displays all project components that were detected to be different from those currently in the control's memory for the same project name. Only checked components and details are downloaded to the control.

Details

The **Details** section displays all sub-components that are available for the currently selected **Option**. By default, the **Delete Currently Active Program** detail is checked every time a modified icon program is downloaded. Project synchronization default settings can be modified from the *VisualMotion Options* window under the **Tools** ⇒ **Options** menu. Refer to Project Synchronization Default Settings on page 2-168 for details.

The **Archive Icon Program** detail from the **Icon Program** option is used to archive the currently opened icon source file (*.str) to the control's memory. This detail is required in order to create a new project from program and data on the control memory. Refer to Create a New Project from Program and Data on the Control on page 2-9 for details.

Go Online Unsynchronized

The **Go Online Unsynchronized** button is used to establish communication with the control without downloading any data.

Import Project Component...

When using the import feature, VisualMotion Toolkit is considered the target location. The source location is dependent on whether VisualMotion Toolkit is in offline or online mode.

Import Data in *Offline Mode*

The “*Import a Program and/or Data into the project*” window opens when selecting **File** ⇒ **Import Project Component...** in offline mode. Data is copied from a selected project or file to the project currently opened in VisualMotion Toolkit.

Transfer Data from

Specify the path and name of a project file (*.vmj) or individual file (i.e., system.prm) from where data will be imported to the current project. Select the browse button to help locate a file. Once a project or individual file is entered into the **Project path** field, the **Components** and **Details** fields display all data that is part of the project or file.

Note: This process can be used to import data from older file formats of VisualMotion into the current project.

Components:

Data stored on the hard drive, either from another project file (*.vmj) or individual file (i.e., system.prm), is display in the **Components** field. The components displayed are only those that are part of the project or file.

Details:

The details displayed are dependent on the component type selected. By default, all details are selected for each component. To de-select a detail, click on the appropriated checkbox. Once all components and details are selected, click the **OK** button to import the data to the current project opened in VisualMotion Toolkit.

Note: All data imported is automatically stored to the current project.

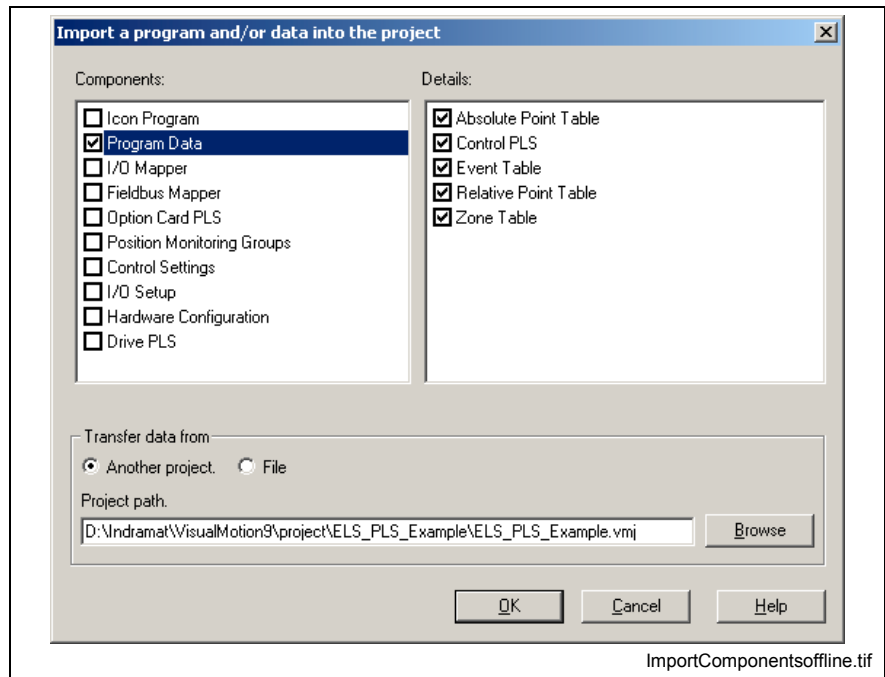


Fig. 2-10: Import a Program Data into the Project

Once the components have been imported from a project or file to the current project, the user can switch VisualMotion Toolkit to online mode. VisualMotion detects any modifications in data and request that the project be recompiled and downloaded to the control before switching to online mode.

Import Data in Online Mode

In online mode, selecting **File ⇒ Import Project Component...** opens the *Transfer Control Data to Project* window. Data is copied from the control's memory to the project currently opened in VisualMotion Toolkit.

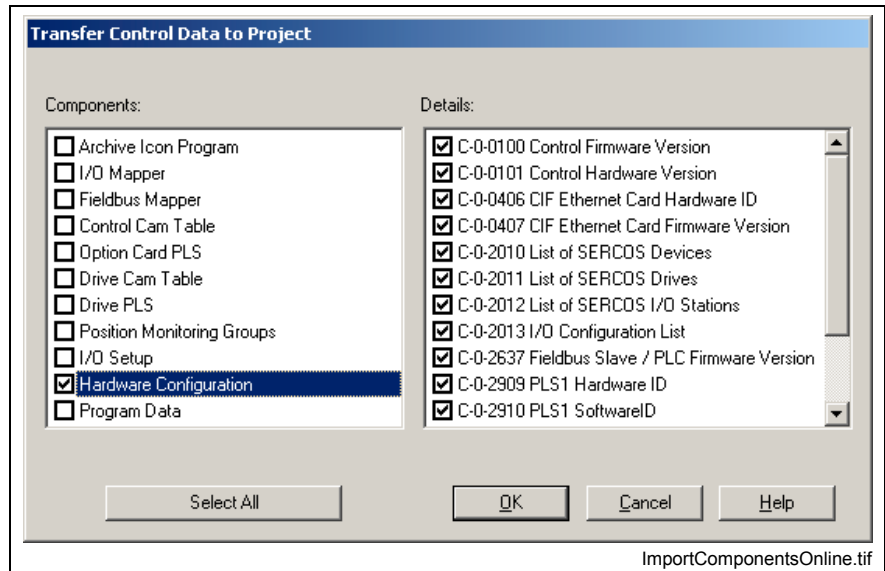


Fig. 2-11: Transfer Control Data to Offline Project

Component and detail data displayed are those found on the control's memory for the currently active program. By default, Hardware Configuration is checked. Select any additional components and details to import from the control's memory to the project currently opened in VisualMotion Toolkit. Click on the **Select All** button to check all available components on the control's memory and click the OK button.

Export Project Component

When using the export feature, the currently opened project is considered the source location. The target location is dependent on whether VisualMotion Toolkit is in offline or online mode.

Export Data in Offline Mode

The *Transfer project data to another project or to files* window opens when selecting **File** ⇒ **Export Project Component...** in offline mode. Data is copied from the project currently opened in VisualMotion Toolkit to another project file (*.vmj) or individual file.

Components:

Data components in the currently opened project are uploaded and displayed as available components that can be stored to another project file (*.vmj) or as individual data files. Components are selected by placing a check in the box next to the component label. Once a component is selected, details associated with the component are displayed to the right.

Details:

The details displayed are dependent on the component selected. By default, all details are selected for each component. To de-select a detail, click on the appropriated checkbox. Once all components and details are selected, click the **OK** button to export the data to the current project opened in VisualMotion Toolkit.

Transfer Data to

Specify the path and name of a project file (*.vmj) or folder (for an individual file) to where data will be exported from the current project. Select the browse button to help locate a file or folder.

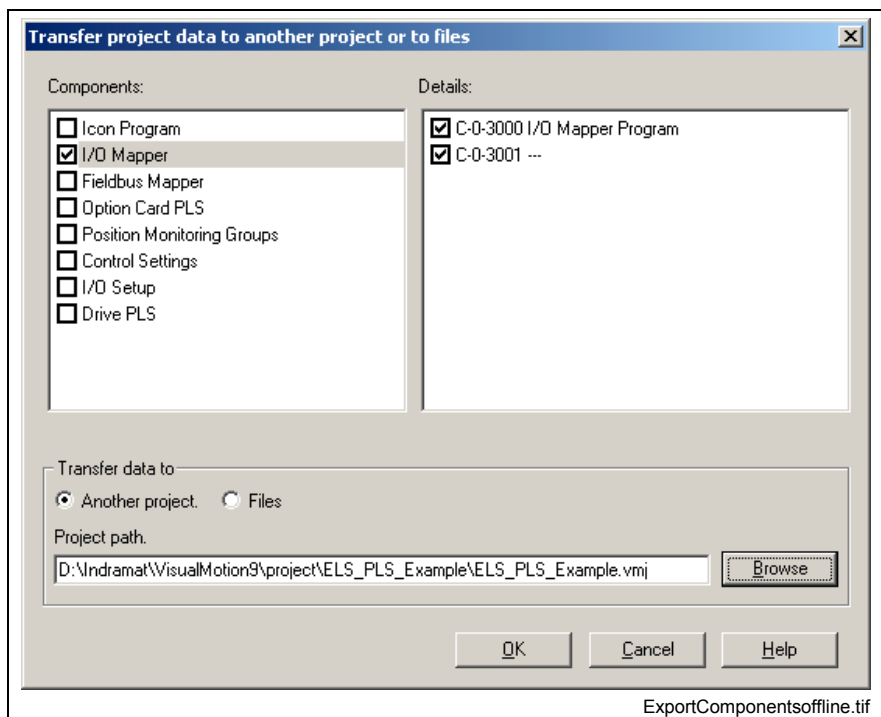


Fig. 2-12: Transfer Project Data to Another Project or to Files

Export Data in Online Mode

In online mode, selecting **File** ⇒ **Export Project Component...** opens the *Transfer Project Data to Control* window. Data is copied from the project currently opened in VisualMotion Toolkit to the control's memory.

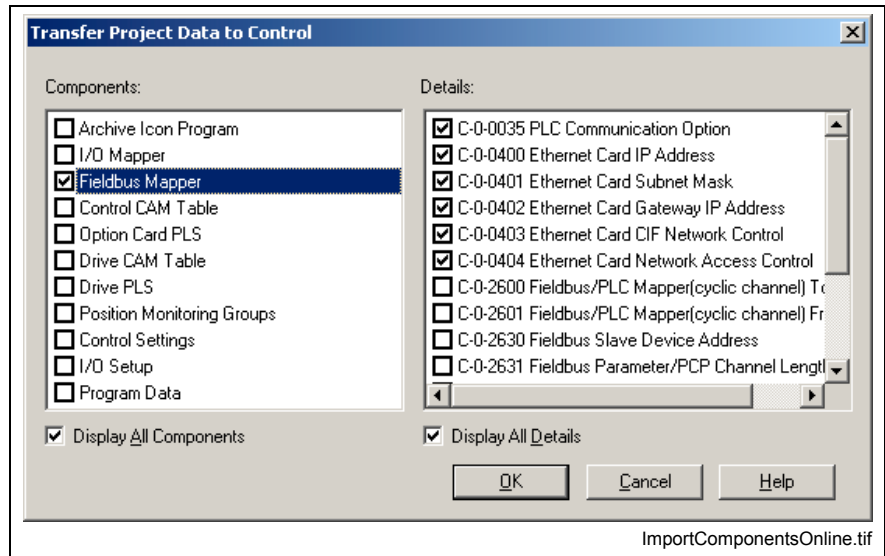


Fig. 2-13: Transfer Project Data to Control

Component and detail data displayed is that of the project currently opened in VisualMotion Toolkit. By default, no components are checked. Select those components and associated details to export to the control's memory and click the **OK** button. Selecting the *Display All Components* and/or *Display All Details* checkboxes displays additional components and details not necessary associated with the current project.

Note: Selecting components and details that are visible only when the *Display All...* checkboxes are selected will write default values to each parameter.

Remove Project Components

Remove project components allows the user to selectively remove complete components or just certain details of a component from the project currently opened in VisualMotion Toolkit.

Note: Project components removed using this feature are only removed from the project files. Project components downloaded to the control will remain resident in the control's memory.

Selecting **File** ⇒ **Remove Project Components** opens the window in Fig. 2-14. To remove a project component, uncheck the component(s) and/or detail(s) to remove from the project and click the **OK** button. The selected component and/or details will be removed from the project files.

Note: If a component or detail was removed in error, and was previously downloaded to the control, use VisualMotion's Import project Component feature. Refer to Import Data in Online Mode on page 2-13 for details.

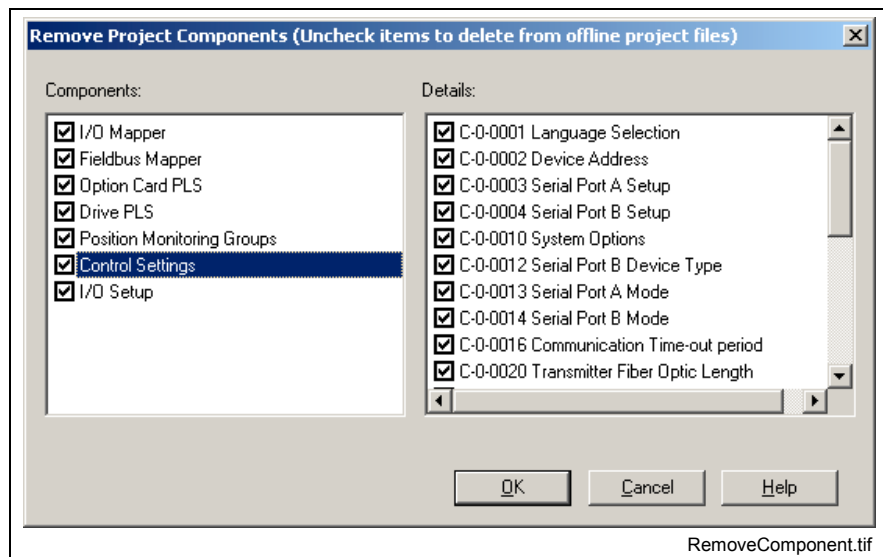



Fig. 2-14: Remove Project Components

Synchronize Project Components

Online editing of a project gives the user the flexibility to modify project components without interfering with a program downloaded and possibly running on the control.

The synchronization of a modified project to the data on the control is performed using the *Synchronize Project Component* toolbar button (). This button is only available when VisualMotion detects unsynchronized project data. An unsynchronized condition occurs when project data does not match the data on the control.

To synchronize modified project components online, select **File** ⇒ **Synchronize Project Component** or click on the *Synchronize Project Component* toolbar button. The *Synchronize Project Data* window opens, displaying project components whose offline data has changed since the last time the project was downloaded to the control.

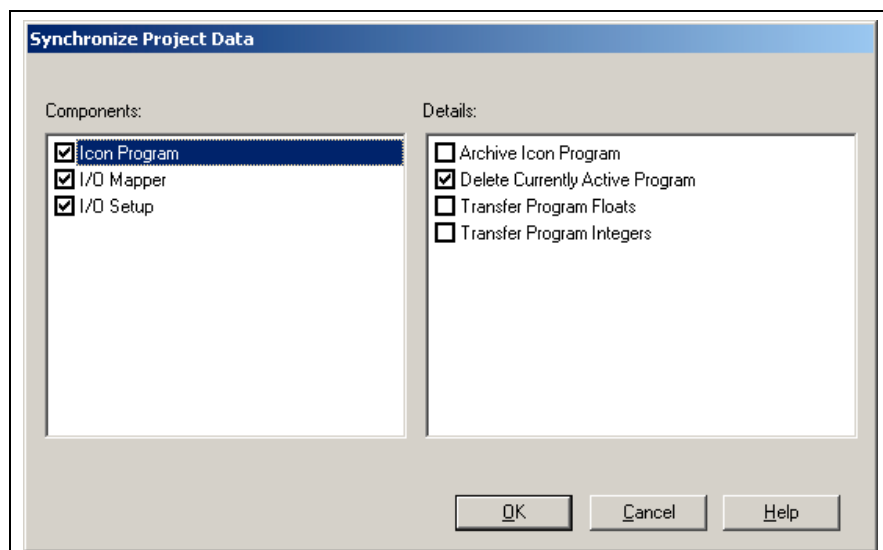



Fig. 2-15: Synchronize Project Data

At this point, the user downloads the modified project components to the control, and if no errors are detected, the project becomes synchronized.

Some components can be downloaded in phase 4, while others require the system to be in parameter mode. VisualMotion will notify the user if

the control needs to be in parameter mode before a component(s) can be downloaded. If all the components are **not** downloaded to the control, the project is considered unsynchronized and the state of the project will remain **unsynchronized**. As indicated by the unsynchronized status bar indicator .

Print Project Data...

Selecting **File** ⇒ **Print ...** from the File menu displays the Documentation Selection window. All **Program Data** is selected by default.

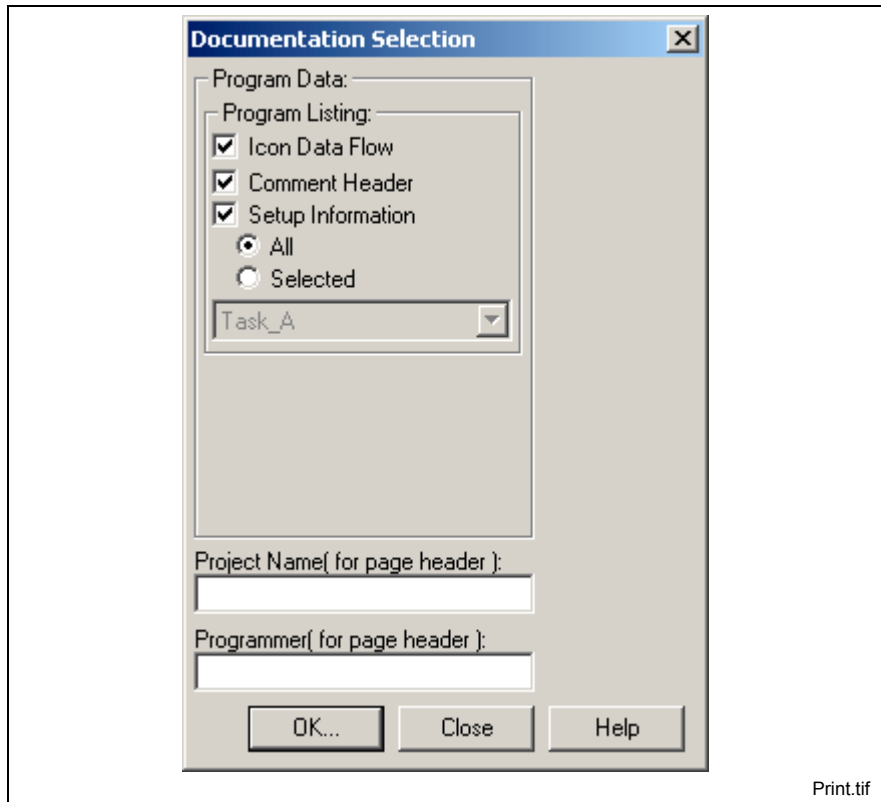


Fig. 2-16: “Documentation Selection” Window

Program Data

Selecting **Icon Data Flow** provides a graphic printout of a VisualMotion Icon Language window. For this function, you must have a graphics-capable printer. Text-based files, such as Text Language user programs or parameter files uploaded for viewing, may also be loaded and printed from Windows Notepad or another editing program.

Setup Information can be printed for “All” or a selected task, subroutine, or event.

Selecting **Variable Labels, Register Labels, Bit Labels, Other References and/or Subroutine List** prints a list of the respective labels for the task, subroutine, or event window currently displayed by VisualMotion.

The fields for **Project Name** and **Programmer** allow for adding the project name and programmer to the header on each printed page. The standard header contains the filename, task name, date, time and page number. Date and time are relative to the time of printout and are based on the time kept by the PC.

Program

Selecting the **Program...** button sends the selected Program Data to the configured printer.

Data

The Data button opens the *Print Variable Data* window. The user selects the variable type(s) to print and selects the Data Source (File or Card). The printout displays the ID, Label and value for all variable types selected.

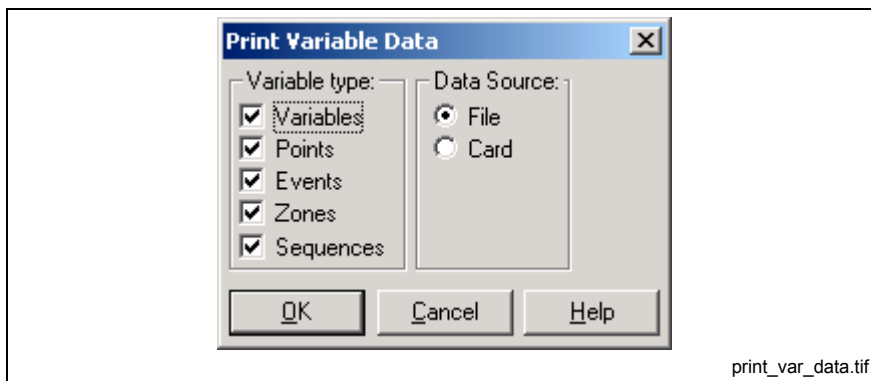


Fig. 2-17: Print Variable Data

Sample Programs

As part of VisualMotion Toolkit, five sample programs ("Measuring Wheel GPP", "Position Mode", "Flow Wrapper", "Coordinated Mode" and "Press Feed") have been included.

Recent Programs

This menu selection displays up to the last 8 (*.str) icon programs that the user can choose from to quickly launch the program in service mode.

Recent Projects

This menu selection displays up to the last 8 (*.vmj) project programs that the user can choose from to quickly launch the project.

Exit

Exits the current program, prompting if it has not been saved.

2.3 The Edit Menu

The **Edit** menu contains Windows editing features for icon-based programs, clearing the icon workspace, find/replace functions and view and adding of VisualMotion data types.

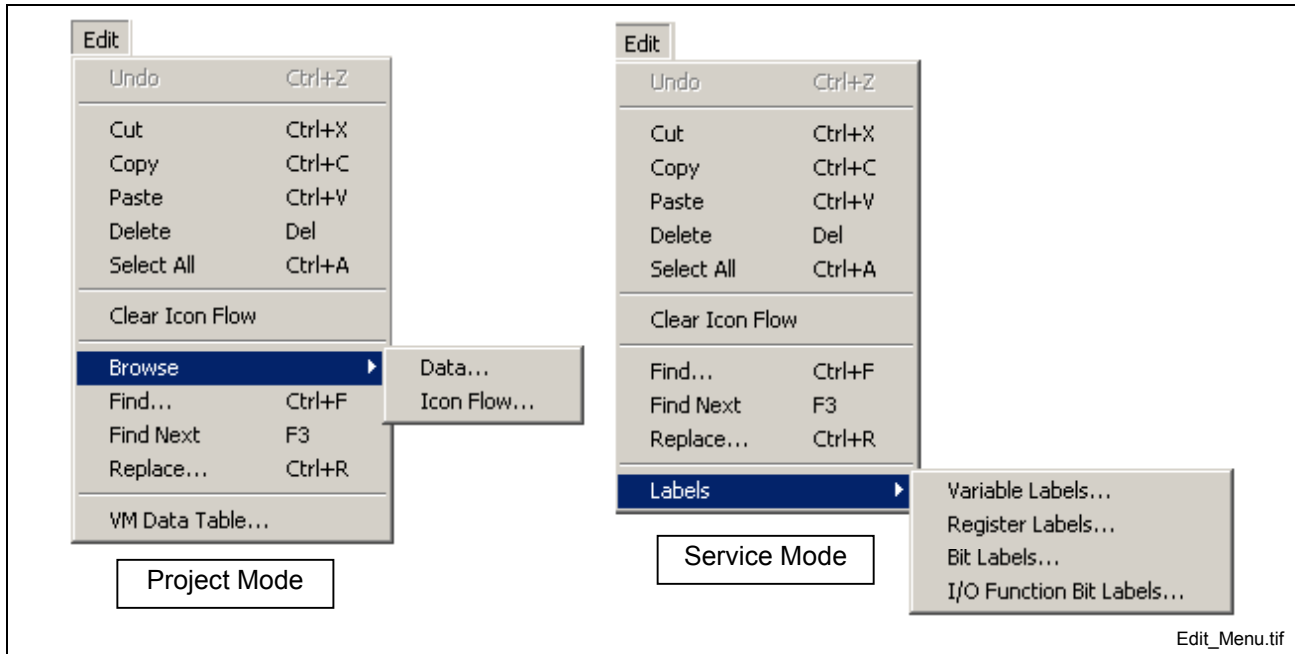


Fig. 2-18: The Edit Menu

Windows Editing Features

The Windows editing features are used to manipulate icons with the user program.

Undo (Ctrl+Z)

This menu command can undo the last icon edit.

Cut (Ctrl+X)

This menu command cuts the selected program flow to the paste buffer.

Copy (Ctrl+C)

This menu command copies the selected program flow to the paste buffer.

Paste (Ctrl+V)

This menu command enables the paste operation into the selected program space.

Delete (Del)

This menu command deletes the selected program flow from the program space.

Select All

This menu command selects the entire program flow of the current task, subroutine or event function.

Clear Icon Flow

This menu selection deletes all contents of the current VisualMotion task, subroutine, or event workspace of the open program. The Undo menu selection or icon can be used to undo only the last modification.

Browse


The browse feature in VisualMotion 9 allows the user to locate usage of program data in a project and usage of subroutines and event functions in the icon flow.

Browse Project for Data Usage

The Data menu selection allows the user to browse the current project for the usage of program data by type. The available data types are listed in the table below.

Register	Register Bit	Program Integer	Program Float
Global Integer	Global Integer Bit	Global Float	Absolute Point
Relative Point	Event Function	Zone	PLS

Table 2-2: Available Data Types for Browse

Selecting **Browse** ⇒ **Data** or clicking  opens the *Browse Project for Data Usage* window in Fig. 2-19. All program tasks, subroutines and event functions are scanned for program data types. In addition, the I/O Mapper and Fieldbus editors are opened, scanned and then minimized.

Note: The window can be resized to allow complete view of the scanned data type. All data types used in the current program are scanned and displayed at the instances the window is opened. Any data type added or modified in the program while the window is opened will not be scanned and added to the list. The window must be close and reopened.

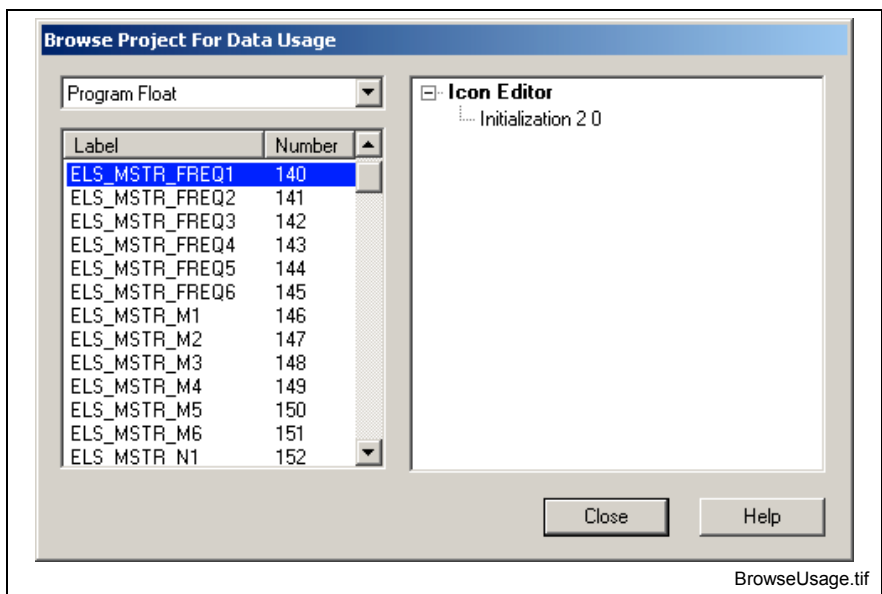


Fig. 2-19: Browse Project for Data Usage

When a data type label is selected from the drop-down list, all assigned labels and numbers used in the program are displayed just below the drop-down list.

Note: Only data types configured and used in the current program will be available in the drop-down list. Data types configured in the VM Data Table but not used in the current program will not be displayed.

When a label is selected from the left-hand list, the editor (Icon, Ladder or Fieldbus) in which the data type is used is displayed in bold text in the right-hand tree structure. Below the editor name, the location or instance of usage, whether in a task or subroutine, is displayed in a tree structure.

Note: Every instance of data usage is displayed in the right-hand section of the window. If a data type is used in more than one area of the current program, duplicate editor names will appear indicating where the data type is used.

When the actual location is selected from below the editor name, the Project Navigator and icon is highlighted. If the data type is used within the Ladder editor or Fieldbus editor, the tool opens displaying the location where used.

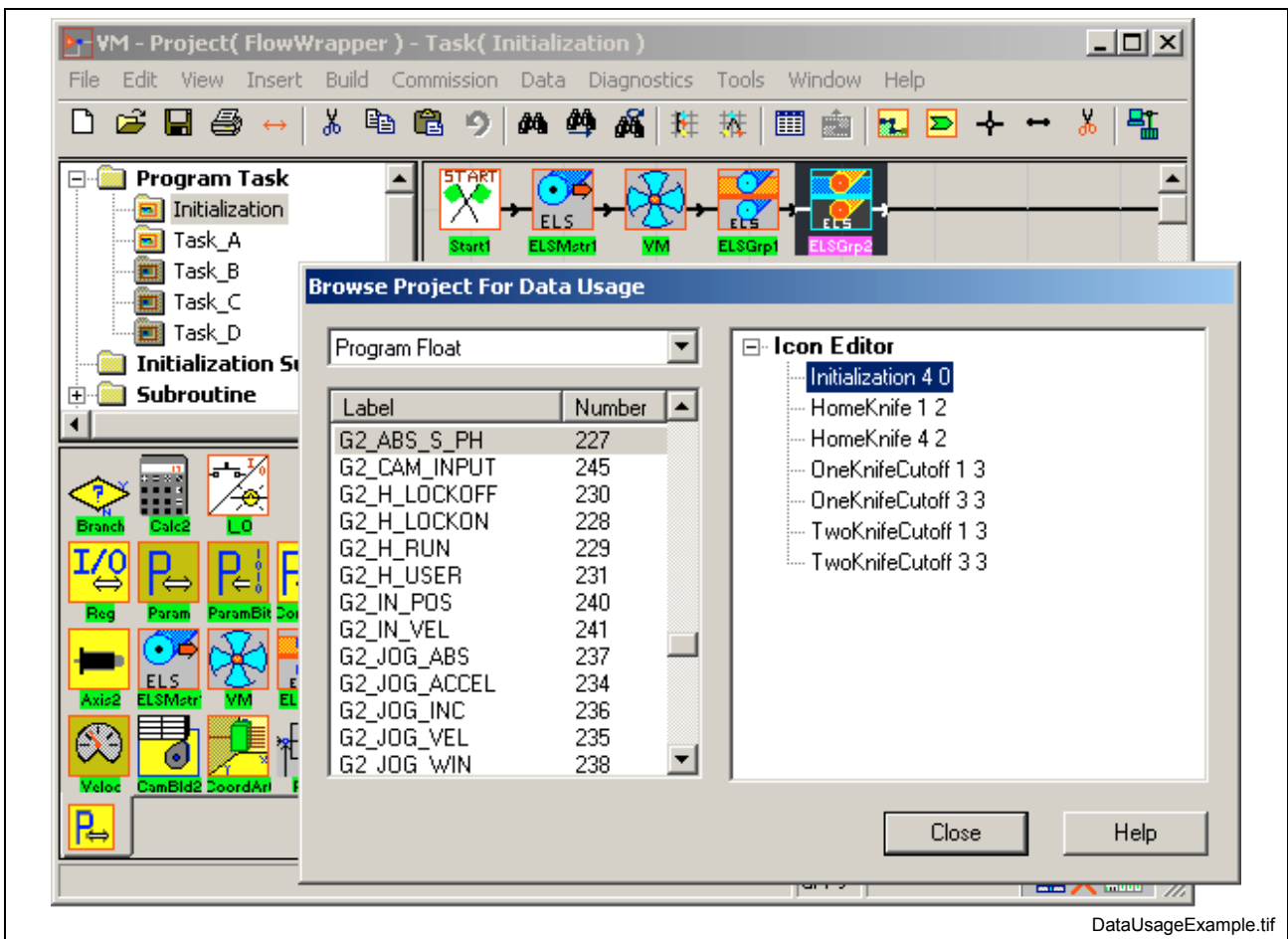



Fig. 2-20: Data Usage Example

Browse Program for Icon Flow

The Icon Flow menu selection allows the user to browse the current project for the use of subroutines and event functions. Subroutines and event functions can be located within any program task, subroutine or event function icon flow. This feature can also be used for:

- visualizing subroutine nesting (up to a maximum of ten) in a tree structure
- locating multiple references of subroutines and/or event functions, and
- overall relationship of subroutines and/or event functions

Selecting **Browse** ⇒ **Icon Flow...** or clicking  opens the *Browse Program for Icon Flow* window in Fig. 2-21. All program tasks, subroutines and event functions are scanned for the use of subroutines and event functions. The available selections in the drop-down list are listed in the table below.

Program Task	Subroutine
Initialization Subroutine	Event Functions

Table 2-3: Icon Flow Type Drop-Down List Selections

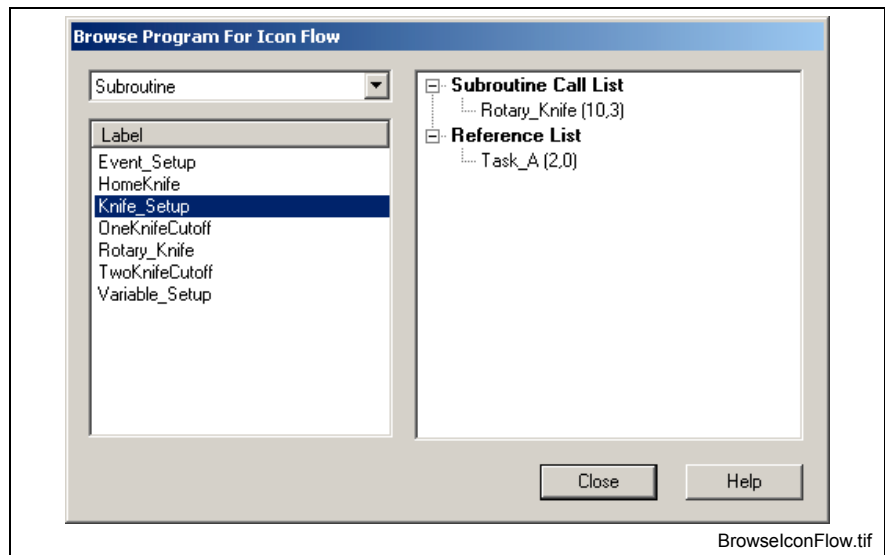


Fig. 2-21: Browse Project for Subroutines and Event Functions

When an item is selected from the drop-down list (i.e., Subroutine), all labels of the selected types are displayed just below the drop-down list.

For example

If a user selects “Subroutine” from the drop-down list, the labels of all subroutines existing in the project are displayed. When a subroutine label is selected, the following tree items are possible:

- *Subroutine Call List* – This tree item is displayed if the selected subroutine or event function label contains subroutines and/or event function as part of its icon flow. (What is called from current icon flow).
- *Reference List* – This tree item is displayed if the selected subroutine or event function label is reference or called by another icon flow.

The sub-items that appear below each main List item are the actual locations in the project where the subroutines or event functions are located.

When a sub-item is selected, the Project Navigator and icon is highlighted.

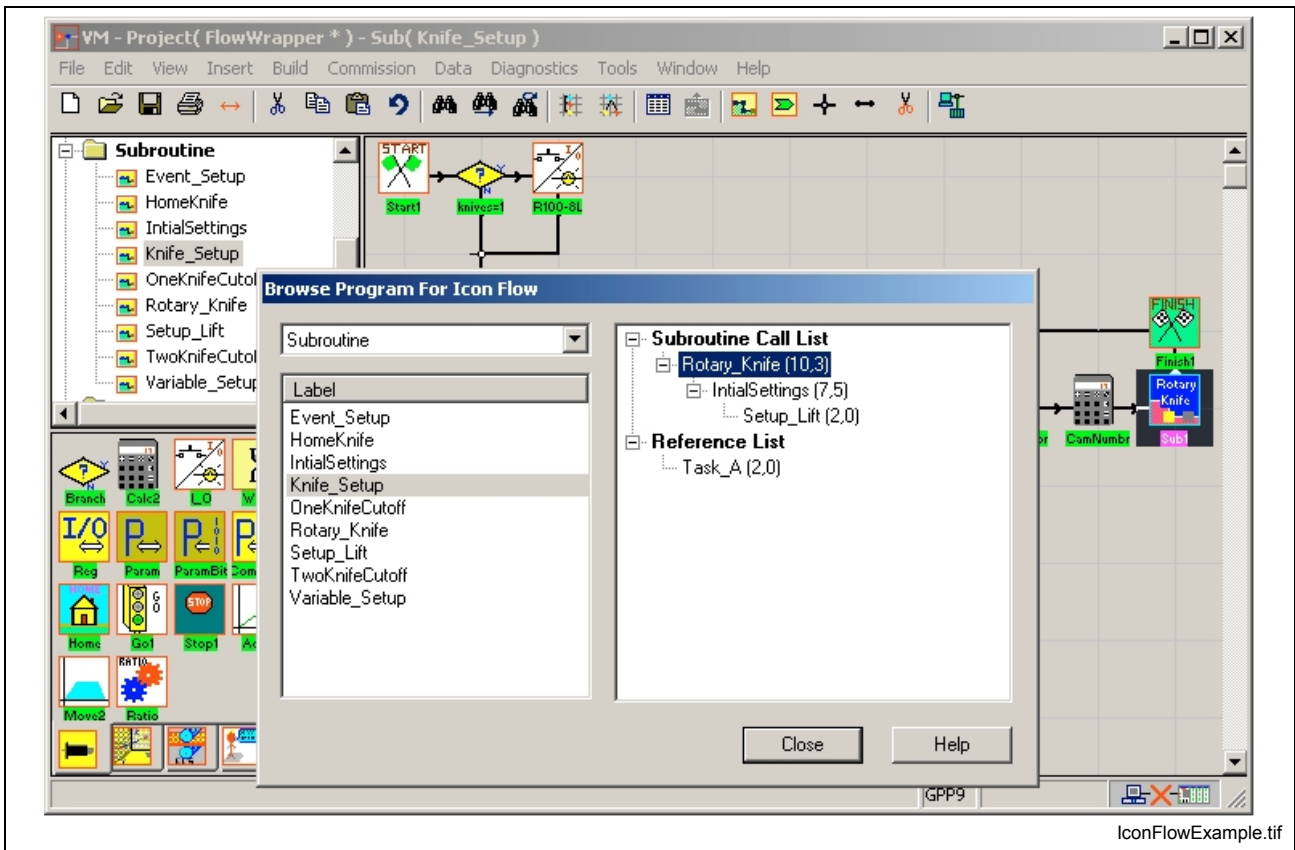


Fig. 2-22: Icon Flow Example

The user can navigate to actual subroutines and event function icon programs by selecting the label of the desired subroutine or event function. VisualMotion Toolkit's title bar indicates if the selected item is either a subroutine or an event function.



Fig. 2-23: Title Bar Descriptions

Subroutine Nesting Example

When a project contains nested subroutines, selecting the subroutine label will display the nested subroutines in an easy to follow tree structure.

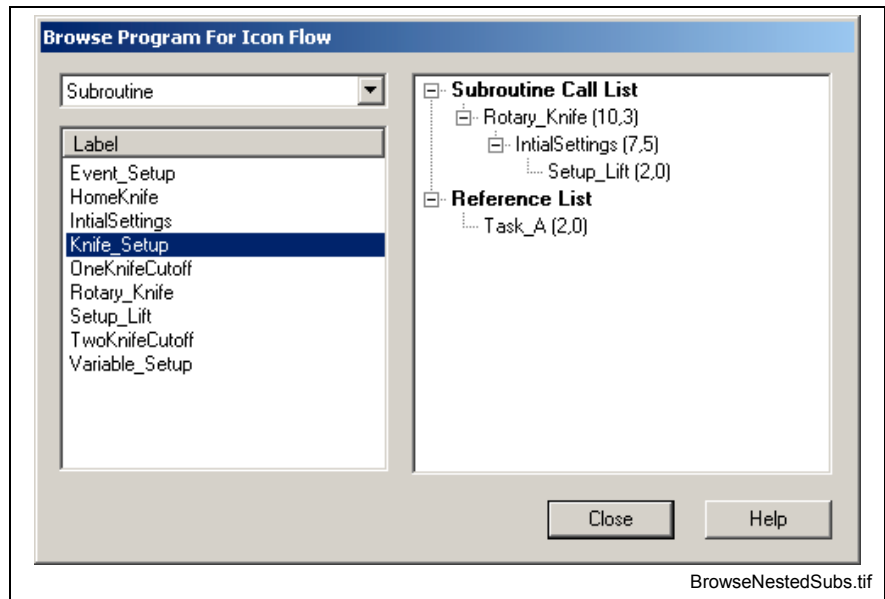


Fig. 2-24: Nested Subroutine Example

Referenced Subroutine Example

The user can expand the Reference List to locate where in the project a subroutine and/or event function is being called).

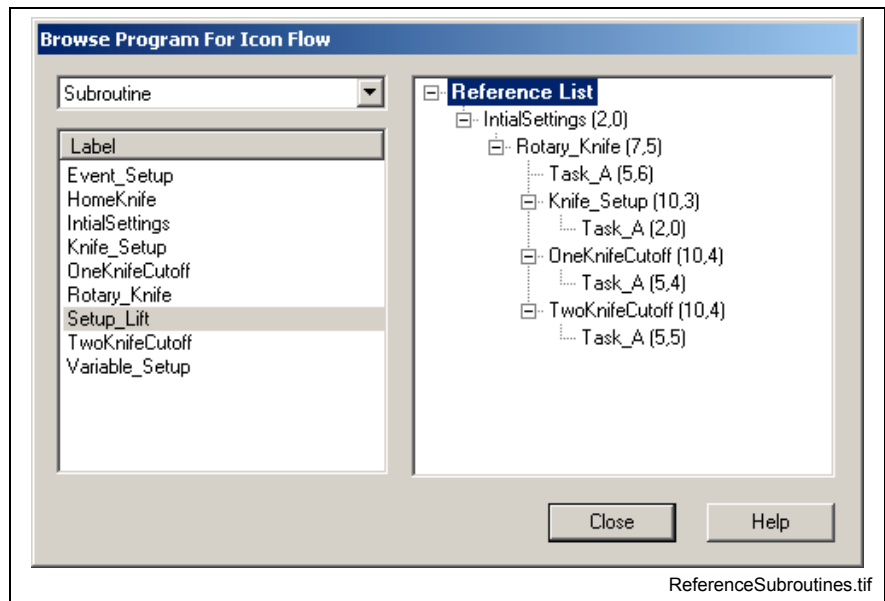


Fig. 2-25: Referencing Subroutine Example

Find, Find Next, Replace

These menu items are used to locate variables or subroutine/event functions in the open program.

Find...

Selecting *Edit* ⇒ *Find...* searches for the first occurrence of specified text and opens the following window:

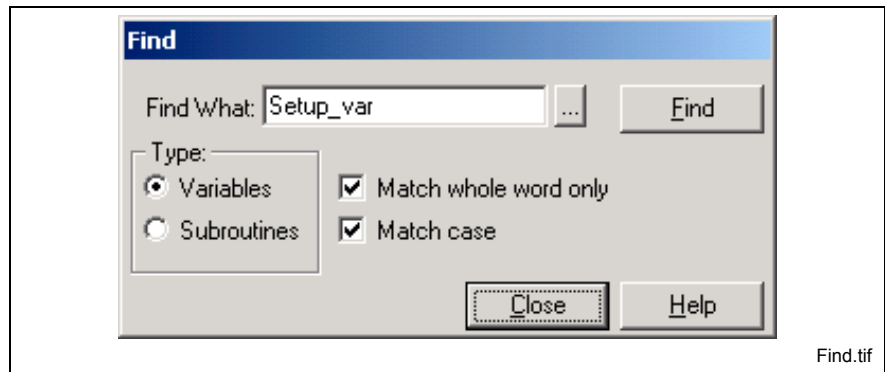



Fig. 2-26: "Find" Window

1. Select **Variables** or **Subroutines** in the **Type:** grouping.
2. Enter the text in the **Find What:** field. The browse button  provides a selection list of all variables or subroutines found in the open program.
3. Check **Match whole word only** or **Match case** to limit the search.
4. Click **Find** to locate the first occurrence of the text.

Note: Only variables and subroutines that are used in the program are found. The VM Data Table may contain variables that are not used in the program.

Find Next

Selecting **Edit** ⇒ **Find Next** (or the **F3** key) searches for the next occurrence of the same text designated for the last **Find** operation.

Replace...

Selecting **Edit** ⇒ **Replace** allows for searching and replacing specified text and opens the following window:

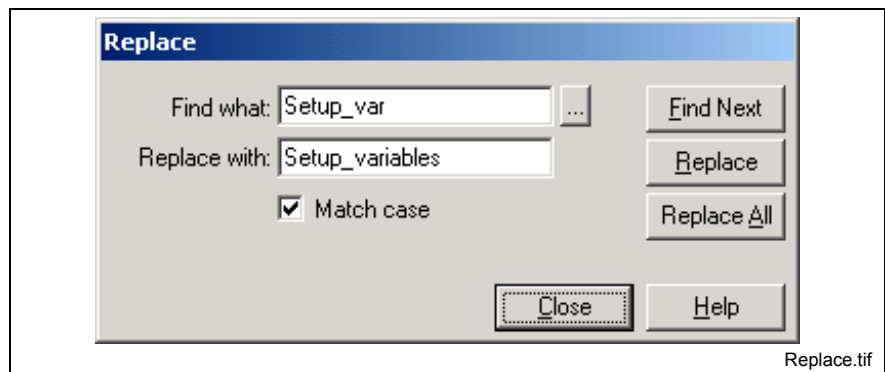



Fig. 2-27: "Replace" Window

VM Data

The VM Data Table is used to Add, Delete, Edit, Find, Browse, Shift, Import and Export the following VisualMotion data types.

- Program Integer (Ix) and Floats (Fx)
- Global Integer (GIx) and Floats (GFX)
- Constants
- Local Variables
- Function Arguments
- Events
- Registers
- Bits (register)
- I/O Bits
- Points (ABSolute and RELative)
- Zones

Note: The VM Data Table can also be opened by clicking on the VM Data icon (). The VM Data Table can be resized by clicking and dragging any corner.

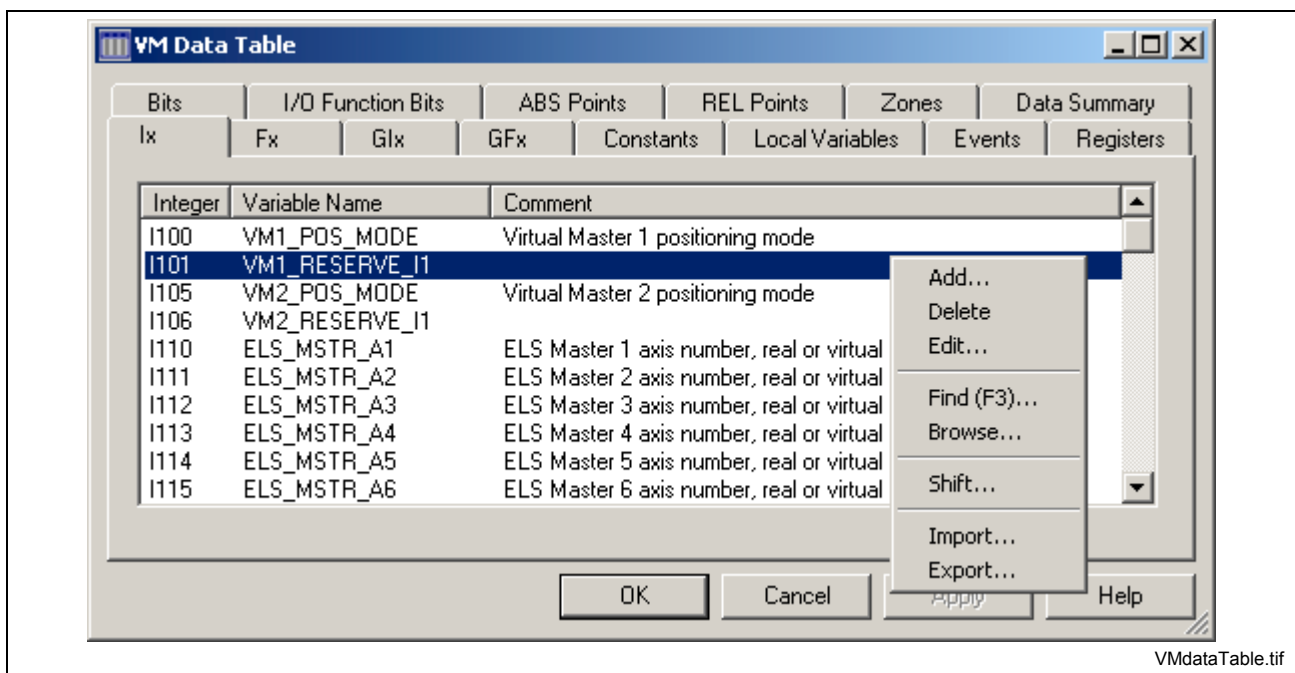


Fig. 2-28: VisualMotion Data Table

Add and Modify VM Data

Right clicking in the display area of any VM Data type, opens the pop-up window in Fig. 2-29. From this window, the functions displayed can be performed under any tab, except for the **Local Variables** and **Data Summary** tabs.

Multiple Item Selection

Items displayed in the VM Data Table can be selected using the following standard Windows™ keyboard and mouse selection methods.

- *Ctrl + A* - selects all the items in the current window.
- *Ctrl + Left Mouse button* – selects items one at a time while holding down the Ctrl key.
- *Shift + Left Mouse button* – selects a group of items while holding down the Shift key.. Using the left mouse button; select the first items, scroll down and then select the last item.

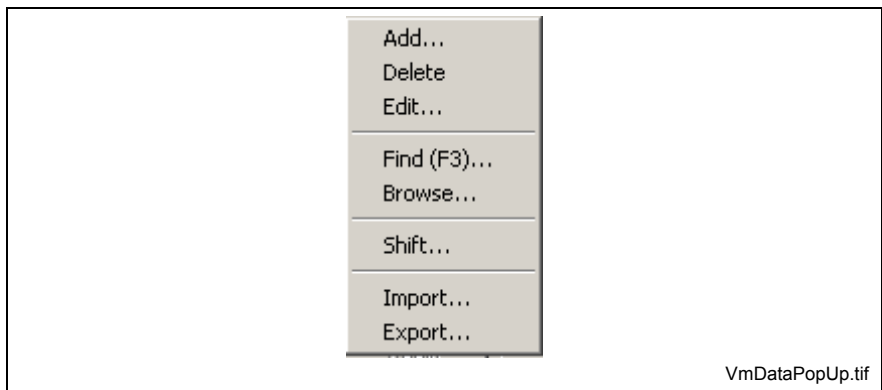


Fig. 2-29: VM Data Pop-up Window

Add, Delete and Edit

These selections allow the user to **Add** a new item in the VM Data Table, **Delete** or **Edit** an existing item.

Find (F3)

Selecting **Find** or pressing the **F3** key opens the *Find* window. From the open window, any items can be found in the current VM Data window by entering a matching or partially matching **Number**, **Variable Name** or **Comment**.

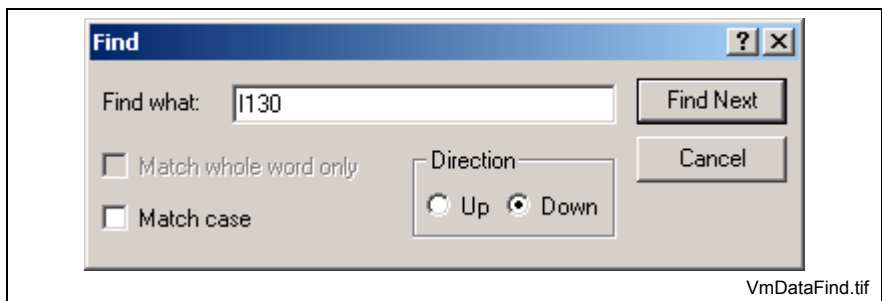


Fig. 2-30: VM Data Find

Browse

Selecting Browse opens the Browse Project for Data Usage window. Refer to Browse on page 2-20 for details.

Shift

The *Shift Indexes* window is used to shift a single item or a group of items **Up (-)** or **Down (+)** by the amount entered in the **Increment** field. To shift an item(s), select the item(s), enter the number of increments to shift and press the **OK** button. The **Up** shift selection, subtracts the **Increment** value from the current index number. The **Down** shift selection, adds the **Increment** value to the current index number.

Note: If the new location is already occupied by an existing item, VisualMotion issues an error and shift request is canceled.

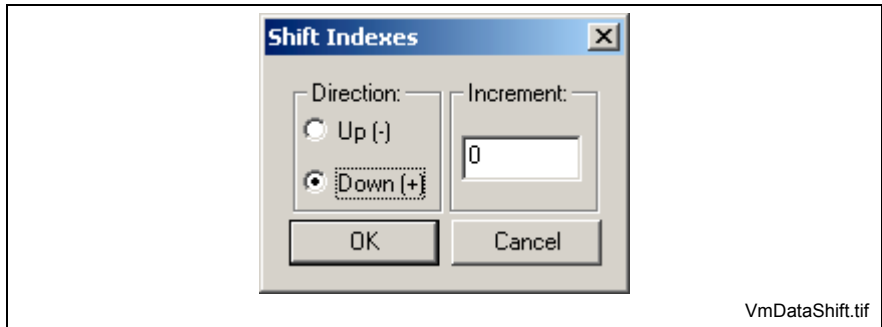


Fig. 2-31: VM Data Shift

Import and Export

This feature is used to import or export the selected items from the VM Data table to a file with a label (*.lbl) extension.

Sorting Numbers, Names or Comments

Any VM Data Table type *Number*, *Name* or *Comment* can be sorted in either ascending or descending order by clicking on the column header.

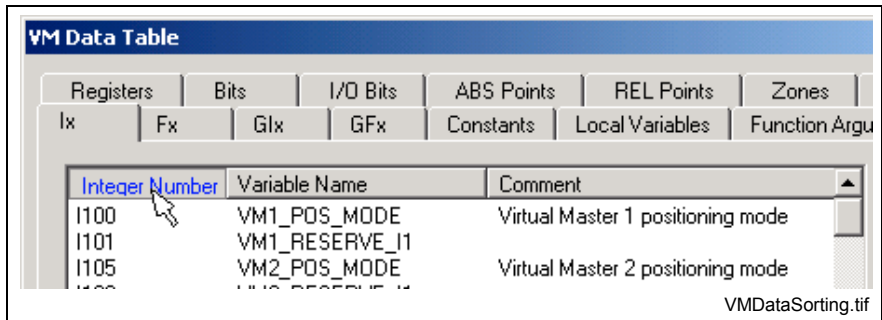


Fig. 2-32: Ascending or Descending Listings

Using VisualMotion Keywords as Names

VisualMotion has a number of keywords, which are used for command instructions. These keywords cannot be used as names (labels). If a keyword is used as a name, VisualMotion will issue the error "Name is a Keyword!" when the user tries to save the name. Following is a list of the VisualMotion keywords:

Current VisualMotion Keywords				
AA_XFER	CALL	EVENT	MOVER_PATH	SET_PARAM
ABORT	CALL_FUNC	EVENT_DONE	MSG_DIAG	SET
ABORT_PATH	CAP_ENABLE	EVENT_ENABLE	MSG_STATUS	START
ABS_INIT	CAP_SETUP	EVENT_END	PARAM_BIT	STOP
ACCEL	CLEAR	EVENT_START	PARAM_INIT	STOP_PATH
AXES	CNC_CIRCLE	EVENT_WAIT	PATH_CALC	Task_A
AXES_GROUP	COMP	EVT_INIT	PATH_HOLD	Task_B
AXES_GROUP1	CONVEYOR_INIT	FUNC_ARG	PID_CONFIG	Task_C
AXIS_ATPOSITION	CONVEYOR_LIST	FUNC_END	PID_CONFIG1	Task_D
AXIS_EVENT	CONVEYOR_PICK	FUNC_START	PLS_WRITE	TEST
AXIS_WAIT	DATA_SIZE	GET_PARAM	PLS_INIT	V_MASTER
BNE	DEC	GO	PLS1_INIT	VAR_INIT
BEQ	DECEL	HOME	POSITION	VEL
BGT	DRVCOM	INC	RA_XFER	WAIT
BLT	DRVCOM_STATUS	INIT	RATIO	WAIT_IO
BGE	EE_XFER	INITIALIZATION	READ	WAIT_PATH
BLE	ELS_ADJUST	KINEMATIC	REGISTRATION	WRITE
BRA	ELS_ADJUST1	KINEMATIC1	REL_INIT	ZONE_INIT
BROADCAST	ELS_GROUPM	LOCAL_VAR	RESUME_PATH	ZZ_XFER
CALC	ELS_GROUPS	MESSAGE	RETURN	
CAM_ADJUST	ELS_GROUPS1	MODE	ROBOT_ORIGIN	
CAM_ACTIVATE	ELS_INIT	MOVE_JOINT	ROBOT_TOOL	
CAM_BUILD	ELS_MODE	MOVEA_AXIS	ROTARY_EVENT	
CAM_BUILD1	ELS_MASTER	MOVEA_CIRCLE	RR_XFER	
CAM_ENGAGE	ELS_MASTER1	MOVEA_PATH	SEQ_LIST	
CAM_INDEX	ELS_STOP	MOVER_AXIS	SEQ_STEP	
CAM_STATUS	END	MOVER_CIRCLE	SEQUENCER	

Table 2-4: Current VisualMotion Keywords

Integers (Ix) and Floats (Fx)

Program integers and floats are displayed listing the *Number*, *Variable Name* and *Comment* of all defined integers and floats in the current program. Integers and floats are variables that can be used within any task and are associated with the project for which they were created.

Note: The value of all integers and floats is saved with the project even during a power lost condition.

When creating a new project, VisualMotion automatically allocates memory for a minimum of 50 integers and 50 floats. These integers and floats are not displayed in the VM Data Table until a number and name (label) has been assigned.

System default integers and floats, such as those used for ELS, will appear in the VM Data Table if assigned in their corresponding icons.

Add an Integer or Float

Right click and select **Add...** to open the *Add Integer(Float)* window. Enter a number and name for the integer or float. The **Comment** field is optional. Since VisualMotion allocates memory for the first 50 integers and floats, any number entered that is greater than 50 will become the top-level number with additional memory allocated.

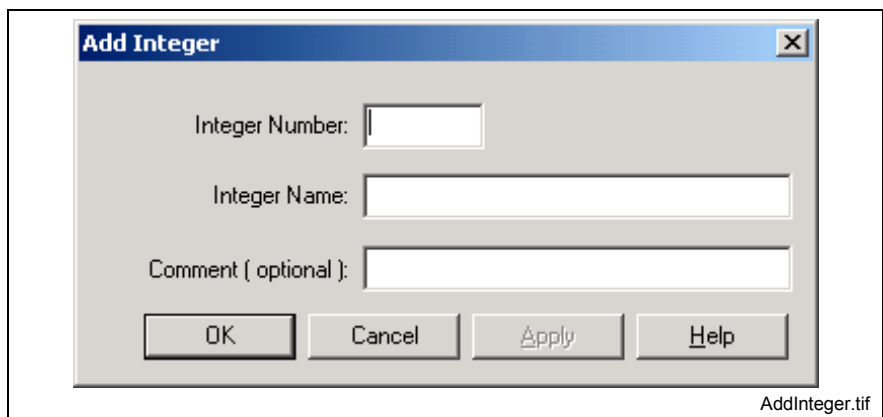


Fig. 2-33: Add Integer or Float

For example:

If the number 200 is entered, VisualMotion will allocate memory for 200 integers. Even if only one integer or float is displayed in the VM Data Table. Each integer and float is allocated 4 bytes of memory. The total memory allocated for integers and floats can be viewed by selecting the *Data Summary* tab.

Note: Edit or Delete an integer or float by first select the item.

Since the VM Data Table only displays the number and name (label) of an integer or float, the actual value of each integer or float in a project can be viewed and defined by selecting **Data** ⇒ **Variables...** (In online mode) and selecting the appropriate *Type*.

Global Integers (Glx) and Floats (GFx)

Global integers and floats are displayed listing the *Number*, *Variable Name* and *Comment* of all defined global integers and floats. Global integers and floats are variables that can be used in any task across different programs resident in the control. The names (labels) assigned

to global variables are saved with each program while their values are saved to the control's memory.

Note: The value of global variables is not saved if the control loses power unless the values are flashed using the *Save Global Variables Command (C-0-0082)*. Refer to Saving Global Variables to Flash on page 2-107 for details.

System default global integers (Glx) for the I/O Mapper are assigned, with fixed memory, for all programs. These global integers are reserved for the I/O Mapper's counter, timer and shift register output functions.

The maximum number of global variables in a program is fixed and defined by the following control parameters:

- **C-0-0080**, Maximum number of global integers
- **C-0-0081**, Maximum number of global floats

The default numbers are 512 for global integers and 256 for global floats.

Add a Global Variable

Right click and select **Add...** to open the *Add Global Integer(Float)* window in offline mode. Enter a number and name for the global variable. The **Comment** field is optional.

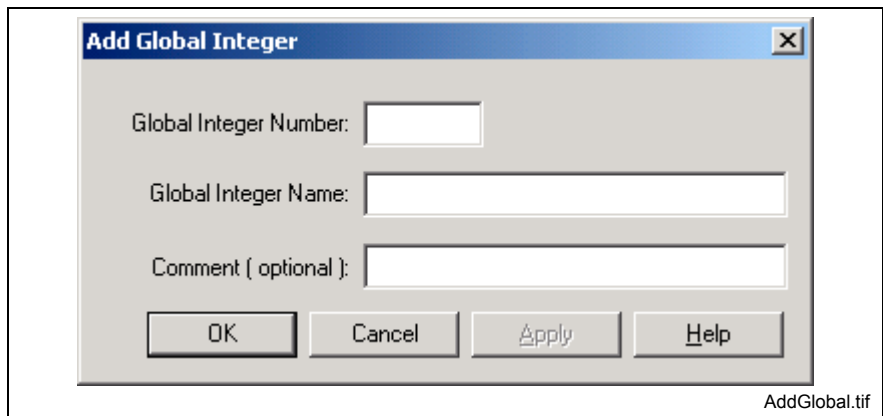


Fig. 2-34: Add Global Variable (Glx or GFx)

Since the VM Data Table only displays the number and name (label) for global variables, the actual value for each global variable in a program can be viewed and defined by selecting **Data** ⇒ **Variables...** (In online mode) and selecting the appropriate *Type*.

Constants

Constants are displayed listing the **Constant Value**, **Constant Name** and **Comment** of all defined constants in a program. Constant names are created to represent a fixed numeric value in an icon program.

The user can use a name assigned to a value instead of the value itself. The benefit of this would be; for example, if the same numeric value is used within multiple icons, the user need only modify the value in the VM Data Table instead of a value in each icon.

Add a Constant

Right click and select **Add...** to open the *Add Constant* window in offline mode. Enter a numeric value and name for the constant. The **Comment** field is optional.

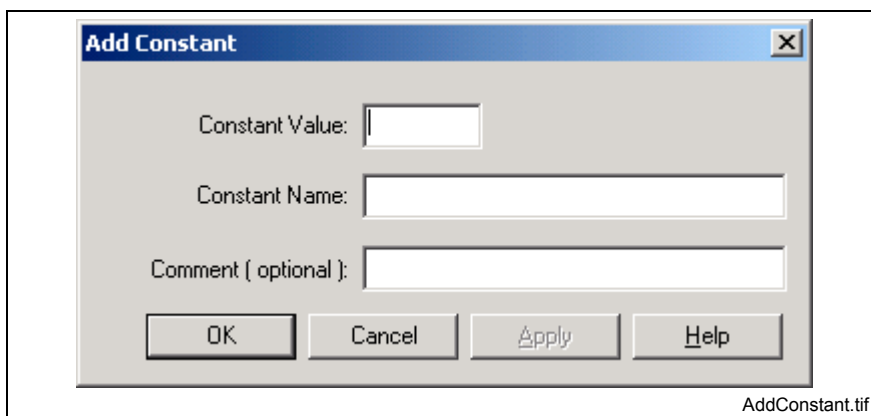


Fig. 2-35: Add a Constant

Local Variables

Only local variables that are used in the current task, subroutine or event are displayed in the VM Data Table. Local variables **cannot** be added in the VM Data Table. Local variables are added in the *Start* icon.

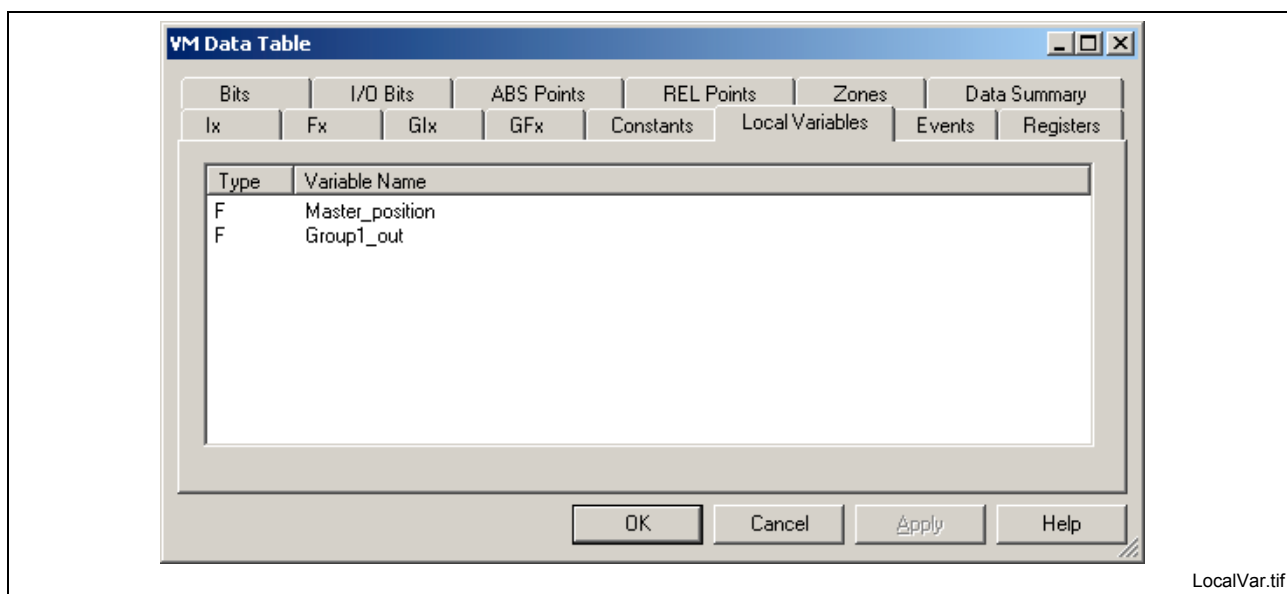


Fig. 2-36: Local Variables

Since the VM Data Table only displays the number and name (label) for local variables, the actual value for each local variable in a task, subroutine or event can be viewed and defined by selecting **Data ⇒ Variables...** (In online mode) and selecting the *Type: (Task A Local Variables)*.

Function Arguments

Only function arguments that are used in the current subroutine are displayed in the VM Data Table. Function Arguments **cannot** be added in the VM Data Table. Function arguments are added in the subroutines' *Start* icon.

Note: Function arguments are only displayed if they exist in the currently selected subroutine.

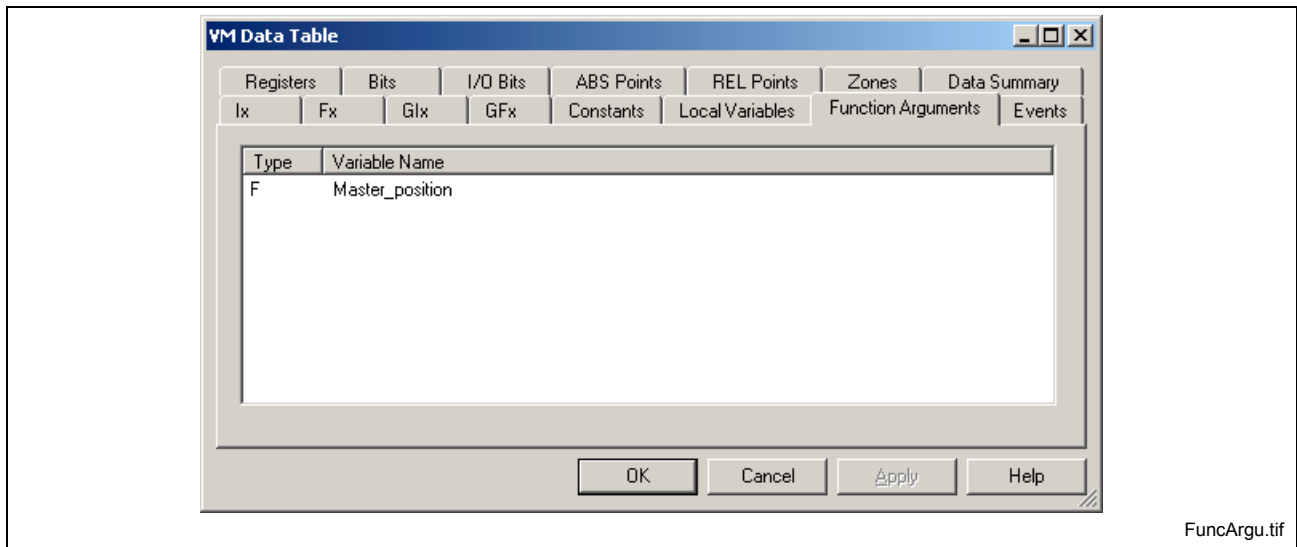


Fig. 2-37: Function Argument

Since the VM Data Table only displays the number and name (label) for function arguments, the actual value for each function argument in a program can be viewed and defined in the *Start* icon of the subroutine where it resides.

Events

Events are displayed listing the *Number*, *Name*, *Argument*, *Type* and *Function* of all events in the current project.

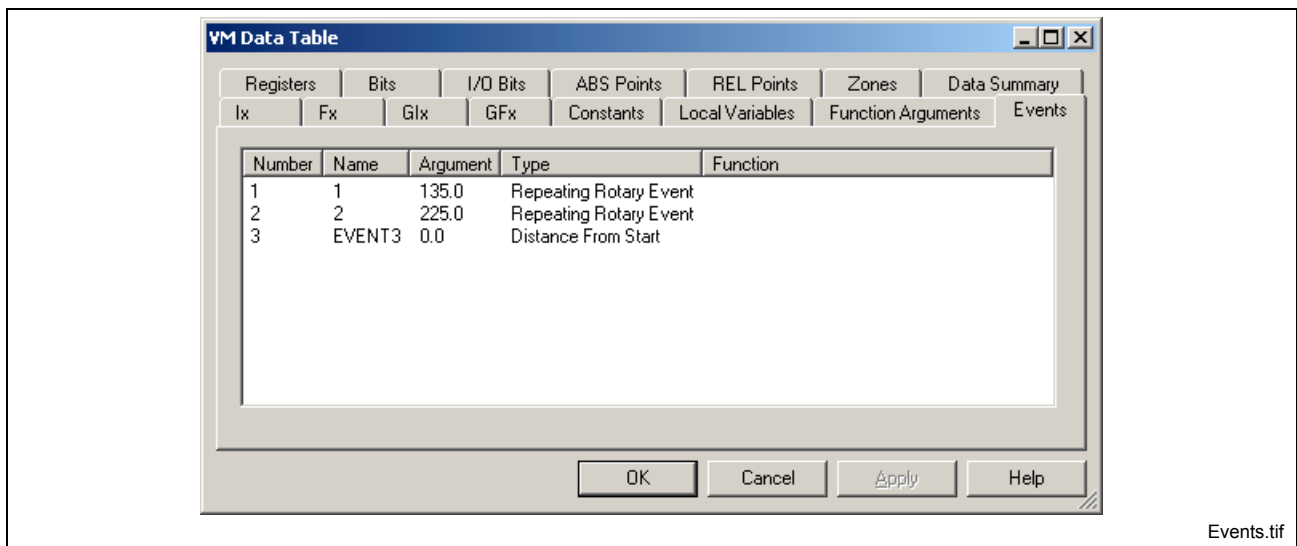


Fig. 2-38: Events

Add an Event

Right click and select **Add...** to open the *Add Event* window in offline mode. From this window, the user can create events, select the event type, initialize an argument value and assign an event function. For detailed examples on creating events and event functions, refer to the *VisualMotion Programming* chapter in the *VisualMotion 9 Application Manual*.

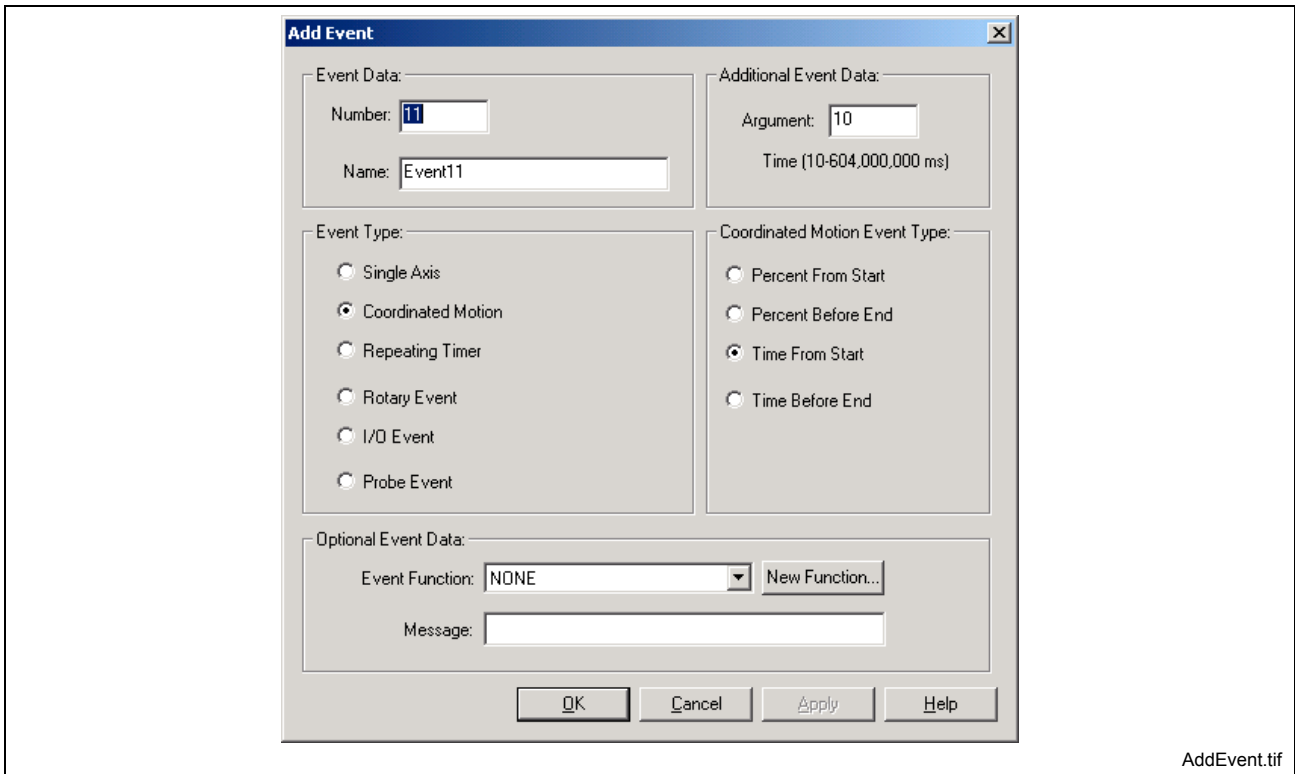


Fig. 2-39: Add an Event

Registers

Registers are displayed listing the **Register Number**, **Register Name** and **Comment** field for all registers that have been assigned a name other than default name (Reg_070). The first 100 registers are reserved for control and system functions, while others are recommended as defaults for applications such as ELS.

Add a Register

Right click and select **Add...** to open the *Add Register* window in offline mode. Enter a number and name for the register. The **Comment** field is optional. A VisualMotion system can contain up to a maximum of 1024 registers.

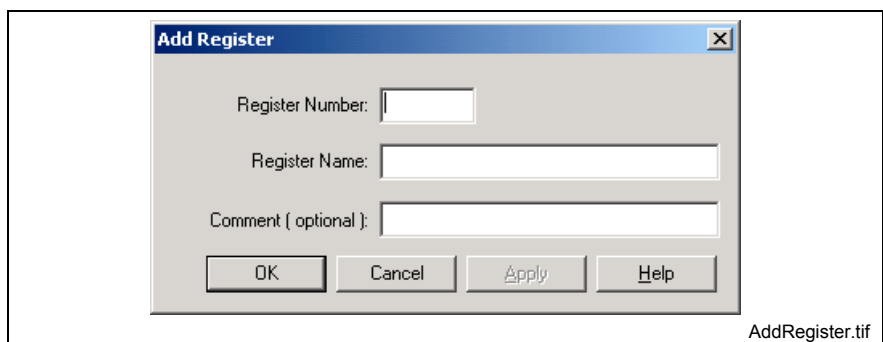


Fig. 2-40: Add a Register

Since the VM Data Table only displays the number and name (label) for assigned registers, the actual value of each register in a program can be viewed and defined by selecting **Data ⇒ Registers** (In online mode).

Bits

The Bits window displays the **Register-Bit Number**, **Bit Name** and **Comment** field for all register bits that have assigned a name other than the default name (Bit_01). Only those bits that have been assigned names will be displayed.

Add a Bit

Right click and select **Add...** to open the *Add Bit* window in offline mode. A bit name is assigned to a register number. Enter a Register Number, Bit Number and Bit Name. The **Comment** field is optional. A VisualMotion register can contain up to 16 bits. Only those bits assigned to a register number and given a name will be displayed in the VM Data Table.

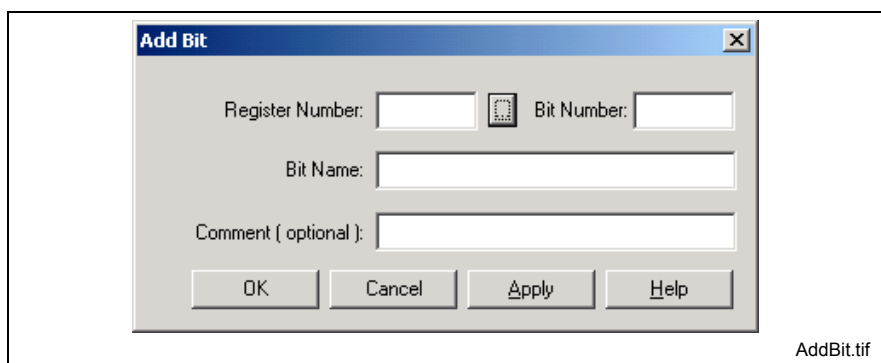


Fig. 2-41: Add a Register Bit

The VM Data Table only displays the number and name (label) for assigned register bits. The actual value of each register bit in a program can be viewed and defined by selecting **Data ⇒ Registers** (In online mode) and double clicking a register number to open the bits window.

I/O Bits

The I/O Bits window displays the **Global Integer-Bit Number** and **Bit Name** for the I/O Bit functions reserved for the I/O Mapper by VisualMotion.

The user can only edit the name or add an optional **Comment** to the existing entries.

Add an I/O Bit

Right click and select **Add...** to open the Add I/O Bit window. From here, the user can select a global integer and bit number.

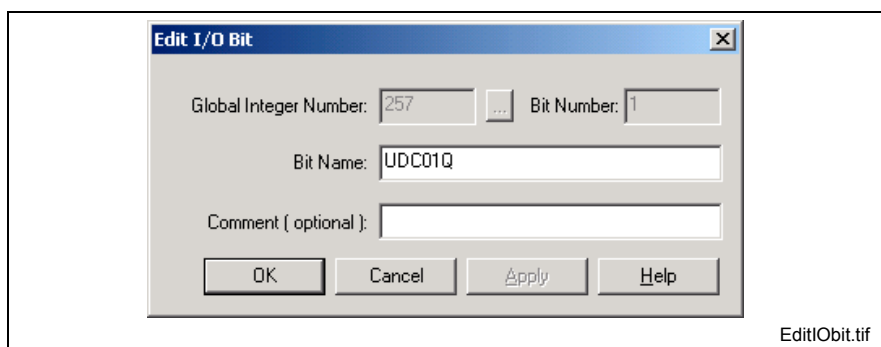


Fig. 2-42: Edit I/O Bit

Points (ABS and REL)

Absolute and Relative points are displayed for all defined points in the current project. Points are not displayed in the VM Data Table until a number and name (label) has been assigned.

Refer to Points on page 2-110 for details

Add an Absolute or Relative Point

Right click and select **Add...** to open the *Add ABS(REL)* window. From this window, the user can assign a number and name, enable and disable events, initialize coordinate and movement values.

Note: VisualMotion allocates memory for points based on the largest number assigned to a point.

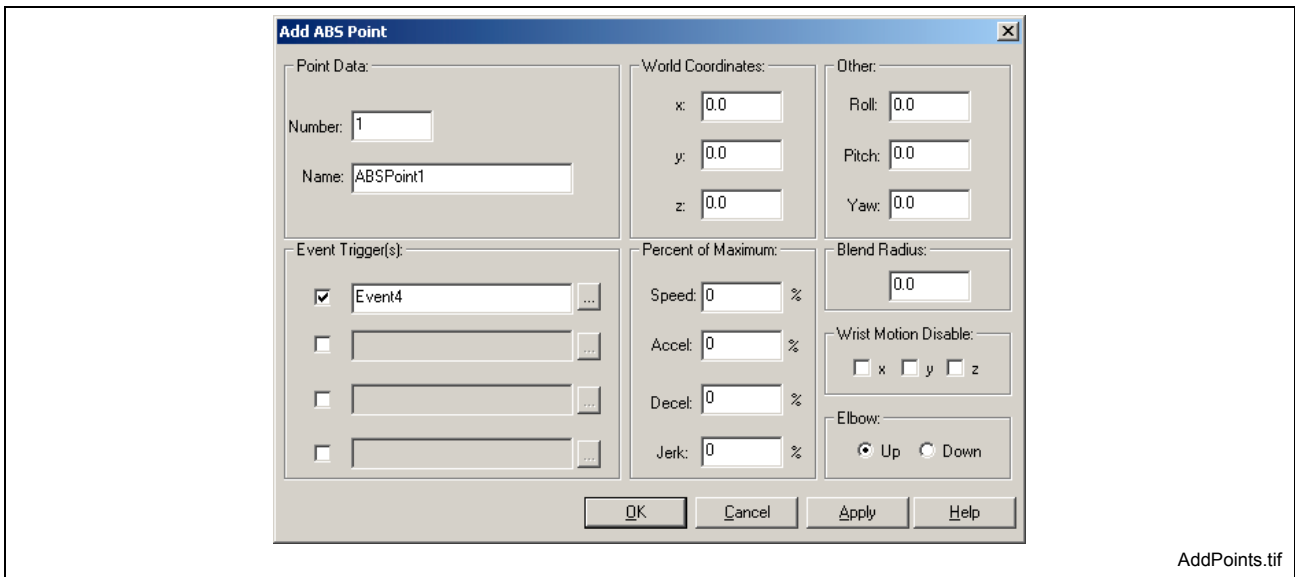


Fig. 2-43: Add ABS(REL) Point

For example:

If the number 200 is entered, VisualMotion will allocate memory for 200 points. Even if only one point is displayed in the VM Data Table. Viewing points by selecting **Data ⇒ Points** will display a list of 200 points. Each ABS or REL point is allocated 44 bytes of memory. The total memory allocated for points can be viewed by selecting the *Data Summary* tab.

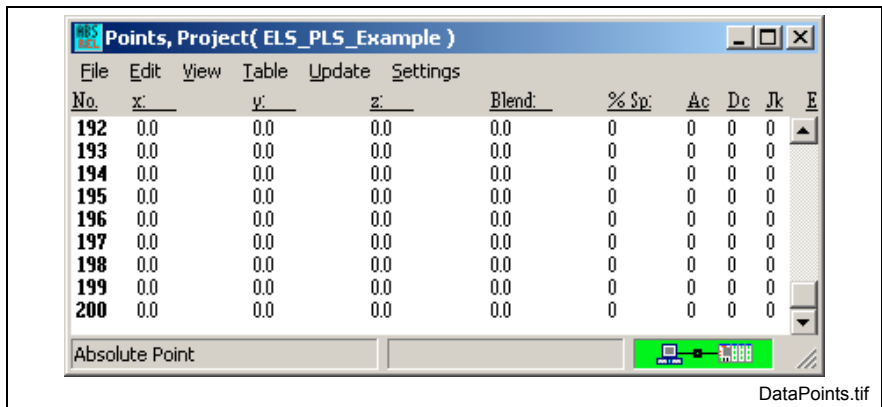


Fig. 2-44: Selecting Data → Points

Zones

Zones are displayed for all defined zones in the current project. Zones are not displayed in the VM Data Table until a number and name (label) has been assigned.

Add a Zone

Right click and select **Add...** to open the *Add Zone* window. From this window, the user can assign a number and name, active zones for specific task and initialize (x, y, z) coordinate values for points 1 and 2.

Note: VisualMotion allocates memory for zones based on the largest number assigned to a zone.

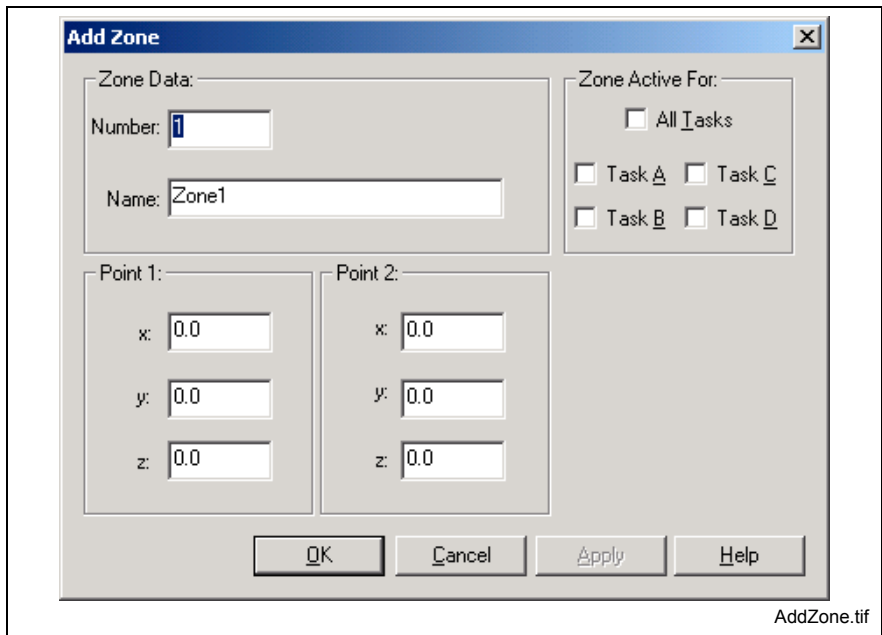


Fig. 2-45: Add a Zone

For example:

If the number 15 is entered, VisualMotion will allocate memory for 15 zones. Even if only zone number 15 is displayed in the VM Data Table. Selecting **Data** ⇒ **Zones** displays a list of 15 zones. Each zone is allocated 28 bytes of memory. The total memory allocated for zones can be viewed by selecting the *Data Summary* tab.

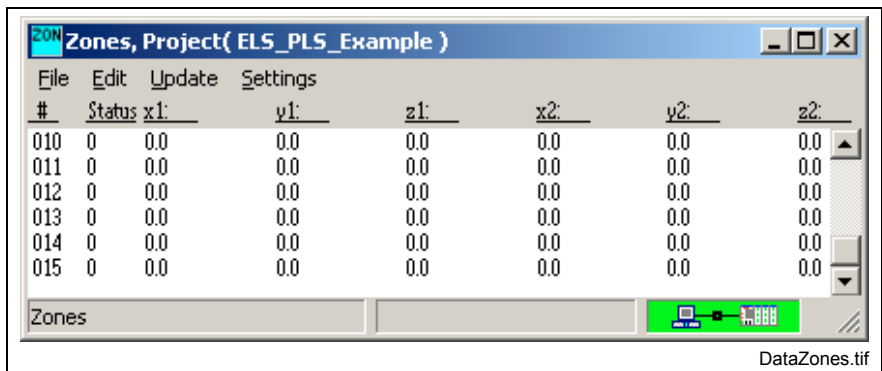


Fig. 2-46: Selecting Data → Zones

Data Summary

The Data Summary tab displays the total memory allocation by VisualMotion for Integers, Floats, Global Integers and Floats, Points and Zones.

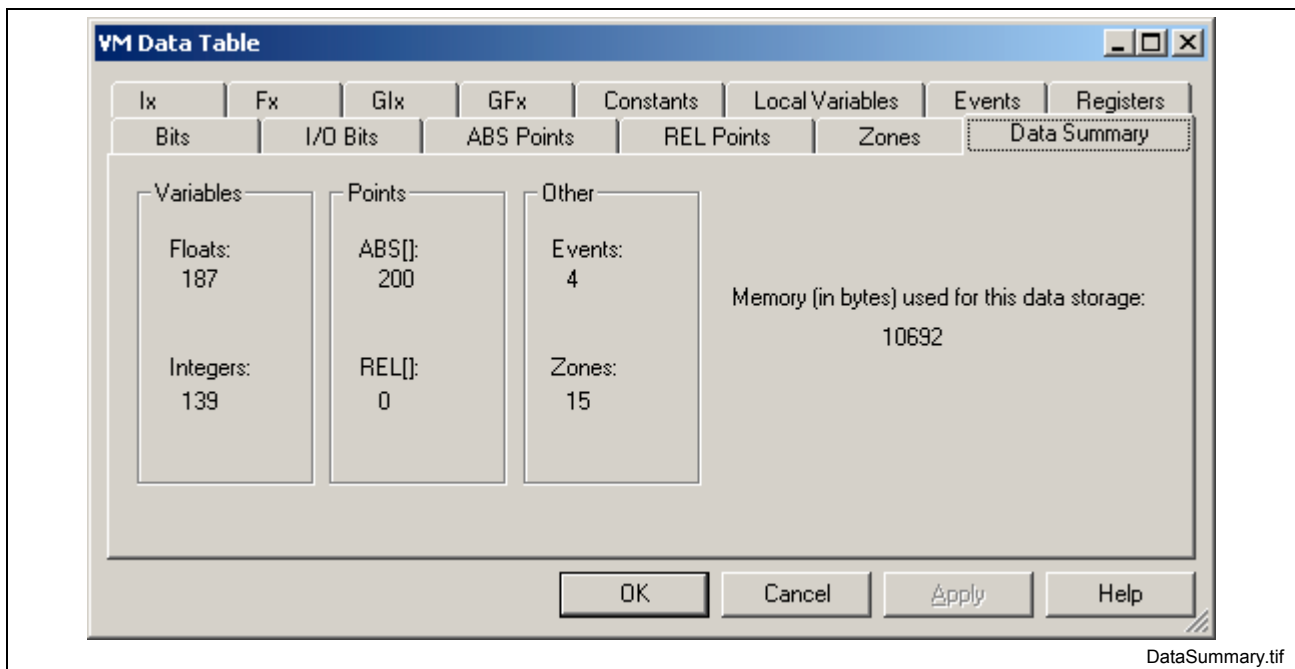


Fig. 2-47: Data Summary

Labels

Note: The Labels menu selection is only available in service mode.

Labels are symbolic names assigned to system resources, such as axes, drives, etc.. Labels may also be used for absolute or relative point names, or in place of "literal" constant or variable values in expressions. For example, once assigned, the label "PI" can be used throughout a program, instead of repeatedly entering the literal value 3.14159.

Labels may use up to twenty ASCII characters and are case-sensitive. Blank spaces are not allowed within a symbol. Use a printable character as a separator if it is required for clarity. For example, "next_move," rather than "next move". The first character of a label must be an alpha character.

The control compiler, used for both icon and text language programming, allows the use of a literal integer value (i.e., a number such as "1" or "5"). Provided it is within the range of integers that are valid for the specified argument. Integers used to specify system devices, such as an axis or drive, must be within the range permitted by the complete VisualMotion system and installed software.

For example: a VisualMotion system with eight digital drives installed can specify an axis or drive using an integer from 1 to 8. The compiler must be able to resolve a symbol used as a table index argument to an integer index within the range, or size, of the table.

VisualMotion has a number of keywords, which it uses for command instructions. These keywords cannot be used as labels. If a keyword is used as a label, VisualMotion will issue the error "Label is a Keyword!" when the user tries to save the label. Refer to Using VisualMotion Keywords as Names on page 2-29 for a listing.

Variable Labels

Selecting **Edit** ⇒ **Edit Labels** ⇒ **Variable Labels** opens the **User Defined Labels** window, used to provide symbolic ASCII names for Float, Integer, Global Float, Global Integer, Absolute or Relative point values.

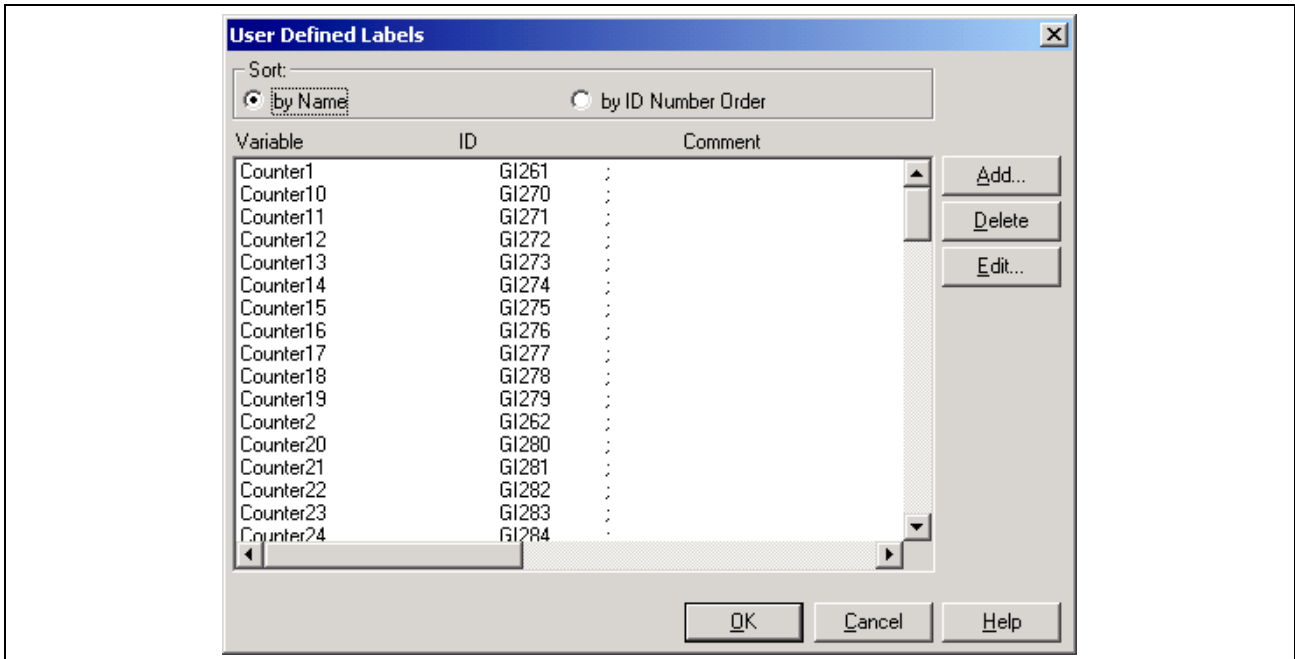


Fig. 2-48: "User Defined Labels" Window

The **User Defined Labels** window allows you to assign an ASCII name to a value or system component, as previously described. To sort the list alphabetically by label, select the radio box **by Name**. To sort the list alphabetically by ID, select the radio box **by ID Number Order**. After labels are defined, instead of explicitly entering a value or redefining a system component, the label can be entered by accessing the **User Defined Labels** window and selecting the appropriate label.

To add a new label:

1. Click **Add...** to open the **Add Variable Label** window below.

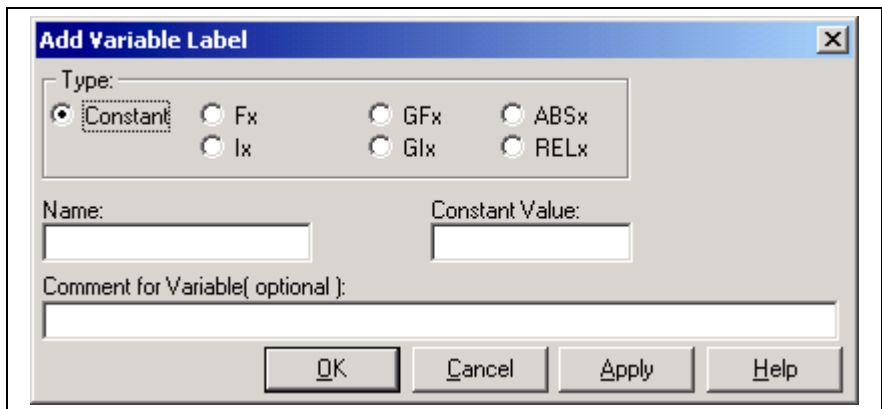


Fig. 2-49: "Add Variable Label" Window

2. Enter the desired **Type**, **Name**, **Constant Value** (if applicable) and **Comment** (up to 80 characters).
3. Click **Apply** or **OK**.

To delete an existing label:

1. Highlight a desired label and click **Delete**.

To edit an existing label:

1. Highlight a desired label and click **Edit...**

The **Edit User Label** window opens just below. The label's **Type**, **Name**, **Global Integer** and **Comment** are filled in as they were originally designated.

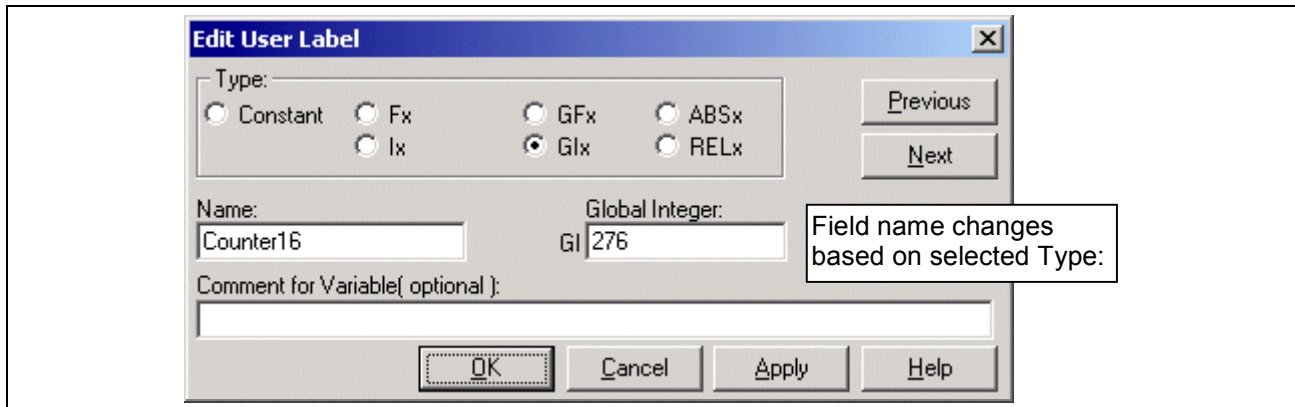


Fig. 2-50: "Edit User Label" Window

2. Enter the desired **Type**, **Name**, **Global Integer** (if applicable) and **Comment** (up to 80 characters).
3. Click **Apply** or **OK**.

Register Labels

Selecting **Edit** ⇒ **Edit Labels** ⇒ **Register Labels** opens the **Register Labels** window, used to provide symbolic ASCII names for the VisualMotion control status and I/O registers.

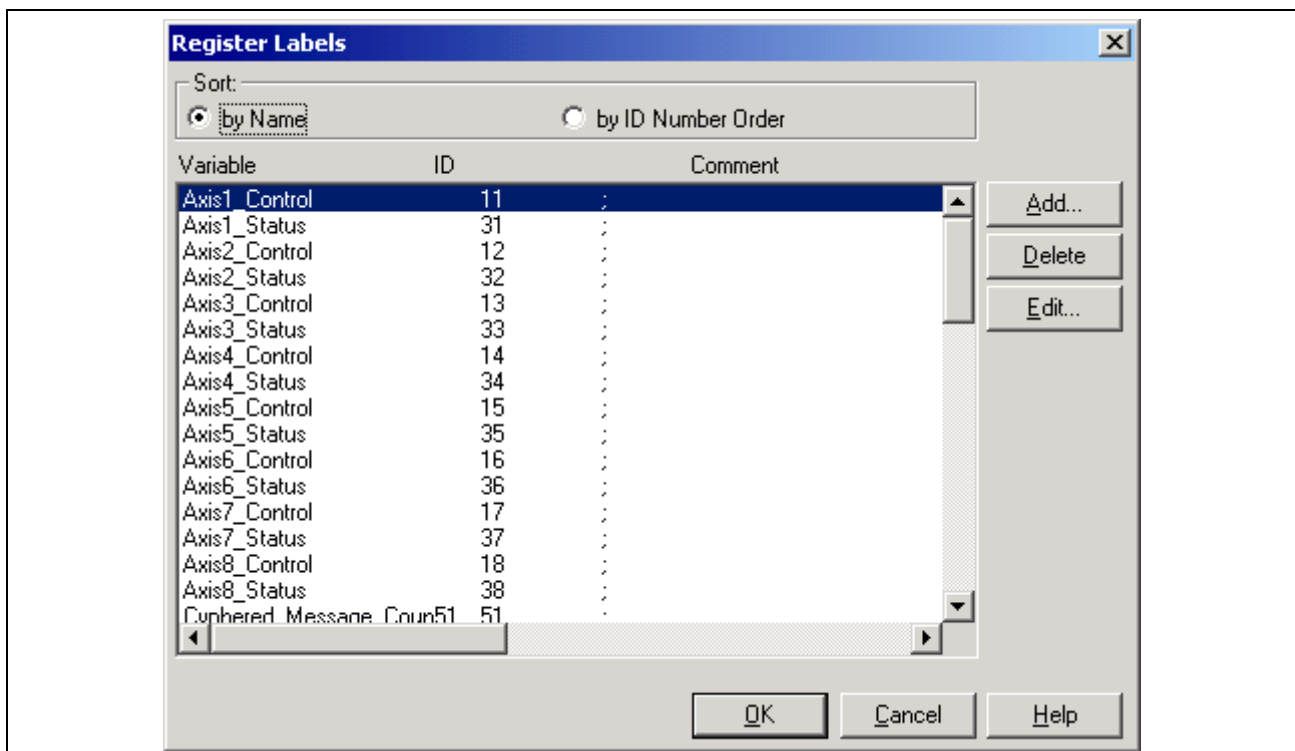


Fig. 2-51: "Register Labels" Window

To sort the list alphabetically by label, select the radio box **by Name**. To sort the list alphabetically by ID, select the radio box **by ID Number Order**. After register labels are assigned and the program is saved, the labels are embedded in the motion program (the .str file) and will not be lost if the program is later transferred to a different VisualMotion system. New programs are loaded with the default register names. Refer to chapter 4, Registers, for default register names.

To add or edit a register label:

1. Click **Add...** or select the desired register and click **Edit...**

An Add/Edit Register Labels window opens. The **Name** must be explicitly entered. (When editing an existing label, the current **Name**, **Register Number** and **Comment** are automatically filled in.)

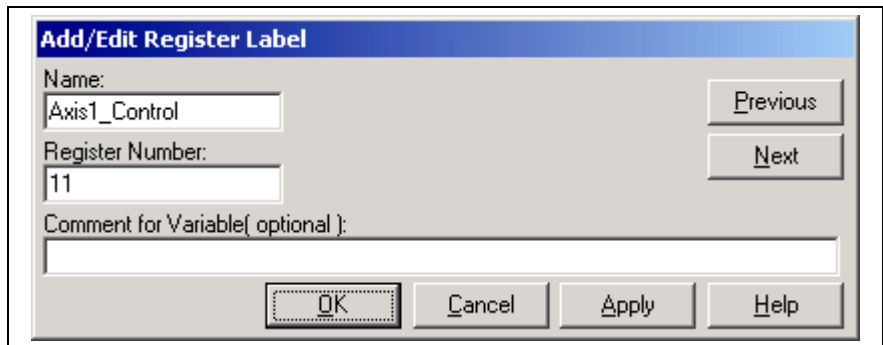


Fig. 2-52: "Add/Edit Register Label" Window

2. Enter the desired **Name**, **Register Number** and **Comment** (up to 80 characters).
3. Click **Apply** or **OK**.
The **Previous** and **Next** buttons scroll through the defined labels.

Bit Labels

Selecting **Edit** ⇒ **Edit Labels** ⇒ **Bit Labels** opens the **Bit Labels** window, used to provide symbolic ASCII names for individual bits within VisualMotion control and I/O registers.

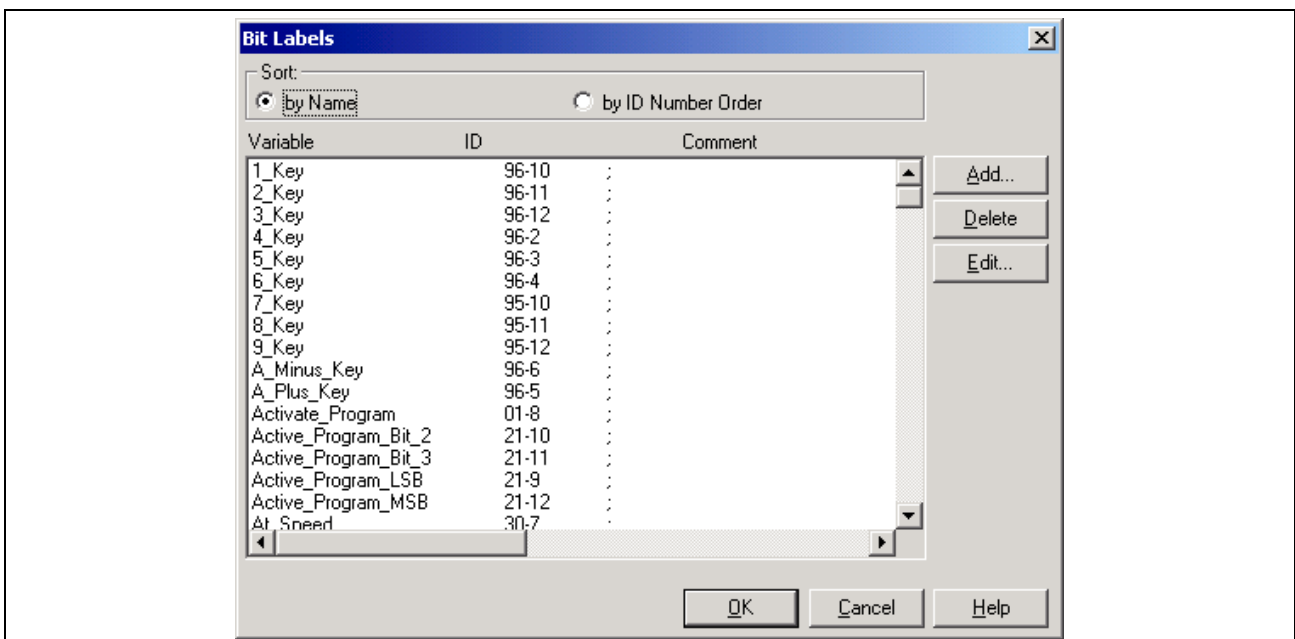


Fig. 2-53: "Bit Labels" Window

To sort the list alphabetically by label, select the radio box **by Name**. To sort the list alphabetically by ID, select the radio box **by ID Number Order**.

After bit labels are assigned and the program is saved, the labels are embedded in the motion program (the .str file) and will not be lost if the program is later transferred to a different VisualMotion system. New programs are loaded with the default bit names.

To add or edit a bit label:

1. Click **Add...** or select the desired bit and click **Edit...**

An **Add/Edit Bit Label** window opens. The **Name** must be explicitly entered. (When editing an existing label, the current **Name**, **Reg-Bit Number** and **Comment** are automatically filled in.)

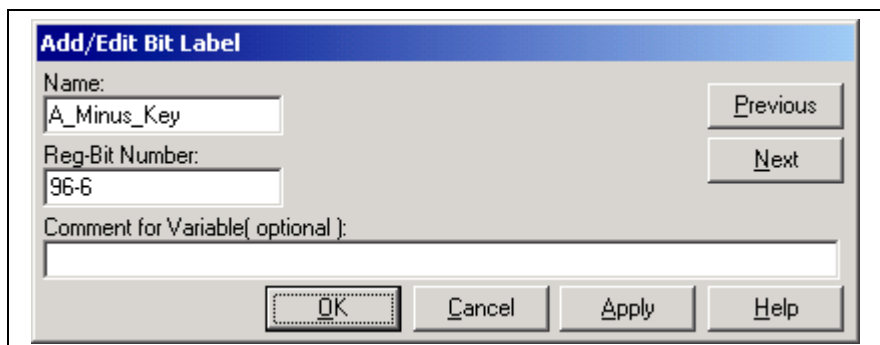


Fig. 2-54: "Add/Edit Bit Label" Window

2. Enter the desired **Name**, **Reg-Bit Number** and **Comment** (up to 80 characters).
3. Click **Apply** or **OK**.
The **Previous** and **Next** buttons scroll through the defined labels.

I/O Bit Function Labels

Selecting **Edit** ⇒ **Edit Labels** ⇒ **I/O Bit Function Labels** opens the **Global Integer Bit Labels** window, used to provide default ASCII names for I/O Mapper functions.

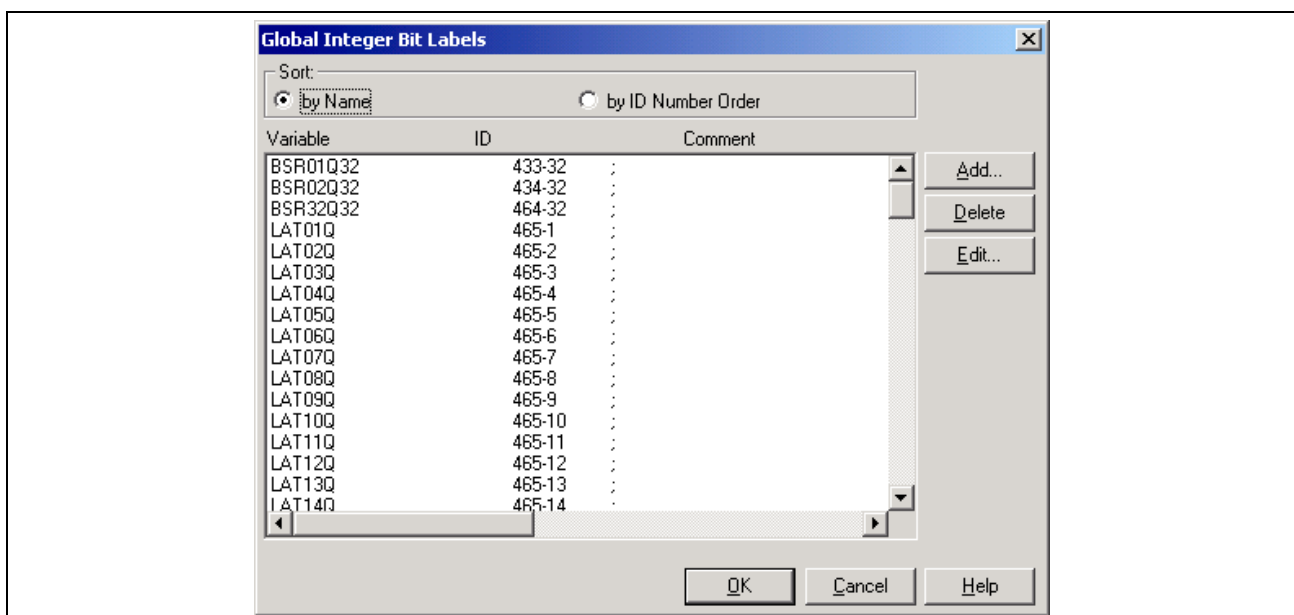


Fig. 2-55: "Global Integer Bit Labels" Window

To sort the list alphabetically by label, select the radio box **by Name**. To sort the list alphabetically by ID, select the radio box **by ID Number Order**.

After bit labels are assigned and the program is saved, the labels are embedded in the motion program (the .str file) and will not be lost if the program is later transferred to a different VisualMotion system. New programs are loaded with the default bit names. Refer to chapter 4, Registers, for default bit names.

To add or edit a bit label:

1. Click **Add...** or select the desired bit and click **Edit...**

An **Add/Edit Bit Label** window opens. The **Name** must be explicitly entered. (When editing an existing label, the current **Name**, **Reg-Bit Number** and **Comment** are automatically filled in.)

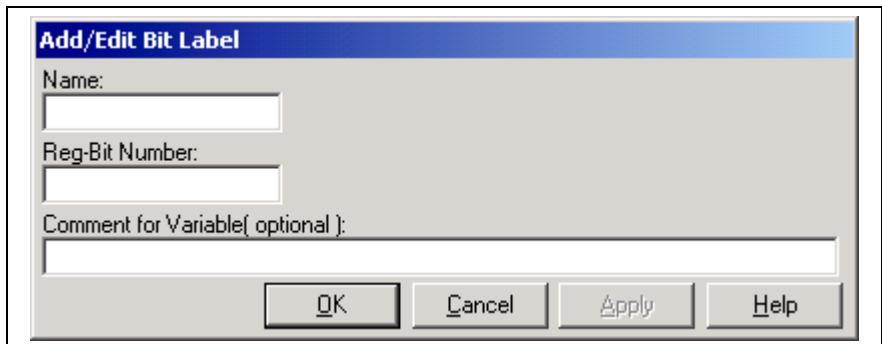


Fig. 2-56: "Add/Edit Bit Label" Window

2. Enter the desired **Name**, **Reg-Bit Number** and **Comment** (up to 80 characters).
3. Click **Apply** or **OK**.

2.4 The View Menu

The **View** menu allows the user to select a task (Initialization or A-D), subroutines, event functions for display or editing, and allows viewing nested subroutines. The user can also select pre-configured icon palettes for single axis, coordinated, Electronic Line Shafting (ELS) and Utility.

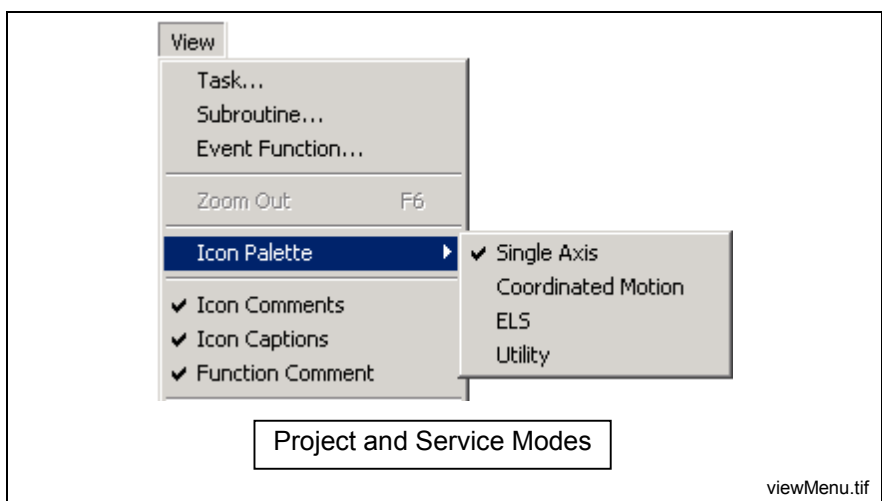


Fig. 2-57: The View Menu

Task...

Selecting **View** ⇒ **Task** displays the System Tasks Window. This window is used to navigate between the different task available in the system.

Note: VisualMotion task can also be viewed by using the Project Navigator. Refer to Project Navigator on page 2-3 for details.

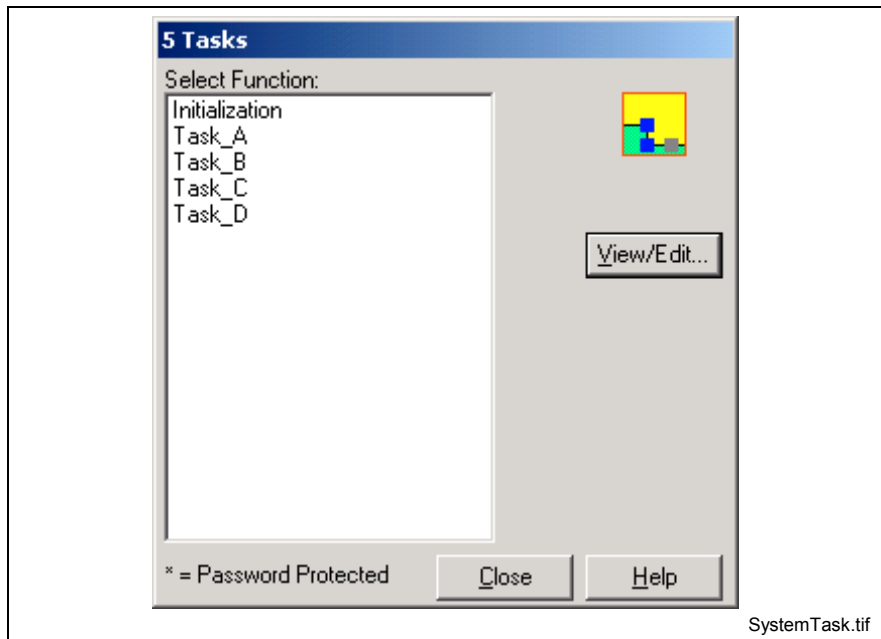


Fig. 2-58: System Tasks


The initialization task is used to configure all icons that are initialized during the SERCOS Phase 2 to Phase 3 transition. The icon palette to the left of the workspace is displayed with only those icons that are initialized in Phase 2. Icons such as the Axis setup or ELS setup icons are placed between the start and finish icons.

Note: The single axis, coordinated motion, ELS and Utility icon palettes are **not** accessible from the Initialization task.

When Task A – D are selected, icon palettes not accessible in the Initialization task can be selected from the View menu or by clicking the different icon tabs at the bottom of the displayed icon palette.

Note: The initialization icon palette is not visible when in task (A-D).

Subroutines

Selecting **View** ⇒ **Subroutines** displays the subroutine window. Only subroutines that were created using **Insert** ⇒ **Add Subroutine** or the subroutine icon () can be viewed. Subroutines can be viewed for both the Initialization and Task A-D. Password protected subroutines are displayed with an asterisk.

Note: VisualMotion subroutines can also be viewed by using the Project Navigator. Refer to Project Navigator on page 2-3 for details.

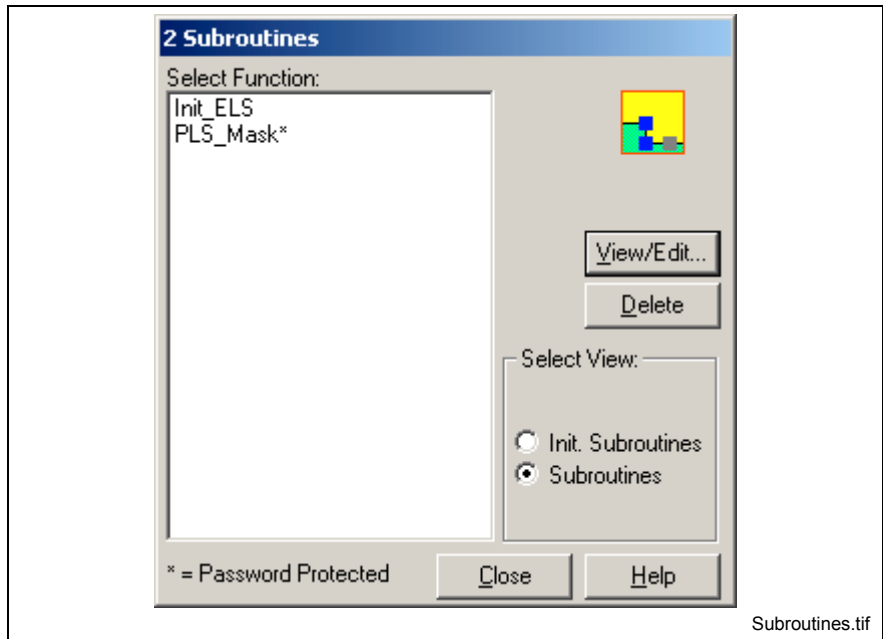


Fig. 2-59: "Subroutines" Window

The programmed subroutines are listed by name with the total number listed in the title bar. Selecting a subroutine from the list permits deleting or replacing the current VisualMotion workspace with the selection.

A subroutine can also be viewed in the workspace by double clicking an existing subroutine icon to display the **Subroutines** icon window. Refer to the Sub1 icon description in chapter 3 for details.

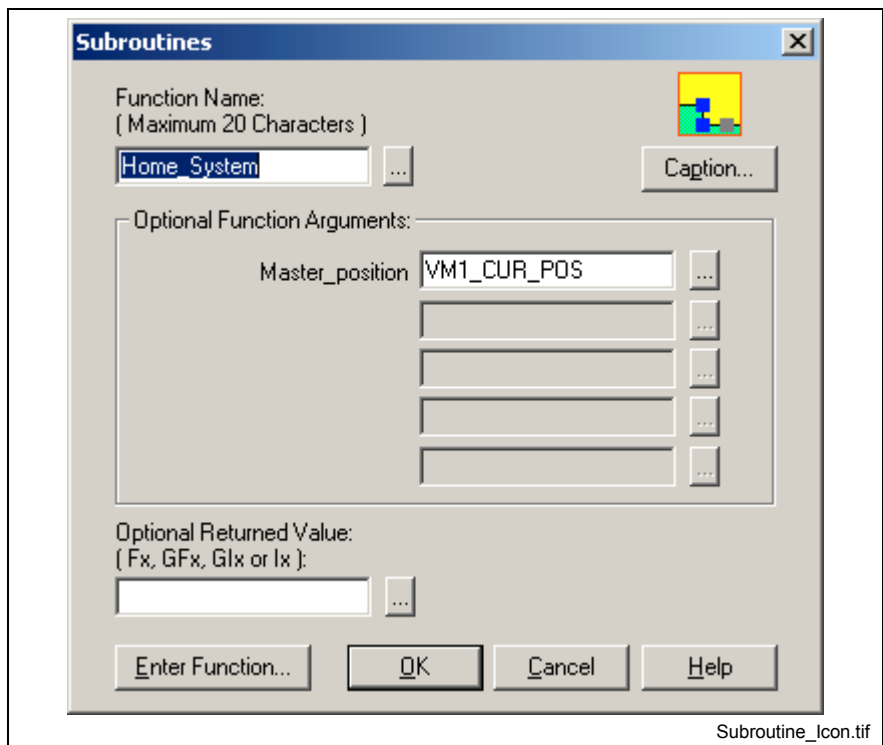


Fig. 2-60: "Subroutines" Icon Window

In this window, optional arguments and an optional returned value could be entered. The subroutine's icon name is displayed as the default. Clicking **OK** exits the window. Clicking **Caption** displays another window, which permits entering/editing of the icon caption and comment text. Clicking **Enter Function...** displays the subroutine in the VisualMotion workspace.

Event Functions

Selecting **View** ⇒ **Event Functions** displays the Event Functions window. Only events that were created with **Insert** ⇒ **Event Function**, the event icon (⚡) or the VM Data Table icon (📊) can be viewed. Password protected subroutines are displayed with an asterisk.

Note: VisualMotion event functions can also be viewed by using the Project Navigator. Refer to Project Navigator on page 2-3 for details.

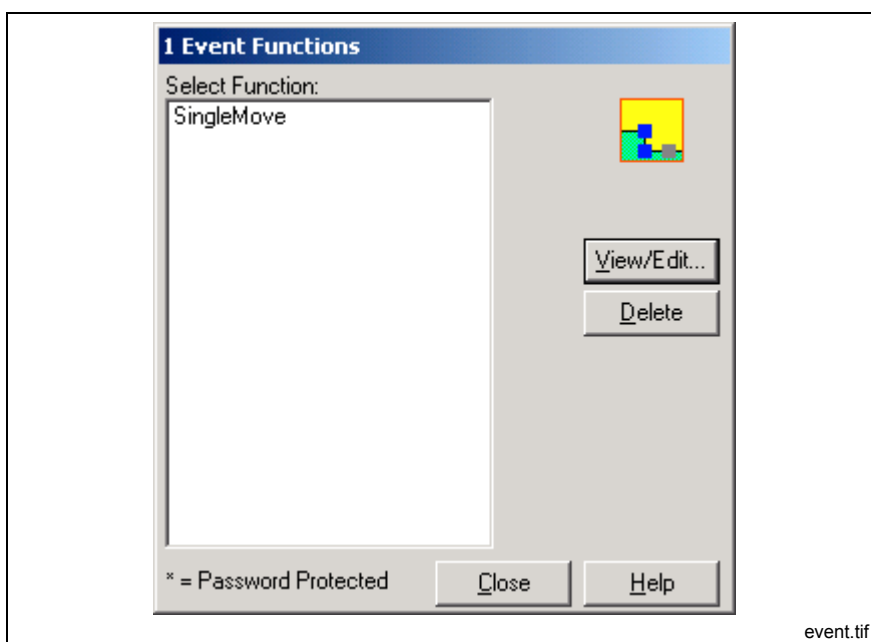


Fig. 2-61: "Event Function" Window

The programmed event functions are listed by name in a selection list and the total number is listed in the title bar. Selecting an event function from the list permits deleting or replacing the current VisualMotion workspace with the selection.

Note: Refer to the VisualMotion 9 Application manual, chapter 5, for examples of how to use the Event icon.

Zoom Out F6

Selecting **View** ⇒ **Zoom Out F6** is used to load the VisualMotion workspace with the parent (the task or other subroutine from which the current subroutine is called) of the presently loaded subroutine.

Note: The **Zoom Out** command is only available when entering a function using the **Enter Function** button within the subroutine icon. In all other cases, the function appears grayed out in the menu selection.

Zoom Out, (or the <F6> key), is used to move back from a nested subroutine queue, one subroutine at a time. The queue of nested subroutines is displayed in the title bar of the VisualMotion workspace.

Example:

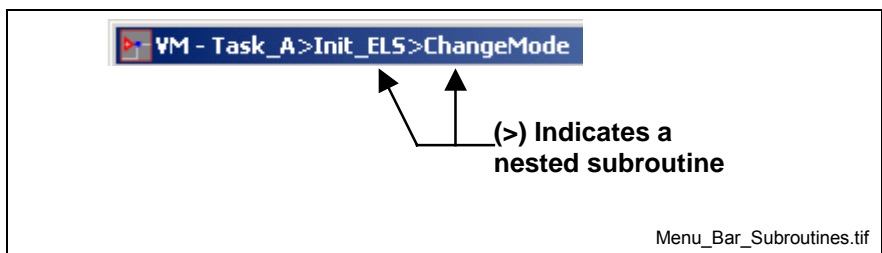


Fig. 2-62: Nested Subroutines in VisualMotion Menu Bar

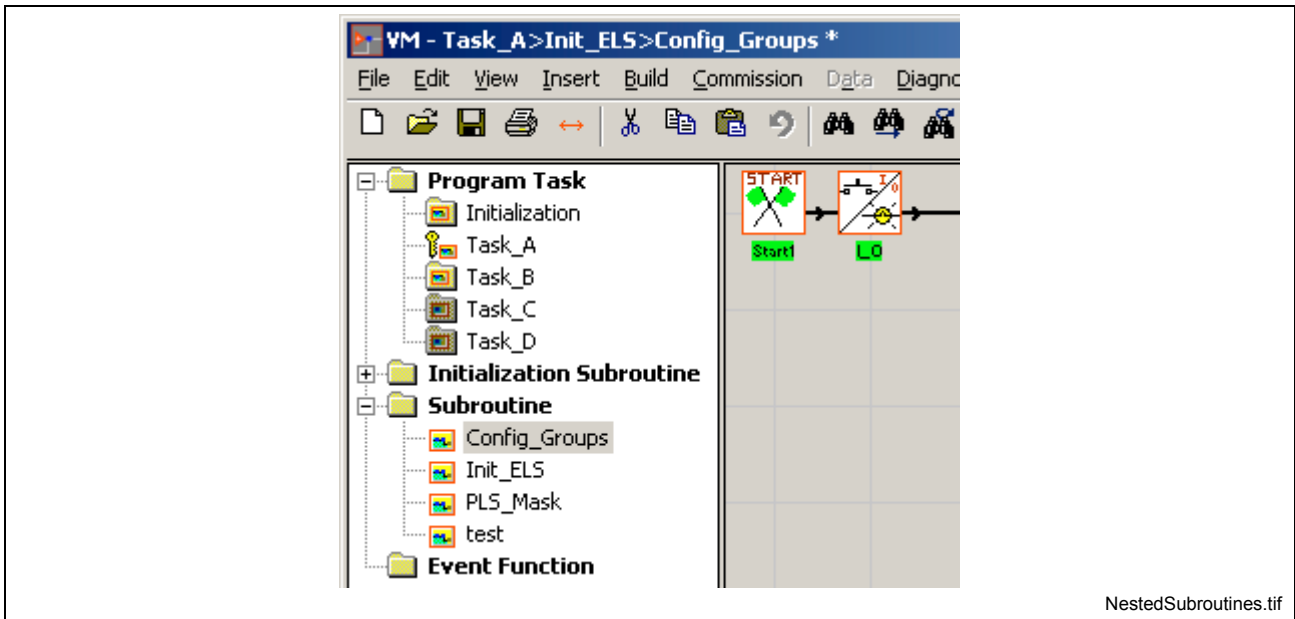


Fig. 2-63: Nested Subroutine Queue in the VisualMotion Menu Bar

The **Zoom Out** command is unavailable when any of the four main tasks (A - D) or Initialization task are loaded in the workspace. When the presently loaded subroutine is opened via the **Subroutines** menu command instead of the **Enter Function...** button. Since event functions are independent and invoked by a specified axis motion or time-based program function, they have no "parent". Therefore, **Zoom Out** cannot be used to return to a higher level from an event.

Icon Palette

VisualMotion Toolkit 9 icon palettes are displayed below the Project Navigator. Each icon palette can be selected by clicking on the small icons at the bottom of the icon palette or by selecting **View ⇒ Icon Palette**.

Note: The icon palette menu selection is only available in offline and service mode when any Task (A-D), subroutine or event function is selected in the Project Navigator.

When a project is opened, VisualMotion Toolkit launches with the Initialization task with the Initialization Icon Palette opened below the Project Navigator for GPP 9 and GMP 9 firmware.

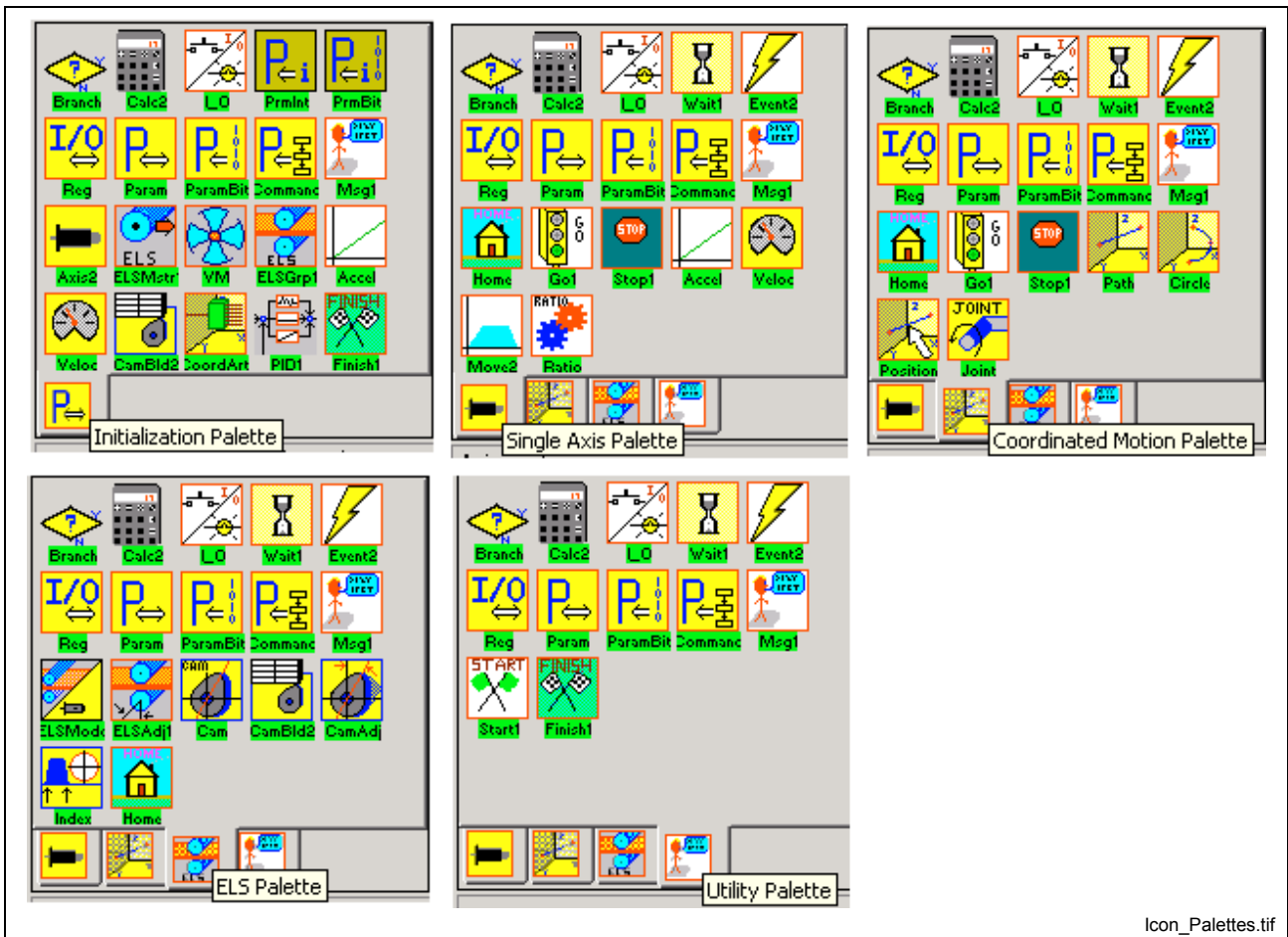
Initialization Icon Palette is used for placing icons that are initialized in Phase 2.

Single Axis Icon Palette selects a set of icons used frequently in single axis control.

Coordinated Motion Icon Palette selects a set of icons used frequently in coordinated control of robotic applications.

ELS Icon Palette selects a set of icons used frequently in Electronic Line Shifting (ELS) control.

Utility Icon Palette selects a set of general-purpose icons used in parameter setup.



Icon_Palettes.tif

Fig. 2-64: Initialization, Single, Coordinated, ELS and Utility Icon Palettes

Icon Comments

Selecting **View** ⇒ **Icon Comments** enables/disables identifying comments that appear when the mouse cursor is moved over the top of the icon. The icon comments in a program flow can be modified in the corresponding icon setup window. The setup window opens when the icon is first placed or by double clicking, the left mouse button, while the pointer is over the icon. Each window has a "Label" button to edit its comment text, by default there is no comment. The comment also appears on the printout of the setup information, if enabled or not.

Icon Captions

Selecting **View** ⇒ **Icon Captions** turns the icon labels in the VisualMotion workspace on or off. If there are no user entered labels, VisualMotion uses the default icon labels.

Function Comment

Selecting **View** ⇒ **Function Comment** enables and disables the comment header for the current task. Each VisualMotion task contains a comment header section below the icon programming workspace where the user can describe any aspect of the task. Any standard ASCII character can be used, up to a maximum of 32K.

2.5 Insert Menu

The Insert menu is available in project and service modes and is used to add subroutines and event functions to a VisualMotion project.

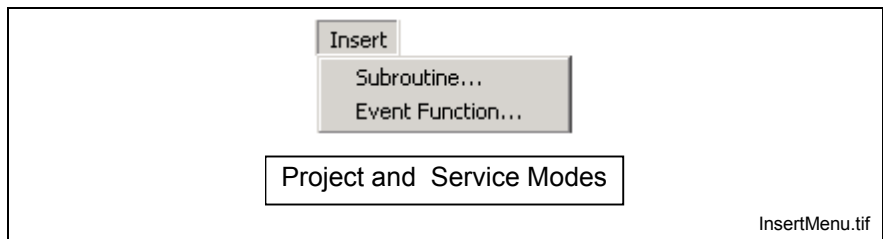


Fig. 2-65: Insert Menu

Subroutine

Selecting **Insert** ⇒ **Subroutine** opens the *Add Subroutine* window. This menu item is used to add a subroutine within a task. Subroutines are called out within the program flow and used to program functions or routines that are used within a task or program. For example, a subroutine can be used to calculate certain values before the program proceeds. Any calculation can be programmed in a subroutine, making the main task icon program less congested.

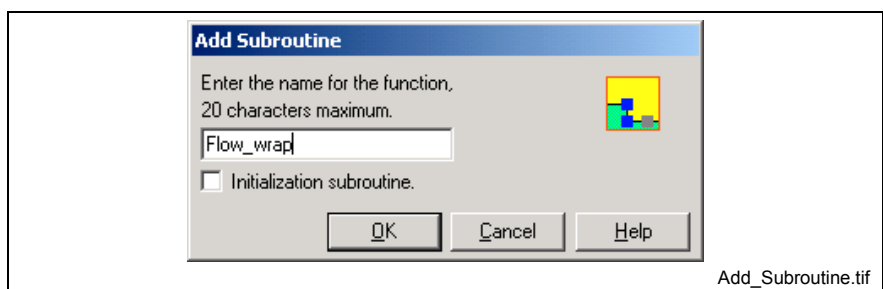


Fig. 2-66: "Add Subroutine" Window

Adding a Subroutine

The subroutine menu selection is used to open a new window where the actual subroutine function (icon program) is created. Once created, the desired program flow location from where the subroutine function will run is chosen by placing a **Subroutine Icon** and selecting the name of the created subroutine function.

Selecting a Name

To create a subroutine function, enter a name for the subroutine. This name will appear when a subroutine icon is placed in the program flow of the desired task. The name has a 20-character limitation and must begin with an alpha character with no spaces. Pressing the **OK** button opens a subroutine workspace with the name entered, a **Start** and **Finish** icon in place. You may then write a subroutine function using icons and connecting lines in the same manner as when writing an icon program task.

Initialization Subroutine

The initialization subroutine checkbox is used to identify whether or not the subroutine function will be used in the initialization task or in one of the 4 standard tasks (if left unchecked). The icons that are allowed for placement in the subroutine function depend on the initialization subroutine checkbox. If checked, only Phase 2 (initialization) icons are allowed. If left unchecked, then the icons available in the Single Axis, Coordinated, ELS and Utility palettes are available for placement.

Note: A maximum of 500 screens, consisting of tasks, subroutines and event functions are allowed.

Event Function

Selecting **Insert** ⇒ **Event Function** opens an *Event Function Control Block* window.

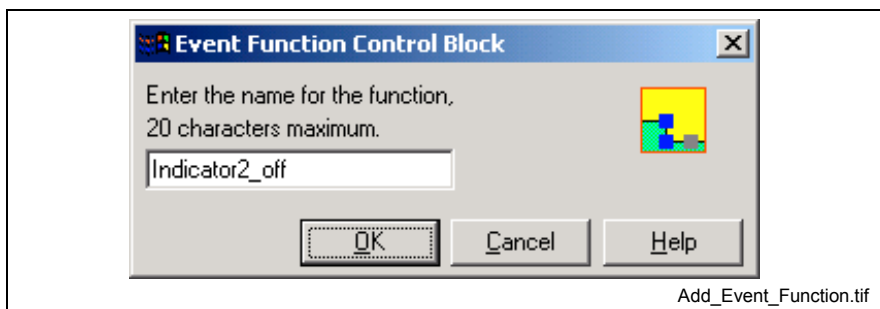


Fig. 2-67: "Event Function Control Block" Window

Unlike subroutines, event functions are not "called" from a program. Instead, they are "triggered" by the conditions (distance, time, etc.) that are specified in an event setup. Refer to the **Calc Icon** and **Event Icon** in Icon Programming chapter for details.

Note: An event function must be written before it can be assigned to an event trigger.

Entering a name for an event function and clicking **OK** opens an event function workspace with the name entered and the **Start** and **Finish** icons in place. You may then write an event function using icons and connecting lines in the same manner as when writing an icon program task.

Note: A maximum of 500 screens, consisting of tasks, subroutines and event functions are allowed.

2.6 The Build Menu

The **Build** menu contains commands for compiling and managing VisualMotion user programs.

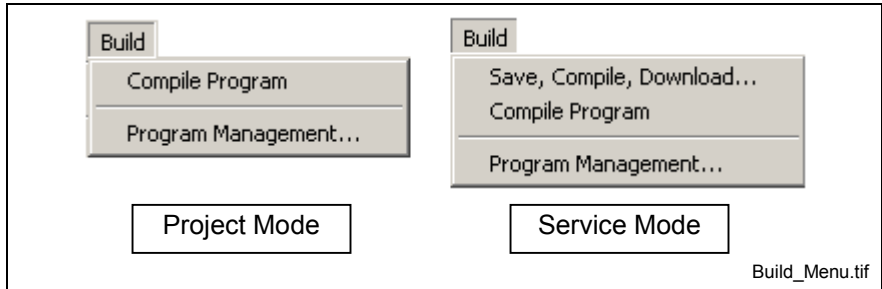


Fig. 2-68: The Build Menu

Save, Compile, Download

Saves the currently open program, compiles it, and downloads it to the control. The Program Management window opens when all three functions have successfully been carried out.

Note: This menu item is only available when VisualMotion Toolkit is in service mode.

Compile Program

This selection checks and converts the current project to code that the control can interpret. When complete, the 2nd Pass Compiler opens the following window relating information about the compiled program. All component files of the project are saved to the project folder.

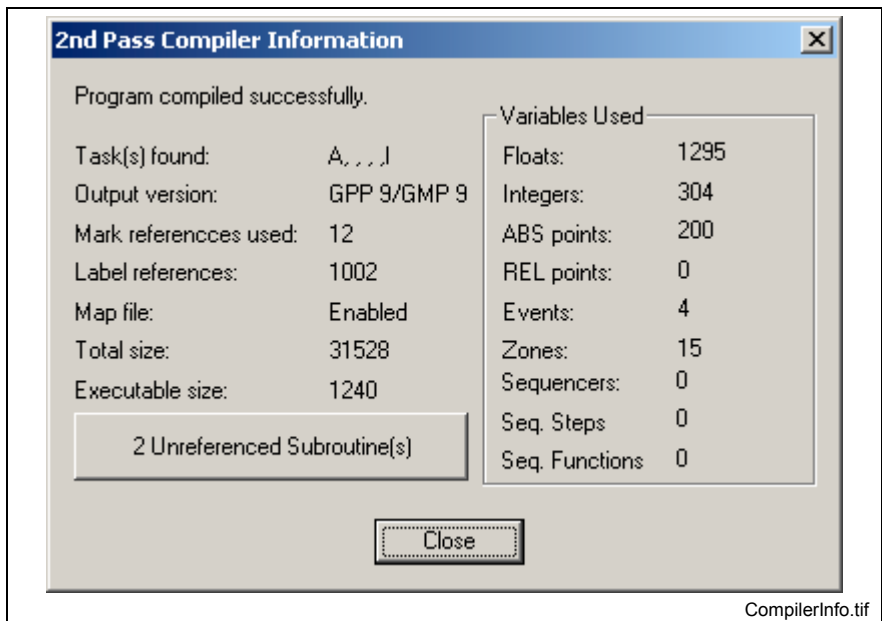


Fig. 2-69: Compiler Information

The **Unreferenced Subroutine(s)** button indicates how many unused subroutines exist in the program. Clicking the button opens the window below indicating the name and type of subroutine(s). Selecting a name displays the icon program in VisualMotion Toolkit's main window.

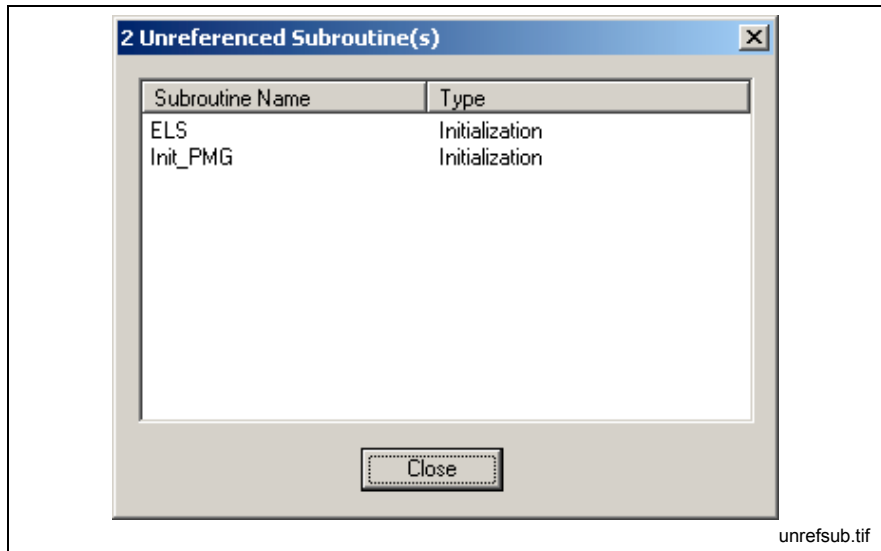


Fig. 2-70: Unreferenced Subroutine(s) Window

Program Management

The Program Management window is used to activate and download VisualMotion user programs to the control or to transfer data between programs in the control's memory. A maximum of ten (10) user programs can be downloaded to the control with only one activated at any given time. The currently active program name is displayed in green text with a (>) symbol to the left of the name.

Note: This menu selection is available only while in online project mode or service mode.

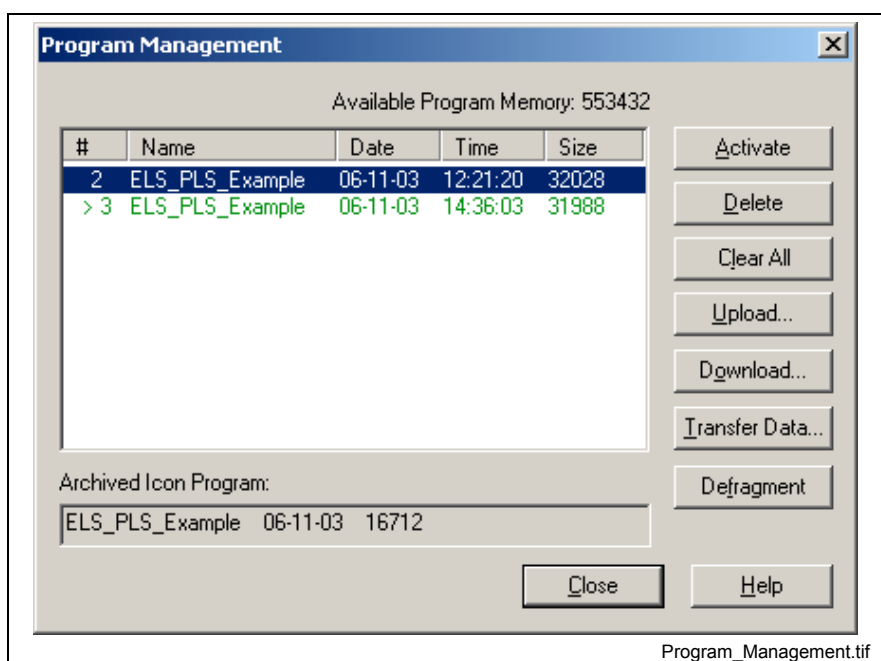


Fig. 2-71: "Program Management" Window

Note: The PPC-R control uses a flash file system for storing user programs. When a new program is activated, the old program is removed from local memory and stored to flash. Every activation of user program reduces the amount of flash memory. This process causes flash memory fragments. When the flash memory fragment threshold is reached, VisualMotion automatically runs a defragmentation process to restore unused memory.

The Program Management window provides the following buttons:

Activate

This button is used to activate a different program previously downloaded to the control's memory. By default, the currently active program is selected and highlighted. A different program cannot be activate when the active program is running.

Delete

This button deletes the highlighted program from the control's memory. A confirmation window opens before a program is deleted. Active programs can not be deleted from the control's memory.

Clear All

This button erases all resident programs and data from the control's memory (including the active program). A confirmation window opens before all programs are deleted. Refer to **Archive** before clearing programs from the control's memory.

Note: The control must be parameter mode to clear all the programs.

Upload...

This button uploads the highlighted program for archiving on the host system's hard drive. When prompted with the **Save As** window, select the file destination and filename. A progress bar indicates the upload status.

Download...

This button downloads a compiled program to the control's memory. After clicking the **Download** button, select or type the desired filename, and then click the **Open** button. Executable programs are stored together with other project files in the project Directory. After clicking **OK**, the download is executed, and a progress bar shows the download status.

Data Transfer

This button transfers data items between the active program (Source) and the (Destination) program selected for activation. This button becomes active when a program, other than the active program, is selected. Clicking on the **Data Transfer** button opens the Data Transfer window.

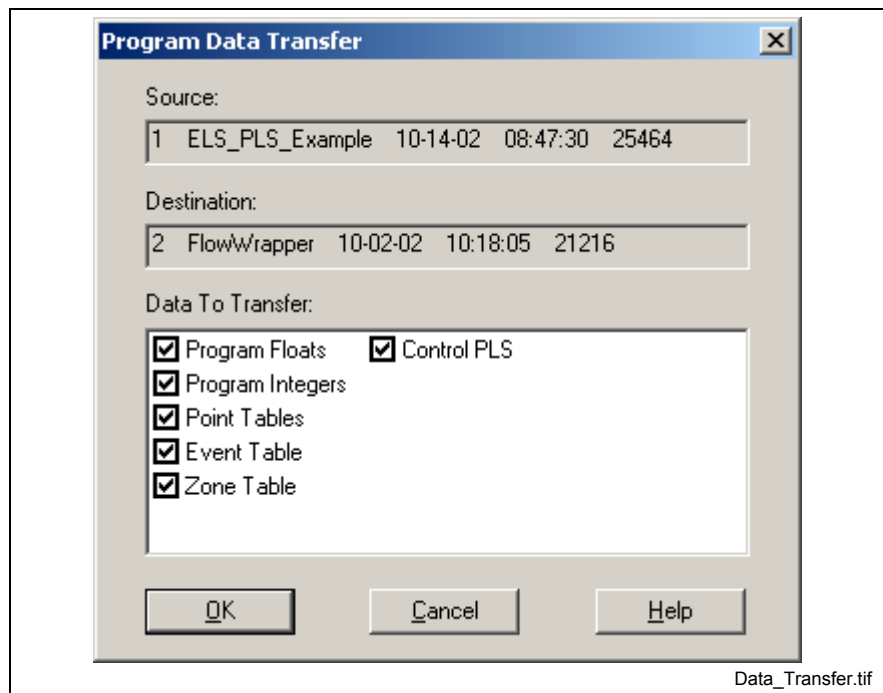


Fig. 2-72: Selective Data Transfer

Clicking on **OK** copies all the data items from the control's active program to the target program selected from the Program Management resident program list.

Clicking on **Selective Transfer** allows the user to selectively choose the following data types:

- Floats
- Integers
- Absolute and Relative Points
- Events
- Zones
- Sequencers
- Control PLS



Data Transfer overwrites the data sets of the destination program.

⇒ If the source program data allocation (i.e., the amount of configured data types) is larger than the target, then only the data elements within the "VM Data" allocated by the target are transferred.

For Example:

If the active program has 75 events and the selected target program has a only 50 events specified in the VM Data Table, only the first 50 events of the active program will be transferred. Data Transfer is a useful tool for developing programs by incrementally testing and modifying sequential copies of a working program without the need to continuously re-input data sets for the new program.

Defragment

This button is used to force a flash compression of unusable control memory. Unusable memory is created when programs and control data is updated or cleared. This low priority task runs in the background. The status bar indicates the progress of the defragmentation.

2.7 The Commission Menu

The **Commission** menu contains tools for setting up and configuring program related interfaces, such as drives, I/O, fieldbus interfaces and PLS functionality. The user can also configure Position Monitoring Groups and Coordinated Motion requirements, archive and transfer program data.

Note: The Transfer menu selection only appears when VisualMotion Toolkit is in service mode.

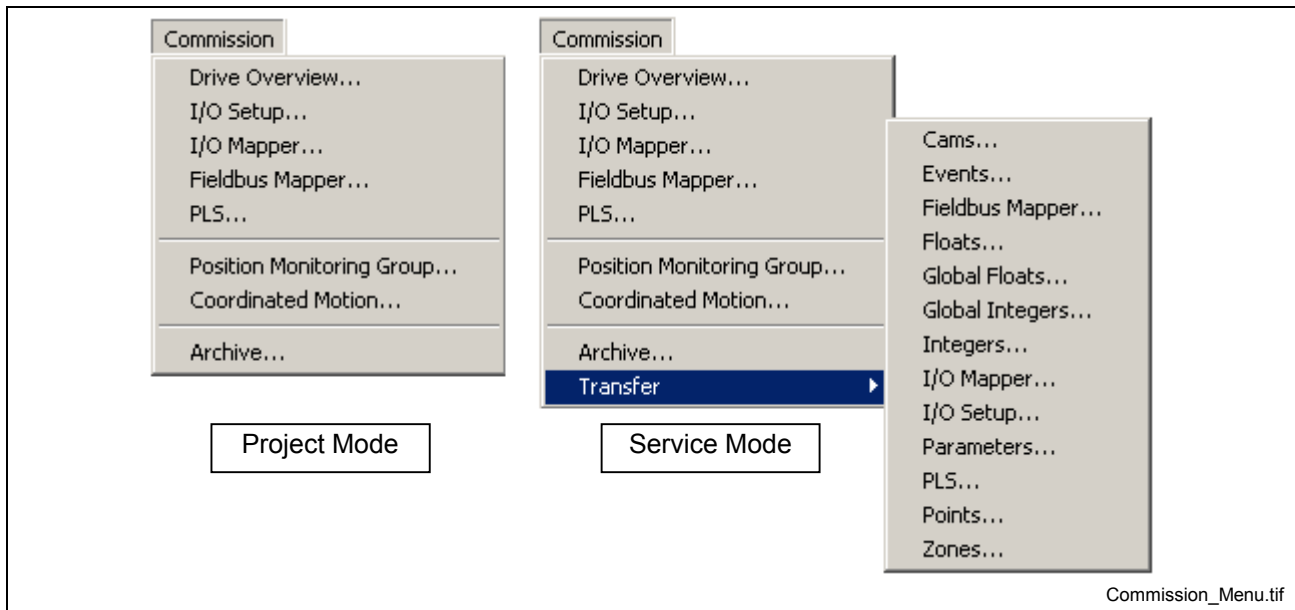


Fig. 2-73: The Commission Menu

Drive Overview

Selecting **Commission** ⇒ **Drive Overview** for a GPP9 program opens the DriveTop initial selection window. From this window, the user can select whether to view an overview of the selected drive or commission the drive.

The Drive Overview is only accessible in service or online mode.

Note: When opening VisualMotion programs created with firmware version such as GPP7 or GPP8, the Drive Parameter Editor found in VisualMotion Toolkit 8 will launch. Refer to the VisualMotion 8 Application manual for details.

For instructions on how to use this program, refer to the *VisualMotion 9 (GPP) Application Manual, Drive Tools*.

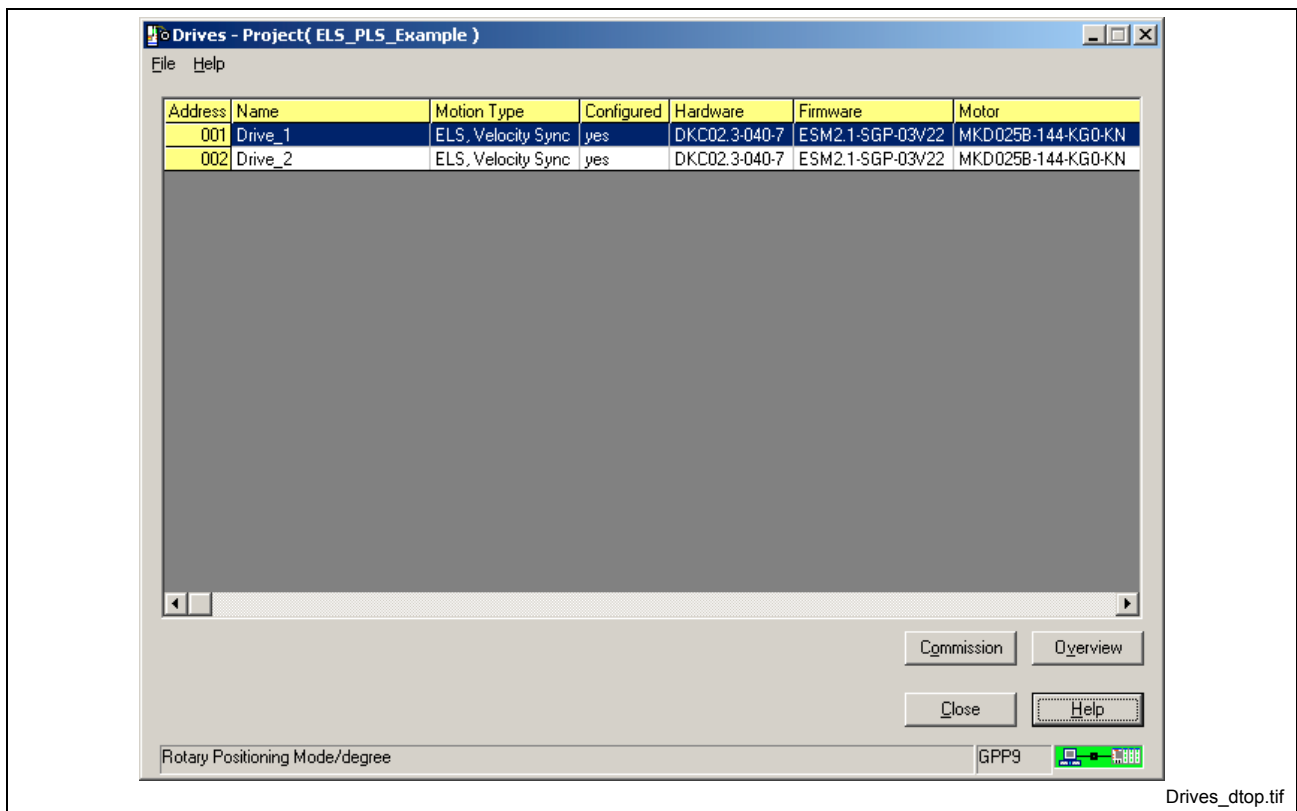


Fig. 2-74: Drive Parameter Editor

Commission

The *Commission* button is used to setup the selected drive. When clicked, a wizard will run, guiding the user through the configuration process.

Overview

Selecting the *Overview* button will open the *DriveTop Status* window in Fig. 2-75. All drive setup windows initially configured using the *Commission* button can be selected from the menu items found in the *DriveTop* status window.

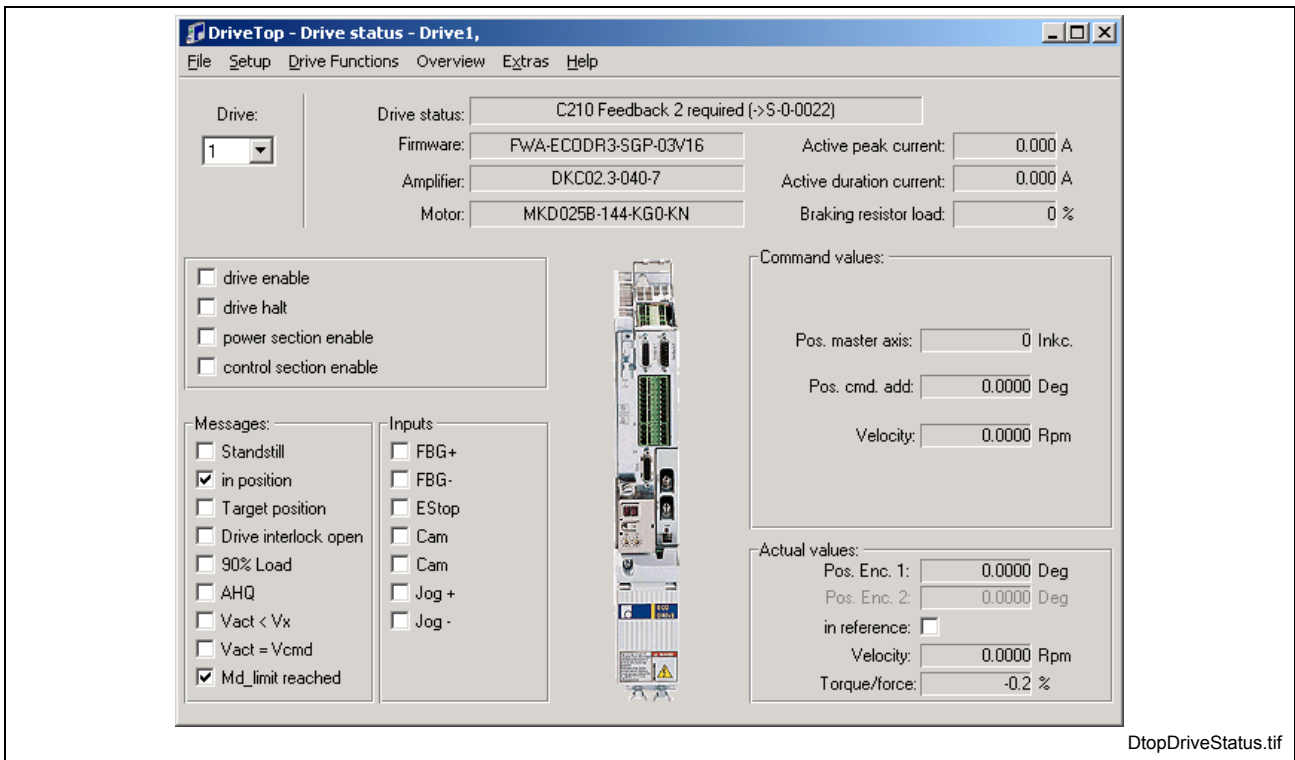


Fig. 2-75: DriveTop Drive Status

I/O Setup

Selecting **Commission** ⇒ **I/O Setup** opens the I/O Configuration window in Fig. 2-76. This tool is used to assign VisualMotion's system I/O (RECO 02 I/O Modules and Drive I/O).

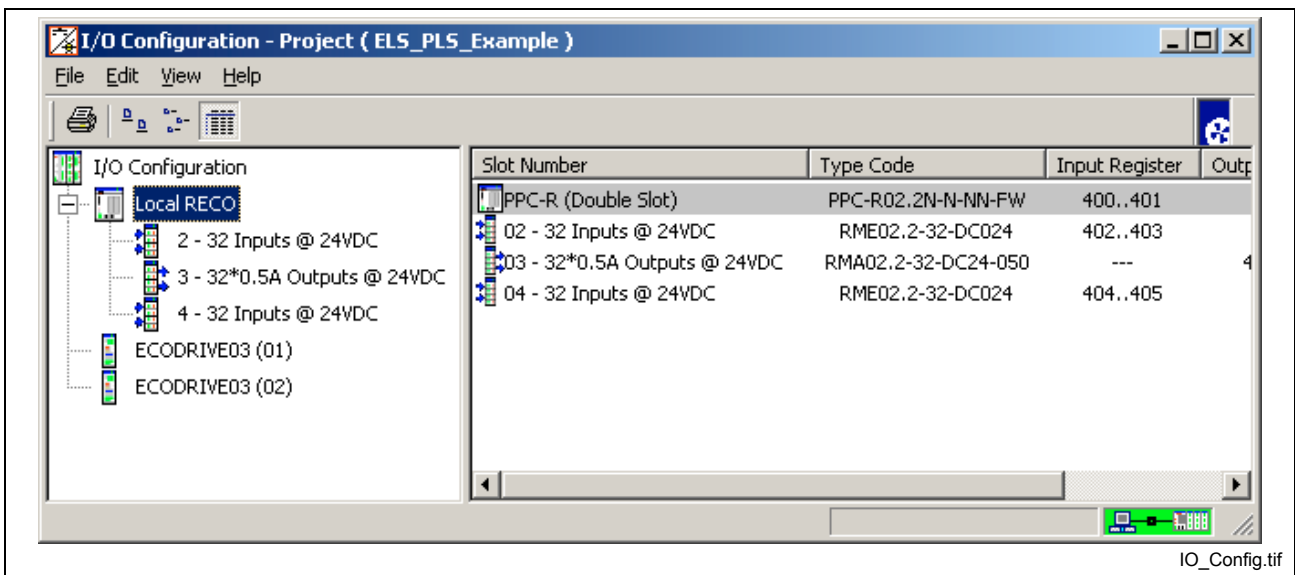


Fig. 2-76: I/O Configuration Window

For information about how to use this tool, refer to chapter 6, **I/O Configuration Tool**.

I/O Mapper

Selecting **Commission** ⇒ **I/O Mapper** opens the following window:

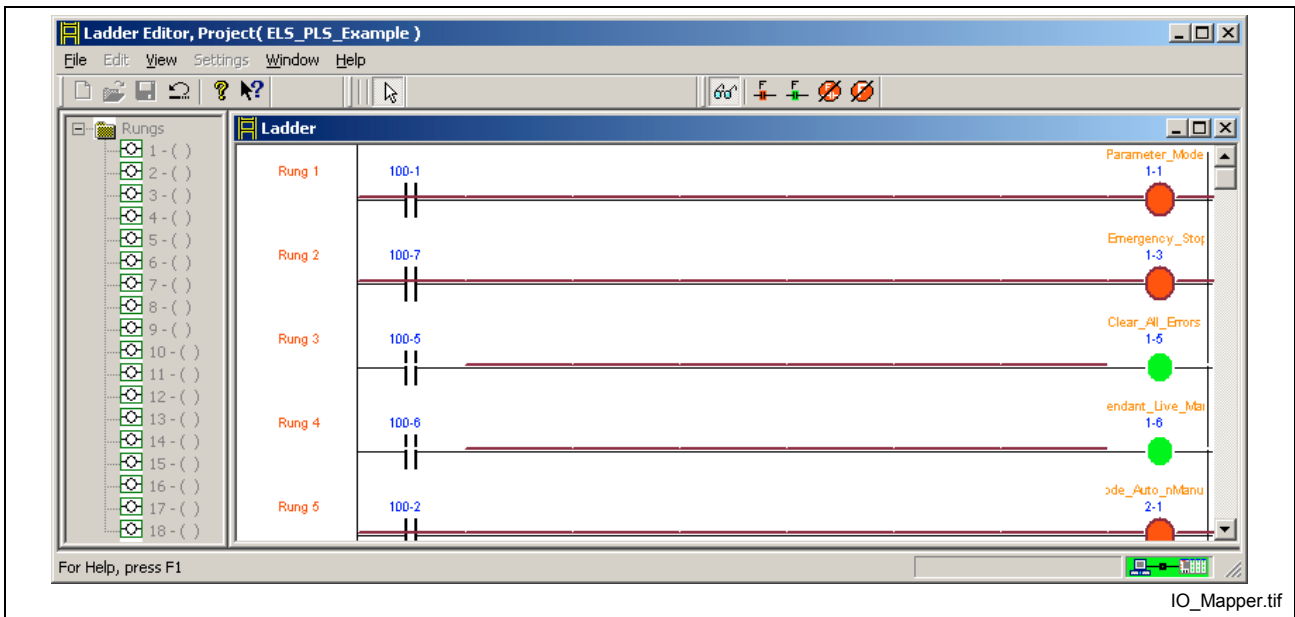


Fig. 2-77: Enhanced I/O Mapper

For information about how to use this tool, refer to chapter 7, **Enhanced I/O Mapper**.

Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to setup fieldbus configuration and data mapping. Using this tool, the user can map data to and from the control, and save specific mapping lists as a file or download/upload these lists from/to the control.

The following figure shows the main Fieldbus Mapper window:

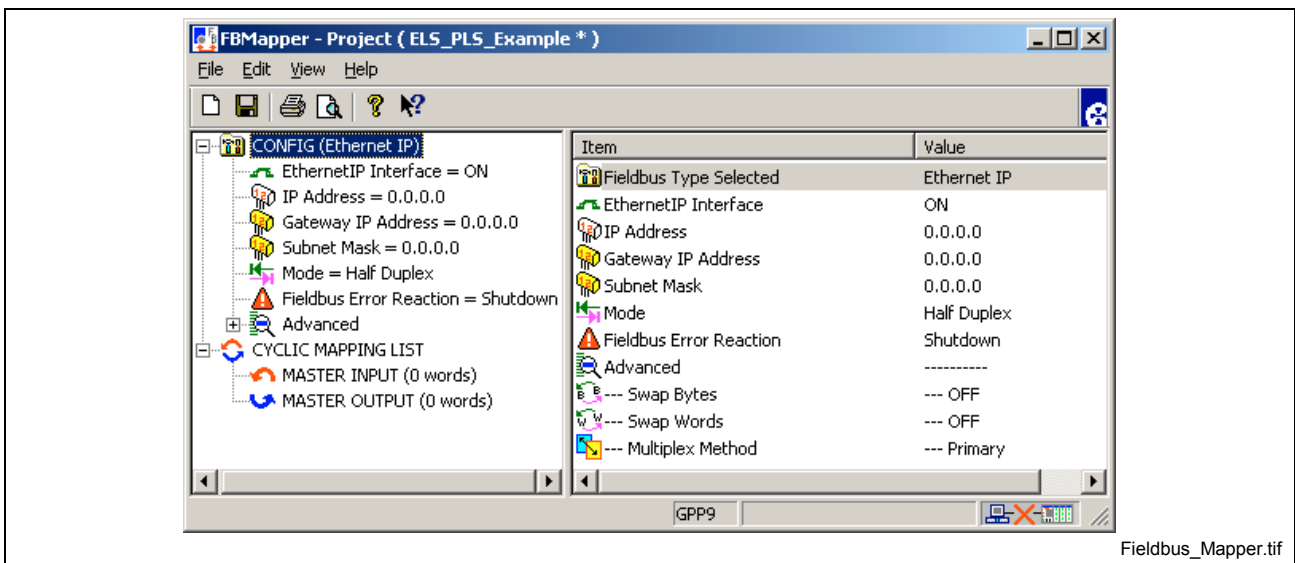


Fig. 2-78: Fieldbus Mapper Main Window

The available fieldbus types for VisualMotion 9 are:

- Profibus
- DeviceNet
- Interbus
- ControlNet
- EtherNet/IP

The data types that can be mapped include:

- Variables (Integer, Global Integer, Float and Global Float)
- Parameters (Axis, Card and Task)
- Registers

For specific information about each fieldbus and for directions on how to map data, refer to the *VisualMotion 9 Application Manual* sections: [Profibus Fieldbus Interface](#), [DeviceNet/ControlNet/EtherNet IP Fieldbus Interfaces](#) and [Interbus Fieldbus Interface](#).

PLS (Programmable Limit Switch)

A Programmable Limit Switch is used to switch on and off digital outputs based on the input position of an associated axis or master. The basic component of a PLS will be referred to as a PLS object.

The parameterization of Control, Drive and Option Card PLSs can be done with the VisualMotion's PLS tool.

To launch the PLS tool, select **Commission** ⇒ **PLS** from VisualMotion's main menu.

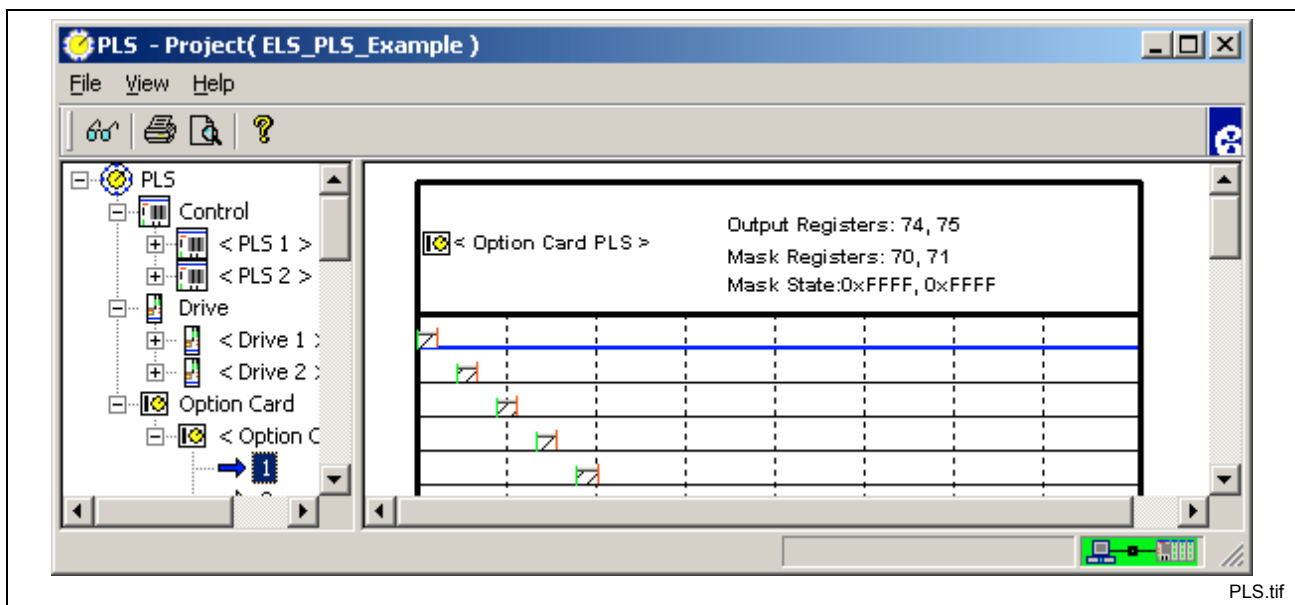


Fig. 2-79: PLS Configuration Tool

For information about how to use this tool, refer to chapter 8, [Programmable Limit Switch Functionality](#).

Position Monitoring Group

Certain applications require position deviation monitoring between multiple axes on a machine when following the same position command signal. Should one of the drives in a group deviate from a set-monitoring window, an error reaction can be set to warn the user of the deviation or stop all drives (issuing a fatal error). This function provides independent monitoring of multiple groups of axes including the X, Y and Z of a coordinate system, simultaneous single axes moves and axes in an ELS group or an ELS system master. The *Position Group Monitoring* window allows the configuration of up to 8 independent groups containing a maximum of 6 axes (1 primary and 5 slaves) per group. All configured axis groups are monitored every SERCOS cycle.

Selecting **Commission** ⇒ **Position Monitoring Group** opens the window in Fig. 2-80.

Note: In offline mode, the Position Group Monitoring window is used to configure up to 8 groups. In online mode, the window is used to view the status of current, peak and maximum deviation values for all groups.

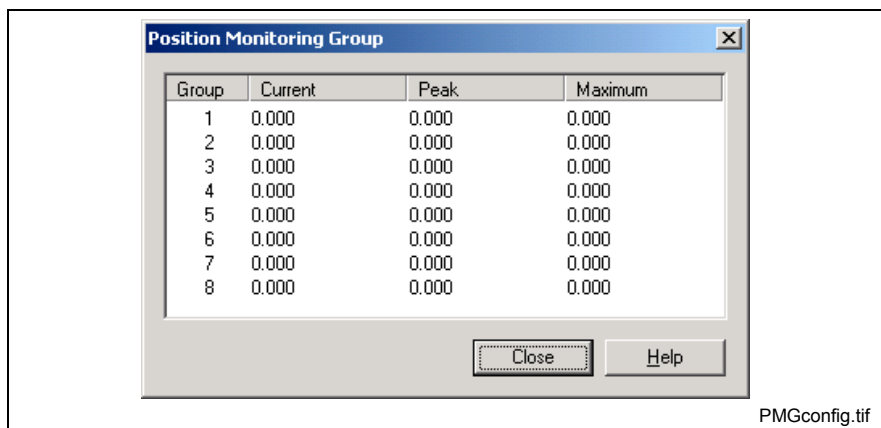


Fig. 2-80: Axis Position Monitor

Relevant Parameters for Axis Group Monitoring

The following control parameters are used when configuring Axis Position Monitoring.

Parameter	Description
C-0-3201, C-0-3211, C-0-3221, C-0-3231, C-0-3241, C-0-3251, C-0-3261, C-0-3271	PMG # Maximum Allowed Deviation Window (# identifies groups 1-8)
C-0-3202, C-0-3212, C-0-3222, C-0-3232, C-0-3242, C-0-3252, C-0-3262, C-0-3272	PMG # List of Axis (# identifies groups 1-8)
C-0-3203, C-0-3213, C-0-3223, C-0-3233, C-0-3243, C-0-3253, C-0-3263, C-0-3273	PMG # List of Position Offsets (# identifies groups 1-8)
C-0-3204, C-0-3214, C-0-3224, C-0-3234, C-0-3244, C-0-3254, C-0-3264, C-0-3274	PMG # Current Peak Group Deviation (# identifies groups 1-8)
C-0-3205, C-0-3215, C-0-3225, C-0-3235, C-0-3245, C-0-3255, C-0-3265, C-0-3275	PMG # Maximum Deviation (# identifies groups 1-8)
C-0-3206, C-0-3216, C-0-3226, C-0-3236, C-0-3246, C-0-3256, C-0-3266, C-0-3276	PMG # Configuration (# identifies groups 1-8)

Table 2-5: Position Monitoring Groups

System/Position Initialization

Before position monitoring can become active, the PMG parameters are processed. This process dynamically creates and initializes internal data structures with valid settings. The processing of the *PMG # List of Axis* and *PMG # LIST of Position Offsets* runs during every occurrence of system initialization (Phase- 2 → Phase-4). If any parameter is set incorrectly, errors are reported before the system is able to run in phase 4. In general, all position signals must have the same data format (in, mm, deg) and likewise, mechanical system (M/N ratios, feed constants/modulo, etc...). The system automatically detects if the position master is a modulo axis (e.g. reads S-0-0076, bit 7) and calculates the deviation window accordingly. The deviation window parameter will be scaled in the units of the master signal.

Note: If an axis requires drive homing, monitoring can only become activated after the homing process has been completed on all the axes in a grouping.

Configure Monitoring Group

A monitoring group is configured by double clicking on a group number row in the *Position Monitoring Group* window. From the *Position Monitoring Group Settings* window, the user selects the Deviation Method, Primary Signal, Error Reaction on Maximum Deviation and Maximum Deviation Window for the group.

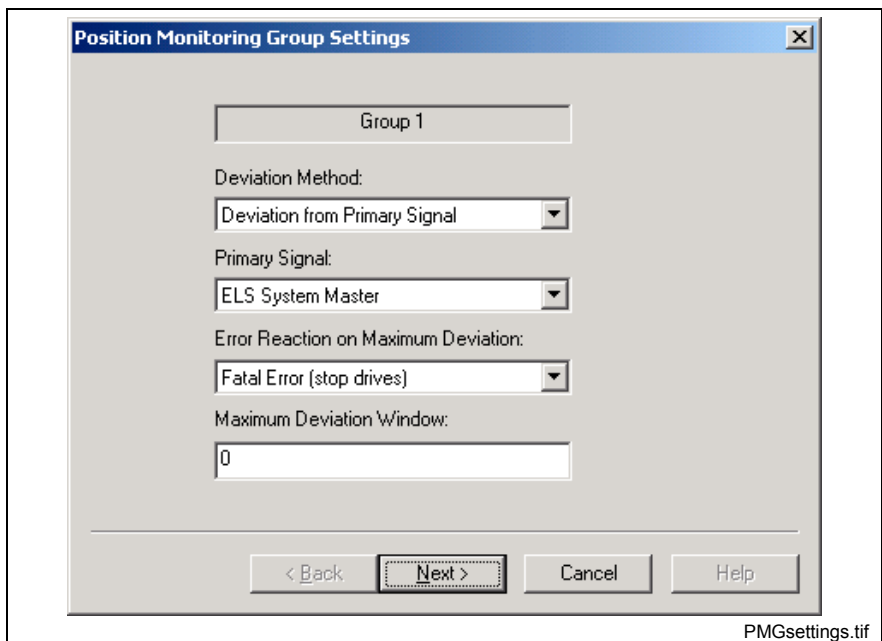


Fig. 2-81: Position Monitoring Group Settings

Deviation Method

This selection sets the method of deviation that will be used in the system. The selections include:

- Deviation from Primary Signal
- Min/Max Group Deviation Window

Deviation from Primary Signal

The system monitors the maximum allowable position difference between the selected primary (master) signal and the group of slave signals.

How it works:

During runtime, the group's corresponding enabled bit (PMG#_ENABLE, REG. 86) is checked. If enabled, the group's maximum deviation window is calculated. This calculated value is compared to each of the slave's position, including offset, to determine if it exceeds the maximum deviation window value. If a slave's position is outside of the maximum deviation window, the selected error reaction is executed.

Min/Max Group Deviation Window

The system monitors the maximum allowable position difference between the slaves in a group and compares it to the set maximum deviation window.

How it works:

During runtime, the group's corresponding enabled bit (PMG#_ENABLE, REG. 86) is checked. If enabled, each slave's position, including offset, in a group is analyzed. The slaves are sequentially checked based on their order. Next, the Min/Max value is derived by taking the absolute difference between the current Min and Max value and comparing it to the group's maximum deviation window. If the group's Min/Max value is greater than the maximum deviation window, the selected error reaction is executed.

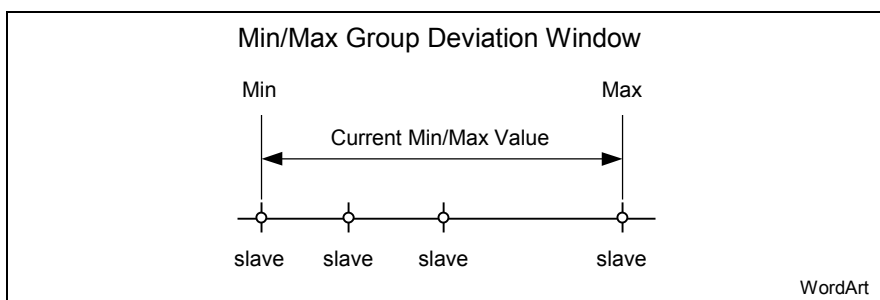


Fig. 2-82: Analyzing Min/Max Value

Primary Signal

The primary signal is the master signal that is used when the deviation method is set to "Deviation from Primary Signal". The available primary signal sources are listed in the table below.

Primary Signal	Description
Feedback Position (A-0-0102)	An axis' feedback position is used as a master signal
Command Position (A-0-0101)	The axis' command position value is used as a master signal
ELS System Group Position	An ELS Groups (1-8) output position is used as a master signal
ELS System Master	An ELS System Master (1-6) is used as a master signal

Table 2-6: Primary Signals

Error Reaction on Maximum Deviation

This selection sets the error reaction that is used when a slave axis is outside the set maximum deviation window. When set to *Fatal Error (stop drives)*, all slaves axis will stop and the control will display the error, "554 Excessive Deviation in PGM%d, see ext. diag". When set to *Warning (user defined)*, motion to all slaves continues and the control will display the warning, "220 Excessive Deviation in PGM%d, see ext. diag".

Maximum Deviation Window

The value in this field represents the maximum positional deviation between the primary signal and slave axes, when using *Deviation from Primary Signal*, and between a minimum and maximum axes positions in a group, when using *Min/Max Group Deviation Window*. This value is stored in control parameter C-0-32x1 (PGM # Maximum Allowed Deviation).

Note: The maximum allowable value that can be entered is 90. The PMG uses the current system unit settings.

The following figure illustrates how the maximum deviation value is applied to both Deviation Methods.

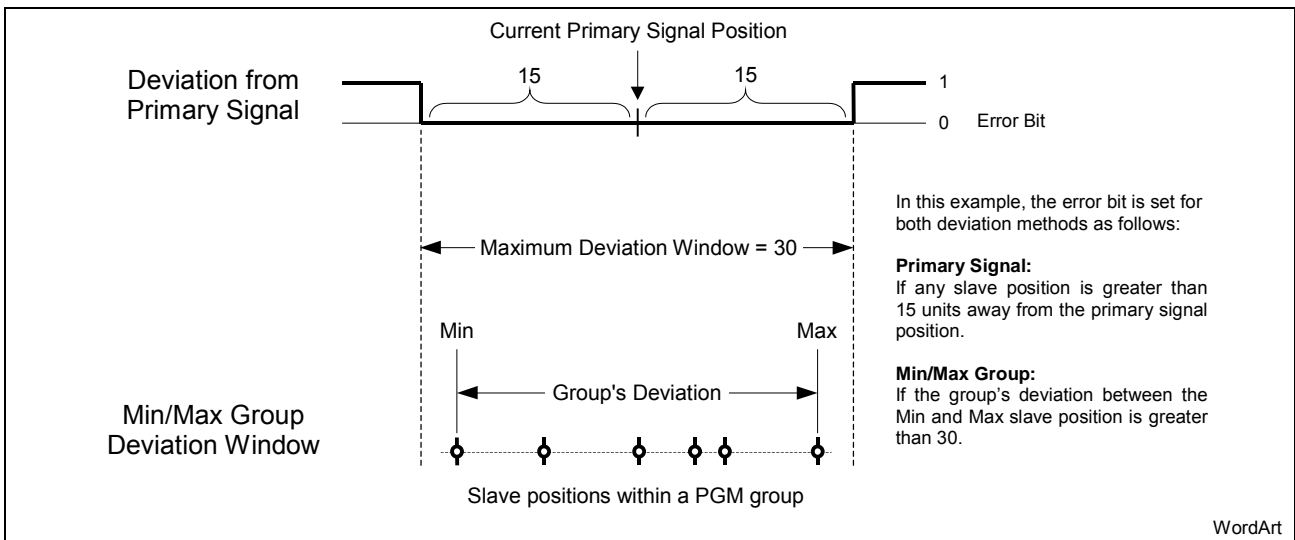


Fig. 2-83: Maximum Deviation Window

Note: Changes to the maximum deviation window, while a group is enabled, has no immediate effect. The group's control register enable bit (Reg. 86 bits 1-8) must be disabled and then set before a deviation window change can take effect.

Error Detection and Reset Monitoring Groups

The system monitors the maximum allowable position difference between any number of drives identified in control parameter C-0-32x2 (PGM # List of Axis). Should one or more axes exceed the maximum deviation window value, the error bit(s) (Reg. 87, bits 9-16) will be set.

Note: PMG error bits are set to 0 when a PMG group is disabled.

Note: The programmer must react to these conditions and program a corrective routine based on the machine's design and/or requirements.

One method could be to arm an Extended Input or Task Interrupt event to fire during the error. After the error has been detected and corrected, the programmer must rearm monitoring by transitioning the group's control register bit (Reg. 86, bits 1-8) from Low (0) to High (1). At this point, if all axes are within the deviation window, the status register bit (Reg. 87, bits 1-8) will be set High (1) and the error bit set Low (0). If any of the axes is outside of the deviation window, then the status register bit will remain Low (0) and the error bit will remain High (1).

Slave Axis Selection

The Slave Axis Selection window is used to select the primary and group slave axis for each group. A maximum of 5 group slave axes can be set for 1 primary slave.

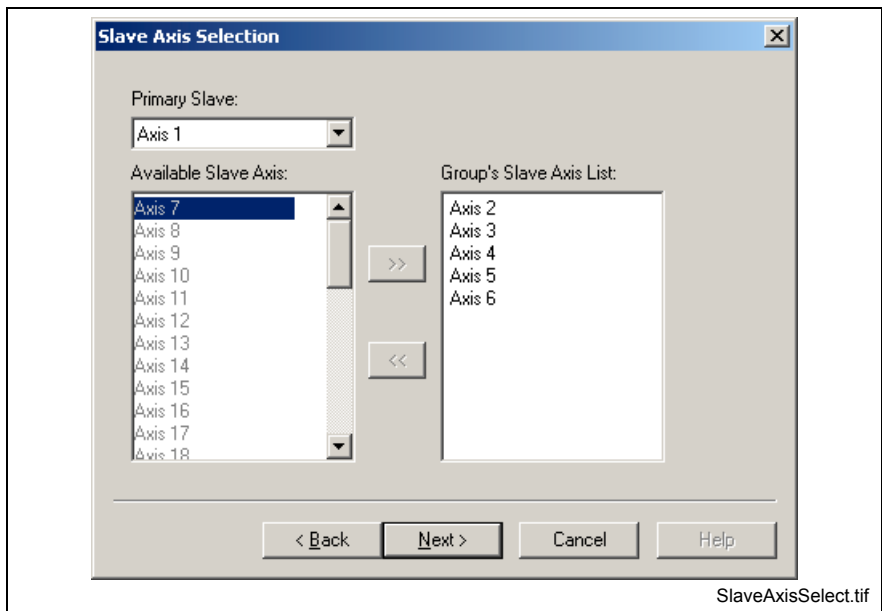


Fig. 2-84: Slave Axis Selection

Primary Slave

The available selections within the **Primary Slave** field are dependent upon the type of Primary Signal selected in the previous window. The selected primary slave is stored in control parameter C-0-32x2 (PGM # List of Axis) as the first entry. The allowable primary slave types are listed based on selected primary signals.

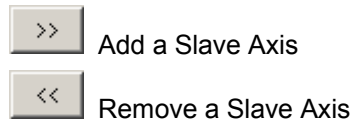
Primary Slave	Primary Signal
Axis 1-40	Feedback Position (A-0-0102) or Command Position (A-0-0101)
Groups 1- 8	ELS System Group Position
ELS System Master 1- 6	ELS System Master

Table 2-7: Available Primary Signals

Note: When an axis number is selected as a primary slave, it is removed from the **Available Slave Axis** field.

Adding Slave Axes to a Group

A maximum of 5 slave axes can be added to a *Groups Slave Axis List*. Axis are added and removed from the group's list by selecting the desired axis number and clicking on the following buttons:



Slave Offset Definition

The *Slave Offset Definition* window is used to manually set an offset position between the Primary Slave and up to 5 additional slave axes in a group.

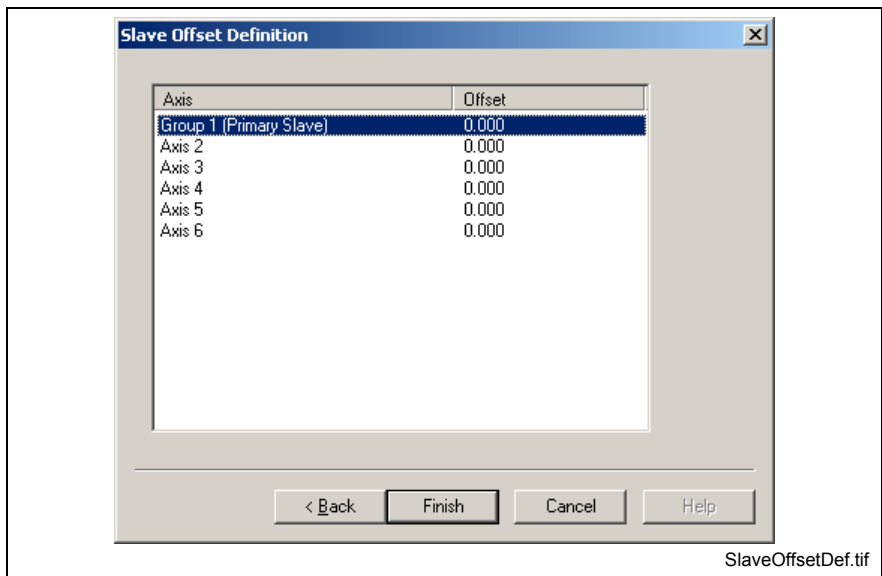


Fig. 2-85: Slave Offset Definition

Note: Slave offsets must be set after all axes in the system have been referenced for the application.

Manually Set a Slave's Offset

To set an offset, double click on any axis to open the *Axis Offset* window. The primary slave offset is normally set to zero. All other axes are assigned an offset value to account for the difference in position from the primary slave.

For example:

If the primary slave's actual feedback position were 15 units, then the slave's offset would be set as follows:

Axis	Feedback Position	Offset
Group 1 (Primary Slave)	15	0
Axis 2	20	-5
Axis 3	16	-1
Axis 4	11	4
Axis 5	09	6
Axis 6	10	5

Table 2-8: Slave Offset

Automatically Set a Slave's Offset

Slave offsets can be automatically calculated by the system to ensure accurate values. This calculation is necessary for axes that have a fixed phase deviation in an application.

Note: Offset calculations are performed before a group is enabled and used to adjust all axes positions to zero. The actual feedback position for each slave is not physically moved, but the system uses the feedback + offset position to zero out all axes positions. From a zero starting value, the PMG feature can monitor all axes positions from a zero value and issue an error if any slave is outside the Maximum Deviation Window.

Control Register 86 (PMG_Control), bits 9-16 are used to calculate the required slave axis offset for each group. After a PMG is commissioned, the offset bits can be programmed as follows:

- Group offset bits can be set using an I/O Setup icon
- A minimum Wait of 2 ms must follow the setting of an offset bit before enabling the group. This provides sufficient time for the bit to set.

Note: Changes to a group's position offset, while a group is enabled, has no immediate effect. The control register enable bit (Reg. 86 bits 1-8) for the group must be disabled and then re-enabled before a position offset change takes effect.

Coordinated Motion

The Coordinated Motion menu selection is available in online or service modes. The standard tabs that are normally displayed are as follows:

- Task Path Limit
- Jogging Accel/Decel
- Coordinated Jogging Percents
- Teach Pendant Security

Task (A-D) Path Limits

The Task Maximum Path Limits window allows speed, acceleration and deceleration limits for coordinated motion to be individually set for each task.

Note: The Task (A-D) Path Limit tabs are displayed only if an axis is configured and associated to a task. The association of axis to task is performed in the Axis icon.

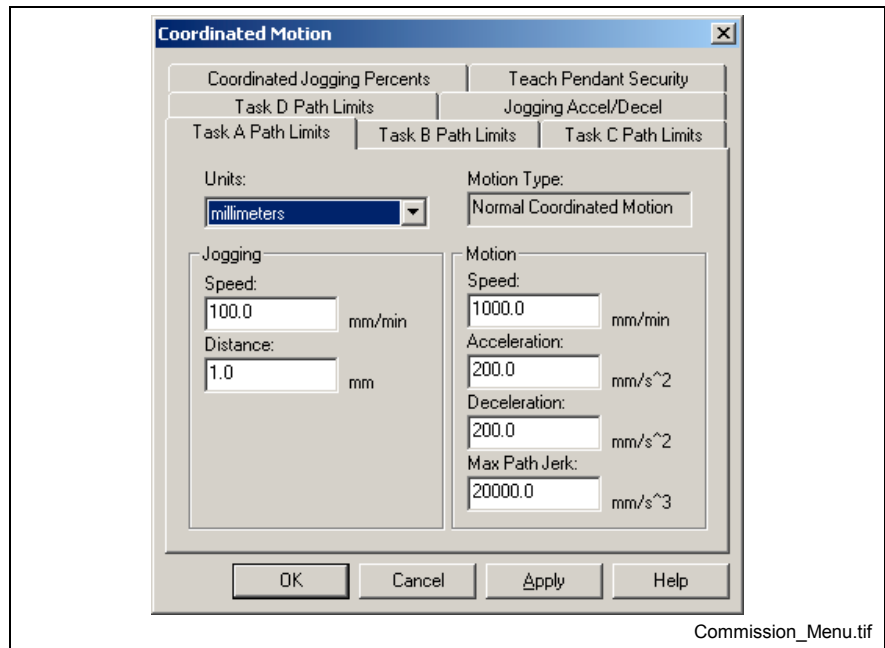


Fig. 2-86: Commission ⇒ Coordinated Motion

This window allows setting the units (inches, millimeters or radians) for the Path Limit values. It also permits setting the jogging speed and distance, setting the motion speed, and assigning separate values for acceleration and deceleration. The drive’s internal function uses the same acceleration/deceleration value for single-axis non-coordinated motion. The user can also set the Max Path Jerk for the task.

Clicking **OK** downloads the changed values to the control, without requiring the control to be in Parameter Mode.

Coordinated Jogging Percentage

This window allows the user to set the increments and velocities for fast and slow jogging for all configured coordinated axes.

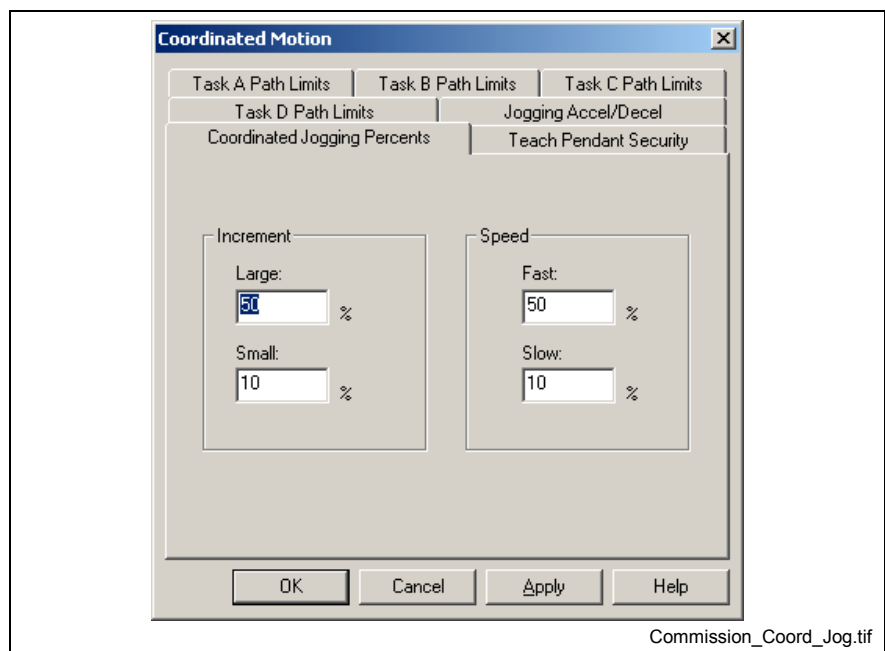


Fig. 2-87: “Coordinated Jogging Percents” Window

The **Increment** data area is used to set the **Large** and **Small** percentage of the maximum distance for a single-step jog operation. The maximum is defined by the axis parameter "Maximum Jog Increment" (T-0-0025). Similarly, the **Speed** data area is used to set the Fast and Slow jog speeds as a percentage of the maximum velocity, which is defined by the axis parameter "Maximum Jog Velocity" (T-0-0026). These values are stored in the following parameters:

- Large Increment (C-0-0052)
- Small Increment (C-0-0053)
- Fast Speed (C-0-0055)
- Slow Speed (C-0-0056)

Jogging Acceleration

This window allows the user to set the jogging acceleration and deceleration for each axis. Refer to Parameter A-0-0021 and A-0-0022 for more information.

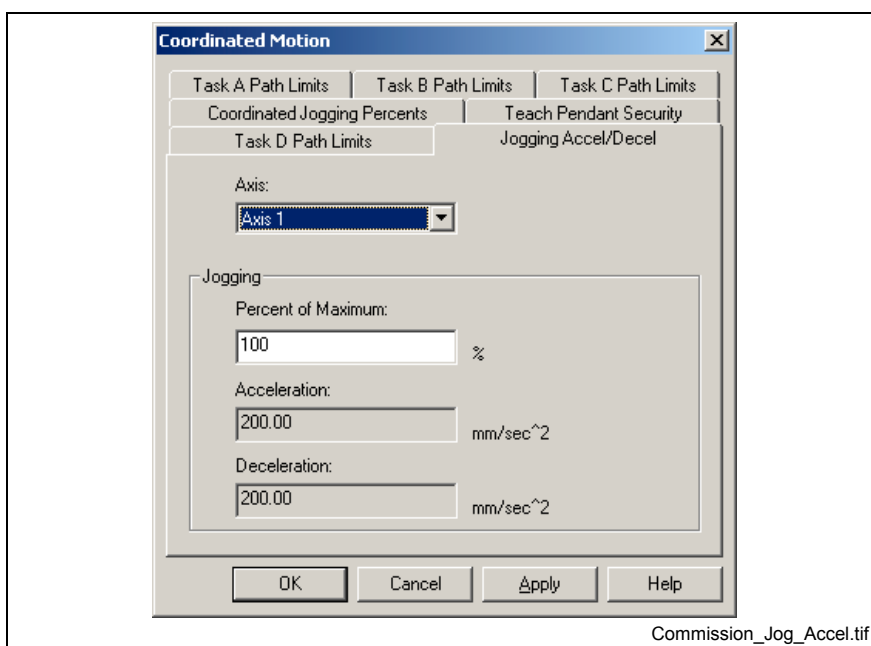


Fig. 2-88: "Jogging Accel/Decel Control" Window

Clicking **OK** downloads the changed values to the control, without requiring the control to be in Parameter Mode.

Pendant Security

This window allows selection of the level of user accessibility for variables and registers when using the BTC06:

- **Access Code** – sets the password (access code) that must be entered to gain access to the designated variables and registers.
- **Password Timeout** – restricts the time the user has to enter the password (in ms).
- **Float Variables Access** – sets user access to floats: None, All, or a range of specified floats.
- **Integer Variables Access** – sets user access to integers: None, All, or a range of specified integers
- **Register Control Access Level** - No Access to Protected Menus, Basic User Level Menu Access, Supervisor Level Menu Access, Full Menu Access
- **Register Control** – Enables Register Control Access Level.

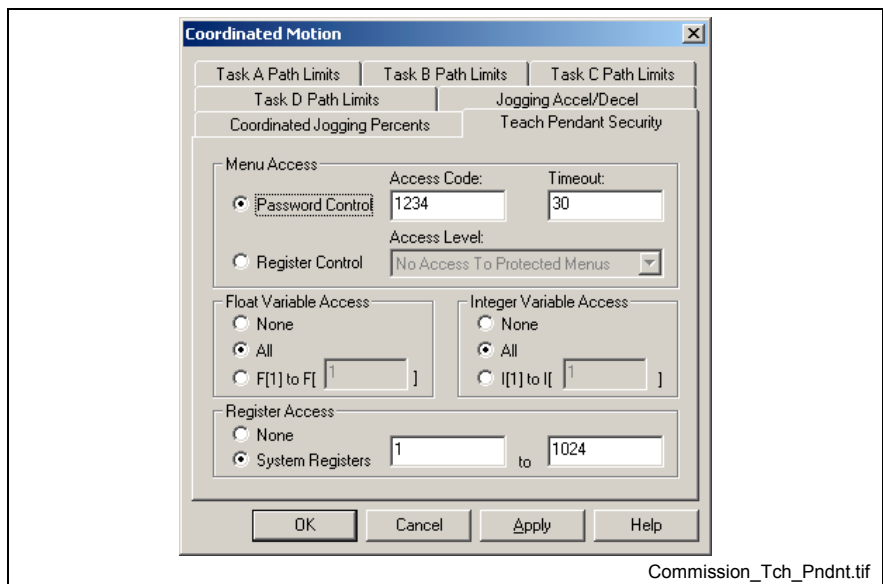




Fig. 2-89: “Teach Pendant Security” Window

Archive

Selecting **Commission** ⇒ **Archive** or clicking the Archive  icon opens the *VmArchive* window in Fig. 2-90.

This window provides backup and restore functionality of project data in online, offline or service modes. A full or selective backup or restore is possible. When project data is archived, a *Backup* folder, containing the PC's date, is created by default. The *Backup_date* folder is placed under a different location based on the current program mode, as listed in the table below. The browse  button allows the user to select a different directory location.

Mode	Archive Location
Online	... \project\project name\SaveSet\Online\Backup_current date
Offline	... \project\project name\SaveSet\Offline\Backup_current date
Service	... \project\project name\SaveSet\Backup_current date

Table 2-9: Archiving Folder Structure

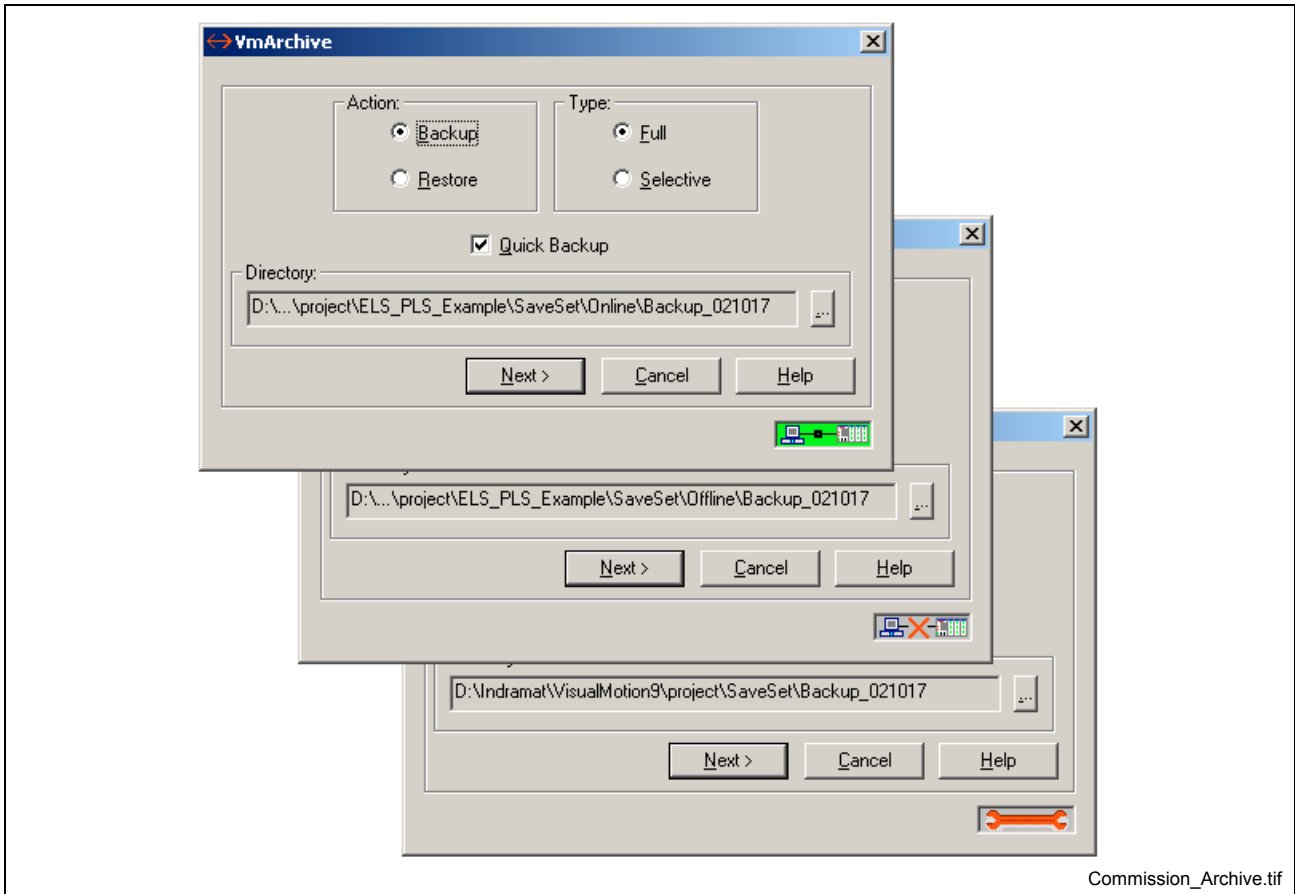


Fig. 2-90: "Archive" Window

Online Mode Archiving

An online archive saves all control and drive related data (parameters listed in C-0-2001, T-0-2001, A-0-2001 and S-0-0192) to the selected Online\SaveSet folder. The possible project components available for archiving are listed in Table 2-11.

Online Archive Data Structure

The following table lists the online archive data structure used by VisualMotion 9.

Data File	Description
Chklist.txt	Text file containing the hardware and firmware configurations and all components that were successfully archived.
Xferlog.txt	Text file log of all archives performed in a given Backup_date folder.
program.ex*	Compiled program file resident on control's memory. (.ex* - represents program #, i.e., .ex1, .ex2 and so on)
Control001.par	Control 1 parameter set (C-0-2001)
TaskA.par	Task A parameter set (T-0-2001)
Axis0**.par	Axis parameter set (A-0-2001) (0** - represents axis number up to a maximum of 40)
Drive0**.par	Both S and P drive parameter sets (S-0-0192) (0** - represents drive number up to a maximum of 40)
globalintegers.dat	program global integer data file
globalfloats.dat	program global float data file

Table 2-10: Online Archive Data Structure

Quick Backup

The Quick Backup checkbox determines which save format to use when creating an backup. By default, the Quick Backup checkbox is selected. Removing the check will generate a more detailed *.par backup file for control and drive parameters. Refer to the following examples:

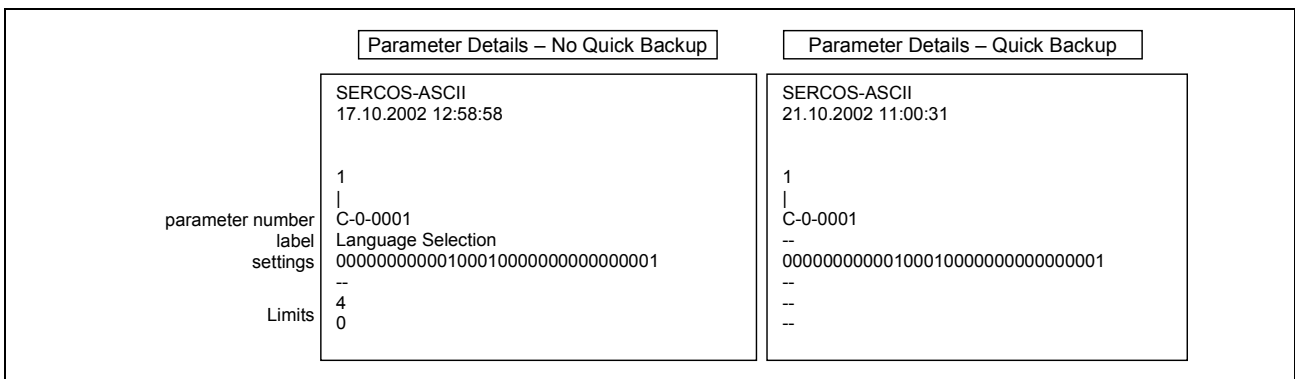


Fig. 2-91: Quick Backup Example

Online Full Backup

An online full backup will automatically select all the following project components and back them up either to the default folder location or to a user-defined folder.

Program (current or archived)	Global Variables	Axis Parameters
Control Parameters	Task Parameters	S/P Drive Parameters

Table 2-11: Full Online Backup

Note: If a full online backup was previously performed early in the day and a second backup is requested in the same day, a file replacement overwrite error will appear instructing the user to click on Yes to continue. If no backups were previously performed, then a dated folder is created under the online SaveSet folder and the files are backed up.

Online Full Restore An online full restore can only be performed if a valid dated backup folder is entered into the **Directory** field and it contains a valid Chklist.txt file. The system will read the Chklist.txt file and display a list of all the components that were originally backed up. Clicking on the **Start** button will begin the restore process. A message will appear warning the user that all programs on the control's memory will be lost.

Note: The control must in *Parameter* mode before a Full Online Restore can be performed.

Online Selective Backup An online selective backup allows the user to selectively choose which components to archive to the SaveSet folder. The *Selective Backup* window displays components for both the control and drive. The *Details* button towards the bottom of the window opens the *Archive Component Selection* window where the user can choose which details, of the selected component, to backup. By default, all items are selected. Unchecking an item will remove it for the list of items to backup.

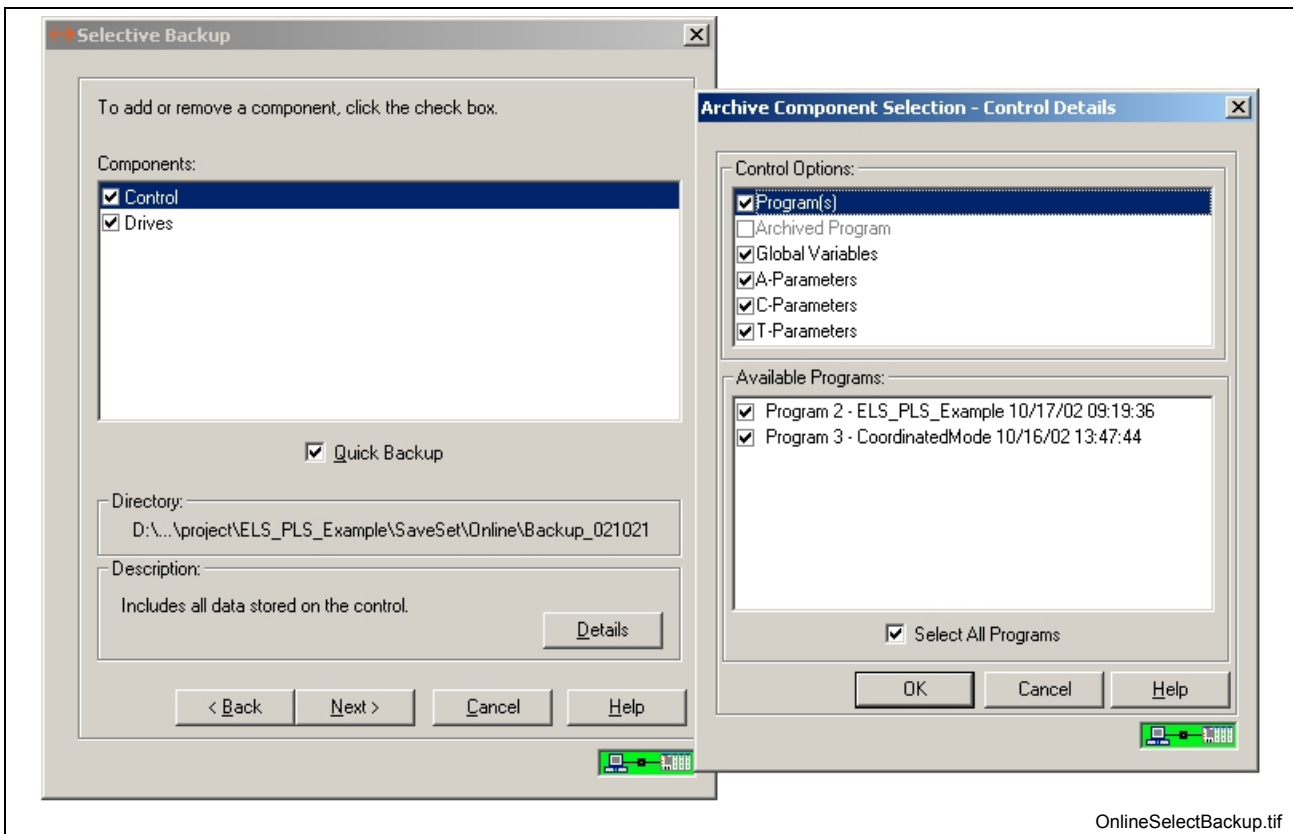


Fig. 2-92: Selective Online Backup

Online Selective Restore An online selective restore allows the user to selectively choose which components to restore from the Online\SaveSet folder to the control's memory. The *Selective Restore* window displays all the components available in the selected SaveSet folder. By default, all items are selected. Unchecking an item will remove it for the list of items to backup. The *Details* button towards the bottom of the window is only available for drive parameter files. This button opens the *Restore File Details* window where a user can restore a parameter file to the same drive number or any other drive available in the system.

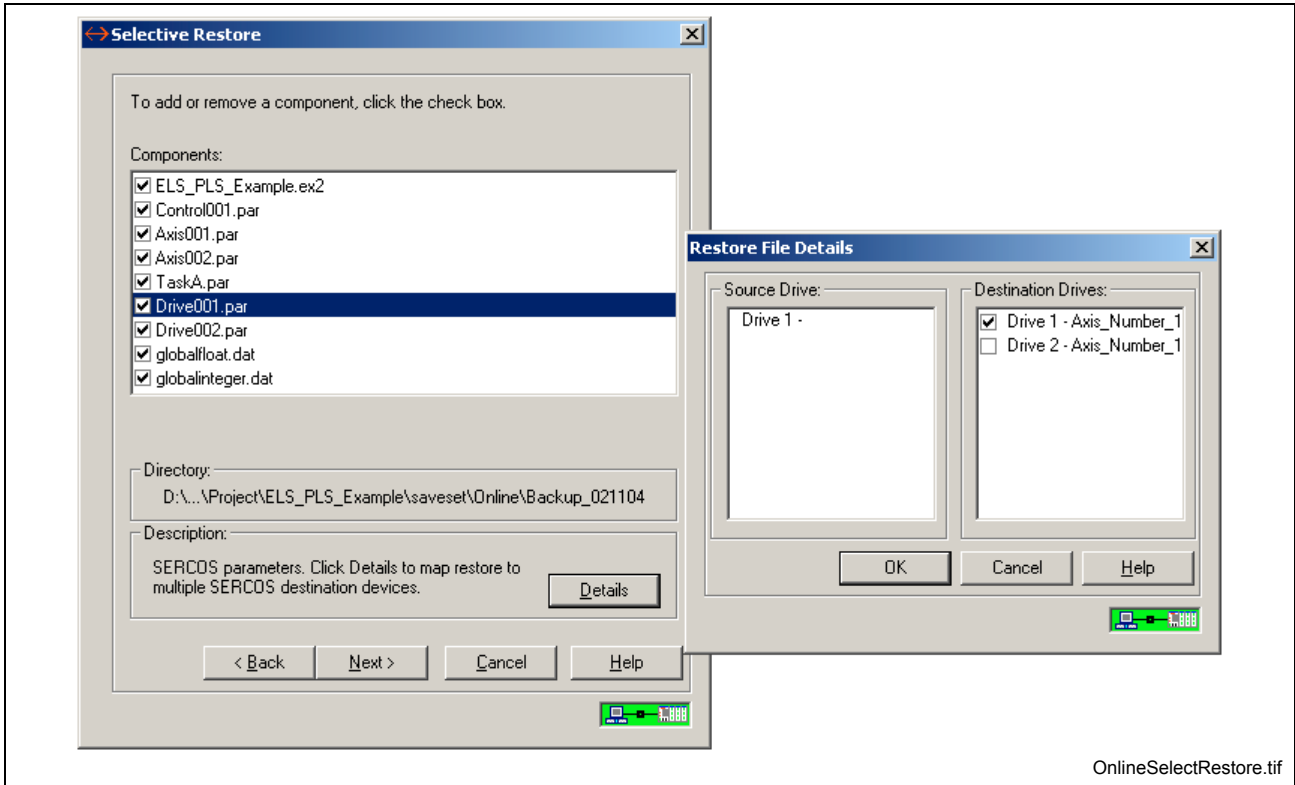


Fig. 2-93: Selective Online Restore

Offline Mode Archiving

Offline mode archiving is used to backup and restore files (excluding subfolders) located under the current project's main project folder and from the "...SaveSet\Offline\Backup" folder.

Note: Any file found under the project folder will be archived, whether or not it is used in the project.

Offline Full Backup An offline full backup automatically selects all the files (excluding subfolders) located under the main project folder of the currently opened project and backs them up to the "...SaveSet\Offline\Backup" folder.

Offline Full Restore An offline full restore copies all the files from the selected offline backup directory and overwrites the offline data of the current project opened in VisualMotion Toolkit. Clicking the **Start** button opens a message window warning the user that all current project files will be overwritten.

Note: After an offline restore, the icon editor reloads the new data into the project.

Offline Selective Backup An offline selective backup allows the user to selectively choose which files to backup to the "...SaveSet\Offline\Backup" folder. The *Selective Backup* window displays all the files located under the main project folder. The user can uncheck those files that are not part of the project before creating a backup.

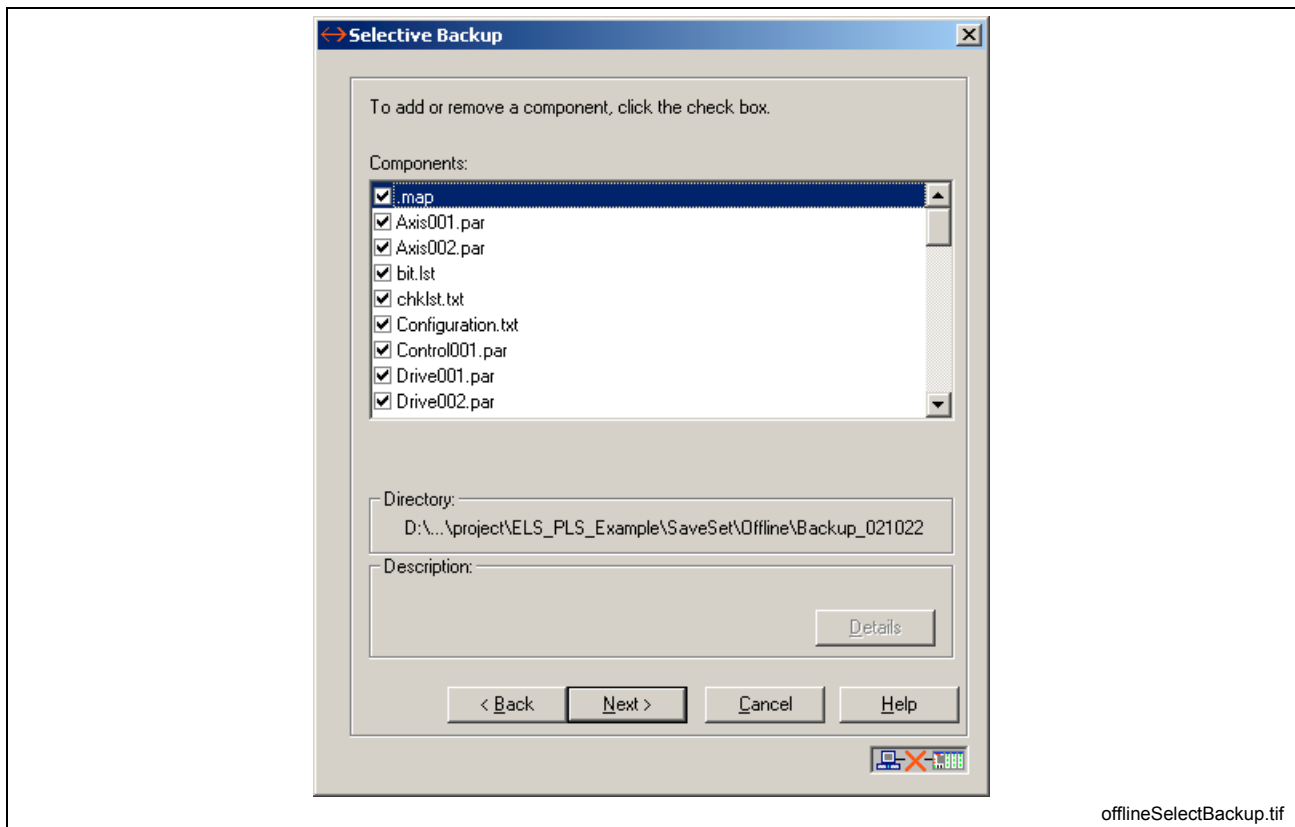


Fig. 2-94: Selective Offline Backup

Offline Selective Restore An offline selective restore allows the user to selectively choose which components to restore from the Offline\SaveSet folder to the current project. The *Selective Restore* window displays all the components available in the selected Offline\SaveSet folder. By default, all items are selected. Unchecking an item will remove it for the list of items to restore. The *Details* button towards the bottom of the window is only available for drive parameter files. This button opens the *Restore File Details* window where a user can restore a parameter file to any drive available in the current project.

Note: After an offline restore, the icon editor reloads the new data into the project.

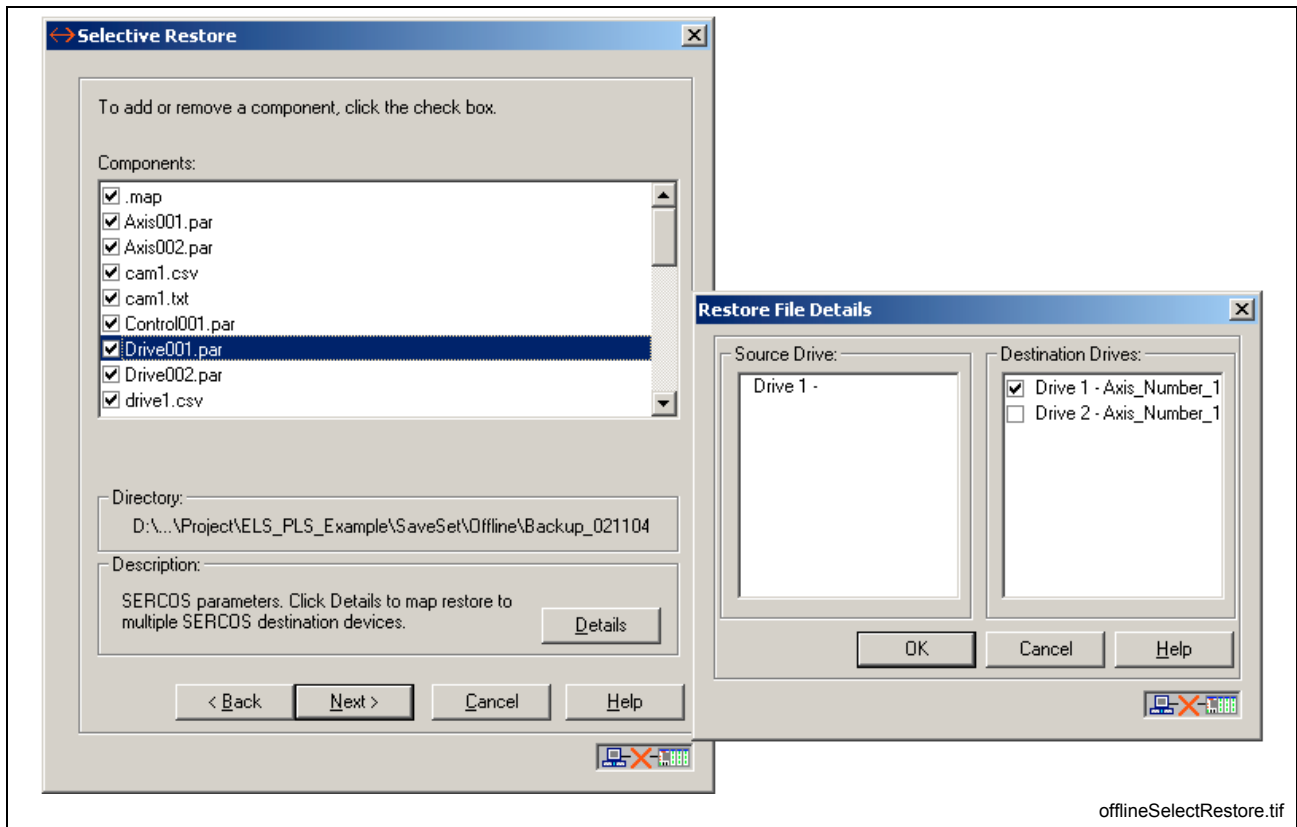


Fig. 2-95: Selective Offline Restore

Transfer

The Transfer command is used to transfer data between the control and the PC. Selecting **Commission** ⇒ **Transfer** opens a third menu level where the user can select the type of data to transfer. The Transfer menu selection is only available in service mode.

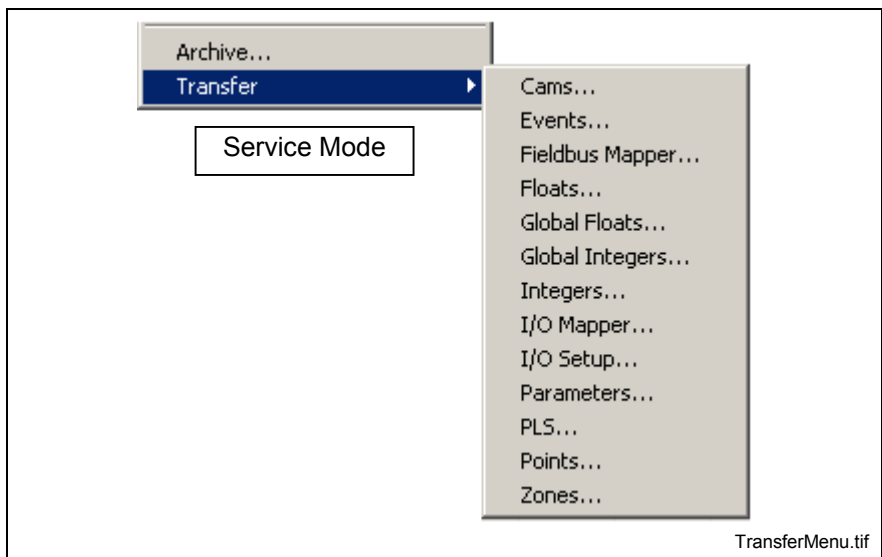


Fig. 2-96: The Transfer Menu

CAM

Selecting **Commission** ⇒ **Transfer** ⇒ **CAM...** opens the *Transfer CAM* window in Fig. 2-97. This window can be used to transfer existing CAM tables from the control or drive to a file, and from a file to the control or drive. CAM table files are stored with the ".csv" file extension in the project directory along with other project files.

Note: The ".csv" format is a standard used by Microsoft Excel and other spreadsheet software programs.

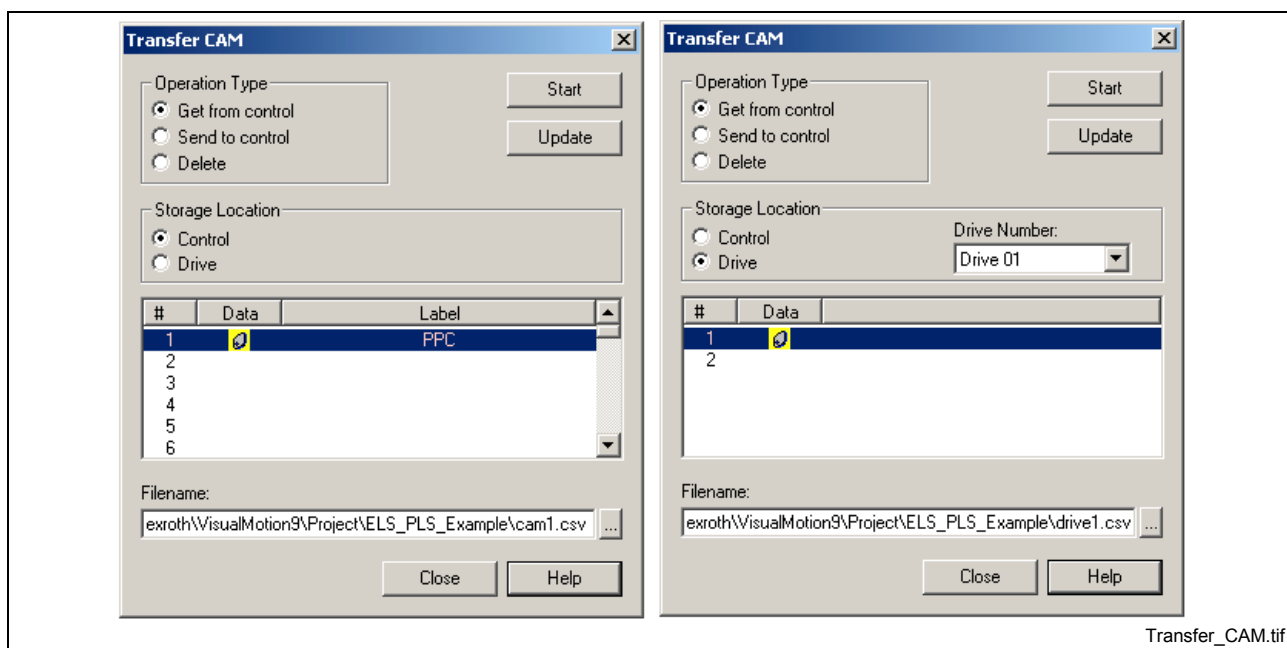


Fig. 2-97: Transfer CAM

Note: Before CAMs can be transferred, they must first exist in the control and/or drive. A CAM graphic under the **Data** column is an indication that the control or drive contains a CAM table. CAM tables are created and downloaded by selecting **Tools** ⇒ **CAM Builder** from VisualMotion Toolkit's main menu.

The *Transfer CAM* window allows access to built CAMs using the following operation types:

- **Get From Control** – retrieves the selected CAM number from the specified **Storage Location** (Control or Drive) on the control's memory.
- **Send to Control** – downloads the selected CAM table file to the control.
- **Delete** - deletes the selected CAM number from the specified storage location (Control or Drive) on the control's memory.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field and a CAM number is selected. Valid filenames for CAM tables have a “csv” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

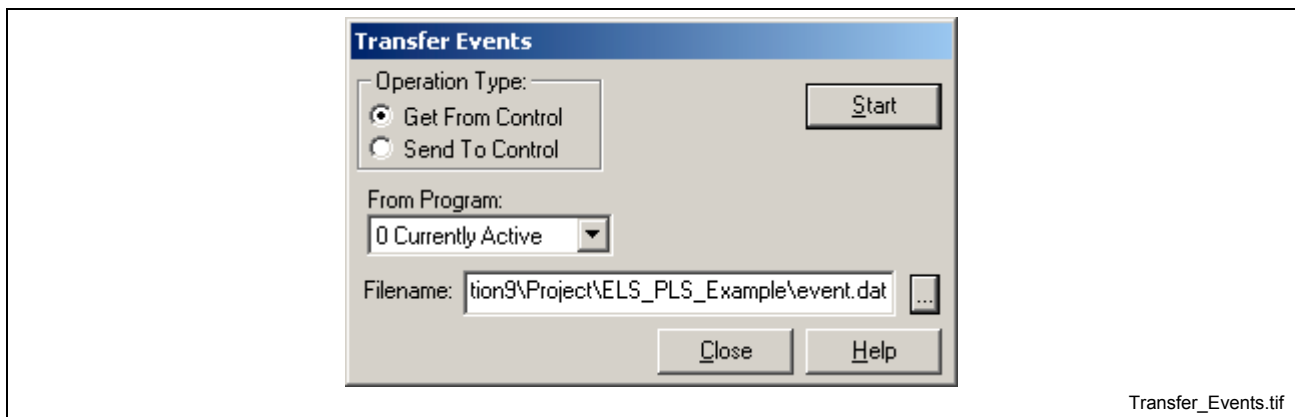
To refresh the list of CAM tables, click on the **Update** button.

There are 40 CAM tables possible on controls. Each drive in a system can have up to 2 CAM tables. These tables are dynamically allocated to conserve memory when not used.

CAM tables may be downloaded or deleted at any time as long as the CAM table number is not active. If the CAM is already active, the control responds with a communication error, “CAM is already active for axis ‘x’”. To download a new table, either switch into parameter mode or deactivate the CAM for all axes. (A CAM is active when it is assigned to an axis by the CAM icon.)

Events

Selecting **Commission** ⇒ **Transfer** ⇒ **Events** opens the following window:



Transfer_Events.tif

Fig. 2-98: Transfer Events

The *Transfer Events* window allows access to event tables using the following two operation types:

- **Get From Control** – retrieves and saves the events table as a data file “event.dat” from the selected program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “event.dat” to the selected program on the control.

Note: Use the **Send to Control** operation type to download older event table file formats “event.evt” to the selected program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for event tables have a “dat” or “evt” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Fieldbus Mapper

Selecting **Commission** ⇒ **Transfer** ⇒ **Fieldbus Mapper** opens the following window:

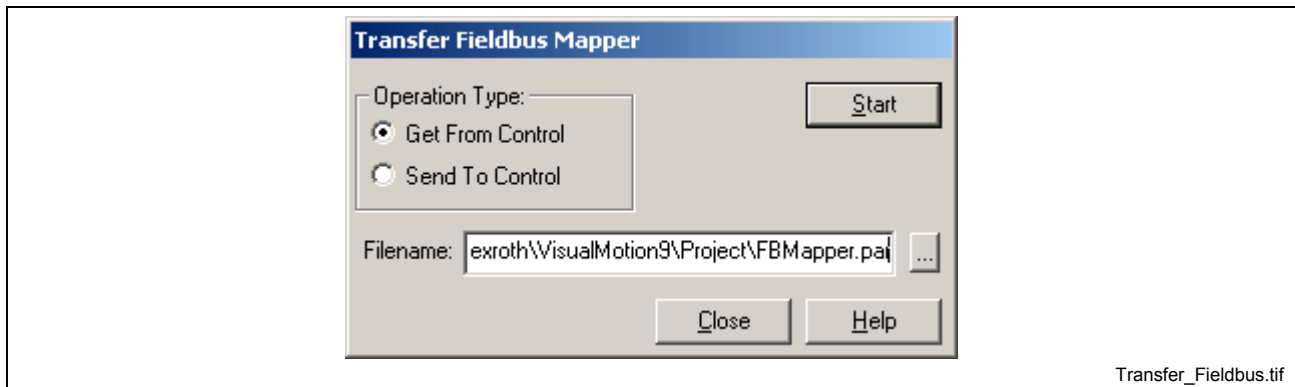


Fig. 2-99: Transfer Fieldbus Mapper

The *Transfer Fieldbus Mapper* window allows access to fieldbus mapper data using the following two operation types:

- **Get From Control** – retrieves and saves the fieldbus mapper as a parameter file “FBMapper.par” from the currently active program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “FBMapper.par” to the currently active program on the control.

Note: Use the **Send to Control** operation type to download older fieldbus mapper file formats “FBMapper.prm” to the currently active program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for a fieldbus mapper have a “par” or “prm” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Floats

Selecting **Commission** ⇒ **Transfer** ⇒ **Floats** opens the following window:

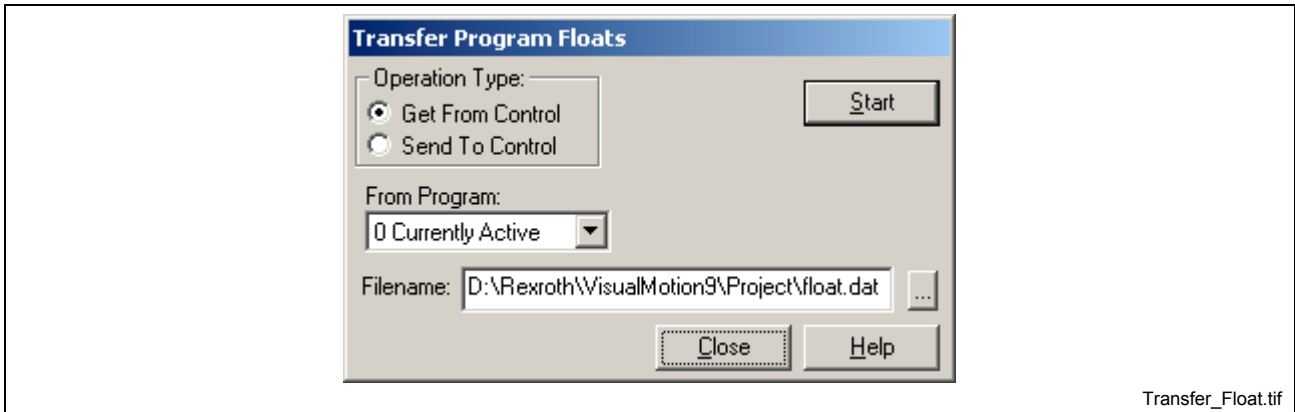


Fig. 2-100: Transfer Program Floats

The *Transfer Program Floats* window allows access to program floats using the following two operation types:

- **Get From Control** – retrieves and saves the program floats as a data file “float.dat” from the selected program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “float.dat” to the selected program on the control.

Note: Use the **Send to Control** operation type to download older float file formats “float.vtr” to the selected program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for floats have a “dat” or “vtr” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Global Floats

Selecting **Commission** ⇒ **Transfer** ⇒ **Global Floats** opens the following window:

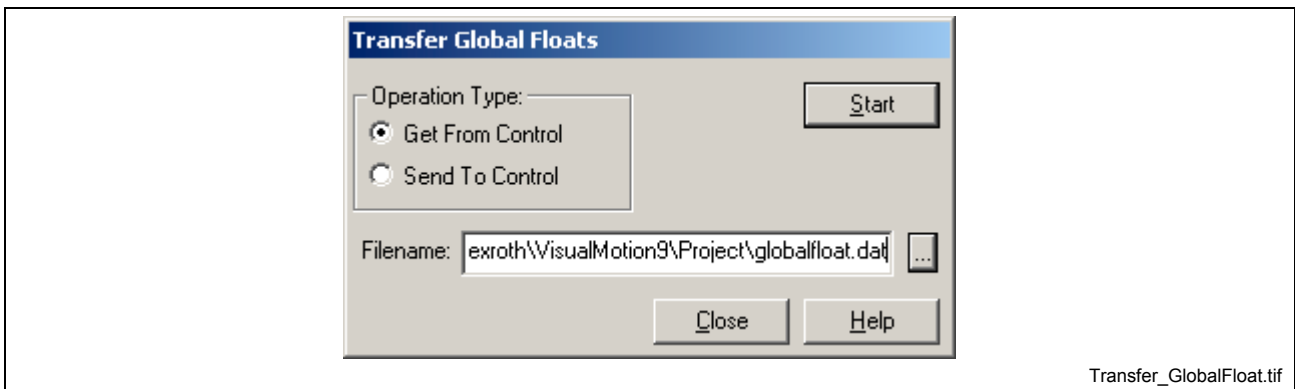


Fig. 2-101: Transfer Global Floats

The *Transfer Global Floats* window allows access to program floats using the following two operation types:

- **Get From Control** – retrieves and saves the global floats as a data file “globalfloat.dat” from the control’s memory to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “globalfloat.dat” to the control’s memory.

Note: Use the **Send to Control** operation type to download older global float file formats “gloabalfloat.vtr” to the control’s memory.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for global floats have a “dat” or “vtr” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Global Integers

Selecting **Commission** ⇒ **Transfer** ⇒ **Global Integers** opens the following window:

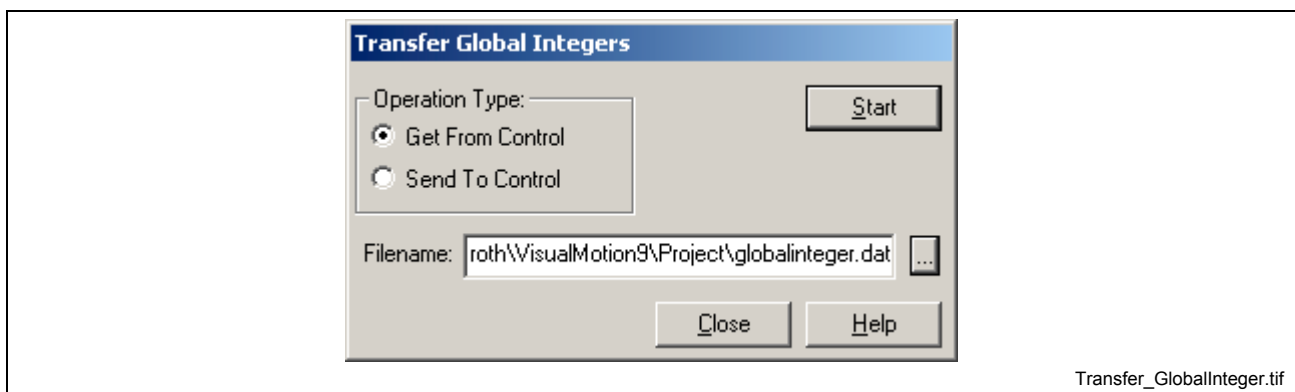


Fig. 2-102: Transfer Global Integers

The *Transfer Global Integers* window allows access to program floats using the following two operation types:

- **Get From Control** – retrieves and saves the global integers as a data file “globalinteger.dat” from the control’s memory to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “globalinteger.dat” to the control’s memory.

Note: Use the **Send to Control** operation type to download older global integer file formats “globalinteger.vtr” to the control’s memory.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for global integers have a “dat” or “vtr” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Integers

Selecting **Commission** ⇒ **Transfer** ⇒ **Integers** opens the following window:

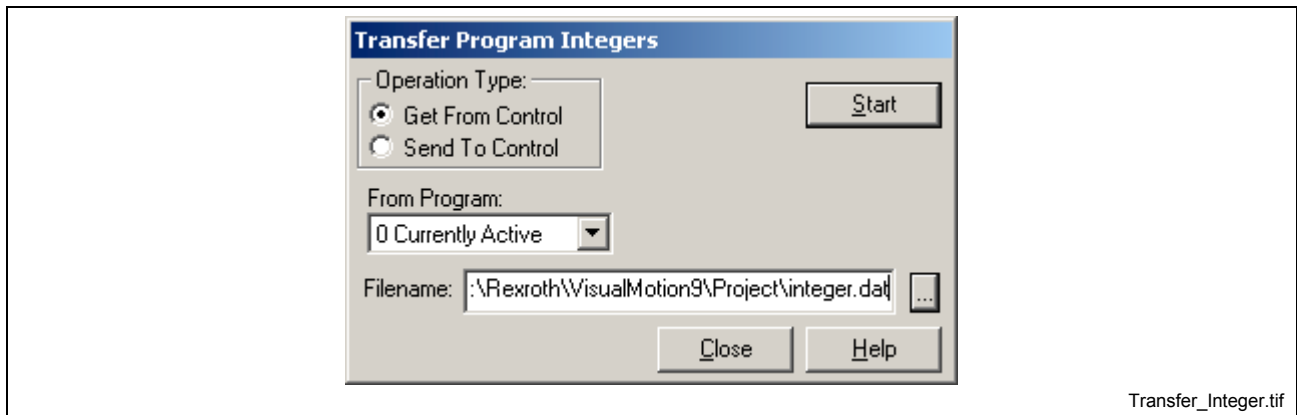


Fig. 2-103: Transfer Program Integers

The *Transfer Program Integers* window allows access to program integers using the following two operation types:

- **Get From Control** – retrieves and saves the program integers as a data file “integer.dat” from the selected program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “integer.dat” to the selected program on the control.

Note: Use the **Send to Control** operation type to download older program integer file formats “integer.vtr” to the selected program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for integers have a “dat” or “vtr” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

I/O Mapper

Selecting **Commission** ⇒ **Transfer** ⇒ **I/O Mapper** opens the following window:

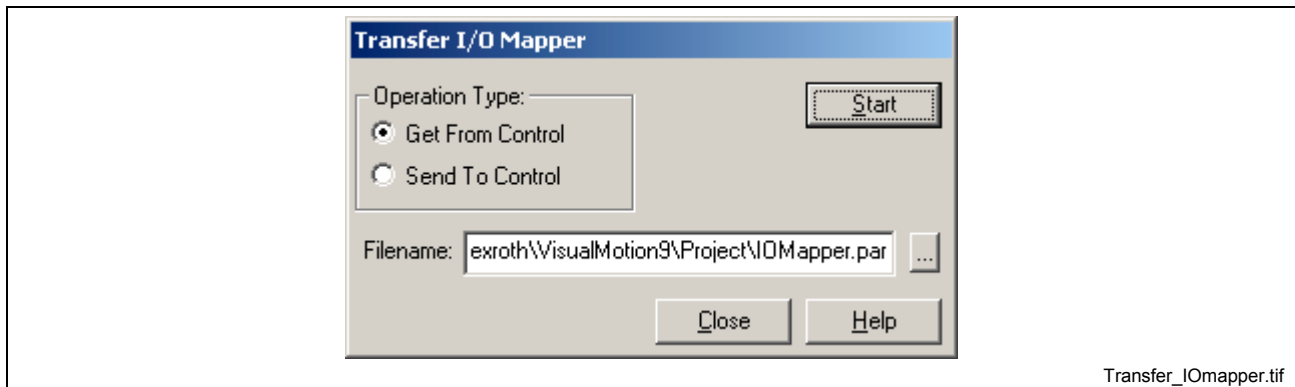


Fig. 2-104: Transfer I/O Mapper

The *Transfer I/O Mapper* window allows access to an I/O Mapper using the following two operation types:

- **Get From Control** – retrieves and saves the I/O Mapper as a parameter file “IOMapper.par” from the currently active program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “IOMapper.par” to the currently active program on the control.

Note: Use the **Send to Control** operation type to download older I/O Mapper file formats “IOMapper.iom” to the currently active program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for I/O Mapper have a “par” or “iom” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

I/O Setup

Selecting **Commission** ⇒ **Transfer** ⇒ **I/O Setup** opens the following window:

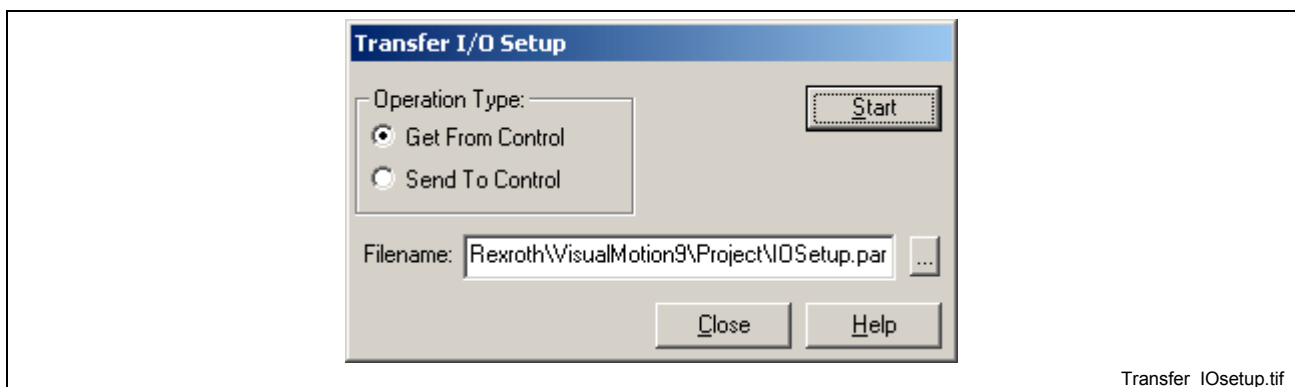


Fig. 2-105: Transfer I/O Setup

The *Transfer I/O Setup* window allows access to the I/O Configuration using the following two operation types:

- **Get From Control** – retrieves and saves the I/O Setup as a parameter file “IOSetup.par” from the currently active program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “IOSetup.par” to the currently active program on the control.

Note: Use the **Send to Control** operation type to download older I/O Setup file formats “IOSetup.prm” to the currently active program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for I/O Setup have a “par” or “prm” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Parameters

Selecting **Commission** ⇒ **Transfer** ⇒ **Parameters** opens the following window:

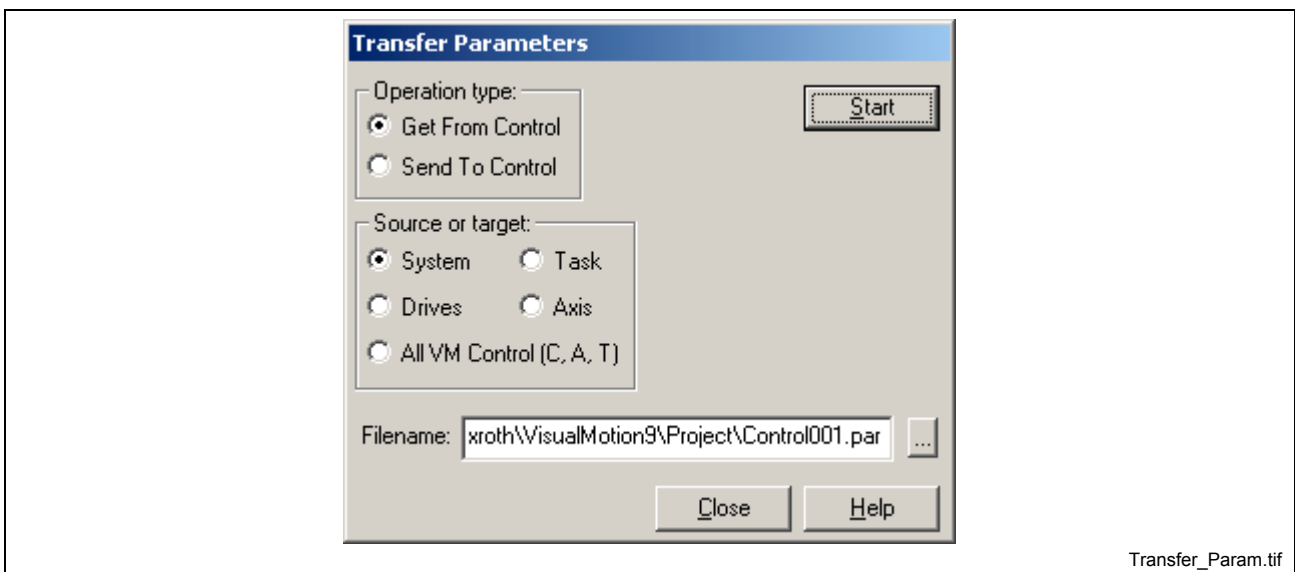


Fig. 2-106: Transfer Parameters

The *Parameters* window allows access to parameters using the following two operation types:

- **Get From Control** - saves the selected source type to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected filename type to the control.

Note: Use the **Send to Control** operation type to download older parameter file formats “*.prm” to the selected **Source or Target**.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for parameters have a “par” or “prm” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

All VM Control (C, A, T) parameters can be transferred at one time, or individually. In addition, the parameter set for a selected drive that is connected to the control can be transferred through the SERCOS communication system. The following table lists the different file types available:

Source or Target	Filename	Description
System	Control001.par	Parameters listed in C-0-2001
Task	Task*.par	Parameters listed in T-0-2001 where * = A, B, C or D
Axis	Axis0**.par	Parameters listed in A-0-2001 where ** = axis number up to 40
Drives	Drive0**.par	Parameters listed in S-0-0192 where ** = drive number up to 40

Table 2-12: Available Parameters for Transfer

PLS

Selecting **Commission** ⇒ **Transfer** ⇒ **PLS** opens the Transfer PLS window in Fig. 2-97. This window can be used to transfer existing Option Card and Drive PLSs from the control or drive to a file, and from a file to the control or drive. Option Card and Drive PLS files are stored as parameters using the “.par” file extension.

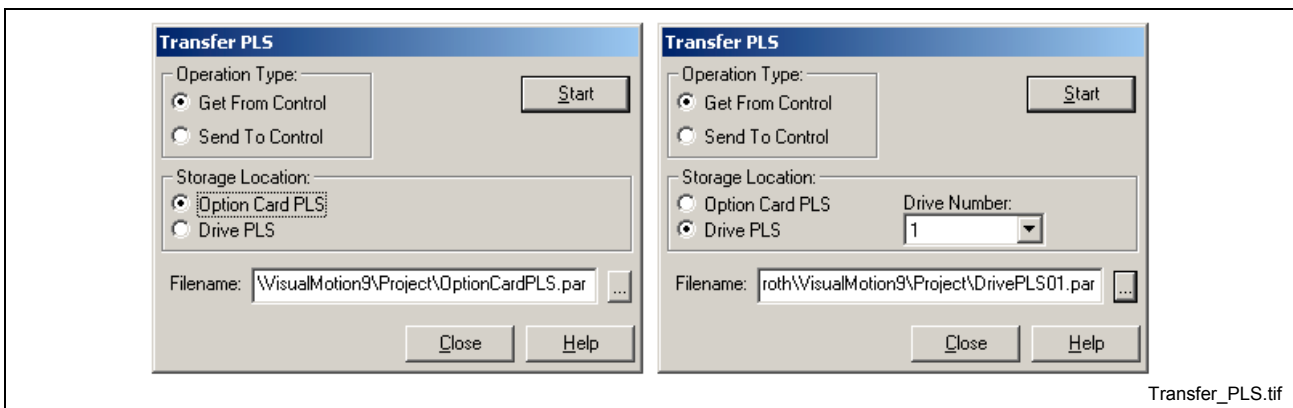


Fig. 2-107: Transfer PLS

Note: Before Option Card or Drive PLS files can be transferred, they must first exist in the control and/or drive. Option Card and Drive PLSs are created and downloaded by selecting **Commission** ⇒ **PLS** from VisualMotion Toolkit’s main menu.

The *Transfer PLS* window allows access to existing PLS configurations using the following two operation types:

- **Get From Control** – retrieves and saves the selected Option Card or Drive PLS file “OptionCardPLS.par” from the specified storage location to the location entered in the **Filename** field.

- **Send to Control** – downloads the selected Option Card or Drive PLS file to the control.

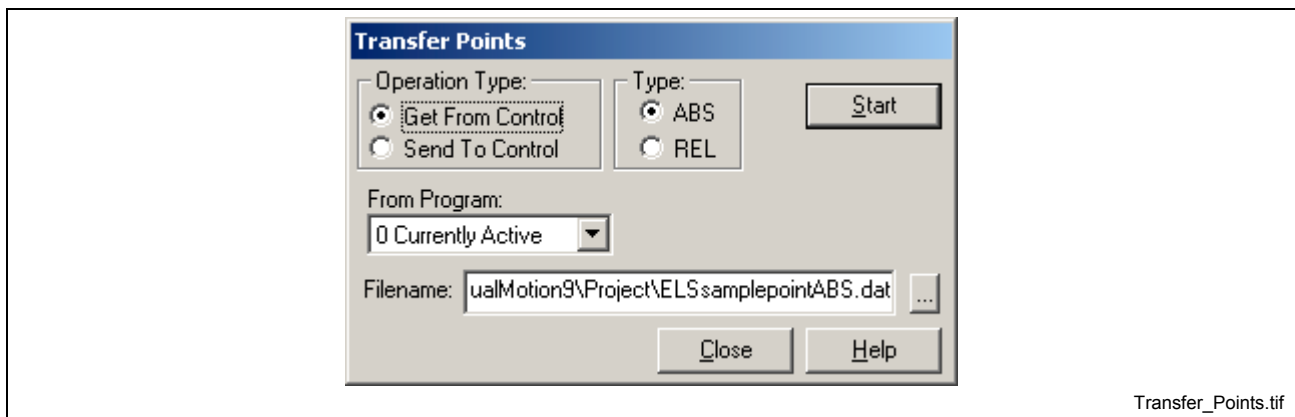
Note: Use the **Send to Control** operation type to download older Option Card or Drive PLS file formats “OptionCardPLS.prm” to the currently active program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for PLS have a “par” or “prm” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Points

Selecting **Commission** ⇒ **Transfer** ⇒ **Points** opens the following window:



Transfer_Points.tif

Fig. 2-108: Transfer Points

The *Transfer Points* window allows access to point tables using the following two operation types:

- **Get From Control** – retrieves and saves the point tables as a data file “pointABS.dat” and/or “pointREL.dat” from the selected program to the location entered in the **Filename** field.

Note: If the selected program contains both relative and absolute point tables, a data file for each type will be created when using the **Get From Control** operation type.

- **Send to Control** – downloads the selected file “pointABS.dat” or “pointREF.dat” to the selected program on the control. Relative and absolute point data files must be sent to the control, one at a time.

Note: Use the **Send to Control** operation type to download older relative and absolute file formats “*.abs” or “*.rel” to the selected program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for relative or absolute points have a “dat” , “rel” or “abs” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

Zones

Selecting **Commission** ⇒ **Transfer** ⇒ **Zones** opens the following window:

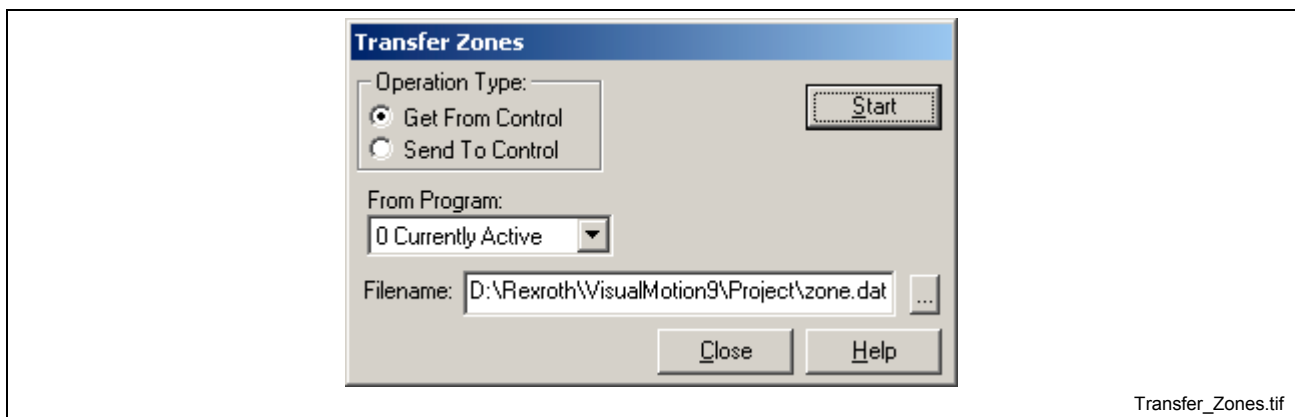


Fig. 2-109: Transfer Zones

The *Transfer Zones* window allows access to the zone table using the following two operation types:

- **Get From Control** – retrieves and saves the zone table as a data file “zone.dat” from the selected program to the location entered in the **Filename** field.
- **Send to Control** – downloads the selected file “zone.dat” to the selected program on the control.

Note: Use the **Send to Control** operation type to download older zone file formats “zone.zon” to the selected program.

Start Button Activation

The **Start** button becomes active only after a valid filename is entered in the **Filename** field. Valid filenames for zones have a “dat” or “zon” extension. Enter a filename, with a valid extension, directly in the **Filename** field or click the browse button, locate a directory and then specify a filename.

2.8 The Data Menu

The **Data** menu provides the user access to VisualMotion system parameters, registers, variables, events, points and zones. In addition to system data, the user can access runtime utilities such as CAM Indexer, ELS, PID and Registration.

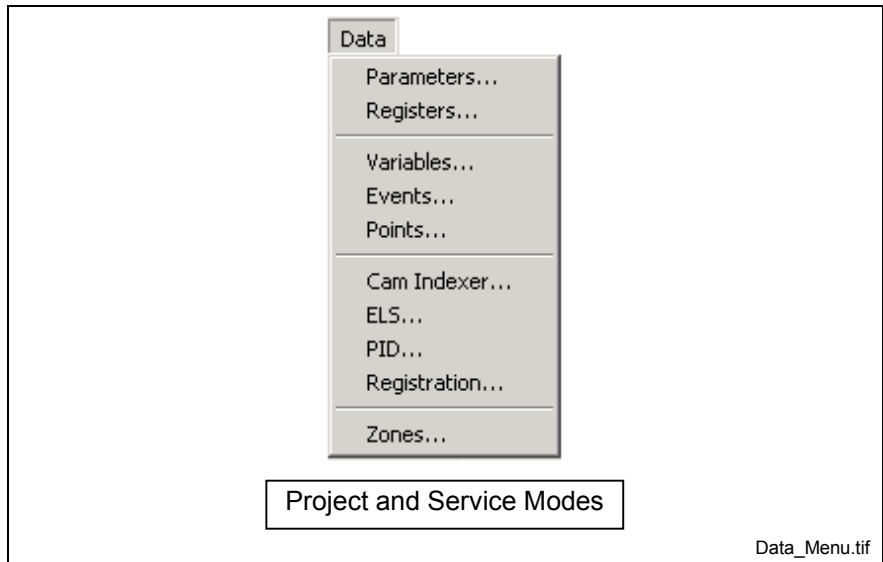


Fig. 2-110: The Data Menu

Parameters

Selecting **Data** ⇒ **Parameters** opens the *Parameter Overview Runtime Tool* window.

Note: VisualMotion parameters can only be viewed and/or edited in either service or online mode.

This tool is used to view and modify existing Control, Task, Axis and SERCOS device parameters. The user can also create and edit a configurable parameter list called Custom list. The following parameter types are displayed in an expandable tree structure, similar to the folder (directory) structure found in Windows™:

Parameter Type	Description
• Control	All control specific parameters are displayed when Control is selected.
• Task	All parameters for the selected task are displayed when Task (A, B, C or D) is selected.
• Axis	All parameters for the selected axis are displayed when Axis # (up to 40) is selected.
• SERCOS	All parameters for the selected SERCOS digital drive, up to a maximum of 40, and SERCOS I/O devices are listed.
• Custom	Any existing custom list created by the user.

Table 2-13: Parameter Types

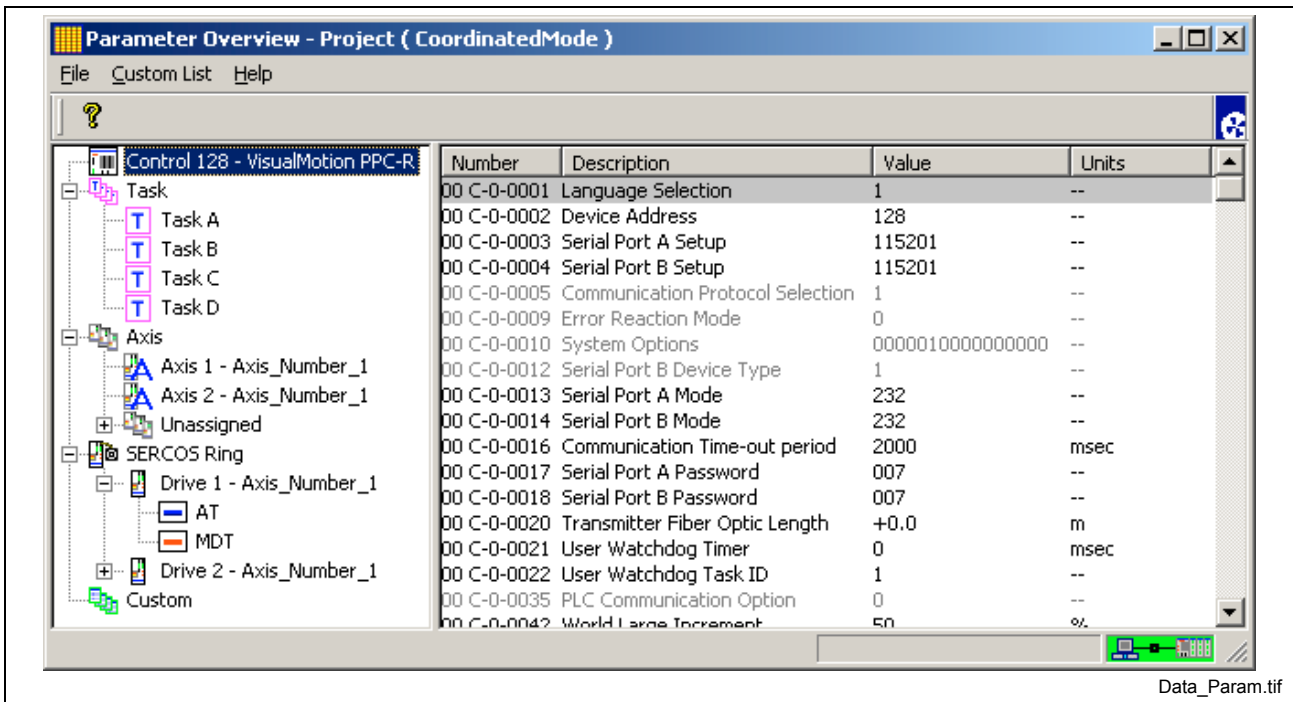


Fig. 2-111: Parameter Overview Window

Parameter Access

Parameter "read" or "write" access is identified by the text color displayed in the parameter overview window. The following table indicates the color code / access combination.

Color Code	Description	Access
grayed out text	read-only parameter or not editable in current phase	read-only
black text	parameter that can be edited	read/write
red text	used to indicate an error	read/write
blue text parameter list	parameter list, denoted by Xs, can be edited	read/write
grayed out parameter list	read-only parameter list, denoted by Xs, or not editable in current phase	read-only

Table 2-14: Parameter Access

Configuring SERCOS Drive AT and MDT

The *Parameter Overview* tool contains a SERCOS Telegram Tool used to configure each drive's SERCOS Drive Telegram (AT) and Master Data Telegram (MDT). VisualMotion's Telegram Tool provides the user with a convenient and comprehensive interface for viewing the AT and MDT and modifying selected portions of each telegram.

The AT and MDT are used to cyclically exchange data between drives and control every SERCOS cycle. The AT is sent from each drive to the control and the MDT is sent from the control to each drive on the system. The AT and MDT are comprised of various parameters stored in each drive. Some parameters displayed in the AT and MDT are automatically configured based on the system's primary and secondary modes of operation. These parameters appear as gray text. User configurable areas are displayed in black text.

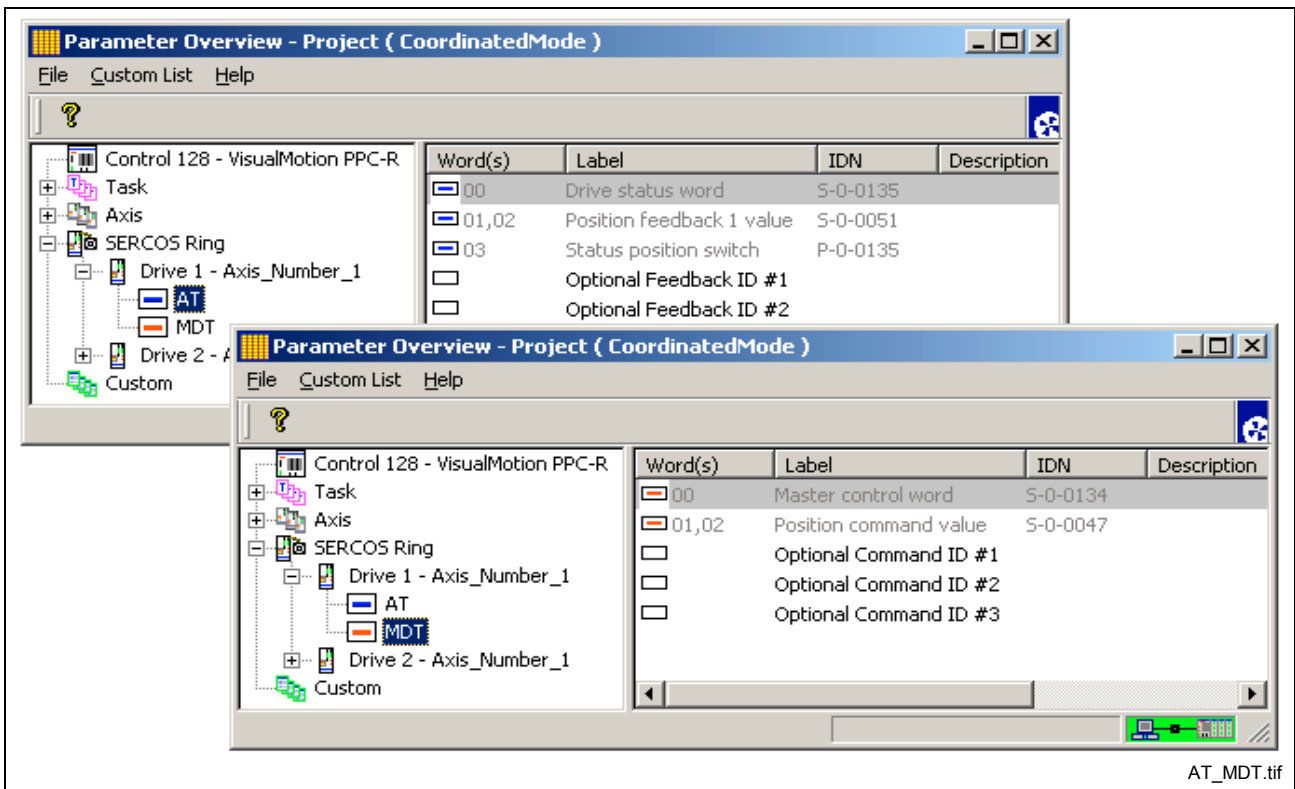


Fig. 2-112: SERCOS Drive AT and MDT

Refer to *Chapter 5, SERCOS Drive Telegram Tool*, for details.

Edit a Parameter

To edit a parameter, select the parameter type (Control, Task, etc.), and then double click on the desired parameter number from the parameter overview window.

Parameter Help System not Found

VisualMotion issues an error message when help is requested for a help system that is not installed on your PC. Some reasons why this may occur are as follows:

- The help files were moved to a different folder.
- The help files were not installed or deleted from the computer.
- The system language was changed and the help files do not exist for the selected language.

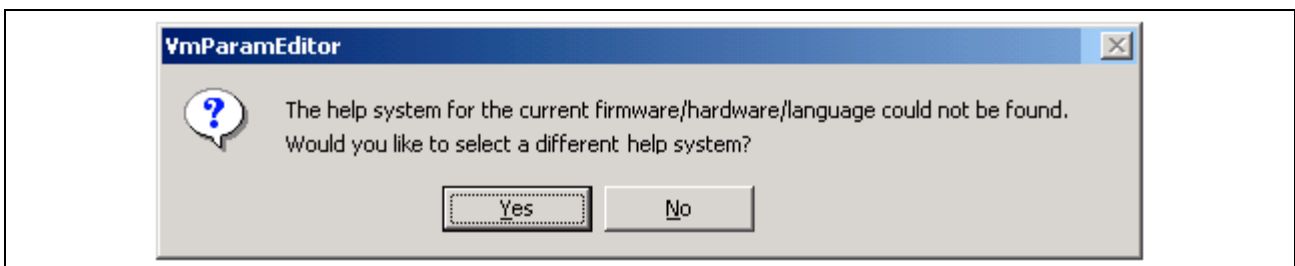


Fig. 2-113: Help System Error Message

Note: This error is typically encountered when help for SERCOS drive parameter is requested. Each drive firmware has its own help system that can be installed. Drive help is available on the following DriveHelp CD with material number 282411:
DOK-GENERL-DRIVEHELP**-GN07-MS-D0600

When the **Yes** button is pressed, the following window will assist the user in locating a suitable help system.

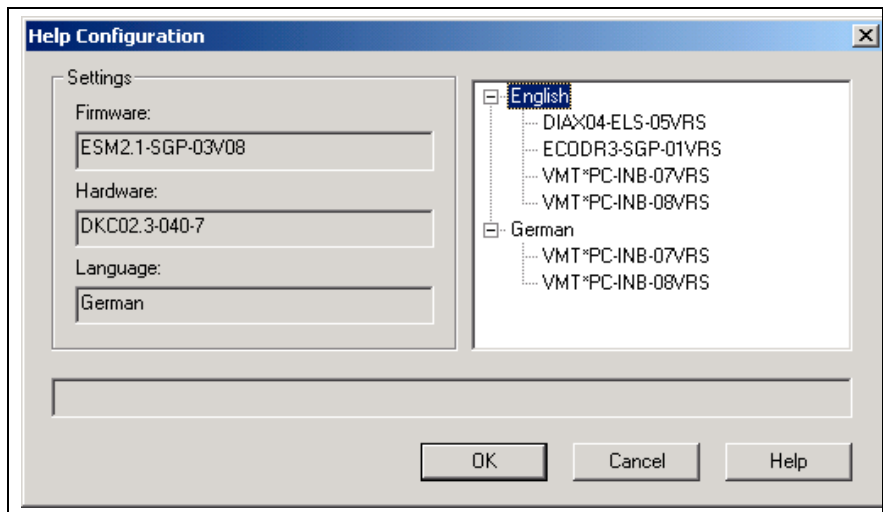


Fig. 2-114: Help System Location Window

Edit a Standard Parameter

A standard parameter can be an Integer, Float, String or Hex value. The standard parameter edit window in the figure below is displayed when editing a standard parameter. The current data **Limits** for a parameter are displayed above the input field, from minimum to maximum values.

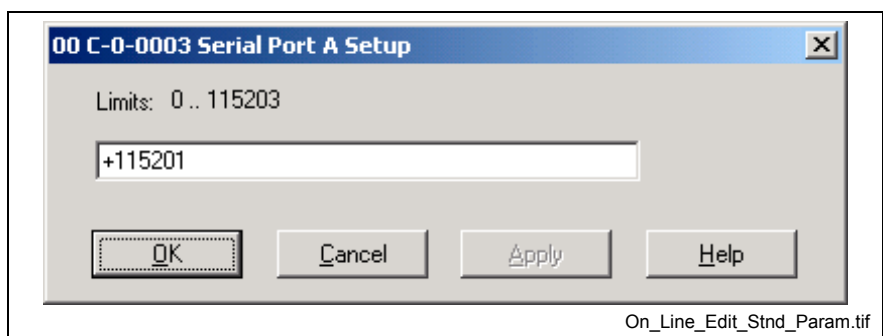


Fig. 2-115: Parameter Edit Window

Edit a Binary Parameter

The binary parameter edit window in the figure below is modified by clicking on the desired bit(s). Holding the mouse cursor over a bit will display a tool tip containing the bit number.

Note: 16 bit parameters will only have the first 16 bits accessible. Bits 17-32 are grayed out.

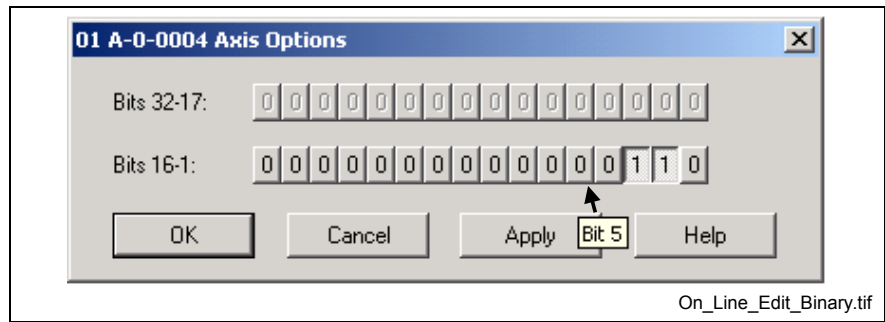


Fig. 2-116: Binary Parameter Edit Window

Edit a Parameter from a Predefined List

This parameter edit window uses a drop down list to selected parameters that have been predefined as valid selections. To edit this parameter, the user selects the desired parameter from the drop down list.

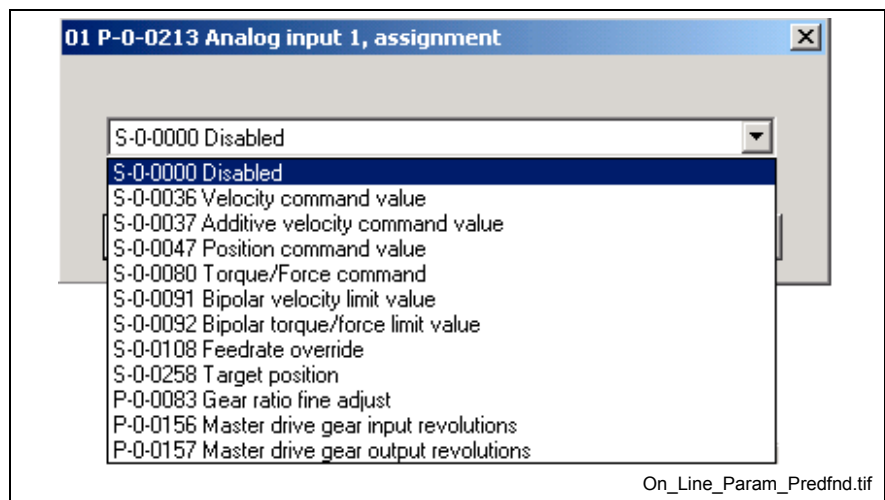


Fig. 2-117: Predefined Parameter List Edit Window

Edit, Refresh or Find a Parameter or List

Right clicking on a selected parameter opens a popup window where the user can **Edit Selection (ENTER)**. Selecting the **Refresh (F5)** option updates all the parameters visibly displayed in the main window. The **Find (F3)** option allows the user to locate a parameter by using a partial description.

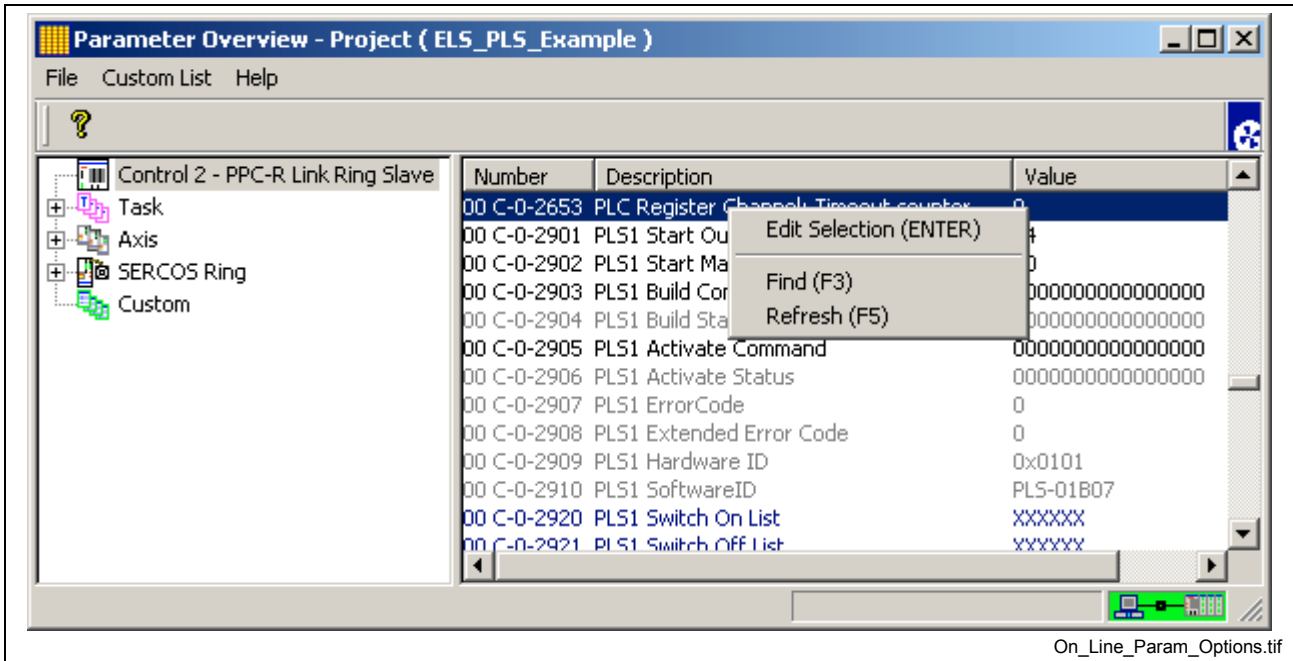


Fig. 2-118: Parameter Options

Note: Parameters can also be edited by double clicking the left mouse button.

Display a Parameter List

The information contained in Parameter lists can be displayed in one of the two following formats:

- Data Format – list of data
- IDN Format – list of parameter numbers

Parameter lists are displayed in either blue text (read/write access) or gray text (read only) with the value column displaying six Xs. To display a parameter list, double click on the desired parameter and VisualMotion will open a new window. The new window will display the parameter number and description in the window's header and display the contents in either Data format or IDN format.

Data Format

Parameter lists displayed in data format will contain an Index and a Value column as shown in the figure below.

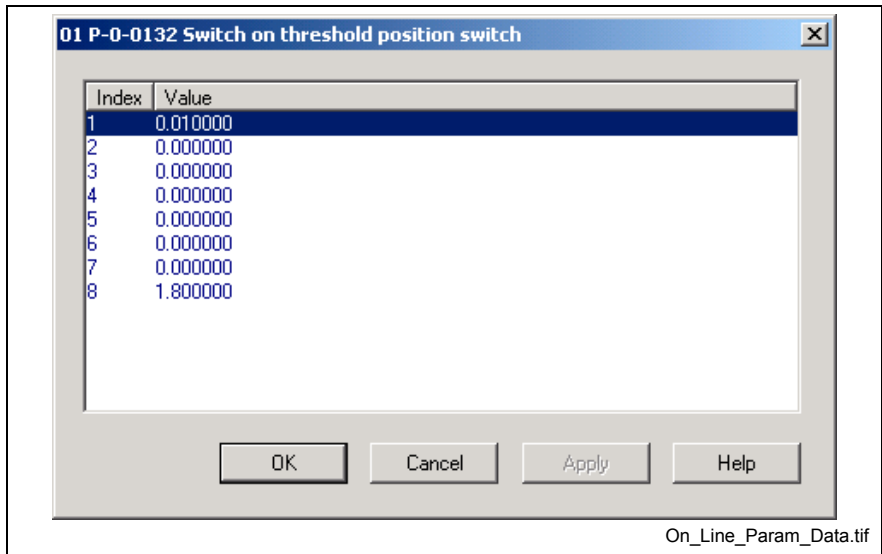


Fig. 2-119: Parameter List Data Format

IDN Format

Parameter lists displayed in IDN (IDentification Number) format is a list of other parameters containing an index, parameter number and description as shown in the figure below.

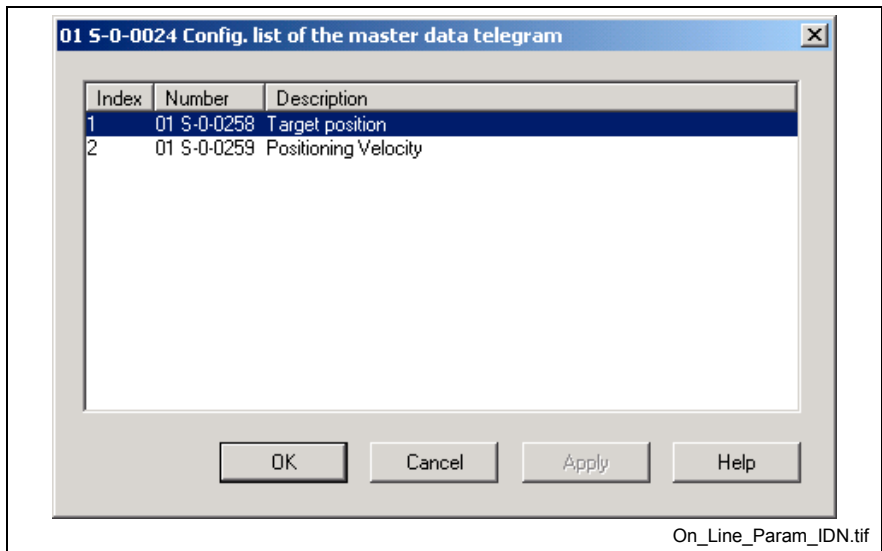


Fig. 2-120: Parameter List IDN Format

Append, Insert and Delete within a Parameter List

Once a parameter list is opened, right clicking anywhere in the window opens a popup window where the user can perform the following functions.

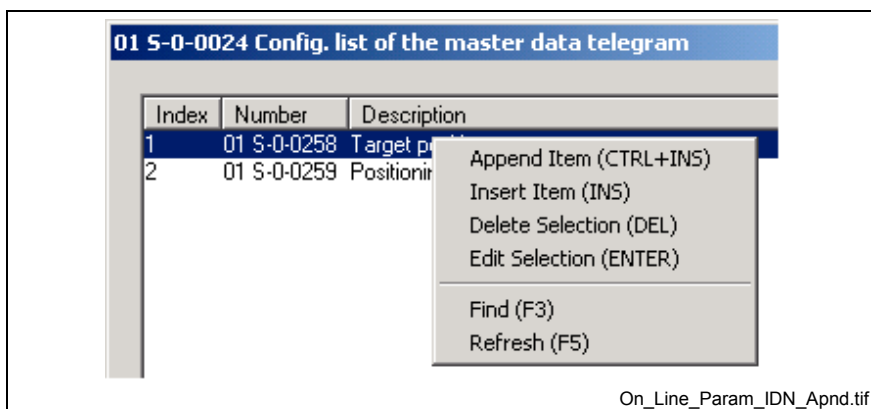


Fig. 2-121: Parameter List Options

Note: Not all parameter lists allow the addition and removal of parameters.

- **Append Item (CTRL+INS)** – adds a new item to the end of the current list.
- **Insert Item (INS)** – inserts an item above the selected value or parameter.
- **Delete Selection (DEL)** – deletes the selected data or parameter from the list.
- **Edit Selection (ENTER)** – opens an edit window where the data or parameter value can be edited.
- **Refresh (F5)** – refreshes all values displayed in the parameter list.
- **Find (F3)** – locates a parameter using the number or description.

Adding Predefined Parameters to a List

The Append and Insert functions within a list can be used to add predefined parameters to certain lists. To add a parameter from a list, right click in the window and select Append or Insert. A parameter selection edit window will open. Next, select the desired parameter from the drop down list. Repeat the process as necessary.

Note: Only certain drive parameters contain a listing of valid parameters. In other cases, the data is manually entered.

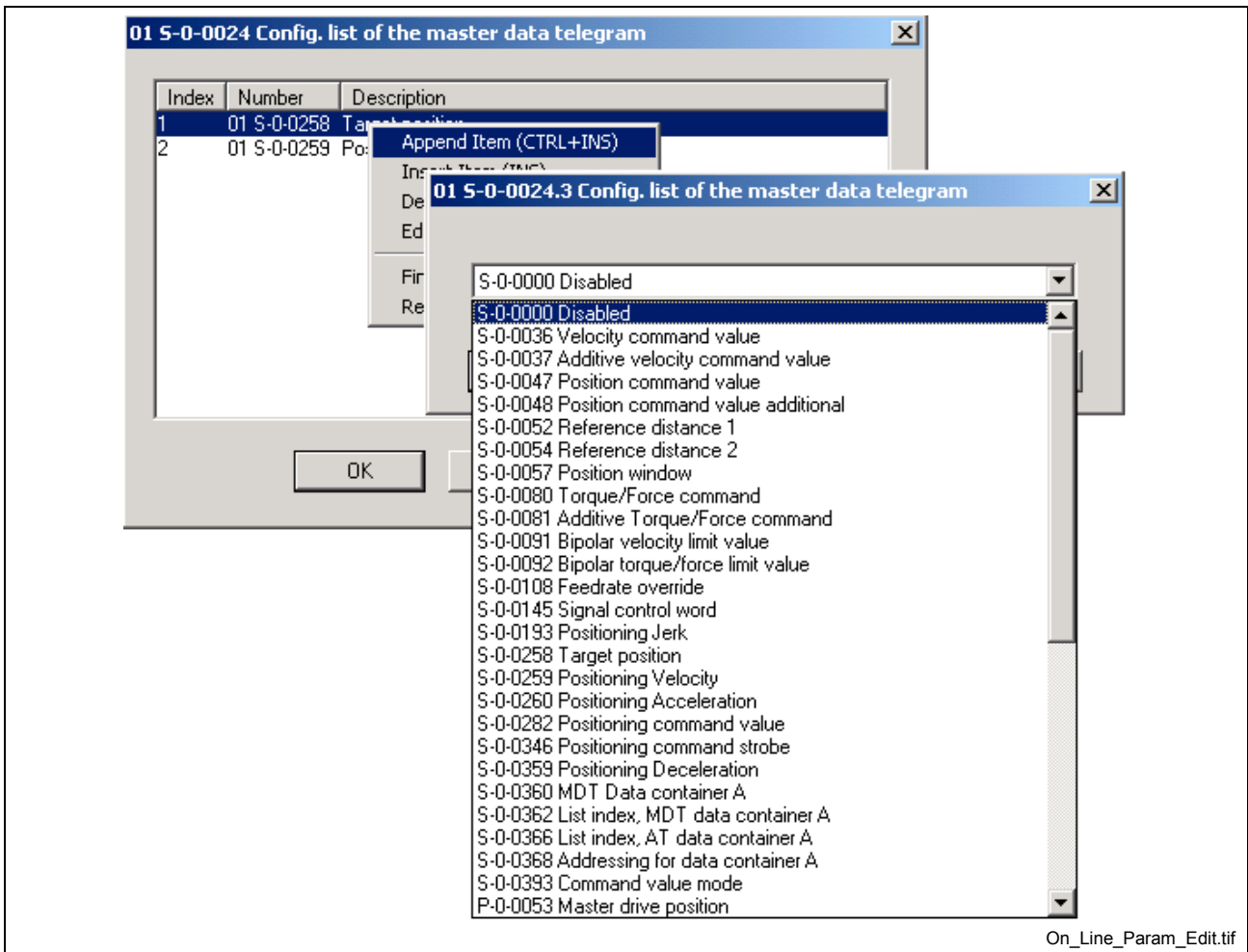


Fig. 2-122: Parameter Selection Edit Window

Custom

The **Custom** tree icon provides the user with easy to create and manage parameter lists that are specific to their application. From this tree icon selection, the user can create, modify and delete custom lists and custom list groups.

A *Custom List* is a grouping of parameters that are user selected.

A *Custom List Group* is a tree icon that is created under the **Custom** tree icon and used to group multiple custom lists together.

Note: All custom groups and custom list are stored under the **Rexroth\VisualMotion9\param** directory. Custom list are saved with a **".custom"** file extension and custom list groups appear as subfolders under **param**.

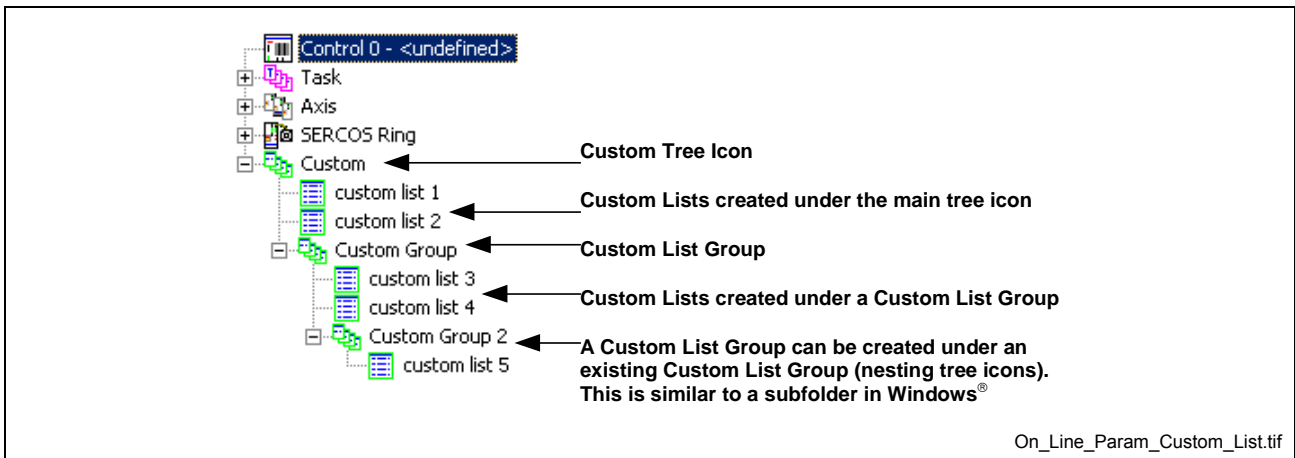


Fig. 2-123: Custom List File Structure

Create a Custom List

Any combination of parameters from the four types (Control, Task, Axis or SERCOS Ring) can be added to a custom list. By creating a custom list, the user minimizes the number of displayed parameters, making navigating and searching for a parameter easier.

To create a new custom list, use the following steps:

1. Select **Custom List ⇒ Create ⇒ List** from the main menu or right click on the **Custom** tree icon and select "Create a new custom list".

Note: In both cases, the **Custom** tree icon must first be highlighted before the selections are made available.

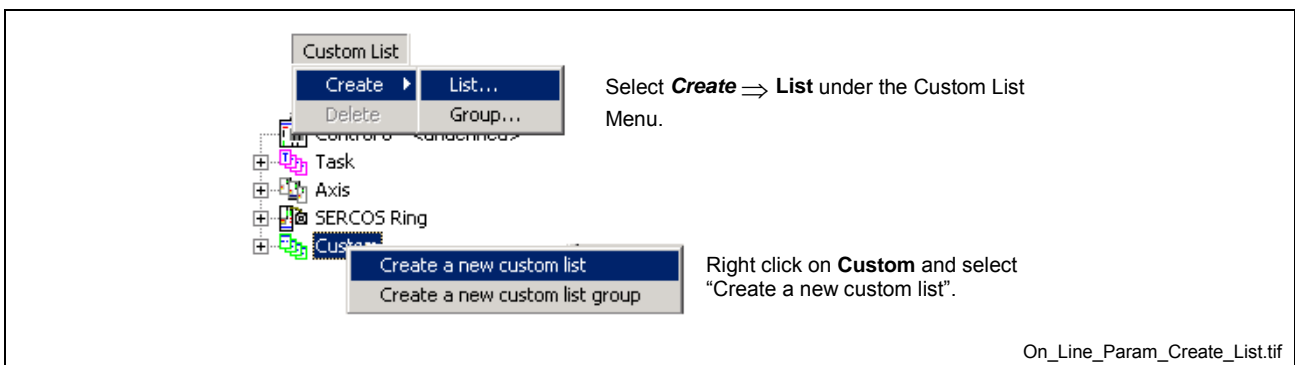


Fig. 2-124: Custom List Menu

2. Enter a name in the *Create Custom List* window, up to a maximum of 80 characters, that identifies the list and press the **OK** button.

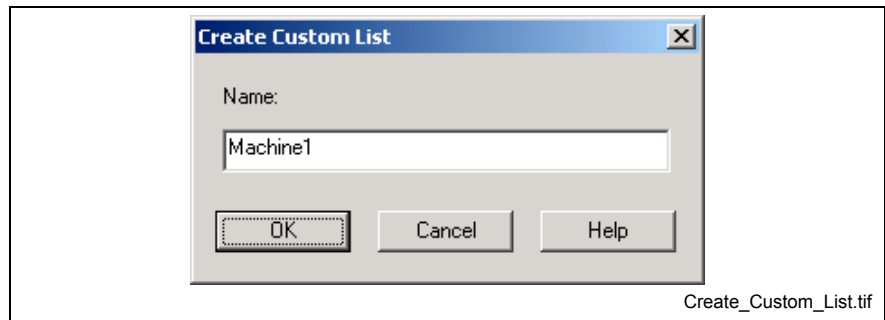




Fig. 2-125: Create Custom List

3. The following sequence is how parameters are added to a custom list within the *Custom List Editor* window.
 - a. Select one of the following parameter sets from the drop down list:
 - Control Parameter Set
 - Task Parameter Set
 - Axis Parameter Set
 - SERCOS Parameter Set (Drive or I/O)
 - b. If applicable, select a task (A, B, C or D) for Task Parameter Set and an address number for Axis or SERCOS Parameter Sets.
 - c. Parameters are added to a list by double clicking on the desired parameter or highlighting the parameter from the left window and pressing on the  insert button. Repeat the process for the another parameter set until all the desired parameters are added to the list.
 - d. Press the **OK** button to save the custom list.

Note: To remove a parameter from the custom list, double click on the parameter in the right window or highlight it and press the  remove button.

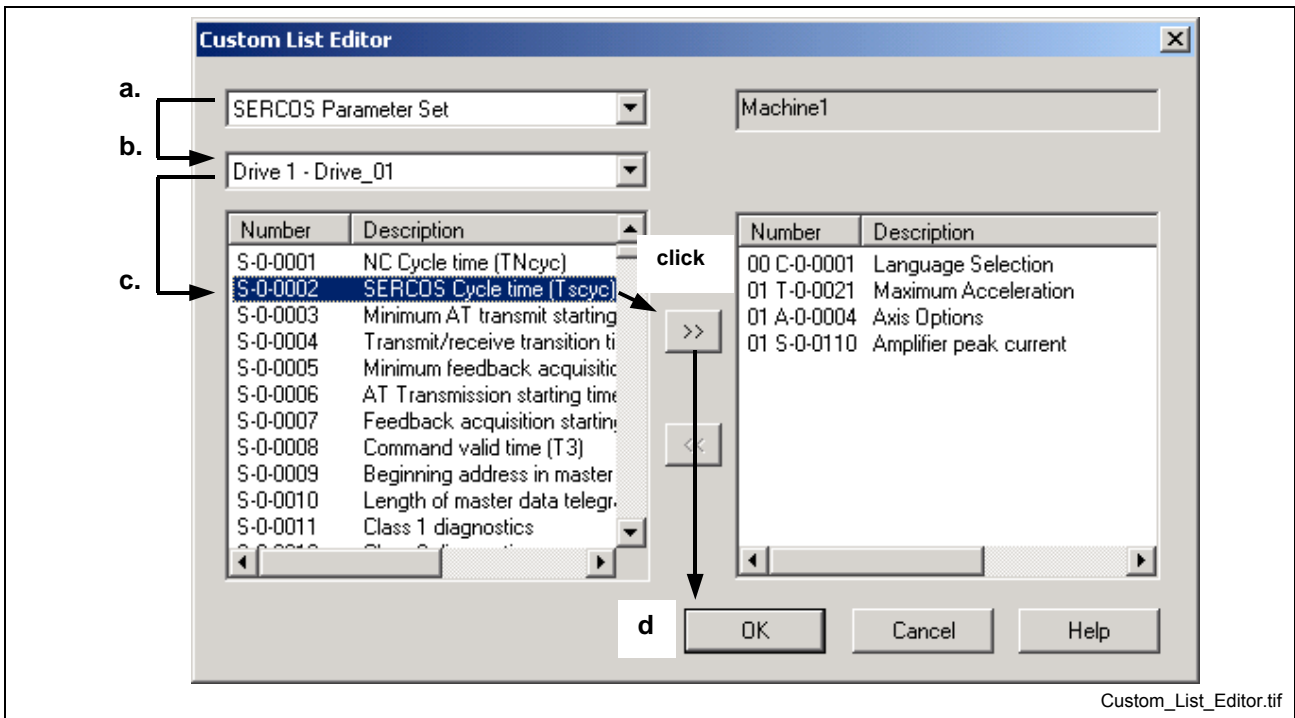


Fig. 2-126: Custom List Editor

The newly created custom list name will appear below the **Custom** tree selection, as shown in the figure below.

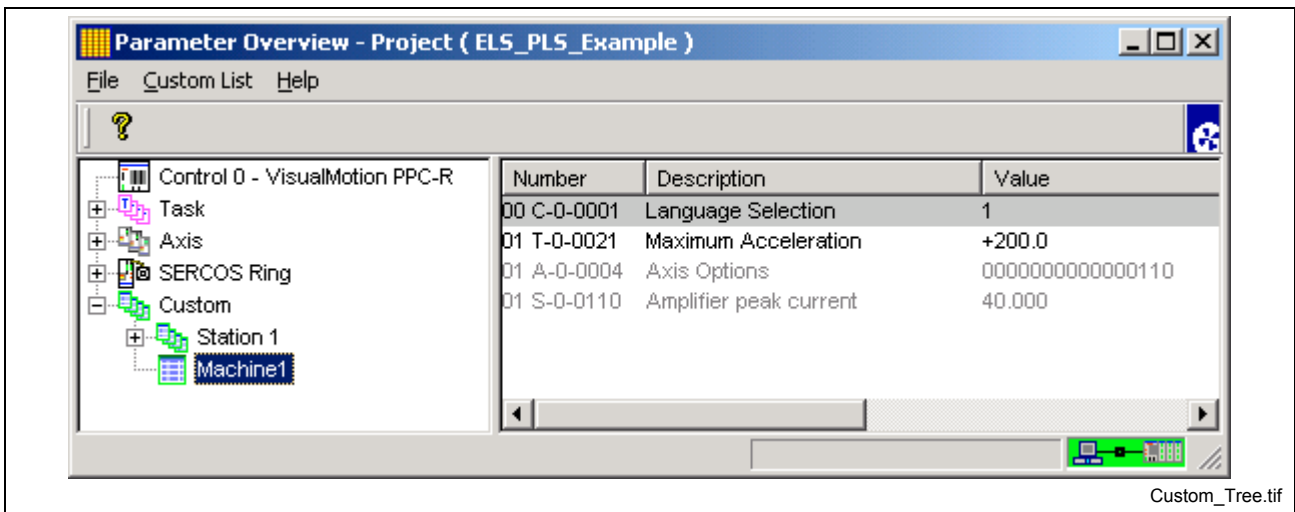


Fig. 2-127: Custom Tree Selection

Note: To delete a custom list, right click on the name and select "Delete the selected custom list item" from the popup window.

Modify a Custom List

To modify a custom list, use the following steps:

1. Select the custom list from the tree structure. The contents of the custom list are displayed in the right window.

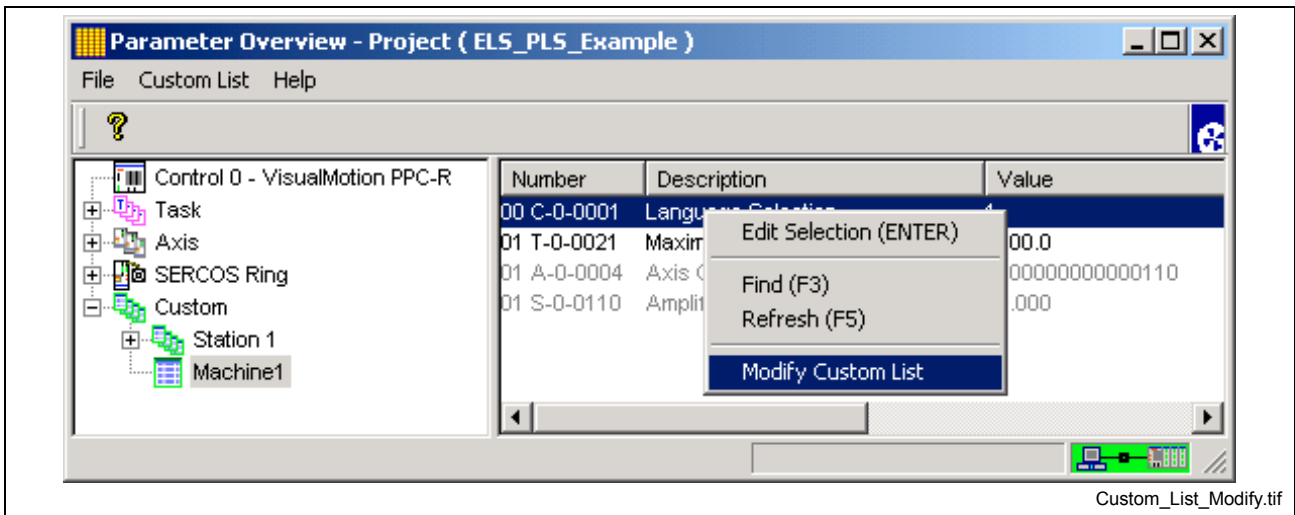


Fig. 2-128: Modify Custom List

2. Right click anywhere in the right window and select "Modify Custom List".

The addition and removal of parameters is described in section, "Create a Custom List".

Create a Custom List Group

A custom list group is a tree icon created under **Custom** where multiple custom lists can be grouped. This option allows the user to group or categorize different custom lists to a particular project or machine. Multiple custom list groups can be created following the previous group (nesting).

To create a custom list group, use the following steps:

1. Select **Custom List** ⇒ **Create** ⇒ **Group** from the main menu or right click on the **Custom** tree icon and select "Create a new custom list group".

Note: In both cases, the **Custom** tree icon must first be highlighted before the selections are made available.

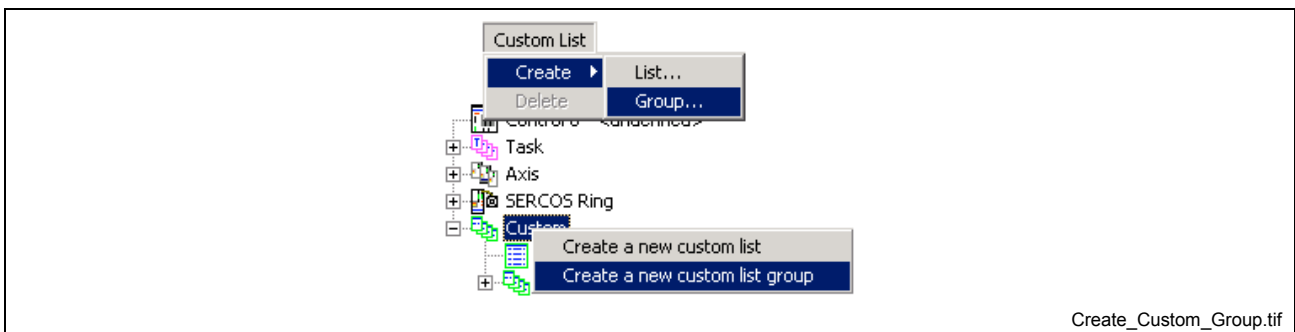


Fig. 2-129: Create Custom Group

2. Enter a name in the *Create Custom List Group* window for the custom, up to a maximum of 20 characters, that identifies the group and press the **OK** button.

The newly created custom list group name will appear below the **Custom** tree icon, as shown in the figure below.

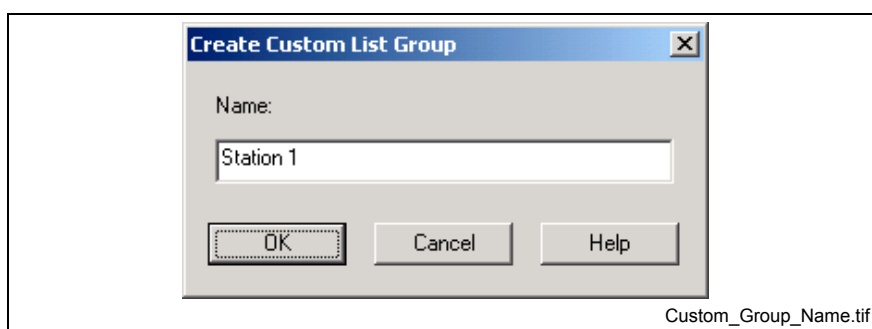


Fig. 2-130: Name Custom Group

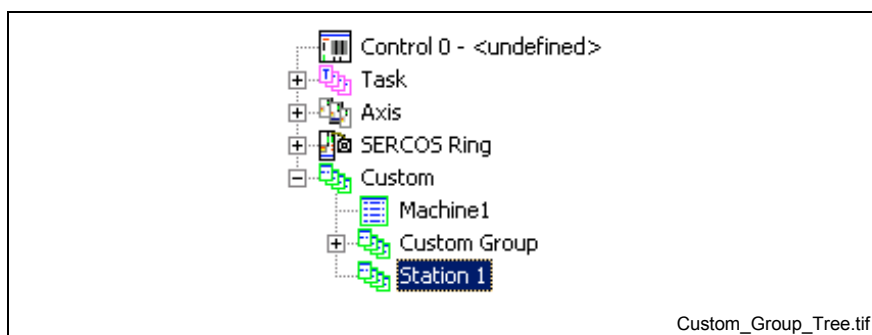


Fig. 2-131: Custom Group Tree

To add a custom list under the group name, select the group and follow the steps in section Create a Custom List.

Note: To delete a custom list group, right click on the group name and select "Delete the selected custom list item". This option is not available if a custom list exists under the group name.

System Configuration

The system configuration overview for the connected system components can be viewed from the Parameter Overview tool by selecting the top-level selection for Task, Axis or SERCOS Ring.

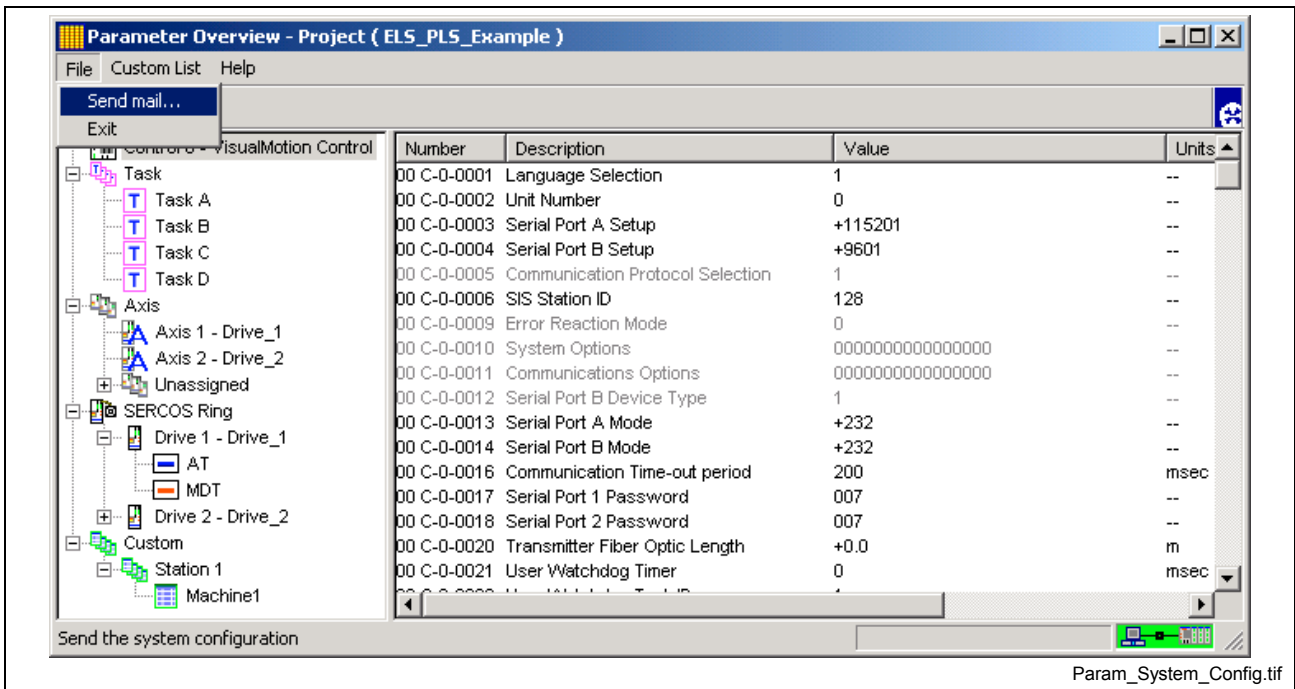


Fig. 2-132: System Configuration

This system configuration can be e-mailed to a recipient by selecting **File ⇒ Send mail**. This option launches the e-mail system on the PC and attaches a text file containing the system configuration as displayed in the figure above.

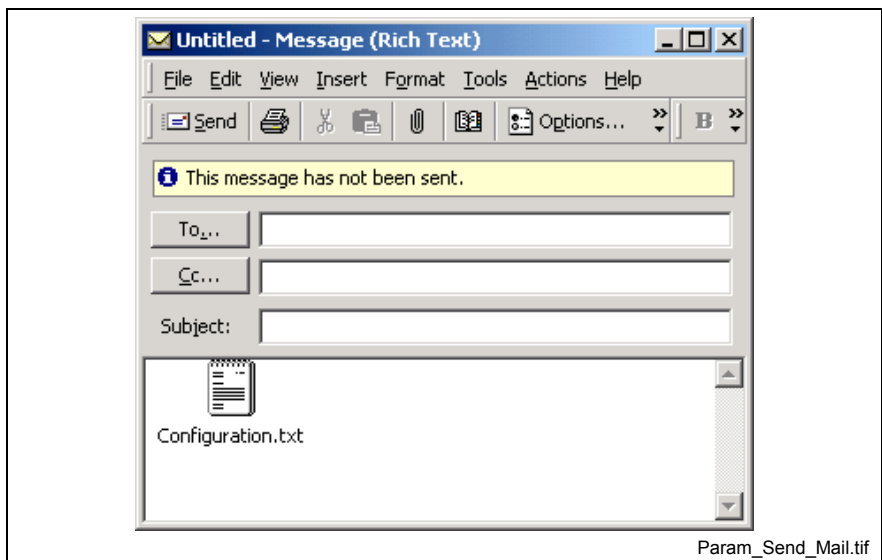


Fig. 2-133: Send Mail

Note: The *Send mail...* option will be visible only if the PC has a configured e-mail client.

Registers

Selecting **Data** ⇒ **Registers** opens the *Data Editor Tool* window displaying all 1024 (16-bit) registers available on the control. From the data type drop-down list, the user can also view and edit program integers, program floats, global integer and global floats. In addition, a user defined custom list can be created containing a combination of the mentioned data types. The available display formats are binary, hexadecimal and decimal.

Note: Refer to chapter 4 for a description of the available system reserved and default registers.

Registers can be modified as a complete binary register, individual bits, or by forcing the register bits.

Sorting Data

Any of the three data type columns (Number, Label and Value) can be sorted in ascending or descending order by clicking on the column heading.

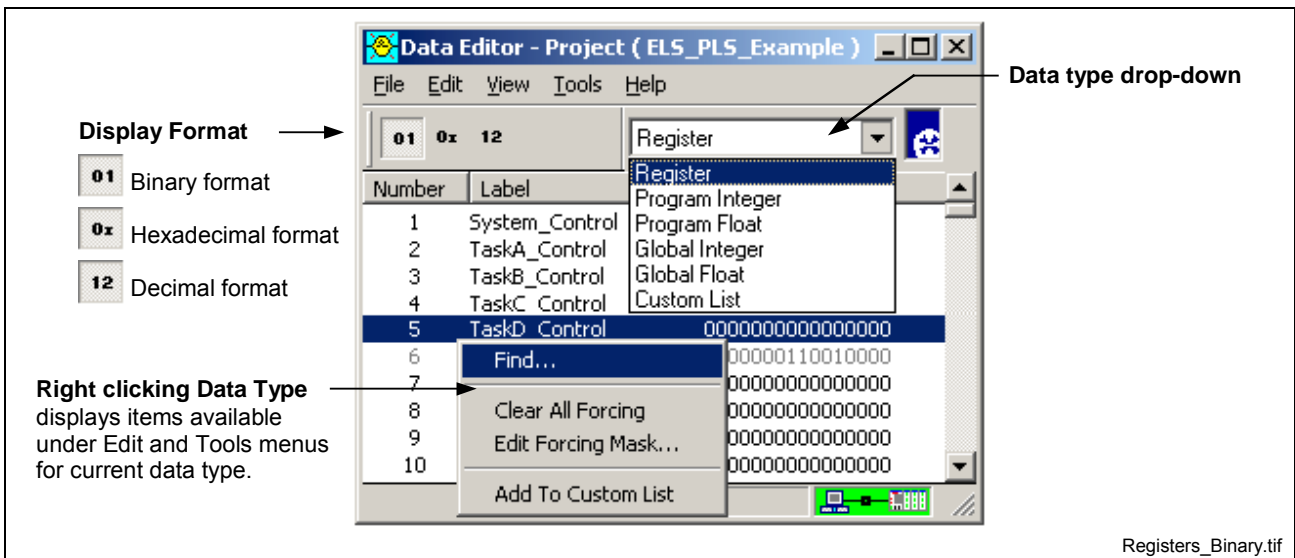


Fig. 2-134: Registers

Register Priority

Register priority is based on its usage in the system. Register priority is listed as follows, from highest-to-lowest:

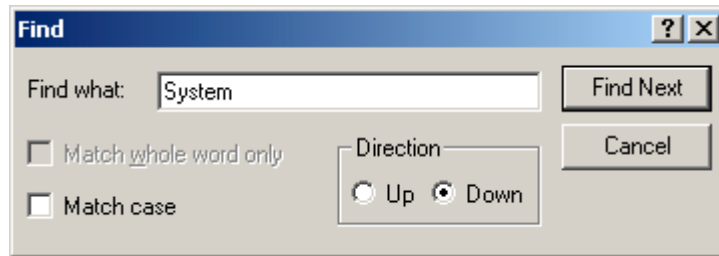
- Highest**
 - I/O Task (I/O subsystem)
 - I/O Setup Inputs
 - I/O Mapper
 - I/O Setup Outputs
 - Events
 - User Program (Tasks A-D)
 - I/O Bit icon
 - Register Transfer icon
- Lowest**
 - Serial Port
 - any access to registers over the serial port

Menu Items

File – *Program* allows the user to choose which program to display and *Exit* closes the window.

Note: *Program* selection is only available in service mode. Project mode will only read the current project data.

Edit – Allows the user the search for a specific data type Number, Label or Value by using the **Find...** feature.



View - Allows the user to change the displayed format of the register data between Hex, Binary, and Decimal.

Tools – The items available are dependent on the current data type selected and displayed. The following menu items are available:

- **Control Selection...** (service mode only)
 - **Force A Register...**
 - **Clear All Forcing...**
 - **Edit Forcing Options...**
 - **Add to Custom List**
 - **Delete From Custom List** (available in Custom List)
 - **Save Global Variables** (available for Global Integers and Floats)
- } (available for registers)

Help - allows the user to launch the help system and provide information about current VisualMotion software installed.

Edit a Register

To edit a register double click on the desired register to open the Edit Register window. The window displayed is dependent on the display format of the selected register.

Binary Register

Binary display registers are edited using the following *Edit Register* window. The register bit label is displayed (if configured) with the current state of the bit (0 or 1). To change the state of a bit click on the label or bit state (0 or 1).

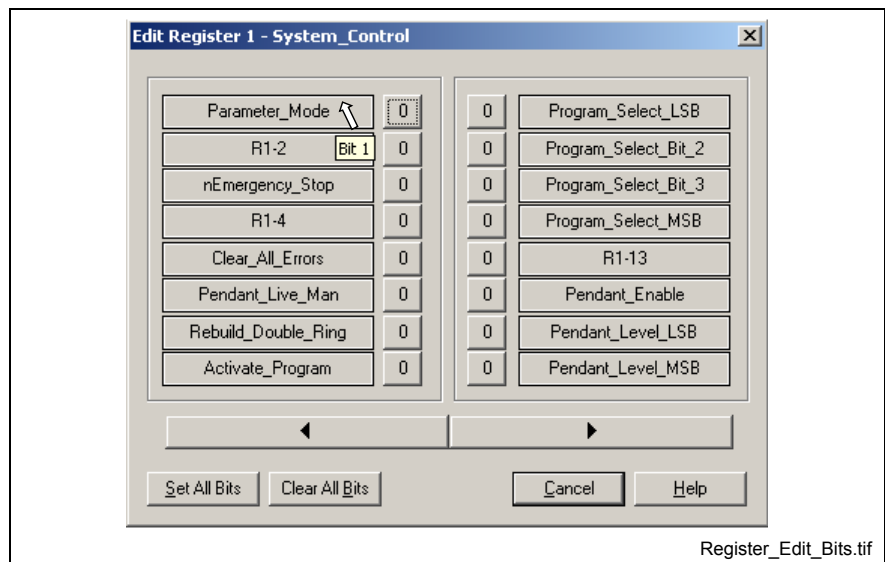


Fig. 2-135: Bits of Registers

Register bit numbers are displayed as a tool tip when the mouse pointer is held over a register bit label or bit state, as illustrated in Fig. 2-135.

The **Set All Bits** and **Clear All Bits** buttons can be used to set or clear all register bits at one time.

The ***left*** (previous) and ***right*** (next) arrow buttons allows the user to modify the bits of additional register without the need to exit back to the main register window. These buttons operate using a spin-control feature. This means the user can either advance to the next register or go back to the previous register

Hexadecimal and Decimal Registers

Hexadecimal and Decimal displayed registers are edited using the following *Edit Register* windows.

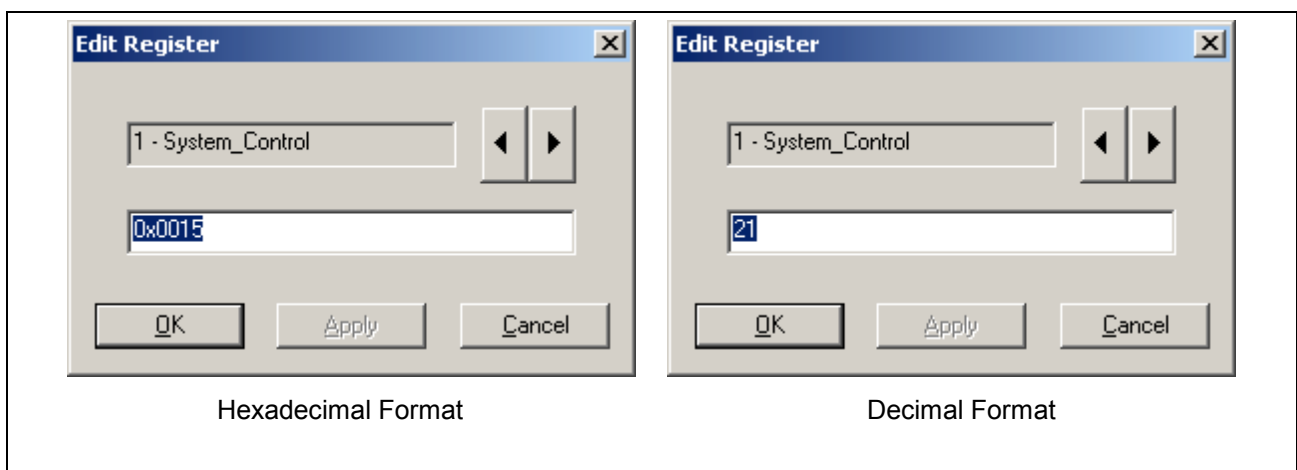


Fig. 2-136 Hexadecimal and Decimal Registers

Custom List

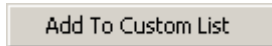
The Data Editor allows the user to create a custom list of the data types available from the data type drop-down list.

Create a Custom List

A custom list containing registers and program variables can be created to simplify the view of registers and program variables in a project.

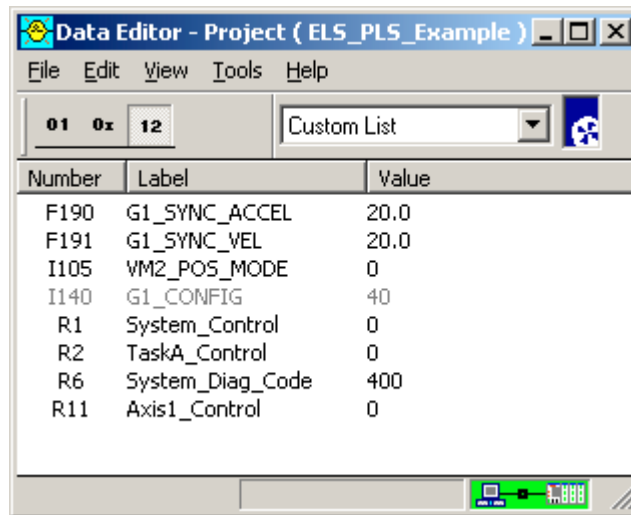
To create a custom list...

1. Display either registers or program variables from the drop-down list.
2. Using the mouse, right click on the desired data item and select Add to Custom List.



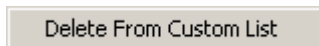
Note: Items can also be selected as a group by selecting the first item, holding the Shift key down, and selecting the last item and then right clicking to select the **Add to Custom List** button.

3. When complete, select *Custom List* from the data item drop-down list to display your custom list. The *Number* column identifies the data type added to the custom list.



Note: The data types displayed can be sorted in ascending and descending order by clicking on the column heading (Number, Label or Value).

4. To remove a data item from your custom list, right click over the data item and select *Delete From Custom List*.



Force a Register...

System installation and troubleshooting may require directly changing the state of register bits without depending on the I/O sub-system. Selecting **Tools** ⇒ **Force a Register** opens a *Register Forcing* window that allows you to setup a forced bit during system setup and debugging.

**WARNING**

Bit forcing directly changes the state of the control's inputs and outputs. Forcing I/O bits can result in harm to people and equipment. Make sure you fully understand all the effects on the system that could result from forcing an I/O line.

Forcing mask – Selecting the label name enables which bits may be affected by forcing.

Forcing state - Selecting the bit state (0 or 1) determines the actual state of the masked bits.

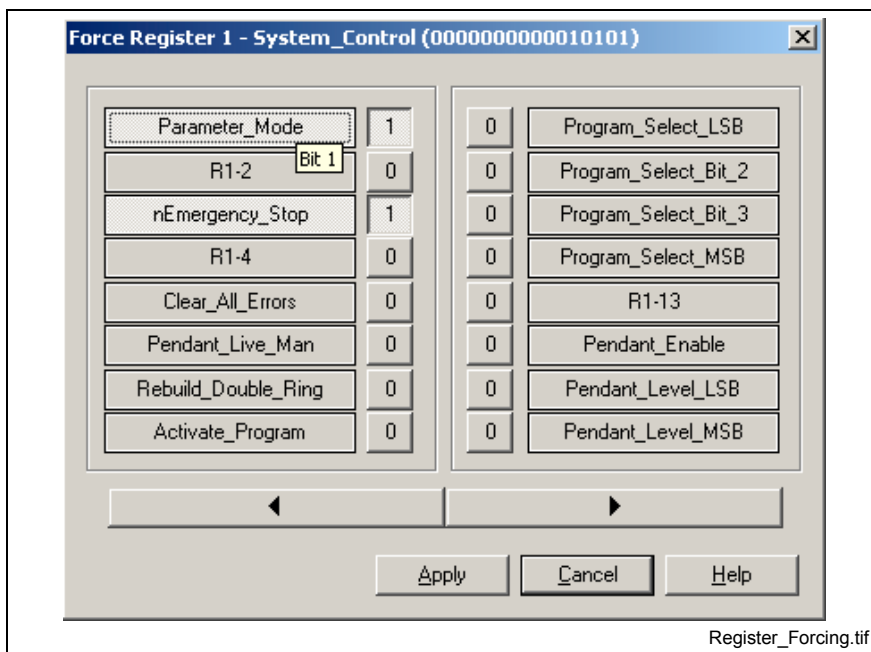


Fig. 2-137: Register Forcing

Forced bits cannot be affected by a *VisualMotion 9* program or the I/O subsystem. Forcing directly accesses the control system's I/O lines. Bits that are forced will remain in the forced state until the forcing is changed, the control is reset or power is cycled.

Forcing employs forcing mask and forcing state 16-bit control words allowing you to change a single bit, or combination of bits, within the register without affecting the other bits.

Note: Registers mapped to runtime tools such as the Fieldbus Mapper, ELS, PLS, PID, Coordinated Articulation, etc., are not affected by register forcing. However, registers used in runtime tools such as the I/O Mapper, I/O Setup and programming icons, used to change the state of register, can be forced.

Note: To clear all forcing from the system, select **Options** ⇒ **Clear All Forcing... F5**.

Clear All Forcing

To clear all forcing from the system, select **Tools** ⇒ **Clear All Forcing...**

A warning message will be displayed informing the user of the possible danger associated with clearing forced registers.

Variables

Selecting **Data ⇒ Variables** opens the *Data Editor Runtime Tool* window. The window automatically uploads and displays variables of the active project on the control. In service mode, a program is selected with **File ⇒ Program Select**.

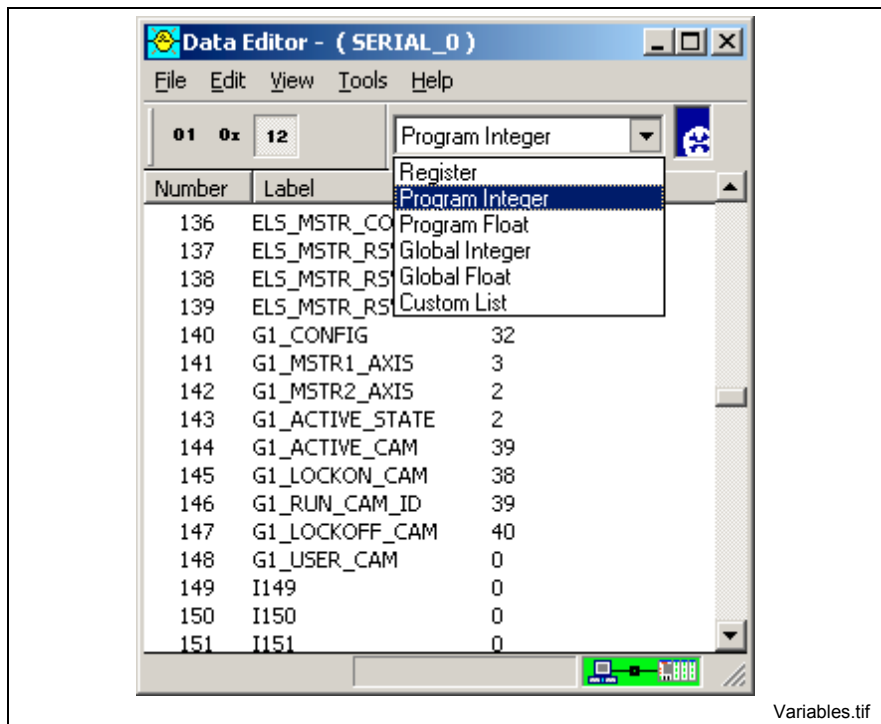


Fig. 2-138: Variable

Saving Global Variables to Flash

Global variables are initialized to zero on power up and do not retain programmed values on power down unless they are saved to flash memory. The number of allowable global variables in a project is determined by control parameters C-0-0080 (Global Integers) and C-0-0081 (Global Floats). The default number of global integers and global floats in a project is 512 and 256, respectively.

Global variables can be saved to flash memory using one of the following methods:

- Data Editor Menu Selection
- Editing Command Parameter C-0-0082

Flash Global Variables from Data Editor

1. Set VisualMotion Toolkit to service or online mode.
2. Switch the control to parameter mode.
3. Select **Data ⇒ Variables** and select either Global Integer or Global Float from the drop-down list.
4. Select **Tools ⇒ Save Global Variables to Flash...**

Flash Global Variables via Control Parameters

1. Switch the control to parameter mode.
2. Edit C-0-0082 and transition bits 1 and 2 from (0 to 1).
Status parameter C-0-0083, bits 1-3 display a (1) when command is set.
3. Switch the control to manual or automatic mode.
Status parameter C-0-0083 indicates success or error as follows:
 - Bits 1 and 2 set to (1) indicating a successful flash
 - Bits 1-4 all set to (1) indicating an error.

Global variables are now stored in flash memory and can be reinitialized with saved values.

Note: To save global variables again, edit C-0-0082 and transition bits 1 and 2 from (1 to 0) and repeat steps 1 - 3.

Events

Selecting **Data** ⇒ **Events** opens the *Events Runtime Tool* window. Events are used to execute an event function on time, position, angle or I/O State conditions. Each event has status, type, direction, distance or time, event function, and message fields. The tool window permits viewing and editing of the event table of a program that has already been downloaded to the control. The window automatically uploads and displays the contents of the event table for the currently active program on the control.

The following figure shows the main window that appears for the Event function:

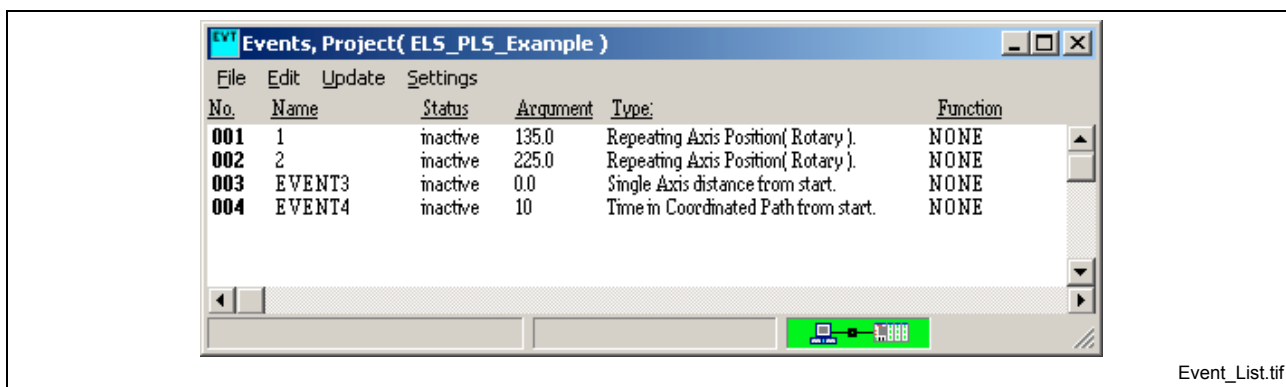


Fig. 2-139: Event Runtime Utility

Edit an Event

To setup or edit an event:

1. Double-click on the event to open the following Edit Event window opens.
2. Select an Event Type and options and click on the **Apply** or the **OK** button to send to the control.

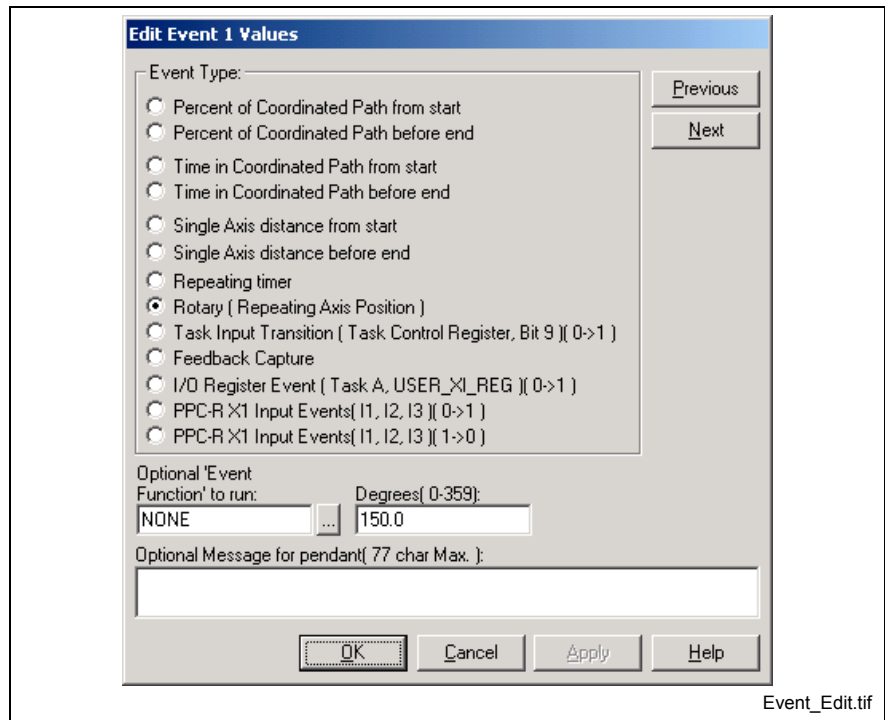


Fig. 2-140: Edit Event Window

The following event types are supported:

- **Percentage of Coordinated Path**

Events for Coordinated path are related to the **start** or **end** of a path segment as a percentage of the total path distance on the path segment.

- **Time in Coordinated from Start/before End**

Coordinated motion can provide time-based events that are related to travel time along a specified geometry segment and are initiated by the path planner. These events execute at a fixed time after motion starts or before motion ends in the specified segment.

- **Single Axis Distance**

Events for single axis motion may be set to take place at an absolute distance from the **start** or **end** of the axis move.

- **Repeating Timer**

A cyclic event triggered by a continuously Repeating Timer.

- **Rotary (Repeating Axis Position)**

An event triggered each time the configured axis encounters an absolute position. The axis motion type can be single-axis, ELS, ratio or velocity mode and configured for modulo or non-modulo positioning.

- **Task Input Transition**

This event type triggers events on a low to high transition of a Task Control Register interrupt event bit. The I/O Mapper is used to map a specified I/O register condition from the I/O register to bit 9 in the appropriate Task Control Register. The approximate latency is ≤ 2 milliseconds.

- **Feedback Capture**

This event type uses the Probe capability of Bosch Rexroth drives to trigger a VisualMotion 9 event based on a positive or negative transition of the drive's Probe 1 or Probe 2 input.

- **I/O Register Event**

This event type uses the bits of input Register 88 (USER_X1_REG) to trigger up to 16 events **only** in Task A. Register 89 (USER_X0_REG) is used to monitor the status of events triggered by Register 88.

- **PPC-R X1 Input**

This event type uses the PPC-R's X1 digital inputs (I1, I2 and I3) to trigger an event based on a positive or negative transition.

Refer to chapter 5 of the VisualMotion 9 Application manual for detailed information on how to create events.

Points

Selecting **Data** ⇒ **Points** opens the *Points Runtime Tool* window. The user can view and edit absolute and relative point tables in the currently active program.

Note: Relative and Absolute points are added to a project by selecting the VM Data icon, selecting the REL or ABS Points tab and clicking on the Add button. Refer to page 2-36 for details.

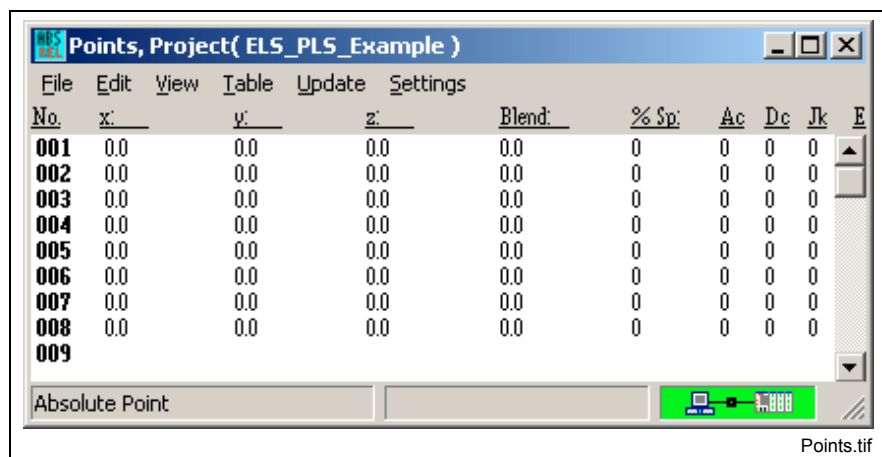


Fig. 2-141: Points Window

The window automatically uploads and displays the contents of the point table for the VisualMotion 9 program that was last selected in the **File** ⇒ **Program Select** command. If no program has been previously selected, VisualMotion defaults to the active program. Points are used in coordinated motion programs to describe a location in Cartesian coordinates, tool orientation and associated events.

Note: The point table can also be referred to as a display of raw information for building CAMs using the VisualMotion CamBuild Icon. For more information, refer to the CamBuild Icon in chapter 3, Icon Programming.

File

The file menu has the following selections:

- **Program Select** allows the user to select an active program.
- **Exit** closes the Points windows.

View

The **View** menu presents a list of the point elements and allows the user to select or deselect any of them to determine which ones appear in the display.

Table

The **Table** menu allows the user to switch between the absolute and relative point tables.

Update

Clicking on **Update** refreshes the window and updates any changes that have been made.

Settings

Clicking on **Settings** opens the *Control Selection* window allowing the user to select a different method of communication.

Edit

The user can highlight a point and click Edit to make changes to existing point values. Another option is to double-click the desired point in the list. Editing a point table entry opens an absolute or relative Edit Point Values window, depending on which type was selected with the Table menu.

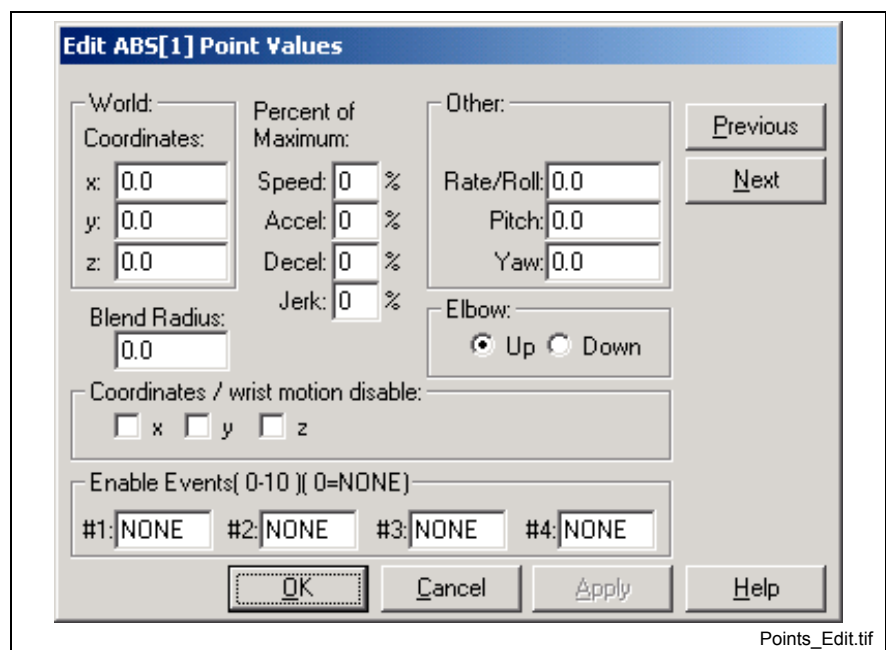


Fig. 2-142: Edit Points

The **Edit Point Values** window permit individually changing the values for each point table entry on the control. Clicking on the **Apply** button immediately downloads the new values to the program on the control. The changed values are displayed at the next automatic update.

Note: Coordinated moves using an ABS/REL table point with zeroes in any of the fields for Speed, Acceleration, Deceleration, and Jerk will default to 1%.

The first time a point is taught (and has zeroes in the Speed, Acceleration, Deceleration and Jerk fields) default values are loaded. The defaults are 10% for speed, 100% for acceleration, 100% for deceleration and 100% for jerk.

In the *Coordinates/wrist motion disable* section of this screen, the user can disable coordinated motion on one or more axes for a particular point. To include this feature in the VisualMotion program, refer to the **Calc** Icon in *Icon Programming*.

In the **Elbow Properties** section of this screen, the user can choose the Up or Down radio button to determine the elbow direction for a particular point. To include this feature in the VisualMotion program, refer to the section *Calc Icon Programming*.

Each point has: {x, y, z, blend, speed, acceleration, deceleration, jerk vent 1, event 2, event 3, event 4, roll, pitch, yaw, and elbow} fields.

To better navigate long tables, the user may click and drag the vertical scroll bar button while looking at the point number indicator that appears at the right of the window title bar. The point number corresponds to the top of the list when the scroll bar button is released. The user must expand the viewing window to view the additional roll, pitch, yaw and elbow properties used with a six-axis control system.

CAM Indexer

Selecting **Data** ⇒ **CAM Indexer** opens the *CAM Indexer Runtime Tool* window.

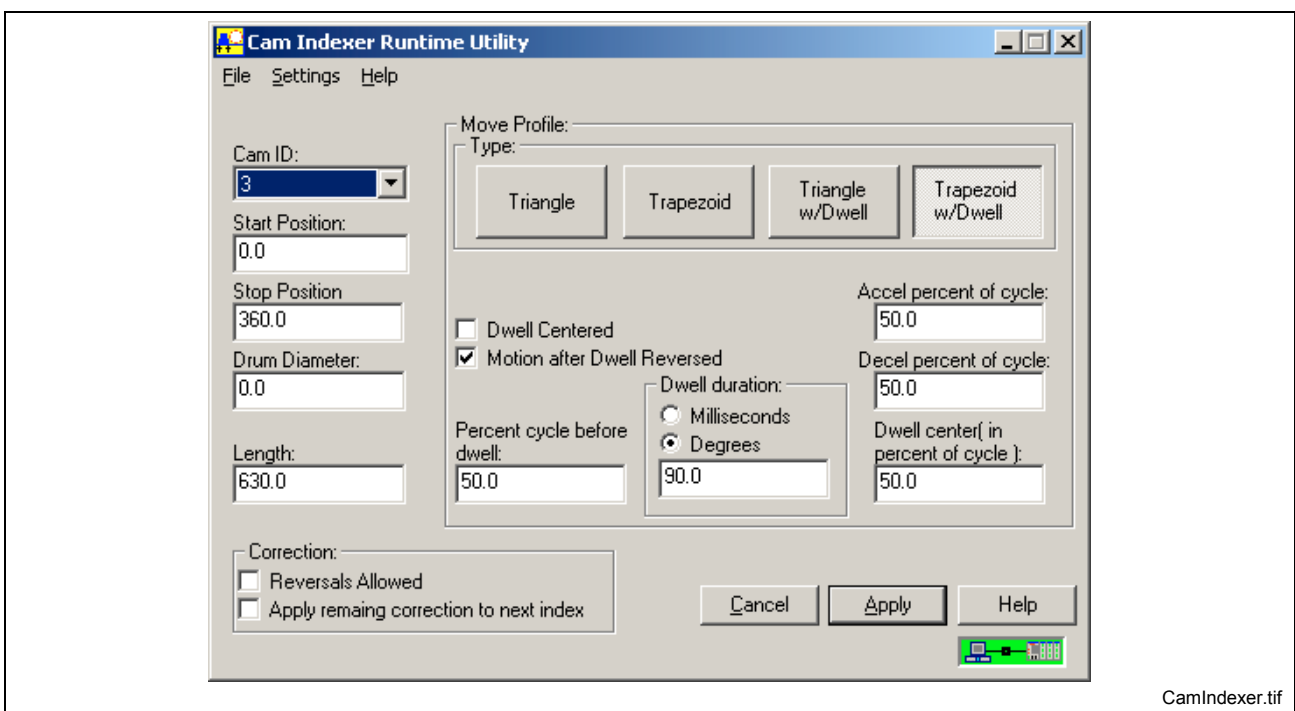


Fig. 2-143: CAM Indexer Runtime Utility

This utility allows for making on-the-fly changes to CAM Indexers in the active program that was setup using the **CAM Indexer** Icon. Variable assignment cannot be changed using this utility. Two options for correction are available in this runtime-utility (bottom left of window), which cannot be changed using the icon window:

- Reversals allowed -
- Apply remaining correction to next index -

Refer to ***CAM Indexer Icon*** for information about setting up a CAM indexer.

ELS

Selecting **Data** ⇒ **ELS** opens the *ELS Runtime Tool* window below:

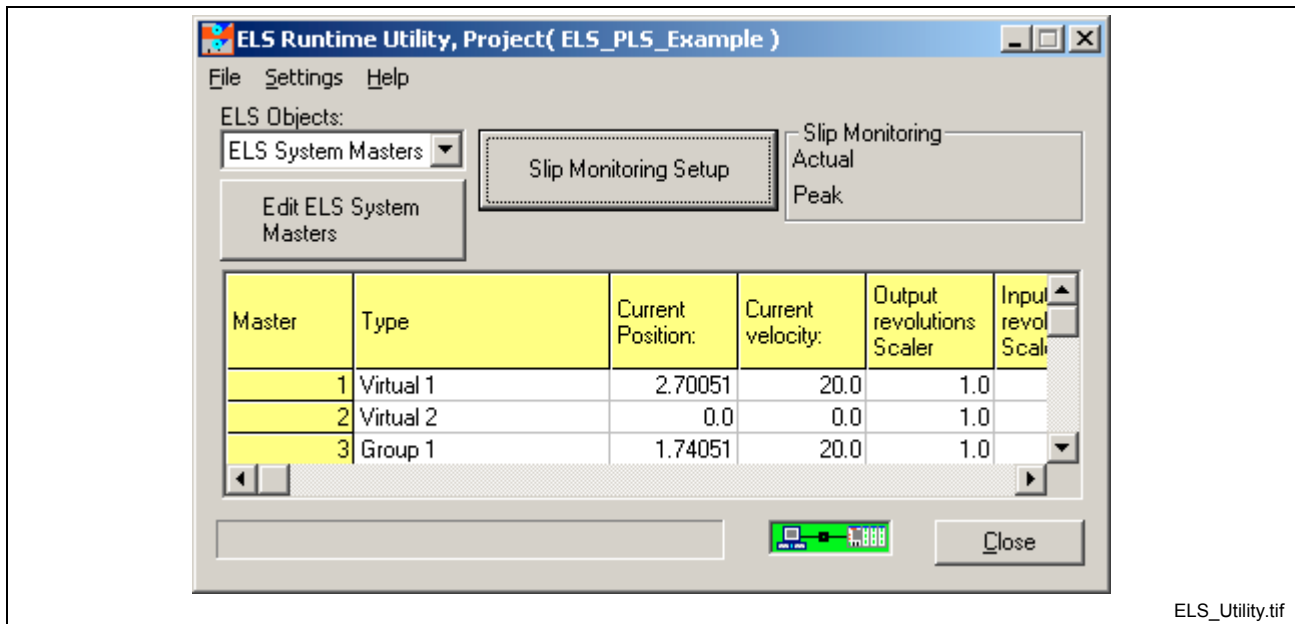


Fig. 2-144: ELS Runtime Utility

This tool is designed for modifying the default program variables used for parametrizing ELS data. These are variables that were initialized at compile time for the Virtual Masters icon (Assign Initial Values window), ELS Group icon (ELS Group x Variables) and the ELS Master Assignment icon (including assigned ELS Group Masters). These variables are used to control moving, stopping and jogging the following multiple master components:

- ELS Masters
- ELS Group Masters
- Virtual Masters

Note: A valid GPP ELS program must be active on the control for the ELS Runtime Utility to open.

Modification to the values overwrites the initially compiled and downloaded values on the control. The values remain active until they are overwritten (e.g. by more changes using the runtime utility or by compiling and downloading a new program to the control).

The **ELS Objects** list box at the top left allows selection of any ELS Masters, ELS Groups or Virtual Masters that have been setup. The button below the list box changes, depending on the ELS object selected.

ELS System Masters

Selecting **ELS System Masters** and clicking the button labeled "Edit ELS System Masters" opens the **Edit ELS System Masters** window below.

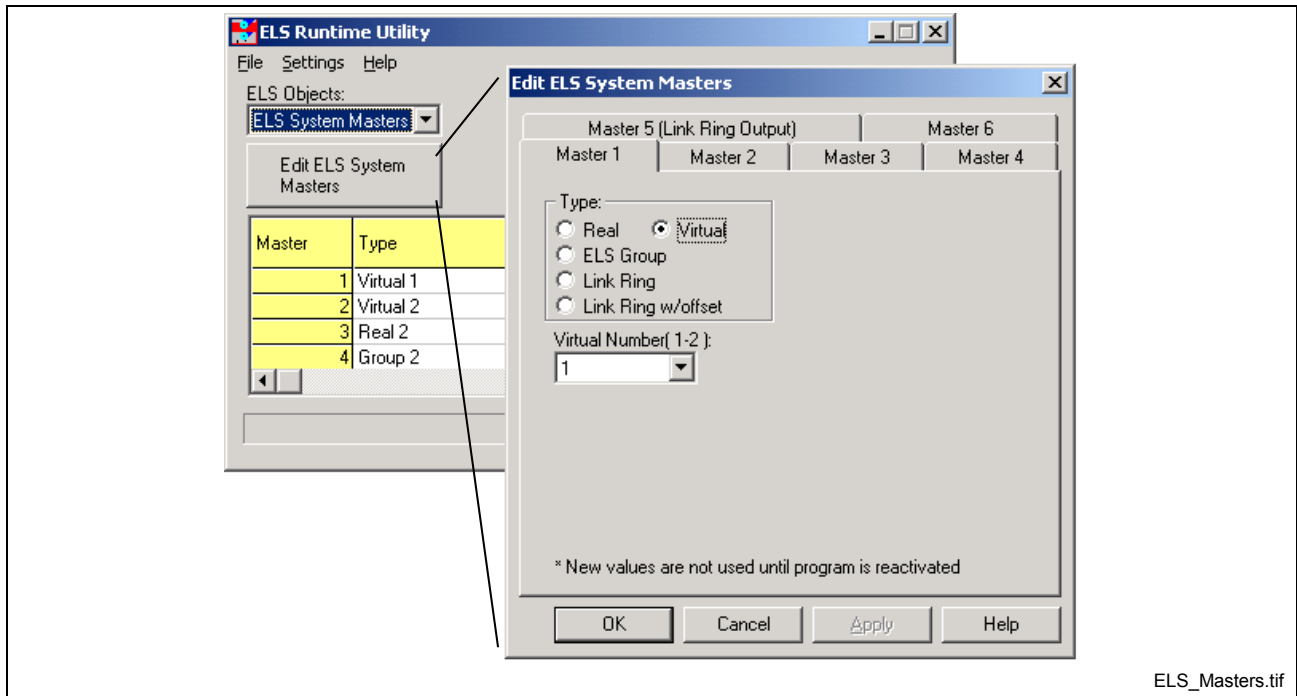


Fig. 2-145: Edit ELS System Masters

For more information about the initial setup of system masters and setting the values in this window. Refer to the **ELSMstr** icon in chapter 3, Icon Programming, for details.

ELS Group

Selecting **ELS Group x** in the list box and clicking the button labeled "Edit ELS Group x Variables" opens the **Edit ELS Group Variables** tab array below.

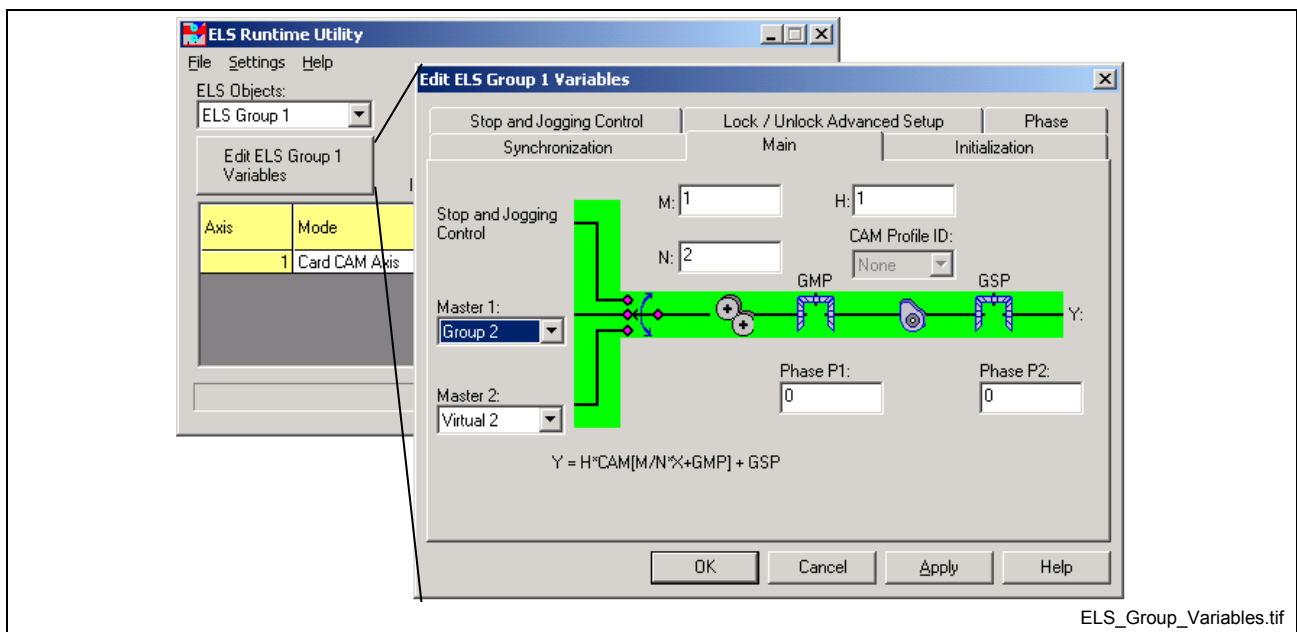


Fig. 2-146: Edit ELS Group x Variables

The tabs open windows that are accessible in the initial setup of ELS elements, using the **ELSGrp** icon.

Virtual Masters

Selecting **Virtual Masters** and clicking the button labeled "Edit Virtual Masters" opens the **Edit Virtual Masters** window below.

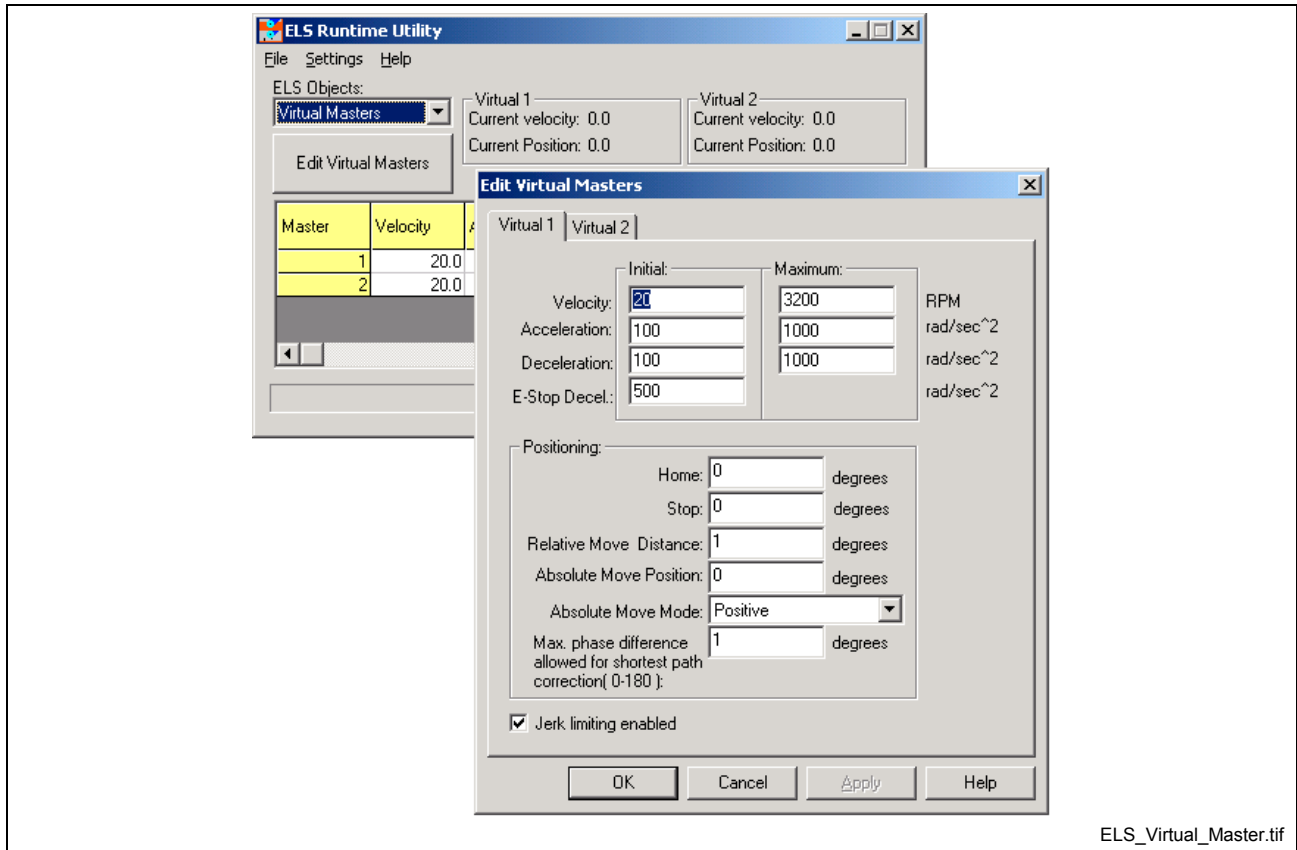


Fig. 2-147: Edit Virtual Masters

For more information about the initial setup of virtual masters and setting the values in this window, refer to the **VM** (Virtual Master) icon in the section titled **Icon Programming**.

Slip Monitoring Setup

ELS System Master slip monitoring is used to initiate an error reaction when the phase difference between two system masters exceeds the maximum allowed deviation window variable. Refer to the *VisualMotion 9 Application* manual under heading Electronic Line Shafting for details.

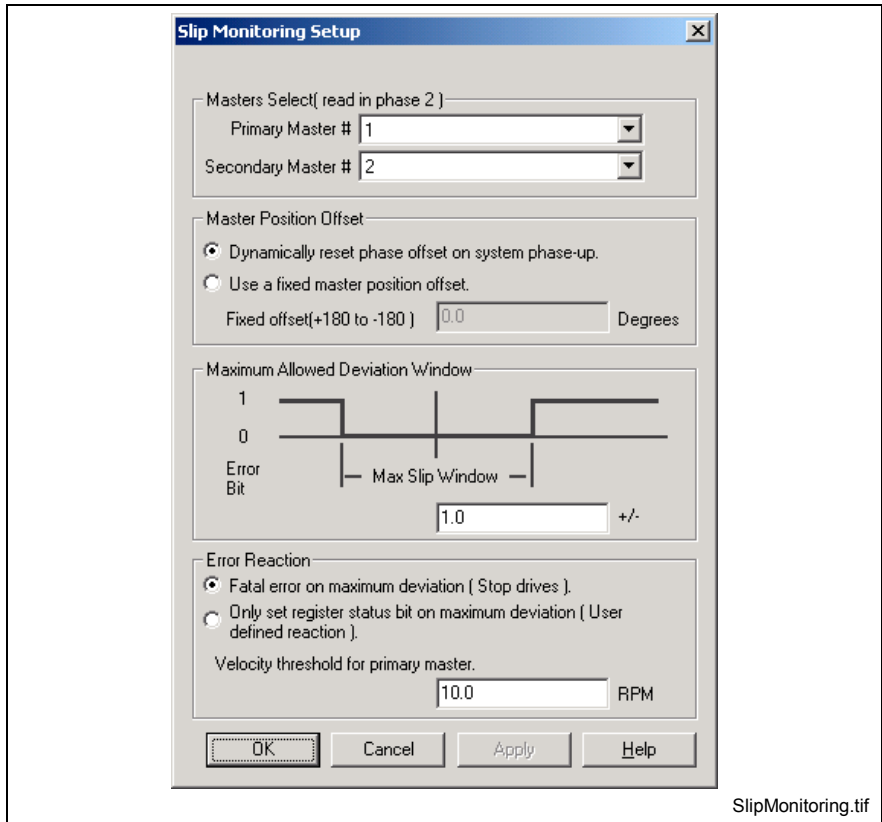


Fig. 2-148: Slip Monitoring Setup

PID

Selecting **Data** ⇒ **PID** opens the *PID Monitor Runtime Tool* window. This tool is used for monitoring and tuning the PID of the active program on the control.

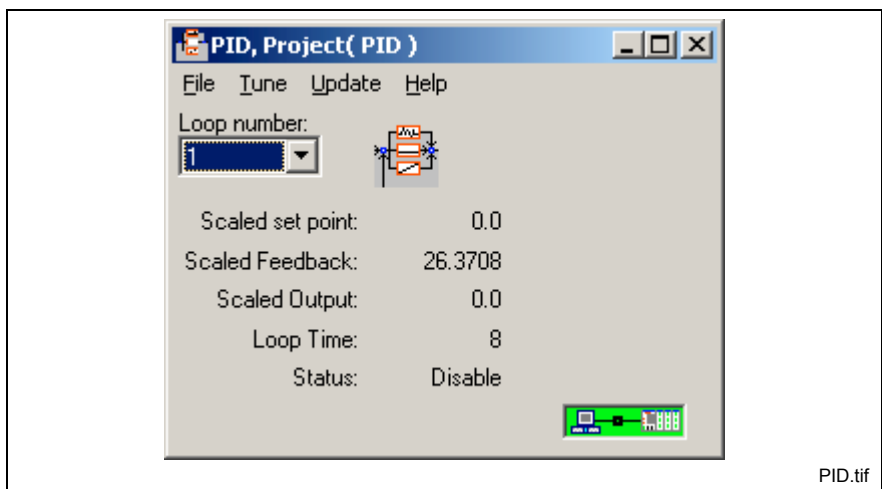



Fig. 2-149: PID Monitor Window

The control can include up to 32 PID (Proportional, Integral, Derivative) control loops with each program. These PID's are parameterized with program variables or registers, and have a minimum update rate of 8ms. A choice of optional filters (Low pass or Butterworth) may be applied to the feedback signal.

The PID causes corrective action to be taken before a problem becomes unmanageable. For instance in a machine dispensing chocolate onto a conveyer the chocolate must be kept at a certain temperature so as to hold it's form once dispensed but not thicken too early restricting the flow rates. The PID function would keep the temperature within the too hot and too cold limits much more precisely than a simple on off switch controlled by a thermostat.

The PID instruction is activated at program activation with SERCOS ring in phase 2 or greater and it's control register "PID Enable" (bit 5) set. The tasks do not need to be running.

PID Properties

Selecting *File* ⇒ *Properties* opens the PID Properties window below. This window displays the variables, registers and Control Block start float that was setup using the PID icon ().

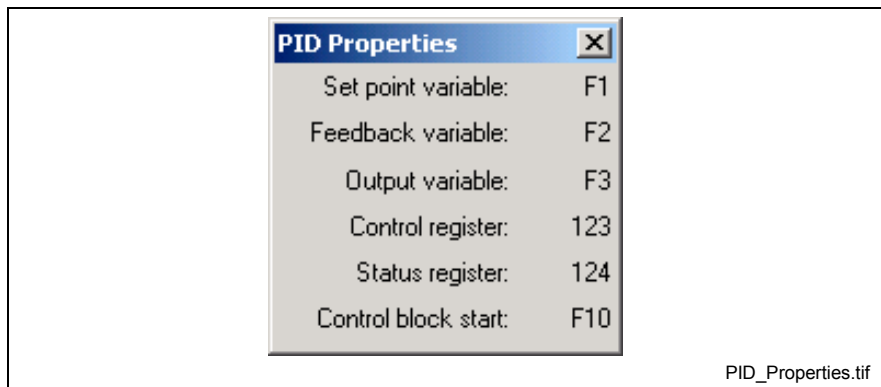


Fig. 2-150: PID Properties

Tune PID Control Block

Selecting **Tune** opens the *Tune PID Control Block* window. The tuning screen is used to adjust the selected PID loop while monitoring its values on the main screen. This screen has several grouping for scaling and adding offset to the process variables.

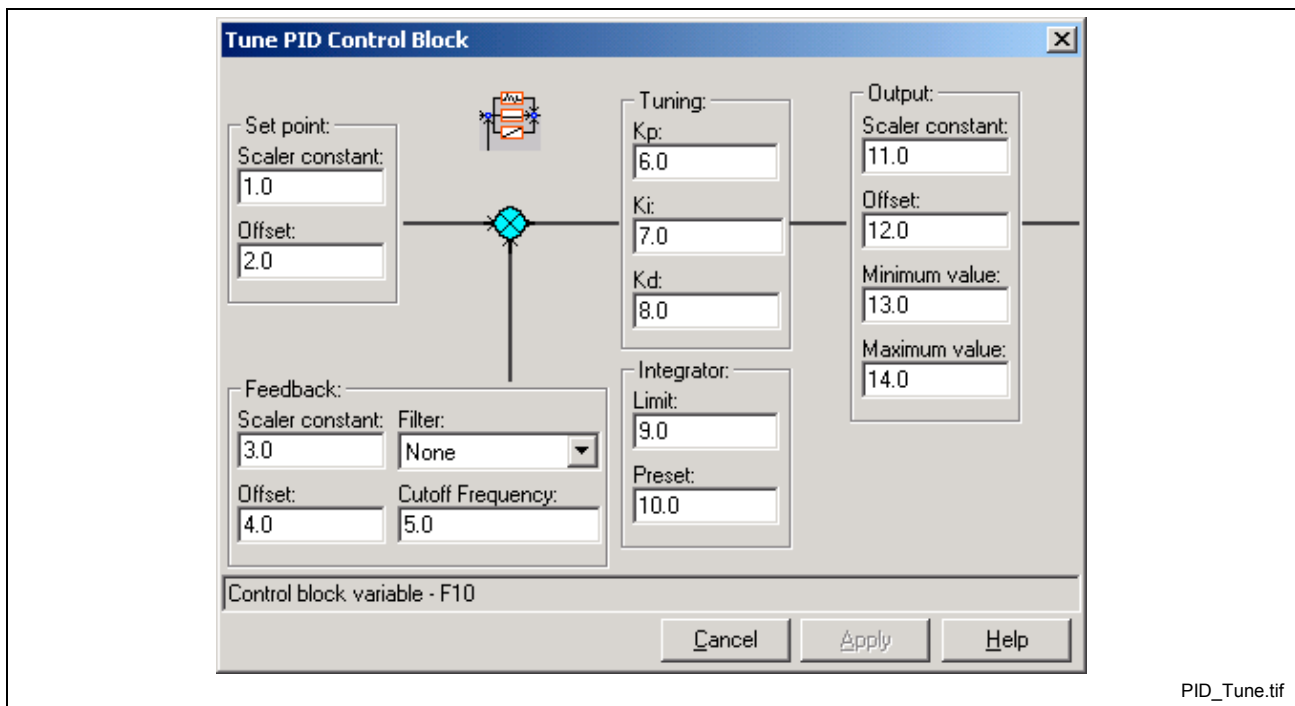


Fig. 2-151: Tuning PID

A grouping adjusting the Kd, Kp, Ki, integral preset, and integral limit are provided. The output grouping also has min, max limits for it. The feedback has an optional digital filter to condition the signal.

Feedback:

Digital filtering is available for PID loops and Real Masters.

Filter Type:

- None
- First order low-pass, $G(s)=1/(s+1)$
- Second order low-pass, $G(s)=1/(s^2 + 2s + 1)$
- Third order low-pass, $G(s)=1/(s^3 + 3s^2 + 3s + 1)$
- Second order Butterworth, $G(s)=1/(s^2 + 2^{1/2}s + 1)$
- Third order Butterworth, $G(s)=1/(s^3 + 2s^2 + 2s + 1)$
- Modified 2nd order low-pass with velocity ramp tracking,
 $G(s)=(2s+1)/(s^2 + 2s + 1)$
- Modified 3rd order low-pass with Accel ramp tracking,
 $G(s)=(3s^2 + 3s + 1)/(s^3 + 3s^2 + 3s + 1)$

Cutoff Frequency(float):

When a filter type is chosen, a cutoff frequency for the filter must be entered. The cutoff frequency is the frequency where the signal is reduced by 3db. When set to 0 the filter is disabled.

To ensure a stable system, use the following calculation when entering a value for the Digital Filter Cutoff Frequency:

$$\text{Cutoff Frequency} \leq \frac{1}{2 * \text{Sampling Rate(sec.)}}$$

The sampling rate for a PID loop is the set PID Loop Time, entered in seconds.

Example For a 8 ms PID Loop Time, the cutoff frequency is calculated as follows:

$$\text{Cutoff Frequency} \leq \frac{1}{2 * 0.008} = 62.5 \text{ Hz}$$

The following figure illustrates the frequency vs. degrees for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

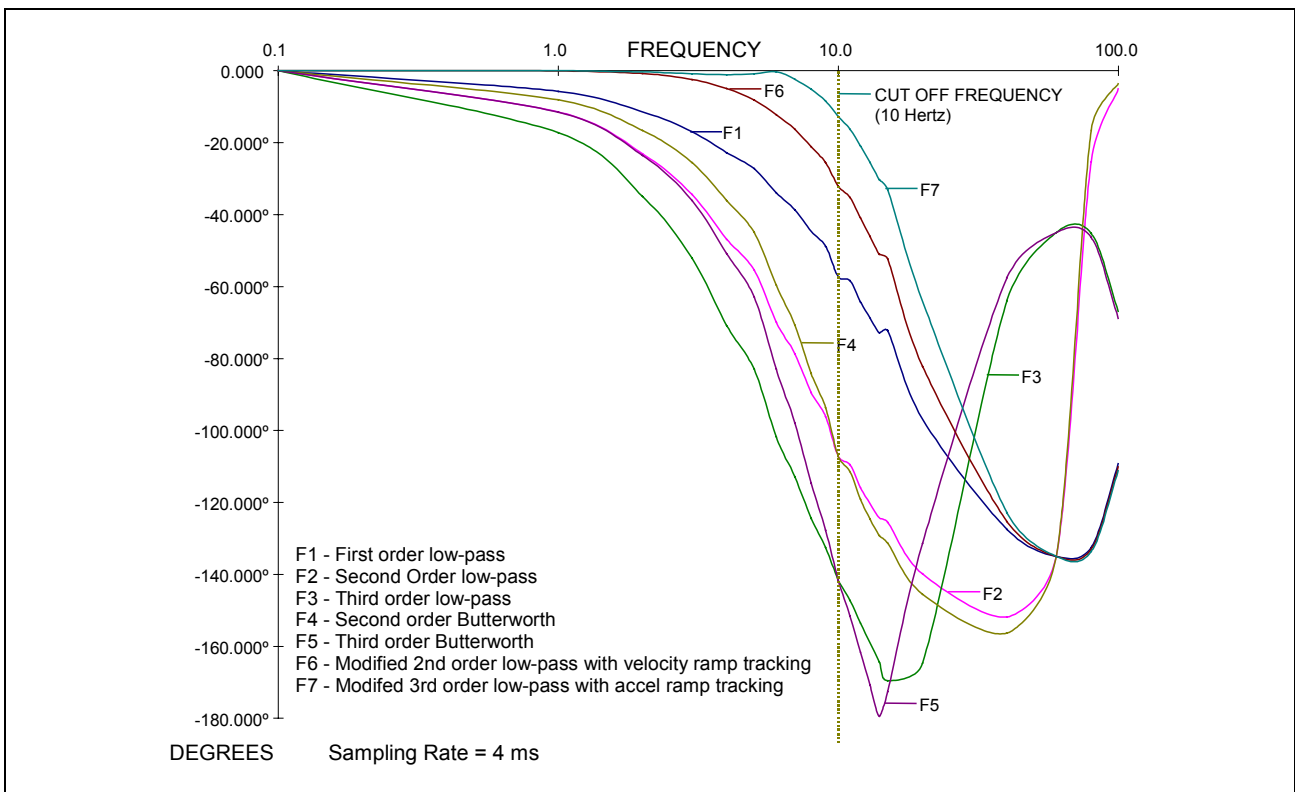


Fig. 2-152: Frequency vs. Degree Filter Chart

The following figure illustrates the gain vs. frequency for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

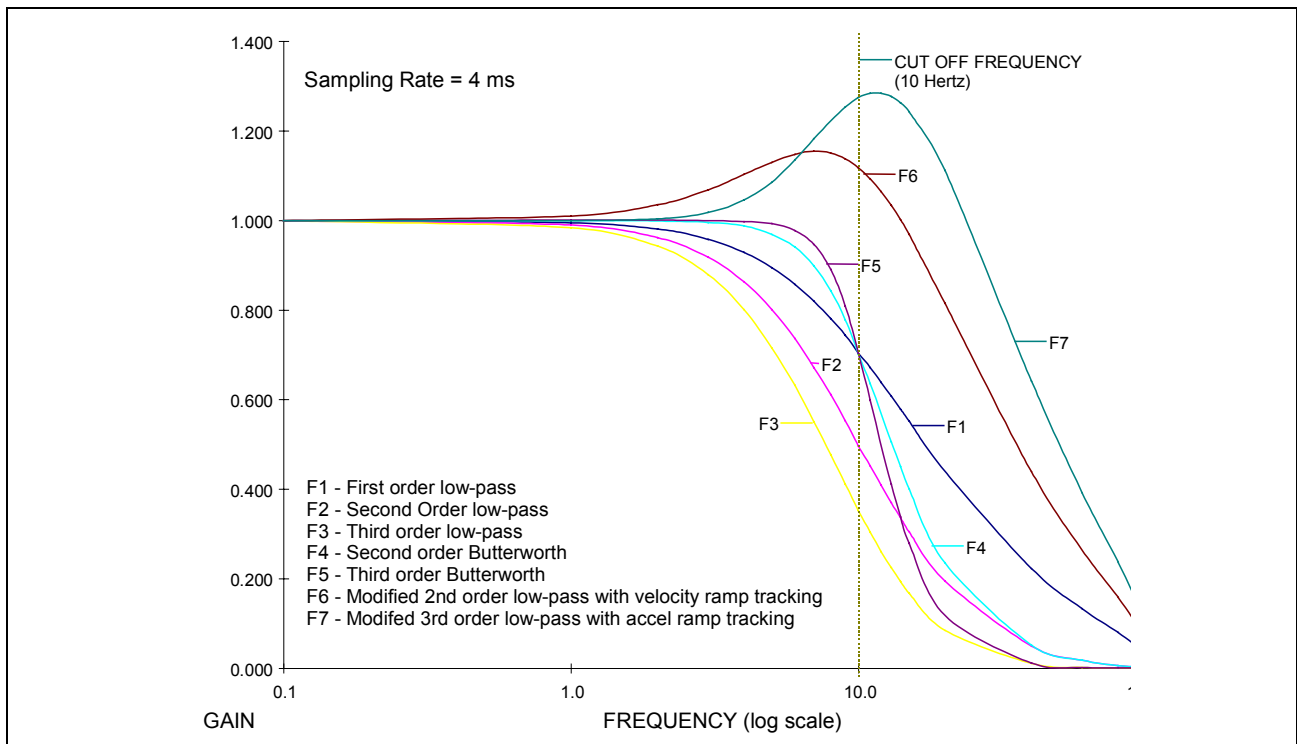
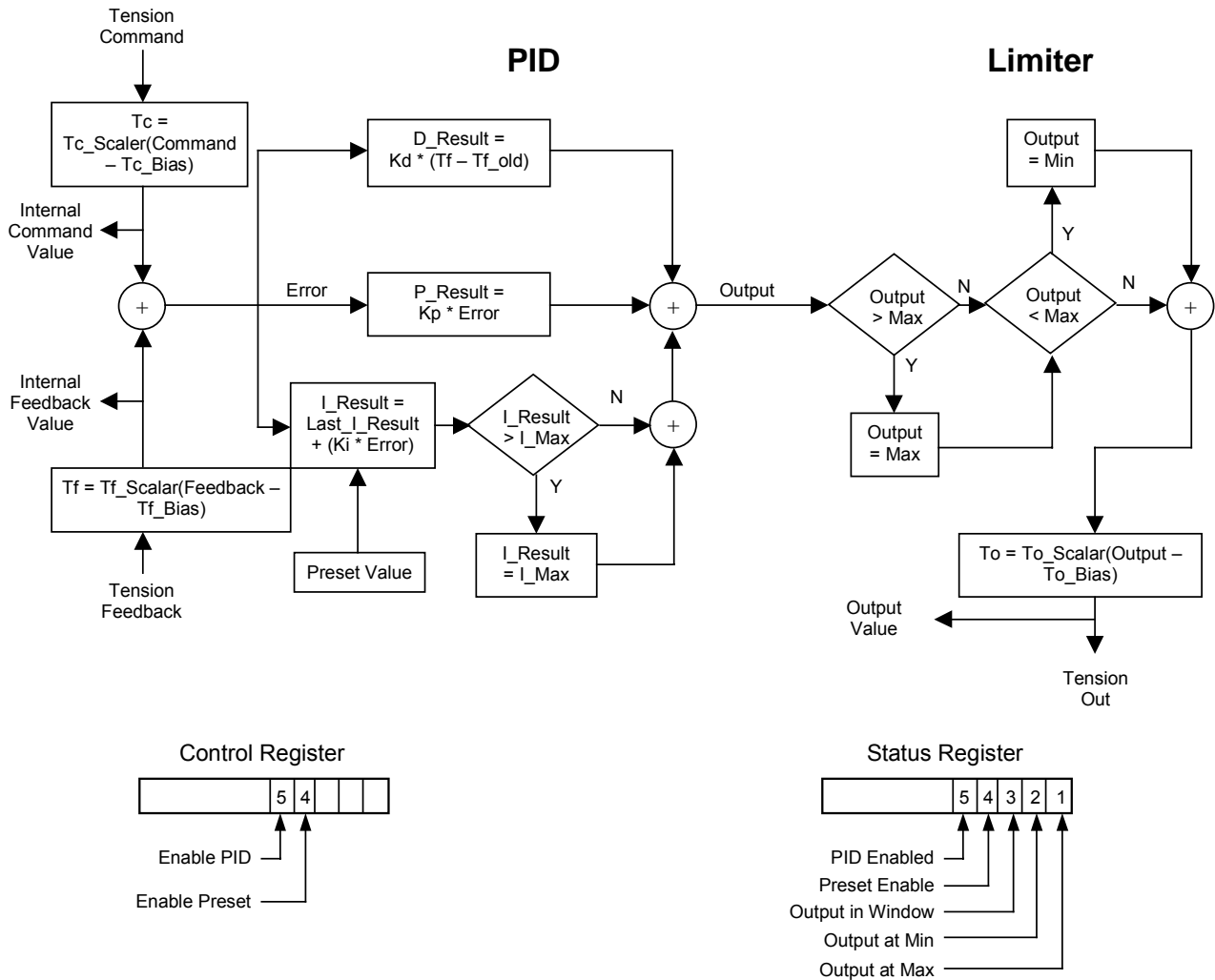


Fig. 2-153: Gain vs. Frequency Filter Chart

When a filter is chosen, the cutoff frequency for the filter must be entered. For example, the cutoff frequency for the First order low-pass filter is the frequency where the signal is reduced 3db [$.707 \text{ gain, } \text{db}=20*\log(\text{gain})$].

PID Instruction

The PID instruction configures a PI or PID control loop. The set point, feedback, and output variables can be registers, integers, floats, or parameters; appropriate conversions are supplied. Control factors (K_i , K_p , K_d , Last_I_Result_Preset) and limits (min., max.) can be constants or variables. Minimum loop update time is 8 milliseconds. In operation, the PID instruction only needs to be executed once in the program flow. The label for the PID loop is its control registers label.



Control Register

Bit 4 Enable Preset - If set, loads integral preset, on program activation or when loop enabled by bit 5.

Bit 5 Enable PID - If set, enables loop, clearing it, disables it. This bit is checked every SERCOS update.

Status Register

Bit 1 Output above max – If the output goes above the user defined max value, this bit will be set.

Bit 2 Output below min - If the output goes below the user defined min value, this bit will be set

Bit 3 Output in Window – If the output is at or above the min value and at or below the max value, this bit will be set. It is redundant to bits 1 and 2.

Bit 4 Enable Preset - Acknowledgment of control register preset bit(4).

Bit 5 Enable PID - Acknowledgment of control register enable bit(5).

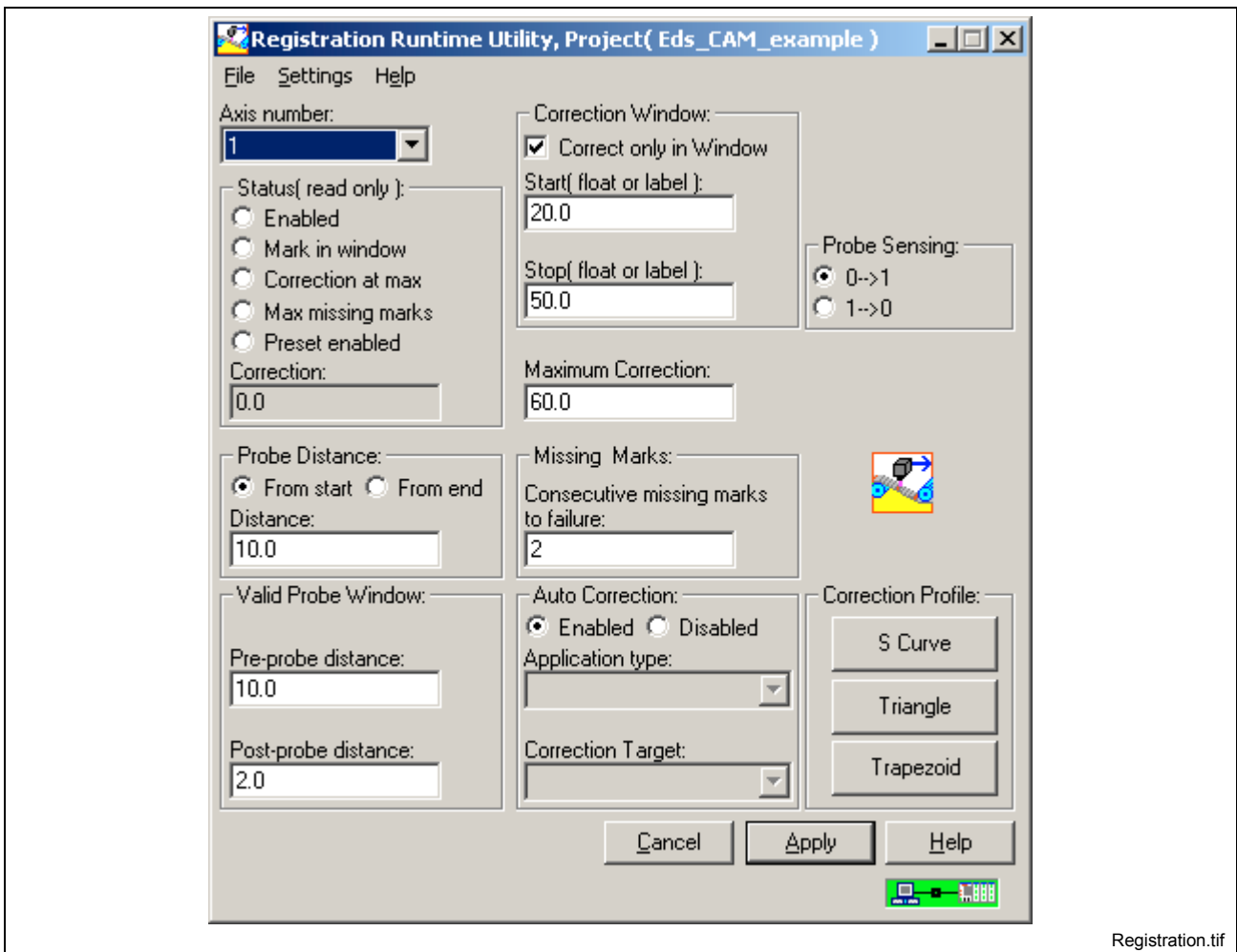
Block of 20 program float variables per PID loop (Fx). (Type 1 PID usage)

Float	Description	Value
F550	Command scalar value, default 1.0	4
F551	Command bias value, default 0.0	4
F552	Feedback scalar value, default 1.0	4
F553	Feedback bias value, default 0.0	4
F554	Kp value, default 1.0	4
F555	Ki value, default 0.0	4
F556	Kd value, default 0.0	4
F557	Ki limit value, default 0.0	4
F558	Minimum output value, default -10.0	4
F559	Maximum output value, default 10.0	4
F560	Preset value, default 0.0	4
F561	Output scalar value, default 1.0	4
F562	Output bias value, default 0.0	4
F563	Feedback cutoff frequency (Hz), default 0	4
F564	Feedback filter type, default 0 0 = None 1 = First order low-pass 2 = Second order low-pass 3 = Third order Butterworth 4 = Second order Butterworth 5 = Third order Butterworth 6 = Velocity tracking 2nd order 7 = Accel ramp tracking 3rd order	4
F565	Internal feedback after conditioning with bias, scalar and filter	4
F566	Internal command value after conditioning with bias and scalar	4
F567	Output value	4
F568	Reserved	4
F569	Reserved	4

Table 2-15: PID Default Program Variables

Registration

Selecting **Data** ⇒ **Registration** in opens the *Registration Runtime Tool* window.



Registration.tif

Fig. 2-154: Registration Utility

Note: Registration is configured in the *Index* icon by checking the **Enable Registration Correction** checkbox and then clicking on the **Registration Correction Setup...** button.

Registration is the process of referencing a edge of a product or register mark, such that any error different from the set-point can be corrected before a downstream process takes place. Based on the machine design, either the web/product or the downstream process (die-cutter, print cylinder, etc.) will be positioned correctly. This function is tightly coupled with the probe event on the drive to which the product is being referenced and the control system. Registration is accomplished by comparing the captured position to a target value and correcting for the difference. The registration error is automatically assimilated into the axis motion profile, providing a smooth, seamless correction. The registration error can be corrected using S-curve, Triangular and Trapezoidal correction profiles.

The Registration function integrates several tasks. First, it automatically executes a high priority process that is triggered by an axis probe event. Then, based on the settings of the drive, system and registration parameters, this process generates a registration error. It automatically reconciles such details as rollover, probe On/Off position window, setting

drive parameters (edge, arming, etc...), and routing the correction data to the appropriate axes.

The Registration setup user program command assigns variables, I-O registers and initializes axis parameters. All values are assigned to blocks of floating point and integer variables, so that they may be read or written from a user interface or a PLC with no additional program code.

When the CAM Indexer is used, it performs the registration correction with a profile based on the *Correction Distance*. The phase offset, equal to *Correction Distance*, is applied according to the *Correction Procedure* and *Correction Maximum* parameters.

The registration function can be used with the following Bosch Rexroth drive firmware versions:

DIAX04 (ELS-05VRS or later)

ECODRIVE03 (SGP-01VRS or later)

Status (read only)

This setting reads the state of the status register setup for the registration functionality.

Enabled

This bit is an acknowledgment of the Enable Registration control bit. Its value will not match that of the control bit until the control has completed the process of enabling or disabling registration. This process could take several milliseconds, depending on the parameters that need to be initialized.

Mark in Window

This bit is set to (1) as soon as a registration mark has been detected within the probe window. It is cleared when registration is first enabled, or on the following cycle when no registration mark is found in the window. This bit can be mapped to a task input or system input event, so that the user can run additional program code when the mark is detected.

Correction at Max

This bit is set to (1) when the *Correction Distance* exceeds the *Correction Max* parameter. It is reset to 0 after the next mark is found and *Correction Distance* is less than or equal to *Correction Max*.

Max Missed Marks

A missed mark counter is incremented when the probe window is traversed without a mark being found. This bit is set when the counter exceeds the *Missed Marks* parameter. The user program or PLC can then take appropriate action or alert the user of this condition. This bit is reset by the control only upon a 1-0 transition of the Registration Enable bit.

Preset Enabled

This bit is an acknowledgment of the Enable Preset control bit.

Probe Distance

This function is to select the distance (from start or from end) within the move cycle that the registration mark can be read. The probe position is captured and the correction distance is calculated only when the mark is found within this position range. The probe window is related to the selected measurement value (master or axis position).

Valid Probe Window

This window selects the period within the move cycle that the registration mark can be read. The probe position is captured and the correction distance is calculated only when the mark is found within this position range. The probe window is related to the selected measurement value (master or axis position).

Correction Window

This function is to select the period in the move cycle to which the registration correction is to be applied. If the option *Correct only in Window* is not selected, the correction is applied immediately after the probe position value is captured. Otherwise, the correction starts at the *Start* position, and must be finished before the *Stop* position.

When using the CAM indexer, the start and end position in relation to the master axis is entered here. Correction motion will only take place within this window. If the probe comes in during the correction window, say at position *x* of the master, then the maximum allowable correction during that cycle is calculated as follows:

$(\text{maximum correction})(x - \text{start})/(\text{end} - \text{start})$

Select Correction Target - These bits select the target parameter for the *Correction Distance*. When using the CAM indexer, the correction uses a CAM, which is added to the CAM position generated by the indexer.

Maximum Correction

This is the maximum amount of correction distance that will be applied to the current index in units. If zero, the full amount of the correction is used for the current index. If a value is entered, the *Correction Distance* value is compared to *Correction Maximum* and correction up to this value will be applied. The remainder will be discarded.

Missing Marks

Enter the number of missing marks until an error is issued.

Auto Correction

Select enabled or disabled. By default, the control sends the *Correction Distance* to the drive to perform a phase offset, based on the correction window and correction target. It is possible to disable automatic correction, so that it can be handled in the user program. This option applies to all motion types except for the CAM indexer, for which automatic correction is always enabled.

Probe Sensing

Set the sensing to be either on the positive transition ($0 \Rightarrow 1$) or a negative transition ($1 \Rightarrow 0$).

Correction Profile

When using the CAM indexer, this value defines the shape of the correction CAM. Current selections are S-curve, Triangular and Trapezoidal.

Zones

Selecting **Data** \Rightarrow **Zones** opens the *Zones Runtime Tool* window. This utility allows viewing and editing of the zone table on the control. Zones can be used in coordinated motion programs to describe a volume of space where motion of any kind is prevented.

Programs on control - program selection is through a menu selection containing a list of programs on the control plus "Currently active".

To edit an event, select it in the list box, and press the **Edit** menu selection, or double click on the list item. Modify the necessary fields and press the **Apply** button to save the changes and keep the edit window open or click the **OK** button to save the changes and close the edit window.

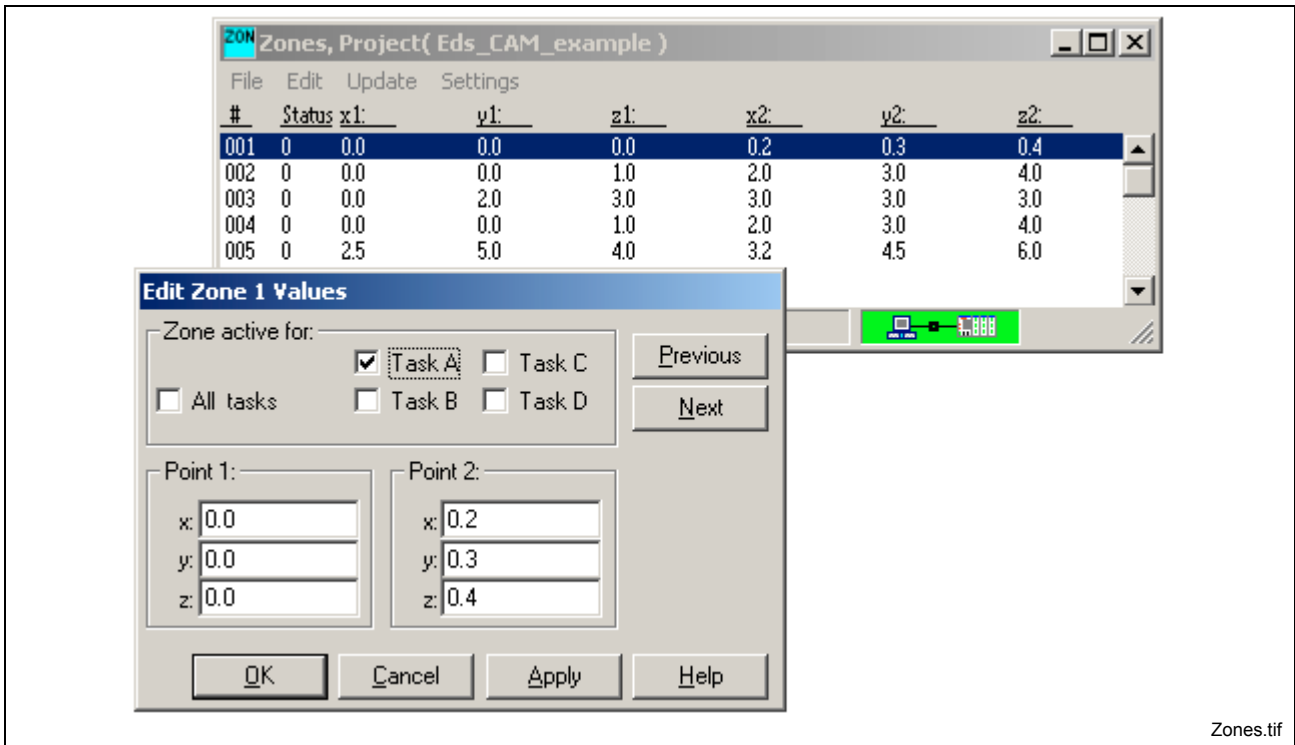


Fig. 2-155: Zones List

Each zone is defined by the { x, y, z } coordinates of two opposite corners (Point 1 and Point 2) of a cube in space. For each defined zone, the status can be ACTIVE (checked) or INACTIVE (unchecked) for selected tasks, no task or all tasks. For example, the zone defined in the "Edit Zone 2 Values" screen below is active for zones A and D.

2.9 The Diagnostics Menu

The **Diagnostics** menu is used to monitor system information such as system status, drives and tasks. The user can also analyze a signal using the Oscilloscope tool, and program flow by enabling a Breakpoint.

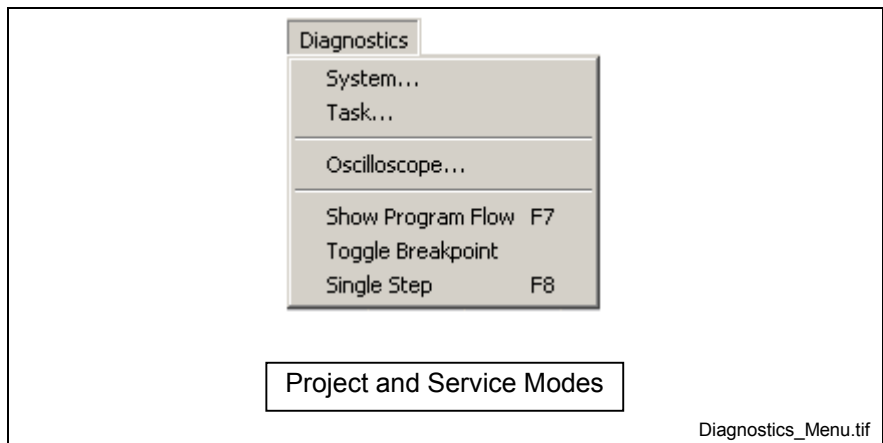


Fig. 2-156: The Diagnostic Menu

System

Selecting **Diagnostics** ⇒ **System** opens the *System Diagnostics* window. This window displays diagnostic information about the active VisualMotion system. The window uses tabs to display diagnostic information for the following:

- Status
- General
- Option Cards
- Diagnostic Log
- Hardware
- Load (System)

Note: Holding the cursor over any field will display the corresponding parameter number.

Status

The *Status* tab displays the current diagnostic message with an extended message, if available. The system's current mode and SERCOS phase is also displayed.

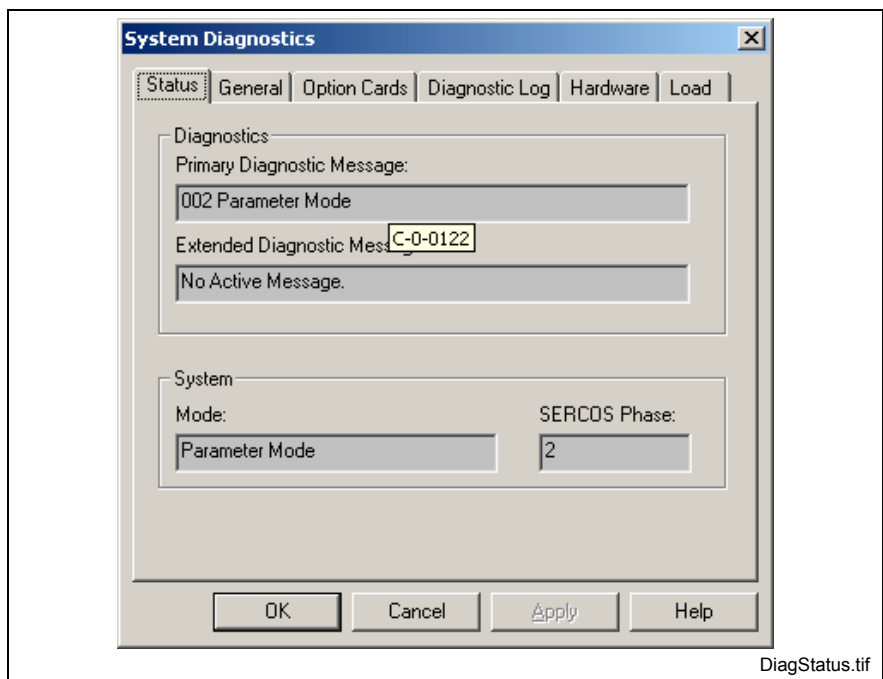


Fig. 2-157: Status

General

The *General* tab displays the control's address and label, hardware typecode, firmware version, firmware compile date and boot loader version.

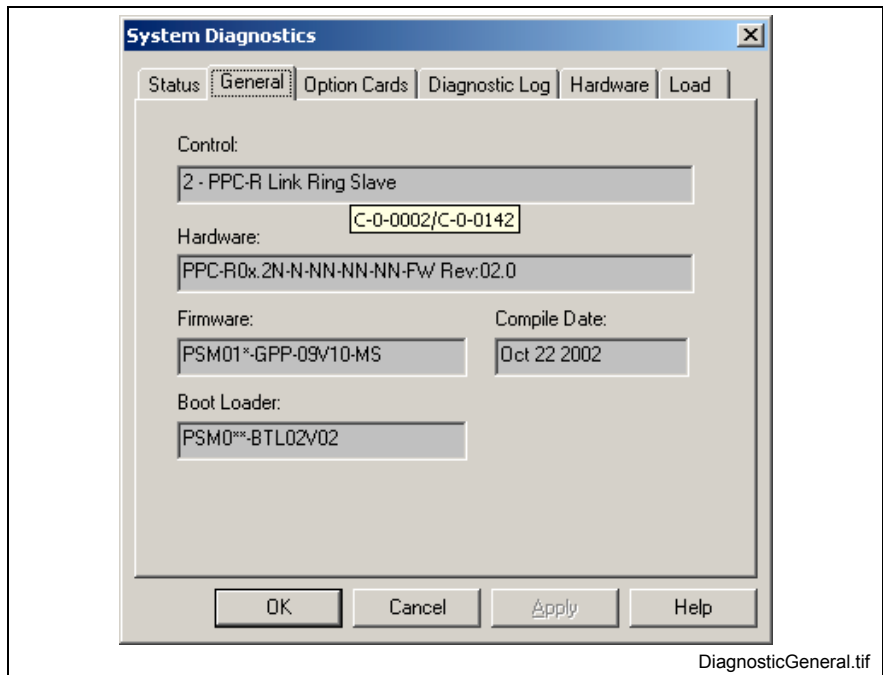


Fig. 2-158: System

Option Cards

The Option Card tab displays the current hardware card(s) installed in the control. Refer to the *VisualMotion 9 Project Planning* manual for a complete listing of available hardware configurations.

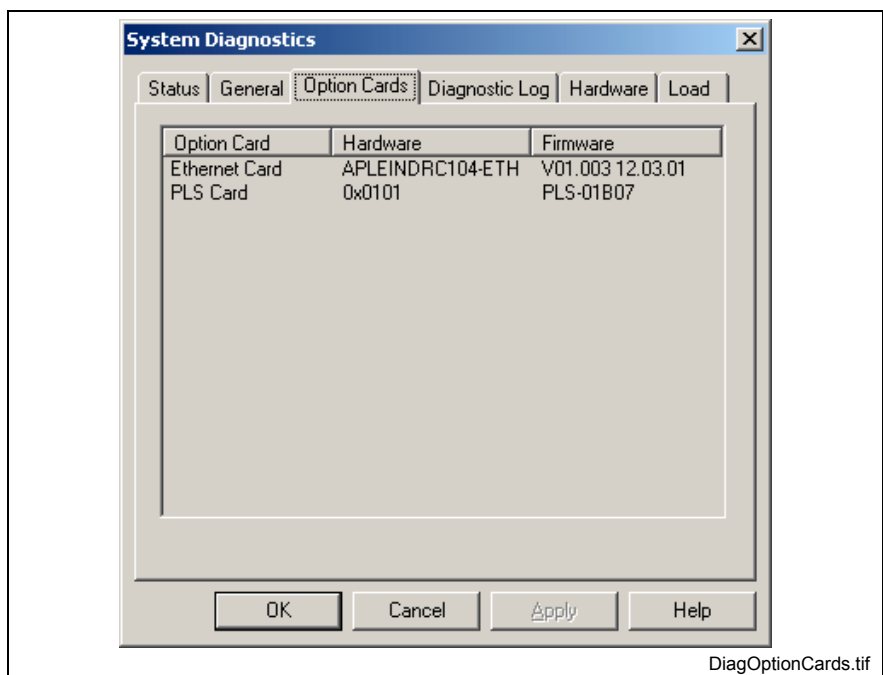


Fig. 2-159: Option Cards

Diagnostic Log

The Diagnostic Log tab displays the last 100 errors the control has encountered. Along with the error messages, the date, time and extended error codes are displayed.

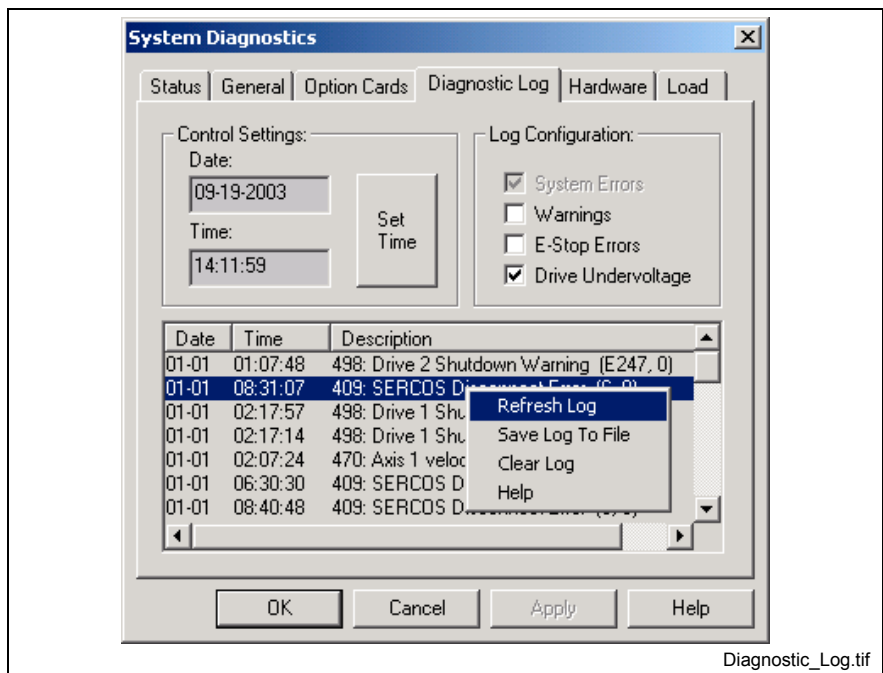


Fig. 2-160: Diagnostic Log

Control Settings

The date and time are relative to the power on of the control; they have no battery backed up clock. The time can be set to the computer's clock by clicking the **Set Time** button.

Note: To maintain the control's date and time current even after the power is recycled, equip the control with an optional SUP battery kit. Refer to the VisualMotion 9 Project Planning manual (DOK-VISMOT-VM*-09VRS**-PR02-EN-P) for details.

Log Configuration

The user can select which errors are to recorded and displayed by checking the appropriate checkbox.

Log Data

The last 100 control errors are displayed and can be sorted by Date, Time or Description by clicking the appropriate heading.

An option pop-up window is displayed by right clicking the mouse. From here, the user can refresh the current display, save the log to a file, clear the log or request help on the selected diagnostic.

Note: Help on any displayed diagnostic error can be selected from the pop-up window or by double clicking on the desired error in the list.

Hardware

The *Hardware* tab displays control related information, such as current operating temperature and control's total operating time. The memory section displays the control's total memory and available free memory.

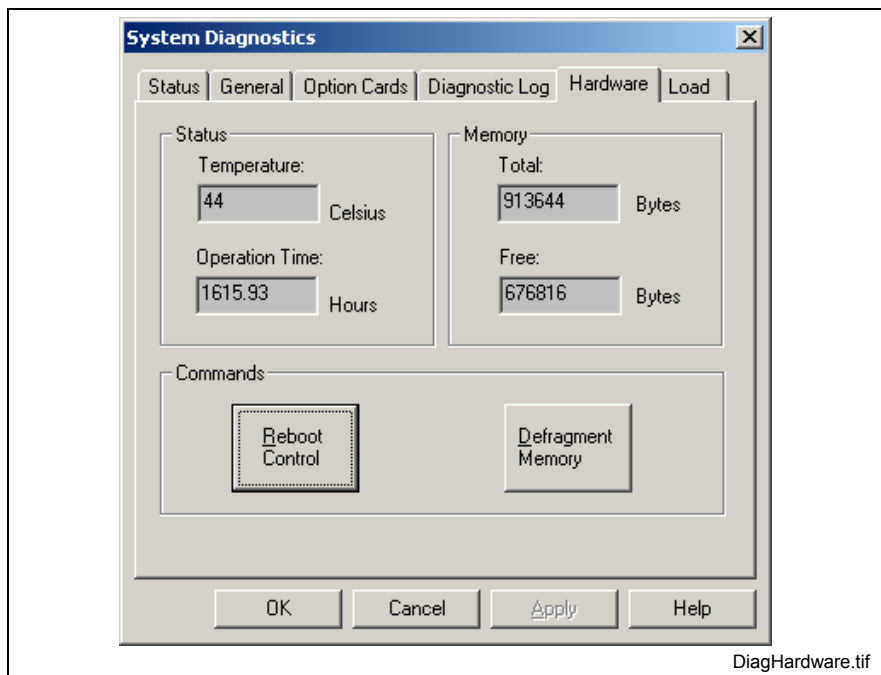


Fig. 2-161: Hardware

Reboot Control

Clicking the ***Reboot Control*** button enables the *Software Reset for Control* command of control parameter C-0-0993. The user is warned before the control is rebooted. The control must be in parameter mode in order to reboot the control using this button.

Defragment Memory

The defragment button is used to request a defragmentation of the control's memory. The determination of a necessary defragmentation is performed by the control and not by clicking on the button. Clicking the button queues the control.

Load

The *Load* tab displays the control's current and peak time required by the processor to process motion task and I/O completion as a percentage of parameters C-0-0099 and C-0-3001, respectively. The User Watchdog Timer section is used to specify a time (in ms) for a selected task to complete its program before an error is issued.

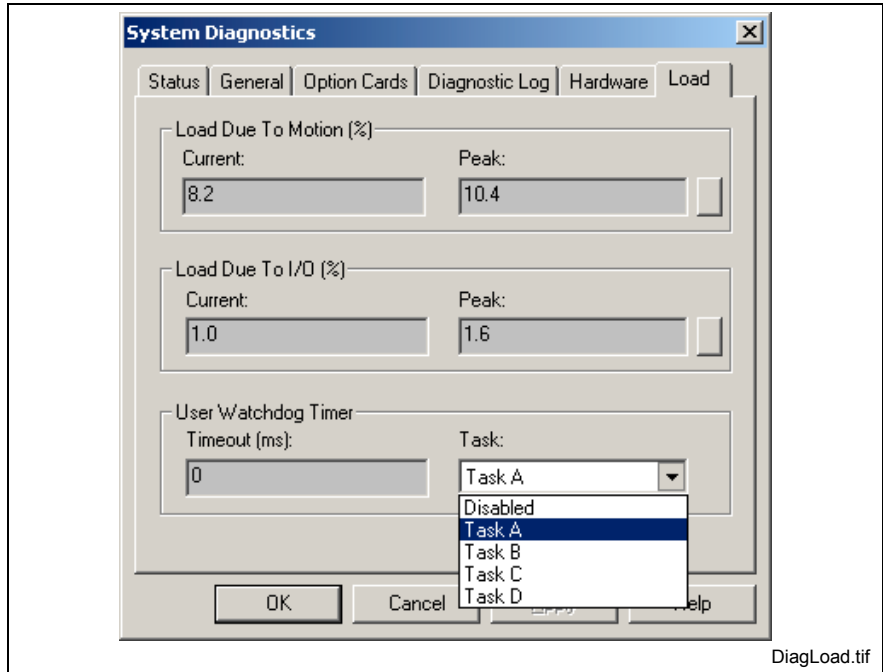


Fig. 2-162: Load

Tasks

Selecting **Diagnostics** ⇒ **Task** opens the *Task Diagnostics* window and uploads data regarding all active VisualMotion tasks. Task letters are displayed only if they contain an icon program that has been compiled and downloaded to the control. All GPP 9 and GMP 9 programs will contain the Initialization Task and Task A tabs. The Coordinated Motion tab is only visible if any task contains a coordinated program. It displays the active coordinated axes and their current X, Y, Z positions.

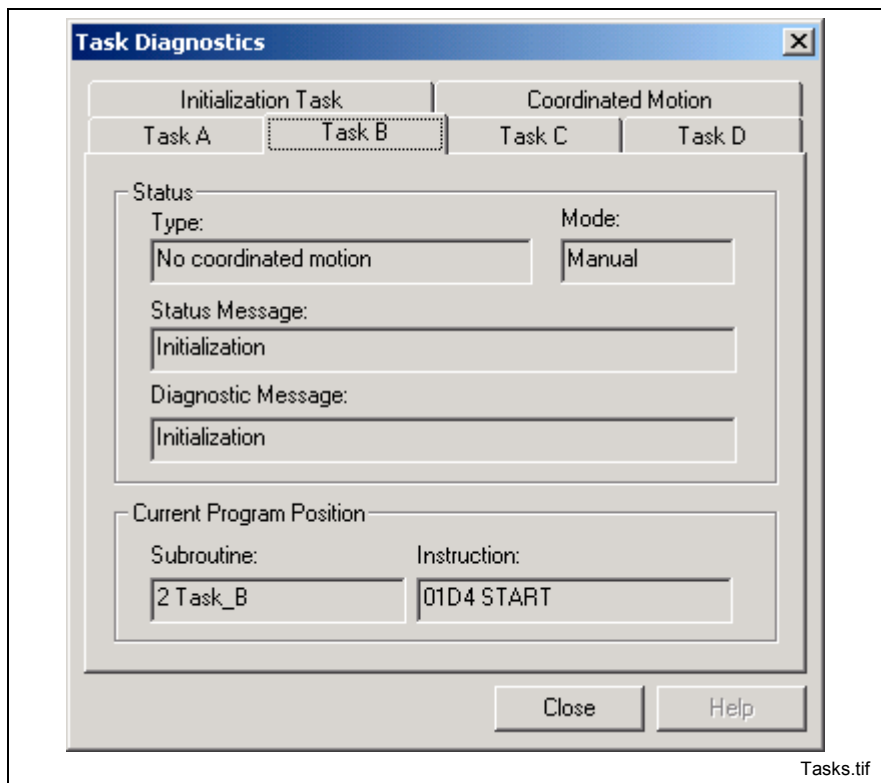


Fig. 2-163: Tasks

Status - indicates the type of motion programmed in the selected task for the active program and the current control mode (Parameter, Initialization, Manual or Automatic). Status and Diagnostic messages for the selected task are also displayed.

Current Program Position - displays the subroutine and instruction executing and its pointer. This display is useful when debugging in single-step mode. If a program is running in automatic mode, the displayed instruction is the instruction that was executing at the time that the operating system sampled the instruction execution.

Oscilloscope

VisualMotion's oscilloscope function is used to capture predefined internal and external signals from the control or connected drive(s). Control signals, that are valid oscilloscope signals, are parameterized and stored in the control. Drive signals are parameterized and stored in the drive, and are dependent on whether or not the connected drive supports the oscilloscope function.

Feature	Control	Drive	Description
Configurable Signals	4	2	
Sample Rate	2ms, 4ms, 8ms, 16ms, 32ms, and 64ms	250µs, 500µs, 1ms, 5ms, 10ms, 50ms	time interval for signal capture
Sample Count	100, 200, 500, 1000, 2000, and 4000	50, 100, 200, 300, 400, and 500	number of captures for entire trace

Table 2-16: Oscilloscope Features

Oscilloscope Signals

Up to 4 different oscilloscope control signals can be parameterized. Three parameters are required to identify each oscilloscope signal. The following table lists the parameters for each signal:

Signal Number	Parameters
Signal 1	C-0-2501, C-0-2504, and C-0-2507
Signal 2	C-0-2502, C-0-2505, and C-0-2508
Signal 3	C-0-2503, C-0-2506, and C-0-2509
Signal 4	C-0-2524, C-0-2525, and C-0-2526

Table 2-17: Oscilloscope Signal Parameters

Drive signals, that are valid oscilloscope signals, are parameterized and stored in the drive. Refer to the specific drive firmware functional description manual for oscilloscope parameters.

Timing and Pretrigger Options

The capture duration, of configured oscilloscope signals, is calculated by multiplying the following two control parameters:

Parameter	Description
C-0-2510	Oscilloscope Sample Rate
C-0-2514	Oscilloscope Sample Count

Table 2-18: Oscilloscope Timing Parameters

For example, $(C-0-2510 = 8ms) * (C-0-2514 = 500) = 4000 ms$

Pretrigger

A pretrigger allows the capture of data for all configured signals before the trigger point. The pretrigger value is stored in control parameter **C-0-2515 Oscilloscope Trigger Post-Count**.

Enable Trigger and Upload

The recording of signal traces begins when the oscilloscope trigger is enabled. Triggers can be configured as either user or internally initiated.

Selecting **Diagnostics** ⇒ **Oscilloscope** opens the *Oscilloscope* window. The oscilloscope utility is used to capture and display run-time data when in online mode. In offline mode, saved captures can be opened and viewed. The capture can be of the control or on a drive that supports this feature. Selected data is acquired on the drive or control, passed to VisualMotion Toolkit, and displayed on the graphical format. The graphical display and supporting data can be printed, or the data can be saved to a file for later review.

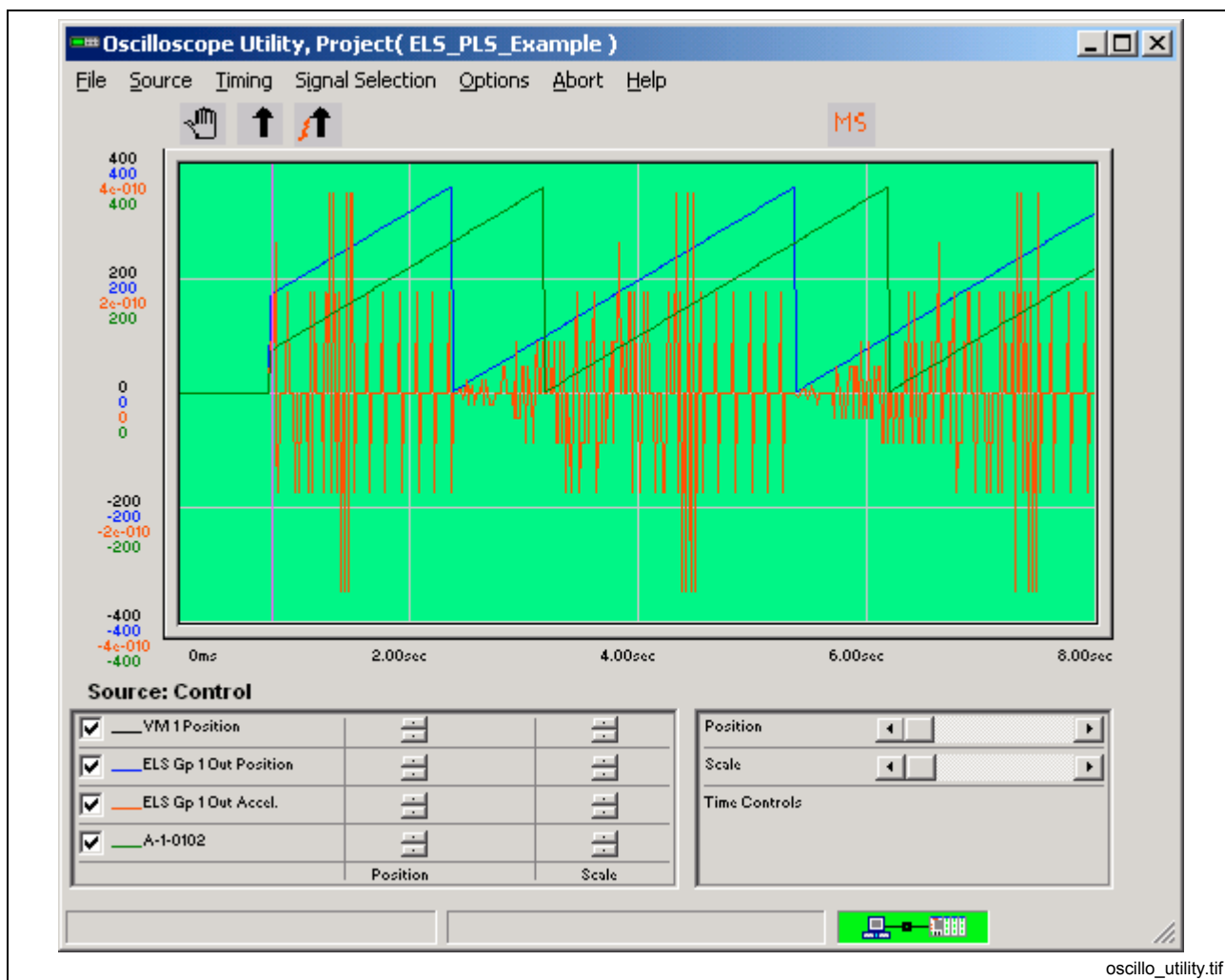


Fig. 2-164: Oscilloscope Utility

File Menu

The File menu is used for retrieving file data, saving data to a file, printing, and exiting the oscilloscope utility.

Open - data from user selected input file is loaded into input data list-box.

Save - data from user selected output is loaded into output data list-box.

Print Output - the oscilloscope graph and its related data table is sent to the printer.

Exit - terminates this utility

Source Menu

When using VisualMotion Toolkit's oscilloscope utility, only those drives that support the oscilloscope function will be available as a signal source.

Selects the source from which the oscilloscope will gather signal data.

Drive 1 to n - lists of all drives on the SERCOS ring that supports the oscilloscope feature.

Control - when selected, the oscilloscope can then read control variables, parameters and registers.

Timing

The Oscilloscope timing options in Fig. 2-165 are used for setting the **Sampling Rate** (How often a trace is captured) and **Sample Count** (How many sampling rates are captured). The **Capture Duration** field displays the total capture duration that is calculated by multiplying the Sample Count and Sampling Rate. A **Pretrigger** can be added and is a percentage of the capture interval.

The Sample Count can be set to a maximum of 4000 for control signals and 500 for drive signals.

The Sampling Rate can be set to a maximum of 64ms for control signals and 50ms for drive signals.

A maximum count and rate setting would generate a total capture duration of 256,000 ms (4 min – 16 seconds).

Note: The pretrigger appears on the oscilloscope screen as a vertical line.

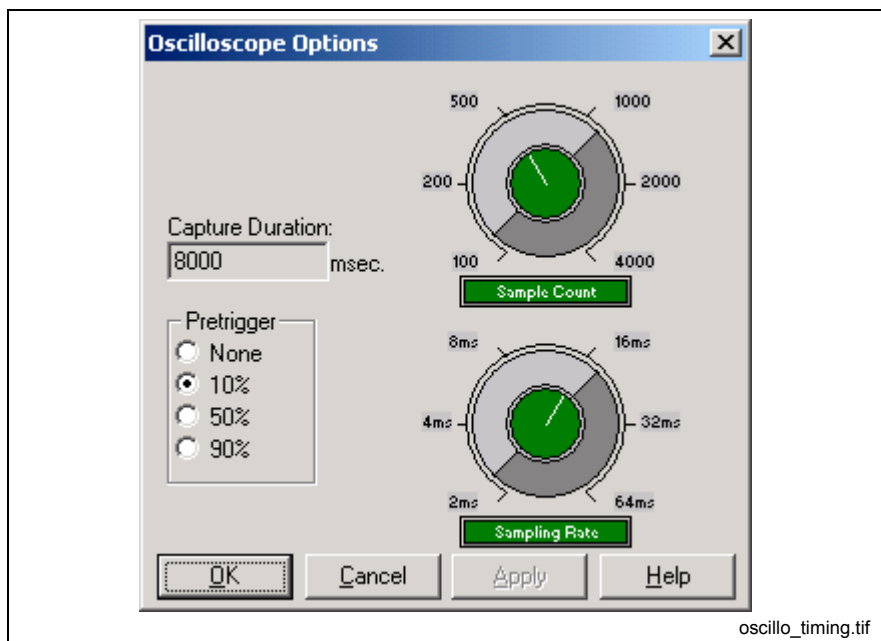


Fig. 2-165: Oscilloscope Options

Signal Selection

The Drive Signal Setup in Fig. 2-166 is available when a drive is selected under the Source menu.

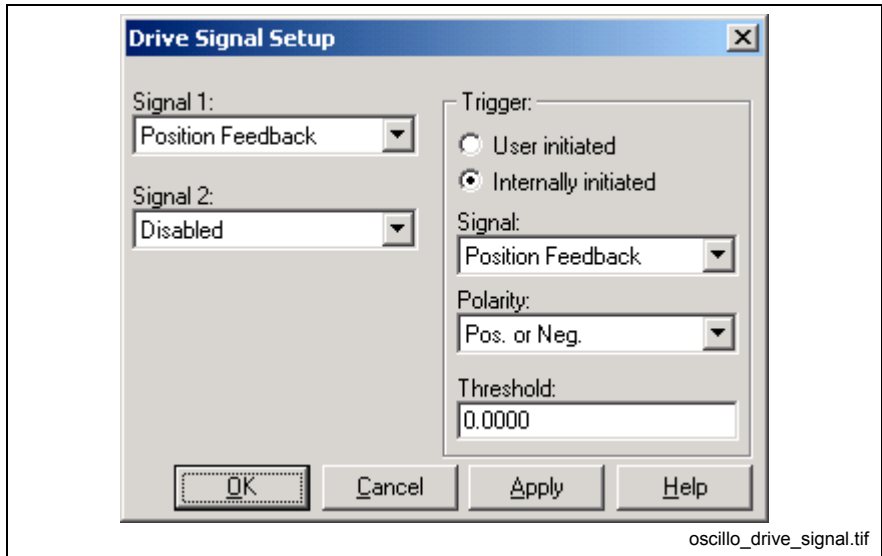


Fig. 2-166: Drive Signal Selection

Two drive signals can be captured and viewed. The following is a listing of the available drive signals.

- Position Feedback
- Velocity Feedback
- Velocity Deviation (from commanded value)
- Position Deviation (from commanded value)
- Torque Command Value (required to maintain the commanded Velocity/Position)
- Disabled (Signal 2 only)

The Control Signal Setup in Fig. 2-167 is available when a Control is selected under the Source menu. Up to 4 different control signals can be selected and compared.

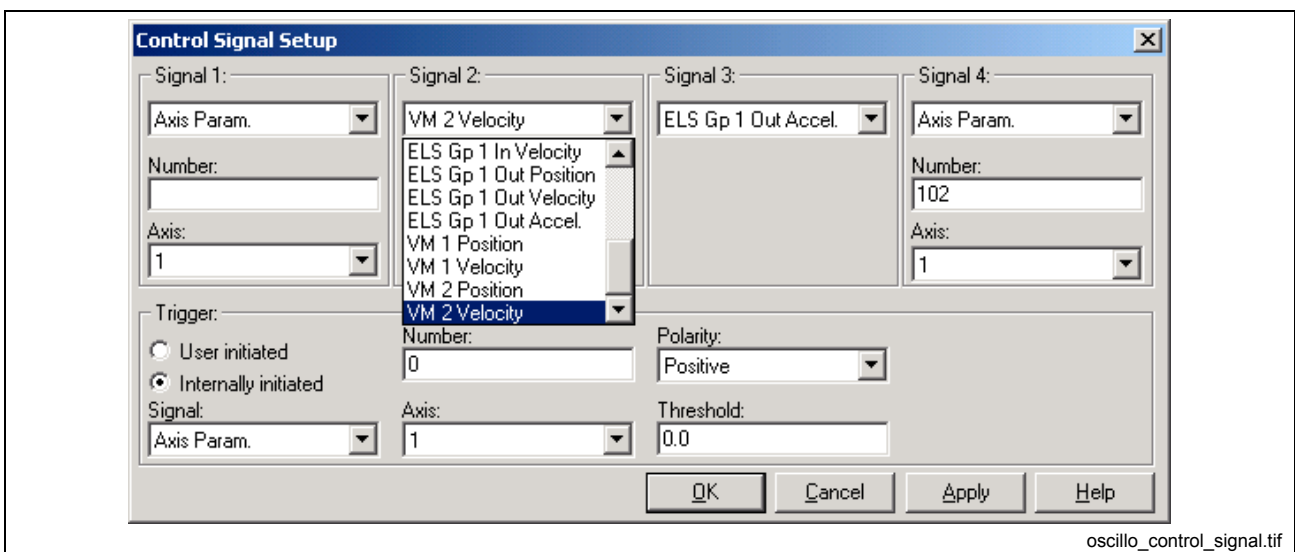


Fig. 2-167: Control Signal Selection


- Program Floats (Fx)
- Program Integer (Ix)
- Global Floats (GFx)
- Global Integers (GIx)
- Axis Parameters * of drives on SERCOS ring ^{NOTE}
- Register Bit
- +/- Register (could be used to monitor a register's value)
- Card Param ^{NOTE}
- Task Param ^{NOTE}
- ELS Gp # In Position.**
- ELS Gp # In Velocity.**
- ELS Gp # Out Position.**
- ELS Gp # Out Velocity.**
- ELS Gp # Out Acceleration.**
- VM1 Position (Virtual master signal)
- VM1 Velocity
- VM2 Position
- VM2 Velocity

Note: Only a selective list of axis, control and task parameters can be used as signal types. Refer to control parameter C-0-2504.

*Axis parameter must be in cyclic telegram. Use parameter A-0-0185 and A-0-0195 to add other drive parameters to cyclic data.

**The # symbol represents ELS Groups 1-8. This same signal is available for each ELS Group in the system.

For either signal source, the sample acquisition may be **User initiated** or **Internally initiated**.

For **User initiated** captures, data acquisition starts as soon as the capture button  is pressed. This type of start capture is not deterministic. All other fields in the trigger section are grayed out.

For **Internally initiated** captures, the available signals are the same as the signals for Signals 1 – 4. The heading in the trigger fields will change based on the signal selected. The trigger polarity options are on positive edge, negative edge, or both. Signal threshold is the signal level to trigger.

Options Menu

From the options menu, a trace's appearance can be changed from *Lines* to *Dots*. When combined with the Time Controls feature on page 2-142, the user can scale (zoom) in to reveal the individual dots that makeup the trace.

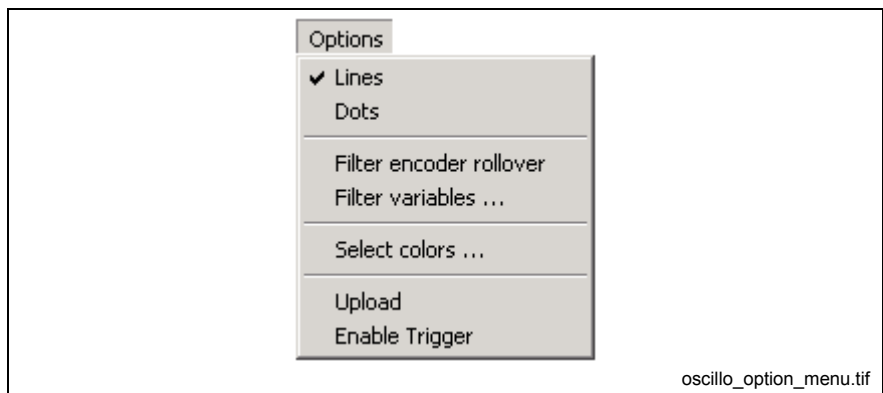


Fig. 2-168: Options Menu

Filter encoder rollover

The position control loop in servo systems is continuously correcting the position of an encoder when at standstill. This continuous correction in position can cause dithering that will be captured by the oscilloscope. When selected, the *Filter encoder rollover* will eliminate any dithering based on the settings of the *Filter variables ...* window in Fig. 2-169.

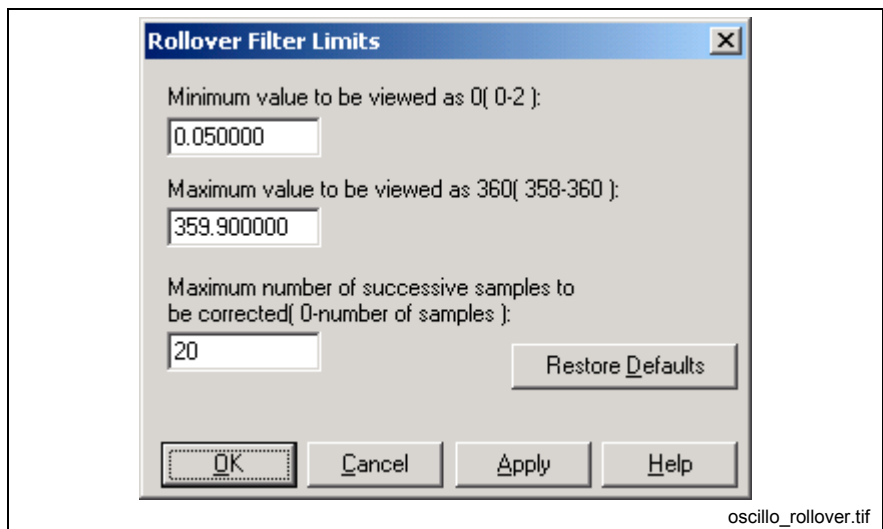


Fig. 2-169: Filter variables

Minimum value to be viewed as 0

A value between 0 and 2 degrees will be used as the filter window for ignoring position dithering. While holding position at 360°, any value between 0 and the minimum value entered will be interpreted as a dither and seen as 360 degrees up to a maximum number of successive samples.

Maximum value to be viewed as 360

A value between 358 and 360 degrees will be used as the filter window for ignoring position dithering. While holding position at 0°, any value between 360 and the maximum value entered will be interpreted as a dither and seen as 0 degrees up to a maximum number of successive samples.

The oscilloscope will immediately capture any position value outside the minimum or maximum filter windows. An example of the filter encoder rollover is shown in Fig. 2-170.

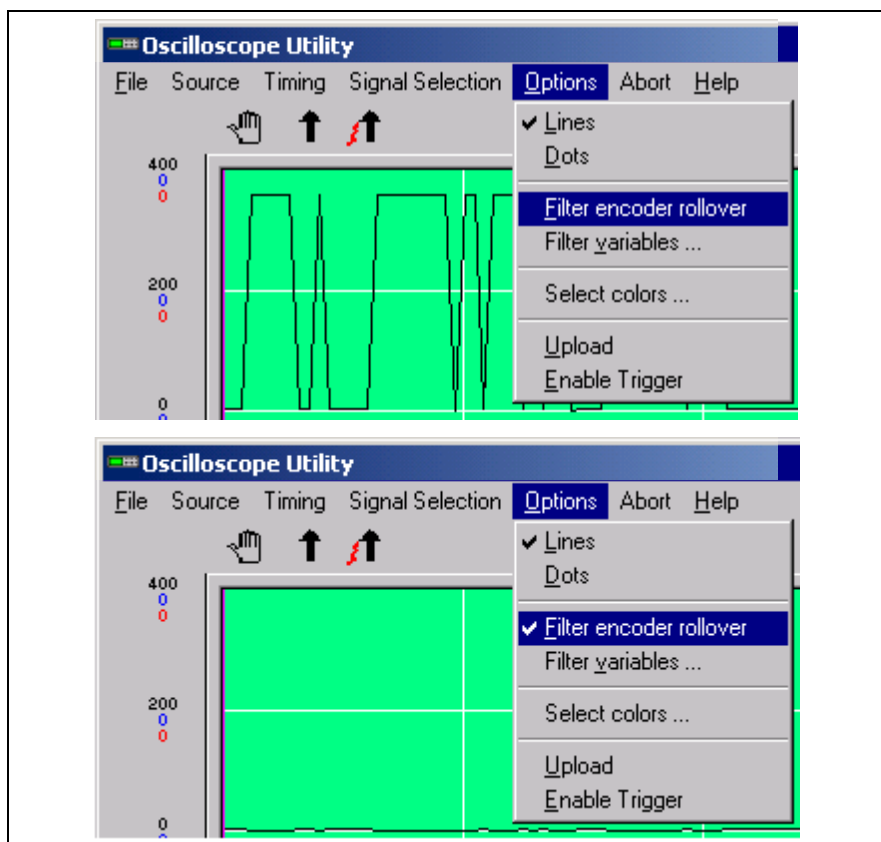


Fig. 2-170: Filter encoder rollover

Select colors...

The three available signal traces are color coded and can be changed for both run time and memory by selecting a color for each trace in the Signal color selection window in Fig. 2-171.

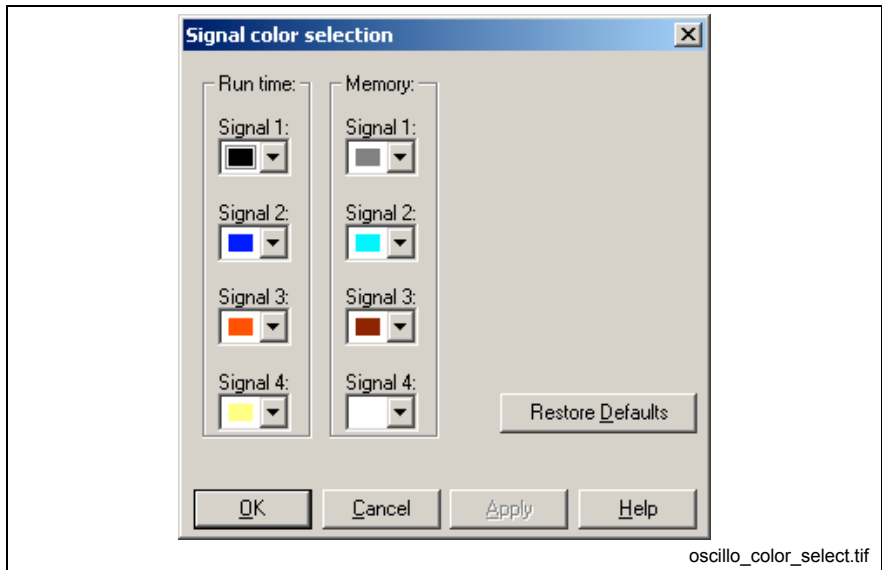


Fig. 2-171: Signal color selection

Abort, Upload and Enable Trigger

The *Abort*, *Upload* and *Enable Trigger* buttons are used to trigger trace captures of configured signals. The *Upload* and *Enable Trigger* functions are also available under the Options menu.

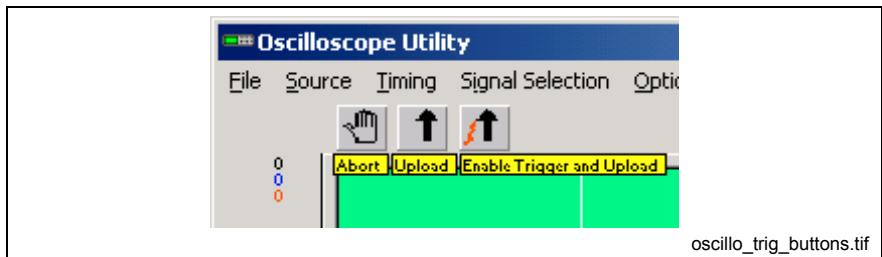


Fig. 2-172: Signal triggering buttons

Oscilloscope memory buttons

Using memory buttons, traces can be stored into memory for viewing and comparing.

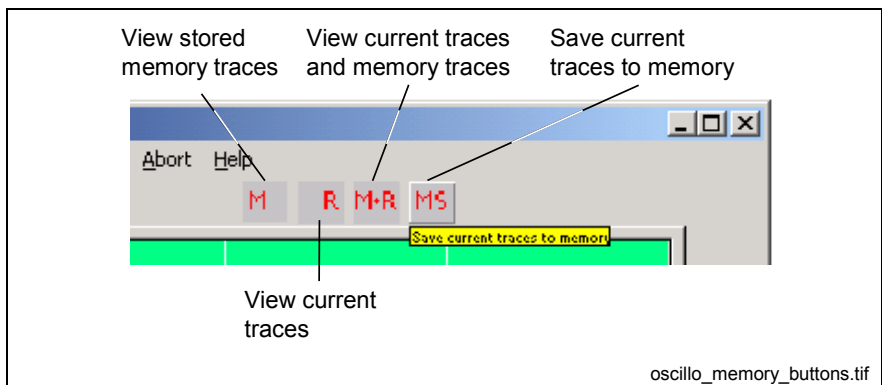


Fig. 2-173: Oscilloscope memory buttons

Manipulating trace signals

When multiple traces are captured at one time, they can be positioned and scaled independently of each other by using the up and down arrows for each trace as shown in Fig. 2-174. The traces can also be turned on or off by clicking on the check boxes to the right of the signal description.

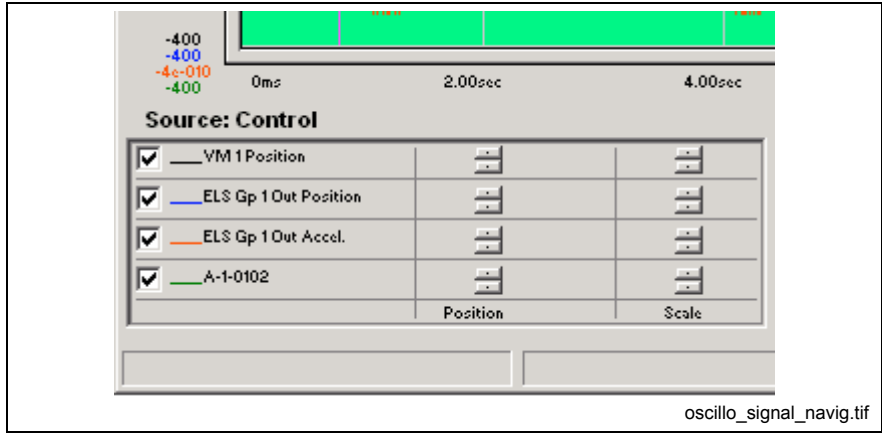


Fig. 2-174: Manipulating trace signal

By positioning the traces above and below each other, the user can more easily distinguish between the signal. Fig. 2-175 shows an example of three traces repositioned for clarity.

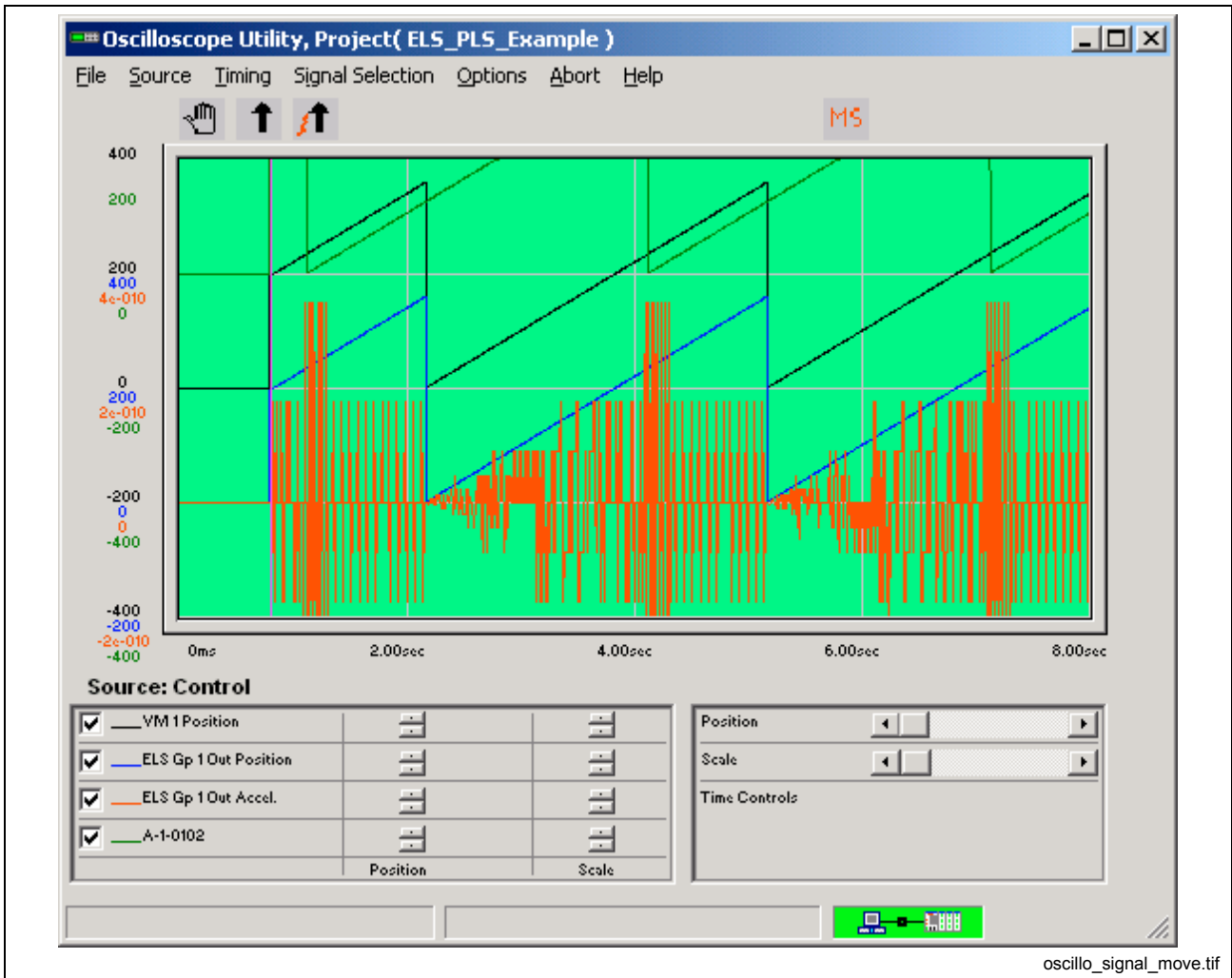


Fig. 2-175: Separated oscilloscope traces

Time Controls

The position and scale functions in Fig. 2-176 are used to analyze a specific area of a captured trace. The scale function acts as a zoom, allowing the user to view smaller sections of a trace, while the position function controls the horizontal scrolling.

Note: The position function only works after using scale.

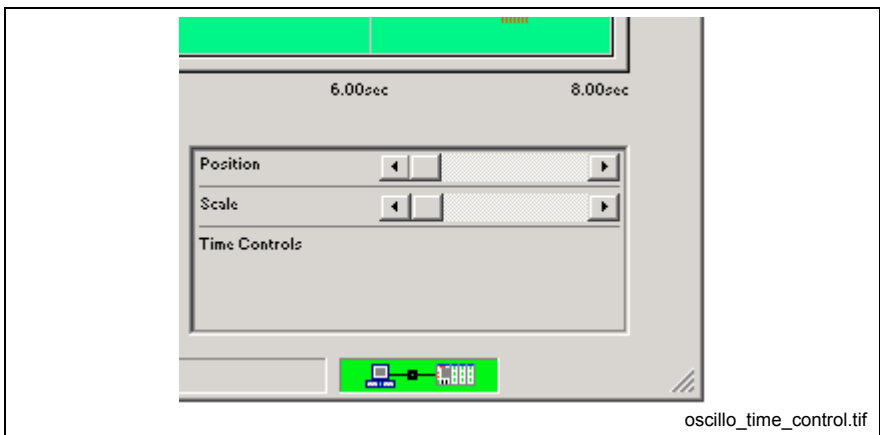


Fig. 2-176: Time Controls

Measuring Trace Signals

Oscilloscope trace signals are measured by placing the cursor at a start position on the graph. When the mouse is clicked and held, a vertical line and small measurement window appear initiating the start of a measurement. As the cursor is moved (while held down), a second vertical line appears.

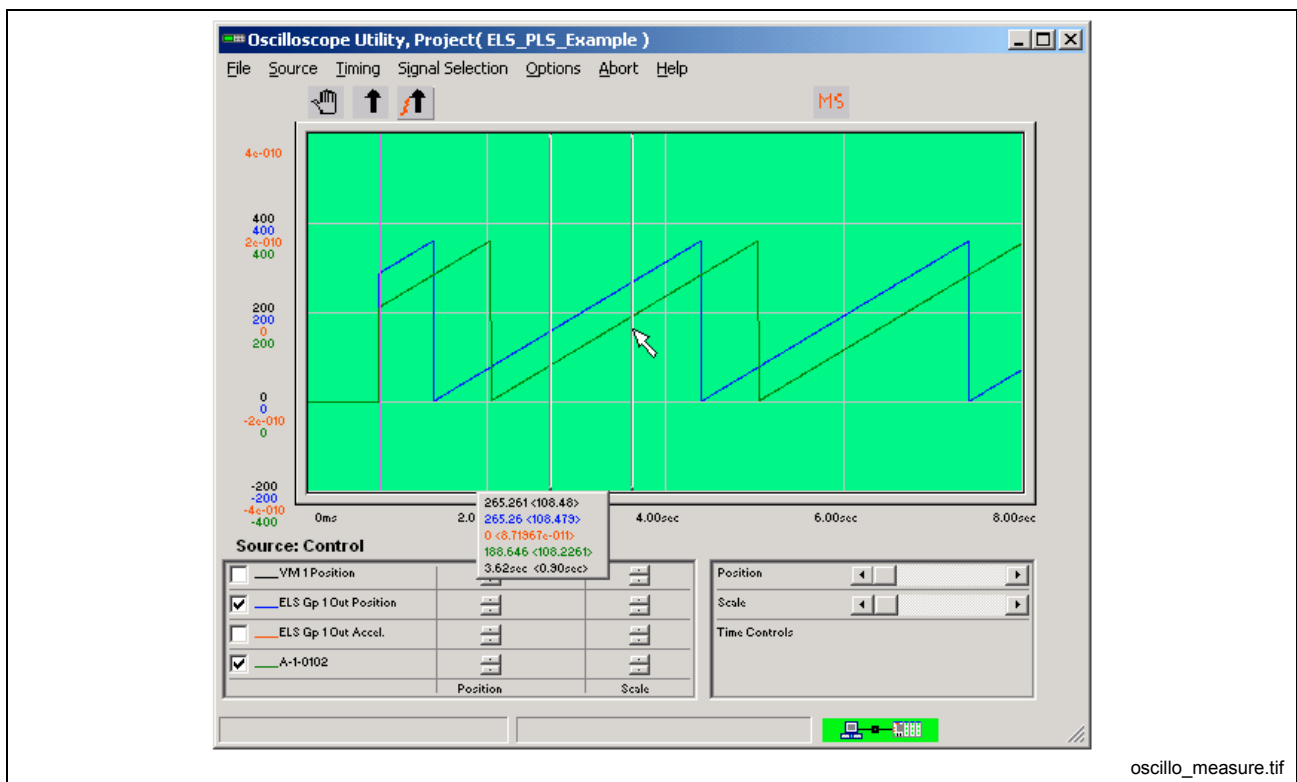


Fig. 2-177: Measuring Signal Traces

The small measurement window simultaneously measures both the current position measurement and a difference measurement from start (in brackets < >) for all 4 signal traces. The first 4 measurements represent the 4 trace signals. The last measurement represents the time line measurement. When the cursor is held and moved, left or right, the measurement is increased or decreased based on direction.

Show Program Flow F7

Selecting **Diagnostics** ⇒ **Show Program Flow F7** highlights the currently executing icon in a running program, as illustrated in Fig. 2-178. During *Show Program Flow*, a check mark appears next to the menu item and other menu items are grayed-out. Re-selecting *Show Program Flow* removes the check mark and re-enables the other menu items.

Note: This menu selection is only available for online mode. Show program flow is not available for *Event Functions*.

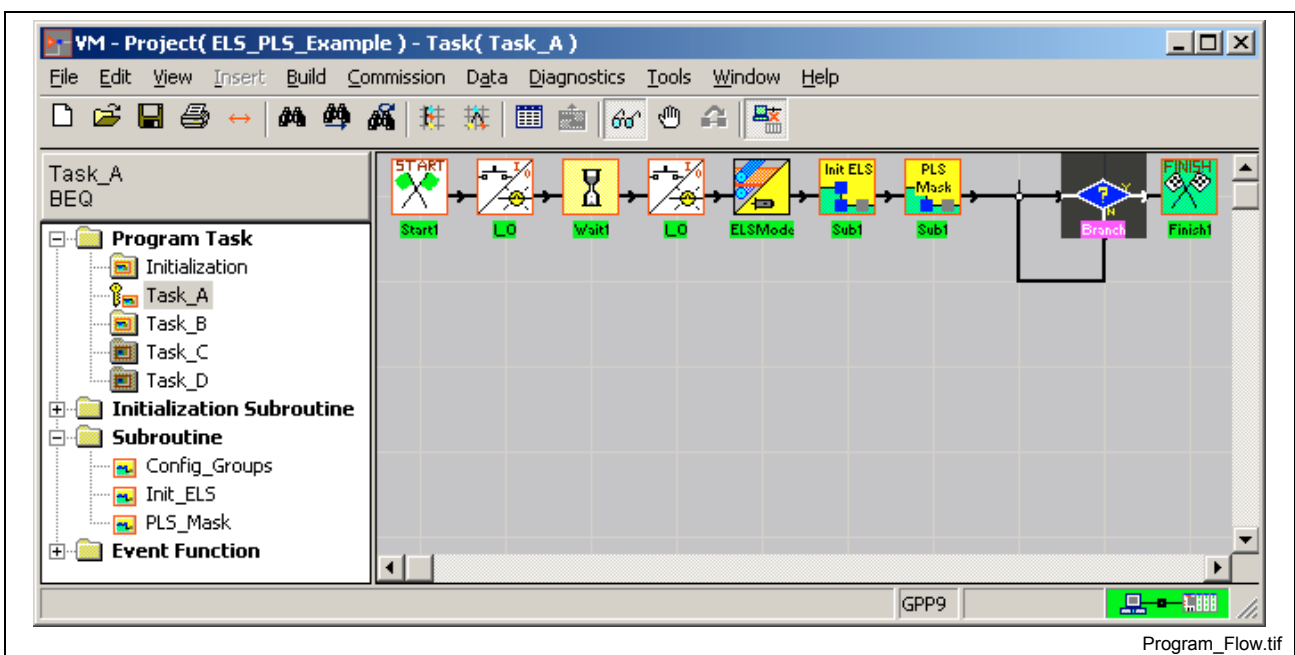


Fig. 2-178: Show Program Flow View

Many icons are scanned quickly and may appear to be skipped. To view the program flow for a subroutine used in multiple tasks, the user must identify the task controlling the subroutine. If a subroutine is opened by entering the function (through the icon) from the task level, VisualMotion knows the task – subroutine association. However, if the subroutine is opened by selecting **View** ⇒ **Subroutine** or selecting it from the *Project Navigator*, a window opens asking the user for task association.

Generate Map File

Show program flow uses a map file generated at compile time to tag the screen location of an instruction. If this map file is not found, the error message "Cannot open file \\...*.map!" appears (this usually means that the file was not compiled or downloaded to the control).

The compiler option for generating a map file is found under the menu selection, **Tools** ⇒ **Options**. By default, the *Generate Map File* option is selected. For larger programs that do not require a map file, removing the options decreases compile time.

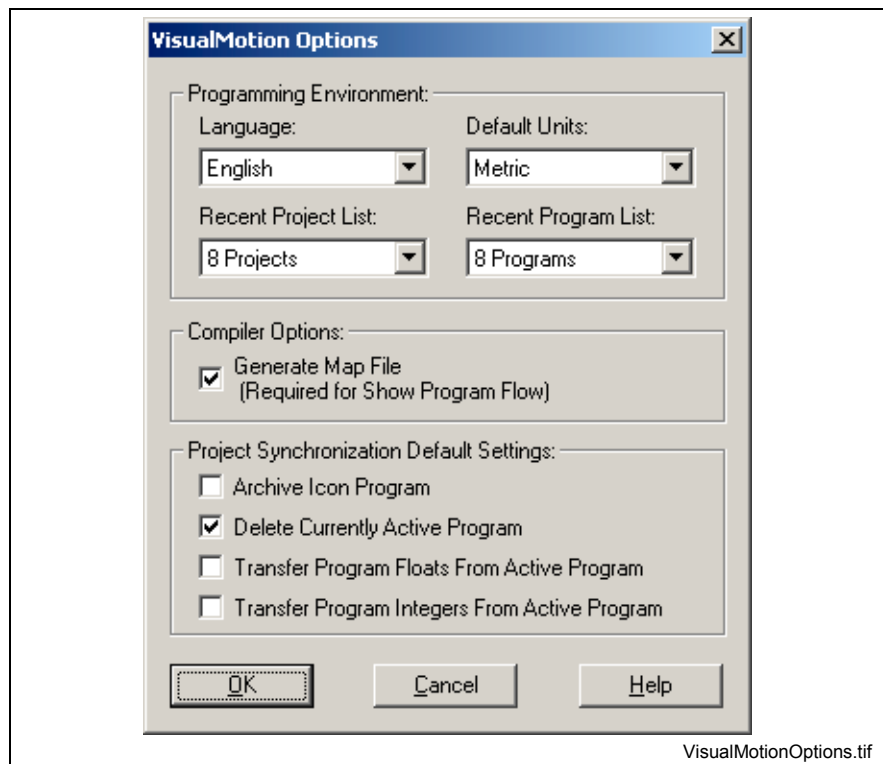


Fig. 2-179: Generate Map File

Toggle Breakpoint

Selecting **Diagnostics** ⇒ **Toggle Breakpoint** sets a breakpoint in the currently displayed task or subroutine. This menu selection is enabled only after the **Show Program Flow** function is selected. Breakpoints can only be set for task A-D and subroutines. The initialization task, initialization subroutines and event functions do not support breakpoints.

When the program is executed, program flow stops on the first icon after the Start icon in the selected function (task or subroutine).

Note: This function is only available in online or service mode.

Set Breakpoint using VisualMotion Toolbar Buttons

Use the following steps to set a breakpoint using the VisualMotion **Toggle Breakpoint State** and **Single Step** toolbar buttons.

1. Select a task (A-D) or standard subroutine.
2. Click the **Show Program Flow** toolbar button and then click on the **Toggle Breakpoint State** button.

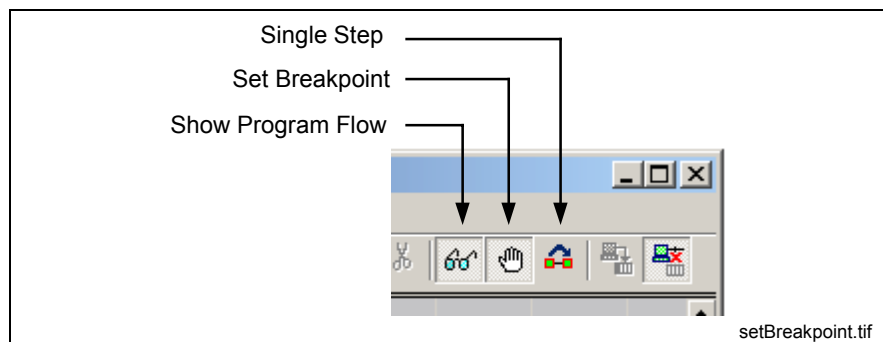



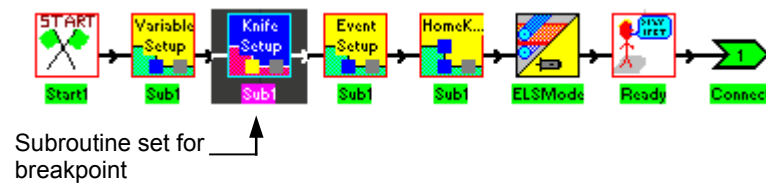
Fig. 2-180: Set Breakpoint

3. Start the VisualMotion program with the task or subroutine displayed. When the breakpoint is reached, the control switches to single step mode, program flow stops at the first icon following the *Start* icon and the *Current Instruction Display* section above the Project Navigator turns red.

Note: Refer to Current Instruction Display on page 2-4 for details.

4. Press the F8 key or click the Single Step  button to step through the program icons. If the task or subroutine contains a subroutine icon, all icons contained within the subroutine will be single stepped before proceeding to the next icon.
5. Once all program icons and/or subroutines are single stepped, program flow returns to the subroutine or task that called the subroutine or task.

For example, if a subroutine (contained in task A) is set for breakpoint, program flow will return to the next icon in task A following the single step completion of the subroutine.



Note: If a different item (in the Project Navigator) is selected, the single step function becomes inactive. The control remains in single step mode and requires that the user toggle the Cycle_Start_Resume (bit 6) of the task control register (2-5) for the selected task to resume program flow.

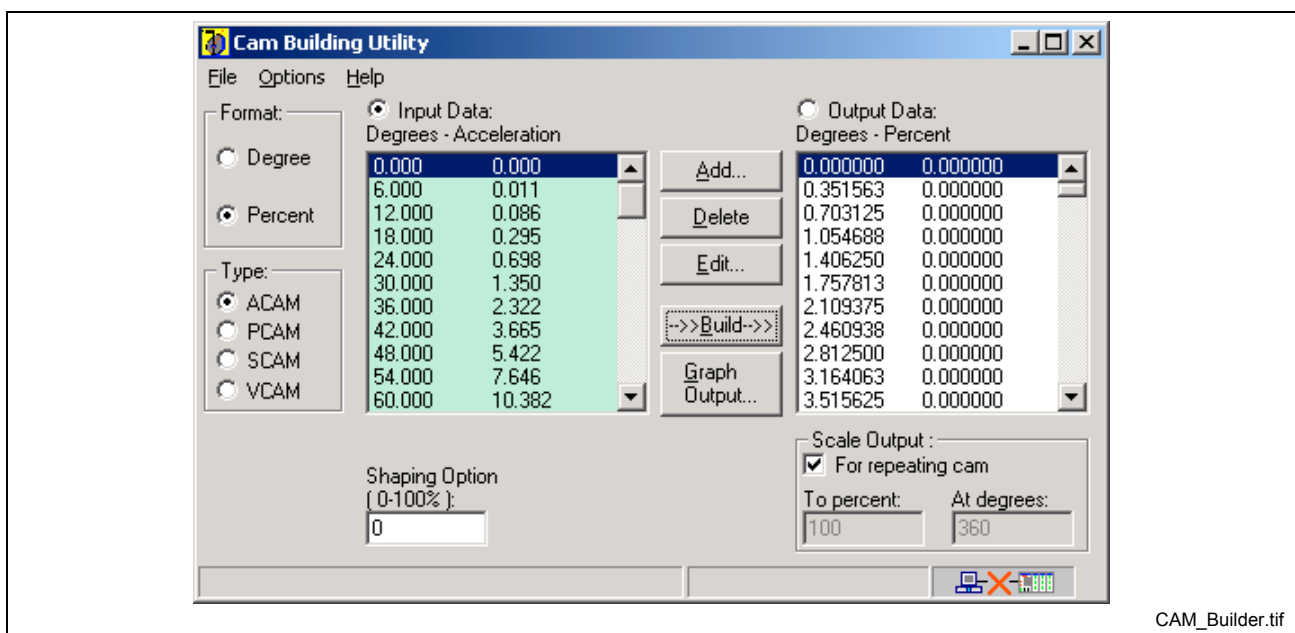
For best results, enable a breakpoint before starting the program or before the task or subroutine program flow is started. Once the Start icon in a task or subroutine is processed, a breakpoint can not be set.

2.10 The Tools Menu

The **Tools** menu contains the CAM Builder and Jogging utilities. The user can also launch additional Bosch Rexroth Registered Tools. Items such as different releases of the DDE Server, IoBox and even other instances of VisualMotion Toolkit can be launched. Communication settings and language selection are also available from the Tools menu.

CAM Builder

Selecting **Tools** ⇒ **CAM Builder** opens the *CAM Building Utility* window. This utility is used to build CAM tables for Bosch Rexroth's motion controls and drives.



CAM_Builder.tif

Fig. 2-181: CAM Building Utility Window

Format

This section defines the format, as *Degree* or *Percent*, that will be used to build a control or drive CAM. Pre-GPP 9 firmware can only build a control CAM using a degree format while GPP 9 or GMP 9 can use a degree or percent format. Drive CAMs can only be built using a percent format.

Degree CAM - output data is calculated where the master output position is in *Degrees* and the slave output position is in system *Units* (degree, in, mm, etc.).

Percent CAMs - output data is calculated where the master output position is in *Degrees* and slave output position is in *Percent*.

Type

Select the algorithm that will be used to build the output data from the user-defined input data. The available types are:

- **ACAM** – slave input data is entered as an acceleration at a given master position in degrees.
- **PCAM** – slave input data is entered as a position at a given master position in degrees.

- **SCAM** – slave input data is entered as a position at a given master position in degrees. Output data is build by applying the input data to a 3rd order **spline** polynomial.
- **VCAM** - slave input data is entered as a velocity at a given master position in degrees.

Input Data

The *Input Data Table* displays the master and slave input values entered by the user or uploaded from an input file. When the **Input Data** radio button is selected, the *Input Data Table* becomes active and the background color changes to green. The available buttons are **Add...**, **Delete**, **Edit...**, **-->Build-->**, and **Graph Output...**

Input data are defined by clicking the **Add...** button and entering input position values for both the master and slave. Existing input data can be modified by selecting the relevant line and clicking the **Edit...** button. Once all input data is entered, click the **-->Build-->** button to calculate the output data.

Output Data

This *Output Data Table* displays the output position data for the master to slave relationship calculated from the input data. Also, the output position data for a control or drive CAM can be uploaded for editing from an output file or by using the **CAM Transfer...** function under the **File** menu. When the **Output Data** radio button is selected, the *Output Data Table* becomes active and the background color changes to green. The available buttons are **Edit...**, and **Graph Output...**

Output Steps

The *Output Steps* are only available for degree CAMs and defines the number of output steps (points) that will be used to build the CAM. The allowable range for output steps is 10 - 1025.

Shaping Option

Select the percentage of S-shaping to be used on the velocity curve to prevent a jump in acceleration (jerk limiting). This option is not available for SCAM profiles.

Output Modulo

The *Output Modulo* is only available for degree CAMS and defines the Output Modulo that will be used for scaling the range of ACAM and VCAM output calculations. By default, this scaling is set to 360.

Scale Output

The *Scale Output* is only available for percent CAMs and is used to scale the CAM's output profile.

For repeating CAM – by default, this checkbox is checked and is used for endless running CAMs, where the percent value is fixed at 100 and the degree value is fixed at 360.

To percent – scalar percentage multiplied to the CAM's H factor.
 $\% \times H \text{ factor} = \text{units of distance}$

At degree – master output position where the "To percent:" value is achieved.

Buttons

Add... - add an entry to the input data.

Delete - deletes selected entry in the input data.

Edit... - edits selected entry in the input or output data.

-->>Build-->> - builds position, velocity and acceleration CAM profiles and displays position data in the *Output Data Table*. Uses the algorithm for the selected *Type* to build the number of data sets chosen in *Output Steps* (10-1025 for degree CAMs and fixed to 1024 for percent or drive CAM) using the input data and the *Shaping Option* to modify.

Note: CAM for percent CAMs are allowed within the range of +/- 200%. If this range is exceeded, an error will be issued and the slave output positions will be scaled back to 100%.

Graph Output... - creates or refreshes a graph of the position data in the *Output Data Table* using the speed in the *Simulation Speed for Graph* setting under the **Options** menu. The velocity is differentiated from the position and the acceleration is differentiated from the velocity.

Note: Measurement of any point of the graph can be taken by positioning the mouse over the desired graph portion and holding the left mouse button.

- **Zoom In/Out** – increases or decreases the size of the graph for more accurate measurements.

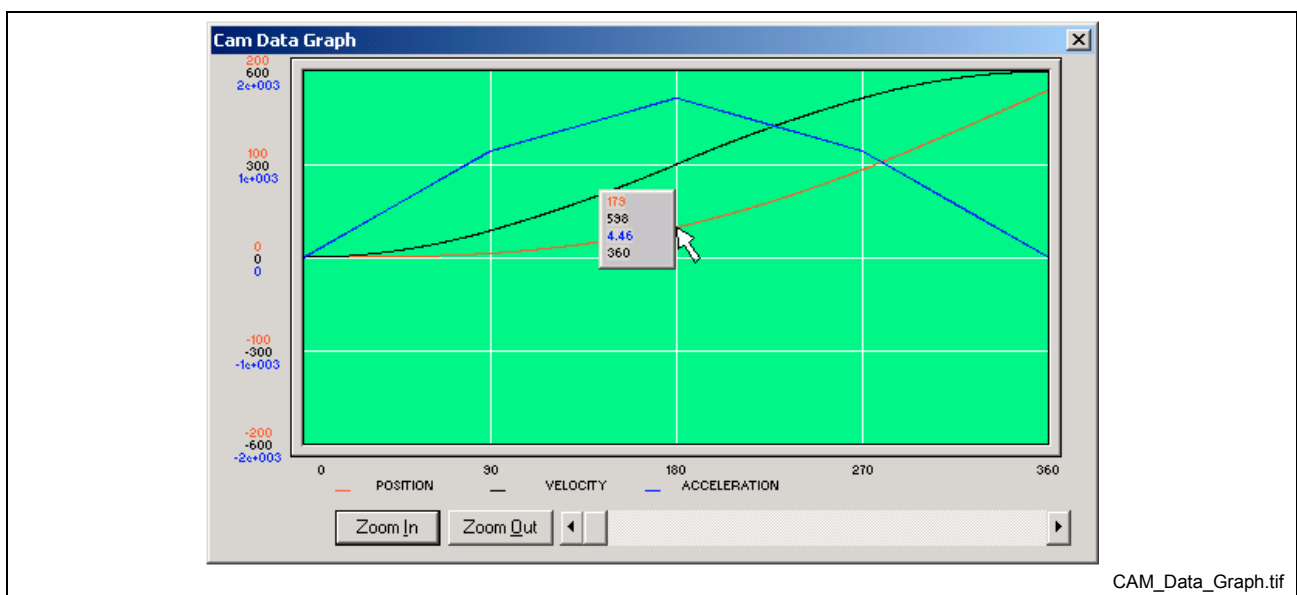


Fig. 2-182: CAM Data Graph

File Menu

The following figure displays the available menu selections under the **File** menu:

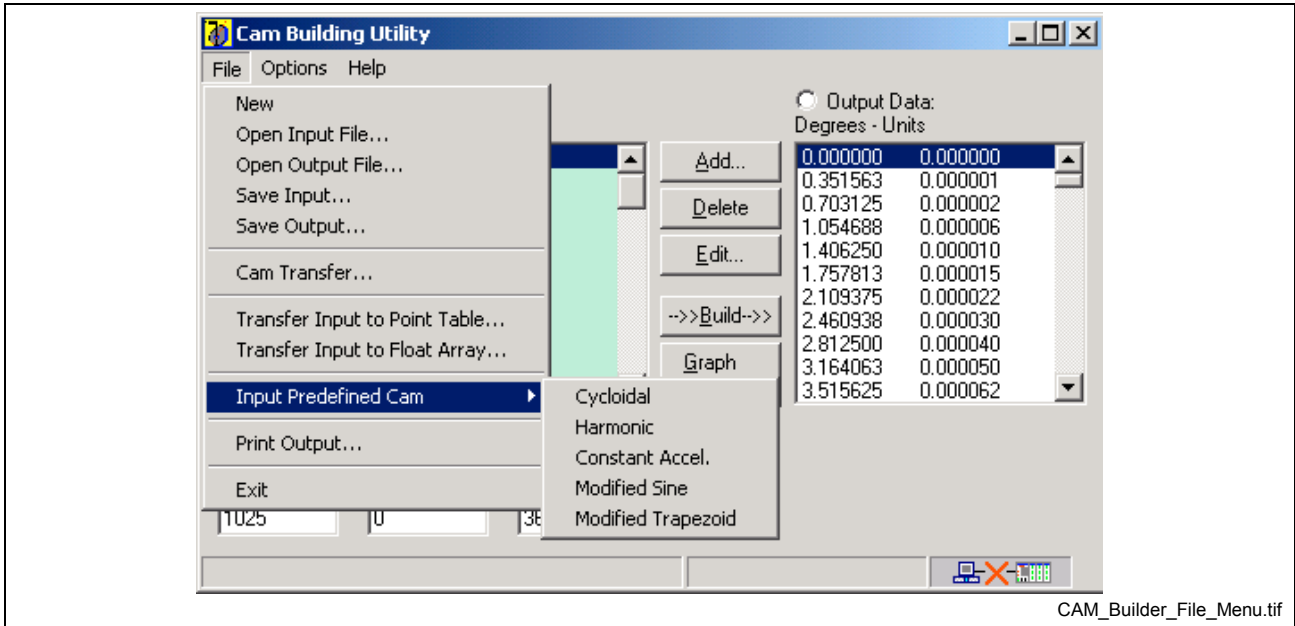


Fig. 2-183: CAM Building Utility - File Menu

New

This menu selection clears the *Input* and *Output Data Tables*.

Open Input File...

This menu selection loads data from an input file (*.txt) into *Input Data Table*.

- User-selected input file. Input files are assumed to have the following format:

```
File identifier string.      ;first line only
master value, slave value  ;second line
master value, slave value  ;third line
----                      ---
----                      ---
master value, slave value  ;last line
```

Minimum and maximum number of data sets:

Build Types	ACAM	PCAM	SCAM	VCAM
Minimum Sets	2	2	5	2
Maximum Sets	1024	512	200	1024

Note: A 5th order polynomial will be used when the maximum values are exceeded for PCAM and SCAM types.

Open Output File...

This menu selection loads data from an output file (*.csv) into *Output Data Table*.

- User-selected output file. Output files are assumed to have the following format:
CAM output files are sent to the control using VisualMotion.

```

master value, slave value ;first line
master value, slave value ;second line
----
master value, slave value ;last line

```

Save Input...

This menu selection saves values from the input data to a user-selected input file. The file is saved using text format (*.txt). Refer to *Open Input File...* for file format.

Save Output...

This menu selection saves values from the output data to a user-selected output file. The file is saved using standard spreadsheet format (*.csv). Refer to *Open Output File...* for file format.

CAM Transfer...

The *Transfer CAM* window is used to upload (Get), download (Send), or delete existing control or drive CAMs from a project in offline mode or from the control or selected drive in online or service mode.

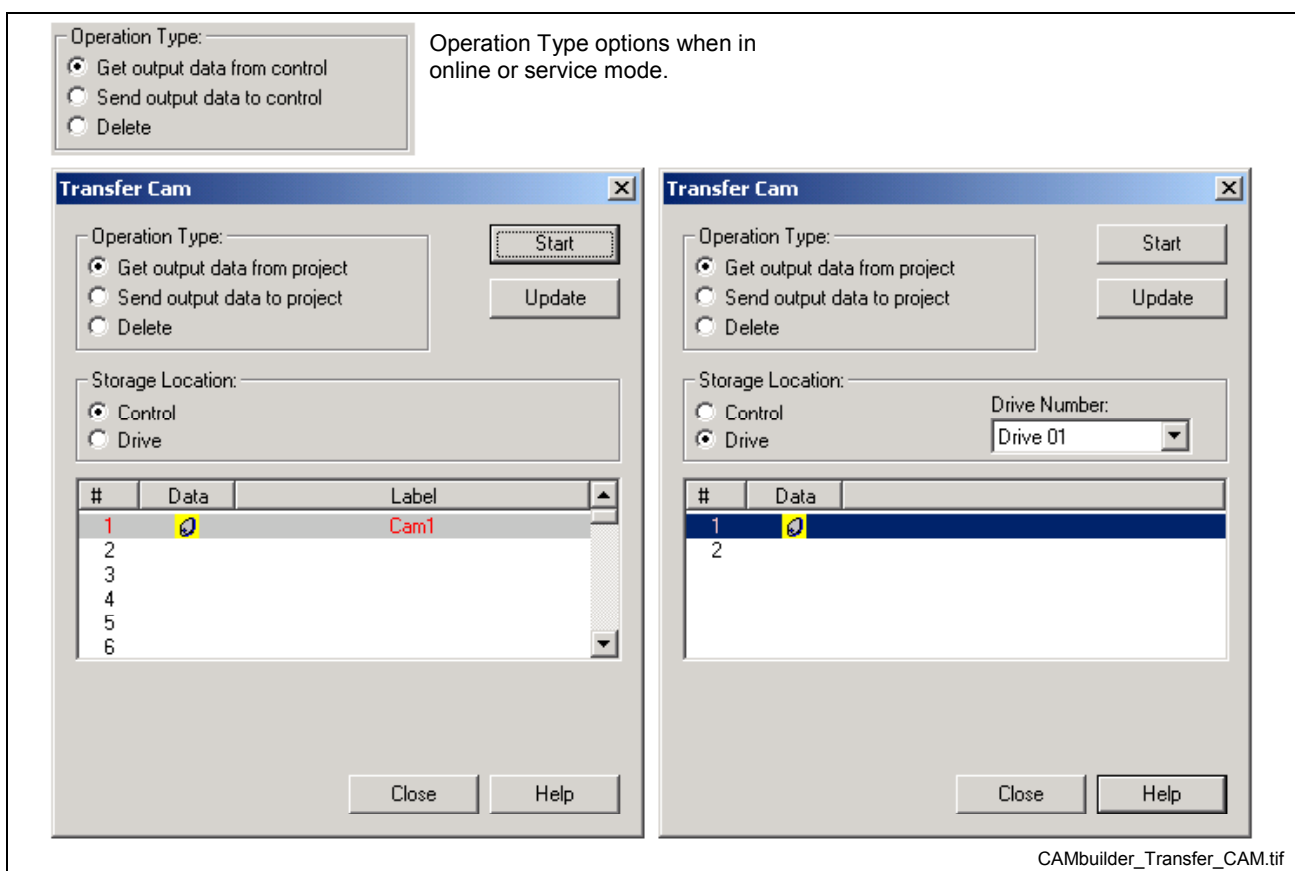


Fig. 2-184: Transfer CAM

Note: A CAM graphic under the **Data** column is an indication that the control or drive contains a CAM. Only existing control and drive CAMs can be retrieved or deleted.

- **Get output data from project (control)** – retrieves the selected CAM output data for the specified **Storage Location** (Control or Drive) from the project in offline mode and from the control's memory in online or service mode. An uploaded CAM is displayed in the *Output Data Table* of the *CAM Builder* utility.

- **Send output data to project (control)** – downloads the current output data in the CAM Builder to the project in offline mode and to the control or drive in online or service mode.
- **Delete** - deletes the selected CAM number from the specified storage location (Control or Drive) from the project in offline mode and from the control's memory in online or service mode.
- **Start** - this button becomes active under the following conditions:

Operation Type	Condition
Get output data from project (control)	An existing control or drive CAM is selected.
Send output data to project (control)	<i>Output Data Table</i> in CAM Builder contains data and a control or drive CAM number is selected
Delete	An existing control or drive CAM is selected.

Table 2-19: Start Button Activation

Transfer Input to Point Table...

This window is used to transfer the contents of the Input Data Table to a set of ABS points located in a project in offline mode or on the control in online or service mode.

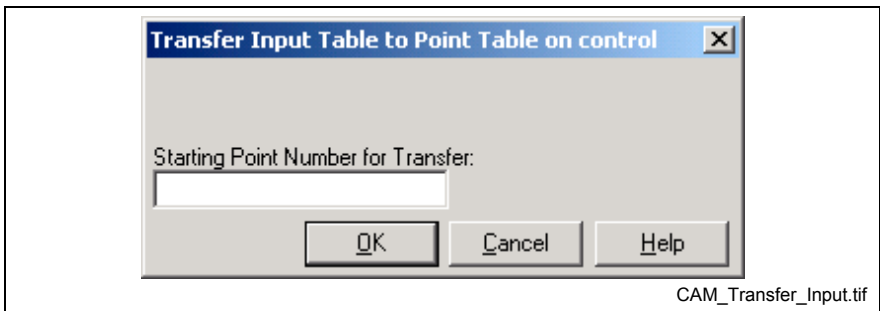


Fig. 2-185: Transfer Input Table to Point

- **Starting Point Number for Transfer** - Enter the beginning ABS point number to where the data should be copied. The values of the first line of the input data will be copied to this point. The values from each additional line of the input data will be copied to the next point. The number of consumed ABS points depends on the number of lines in the *Input Data Table*. The **OK** button starts the checking process and transfer. Before the transfer is started, checks are made for valid point and valid range. A window opens for confirmation. The **Cancel** button exits this window.

Transfer Input to Float Array...

This window is used to transfer the contents of the *Input Data Table* to a set of consecutive program floats in a project in offline mode and to the control in online or service mode.

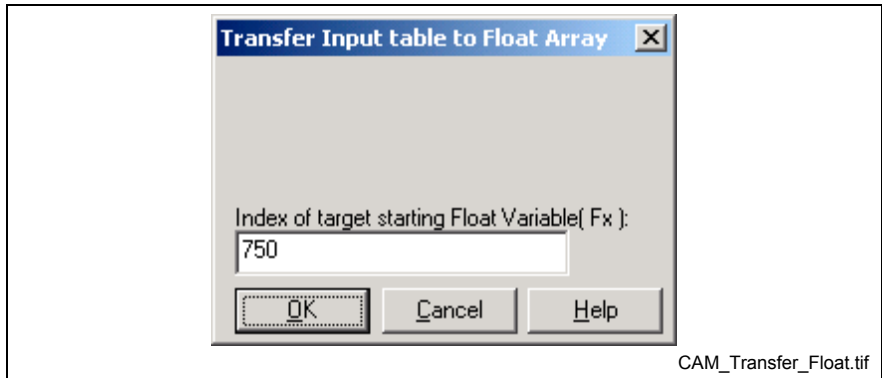


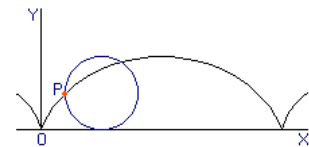
Fig. 2-186: Transfer Input to Float Array

- **Index of target starting Float Variable(Fx)** - Enter the program float number that will be the start for the transfer. The **OK** button starts the checking process and transfer. Before the transfer is started, checks are made for valid program float and valid range. A window opens for confirmation. The **Cancel** button exits this window.

Input Predefined CAM ▶

This menu selection uploads the input data of a predefined CAM file for the following types:

- Cycloid
- Harmonic
- Constant Acceleration
- Modified Sine
- Modified Trapezoid



Cycloid - The curve traced by a point on the circumference of a circle that rolls on a straight line

Print Output

This menu selection prints graph and output data.

Exit

This menu selection closes the CAM Building Utility.

Options Menu

This *Options* menu is used for configuring control CAM format, graphing, and simulating speed for the graph.

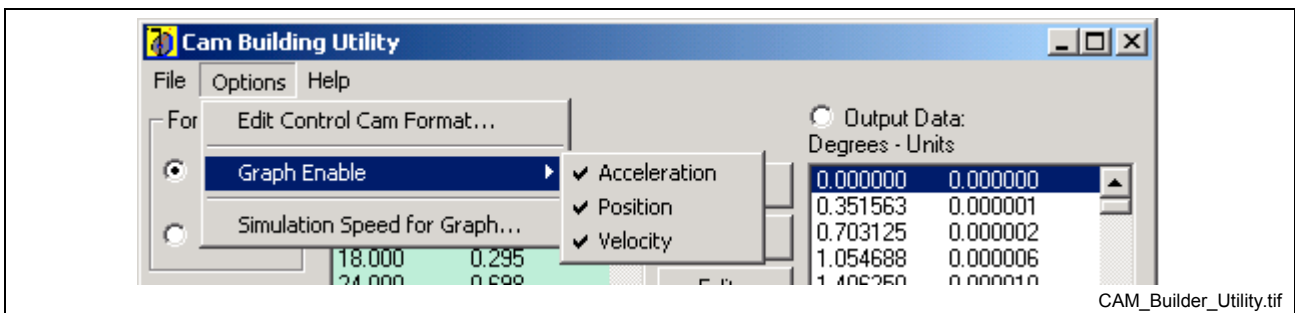


Fig. 2-187: CAM Building Utility, Options Menu

Edit Control CAM Format...

This option specifies the format (degree or percent) used by the control when running control CAMs. In offline mode, this setting is saved to the project and is written to control parameter C-0-3141 when the project is synchronized. In online mode, this setting is written to C-0-3141 in the control and project.

Note: When running control CAMs, the format of the active control CAM must match the format in C-0-3141 to ensure proper CAM profiles.

Graph Enable

This selection allows the user to choose between acceleration, position, and velocity graphs.

- *Acceleration Graph* - when checked, the acceleration graph is visible on the graph and printout.
- *Position Graph* - when checked, the position graph is visible on the graph and printout.
- *Velocity Graph* - when checked, the velocity graph is visible on the graph and printout.

Simulation Speed for Graph

This selection allows the user to specify a speed used in graphing the velocity and acceleration profiles.

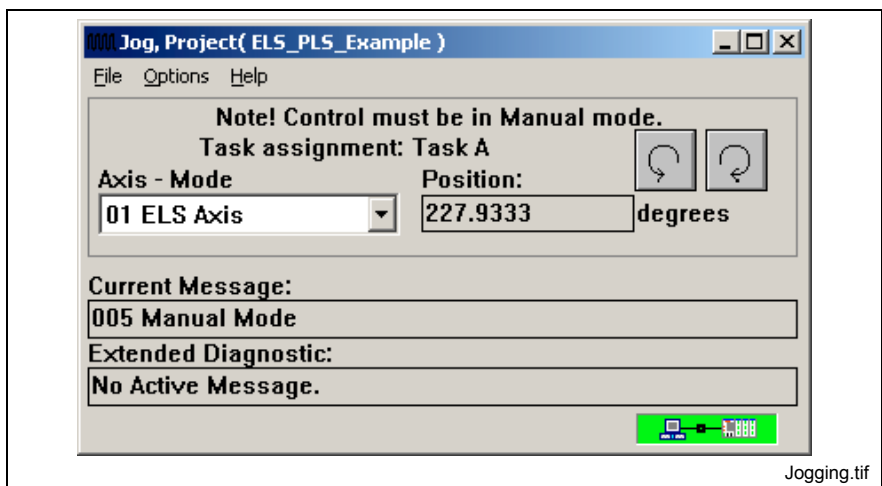
Help Menu

The Help menu is used for accessing help system and identifying the product.

Jogging

Selecting **Tools** ⇒ **Jogging** opens the *Jog Runtime Tool* window. This tool is used to jog an axis in manual mode. Only axes configured for either *Velocity* or *Single Axis* operating modes can be jogged using this tool.

Note: This tool is only available for online and service modes.



Jogging.tif

Fig. 2-188: Jog Window

Note: *Coordinated Motion* axes cannot be jog using this tool and must be jogged in manual mode using the Task Jog registers 007-010 for task A-D, respectively.

The *Jog* window displays the current task assignment, mode and position for the active axis along with status messages and extended diagnostics.

Holding down the jog reverse  or jog forward  buttons, respectively, changes the state of bit 3 (Jog_Reverse) or bit 2 (Jog_Forward) in the Axis_Control Register from 0 to 1. Releasing these buttons change the state of these bits from 1 to 0.

A low-to-high (0-1) transition on these bits causes motion to start in the negative (bit 3) or positive (bit 2) direction. A high-to-low (1-0) transition immediately stops the motion. Motion is also stopped when the task mode selection changes, or when a travel limit or incremental distance is reached.

Jogging in Automatic Mode

An axis configured for either *Velocity* or *Single Axis* modes can be jogged while the system is in automatic mode or while a task is running. Control parameter C-0-0010, bit 11 (Jog in Auto) must be set to 1 before the system is switched to automatic mode.

If an axis is jogged while the task is running, the axis can be manually jogged in either direction but is no longer part of the running program. The program must be stopped and restarted to reinitialize the axis to the program.

Axis Jogging Options

Selecting **Options** ⇒ **Axis** from the main *Jog* window opens the *Axis Jogging Options* window.

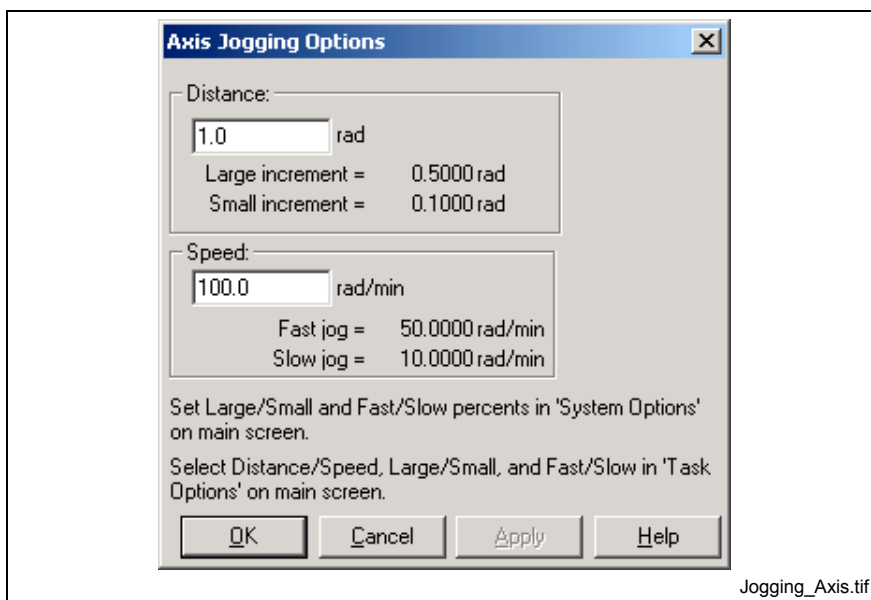


Fig. 2-189: "Axis Jogging Options" Window

The *Axis Jogging Options* window allows input of the maximum jog distance and speed. The maximum distance value is stored in parameter A-0-0025. The maximum speed is stored in parameter A-0-0026. The large/small increment values and the fast/slow velocities are calculated according to the values entered in the *System Jogging Options* window.

System Jogging Options

Selecting **Options** ⇒ **System** from the main *Jog* window opens the System Jogging Options window.

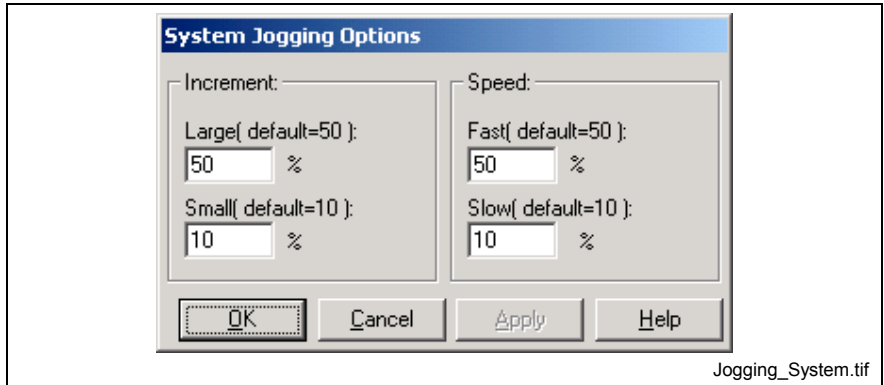


Fig. 2-190: “System Jogging Options” Window

The *System Jogging Options* window is used for setting the increments and velocities used for fast and slow jogging. The *Increment* data area is used to set the *Large* and *Small* percentage of the maximum distance for a single-step jog operation. The maximum is defined by axis parameter [A-0-0025](#), Maximum Jog Increment. Similarly, the *Speed* data area is used to set the Fast and Slow jog speeds as a percentage of the maximum velocity, which is defined by axis parameter [A-0-0026](#), Maximum Jog Velocity.

The values are stored in the following parameter locations:

Large Increment - [C-0-0052](#)

Small Increment - [C-0-0053](#)

Fast Speed - [C-0-0055](#)

Slow Speed - [C-0-0056](#)

Task Options

Selecting **Options** ⇒ **Task** from the main *Jog* window opens the *Task Options* window.

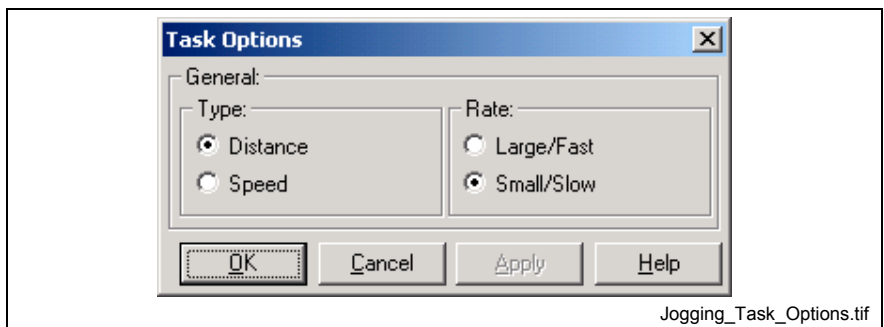


Fig. 2-191: “Task Options” Window

The *Task Options* window allows the user to select the jog *Type* and *Rate*. The *Type* can be Distance (incremental) or Speed (continuous). The type of jog takes effect when the next jog is started, with a transition on the jog forward or reverse buttons. When distance is selected, the jogging motion stops after the large or small travel limit is reached. When speed is selected, the jogging motion continues until the jog button is released or the travel limit is reached. Refer to *VisualMotion 9 (GPP) Application Manual*, DriveTop, **Drive Functions** ⇒ **Drive Limitations**.

The *Rate* can be Large/Fast or Small/Slow. When *Distance* is selected, the options are Large or Small. When *Speed* is selected, the options are Fast or Slow. Refer to Bit # 6 in the [Task Jog Control Register](#) for details.

Single-Axis Mode Jogging

Position Limits Enabled

When a jog forward is started, the control sets the target position of the axis to the positive travel limit. When a reverse jog is started, the target position is set to the negative travel limit. When the jog is stopped, the target velocity is set to zero, but the target position remains at the travel limit.

Position Limits Not Enabled - DIAX03 Drives, Version 04 and Greater

On DIAX03 drives, version 4 and later, the drive is switched to velocity mode. The ramps are generated internally by the drive using the axis jog acceleration parameter [A-0-0023](#).

Before performing single-axis positioning using the `axis_move` command, it is necessary to execute the `els_mode` command to switch the drive back into single-axis mode. A cycle stop followed by a cycle start will also reset the drives to single-axis mode.

Position Limits Not Enabled - Other Drives

The drive remains in single-axis mode. The control will continually increase or decrease the target position by a small amount to keep the drive moving, until the jog is stopped. It sets the acceleration to a value high enough so that the drive does not decelerate. The jog acceleration parameter will be used only if it is lower than this value.

Velocity Mode Jogging

Drives

The ramp selection in parameter [A-0-0004](#) bit 9 determines if the programmed acceleration is used. The control generates a ramp if ramping is enabled; otherwise, the velocity is immediately stepped to the programmed value.

Refer to the following parameter and register descriptions for more jogging information:

<u>C-0-0042</u>	World Large Increment
<u>C-0-0043</u>	World Small Increment
<u>C-0-0045</u>	World Fast Jog Speed
<u>C-0-0046</u>	World Slow Jog Speed
<u>C-0-0160</u>	Virtual Master Maximum Jog Velocity
<u>T-0-0025</u>	Maximum Jog Increment
<u>T-0-0026</u>	Maximum Jog Velocity

[Registers 31-38](#) Axis(n) Status - bit 2, jogging fwd, bit 3, jogging rev.

Registered Tools

The menu selection **Tools** ⇒ **Registered Tools** displays all Bosch Rexroth tools that are registered in the user PC. All tools are group by version as illustrated in the following figure.

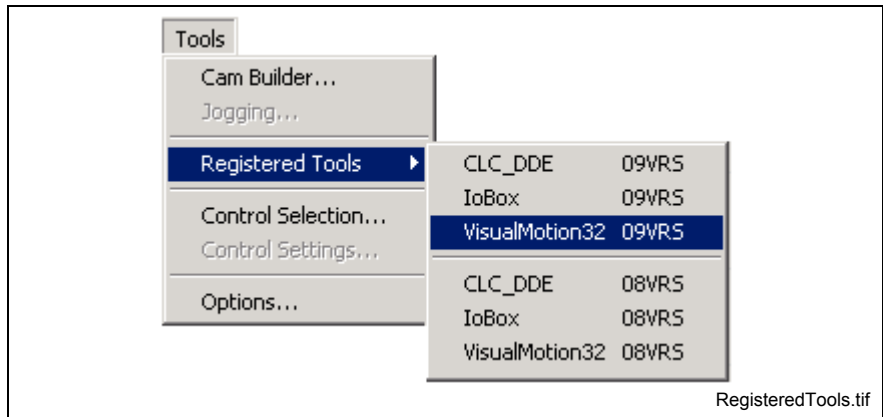


Fig. 2-192: Registered Tools

CLC_DDE

This selection opens a new instance of the DDE Server if the selected version is not already opened.

IoBox

IoBox is a freestanding utility that allows the user to change bits and registers using a visual interface. The interface looks like a hardware I/O box with on/off buttons (pressed = on, depressed = off).

Button labeling and data output is programmed in the **I/O DeskTop Set-Up** window, accessed by clicking the **Set-Up** button. These preferences are saved in an *.iob file after clicking **OK**. A new configuration can be saved for each program, if desired, with descriptive buttons for each subroutine.

IoBox provides access to all system registers in addition to the programmed buttons.

VisualMotion32

This selection opens a new instance of the selected VisualMotion Toolkit release version.

Control Selection

Selecting **Tools** ⇒ **Control Selection** opens the *Control Selection* window. This window is used to select the method of communication (serial or Ethernet) between the Host and control in offline or service mode.

Note: The *Control Selection* window is available in online mode and displays the current method of communication. However, no changes can be made while online.

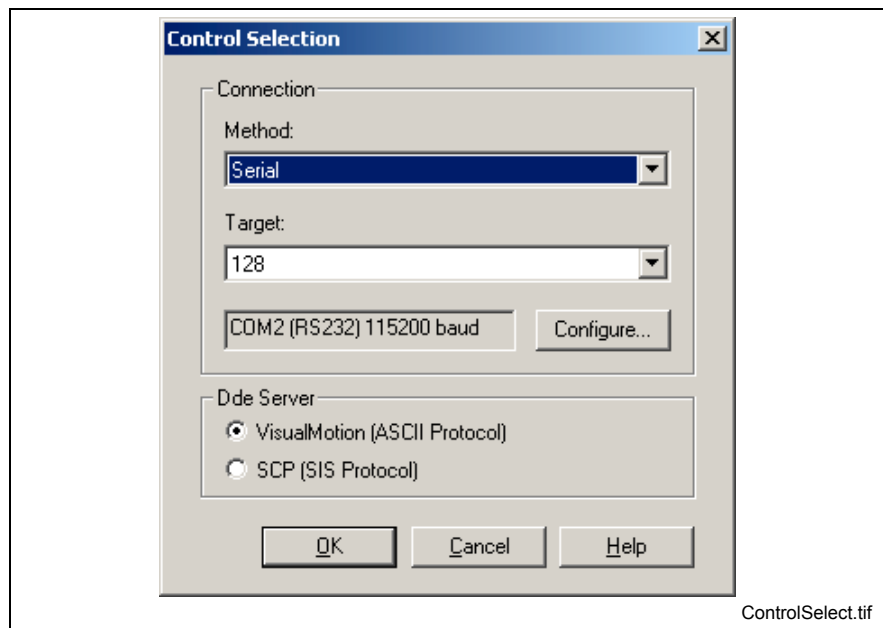


Fig. 2-193: Card Selection Setup Window

Method (Connection)

Serial – the connection between the PC and control using an IKB0005 RS-232 serial interface cable.

Network – the connection to a control across a TCP/IP EtherNet connection.

Note: Method is not available when using SCP as the DDE Server. The method of communication is configured in the SCP.

Target

This field identifies the control's address in the SERCOS ring for a serial connection. When connected to the EtherNet card, the target represents the network communication label assigned in the DDE Server.

Configure

Clicking the **Configure...** button for a *Serial* connection opens the Serial Communications window in the DDE Server and the Network Communication window for a *Network*.

When SCP is selected as the DDE Server, the **Configure...** button opens the SCP Systemconfigurator. Refer to chapter 3 of the VisualMotion Application manual for details.

For these windows, the user can modify the current settings for each connection method.

The read-only field just to left of the configure button shows the COM port settings for a serial connection and the IP address for a Network connection.

DDE Server

The radio buttons in this section allows the user to switch between the VisualMotion DDE Server (ASCII Protocol) and the SCP (SIS Protocol). Refer to Chapter 11 for ASCII and SIS Protocol details.

Control Settings

The **Control Settings** menu selection is used to configure the communication settings between the control and any external interface. In addition to communication settings, the user can also set a password for the X10 Program Port, X16 Communication Port and the Ethernet Card. All fields within the *Communication Settings* window are stored to control parameters. Holding the cursor above any field will display the corresponding parameter number.

General

The General tab is used to set the Control's Address and Communication Timeout and Protocol.

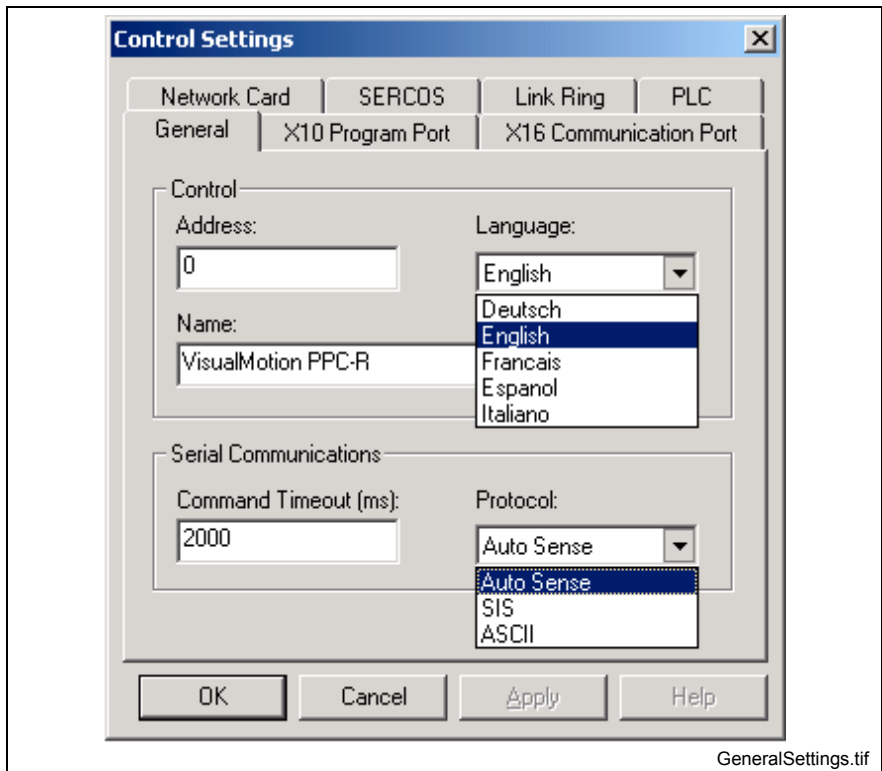


Fig. 2-194: General Communication Settings

Setting	Description	Parameter
Control Address:	Sets the control's address	C-0-0002
Language:	Sets the language of the control and all active drives. Note: To change the language of a specific drive, use the Parameter Overview tool and modify SERCOS parameter S-0-0267.	C-0-0001
Name:	Sets the control's name	C-0-0142
Command Timeout:	Sets the communication time-out period for serial communication. The state of the communication error timer is set to enabled/disabled by start/stop commands from the serial device.	C-0-0016
Protocol:	Sets the current communication protocol recognized by VisualMotion for system communication. Change only in phase 2.	C-0-0005

Table 2-20: General Communication Settings

X10 Program Port

This window is used to modify the X10 serial connection on the control.

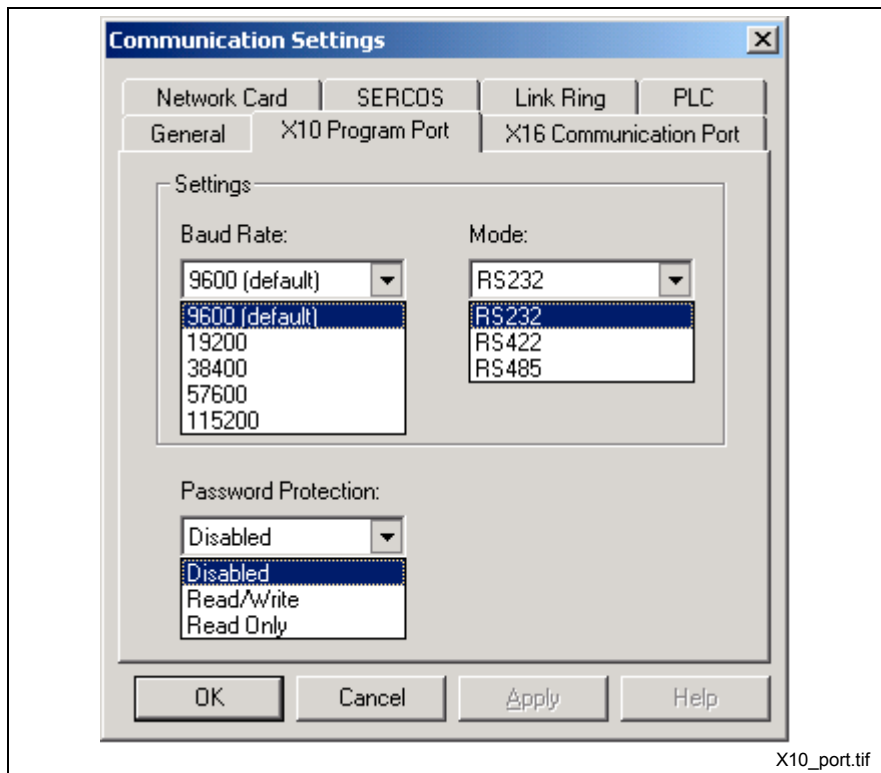


Fig. 2-195: X10 Program Port Settings

Setting	Description	Parameter
Baud rate:	The allowable baud rates for the X10 program port are... 9600 (default), 19200, 38400, 57600 and 115200.	C-0-0003
Mode:	The allowable serial interface modes are RS232, RS422 and RS485.	C-0-0013
Password Protection:	The password protection drop down list allows the user to set a password to limit the access to the X10 port. The allowable settings are Disabled (default), Read/Write and Read Only. Refer to Password Protection on page 2-166 for details.	C-0-0017

Table 2-21: X10 Program Port Settings

X16 Communication Port

This window is used to modify the X16 serial connection on the control. The port is typically used to communicate with a teach pendant (default 9600 baud), if one is installed. If an ASCII "dumb" terminal (e.g. a BTC HMI) is used to communicate with a control, the checksum should be disabled.

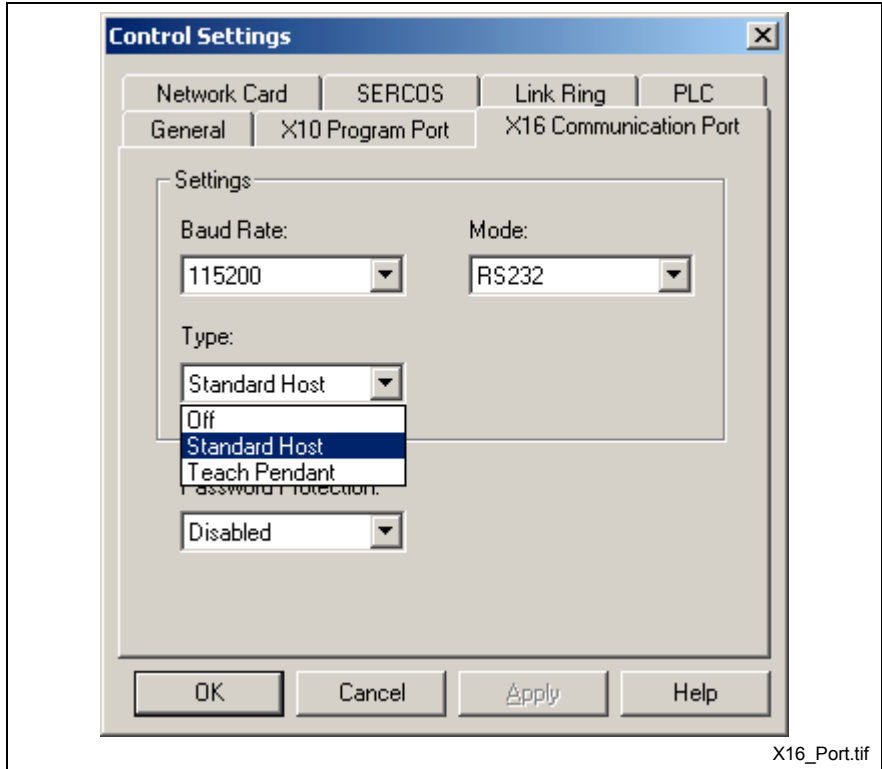


Fig. 2-196: X16 Program Port Settings

Setting	Description	Parameter
Baud rate:	The allowable baud rates for the X10 program port are... 9600 (default), 19200, 38400, 57600 and 115200.	C-0-0004
Mode:	The allowable serial interface modes are RS232, RS422 and RS485.	C-0-0014
Type:	Select Standard host for a PC and Teach Pendant for an HMI.	C-0-0012
Password Protection:	The password protection drop down list allows the user to set a password to limit the access to the X10 port. The allowable settings are Disabled (default), Read/Write and Read Only. Refer to Password Protection on page 2-166 for details.	C-0-0018

Table 2-22: X16 Program Port Settings

Network Card

This window is used to configure the EtherNet card settings. The user can set a password to limit access to the control via the network and set the mode of transmission.

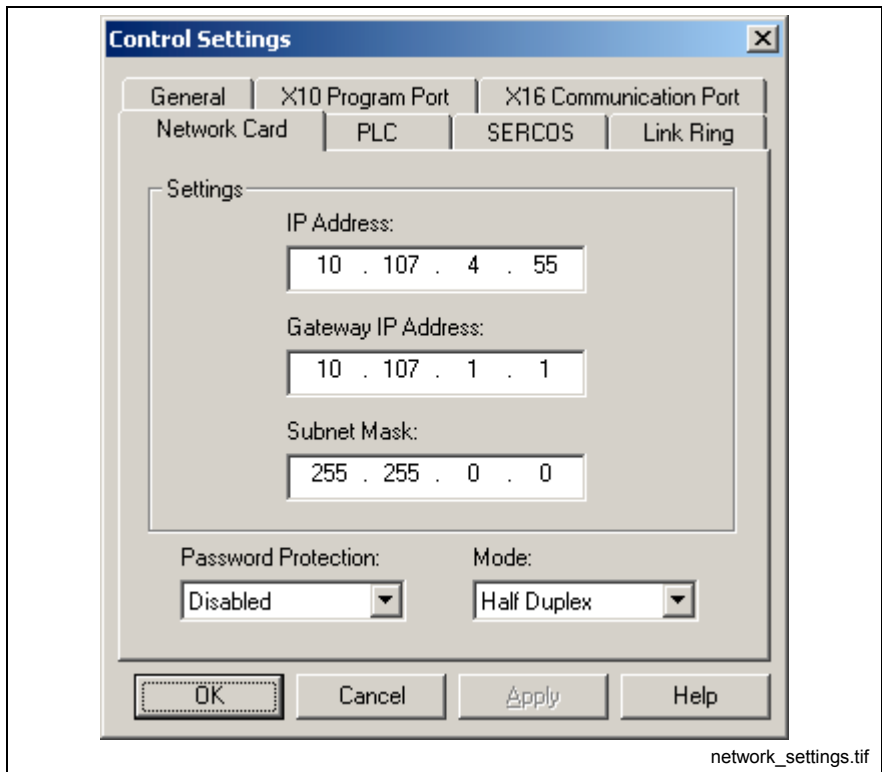


Fig. 2-197: Ethernet Card Settings

Setting	Description	Parameter
Settings:	The EtherNet card's IP Address, Gateway IP Address and Subnet Mask are provided to the user by their respective IT department. Every EtherNet card must have a unique IP Address assigned.	C-0-0400 C-0-0401 C-0-0402
Mode:	Half Duplex (default) allows transmission in only one direction at a time (receive or transmit). Full Duplex allows bi-directional transmission to and from the control. Full Duplex requires a connection via a LAN switch that supports manual setting of the Duplex mode. The EtherNet card does not support auto-negotiation, so this specific switch setting is necessary for full duplex support.	C-0-0403
Password Protection:	The password protection drop down list allows the user to set a password to limit the access to the EtherNet card. The allowable settings are Disabled (default), Read/Write, Read Only and No Access. Refer to Password Protection on page 2-166 for details.	C-0-0404 C-0-0405

Table 2-23: Ethernet Card Settings

Note: Cycle power to the control for Ethernet settings to take effect.

SERCOS

This window is used to set the SERCOS transmission rate and cycle time. In addition, the user can specify the length of the fiber optic transmission cable.

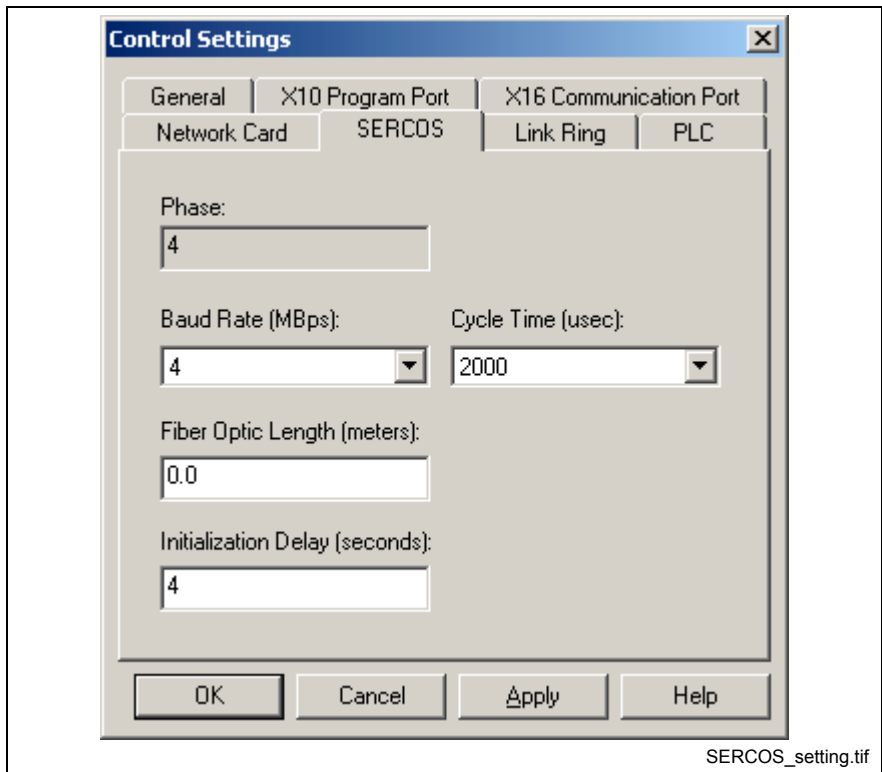


Fig. 2-198: SERCOS Settings

Setting	Description	Parameter
Phase:	This read only field displays the current SERCOS phase.	C-0-0121
Baud Rate MBps:	Sets the SERCOS transmission rate. (Default: 2MBps)	C-0-0010
Cycle Time (usec):	Sets the SERCOS cycle time. Refer to Card parameter C-0-0099 for details. (Default: 2000usec)	C-0-0099
Fiber Optic Length:	Sets the intensity of the output from the control's SERCOS transmitter, based on the length of the cable in meters. (Default: 0.0)	C-0-0020
Initialization Delay	Causes the control to delay for the specified number of seconds before it initializes the SERCOS ring.	C-0-0098

Table 2-24: SERCOS Settings

Link Ring

This window is used to set the Link Ring communication settings.

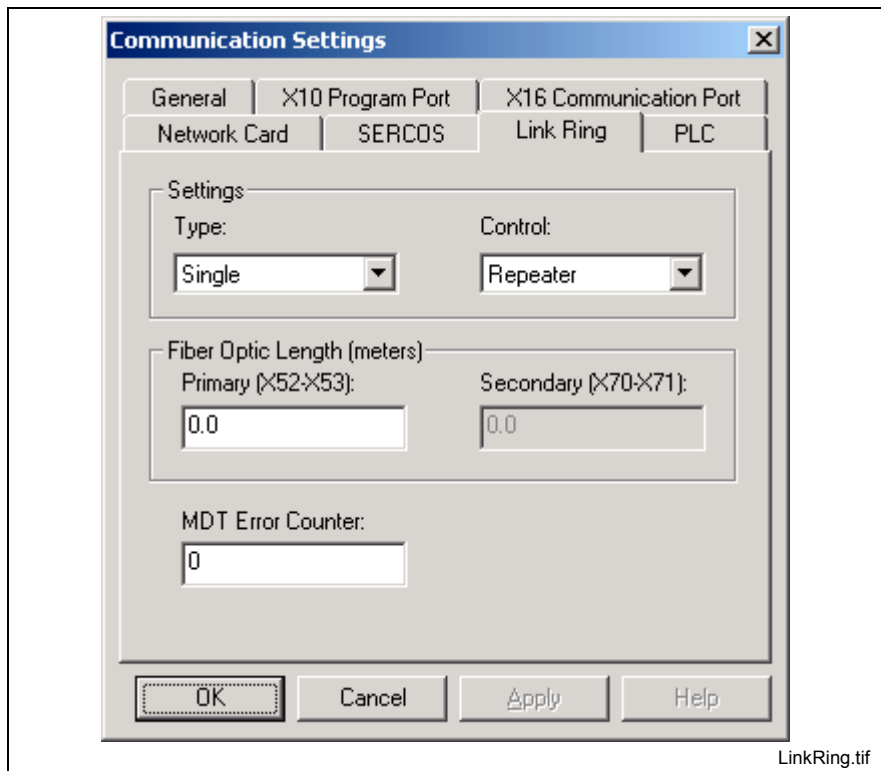


Fig. 2-199: Link Ring Settings

Setting	Description	Parameter
Type:	Sets the Fiber Optic structure as either Single or Double.	C-0-0300
Control:	Sets the control as an active (Master or Slave) participant or a Passive (Repeater) Participant in a Link Ring.	C-0-0300
Primary:	Sets the output power of the DAQ card to the length of the connected primary fiber optic cable.	C-0-0301
Secondary:	Sets the output power of the DAQ card to the length of the connected secondary fiber optic cable. Used in a Double ring structure.	C-0-0302
MDT Error Counter:	This field displays the illegal master data telegrams (MDT) count by the slave. Entering a 0 resets the counter.	C-0-0303

Table 2-25: SERCOS Settings

Refer to *Chapter 10, Link Ring Functionality*, of this manual for details.

PLC

This window is used to set the PLC communication settings.

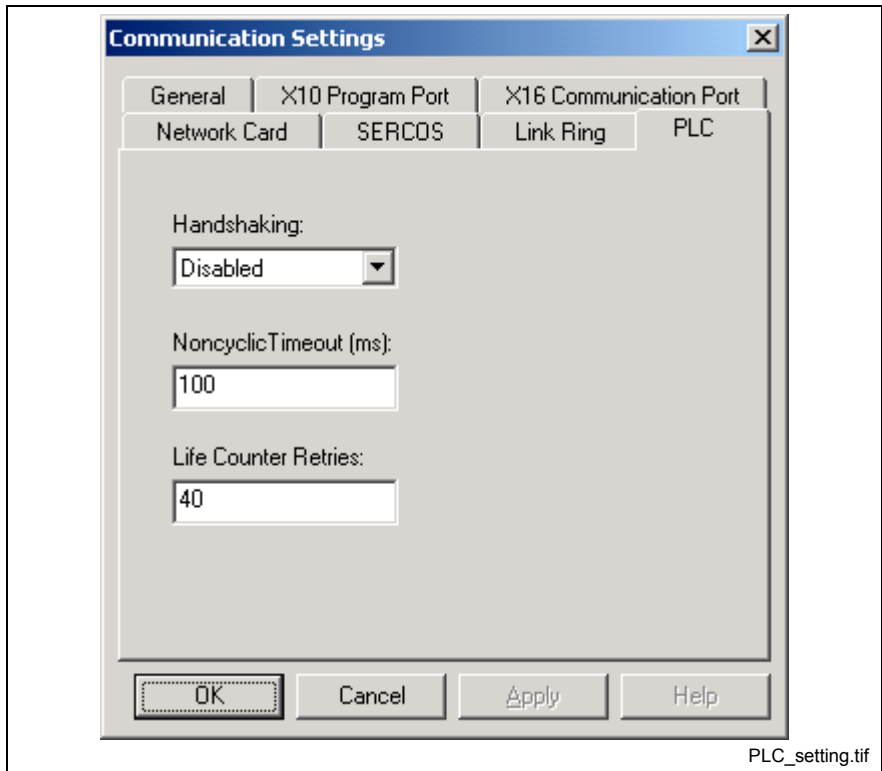


Fig. 2-200: PLC Communication Settings

Setting	Description	Parameter
Handshaking:	Initializes the handshaking between the control and the PLC. The control monitors the state of this parameter only at power up. Note: Must be in Parameter mode.	C-0-0035
Non-cyclic Timeout (ms):	Sets the non-cyclic channel communication timeout value in the DPRAM of the MTS-R.	C-0-2647
Life Counter Retries:	The control checks the life counter every SERCOS cycle in the I/O task. This field sets the number of allowable retries before an error is issued.	C-0-2643

Table 2-26: SERCOS Settings

Password Protection

Password protection can be set for the X10 and X16 serial control ports as well as the EtherNet card port. The user can select from either Disabled (default), Read/Write, Read Only and No Access (EtherNet only). From the Password Protection drop-down list (*Tools* ⇒ *Combinations* ⇒ *Control Settings*), select the desired access level and click the **Apply** button to open the *Modify Password Setting* window.

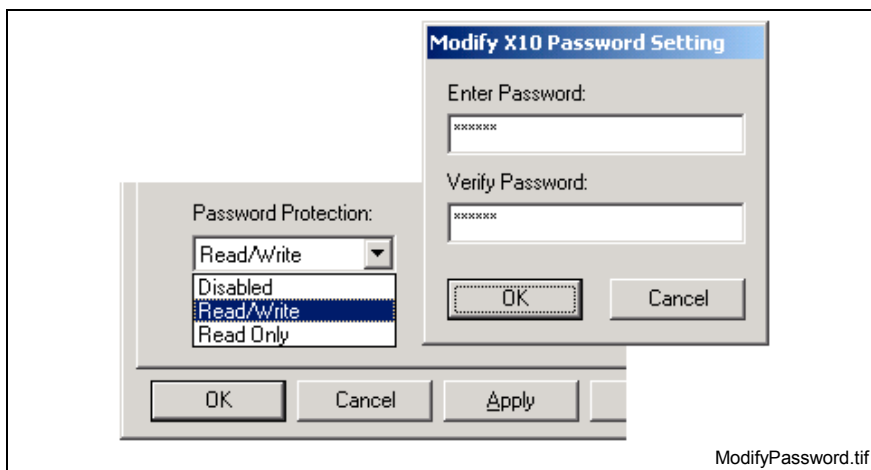


Fig. 2-201: Modify Password Setting

Select a password between 3 to 10 (alpha and/or numeric) characters. The password is not case sensitive and special characters such as "\$" or "%" are not allowed.

Changing the Current Password

To change the password, select Disabled and enter the current password. This will clear the current password and allow the user to enter a different password when the process is repeated.

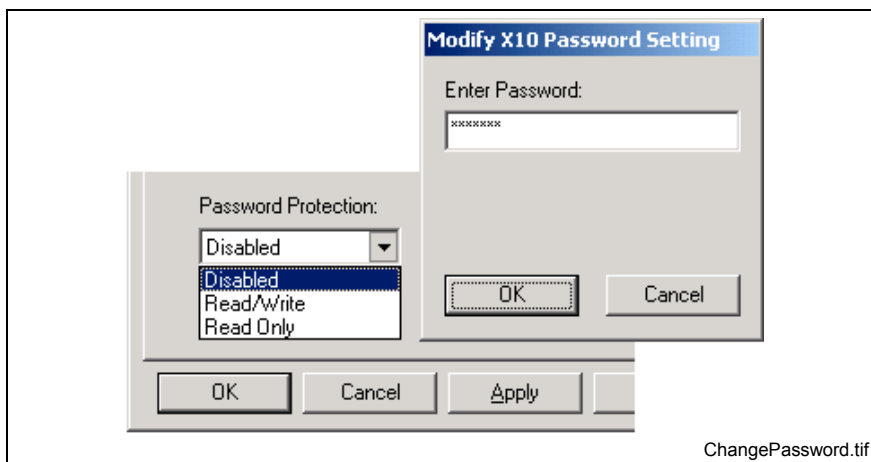


Fig. 2-202: Change Password

Software Restart of Control

VisualMotion control settings that require a reset for changes to take effect are monitored. The following control settings require a restart:

- **X10 and X16:** Baud Rate and Mode
- **Network:** IP Address, Gateway Address and Subnet
- **PLC:** Handshaking

After changes are made to the above settings and the OK button is clicked, VisualMotion checks the current phase of the system and displays the following message when in phase 4.

Note: The user must manually reset power to the control or switch to and from parameter mode for the changes to take effect.

If the system is in parameter mode (phase 2), VisualMotion can perform a software restart of the control.

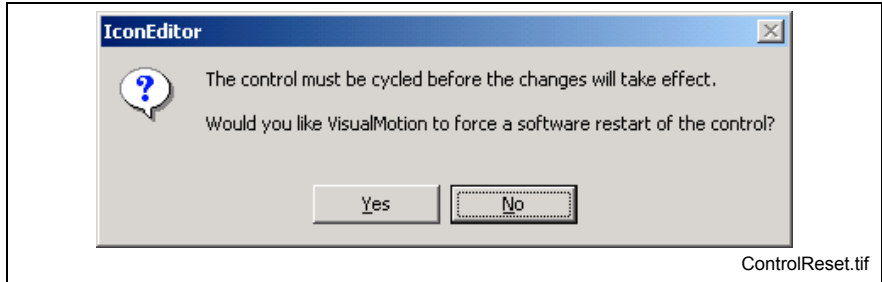


Fig. 2-203: Control Resetting

Options

Selecting **Tools** ⇒ **Options** opens the VisualMotion Options window where the user can set VisualMotion Toolkit programming language and units.

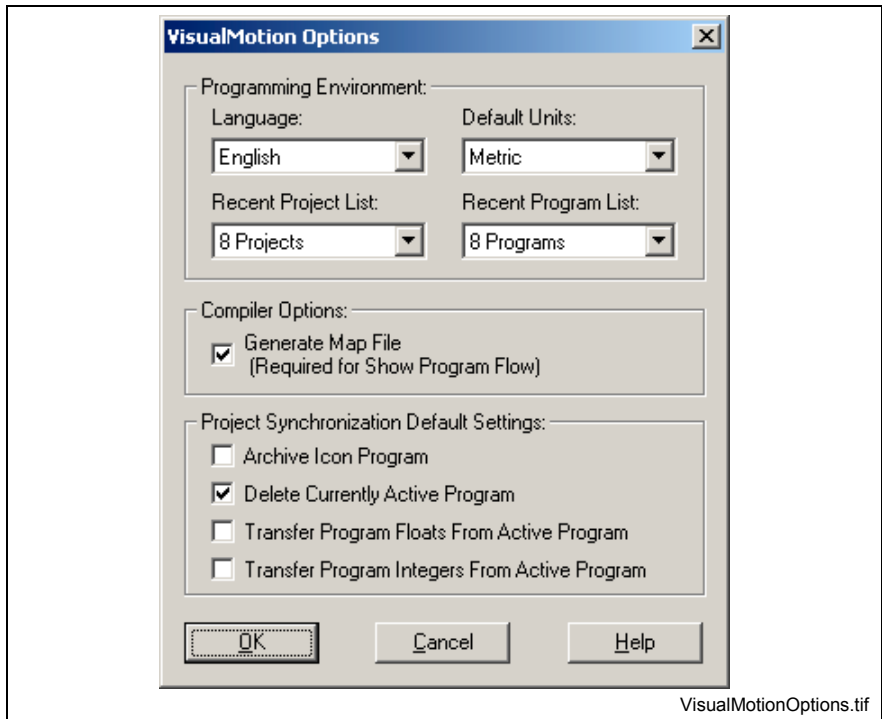


Fig. 2-204: VisualMotion Options Window

Language

Choose between English and Deutsch (German). This is the language used for displaying menus and tools in VisualMotion 9 **only**. Language changes do not take effect for VisualMotion windows that are currently opened. Any opened tool window must be closed and reopened for

language settings to take effect. The following message appears when changing the language of VisualMotion Toolkit.

Note: Refer to Control Settings, *General* tab to change the language of the control and drives.

Default Units

Choose between English and Metric to set the units of measurement that will be used as the default when placing icons that have units selection. *For example:* Single Axis Setup

Note: This selection does not modify any existing icons in a program but is used as the default unit when new icons are added.

Recent Project / Recent Program List:

The **Recent Project** and **Recent Program List** drop-down lists set the maximum number of projects and programs displayed when opening a new instance of VisualMotion Toolkit. Refer to *Open Existing Project* on page 2-9 for details.

Generate Map File

When icon-based programs are compiled, a map file is generated and used to show program flow. On larger programs or older PCs, the generation may take several minutes. By default, this option is checked.

Project Synchronization Default Settings

These options set the default settings for synchronizing project data when switching to online mode. Refer to *Synchronize Project Data* on page 2-11 for details.

The **Archive Icon Program** setting is used to archive the currently opened icon source file (*.str) to the control's memory. The **Delete Currently Active Program** is used to delete the currently active program from the control's memory. To transfer program floats and Integers from the active program to the program that will be downloaded, selecting the appropriate **Transfer Program Floats or Integers from Active Program**.

2.11 The Window Menu

The **Window** menu selection is used to manage the window display of VisualMotion tools, such as I/O Mapper, Variables, etc., while opened. The submenu items within the Window menu are grayed-out when no VisualMotion Tool is opened.

Note: The Window menu is only available in project mode.

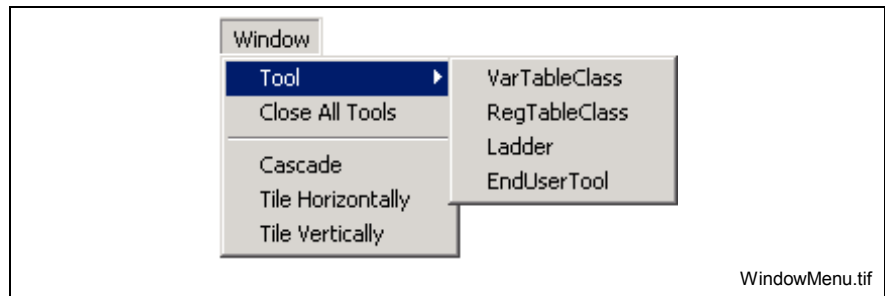


Fig. 2-205: Window Menu

Tool

This menu selection is used to quickly Activate any currently opened VisualMotion Runtime Tool.

Close All Tools

This menu selection closes all VisualMotion Tools that are opened.

Cascade

This menu selection cascades all opened VisualMotion Tools aligning the upper left-hand corner of the title bar.

Tile Horizontally

This menu selection tiles all opened VisualMotion Tools horizontally.

Tile Vertically

This menu selection tiles all opened VisualMotion Tools vertically.

2.12 The Help Menu

The **H**elp menu provides assists to users in the form of an online help system. Any drive help system registered on the PC will be displayed as Registered Help.

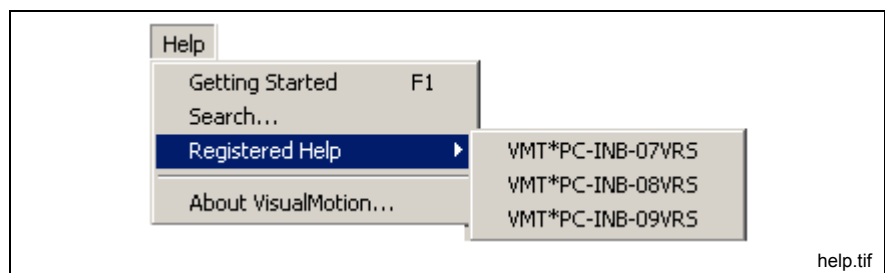


Fig. 2-206 Help Menu

Getting Started

Selecting **Help** ⇒ **Getting Started** or pressing <F1> opens the VisualMotion Help system.

Search

Selecting **Help** ⇒ **Search** opens the help's index search window into which you can type a keyword to go directly to a specific help topic.

Registered Help

During the installation of VisualMotion Toolkit, any Bosch Rexroth help system (VisualMotion or Drive) found registered in the PC's registry is added to the list. Any new help system that is installed will be displayed the next time VisualMotion Toolkit is opened.

SERCOS Drive Parameters

The Parameter Overview window uses this information to display context-sensitive help for a specific drive parameter. From the Parameter Overview window, the user can double-click on a specific parameter to open the Drive Parameter Edit window. Pressing F1 launches the specific help topic for the selected drive parameter. If you do not have the correct help files for your drive, they can be requested from a Bosch Rexroth office.

About VisualMotion

Selecting **Help** ⇒ **About VisualMotion** displays VisualMotion Toolkit version, licensed and Contact information. For a listing of Bosch Rexroth service and Support locations throughout the World, click on the **Support** button.

3 Icon Programming

3.1 Overview

VisualMotion icons are grouped into five (5) icon palettes. Icons are selected and placed in the workspace and connected to create a logical program flow. Icon palettes are only visible when VisualMotion Toolkit is either in "Offline" project mode or when an ".str" icon file is opened in "Service" mode. VisualMotion Toolkit 9 has an "initialization" task used to place icons that are initialized during a phase 2 to phase 4 transition. The icons in the initialization task are also executed when a program is activated. This chapter describes how to setup icons and other VisualMotion commands.

Working with VisualMotion Toolkit's Icon Palettes

VisualMotion Toolkit icon palettes are displayed below the *Project Navigator* window. Five standard palettes are provided for Initialization, Single, Coordinated, ELS and Utility icons. Icon palettes can be selected from the **View ⇒ Icon Palette** menu selection or by clicking on an icon tab, just below the icons in the palette. The initialization icon palette is available only when the **Initialization Task** is selected from the Project Navigator. Icons are selected from the palette using a single click of the left mouse button. The selected icon is placed in the VisualMotion workspace by positioning the cross-hair cursor over the grid area where you want the icon to appear and clicking once.

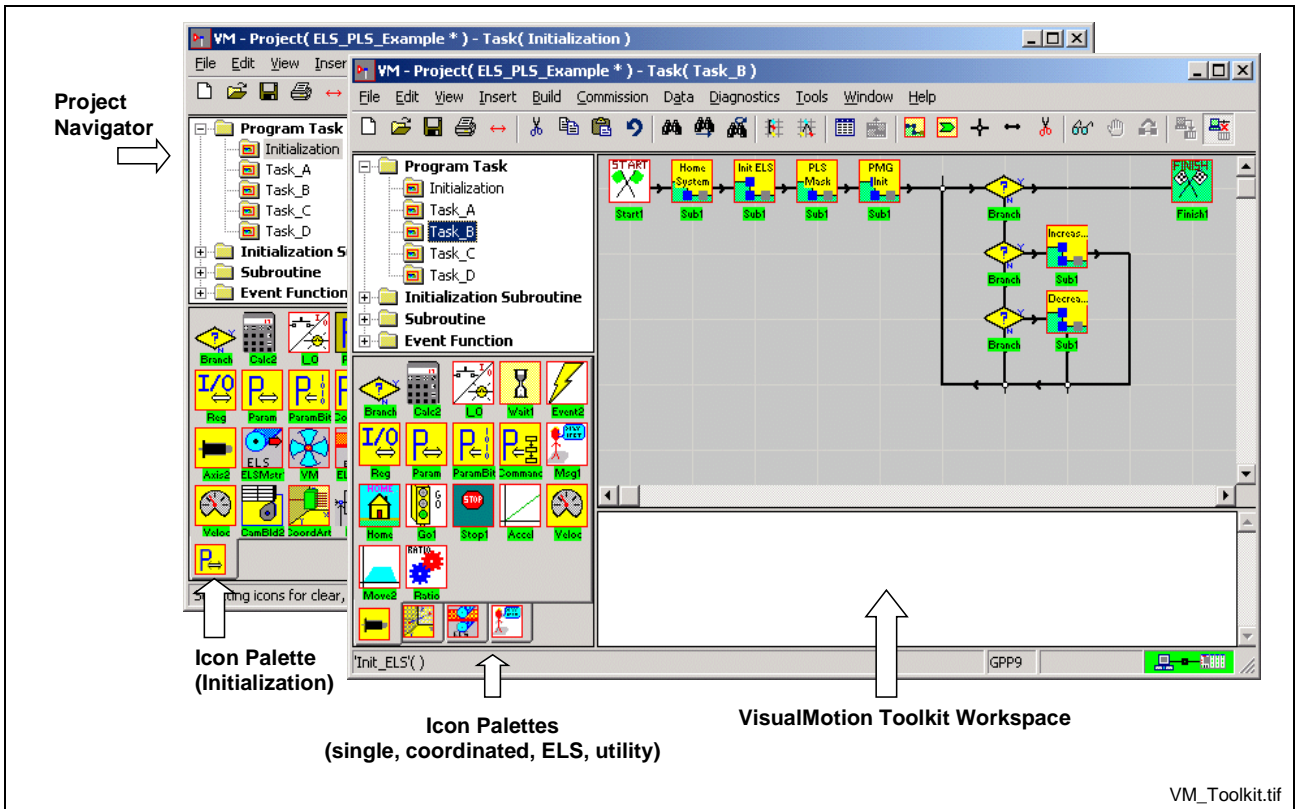


Fig. 3-1: Selecting Icon Palettes

A set of frequently used toolbar buttons is always visible above the workspace regardless of the icon palette selected.

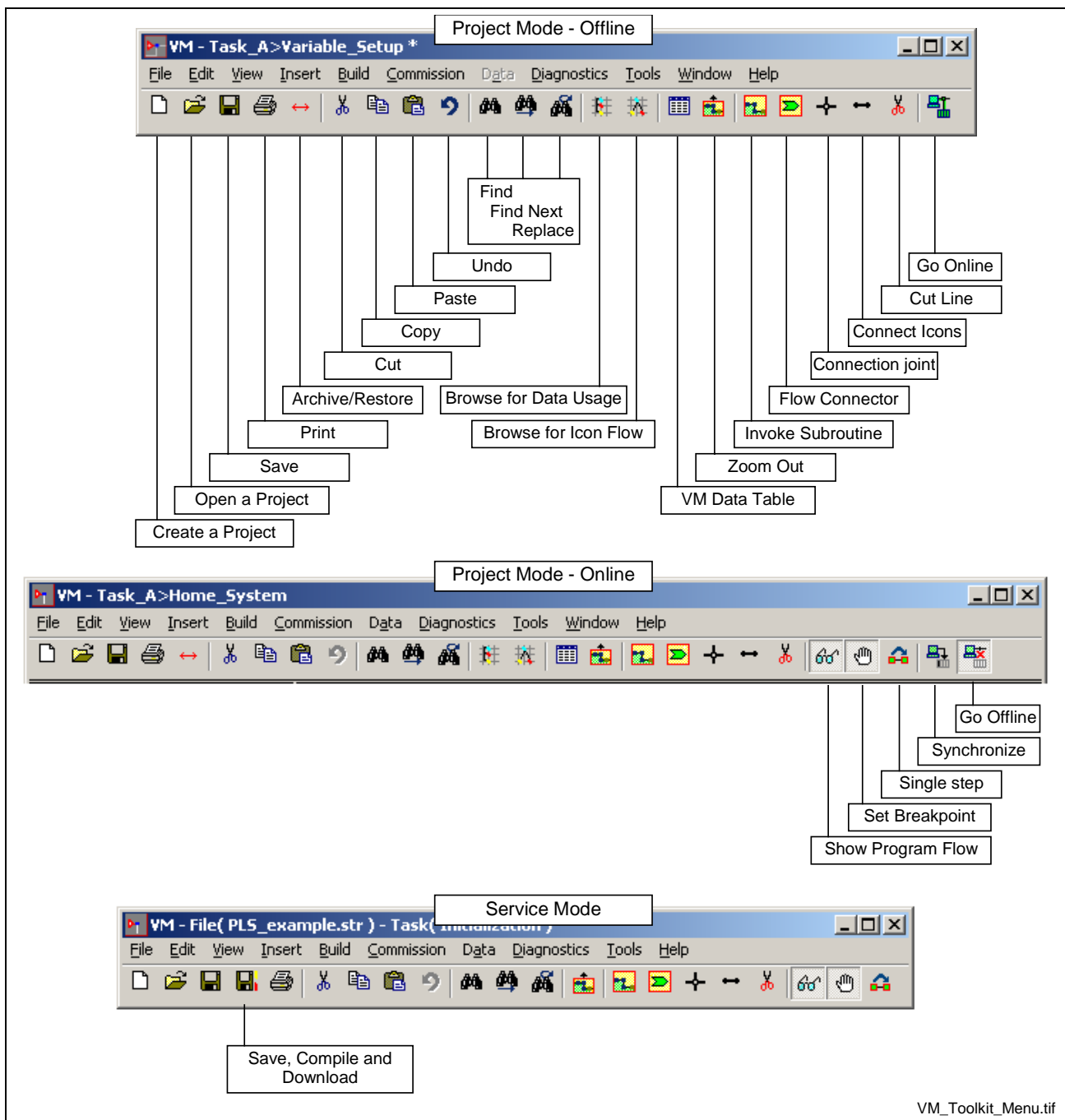


Fig. 3-2: VisualMotion Toolkit Program Menu and Icon Button Bar

Selecting Icons

Single icons are selected by clicking and releasing the left mouse button over the icon. This process creates a red box around the icon, indicating that the icon has been selected.

A group of icons is selected in the same manner. While clicking and holding the left mouse button, drag the cursor and create a window around the icons. Any complete line connections that are included within the selection window will also be selected.

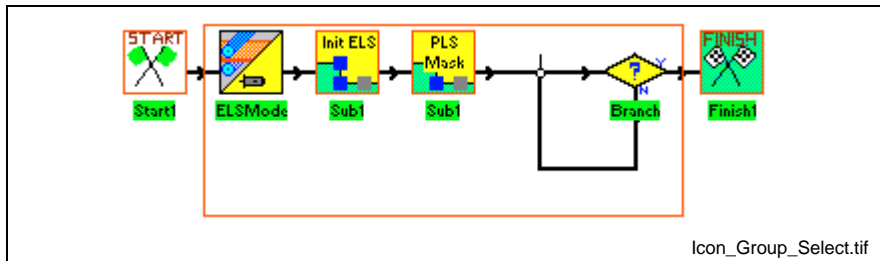


Fig. 3-3: Selecting Program Segment

Cut, Paste and Copy an Icon

Once a selection has been made, the cut and paste menu selections under *Edit* can be used. Once a selection has been cut or copied, it is saved to the Windows clipboard. If the paste button is pressed, the selection will appear in the window and follow the cursor until the left mouse button is clicked.

Deleting an Icon or Selection

To delete or clear a selection or icon, press the delete key or select delete from the *Edit* menu after the icon(s) are selected.

Note: Deletions can be undone by selecting the Undo icon. Only one undo can be performed.

Moving an Icon or Selection

To move an icon(s), click and release the left mouse button over the icon(s), then click and hold the icon(s) again. This process places a red box around the icons and changes the appearance of the icon(s). While holding the left mouse button, drag the selection to a different position. To place the icon(s), release the mouse button.

Program flow connection lines that are touching the icon(s), but are not included within the selection area, will be deleted and must be recreated once the move is complete.

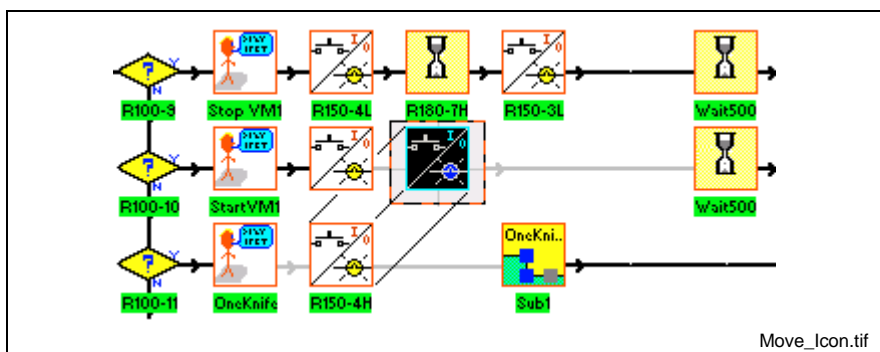


Fig. 3-4: Moving an Icon

Moving a selection window over existing icons will not delete the icons or connection lines unless they are directly being replaced by icons or lines within the selection window. Refer to the following figure.

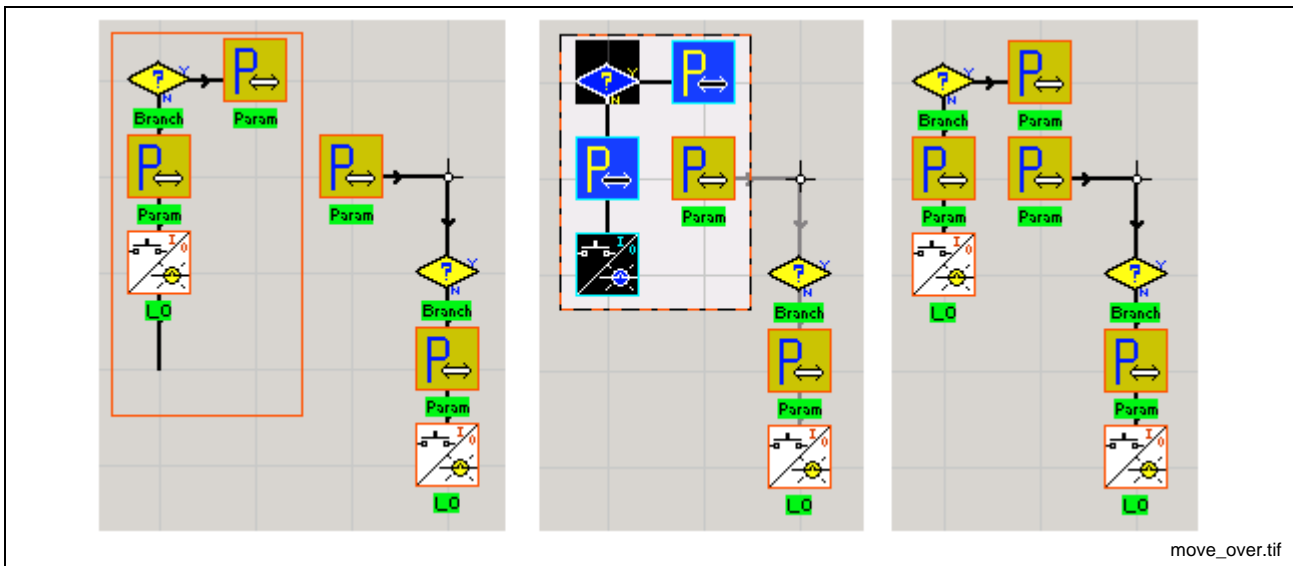


Fig. 3-5: Moving over Existing Icons

Right Mouse Click

The right mouse button is used for additional cut and paste operations. The **Clear**, **Cut** and **Copy** selected options operate the same as the described above. The **Undo** command will only undo the last operation performed.

Clear selected
Cut selected
Copy selected
Paste
Undo
Insert Column
Insert Row
Delete Column
Delete Row

The **Insert Column** and **Row** commands allow you to move several icons at once in order to add a new section to the program. In order to insert a column or a row, an icon or a blank section in the workspace must be selected first. The **Insert Column** command will move all the icons above, below and to the right of the selection, over one space to the right. The **Insert Row** command will move all the icons to the left, right and below the selection, down one space. The **Delete Column** and **Row** commands will delete all the icons above and below or to the left and right of the selection.

Connecting Icons

After you have placed a number of icons on the workspace, they must be connected to indicate the program flow. Most icons have a maximum of three possible inputs and one output. The exception is the Branch icon, which has two outputs.

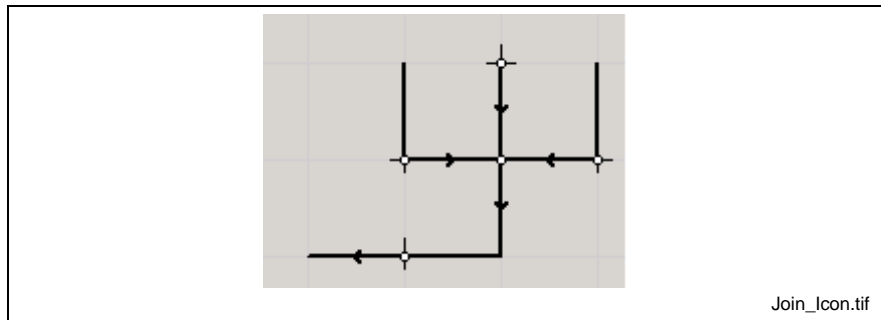
To draw a line, select the Line icon from the icon toolbar. Position the cursor on the first icon that you wish to connect and click. A rectangle appears, surrounding the icon. Move the cursor to the destination icon where the line is to end and click again. VisualMotion automatically draws a line from the first to the second icon, using square corners where appropriate. Arrows on the line indicate the direction of program execution. You may continue this process by clicking successive icons, without re-selecting the previous icon or the Line icon.

You may wish to manually route an interconnect to provide room for additional icons later. Under some circumstances, the Line icon's auto

routing may fail to route an interconnecting line, displaying the *Connection could not be made, try connecting adjacent blocks!* window. Lines may be drawn manually by sequentially clicking on adjacent squares on the invisible workspace grid. A manually placed line may not cross another line, attempting to do so displays an error box.



A **Join** icon makes it possible to connect one line to another from different directions.



Join_Icon.tif

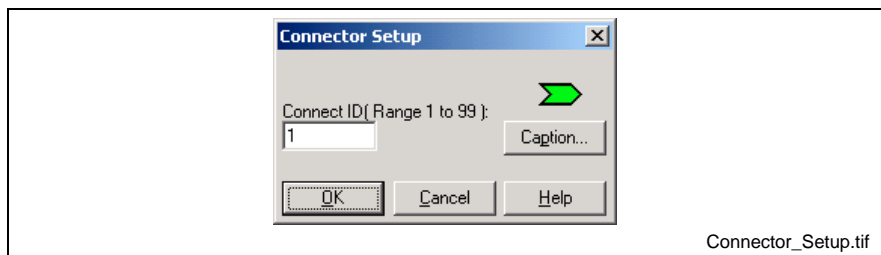
Fig. 3-6: Using Join Icons



A line connecting two icons may be removed by using the Scissors icon from the icon toolbar. Simply select the scissors icon, position the scissors over the line to be removed, and press the left mouse button.



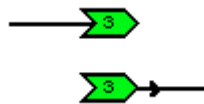
The **Add a connector** icon allows the path of a program to flow between two points that are not connected by a line. This allows complex programs to logistically fit in a relatively small workspace.



Connector_Setup.tif

Fig. 3-7: Connector Icon Setup

When the icon is selected and placed in the workspace a *Connector Setup* window will open and allow you to assign the icon with a Connect ID number (from 1 to 99).



Another **Add a connector** icon with the same Connect ID number can be placed anywhere within the same task. The order of program execution will jump from the first connector icon to the second one.

Icon Labels and Comments

Each Icon has a label which can be displayed when "Icon Labels" is selected from the View Menu. Descriptive comments, up to 80 characters long, are automatically added by VisualMotion describing the selects made in the icon setup window. The user may also modify the default comments. When "Icon Comments" is selected, the comments will appear in a tool tip window as the cursor passes over the icon.

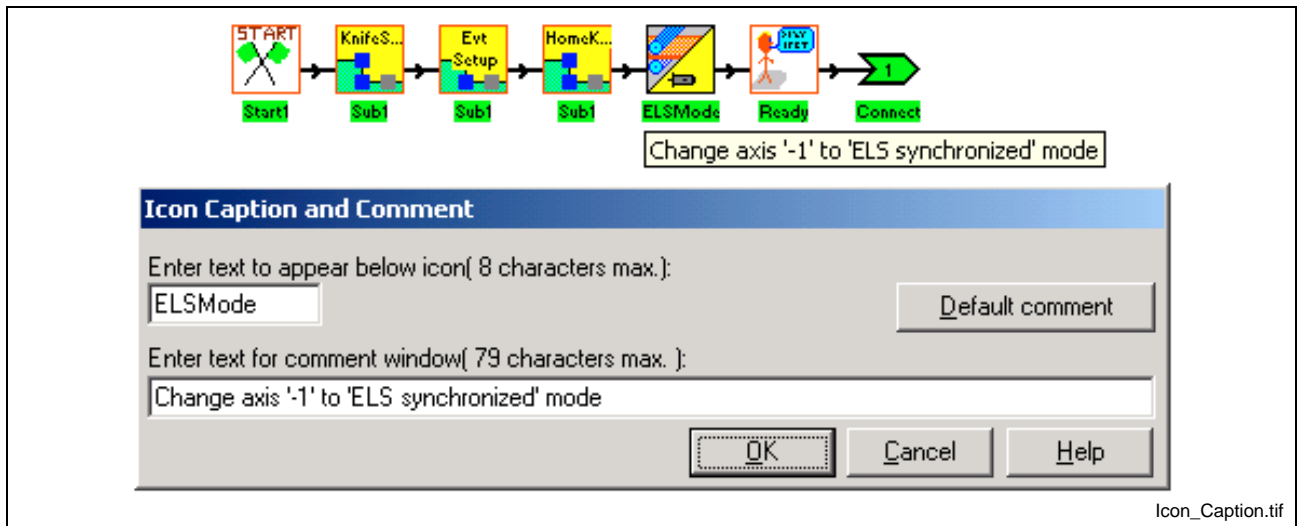


Fig. 3-8: Icon Captions and Comments

The comments can be entered by pressing the **Caption...** button within the icon window. They are used to document the purpose of the icon relative to the program. The Caption under each icon can also be changed independently.

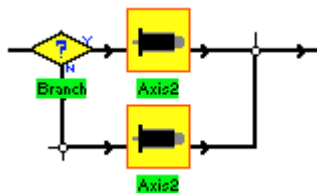
3.2 VisualMotion Toolkit Icons

VisualMotion Toolkit 9 provides five (5) standard palettes for Initialization, Single, Coordinated, ELS and Utility icons. The following table list the icons that are contained in each palette along with a description, valid tasks where they can be used and whether or not they are executed during initialization (P2 ⇒ P4) or at runtime.

Note: **Runtime** icons are executed when encountered in the program flow.

Initialization icons are executed during system initialization (Phase 2 to Phase 4). These icons are executed regardless of where they appear in the program flow. Conditional icons, like the Branch icon, will not stop the icon from initializing.

For Example:



Start1 and Finish1 Icons



Each VisualMotion task icon program must begin and end with a Start1 and Finish1 icon. These icons are located in the Utility Icon Palette. When a new project is created, the Initialization task is displayed with a Start1 and Finish1 icon. VisualMotion Toolkit 9 can automatically place a Start1 and Finish1 icon when an unused Task (A-D) is selected from the **View ⇒ Task** menu selection or Project Navigator. The following figure is displayed when selecting a task that does not contain icons.

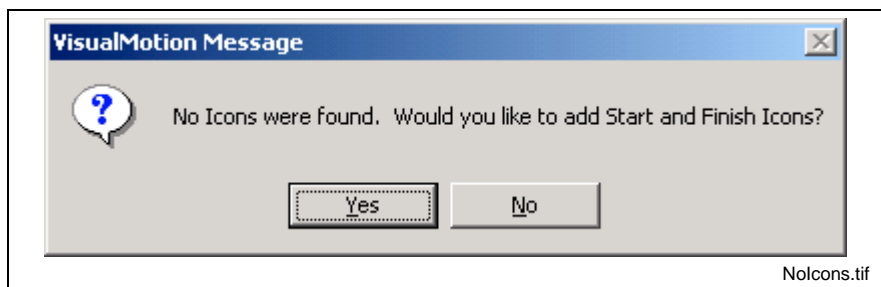


Fig. 3-9: No Icons Found

Initialization Icon Palette




























Icon	Name	Function	Valid Tasks		Executed	Page
			Init.	A-D		
	Branch	Directs program flow	x	x	Runtime	3-20
	Calc2	Calculation and initialization	x	x	Runtime	3-22
	I_O	Sets an I/O register bit	x	x	Runtime	3-89
	PrmInt Parameter Initialization	Initializes a control or drive parameter at program activation	x		Initialization	3-105
	PrmBit Parameter Bit Initialization	Initializes the state of a bit(s) for a control or drive parameter at program activation	x		Initialization	3-104
	Reg Register Transfer	Transfer data between registers and variables	x	x	Runtime	3-108
	Param Parameter Transfer	Transfers a control or drive parameter	x	x	Runtime	3-94
	ParamBit Parameter Bit	Sets bit(s) in control or drive parameter	x	x	Runtime	3-96
	Command	Initiates drive resident commands	x	x	Runtime	3-50
	Msg1 Message	Sets status or diagnostic message	x	x	Runtime	3-93
	Axis2	Defines and initializes an Axis	x		Initialization	3-14
	ELSMstr1 ELS Master Setup	Assigns and initializes ELS Masters	x		Initialization	3-73
	VM Virtual Master Setup	Initializes Virtual Masters	x		Initialization	3-117
	ELSGrp1 ELS Group Setup	Assigns and initializes axes in an ELS Group	x		Initialization	3-62
	Accel Acceleration	Sets the acceleration of an axis	x	x	Runtime	3-13
	Veloc Velocity	Sets velocity for non-coordinated motion	x	x	Runtime	3-117
	CamBld2 CAM Build	Builds a CAM table	x	x	Runtime	3-33
	CoordArt Coordinated Articulation	Sets and initializes coordinated articulation	x		Initialization	3-51
	PID1	Installs and initializes a PID loop	x		Initialization	3-100
	Finish1	Sets the end (finish) of the program flow in each task.	x	x	Runtime	3-87
<p>Initialization - Icons are executed at program activation. Program activation occurs at power-up, when a program is downloaded, or when a different program is activated.</p> <p>Runtime - Icons are executed as part of the logical icon flow.</p>						

Table 3-1: Initialization Icon Palette


















Single Axis Icon Palette

Icon	Name	Function	Valid Tasks		Executed	Page
			Init.	A-D		
	Branch	Directs program flow	x	x	Runtime	3-20
	Calc2	Calculation and initialization	x	x	Runtime	3-22
	I_O	Sets an I/O register bit	x	x	Runtime	3-89
	Wait1	Conditionally holds program execution		x	Runtime	3-122
	Event2	Event Setup Box		x	Runtime	3-80
	Reg Register Transfer	Transfer data between registers and variables	x	x	Runtime	3-108
	Param Parameter Transfer	Transfers a control or drive parameter	x	x	Runtime	3-94
	ParamBit Parameter Bit	Sets bit(s) in control or drive parameter	x	x	Runtime	3-96
	Command	Initiates drive resident commands	x	x	Runtime	3-50
	Msg1 Message	Sets status or diagnostic message	x	x	Runtime	3-93
	Home	Initiates drive homing procedure		x	Runtime	3-88
	Go1	Enables an axis (axes)		x	Runtime	3-88
	Stop1	Halts motion		x	Runtime	3-113
	Accel Acceleration	Sets the acceleration of an axis	x	x	Runtime	3-13
	Veloc Velocity	Sets velocity for non-coordinated motion	x	x	Runtime	3-117
	Move2	Axis position move		x	Runtime	3-91
	Ratio	Sets the ratio between two axes		x	Runtime	3-107

Runtime - Icons are executed as part of the logical icon flow.

Table 3-2: Single Icon Palette

Coordinated Motion Icon Palette

Icon	Name	Function	Valid Tasks		Executed	Page
			Init.	A-D		
	Branch	Directs program flow	x	x	Runtime	3-20
	Calc2	Calculation and initialization	x	x	Runtime	3-22
	I_O	Sets an I/O register bit	x	x	Runtime	3-89
	Wait1	Conditionally holds program execution		x	Runtime	3-122
	Event2	Event Setup Box		x	Runtime	3-80
	Reg Register Transfer	Transfer data between registers and variables	x	x	Runtime	3-108
	Param Parameter Transfer	Transfers a control or drive parameter	x	x	Runtime	3-94
	ParamBit Parameter Bit	Sets bit(s) in control or drive parameter	x	x	Runtime	3-96
	Command	Initiates drive resident commands	x	x	Runtime	3-50
	Msg1 Message	Sets status or diagnostic message	x	x	Runtime	3-93
	Home	Initiates drive homing procedure		x	Runtime	3-88
	Go1	Enables an axis (axes)		x	Runtime	3-88
	Stop1	Halts motion		x	Runtime	3-113
	Path	Coordinated straight line move		x	Runtime	3-96
	Circle	Coordinated circular move		x	Runtime	3-47
	Position	Gets position of coordinated move		x	Runtime	3-104
	Joint	Six axis coordinated move		x	Runtime	3-90

Runtime - Icons are executed as part of the logical icon flow.

Table 3-3: Coordinated Motion Icon Palette

ELS Icon Palette


















Icon	Name	Function	Valid Tasks		Executed	Page
			Init.	A-D		
	Branch	Directs program flow	x	x	Runtime	3-20
	Calc2	Calculation and initialization	x	x	Runtime	3-22
	I_O	Sets an I/O register bit	x	x	Runtime	3-89
	Wait1	Conditionally holds program execution		x	Runtime	3-122
	Event2	Event Setup Box		x	Runtime	3-80
	Reg Register Transfer	Transfer data between registers and variables	x	x	Runtime	3-108
	Param Parameter Transfer	Transfers a control or drive parameter	x	x	Runtime	3-94
	ParamBit Parameter Bit	Sets bit(s) in control or drive parameter	x	x	Runtime	3-96
	Command	Initiates drive resident commands	x	x	Runtime	3-50
	Msg1 Message	Sets status or diagnostic message	x	x	Runtime	3-93
	Home	Initiates drive homing procedure		x	Runtime	3-88
	ELSMODE Mode Change	Switches phase or velocity of ELS axis		x	Runtime	3-79
	ELSAJ1 ELS Adjust	Shifts phase or velocity of ELS axis		x	Runtime	3-61
	Cam	Assigns a CAM to an axis		x	Runtime	3-31
	CamBld2 Cam Build	Builds a CAM table	x	x	Runtime	3-33
	CamAdj Cam Adjust	Phase shift a CAM axis		x	Runtime	3-32
	Index	Initializes a CAM indexer	x	x	Initialization / Runtime	3-37
<p>Initialization - Icons are executed at program activation. Program activation occurs at power-up, when a program is downloaded, or when a different program is activated.</p> <p>Runtime - Icons are executed as part of the logical icon flow.</p>						

Table 3-4: ELS Icon Palette

Utility Icon Palette













Icon	Name	Function	Valid Tasks		Executed	Page
			Init.	A-D		
	Branch	Directs program flow	x	x	Runtime	3-20
	Calc2	Calculation and initialization	x	x	Runtime	3-22
	I_O	Sets an I/O register bit	x	x	Runtime	3-89
	Wait1	Conditionally holds program execution		x	Runtime	3-122
	Event2	Event Setup Box		x	Runtime	3-80
	Reg Register Transfer	Transfer data between registers and variables	x	x	Runtime	3-108
	Param Parameter Transfer	Transfers a control or drive parameter	x	x	Runtime	3-94
	ParamBit Parameter Bit	Sets bit(s) in control or drive parameter	x	x	Runtime	3-96
	Command	Initiates drive resident commands	x	x	Runtime	3-50
	Msg1 Message	Sets status or diagnostic message	x	x	Runtime	3-93
	Start1	Sets the start of program flow in each task	x	x	Runtime	3-109
	Finish1	Sets the end (finish) of the program flow in each task.	x	x	Runtime	3-87

Table 3-5: Utility Icon Palette

Accel



The Accel icon is used to set or change the acceleration of a single non-coordinated axis.

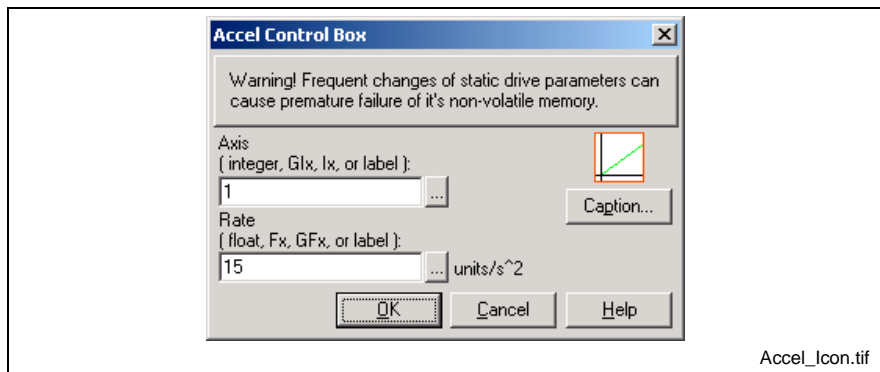


Fig. 3-10: Accel Icon Setup

Axis indicates the axis to accelerate. The axis may be entered as an integer constant, integer variable, global integer variable or an equivalent label.

Rate specifies the acceleration rate in units per second squared. The entry may be a float constant, variable, global variable or an equivalent label. The variable must be greater than 0. A value ≤ 0 will generate an error "469 Axis D accel ≤ 0 or $>$ maximum."

Units: Axis Definition

Single Axis Linear - inches	inches/s ²
Single Axis Linear - mm	mm/s ²
Single Axis Rotary	rads/s ²
Velocity Mode	rads/s ²

Clicking the browse button to the right of each data field opens the VM Data Table.

Warning! Frequent changes of static drive parameters can cause premature failure of it's non-volatile memory.

The warning message that frequent changes to the static drive parameters can damage the non-volatile memory, applies only to the following drives with all versions of firmware:

- ECODRIVE01
- DIAX04
- DIAX03

It is possible to prevent damage to the EEPROM by changing the default setting of the drive parameter S-0-0269 (Buffer Mode) from 0 to 1. The change forces the SERCOS control to disable the parameter buffer on every power up sequence, limiting the number of times parameter changes are written to memory.

Note: No memory problem occurs in the following version of ECODRIVE03:
 ECODR3-SGP-01, ECODR3-SMT-01, ECODR3-SMT-02
 ECODR3-SGP-03, ECODR3-SGP-20, ECODR3-SMT-20

Axis2



The Axis2 icon is used to specify drives that are assigned to the current task. The Axis2 icon can only be used to assign a drive within the Initialization task; it should not be used in a subroutine or event function.

Note: The Axis2 icon is executed during the control's phase 2 to phase 4 transition regardless of where it appears in the program flow. For this reason, conditional programming using Axis2 icon will not provide the desired effect.

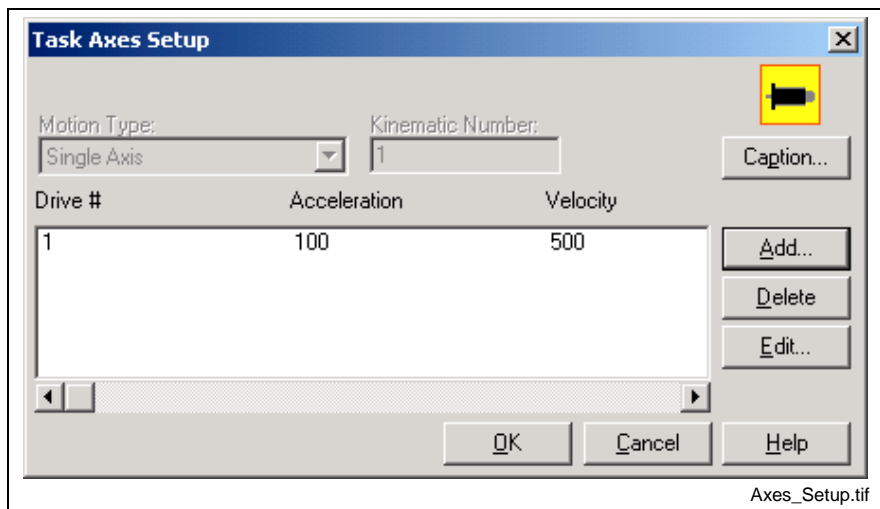


Fig. 3-11: Task Axes Setup

Motion Type provides a pop-down menu of the axis modes available for this occurrence of the axis icon. Only one motion type is allowed for each occurrence of a single icon.

Kinematics Number permits entry of a kinematics library routine number or label identifying the kinematics to be used for **coordinated** motion. Kinematics routines are unique to hardware, consult your Bosch Rexroth sales office for kinematics to drive your hardware.

The drive list box displays setup information for each assigned drive. The type of information depends upon the selected Motion Type. For each drive number the list may contain some of the following: Acceleration, Velocity, Halted, Mode, Trigger 1, Trig 1 Event, Trigger 2, Trig 2 Event, Master number, Slave Ratio, Master Ratio and Change values for each axis.

A horizontal scroll bar on the bottom of the list box allows data to be scrolled into view. Data fields not enabled have "- - -" in them. Axes can be added, edited or deleted with the buttons on the bottom.

The **Add**, **Edit** and **Delete** buttons are used respectively to add a drive, modify axis values, and delete an assigned drive. You may also edit a listed drive by double-clicking the list entry. After editing, the window is closed by clicking the **Cancel** button; all list items are saved as displayed.

Units:

Motion Type	Position	Velocity	Acceleration
Single Axis Linear	inches	inches/min	inches/s ²
Single Axis Linear	mm	mm/min	mm/s ²
Single Axis Rotary	degrees(grads)	RPM	rads/s ²
Velocity Mode	N/A	RPM	rads/s ²

Torque Mode	N/A	N/A	N/A
Coordinated	inches	inches/min	inches/s ²
Coordinated	mm	mm/min	mm/s ²
Ratioed (master)	<i>see single axis definitions</i>		
Ratioed (slaves)	<i>same as master</i>		

In Torque Mode, the list displays only the drive number. In Coordinated mode, the list displays the drive number and its designation as a x-, y-, z-, roll, pitch or yaw axis.

Clicking the browse button to the right of each field opens the VM Data Table.

Selecting **Single Axis** or **Velocity Mode** motion type in the *Task Axes Setup* window displays another window requiring the entry of an axis to be assigned to the drive. The axis may be entered as an integer constant or equivalent label.

Single Axis Setup

This window is used to configure each individual axis that will be used in the VisualMotion icon program.

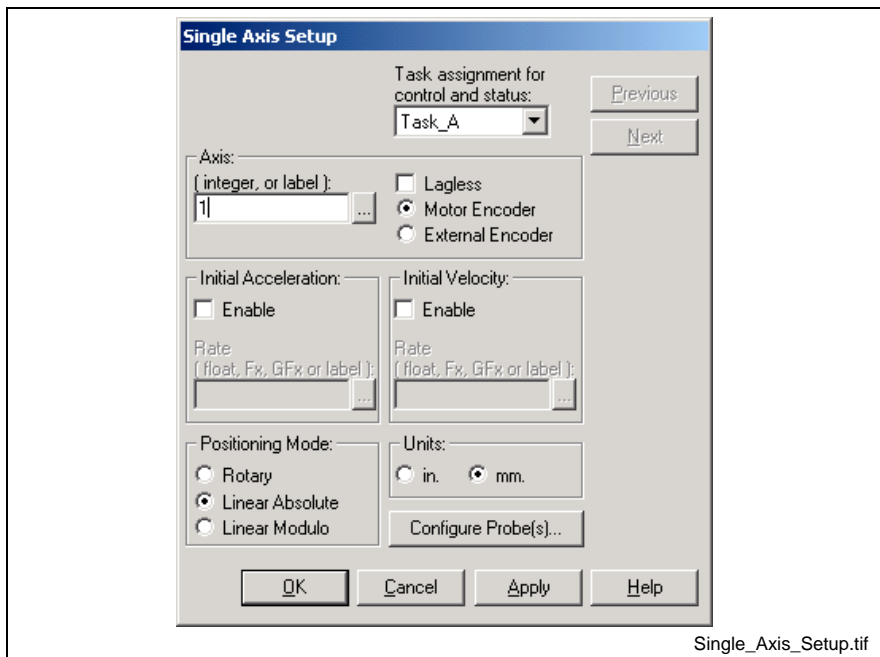


Fig. 3-12: Single Axis Setup

Task Assignment

Since the Axis2 icon can only be placed in the Initialization task, the **Task Assignment** for control and status of the axis must be selected within the *Single Axis Setup* window.

Axis

An **Axis** is identified by its SERCOS address, integer variable or by an assign label in VisualMotion Toolkit. Once addressed, the axis' measuring device is selected as either **Motor Encoder** or **External Encoder** with or without **Lagless** following.

Note: By default, an Axis is configured as **Initially Halted**. This disables the axis at the start of the task and must be enabled using a Go icon in the user program.

Initial Acceleration and Velocity

You may enable and enter an **initial acceleration** and **velocity** as a float constant, global float variable(GF1- GF256), program float variable (Fx), or an equivalent label. If the initial acceleration and velocity are not defined, they must be subsequently specified in the program using the Accel and Veloc (Velocity icons. Both Single and Velocity Mode axes may be assigned linear (units) or rotary (degrees) positioning.

Configure Probe(s)

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

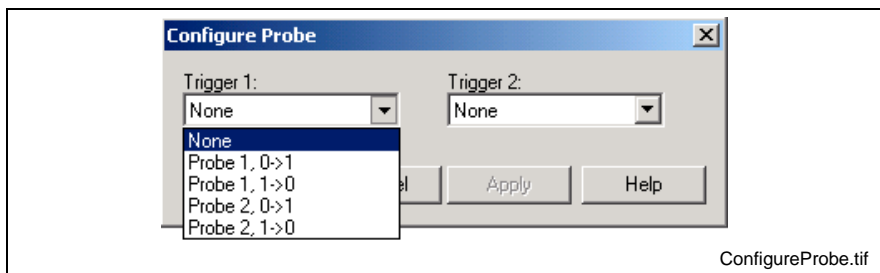


Fig. 3-13: Configure Probe

Coordinated Motion

Selecting **Coordinated** motion type in the *Task Axes Setup* window and clicking the **Add** button opens a *Coordinated Setup* window. Axes are assigned by entering an integer or equivalent label for each axis and clicking the **OK** button. The *Coordinated Setup* window remains open until it is explicitly closed using the **Cancel** button. Although you may assign up to 99 axes using one coordinated motion Axis2 icon, this window is designed to prevent you from assigning multiple axis numbers or labels to a single axis. A pop-up list of labels is available from the field.

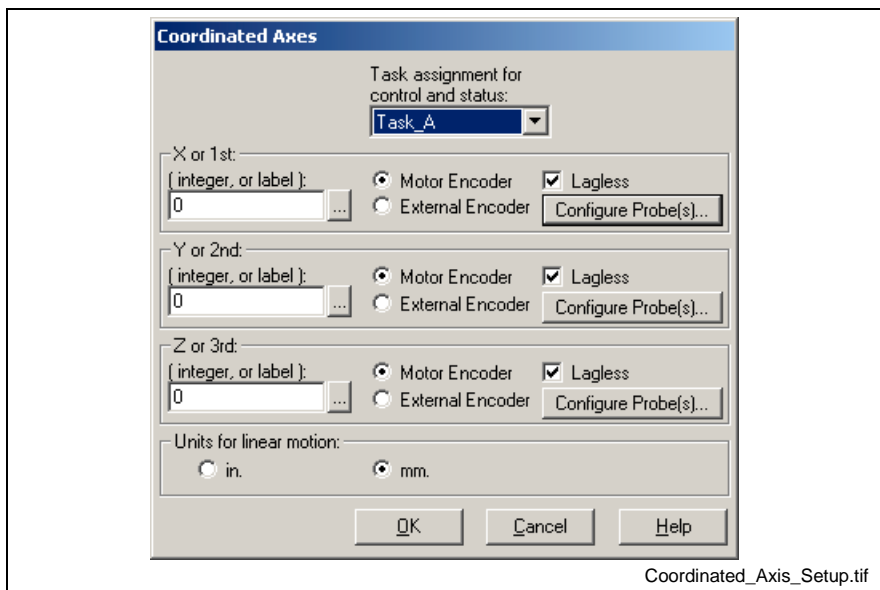


Fig. 3-14: Coordinated Axis Setup

Configure Probe(s)

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

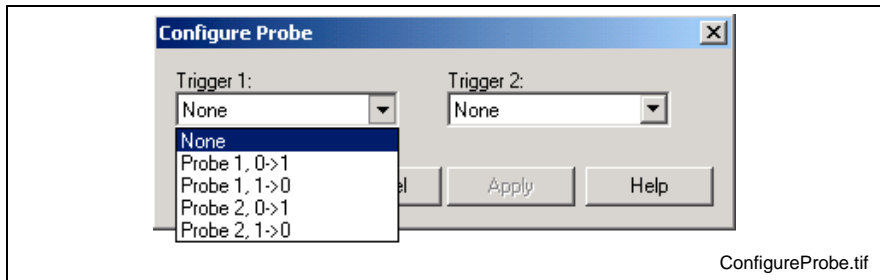


Fig. 3-15: Configure Probe

Velocity Mode

Selecting Velocity Mode displays a similar window; however, **Velocity Change** is also enabled. This permits selection of Ramp instead of the default Step velocity changes. Ramp velocity changes are based on current acceleration rate, whereas Step changes are made at the maximum rate allowed by the drive and motor.

Ratioed Axes

Ratioed Axes mode is used to slave one axis to a master (or controlling) axis. The axes may be assigned an **initial ratio** that determines the number of slave revolutions per revolutions of the master. If an initial ratio is not defined, this ratio must be set using a Ratio icon. A Ratio icon may also be used in the program to change the ratio. More than one axis may be slaved to a single master. A ratioed axis may be assigned linear or rotary positioning, and may use the drive's probe capability to trigger events.

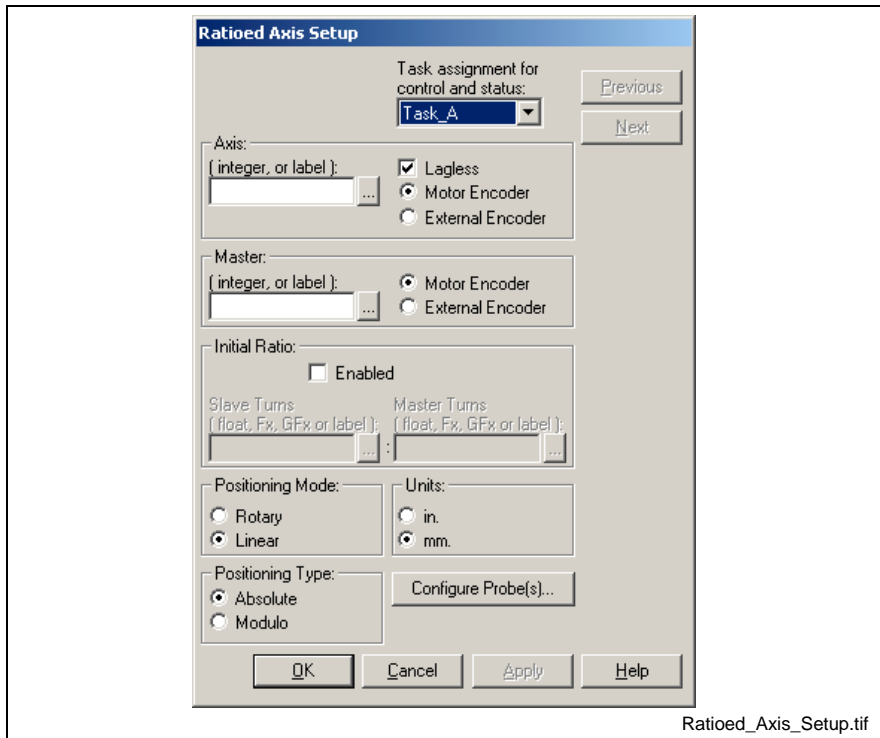


Fig. 3-16: Ratioed Axis Setup

Note: For ELS programs, setting the **Master** field to 0 sets ELS Group 1 as the master.

In control Ratio Mode, the slave axis follows the command position of the master axis, if the master is a coordinated motion or control CAM axis. If the master is any other type of axis, the slave can follow the axis feedback or an external secondary encoder feedback position.

The axis mode can be switched to single-axis or velocity mode from within the program or for manual mode jogging. It is also possible to “slave off” from the master and position the axis independently. Setting the following ratio to 0 will stop the slave from following the master. A step rate (A-0-0037) allows gradual adjustment of the ratio.

Note: To keep the slave axis synchronized when the master axis is jogging, set Axis Control register bit 4 (Synchronized Jog bit). If the slave axis will never be switched to single-axis mode, this bit should always be mapped high (Ratio Mode Enhancements).

Configure Probe(s)

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

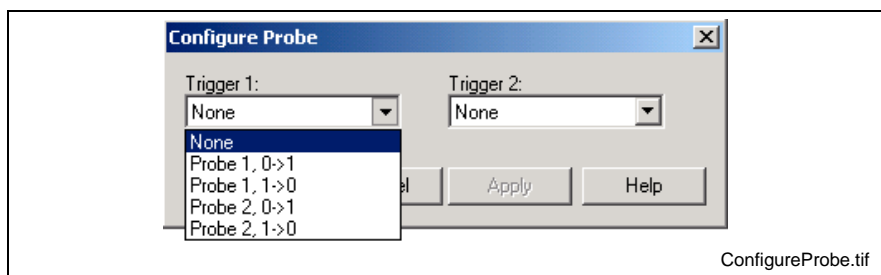


Fig. 3-17: Configure Probe

Torque Mode

Torque Mode is used to apply constant torque to the specified axis. The update rate is limit by the SERCOS cycle.

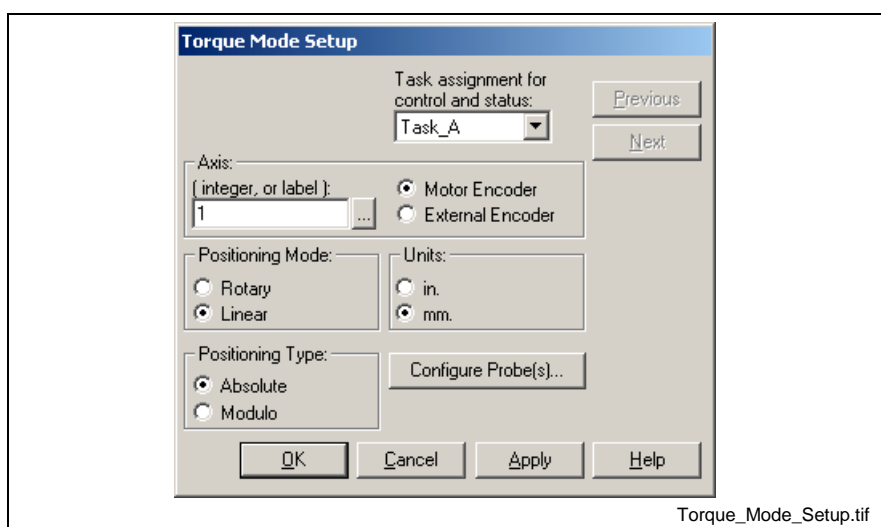


Fig. 3-18: Torque Mode Setup

Configure Probe(s)

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

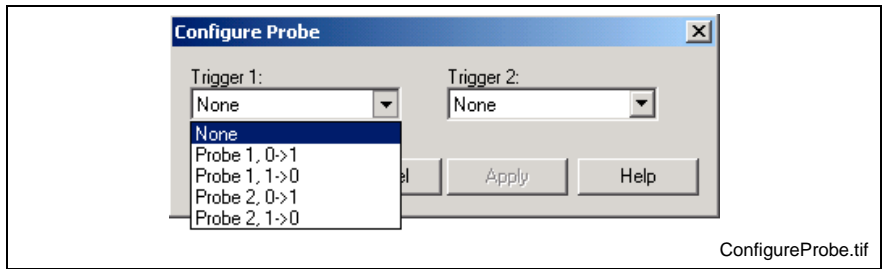


Fig. 3-19: Configure Probe

Torque Following Mode

Torque Following Mode is used to setup a slave axis that follows the torque value of a master axis. The torque value is transmitted from the master to the slave via an analog interface cable.

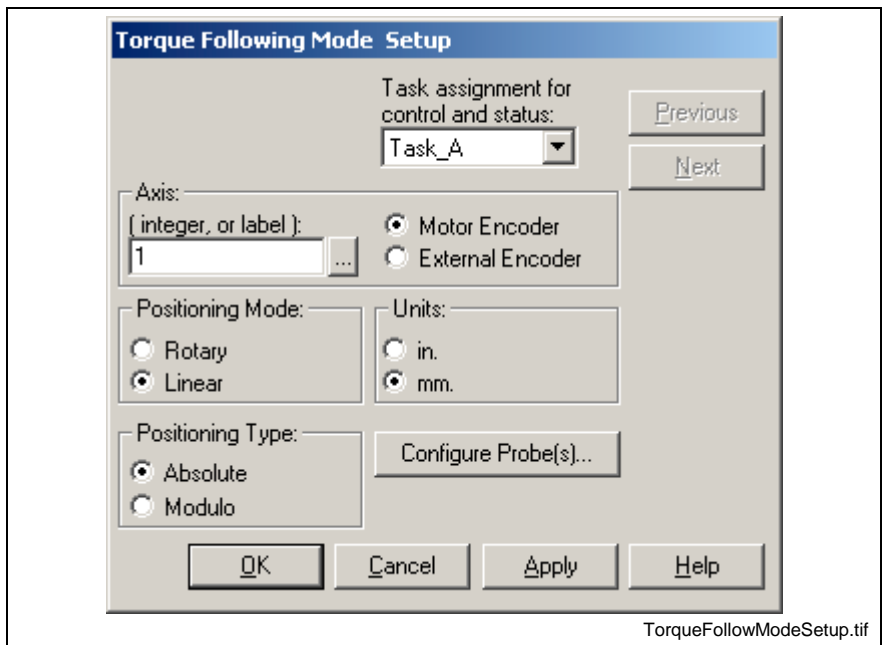


Fig. 3-20: Torque Following Mode Setup

Configure Probe(s)

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

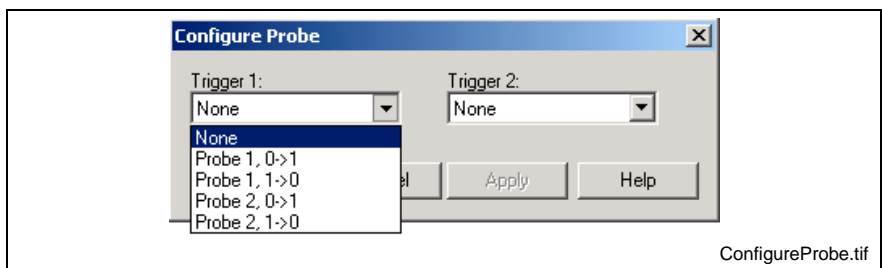


Fig. 3-21: Configure Probe

Branch



A Branch icon re-directs the program flow depending upon a true/false logical value.

Four radio buttons at the bottom of the window permit selection of one of four Branch icon graphics, each with a different positioning of the branch test outputs.

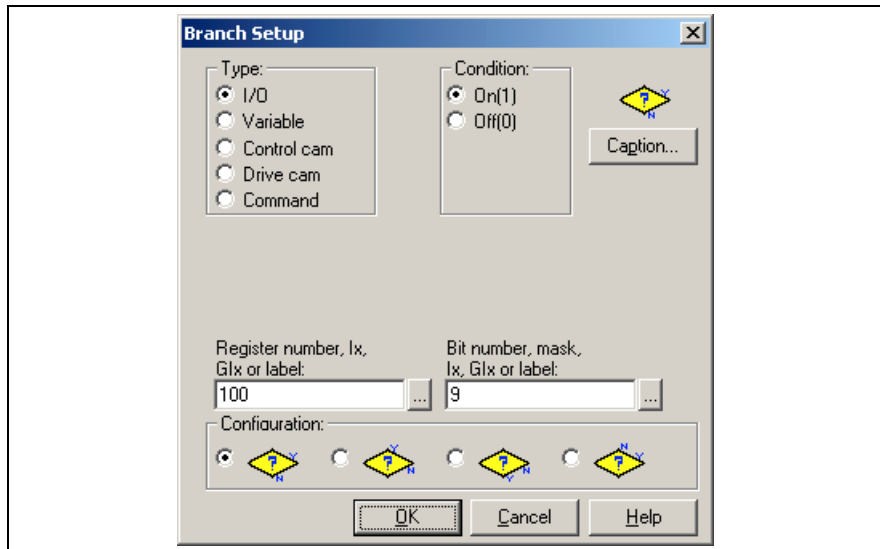


Fig. 3-22: Branch Icon Setup for I/O

Selecting an I/O type branch permits testing a bit in a specified I/O register for a logical (true/false) condition. The I/O register for comparison is specified by an integer or an equivalent label. A bit mask only permits testing of selected bits within the specified register to be tested. The bit mask may be entered as an integer or equivalent label. I/O tests are true/false only the equality or inequality relationships are not allowed.

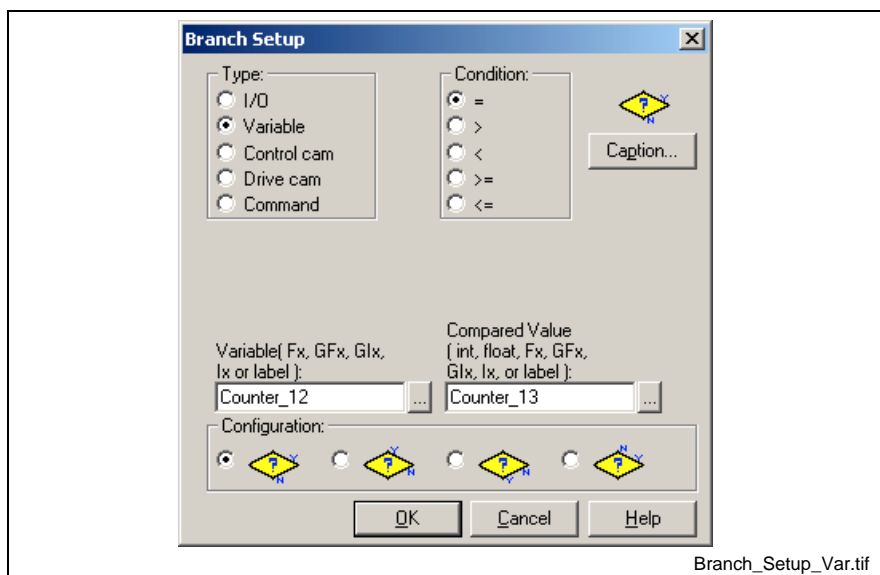


Fig. 3-23: Branch Setup Icon for Variable

Selecting Variable permits a comparison of the contents of an integer variable, float variable or an equivalent label with a compared value (integer or float constant, or an equivalent label). Clicking the browse button to the right of any field opens the VM Data Table.

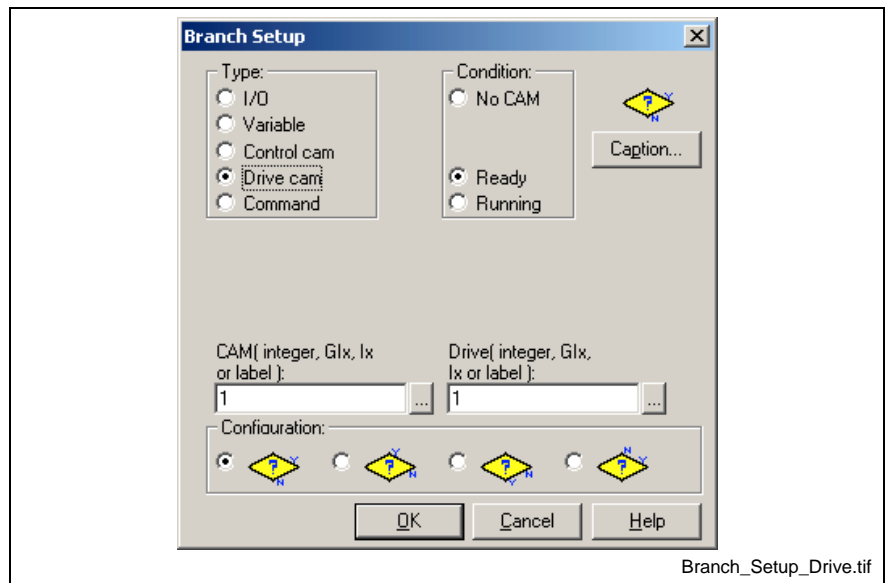


Fig. 3-24: Branch Icon Drive Cam Setup

Selecting Control or Drive CAM permits a specified CAM check for one of three conditions. The branch will return a yes or no value if there is No CAM or if the CAM is Ready or Running.

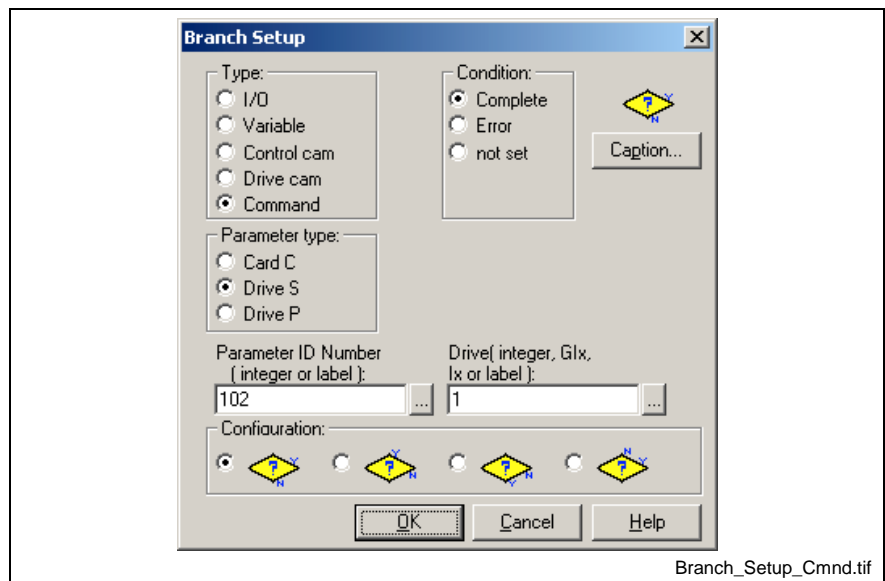


Fig. 3-25: Branch Icon Command Setup

Selecting Command enables the Branch icon to monitor the condition of a Control, or Drive parameters before allowing program flow to continue.

Note: When entering a parameter ID for an S or P class drive parameter (S-0-0102), simply enter to number (102).

Calc2



The Calc2 icon can be used for placing calculations within the program flow or to initialize variables, points, events, zones or PLS elements. By using Calc2 icons in this fashion, the user program is initialized with operational values, thereby eliminating the need to specify them after downloading the program to the control.

The **Resultant** field is used to specify the element for a variable, point table, event, zone table or PLS element.

- Variable {Fx, GFx, GIx, Ix, label}
- Point table element {ABS[x], REL[x]}
- Event table element {EVT[x]}
- Zone table element {ZONE[x]}
- Programmable limit switch {PLS[x]}



Resultants can be selected or created from the VM Data Table window by clicking on the button to the right of the **Resultant** and **Equation** fields.

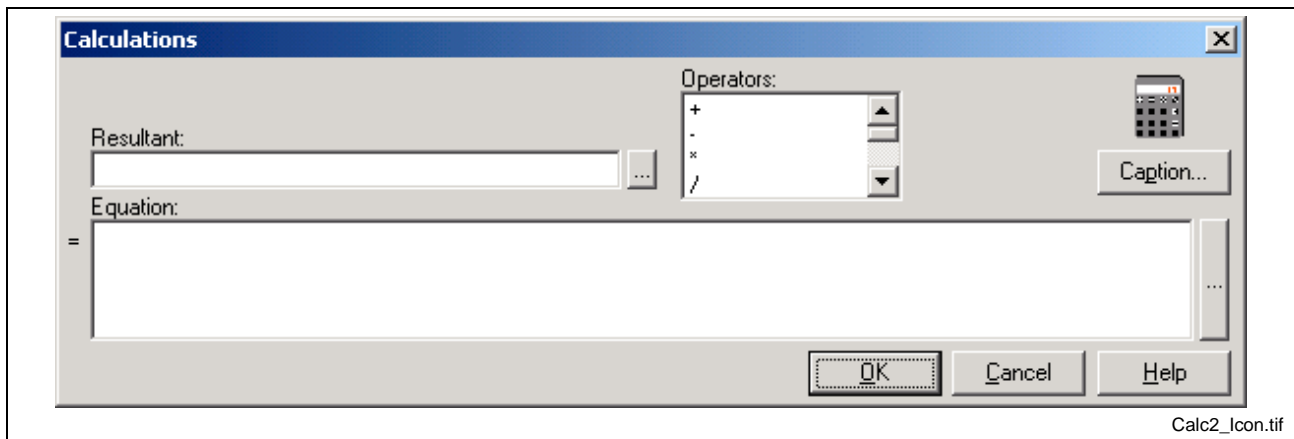


Fig. 3-26: Calc2 Icon Calculations

Enter the expression to be evaluated in the **Equation** field. Operators may be selected from the Operators scrolling menu or entered using the keyboard. Symbolic labels that equate to a constant may be used in equations. The equation is evaluated from left to right. Parentheses can be used to control the order of evaluation.

Note: The Operators menu includes type conversion operators for float and integer data types. These operators should be placed immediately to the left of the operand (i.e., as the equation is parsed from left to right, the compiler first obtains the value, then performs the conversion).

Although an equation can contain a combination of float variables, integer variables, and constants, the data types used for a single operation must match. Data type mismatches are especially easy to overlook when using symbolic label names that do not implicitly identify the data type. Use the "float()" and "int()" conversion operators to force the proper data types.

Operators (Mathematical and Logical)

The operators available for use in the Calc2 icon are listed in the following table:

Arithmetic operators	
+	addition (floating point or integer)
-	subtraction (floating point or integer)
*	multiplication (floating point or integer)
/	division (floating point or integer)
Logical operators Note: Logical operations are limited to unsigned integers.	
&	logical bitwise AND of two integers
	logical bitwise Inclusive OR or two integers (101 011 is 111)
^	logical bitwise "exclusive or" of two integers (101 ^ 110 is 011)
not	logical bitwise inversion of an int (changes "0s" to "1s" and "1s" to "0s")
<<n	Int shift "n" bits left (1-16 bits), low bits padded with 0, high bits are lost
>>n	Int shift "n" bits right (1-16 bits), high bits padded with 0, low bits are lost
Transcendental functions Note: All transcendental functions are in radians.	
sin(n)	returns the floating point sine of an integer or float
cos(n)	returns the floating point cosine of an integer or float
tan(n)	returns the floating point tangent of an integer or float
asin(n)	returns the floating point arcsine of an integer or float
acos(n)	returns the floating point arccosine of an integer or float
atan(n)	returns the floating point arctangent of an integer or float
ln(n)	returns the floating point natural logarithm (base e) of an integer or float
log(n)	returns the floating point logarithm (base 10) of an integer or float
Exponential functions:	
n**p	returns the floating point value of "n" raised to the "p" power
sqrt(n)	returns the floating point square root of an integer or float
Conversion functions:	
%	modulus (the remainder or fractional portion) of the result of the division of 2 integers or floating point numbers
int(n)	returns the integer portion of a floating point value as an integer
float(n)	returns a floating point value equal to an integer
frac(n)	returns a floating point value equal to an integer
absolute(n)	converts a positive or negative integer or float to a positive integer
bintoBCD(n)	converts a binary value to a packed BCD integer
BCDtobin(n)	converts a packed BCD to a binary value

Integers and Floats

Integers are signed or unsigned whole numbers, such as 5 or -3. Integers cannot have a fractional component.

Floats are signed and unsigned numbers using a decimal point, such as 5.0 or -3.0.

Constants, Variables and Global Variables

Constants may be any valid integer or float, such as 3.14159. Constants cannot be modified once a user program is compiled and downloaded to the control.

Variables are names (or labels) used as a symbolic representation for an integer (Ix) or float (Fx) value. Variable can be modified after a user program is compiled and downloaded to the control by selecting **Data ⇒ Variables**.

Global Variables are also names used to represent values for a global integer (GIx) or global float (GFx). Global variables reside in the control's RAM memory and are not retained during power-off. Global variables in user programs are not automatically initialized by VisualMotion's compiler. It is the programmer's responsibility to explicitly initialize global variables in a program.

Direct and Indirect Addressing

VisualMotion stores program variable values and data in separate areas of memory called tables, also referred to as arrays.

A variable table is addressed by using an "I" or "F" prefix before the integer number (i.e., I5 or F20).

An absolute or relative points table or event table is addressed by using an "ABS", "REL" or "EVT" prefix before the table number in brackets (i.e., ABS[1]).

Direct Addressing is performed when a variable or table number is used to access the value directly. Variable labels can be used to access float and integer tables.

For Example:

F15 = 5	accesses the 15 th entry (row) of the float table and writes a value of 5
move = 3.2	accesses the entry associated with the label "move" in the float table and writes a value of 3.2
ABS[2].x = 2	accesses the 2 nd point of the absolute points table and writes a value of 2 for it's x element

Indirect Addressing is performed when a variable or table entry is accessed indirectly by enclosing an integer variable within square brackets. The indirect element must be an integer variable; Global or Program ($GIx, Ix, label$).

For Example:

Using an Integer Variable Indirectly	
$F[I5] = 5.0$ \Downarrow $F[2] = 5.0$	<p>First, the value in I5 is used as the entry number for the float variable</p> <p>If I5=2, then the 2nd entry (row) of the float table is accessed and a value of 5.0 is written</p>
$ABS[I2].x = 4$ \Downarrow $ABS[3].x = 4$	<p>First, the value in I2 is used as the point in the absolute table</p> <p>If I2=3, then the 3rd point of the absolute points table is accessed and 4 is written to it's x element</p>
Using a Label Assigned to an Integer Variable Indirectly	
$F[move] = 3.2$ \Downarrow $F[2] = 3.2$	<p>First, the integer's value assigned to the label "move" is used as the entry number for the float variable</p> <p>If move=I5=2, then the 2nd entry (row) of the float table is accessed and a value of 3.2 is written</p>
$ABS[set].x = 4$ \Downarrow $ABS[3].x = 4$	<p>First, the integer's value assigned to the label "set" is used as the point in the absolute table</p> <p>If set=I2=3, then the 3rd point of the absolute points table is accessed and 4 is written to it's x element</p>

Points Table (Coordinated Motion Variables)

Absolute (ABS[x]) and Relative (REL[x]) points, used to define a geometric path segment for coordinated motion, are stored in an absolute and relative point table. The elements used for an absolute or relative point table are listed as follows:

Resultant	Equation (Elements)
ABS[n]	= {x,y,z,b,s,a,d,j,e1,e2,e3,e4,r,p,ya,e1}
REL[n]	= {x,y,z,b,s,a,d,j,e1,e2,e3,e4,r,p,ya,e1}

Note: Elements in the data structure are separated by a comma(,) and no spaces are allowed between element values.

Element	Description
x	x Cartesian coordinate
y	y Cartesian coordinate
z	z Cartesian coordinate
b	blend radius
s	% of max. speed
a	% of max. acceleration
d	% of max. deceleration
j	% of jerk limiting
e1 - e4	event ID number
r	degree of roll (rate for kinematic #8)
p	degree of pitch
ya	degree of yaw
e1	elbow state/coordinated axis mask

Examples of data structure initialization:

The Calc2 icon can be used to initialize all the elements of a data structure.

```
ABS[1]={5.1,4.2,2.3,1.0,55,66,77,88,1,2,3,4,3.0,4.0,5.0,1}
```

```
REL[1]={1.1,1.2,1.3,1.0,55,66,77,88,1,2,3,4,3.0,4.0,5.0,1}
```

Examples using data structure elements:

The Calc2 icon can be used to write a value to a single element in the data structure.

```
ABS[1].x = 5.1
```

```
REL[5].z = 1.3
```

```
REL[I1].y = 1.2 ; indirect addressing
```

Examples of data transfer between like structures

The Calc2 icon can be used to transfer data between like data structures.

```
ABS[1]= ABS[2]
```

```
REL[1]= REL[2]
```

Events

Events are used to run a specific process in applications. Events interrupt task motion until complete. Using the Calc2 icon, the complete event data structure or a single element can be initialized when the user program is started. The Event data structure is listed in the following table.

Resultant	Equation (Elements)
EVT[n]	= {s,t,d,a,f,m}

Note: Elements in the data structure are separated by a comma(,) and no spaces are allowed between element values.

Element	Description	Selections
s	Status The Status element is read-only and is used as a placeholder. Enter a 0 when initializing. The Calc icon can be used to read the status of this element. Example: F20 = EVT[1].s	0 = Inactive 1 = Queued 2 = Pending 3 = Executing 4 = Done 6 = Repeat
t	Type	0 = <undefined> 1 = Repeating Timer 2 = Time in Coordinated Path 3 = Percent in Coordinated Path 4 = Single Axis Distance 5 = Rotary 6 = Task Input Transition 7 = VME (pre-GPS7 only) 8 = VME (pre-GPS7 only) 9 = Feedback Capture (Probe) 10 = I/O Register Event 11 = PPC-R X1 Input Events
d	Direction	0 = Start of Move (positive edge for input event) 1 = End of Move (negative edge for input event)
a	Argument When Type = 1 -----> = 2 -----> = 3 -----> = 4 -----> = 5 -----> = 6 -----> = 9 -----> = 10 or 11 ----->	This element value is entered based on the selected Type: time in ms time in ms from start/end of move % distance from start/end of move distance from start/end of single move degree to activate the event set to zero, no meaning set to zero, no meaning bit number of input signal
f	Function	Name of Event function
m	Message	text from 0-80 characters in quotation marks " ".

Example of data structure initialization:

The Calc2 icon can be used to initialize all the elements of a data structure.

```
EVT[1]={0,1,0,20,IO_On,"Enables IO"}
```

Examples using data structure elements:

The Calc2 icon can be used to write a value to a single element in the data structure.

```
EVT[1].t = 1
EVT[1].m = "Enables IO"
EVT[I1].f = IO_On ; indirect addressing
```

Example of data transfer between like structures

The Calc2 icon can be used to transfer data between like data structures.

```
EVT[1]= EVT[2]
```

Zone Tables

Zones are used in Coordinated motion programs to describe a volume of space where motion of any kind is prevented. Each zone is defined by the { x, y, z } coordinates of two opposite corners (Point 1 and Point 2) of a cube in space. A Zone status is defined as active or inactive for one or more tasks (or all tasks).

Resultant	Equation (Elements)
ZONE[n]	= {s,a,b,c,d,e,f}

Note: Elements in the data structure are separated by a comma(,) and no spaces are allowed between element values.

Element	Description	Selections
s	Status	1 = All Task 2 = Task A 4 = Task B 8 = Task C 16 = Task D
a	x Cartesian coordinate, Point 1	
b	y Cartesian coordinate, Point 1	
c	z Cartesian coordinate, Point 1	
d	x Cartesian coordinate, Point 2	
e	y Cartesian coordinate, Point 2	
f	z Cartesian coordinate, Point 2	

Example of data structure initialization:

The Calc2 icon can be used to initialize all the elements of a data structure.

```
ZONE[1]={1,2.5,5,4,3.2,4.5,6}
```

Examples using data structure elements:

The Calc2 icon can be used to write a value to a single element in the data structure.

```
ZONE[1].a = 2.5
```

```
ZONE[I1].b = 5 ; indirect addressing
```

Example of data transfer between like structures

The Calc2 icon can be used to transfer data between like data structures.

```
ZONE[1]= ZONE[2]
```

PLS Data

Unlike variables, points tables and event tables, PLS data structures cannot be completely initialized within one Calc2 icon. Each element of a PLS must be configured individually.

Typically, PLS configurations are first created using the PLS Tool under **Commission** ⇒ **PLS**. Afterwards, the Calc2 icon can be used to modify different elements in the configuration. The following three PLS types can have their elements written to using the Calc2 icon.

- Option Card PLS
- Control PLS
- Drive PLS

Option Card PLS

A user program can contain only one active Option Card PLS. The following tables list the elements of a Option Card PLS.

Master Elements

The following Master elements are available for modification using the Calc2 icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].ox	Phase Offset	PLSP[1].o1=20	add an offset of 20 to PLS Master 1	C-0-2943
x = Master range 1-8				

Switch Elements

The following switch elements are available for modification using the Calc2 icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].onx	On Position	PLSP[1].on3=40	switch 3 On position is 40	C-0-2920
PLSP[1].offx	Off Position	PLSP[1].on3=60	switch 3 Off position is 60	C-0-2921
PLSP[1].outx	Assigned Output	PLSP[1].out3=2	switch 3 is assigned to output 2	C-0-2922
x = switch range 1- 96				

Output Elements

The following Output elements are available for modification using the Calc2 icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].ltx	Lead Time	PLSP[1].lt2=500	add a lead time of 500µs to output 2	C-0-2931
PLSP[1].lgx	Lag Time	PLSP[1].lg2=500	add a lag time of 500µs to output 2	C-0-2932
PLSP[1].ssx	PT (Time Duration)	PLSP[1].ss2=1000	maintain output 2 On for 1000µs	C-0-2933
PLSP[1].hxx	Hysteresis	PLSP[1].h2=0.1	add a Hysteresis of 0.1 to the output 2	C-0-2936
PLSP[1].mdx	Mode	PLSP[1].md2=0	set mode of output 2 to Lag Time (0=Lag Time, 1=PT)	C-0-2934
PLSP[1].drx	Direction	PLSP[1].dr2=0	set direction of output 2 to positive (0=pos, 1=neg, 2=pos/neg)	C-0-2935
x = output range 1-32				

Control PLS

VisualMotion Control PLS is stored with the user program as a separate table, and can be assigned to an ELS Master, ELS Group or Real Master. A user program can contain a maximum of two active Control PLS. The following table lists the elements of a Option Card PLS that can be easily modified using the Calc2 icon.

Syntax	Description	Variable Range	Example	Comment
PLS[x].a	Number	1-40 when t=3 or 4, Drive No. 1-6 when t=5, ELS Master 1-8 when t=6, ELS Group	PLS[1].a=2	set drive number of PLS 1 to 2
PLS[x].o	Phase Offset		PLS[1].o=20	add an offset of 20 to PLS 1
PLS[x].r	Output Register		PLS[1].r=70	set output register of PLS 1 to 70
PLS[x].t	PLS Master Type	3 = Drive's primary feedback 4 = Drive's secondary feedback 5 = ELS Master 6 = ELS Group	PLS[1].t=3	set master type of PLS 1 to 3 (drive's primary feedback)
PLS[x].on1-on16	On Position		PLS[1].on1=10	switch 1 On position is 10
PLS[x].off1-off16	Off Position		PLS[1].off1=20	switch 1 Off position is 20
PLS[x].lt1-lt16	Lead Time (ms)		PLS[1].lt1=5	switch 1 lead time is 5
x = PLS number 1-2				

Drive PLS

VisualMotion 9 supports drive based PLS configurations for DIAX04 and ECODRIVE03 digital drives. The drive based PLS is stored on the drive in parameter lists. The elements of this list cannot be modified individually. Modifying one element of the list requires sending the entire list over the service channel.

Syntax	Description	Example	Comment	Parameter
PLSD[x].r	Output Register	PLSD[1].r=72	set output register of drive 1 to 72	A-0-0009
PLSD[x].t	PLS Master Type	PLSD[1].t=1	set master type of drive 1 to 1 (0=disable, 1=motor encoder, 2=external encoder)	P-0-0131
PLSD[x].on1-on16	On Position	PLSD[1].on1=10	switch 1 On position for drive 1 is 10	P-0-0132
PLSD[x].off1-off16	Off Position	PLSD[1].off1=20	switch 1 Off position for drive 1 is 20	P-0-0133
PLSD[x].lt1-lt16	Lead Time (ms)	PLSD[1].lt1=5	switch 1 lead time for drive 1 is 5	P-0-0134
x = drive number range 1-40				

CAM



The CAM icon is used to associate a CAM to a slave axis and to supply coefficients for using the CAM. To work properly, the axis must be configured as an ELS slave and synchronized to the master. The CAM can be drive or control resident.

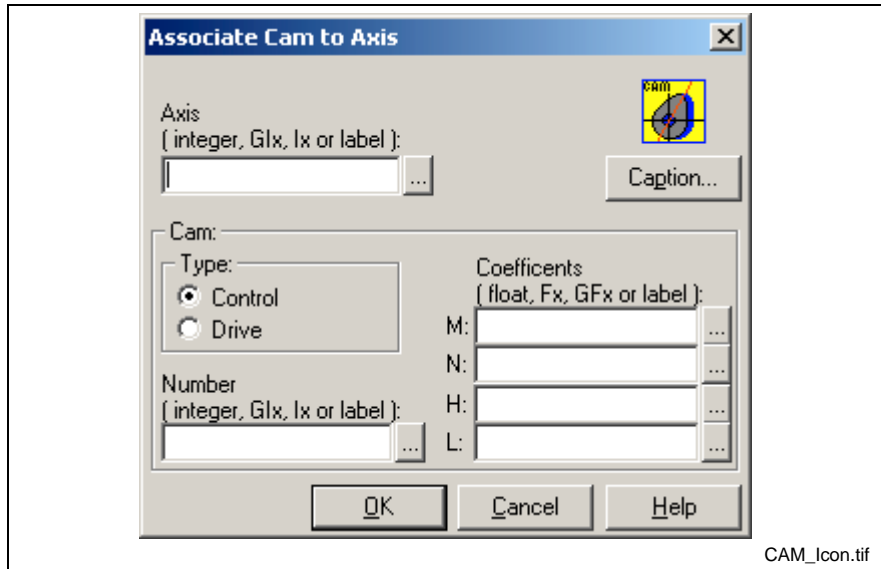


Fig. 3-27: CAM Icon Setup

Axis: Specifies which axis is associated with the CAM. The entry may be an integer, global integer variable (Glx), program integer variable (Ix) or an equivalent label.

CAM Number: Specifies which CAM table to use. The entry may be an integer, global integer variable (Glx), program integer variable (Ix) or an equivalent label (Range 1 to 37 for control CAMs and 1 to 2 for drive CAMs.)

Coefficients: M, N, H and L may be entered as floats, global float variables (GFx), program float variables (Fx), or equivalent labels (control CAMs only).

Note: The H coefficient of a CAM verifies based on the target firmware:

For GPP7/ 8 the H value is scaled in units.

For GPP9 or Drive, the H value is scaled in percent.

CAMAdj



This icon selects and starts a phase adjust. There are two phase offset values for a CAM axis: a **Master Phase Adjust**, and a **Slave Phase Adjust**. The **Master Phase Adjust** shifts the position in the CAM table relative to the master position. The **Slave Phase Adjust** shifts the position of the slave axis. Since it is not related to the shape of the CAM, the **Slave Phase Adjust** is not multiplied by any of the CAM factors.

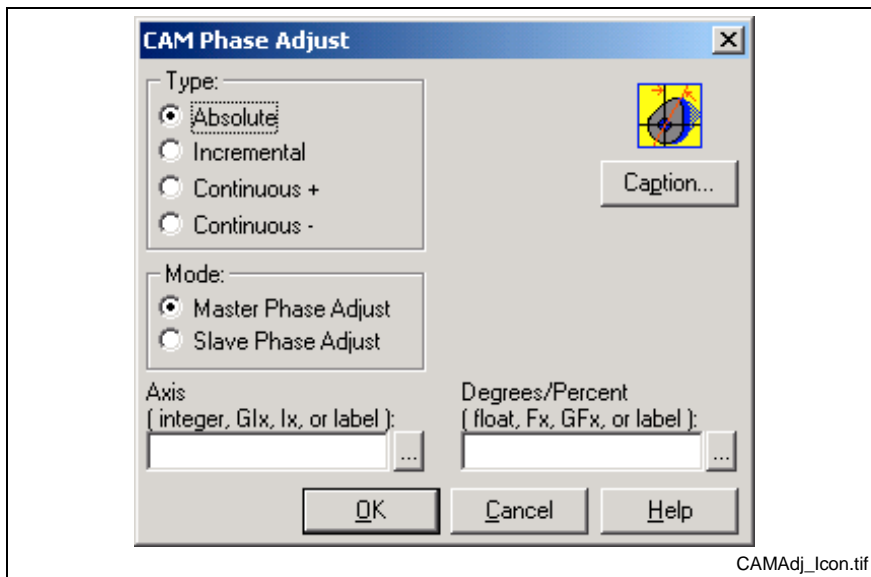


Fig. 3-28: CAMAdj Icon Setup

The **Master Phase Adjust** adds the phase adjust value to the sampled ELS master position before the slave position is referenced in the CAM table. For control CAMs, the **Master Phase Adjust** is written to axis parameter A-0-0151. For drive based CAMs, the **Master Phase Adjust** is written to drive parameter P-0-0061.

The **Slave Phase Adjust** adds the phase adjust value to the output position of the CAM table. For control CAMs, the **Slave Phase Adjust** is written to axis parameter A-0-0161. For drive based CAMs, the **Slave Phase Adjust** is written to axis parameter A-0-0151.

Type selects the method of adjustment:

If **absolute**, the **degrees or percent** edit field is the new offset.

If **incremental**, the **degrees** edit field is added to the current offset. If in phase mode and sum exceeds 360, it rolls over. If in velocity mode the sum is limited to -100 or +300 percent.

If **continuous +** or **continuous -** (phase sync mode only), a velocity dependent amount is added to the degree offset.

The control can perform only one phase adjust at a time. If two different phase adjust icons are used the second one will have no effect until the previous phase adjust is complete. Bit 4 in the axis status register is set to (0) when a phase offset is in progress and (1) if the phase offset is complete.

CamBld2



The control can build a CAM on-line based on a set of input positions. Program instructions use the control's ABS point table or float array and an internal utility to build an internal control CAM which is stored on the Control or Drive as the CAM number indicated.

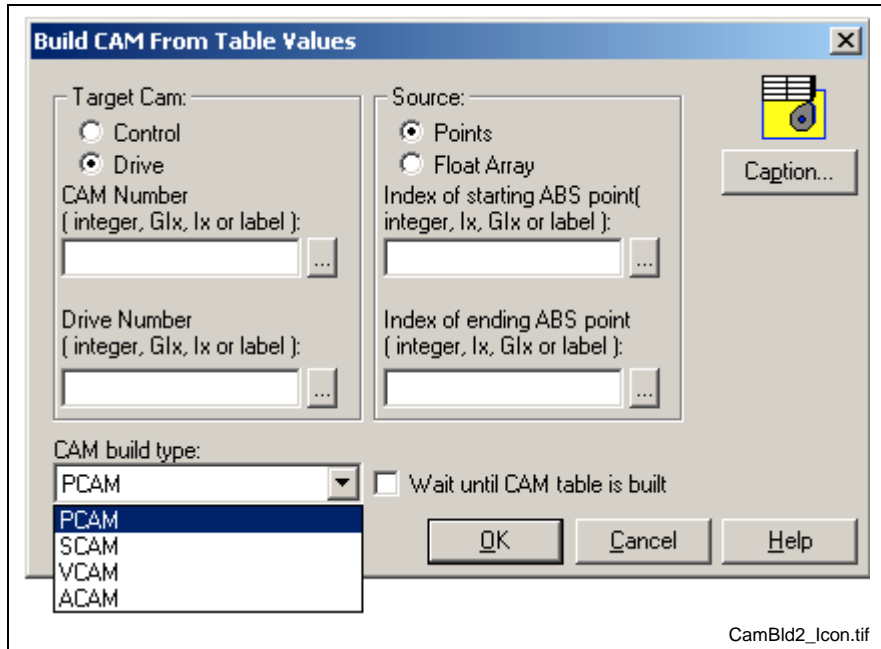


Fig. 3-29: CamBld2 Icon Setup

Target Cam:

From the Target Cam section, the user selects the location to store the control CAM. When **Control** is selected, the user can only specify the CAM number. When **Drive** is selected, the user can specify the CAM number and the Drive number to where the CAM will be stored.

CAM Number

This number indicates the CAM table number to build. The entry may be an integer, global integer variable (Glx), program integer variable (Ix) or an equivalent label (range 1 to 37 for control CAMs and 1 to 2 for drive CAMs). The CAM cannot be in use when this icon is executed.

Drive Number

This number (drive CAMs only) indicates which drive to store the CAM. The entry may be an integer, global integer variable (Glx), program integer variable (Ix) or an equivalent label.

CAM Build Type:

The **PCAM** build type accepts input in the form of a table of target positions. The **VCAM** build type accepts a velocity profile as input. The **ACAM** build type accepts an acceleration profile as input and outputs a normalized profile. When **Spline** is selected, the control builds the CAM by connecting the points with third order splines. A minimum of 5 and a maximum of 200 user defined points are allowed. The X elements of the point table are the master positions, and the Y elements are the corresponding slave positions. All GPP9 CAM profiles are normalized.

Source

The user selects the source of the control CAM. When selecting **Points**, the user must specify an *Index of starting ABS point* and an *Index of the ending point*.

ABS Starting Point

Specifies the starting ABS point number used for CAM generation. The entry may be an integer, global integer variable (Glx), program integer variable (Ix), or an equivalent label.

ABS Ending Point

Specifies the ending ABS point number used for CAM generation. The entry may be an integer, global integer variable (Glx), program integer variable (Ix), or an equivalent label.

Points Example:

This example sets up a 3 points table that contains an x and y position for both a master and slave, starting at ABS point 4 and ending at ABS point 6.

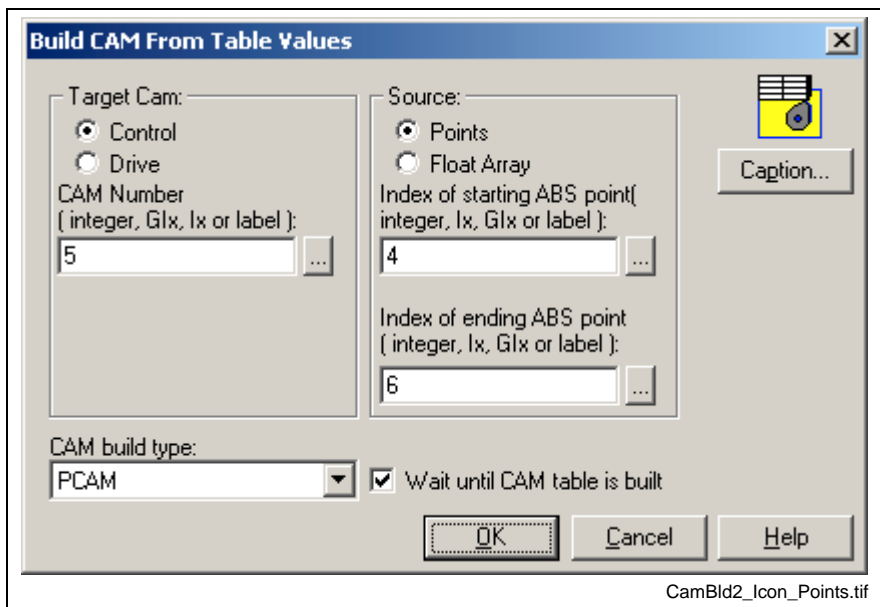


Fig. 3-30: CamBld2 Icon Points Configuration

Once compiled and downloaded to the control, the user can write to the points table by selecting **Data ⇒ Points** and modifying the points directly, or within the user program using the Calc2 icon.

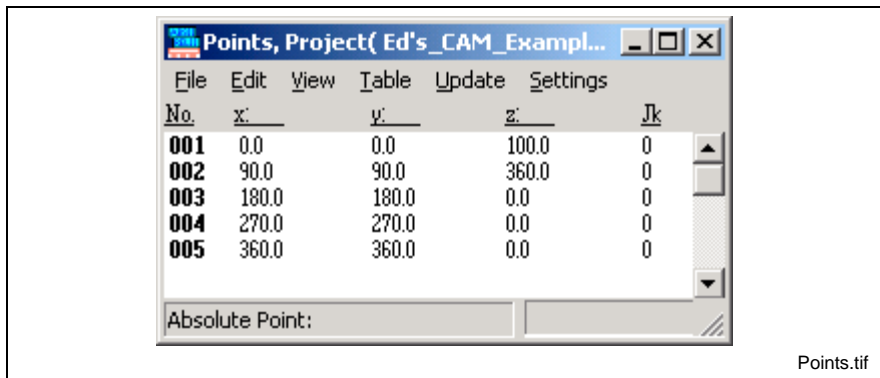


Fig. 3-31: Points Table

No.	X	Y	Z	Jk (Jerk)
004	First master position	First slave position	Slave Scaling value	Shaping Factor, 0 –100%
005	Second master position	Second slave position	Slave scaling position	not applicable
006	Last master position	Last slave position	not applicable	not applicable

Table 3-6: Example Points Table Description

When selecting **Float Array**, the user must specify the *Index of starting float variable*. This number identifies the first float variable number that will be used to store the data for the float array table.

Float Array Example:

This example sets up a float array table that contains a count of 3 equally spaced CAM values, starting at float variable 390.

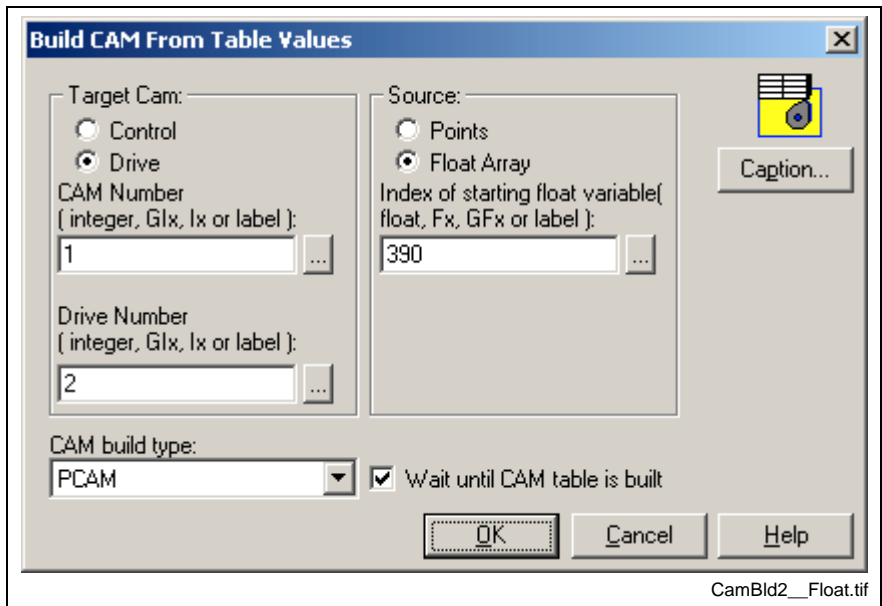


Fig. 3-32: CamBld2 Icon Setup for Float Array

Once compiled and downloaded to the control, the user can write to the float variables by selecting **Data ⇒ Variables** and modifying the variables directly, or within the user program using the Calc2 icon.

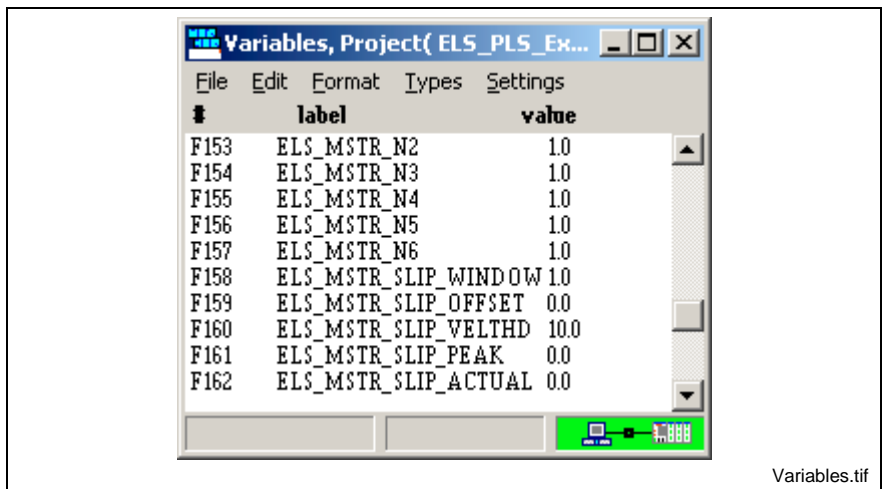


Fig. 3-33: Variables Table

Float Variable	Description
F1	Number of equally spaced CAM values
F2	Slave scaling value
F3	Slave scaling position
F4	Shaping factor, 0 – 100%
F5	1 st CAM value
F6	2 nd CAM value
F7	3 rd CAM value

Table 3-7: Example Float Table Description

General Information

The ABS point elements may be changed within the program by using CALC statements. Changes in the point table do not affect the CAM until the CamBld2 icon is executed. It is necessary to size the point table at compile-time to allow enough points for the profiles that will be needed. The x value of the first point must be 0, and the x value of the last point must be 360.

The CAM generation may take one second or longer. The 'wait' checkbox can be cleared to exit this icon immediately and keep executing instructions while the CAM is being built. The branch icon can be used to check if a CAM is ready for activation. If the 'wait' checkbox is checked, the program flow will be stopped in this icon until the CAM is ready for activation.

The CamBld2 icon can be used to store a CAM to an inactive location on the control or drive. After the CAM has been built, the CAM icon can select it for an axis.

Because the CAM is stored in nonvolatile memory on the control or the drive, it is not necessary to execute this command each time through the program. A flag variable can be set and checked the next time through the program to avoid long delays when starting the program. For on-line changes, a register bit or variable should be checked each time through the program loop to avoid continually generating the CAM, which consumes control resources and can slow down the program.

Control CAMs enter the control as text files in CSV format, the kind most spreadsheets generate. Drive CAMs enter a single column text file indicating the slave position.

If the CSV file for control CAMs contains less than 1024 points, an algorithm within the control fills in the missing points. As a rule, a CSV CAM file should contain at least 200 points, anything less than that does not sufficiently define the CAM and unexpected results may occur. Drive CAMs must contain exactly 1024 points.

Errors at runtime

The selected CAM is currently active for any axis.

The point range exceeds the bounds of the point table.

Less than two points are defined.

The CAM number is not valid (out of range or drive is not configured).

An error occurred when sending the CAM to the drive.

When using PCAM option, and the first x position isn't 0 and the last x position isn't 360.

The x position exceeds the modulo of the master.

CAM Indexer



Index CAMs are control CAMs that use equations to compute a position, as opposed to a normal CAM, which uses a point table. They operate in real-time, allowing their start and end positions to be freely changed within the next index cycle. This makes them ideally suited for high-speed film feed applications where the seal time must be held constant over line speed.

Note: When using Indexer CAMs, increase the SERCOS cycle time (C-0-0099) to 4ms or more to allow time for the equation calculations.

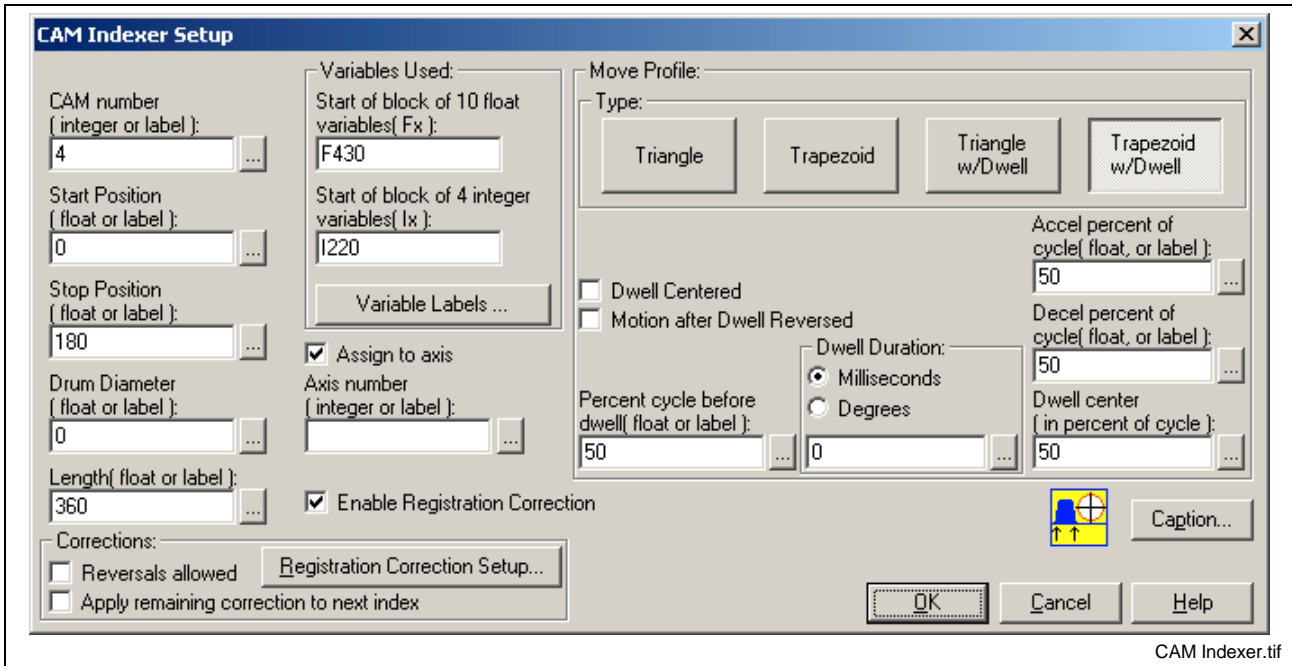


Fig. 3-34: CAM Indexer Setup

Number of Allowable CAMs

CAM Indexers are identified by a unique number. The number of active CAMs (control table CAMs and CAM Indexers combined) in a VisualMotion control system is limited by the amount of processing power.

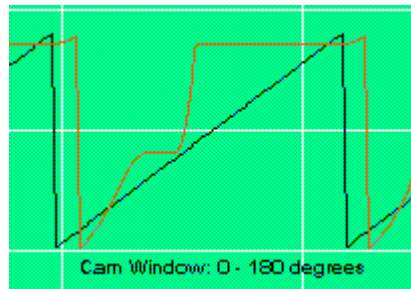
- GPP controls limit the number of active running CAM Indexers to 4.
- GPP control systems can have a maximum of 40 built CAMs.

CAM Indexer Setup Fields

CAM number: assigned based on the number of allowable CAM Indexers for a GPP VisualMotion system.

Start Position (Ps): Defines the virtual or real master position (in degrees) at which the CAM index profile starts.

Stop (Pe) Position: Defines the virtual or real master position at which the CAM index profile stops.



These two parameters define the CAM's cycle time, T_c ,

$$T_c = \text{ABS}(\text{Pe}-\text{Ps})/\text{Vm}$$

where ABS is the absolute value of the difference and Vm is the velocity of the master in degrees/sec.

Length (D): Defines the distance (in EE units) the slave axis moves during an index cycle. The velocity profile developed depends on the start/stop positions, profile type, index length and master speed.

Drum Diameter (Dia): Allows for automatic scaling of the index distance into degrees. The following equation is applied:

$$\text{Angle} = (\text{D}/\text{Dia})(360/\pi)$$

Internal CAM units (degrees) are computed automatically through this parameter.

Note: When the Drum Diameter is set to zero, its default value is $360^\circ/\text{rev}$. The benefit of this is that internal 64 bit math can be used to minimize rounding errors.

Variables Used: Defines the starting point for a block of variables that will be assigned values for the CAM Indexer function. The value of these variables is set according to the data entered in this window.

10 float variables (F(n) , F(n+1), F(n+2)... (Fn+9)

CAMI_01_ PRO_LENGTH	;CAM 1 Indexer, profile length
CAMI_01_ PRO_START	;CAM 1 Indexer, start position of profile
CAMI_01_ PRO_STOP	;CAM 1 Indexer, stop position of profile
CAMI_01_ DIAMETER	;CAM 1 Indexer, drum diameter
CAMI_01_ EVENT_TIME	;CAM 1 Indexer, time period before end of move
CAMI_01_ ACCEL	;CAM 1 Indexer, percent cycle for acceleration
CAMI_01_ DECEL	;CAM 1 Indexer, percent cycle for deceleration
CAMI_01_ DWELL	;CAM 1 Indexer, time period of dwell
CAMI_01_ PRE_DWELL	;CAM 1 Indexer, percent cycle before dwell
CAMI_01_ RESERVE_F1	;CAM 1 Indexer, reserve float 1

4 integer variables (I(n) , I(n+1)... I(n+3)

CAMI_01_ FLAG1	;CAM 1 Indexer, flag word 1
CAMI_01_ PRO_TYPE	;CAM 1 Indexer, profile type
CAMI_01_ RESERVE_I2	;CAM 1 Indexer, reverse integer 2
CAMI_01_ RESERVE_I1	;CAM 1 Indexer, reverse integer 1

Assigning a CAM Indexer

Note: Using the same CAM Indexer in a VisualMotion system can cause unexpected errors and instabilities.

Assign to Axis or ELS Group

A CAM Indexer can be assigned to any ELS Group or to any slave axis within an ELS Group as long as the same CAM Indexer is not used more than once. In GPP systems, CAM Indexers are created in the CAM Indexer icon and then assigned to an ELS Group in the ELS Group icon. In order for CAM Indexer to function in an ELS Group, the Lock On / Lock Off State Machine must be disabled.

Assigning a slave axis to a CAM Indexer requires that the axis be declared as a control CAM and then assigned within the CAM Indexer icon. A CAM Indexer can function in an assigned slave axis while the ELS Group has an active Lock On / Lock Off State Machine.

Move Profile

The following types of Move Profiles are available:

Profile	Description
Triangle	Step Acc/Dec of 50%
Triangle (S-Curve)	Ramp Acc/Dec of 50%
Trapezoid (Fixed)	Step Acc/Dec of 33%
Trapezoid	Step Acc(x%)/Dec(y%), $x+y \leq 100\%$
Trapezoid (S-Curve shaper)	Ramp Acc(x%)/Dec(y%), $x+y \leq 100\%$
Triangle w/Dwell	Step Acc/Dec, center dwell
Triangle w/Dwell (S-Curve)	Ramp Acc/Dec, center dwell
Trapezoid w/Dwell (S-Curve)	Ramp Acc/Dec, center dwell
Trapezoid w/Dwell	Step Acc/Dec, center dwell
Trapezoid w/Dwell	Step Acc(x%)/Dec(y%) $x+y \leq 100\%$, Sw% dwell

Table 3-8: Move Profiles

CAMs with ramped acceleration (S-shaped velocity profile) are defined as third order polynomials, those with stepped acceleration (ramp velocity profile) as second order polynomials.

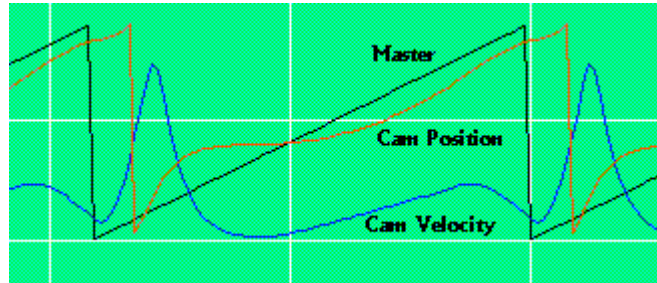
The Triangle profile without the S-Curve shaper has the simplest equation, thus it takes the least amount of time to execute, but has maximum jerk. Selecting the S-Curve shaper takes longer but minimizes jerk.

The Trapezoid profile provides CAM shaping through an accel./decel. percent setting but takes the longest to execute due to equation complexity.

CAMs with dwells are slightly more complex than those without dwells and so require slightly more time to execute than those without dwells.

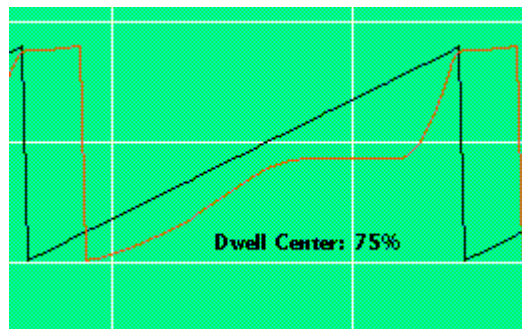
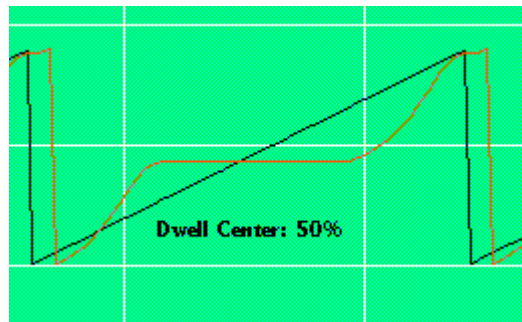
CAM Acceleration/Deceleration(Acc/Dec) - Some CAM types (see table) use these parameters to define the percent of their cycle time devoted to acceleration and deceleration.

Example: A Trapezoid CAM profile with an acceleration setting to 20% and deceleration setting to 50% would step accelerate for 20% of its cycle time, constant velocity for 30% and decelerate for 50%.

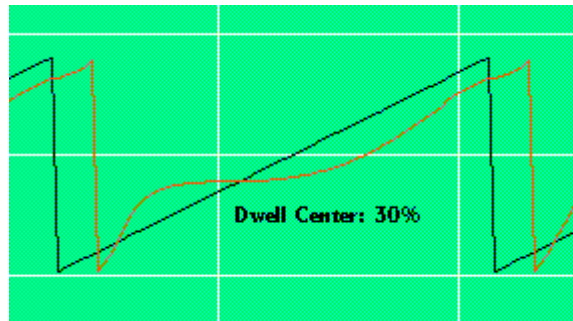


CAM Dwell (Cw) - Some CAMs (see table) use this parameter to determine the center of the dwell. This is defined as a percent, which is limited to 25% to 75% of the CAM cycle.

Example1: $Cw = 50\%$ $Ps = 0$ $Pe = 360$ then dwell center
 $Pw = (360 - 0) * 0.5 + 0 = 180 \text{ deg.}$



Example2: $C_w = 30\%$ $P_s=90$ $P_e=270$ then dwell center
 $P_w = (270-90)*0.3+90 = 144$ deg.



CAM Dwell (T_w) - Some CAMs use this parameter to determine the amount of dwell within their cycle. A dwell is a period of time during which the CAM is stopped. T_w is limited by the following relationship:

$$\text{Max}(T_w) = 2.0 * \text{Min} [0.40 * (P_e - P_w) / V_m, 0.40 * (P_e - P_w) / V_m]$$

where P_w (deg) is the dwell center angle and V_m (deg/sec) is master speed.

Enable Registration Correction

This enables the auto-registration function for the indexed CAM.

Registration Correction Setup...

Registration

This icon is used to initialize an auto registration procedure. Registration is the process of referencing a product edge or register mark. This allows positioning errors to be detected and corrected before an upstream (web, product) or downstream (die-cutter, print cylinder, etc.) process takes place, depending on the machine design.

This function tightly couples the control system with the **Probe Event** on the drive that the product is being referenced to. Registration is accomplished by comparing the captured position to a target value and correcting for the difference. The registration error is automatically assimilated into the axis motion profile, providing a smooth, seamless correction. The registration error can be corrected using S-curve, Triangular, and Trapezoidal correction profiles.

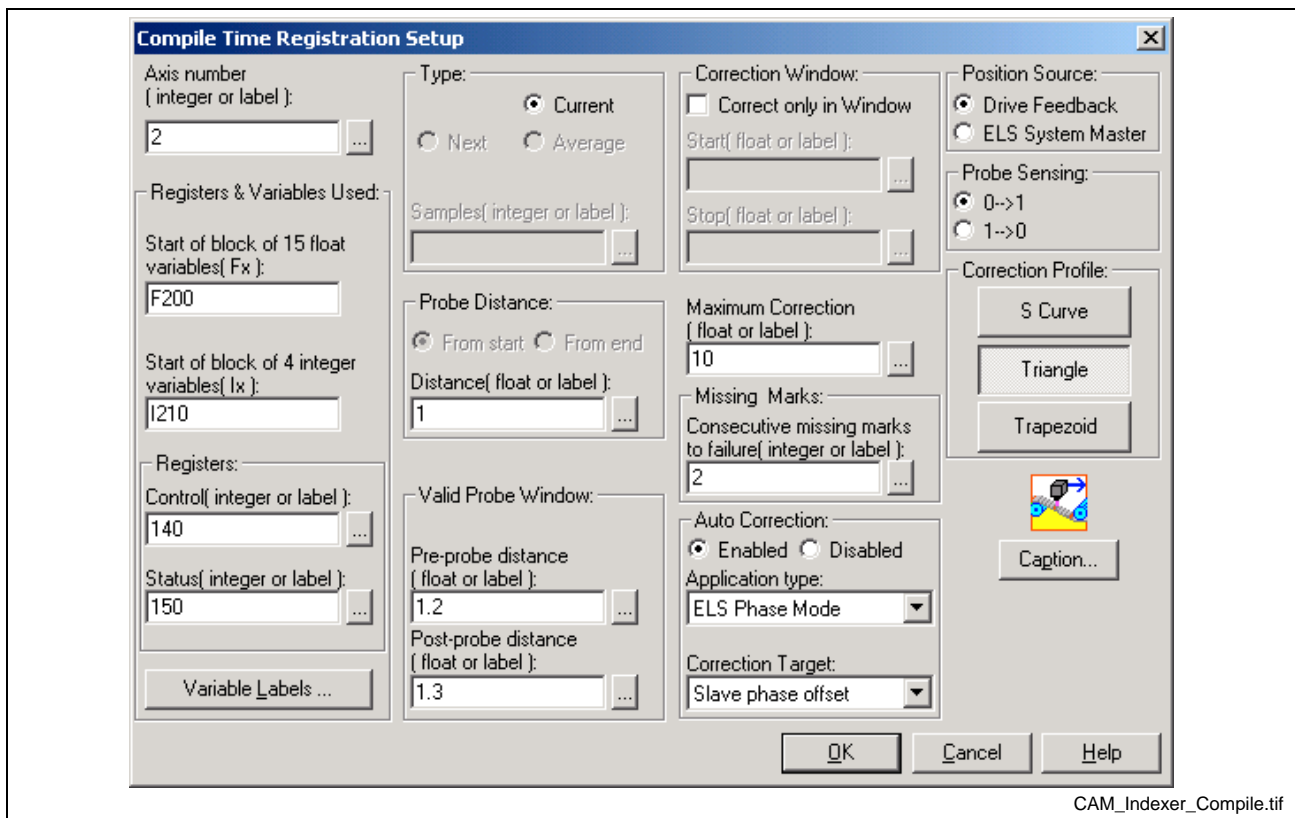


Fig. 3-35: CAM Indexer Compile Time Registration

Functional Description

The Registration function integrates several tasks:

It automatically executes a high priority process that is triggered by an axis probe event.

This process generates a *registration error* based on the settings of the drive, system and registration parameters.

The control will then automatically reconcile details such as rollover, probe On/Off position window, setting drive parameters (edge, arming, etc...), and routing the correction data to the appropriate axes.

The registration function is compatible with axes defined in ELS Phase, Drive CAM, and control CAM (table or indexer) modes. The values needed for registration are read from and written to the control with user program variables. Up to four axes can use the registration function.

The Registration icon window assigns variables and I-O registers, and initializes axis parameters. All values are assigned to blocks of float and integer variables. This allows them to be read or written from a user interface or a PLC with no additional program code.

When the CAM Indexer is used, it performs the registration correction with a profile based on the *Correction Distance*. For other types of axes, a relative phase offset (either master or slave) is sent to the drive, and the drive performs the correction. The phase offset, equal to *Correction Distance*, is applied according to the *Correction Procedure* and *Correction Maximum* parameters.

The registration function can be used with the following Bosch Rexroth drive firmware versions:

DIAX04 - ELS-04VRS or later

ECODRIVE03- SGP-01VRS or later

Registration Parameters

The following Parameters are used by the registration function. Most of them are written by the user with the registration icon window. Some of them are also calculated and written by the control or the Drive.

Parameters	Description	Variable	Written by
Type	Current, Next, Average	Int 0 bits 0-1	user
Probe Distance from Start/End	Select Setpoint to be calculated from start/end of move	Int 0 bit 2	user
Probe Distance	Expected position of mark from start or end of move	Float 0	user
Mark Setpoint	Absolute expected position of mark to calculate error	Float 6	control
Correction Distance	Correction distance (Error amount)	Float 7	control
Probe Value	Feedback position at probe 1 trigger	Float 8	drive
Valid Probe Window Start	Distance before Mark Setpoint to look for mark	Float 1	user
Valid Probe Window End	Distance after Mark Setpoint to stop looking	Float 2	user
Correction Window	Correct only in the correction window.	Int. 0 bit 12	user
Correction Window Start	Correction to start at this point	Float 3	user
Correction Window Stop	Correction to be completed before this point	Float 4	user
Maximum Correction	Maximum correction per period	Float 5	user
Missing Marks	Number of missed marks allowed	Int. 1	user
Auto Correction	Enable correction sent to phase adjust, etc.	Int. 0 bit 8	user
Correction Target	Select correction target: master offset, slave offset, etc.	Int. 0 bit 9-10	user
Position Source	Measure Master or Slave position	Int. 0 bit 4	user
Probe Sensing	Use Leading/Trailing edge of mark	Int. 0 bit 3	user
Correction Profile	Shape of the correction CAM	Int. 2	user

Type

This selects the desired type of registration correction calculation to be used.

- Current:
- Next:
- Average:

Samples (Integer or Label) -

Probe Distance from Start/End

This bit determines if the *Mark Distance* is relative to the Start or End of move. This option applies to CAM axes that use the indexer feature. For other modes, the mark distance is an absolute position.

0 - End of Move

1 - Start of Move

Probe Distance

This value is used to determine the calculated position of the axes that a mark is expected to be detected.

CAM Indexer:

Based on the setting of *Mark at S/E* bit, the *Mark Setpoint* value is automatically calculated as follows. *Starting Position* is the position of the axis at the start of the index. *Ending Position* is the expected position at the end of the index (minus registration correction).

If Start of move: $Mark\ Setpoint = Starting\ Position + Mark\ Distance$

If End of move: $Mark\ Setpoint = Ending\ Position - Mark\ Distance$

ELS Phase, Control Table CAM, or Drive CAM Mode:

The *Probe Setpoint* is an absolute position on either the master or the slave (set via an integer option bit).

$Probe\ Setpoint = Probe\ Distance$

Mark Setpoint: This value is used to calculate registration error and various position points used in the registration process. See Correction Distance, Probe Window On/Off.

Correction Distance: This is the amount of correction distance that will be applied to the current index in units. Based on the selected *Correction Profile* type this error is added to the CAM profile or ELS slave position at the position defined by the correction window. Only one value is read during each index cycle.

$Correction\ Distance = Mark\ Setpoint - Probe\ Value$

In future versions, correction distance may be optionally buffered through several cycles in case the measurement is upstream, or may be averaged or filtered for a smoother correction.

Probe Value: This value indicates the measured probe position value that was captured by the drive when the registration mark was detected on the probe 1 input. It is saved only when the axis is within the probe window. It is used in the calculations associated with determining registration error.

Valid Probe Window Start/End

This window selects the period within the move cycle that the registration mark can be read. The probe position is captured and the correction distance is calculated only when the mark is found within this position range. The probe window is related to the selected measurement value (master or axis position).

Correction Window

This bit enables the period within the move cycle that the correction can be applied (defined by "Correction Window Start/End"). If this bit is clear, the correction is applied immediately after the probe position is captured.

Correction Window Start/Stop

This window selects the period in the move cycle to apply the registration correction. If the option *Correction Window Enabled* is not selected, the correction is applied immediately after the probe position value is captured. Otherwise, the correction starts at *Correction Window Start* and finishes before *Correction Window End*.

When using the CAM indexer, the start and end positions in relation to the master axis are entered in these parameters. Correction motion will only take place during this window. If the probe comes in during the correction window, for example at position *x* of the master, the maximum allowable correction during that cycle is calculated as follows:

$(maximum\ correction)(end - x)/(end - start)$

For other motion types, the positions are related to the measured value (master or axis position). The correction is added to the value selected in the *Correction Target* option. The drive handles the phase offset internally, based on its velocity, acceleration, and/or filter parameters. It is

necessary to set these drive parameters so that the correction move finishes before the next cycle.

Maximum Correction

This is the maximum amount of correction distance that will be applied to the current index in units. If zero, the full amount of the correction is used for the current index. If a value is entered, the *Correction Distance* value is compared to *Correction Maximum* and correction up to this value will be applied. The remainder will be discarded.

Missing Marks

This defines the number of consecutive missed marks that are allowed before the control sets the Max Missed Marks status bit.

Auto Correction

By default, the control sends the *Correction Distance* to the drive to perform a phase offset, based on the correction window and correction target. It is possible to disable automatic correction, so that it can be handled in the user program. This option applies to all motion types except for the CAM indexer, for which automatic correction is always enabled.

0 - Enable Automatic Correction

1 - Disable Automatic Correction

Correction Target

These bits select the target parameter for the *Correction Distance*. When using the CAM indexer, the correction uses a CAM which is added to the CAM position generated by the indexer. For other modes, the following selections determine the affected parameter:

Bit 10	Bit 9	ELS Phase Sync	ELS Velocity Sync	Drive CAM	Control CAM (Table)
0	0	slave phase offset (S-0-0048)	additive velocity command (S-0-0037)	master phase offset (P-0-0061)	master phase offset (A-0-157)
0	1	invalid	ratio fine adjust (P-0-0083)	slave phase offset (S-0-0048)	slave phase offset (A-0-162)
1	0	invalid	invalid	CAM shaft distance (S-0-0093)	CAM H factor (A-0-33)

Position Source

This bit determines if the slave axis (0) or ELS master (1) position is used in the registration correction calculation. This option applies to ELS, control CAM (no indexer), and drive CAM modes only. With the CAM indexer, only the slave position can be measured. If slave position is enabled, the *Probe Value* captured by the drive is the axis feedback position. If master position is enabled, the *Probe Value* is the ELS master position.

0 - Axis Feedback Position

1 - ELS Master Position

Probe Sensing

This bit determines if the drive captures the axis position at the leading or trailing edge of the mark. The leading edge is indicated by a high (0-1 transition) on the probe input.

0 - Leading Edge

1 - Trailing Edge

Correction Profile

When using the CAM indexer, this value defines the shape of the correction CAM. Current selections are: S-curve, Triangular, and Trapezoidal.

Registration Control and Status Registers

The control and status registers that are used for each registration axis are allocated in the registration user program command.

Registration Control Register

9 - Enable Preset

10 - Enable Registration

Registration Status Register

9 - Correction at Max

10 - reserved

11- Mark in Window

12- Preset Enabled

13 - Registration Enabled

14 - Max Missed Marks

Enable Preset

The function of this bit depends on the selected correction procedure. If buffering or averaging is used (future versions), the FIFO buffer and/or the average is reset. Otherwise, this bit is ignored by the control.

Enable Registration

Set this bit to (1) to enable the registration function. All relevant parameters will be initialized, then the 'Registration Enabled' bit in the registration status register will be set. To disable the registration function, set this bit to (0).

Correction at Max

This bit is set to (1) when the *Correction Distance* exceeds the *Correction Max* parameter. It is reset to 0 after the next mark is found and *Correction Distance* is less than or equal to *Correction Max*.

Mark in Window

This bit is set to (1) as soon as a registration mark has been detected within the probe window. It is cleared when registration is first enabled, or on the following cycle when no registration mark is found in the window. This bit can be mapped to a task input or system input event, so that the user can run additional program code when the mark is detected.

Preset Enabled

This bit is an acknowledgment of the Enable Preset control bit.

Registration Enabled

This bit is an acknowledgment of the Enable Registration control bit. Its value will not match that of the control bit until the control has completed the process of enabling or disabling registration. This process could take several milliseconds, depending on the parameters that need to be initialized.

Max Missed Marks

A missed mark counter is incremented when the probe window is traversed without a mark being found. This bit is set when the counter exceeds the *Missed Marks* parameter. The user program or PLC can then take appropriate action or alert the user of this condition. This bit is reset by the control only upon a 1-0 transition of the Registration Enable bit.

Circle



The Circle icon is used for multi-axis coordinated circular interpolation, moving in a circular arc from one point in 3D space to a second point in 3D space. The plane of the arc is two-dimensional and the plane may have any orientation in three-dimensional space. Placing a Circle icon on a Task or Subroutine workspace opens a *Coordinated Circle Setup* window. The Go Icon is not required with Circle function. When executed, the motion will immediately be sent to the path planner. Stepping to the next icon in the program will take place immediately.

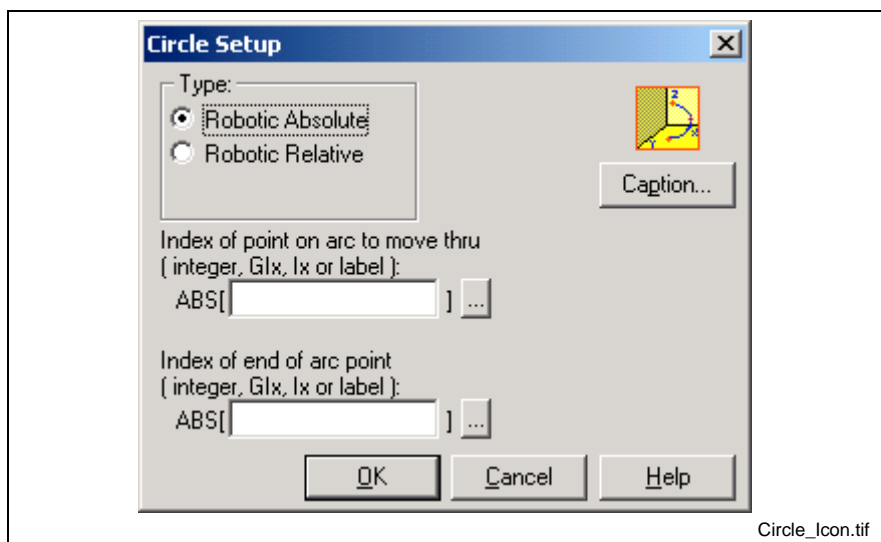


Fig. 3-36: Circle Setup

Robotic Move

Move in a circular arc from the current point in space past the “thru” point to the end point. The plane of the arc is two-dimensional and may have any orientation in three-dimensional space.

A circular move may be Absolute or Relative and is defined by three points on the circle. These points must have been previously defined in the absolute and/or relative point tables. Points are identified by a point table ID number or equivalent label. Indirect point table entries may be used by specifying an integer variable, global or program (Glx, Ix) or equivalent label.

An absolute circular move begins from the endpoint of the previous segment (or current position if the system is halted), moves through an absolute point, and terminates at another absolute point.

A relative circular move is similar; however, the intermediate and endpoints are defined as relative offsets from the starting point. The relative circular move begins at the endpoint of the previous segment (or current position if the system is halted), moves through the first relative offset, and terminates at the second relative offset.

Event Types for Coordinated Motion

Event triggers for coordinated motion are associated with the points table that defines the path. Up to 4 event triggers can be configured for each point in a coordinated motion path. The following table lists the event types available for the Circle icon along with their arming behavior, maximum number and description.

Event Type	Auto Rearming	Maximum Number	Description
Percent from Start	No	4 events per point	A position-based event that executes an event function at the set percentage of the overall travel from the start.
Percent before End	No	4 events per point	A position-based event that executes an event function at the set percentage of the overall travel before the end.
Time from Start	No	4 events per point	A time-based event that executes an event function at a set time from the start of the move.
Time before End	No	4 events per point	A time-based event that executes an event function at a set time before the end of the move.

Table 3-9: Coordinated Motion Event Types

The following figure illustrates that different windows that are displayed when adding an event trigger to the coordinated motion point. Refer to *Adding Events* on page 3-82 for an example.

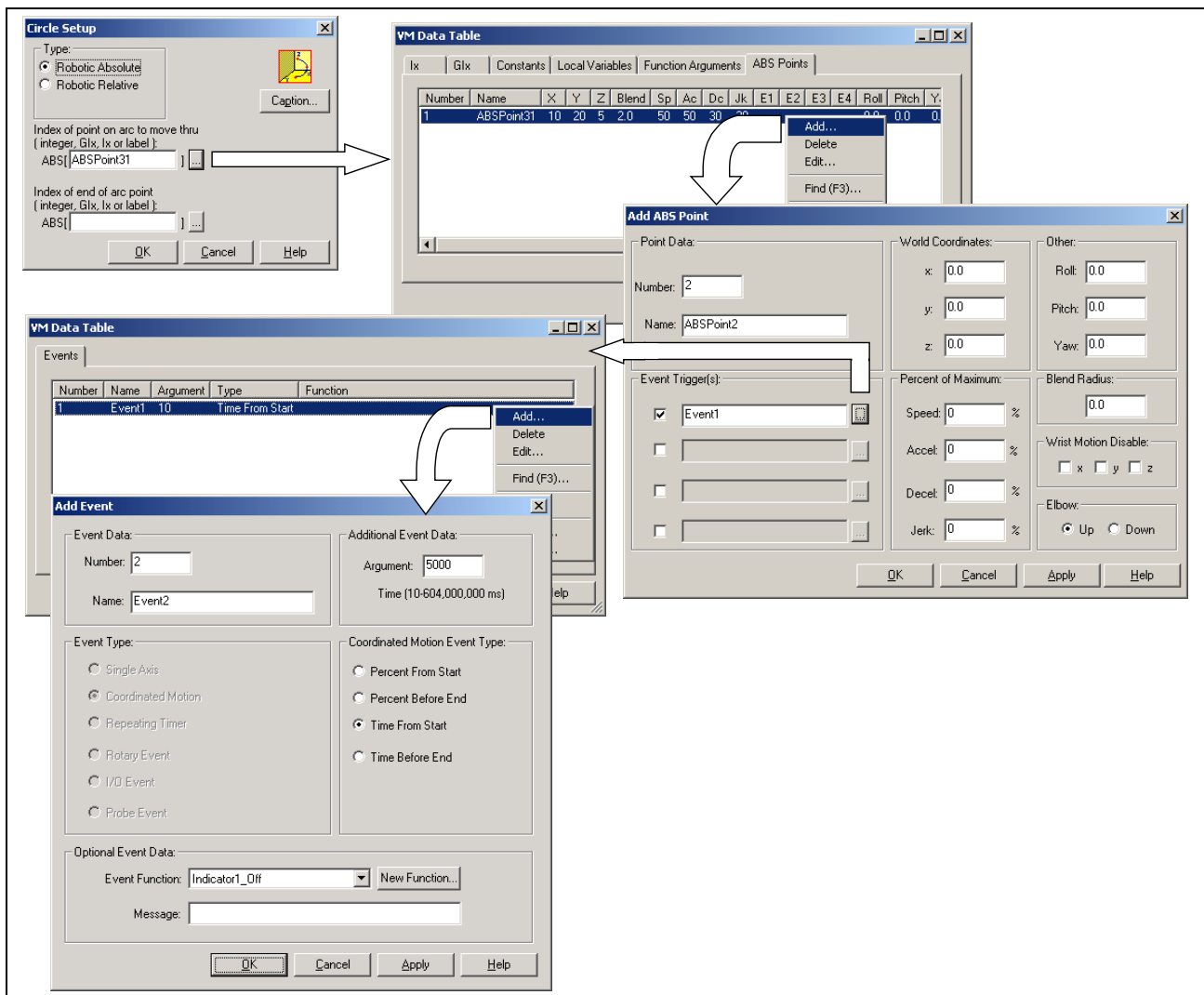


Fig. 3-37: Circle Setup

Changing Arguments during Runtime

Any argument, event function or both can be modified during runtime by placing a Calc icon in the desired program flow location. Refer to the Calc2 icon, Events section on page 3-27 for the correct syntax when defining an argument for a coordinated motion point.

Note: The Event2 icon cannot be used to change the argument or event function of a coordinated motion point.

Command



The Command icon is used to initiate control or drive-resident commands. Select the **Operation** to be performed and identify the source of the command as either Control or Drive. The **Drive** field can be identified by using the drive's SERCOS address, integer variable or label. To identify the **Parameter ID Number**, enter the simple form of the control or drive parameter (i.e., C-0-0082: enter 82).

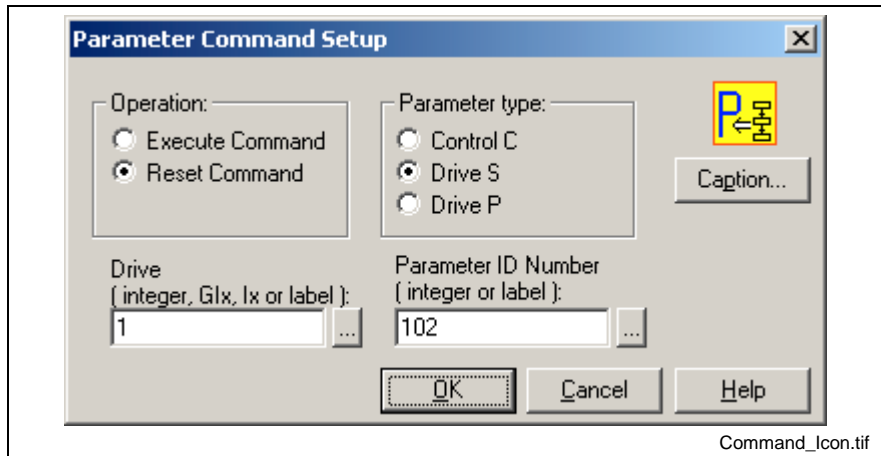


Fig. 3-38: Command Setup

The following table lists available control commands that can be executed using the Command icon.

Control Parameter	Description
C-0-0082	Save Global Variables Command
C-0-2903	PLS1 Build Command
C-0-2905	PLS1 Activate Command

Table 3-10: Control-resident Commands

Note: Drive-related commands vary based on drive type (DIAX04 and ECODRIVE03) and firmware. Refer to the relevant drive manuals for available commands.

CoordArt (Coordinated Articulation)



Coordinated Articulation is an advanced feature in VisualMotion 9 used to move up to six axes in coordinated fashion based on world coordinate inputs from an ELS Group output or manual positions. This feature provides the ability to link cyclic coordinated motion to an ELS master.

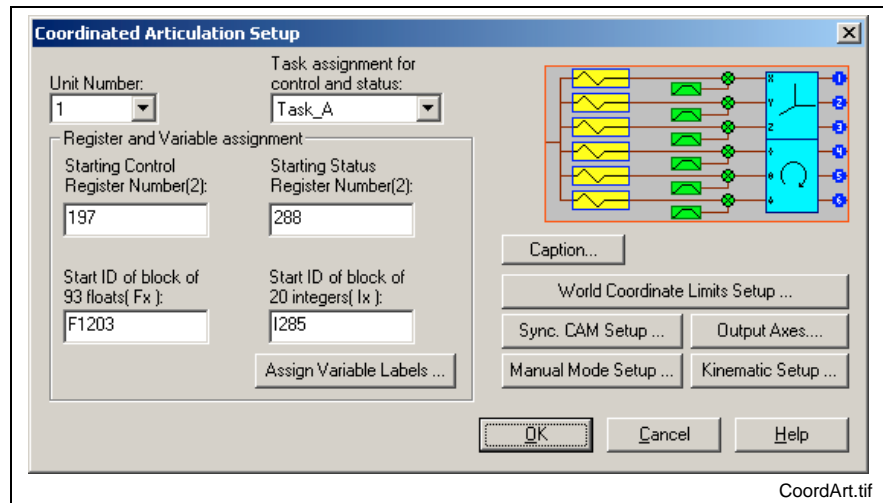


Fig. 3-39: Coordinated Articulation

Unit Number

This number identifies the Coordinated Articulation configuration in a VisualMotion 9 project. Up to 2 Coordinated Articulated configurations can exist in a project at one time.

Task Assignment

Since the CoordArt icon must be placed in the Initialization task, the actual task that will control and monitor the function must be specified from the **Task assignment for control and status** drop-down list.

Register and Variable Assignment

The **Register and Variable Assignment** section is used to define the registers and program variable blocks that will be used for the selected Unit number.

Note: Default values are automatically assigned for registers and variable blocks. It is strongly recommended that the programmer use the default values for registers and variables. This makes documentation and modifications to user program an easier task over the scope of the project.

Starting Control Register Number

A Coordinated Articulation configuration uses two control registers. One control register is used for synchronized mode and one for local mode. When a starting control register number is entered, the next consecutive number is used for the local mode control register. The following tables lists the functions used in each control register.

Synchronized Mode Control Register The first control register selects the operation mode (sync or local) for each axis.

Register Bit	Label	Description
1	CA#_X_SYNC	<i>Synchronized Mode Bits</i> 0 -> 1 transition enables each axis to Sync mode. This transition is delayed until the ramp generator's velocity is 0. 1 -> 0 transition set each axis to Local Mode. Local mode is used for the manual positioning of each axis.
2	CA#_Y_SYNC	
3	CA#_Z_SYNC	
4	CA#_ROLL_SYNC	
5	CA#_PITCH_SYNC	
6	CA#_YAW_SYNC	
7	CA#_X_REL_MOD	<i>Positioning Mode Bits</i> Enable before setting Synchronized Mode Bit. 1 = Relative Mode 0 = Absolute Mode (default)
8	CA#_Y_REL_MOD	
9	CA#_Z_REL_MOD	
10	CA#_ROLL_REL_MOD	
11	CA#_PITCH_REL_MOD	
12	CA#_YAW_REL_MOD	

Table 3-11: Synchronized Mode Control Register

Local Mode Control Register The second control register enables the ramp generator in local mode for each axis. Local mode allows the user to manually position the world coordinate in-position value to the current Cam output position before synchronizing.

Register Bit	Label	Description
1	CA#_X_MAN_EN	<i>Local Mode (Manual) Bits [used when Sync mode is inactive]</i> Normal Local Mode: 0 -> 1 transition enables the ramp generator, creates a move profile for variable CA#_%_TAR_POS value to variable CA#_%_IN_POS. (where % = X, Y, Z, ROLL, PITCH or YAW) 1 -> 0 transition disable the ramp generator. If move is at velocity, the velocity is ramped down using the value in variable CA#_LIN_DECEL.
2	CA#_Y_MAN_EN	
3	CA#_Z_MAN_EN	
4	CA#_ROLL_MAN_EN	
5	CA#_PITCH_MAN_EN	
6	CA#_YAW_MAN_EN	
7	CA#_X_MAN_IMD	<i>Immediate Mode Bits [used when Sync mode is inactive]</i> Enable before setting Local Mode Bit. 1 = Immediate Mode 0 = Normal Local Mode
8	CA#_Y_MAN_IMD	
9	CA#_Z_MAN_IMD	
10	CA#_ROLL_MAN_IMD	
11	CA#_PITCH_MAN_IMD	
12	CA#_YAW_MAN_IMD	

Table 3-12: Synchronized Mode Control Register

Starting Status Register Number

A Coordinated Articulation configuration uses two status registers. One status register is used for synchronized mode and one for local mode. When a starting status register number is entered, the next consecutive number is used for the local mode status register. The following tables lists the functions used in each status registers.

Synchronized Mode Status Register

The first register is used to monitor the status of each axis enabled to synchronized mode. It also monitors the equivalence between variables CA#_%_IN_POS and CA#_%_TAR_POS (where % = X, Y, Z, ROLL, PITCH or YAW).

Register Bit	Label	Description
1	CA#_X_SYNCED	1 = In Sync Mode. Using CAM output world coordinated input value. 0 = In Local Mode. Use target position for world coordinated input value.
2	CA#_Y_SYNCED	
3	CA#_Z_SYNCED	
4	CA#_ROLL_SYNCED	
5	CA#_PITCH_SYNCED	
6	CA#_YAW_SYNCED	
7	CA#_X_READY	1 = CAM output value (CA#_%_CAM_OUT) equals world coordinated input value (CA#_%_IN_POS) and ramp generator is at 0 velocity. 0 = Not ready for synchronization
8	CA#_Y_READY	
9	CA#_Z_READY	
10	CA#_ROLL_READY	
11	CA#_PITCH_READY	
12	CA#_YAW_READY	

Table 3-13: Synchronized Mode Status Register

Local Mode Status Register

The second register is used to monitor the status of CA#_%_IN_POS equal to CA#_%_TAR_POS.

Register Bit	Bit Label	Description
1	CA#_X_AT_TAR_POS	Only valid in Local Mode. (CA#_%_SYNC = 0) 1 = CA#_X_IN_POS at CA#_X_TAR_POS position and ramp generator at 0 velocity. (where % = X, Y, Z, ROLL, PITCH or YAW) 0 = not, goes to 0 when new target position entered. NOTE: This bit is not set to 1 if the ramp generator move is interrupted (dropped Enable, user task stopped, etc.) before the target position is reached. In such an instance, it is not possible to 'force' the bit to 1 by setting the ramp generator's target value equal to the current input value. Rather, another ramp generator move must be completed before this bit is set to 1.
2	CA#_Y_AT_TAR_POS	
3	CA#_Z_AT_TAR_POS	
4	CA#_ROLL_AT_TAR_POS	
5	CA#_PITCH_AT_TAR_POS	
6	CA#_YAW_AT_TAR_POS	

Table 3-14: Synchronized Mode Status Register

Coordinated Articulation Program Variables

The following program variables (20 integer variables, 93 float variables) are used for Coordination Articulated. In the following tables, “#” denotes the unit number.

Integer Variables The following 20 integers are available for Coordination Articulated. The integer label column displays the default label issued by VisualMotion.

Number Offset	Default Value	Access	Integer Label	Description
0	0	read-only	CA#_NUMBER	Unit number of coordinated articulation function 1-2 Note: This feature supports 2 robots however they must be defined in different user tasks.
1	0	read/write in phase 2	CA#_OUT_AXIS1	Number of axis used for output 1
2			CA#_OUT_AXIS2	Number of axis used for output 2
3		read-only in phase 4	CA#_OUT_AXIS3	Number of axis used for output 3
4			CA#_OUT_AXIS4	Number of axis used for output 4
5			CA#_OUT_AXIS5	Number of axis used for output 5
6			CA#_OUT_AXIS6	Number of axis used for output 6
7	0	read/write in any phase	CA#_X_CAM_NUM	Input CAM number for % world coordinated position
8			CA#_Y_CAM_NUM	
9			CA#_Z_CAM_NUM	
10			CA#_ROLL_CAM_NUM	
11			CA#_PITCH_CAM_NUM	
12			CA#_YAW_CAM_NUM	
13	0	read/write in phase 2 read-only in phase 4	CA#_TASK_ID	Task function is associated with
14	0	read/write in any phase	CA#_Reserved14	
15	0	Read only	CA#_KINEMATIC	Number of kinematic equation used in transformation
16	1	read/write in phase 2 read-only in phase 4	CA#_ELS_GROUP	ELS Group number used as CAM input
17	0	read/write in any phase	CA#_RESERVED17	
18	0	read/write in any phase	CA#_RESERVED18	
19	0	read/write in any phase	CA#_RESERVED19	

Table 3-15: Program Integers

Float Variables The following 93 floats are available for Coordination Articulated. The float label column displays the default label issued by VisualMotion.

Number Offset	Default Value	Access	Float Label	Description
0	0.0	read-only	CA#_X_IN_POS	X world coordinate input value in linear units, initialized to 0 at program activation.
1			CA#_Y_IN_POS	Y world coordinate input value in linear units, initialized to 0 at program activation.
2			CA#_Z_IN_POS	Z world coordinate input value in linear units, initialized to 0 at program activation.
3			CA#_ROLL_IN_POS	Roll world coordinate input value in rotary units, initialized to 0 at program activation.
4			CA#_PITCH_IN_POS	Pitch world coordinate input value in rotary units, initialized to 0 at program activation.
5			CA#_YAW_IN_POS	Yaw world coordinate input value in rotary units, initialized to 0 at program activation.
6	0.0	read/write in any phase	CA#_X_TAR_POS	Local mode, X target position for "single axis" move in world coordinates, linear units
7			CA#_Y_TAR_POS	Local mode, Y target position for "single axis" move in world coordinates, linear units
8			CA#_Z_TAR_POS	Local mode, Z target position for "single axis" move in world coordinates, linear units
9			CA#_ROLL_TAR_POS	Local mode, Roll target position for "single axis" move in world coordinates, rotary units
10			CA#_PITCH_TAR_POS	Local mode, Pitch target position for "single axis" move in world coordinates, rotary units
11			CA#_YAW_TAR_POS	Local mode, Yaw target position for "single axis" move in world coordinates, rotary units
12	100.0	read/write in phase 2	CA#_LIN_ACCEL	Local mode, linear acceleration used for "single axis" move(units/sec^2)
13	100.0	read-only in phase 4	CA#_LIN_DECEL	Local mode, linear deceleration used for "single axis" move(units/sec^2)
14	50.0	read/write in phase 4	CA#_LIN_VELOCITY	Local mode, linear velocity used for "single axis" move(units/min)
15	100.0	read/write in phase 4	CA#_ROT_ACCEL	Local mode, rotary acceleration used for "single axis" move(units/Sec^2)
16	100.0	read/write in phase 2	CA#_ROT_DECEL	Local mode, rotary deceleration used for "single axis" move(unitss/Sec^2)
17	50.0	read-only in phase 4	CA#_ROT_VELOCITY	Local mode, rotary velocity used for "single axis" move(RPM)
18	0.0	read-only	CA#_X_CAM_OUT	(CAM[#]*H) + Offset), for % world coordinate
19			CA#_Y_CAM_OUT	
20			CA#_Z_CAM_OUT	
21			CA#_ROLL_CAM_OUT	
22			CA#_PITCH_CAM_OUT	
23			CA#_YAW_CAM_OUT	

Table 3-16: Program Floats (1 of 2)

Number Offset	Default Value	Access	Float Label	Description
24	1.0	read/write in any phase	CA#_X_CAM_H	H CAM factor for % world coordinate
25			CA#_Y_CAM_H	
26			CA#_Z_CAM_H	
27			CA#_ROLL_CAM_H	
28			CA#_PITCH_CAM_H	
29			CA#_YAW_CAM_H	
30	0.0	read/write in any phase	CA#_X_CAM_OFF	Offset for % world coordinate
31			CA#_Y_CAM_OFF	
32			CA#_Z_CAM_OFF	
33			CA#_ROLL_CAM_OFF	
34			CA#_PITCH_CAM_OFF	
35			CA#_YAW_CAM_OFF	
36	-100.0	read/write in phase 2	CA#_X_MIN	Minimum value for % world coordinate
37			CA#_Y_MIN	
38			CA#_Z_MIN	
39	-180.0	read-only in phase 4	CA#_ROLL_MIN	
40			CA#_PITCH_MIN	
41			CA#_YAW_MIN	
42	100.0	read/write in phase 2	CA#_X_MAX	Maximum value for % world coordinate
43			CA#_Y_MAX	
44			CA#_Z_MAX	
45	180.0	read-only in phase 4	CA#_ROLL_MAX	
46			CA#_PITCH_MAX	
47			CA#_YAW_MAX	
48 ... 86	0.0	read/write in phase 2	CA#_KINEMATIC_V1 through CA#_KINEMATIC_V39	Kinematic setup data value 1 through Kinematic setup data value 39
		read-only in phase 4		
87	0.0	read-only	CA#_X_FDBK_POS	X world coordinate forward transform value in linear units, updated every SERCOS cycle in Phase 4.
88			CA#_Y_FDBK_POS	Y world coordinate forward transform value in linear units, updated every SERCOS cycle in Phase 4.
89			CA#_Z_FDBK_POS	Z world coordinate forward transform value in linear units, updated every SERCOS cycle in Phase 4.
90			CA#_ROLL_FDBK_POS	Roll world coordinate forward transform value in rotary units, updated every SERCOS cycle in Phase 4.
91			CA#_PITCH_FDBK_POS	Pitch world coordinate forward transform value in rotary units, updated every SERCOS cycle in Phase 4.
92			CA#_YAW_FDBK_POS	Yaw world coordinate input value in rotary units, initialized to 0 at program activation.

Table 3-17: Program Floats (2 of 2)

Assign Variable Labels

Default variable labels and comments can be added individually for Variable, Registers and Bits by selecting the appropriate **Data Type** radio button and clicking the **Add Default Labels** button. Clicking the **Add All Default Labels** button will add the default variable labels and comments to all Data Types at one time.

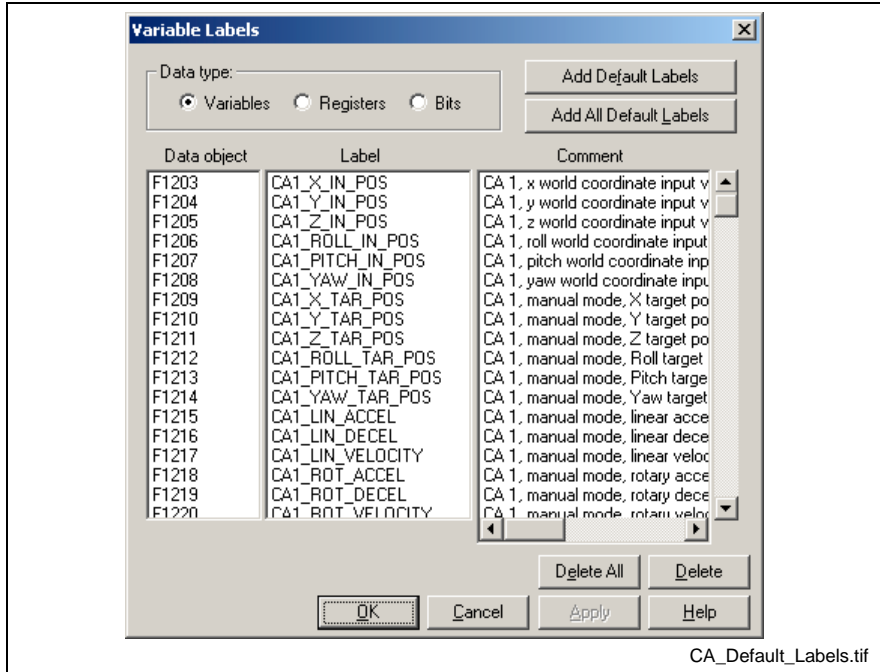


Fig. 3-40: Coordinated Articulation Default Variable Labels

Caption

Selecting the **Caption...** button allows to user to enter a description, up to a maximum of 79 characters, that will appear as a tool tip when holding the cursor over the CoordArt icon. The **Default comment** button places default descriptions (shown below) in both fields.

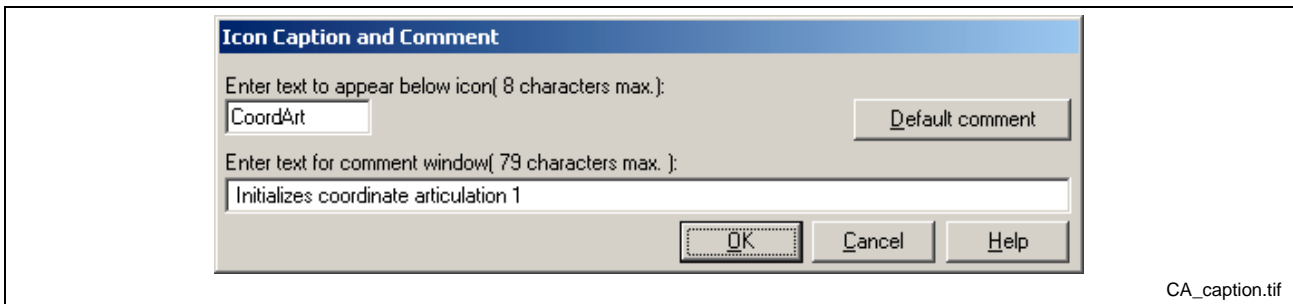


Fig. 3-41: Coordinated Articulation Caption

World Coordinated Limits Setup...

The *World Coordinated Limits* window is used to set the minimum and maximum boundaries that the system will use for the X, Y, Z, Roll, Pitch and Yaw input values.

Note: It is strongly recommended that the world coordinated limits be setup to ensure protection of all hardware.

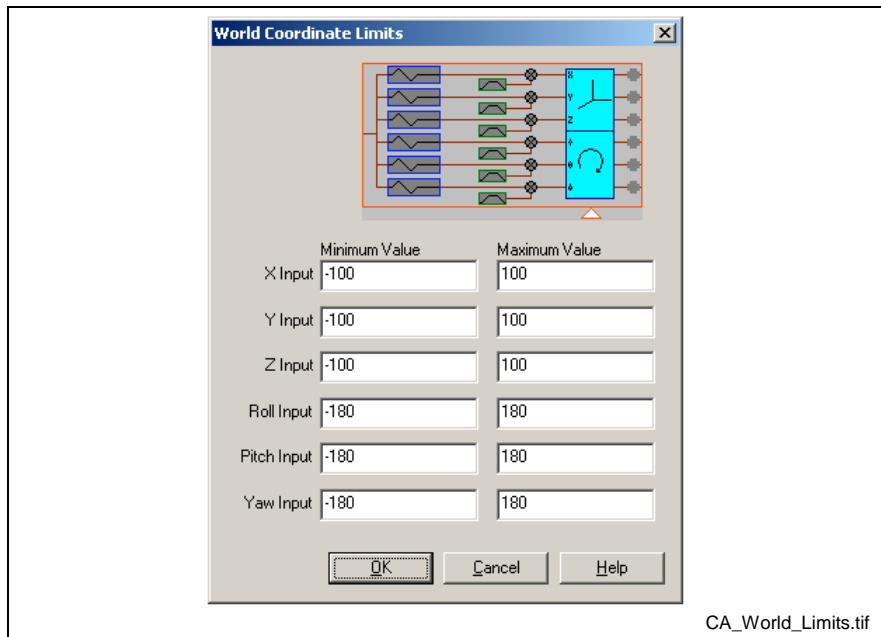


Fig. 3-42: World Coordinated limits Setup Window

Sync. CAM Setup

The *Sync. CAM Setup* window is used to configure the H Factor and Offset for the selected ELS Group number.

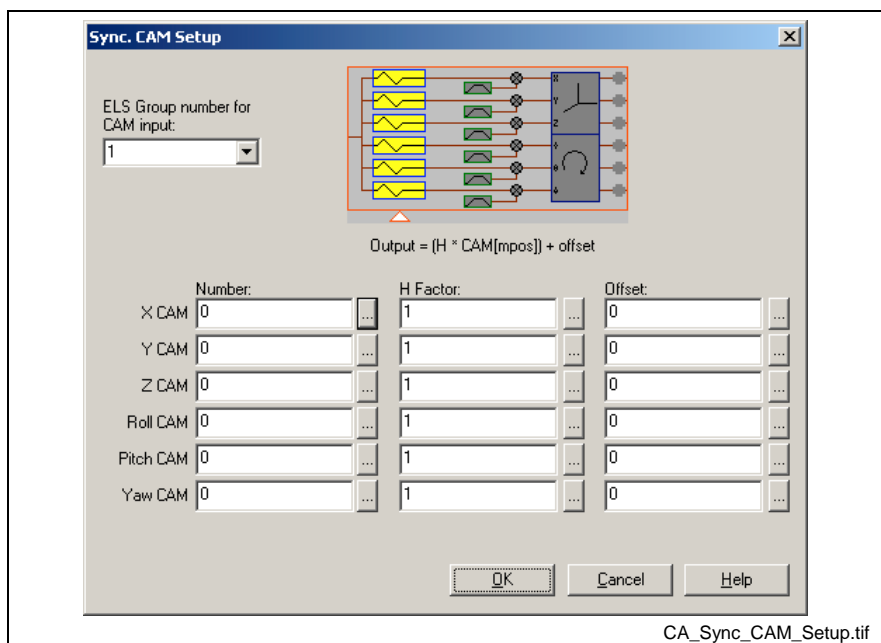


Fig. 3-43: Sync. CAM Setup Window

Manual Mode Setup

The *Manual Mode Setup* window is used to set the acceleration, deceleration and velocity that will be used when moving Coordinated Articulated axes in local mode.

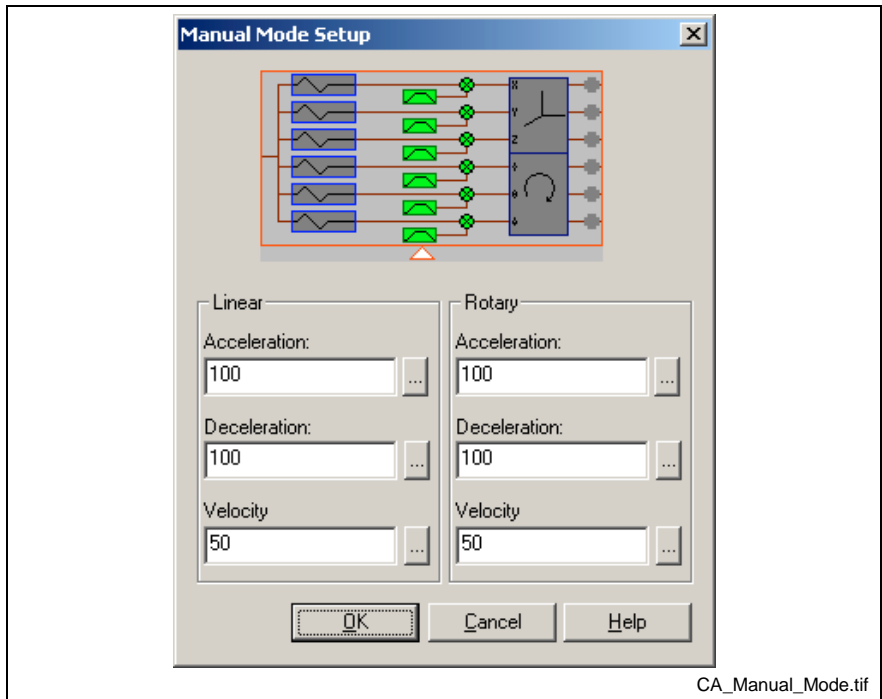


Fig. 3-44: Manual Mode Setup Window

Output Axes

The *Output Axes* window is used to select the axes that will be used with Coordinated Articulation. Up to 6 axes can be used. The user can enter a axis number or click the browse button and choose a variable label that represents the axis. Used axes should be left with a zero value.

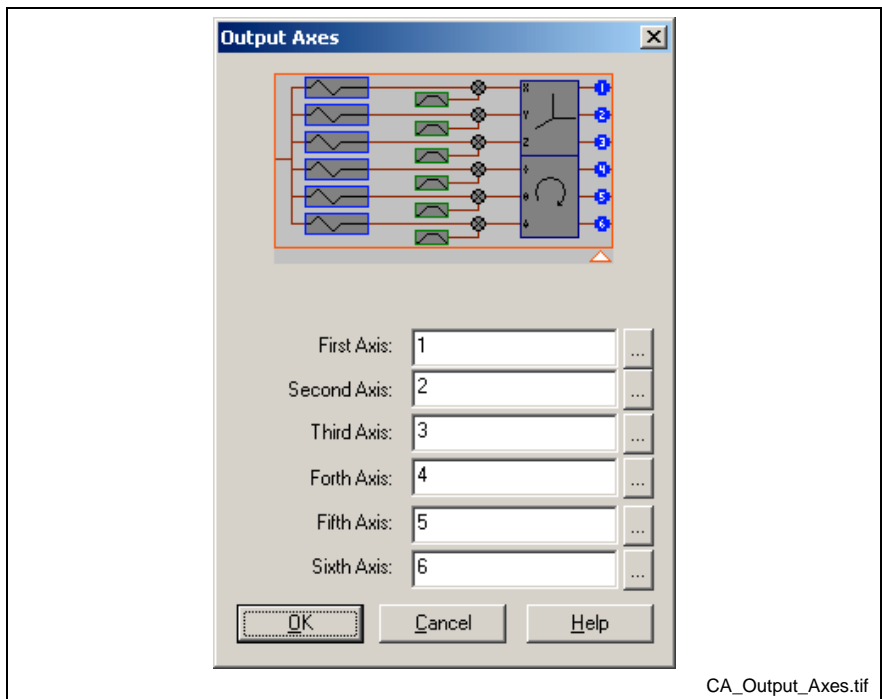


Fig. 3-45: Output Axes Window

Kinematic Setup

The *Kinematic Setup* window is used to enter the predefined values of the kinematic equation that will be used to determine the positioning of up to six coordinated axes.

Note: Kinematic numbers and data are unique to each application.

Kinematic 13 is a pass-through kinematic. Positional values are passed directly to each axis without any manipulation by a kinematic. No **Kinematic Data** values are required for this kinematic.

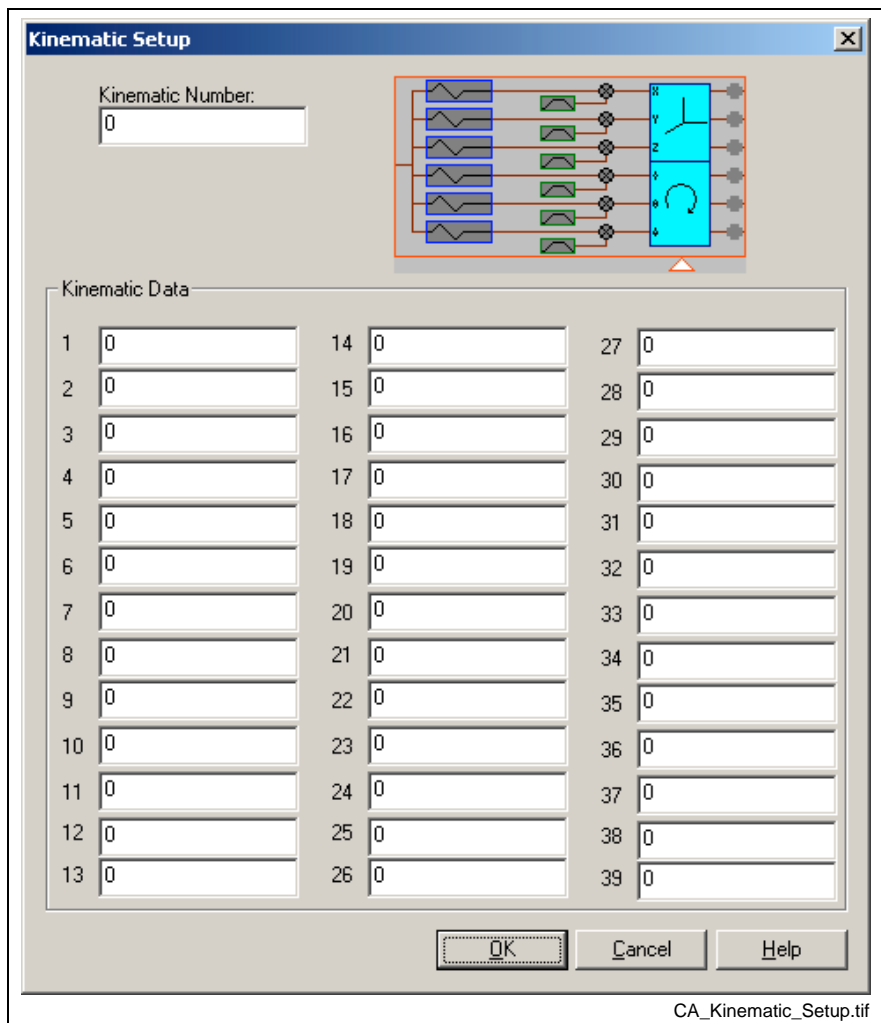


Fig. 3-46: Kinematic Setup Window

ELSAAdj1



The ELSAAdj1 icon is used to adjust the velocity or phase of an ELS configured axis to compensate for mechanical variations between master and slave axes. The axis may be specified by an integer constant, variable, global variable or an equivalent label. Clicking the browse button to the right of any field opens the VM Data Table.

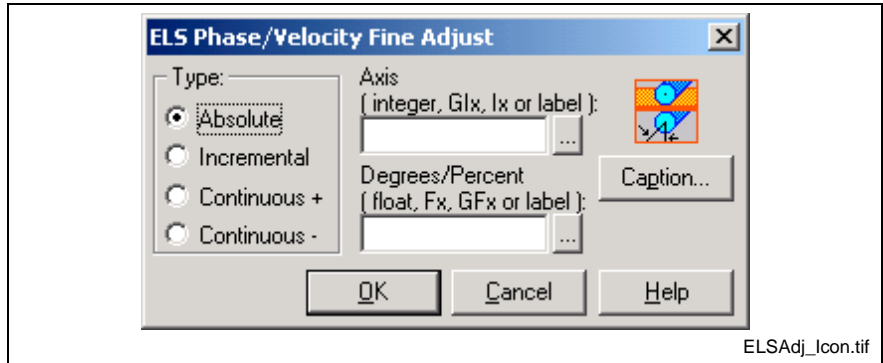


Fig. 3-47: ELSAdj Setup

Type

This section selects the method of adjustment:

- **absolute** - the **degrees or percent** edit field is the new offset.
- **incremental** - the **degrees or percent** edit field is added to the current offset. If in phase mode and sum exceeds 360, it rolls over. If in velocity mode, the sum is limited to -100 or +300 percent.
- **continuous +** or **continuous -** (phase sync mode only), a velocity dependent amount is added to the degree offset.

For phase synchronization the resulting slave phase is:

$$\varnothing_s = (\varnothing_m * (K_s/K_m)) + \text{adjust}$$

For velocity synchronization the resulting slave velocity is:

$$V_s = V_m * ((1 + \text{adjust}) * (K_s/K_m))$$

Where:

\varnothing_m = master axis phase

V_m = master axis velocity

K_s = slave axis master/slave ratio turns value

K_m = master axis master/slave ratio turns value

adjust = a value in the range of -100% to +300% for ratio, or 0° to +360° for phase.

Axis

This field indicates the axis to be adjusted. The entry may be an integer, global integer variable (GI1- GI256), program integer variable (Ix), or an equivalent label.

Degrees or Percent

Degrees (phase) or percent (velocity) - When enabled this allows fine offset to be added. This entry may be a float, global float variable (GF1- GF256), program float variable (Fx), or equivalent label. Valid adjustment ranges are 0 to 360 degrees or -100% to +300%.

ELSGrp1



GPP supports a maximum of eight ELS Groups. The initialization of the ELS Group's registers and program variables is defined in the ELSGrp1 setup icon. The following window appears when the ELSGrp1 icon is placed.

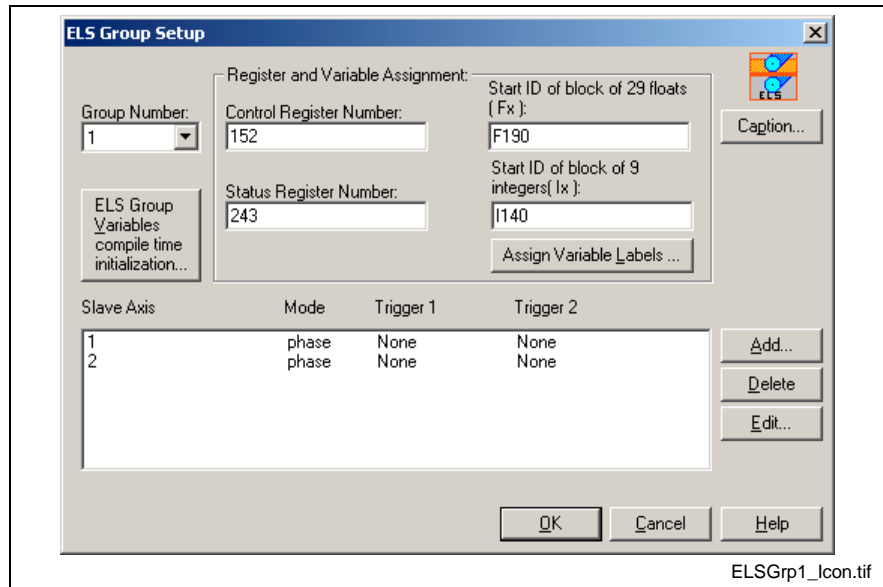


Fig. 3-48: ELS Group Setup

Group Number

Select an ELS Group number from the drop-down list.

Register and Variable Assignment

The "*Register and Variable Assignment*" section is used to define the registers and program variable blocks that will be used for the selected ELS Group number.

Note: Default values are automatically assigned for registers and variable blocks. It is strongly recommended that the programmer use the default values for registers and variables. This makes documentation and modifications to user program an easier task over the scope of the project.

Control Register Number

This number represents the control register number assigned to the ELS Group. Default control register labels and numbers for all 8 ELS Groups are listed in the table below.

Default Label ELS Group 1-8 Control Register	Data Object ELS Group Control Register-Bit								Comment (80 character limit) ELS Group 1-8 Control Register
	1	2	3	4	5	6	7	8	
G#_CT_LOCK_OFF	152-1	153-1	154-1	155-1	156-1	157-1	158-1	159-1	Group # control, 0 → 1 start lock cycle, 1 → 0 start unlock
G#_CT_M_REL_PH	152-2	153-2	154-2	155-2	156-2	157-2	158-2	159-2	Group # control, 0 → 1 triggers master relative phase adjust
G#_CT_S_REL_PH	152-3	153-3	154-3	155-3	156-3	157-3	158-3	159-3	Group # control, 0 → 1 triggers slave relative phase adjust
G#_CT_MSTR_SEL	152-4	153-4	154-4	155-4	156-4	157-4	158-4	159-4	Group # control, 0=master 1, 1=master 2
G#_CT_VAR_CLK	152-5	153-5	154-5	155-5	156-5	157-5	158-5	159-5	Group # control, 0 → 1 forcing
G#_CT_LOCAL	152-6	153-6	154-6	155-6	156-6	157-6	158-6	159-6	Group # control, 0 → 1 local mode, 1 → 0 selected master
G#_CT_JOG_INC	152-7	153-7	154-7	155-7	156-7	157-7	158-7	159-7	Group # control, 0=continuous jog mode, 1=incremental jog mode
G#_CT_JOG_ABS	152-8	153-8	154-8	155-8	156-8	157-8	158-8	159-8	Group # control, 0=absolute incremental mode, 1=relative incremental mode
G#_CT_JOG_PLUS	152-9	153-9	154-9	155-9	156-9	157-9	158-9	159-9	Group # control, 0 → 1 starts jog mode in positive direction
G#_CT_JOG_MINS	152-10	153-10	154-10	155-10	156-10	157-10	158-10	159-10	Group # control, 0 → 1 starts jog mode in negative direction
G#_CT_M_ABS_PH	152-11	153-11	154-11	155-11	156-11	157-11	158-11	159-11	Group # control, 0 → 1 triggers master absolute phase adjust
G#_CT_S_ABS_PH	152-12	153-12	154-12	155-12	156-12	157-12	158-12	159-12	Group # control, 0 → 1 triggers slave absolute phase adjust
Each # symbol represents an entry for the number of the ELS Group									

Table 3-18: ELS Group 1-8 Default Control Register Bits

Status Register Number

This number represents the status register number assigned to the ELS Group. Default status register labels and numbers for all 8 ELS Groups are listed in the table below.

Default Label ELS Group 1-8 Status Register	Data Object ELS Group Status Register-Bit								Comment (80 character limit) ELS Group 1-8 Status Register
	1	2	3	4	5	6	7	8	
G#_ST_LOCK_ON	243-1	244-1	245-1	246-1	247-1	248-1	249-1	250-1	Group # status, 0=unlocked, 1=locked to master
G#_ST_M_REL_PH	243-2	244-2	245-2	246-2	247-2	248-2	249-2	250-2	Group # status, 1=acknowledges master relative phase adjust
G#_ST_S_REL_PH	243-3	244-3	245-3	246-3	247-3	248-3	249-3	250-3	Group # status, 1=acknowledges slave relative phase adjust
G#_ST_MSTR_SEL	243-4	244-4	245-4	246-4	247-4	248-4	249-4	250-4	Group # status, 0=master 1, 1=master 2
G#_ST_VAR_ACK	243-5	244-5	245-5	246-5	247-5	248-5	249-5	250-5	Group # status, 1=variables updated
G#_ST_LOCAL	243-6	244-6	245-6	246-6	247-6	248-6	249-6	250-6	Group # status, 1=local mode active
G#_ST_RSVD7	243-7	244-7	245-7	246-7	247-7	248-7	249-7	250-7	
G#_ST_RSVD8	243-8	244-8	245-8	246-8	247-8	248-8	249-8	250-8	
G#_ST_MOTION	243-9	244-9	245-9	246-9	247-9	248-9	249-9	250-9	Group # status, 0=no motion, 1=group is in motion
G#_ST_JOG_POS	243-10	244-10	245-10	246-10	247-10	248-10	249-10	250-10	Group # status, 1=jog is at absolute target
G#_ST_M_ABS_PH	243-11	244-11	245-11	246-11	247-11	248-11	249-11	250-11	Group # status, 1=acknowledges master absolute phase adjust
G#_ST_S_ABS_PH	243-12	244-12	245-12	246-12	247-12	248-12	249-12	250-12	Group # status, 1=acknowledges slave absolute phase adjust

Each # symbol represents an entry for the number of the ELS Group

Table 3-19: ELS Group 1-8 Default Status Register Bits

ELS Group Program Variable Start ID Blocks

Start ID of block of 29 floats (Fx):

This number represents the first float in a block of 29 floats set aside for each ELS Group. The float number must be preceded with an "F".

Example: F190

Default float labels and numbers for all 8 ELS Groups are listed in the table below.

ELS Group Float Variable Default Label	ELS Group Float Number								ELS Group Float Variable Default Comment (80 character limit)	Update Mode
	1	2	3	4	5	6	7	8		
G#_SYNC_ACCEL	F190	F220	F250	F280	F310	F340	F370	F400	Group #, dynamic sync acceleration	Phase 4
G#_SYNC_VEL	F191	F221	F251	F281	F311	F341	F371	F401	Group #, dynamic sync velocity	Phase 4
G#_M1	F192	F222	F252	F282	F312	F342	F372	F402	Group #, M factor	Phase 4 & Forcing*
G#_N1	F193	F223	F253	F283	F313	F343	F373	F403	Group #, N factor	Phase 4 & Forcing*
G#_PROG_M_PH	F194	F224	F254	F284	F314	F344	F374	F404	Group #, master phase adjust	Phase 4
G#_PROG_S_PH	F195	F225	F255	F285	F315	F345	F375	F405	Group #, slave phase adjust	Phase 4
G#_ABS_M_PH	F196	F226	F256	F286	F316	F346	F376	F406	Group #, absolute master phase adjust	Phase 4 (read-only)
G#_ABS_S_PH	F197	F227	F257	F287	F317	F347	F377	F407	Group #, absolute slave phase adjust	Phase 4 (read-only)
G#_H_LOCKON	F198	F228	F258	F288	F318	F348	F378	F408	Group #, H factor lock on cam profile	Phase 4 & Forcing*
G#_H_RUN	F199	F229	F259	F289	F319	F349	F379	F409	Group #, H factor 1:1 cam profile	Phase 4 & Forcing*
G#_H_LOCKOFF	F200	F230	F260	F290	F320	F350	F380	F410	Group #, H factor lock off cam profile	Phase 4 & Forcing*
G#_H_USER	F201	F231	F261	F291	F321	F351	F381	F411	Group #, H factor user cam profile	Phase 4
G#_LOCK_WIN	F202	F232	F262	F292	F322	F352	F382	F412	Group #, shortest path window for dynamic sync. phase correction	Phase 4
G#_STOP_DECEL	F203	F233	F263	F293	F323	F353	F383	F413	Group #, stop ramp deceleration	Phase 4
G#_JOG_ACCEL	F204	F234	F264	F294	F324	F354	F384	F414	Group #, jog acceleration	Phase 4
G#_JOG_VEL	F205	F235	F265	F295	F325	F355	F385	F415	Group #, jog velocity	Phase 4
G#_JOG_INC	F206	F236	F266	F296	F326	F356	F386	F416	Group #, relative position distance (incremental jog)	Phase 4
G#_JOG_ABS	F207	F237	F267	F297	F327	F357	F387	F417	Group #, absolute position target (absolute jog)	Phase 4
G#_JOG_WIN	F208	F238	F268	F298	F328	F358	F388	F418	Group #, shortest path window for absolute jog	Phase 4
G#_LOCKON_OFFSET	F209	F239	F269	F299	F329	F359	F389	F419	Group #, offset added to the output when lock on cam profile is being forced	Phase 4
G#_IN_POS	F210	F240	F270	F300	F330	F360	F390	F420	Group #, input position	Phase 4 & Forcing*
G#_IN_VEL	F211	F241	F271	F301	F331	F361	F391	F421	Group #, input velocity (read only)	Phase 4
G#_OUT_POS	F212	F242	F272	F302	F332	F362	F392	F422	Group #, output position (read only)	Phase 4 & Forcing*
G#_OUT_VEL	F213	F243	F273	F303	F333	F363	F393	F423	Group #, output velocity (read only)	Phase 4
G#_OUT_ACC	F214	F244	F274	F304	F334	F364	F394	F424	Group #, output acceleration (read only)	Phase 4
G#_CAM_INPUT	F215	F245	F275	F305	F335	F365	F395	F425	Group #, group cam profile ID input position	Phase 4 & Forcing*
G#_MST1_TRIGPOS	F216	F246	F276	F306	F336	F366	F396	F426	Group #, master 1 switching trigger position	Phase 4
G#_MST2_TRIGPOS	F217	F247	F277	F307	F337	F367	F397	F427	Group #, master 2 switching trigger position	Phase 4
G#_STANDSTILL_WIN	F218	F248	F278	F308	F338	F368	F398	F428	Group #, standstill velocity threshold	Phase 4

Each # symbol represents the number of the ELS Group
 * Forcing is reinitializing an ELS Group in Phase 4 when local mode is active (G#_ST_LOCAL) and the ELS Group Master is at standstill (G#_ST_MOTION) is 0.

Table 3-20: ELS Group 1-8 Default Status Register Bits

Start ID of block of 9 integers (Ix):

This number represents the first integer in a block of 9 integers set aside for each ELS Group. The integer number must be preceded with an "I".

Example: I170

Default integer labels and numbers for all 8 ELS Groups are listed in the table below.

ELS Group Integer Variable Default Label	ELS Group Integer Variable								ELS Group Integer Variable Default Comment (80 character limit)	Update Mode
	1	2	3	4	5	6	7	8		
G#_CONFIG	I140	I150	I160	I170	I180	I190	I200	I210	Group #, configuration word	
G#_MSTR1_AXIS	I141	I151	I161	I171	I181	I191	I201	I211	Group #, ELS master ID, number 1	Phase 4
G#_MSTR2_AXIS	I142	I152	I162	I172	I182	I192	I202	I212	Group #, ELS master ID, number 2	Phase 4
G#_ACTIVE_STATE	I143	I153	I163	I173	I183	I193	I203	I213	Group #, active state of state machine for lockon/lockoff	Phase 4 & Forcing*
G#_ACTIVE_CAM	I144	I154	I164	I174	I184	I194	I204	I214	Group #, active cam profile table number	Phase 4
G#_LOCKON_CAM	I145	I155	I165	I175	I185	I195	I205	I215	Group #, lock on cam profile table number	Phase 4 & Forcing*
G#_RUN_CAM_ID	I146	I156	I166	I176	I186	I196	I206	I216	Group #, 1:1 cam profile table number	Phase 4 & Forcing*
G#_LOCKOFF_CAM	I147	I157	I167	I177	I187	I197	I207	I217	Group #, lock off cam profile table number	Phase 4 & Forcing*
G#_USER_CAM	I148	I158	I168	I178	I188	I198	I208	I218	Group #, user cam profile table number (state machine disabled)	Phase 4

Each # symbol represents the number of the ELS Group
 * Forcing is reinitializing an ELS Group in Phase 4 when local mode is active (G#_ST_LOCAL) and the ELS Group Master is at standstill (G#_ST_MOTION) is 0.

Table 3-21: ELS Group 1-8 Default Integer Variables

Assign Variable Labels

Default variable labels and comments can be added individually for Variable, Registers and Bits by selecting the appropriate **Data Type** radio button and clicking the **Add Default Labels** button. Clicking the **Add All Default Labels** button will add the default variable labels and comments to all Data Types at one time.

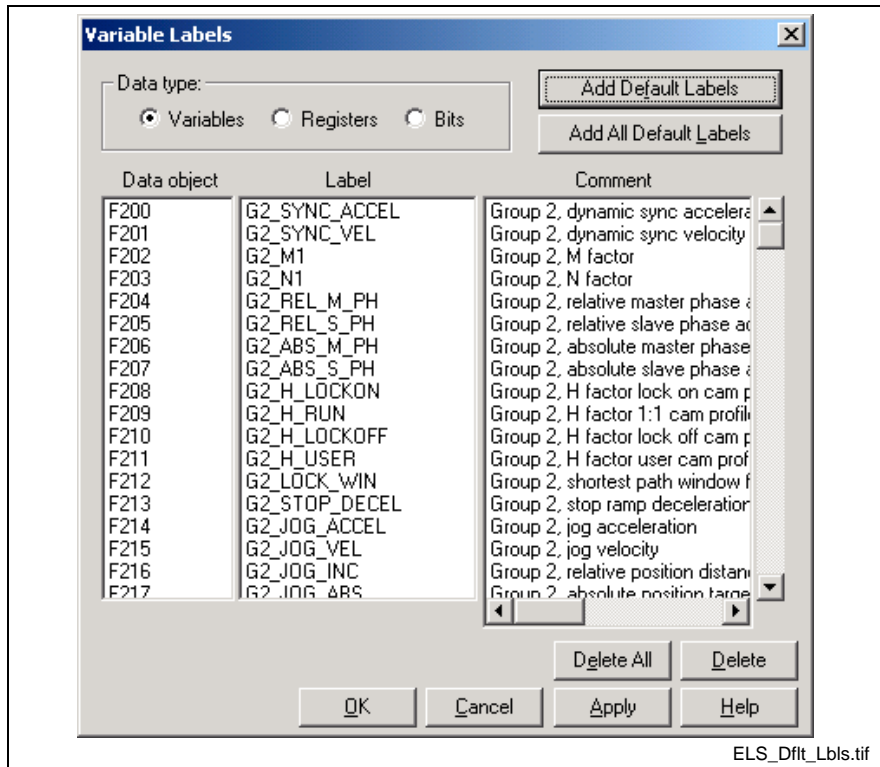


Fig. 3-49: ELS Default Variable Labels

ELS Axis Configuration

Clicking on the **Add** button in the *ELS Group Setup* window opens the *ELS Axis Configuration Edit* window.

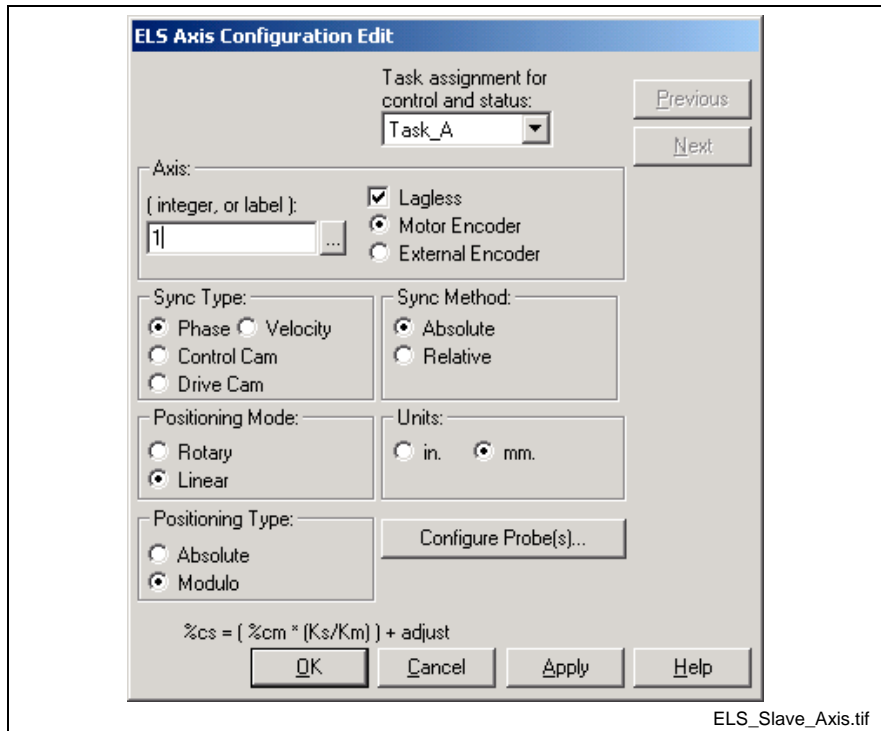


Fig. 3-50: ELS Axis Configuration

Task Assignment

Since the ELS Group icon can only be placed in the Initialization task, the **Task Assignment** for control and status of the ELS axis must be selected within the *ELS Axis Configuration Edit* window.

Axis

An **Axis** is identified by its SERCOS address, integer variable or by an assign label in VisualMotion Toolkit. Once addressed, the axis' measuring device is selected as either **Motor Encoder** or **External Encoder** with or without **Lagless** following.

Note: By default, an Axis is configured as **Initially Halted**. This disables the axis at the start of the task and must be enabled using a Go1 icon in the user program.

Sync Type

Select the radio button for the appropriated synchronization type that the axis will follow in the assigned ELS group. The following selections are available:

- **Phase** – the slave axis matches and follows an ELS Group's output position.
- **Velocity** – the slave axis matches and follows an ELS Group's output velocity regardless of any phase differences.
- **Control CAM** – the slave axis follows a control CAM profile whose master position is the ELS Group's output.
- **Drive CAM** – the slave axis follows a drive CAM profile whose master position is the ELS Group's output.

Sync Method

- Absolute
- Relative

Configure Probe(s)

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

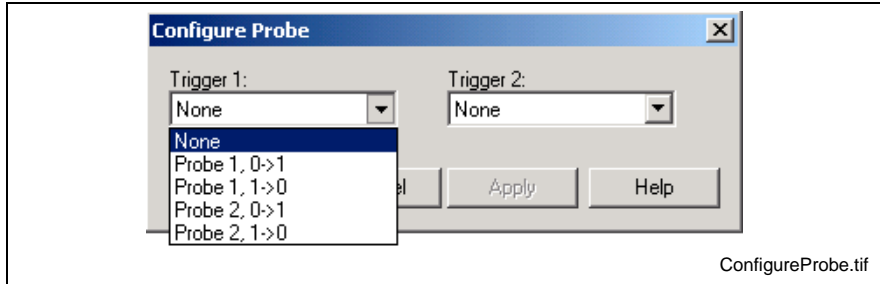


Fig. 3-51: Configure Probe

ELS Group Compile Time Initialization

Clicking the **ELS Group Variables and Compile Time Initialization** button opens the *Initialize ELS Group Variables* window. This window is used to assign ELS System Masters that the ELS Group will follow. It is also used to change the Master's positional data by adding a M/N ratio, CAM profile and a relative Master and/or Slave phase difference.

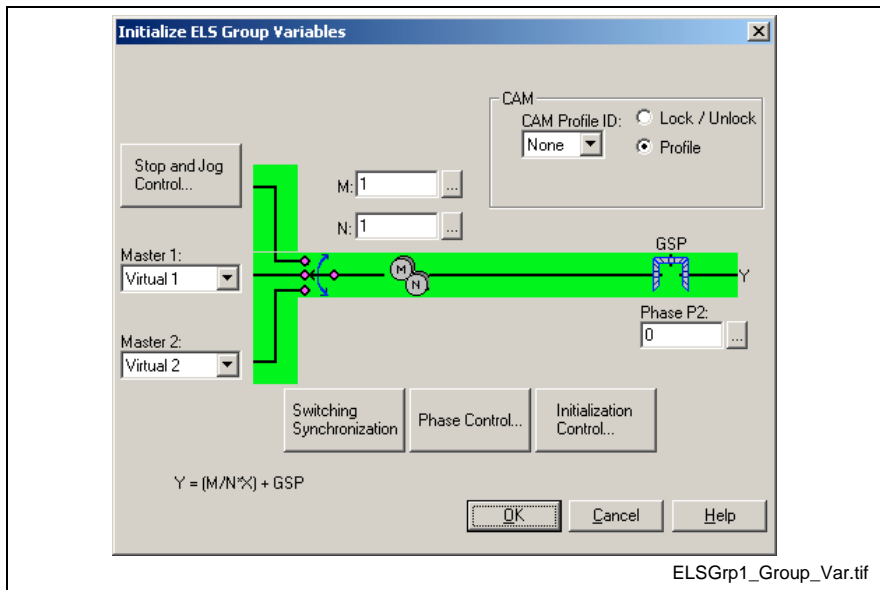



Fig. 3-52: Initializing ELS Group Variables

Refer to **ELS Group** in chapter 6 of the *VisualMotion 9 Application Manual* for details.

Clicking the browse  button to the right of M, N, H, Phase P1 or Phase P2 allows selection of a variable from the VM Data Table.

From this window, four main buttons are provided to access setup windows:

- CAM Profile

- Stop and Jog Control
- Switching Synchronization
- Phase Control
- Initialization Control

CAM Profile

The **CAM** section is used to enable a CAM profile or VisualMotion's Lock/Unlock CAM feature. The ELS Group's output graphic and equation (in the lower left-hand corner) change based on the selections made in the **CAM** section. Refer to Fig. 3-53 for details.

CAM profile ID = none
 ELS Group output will not have a GMP (relative Master phase) or CAM profile applied.
 CAM equation reads: $Y = (M/N * X) + GSP$

CAM profile ID = CAM number
 ELS Group output can have a GMP (relative Master phase) or CAM profile applied.
 CAM equation reads: $Y = H * CAM[M/N * X + GMP] + GSP$

CAM = Lock / UnLock
 ELS Group output can have a GMP (relative Master phase) and default Lock / Unlock CAMs applied.
 CAM equation reads: $Y = H * CAM[M/N * X + GMP] + GSP$

Advanced...
 Displays default Lock / Run / Unlock CAMs (38, 39 and 40). User defined CAMs can be selected.

Fig. 3-53: ELS Group CAM Section

Stop and Jog Control

Refer to **Stop and Jog Control** in chapter 6 of the *VisualMotion 9 Application Manual* for details.

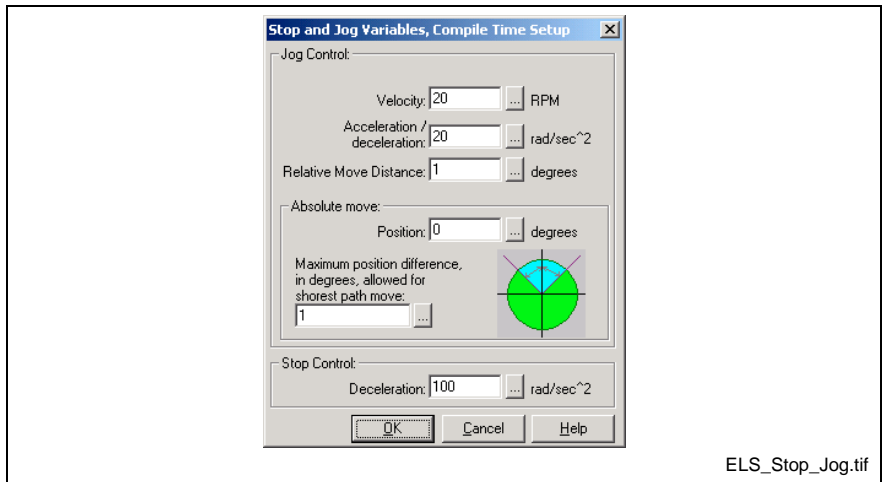


Fig. 3-54: Stop and Jog Control Configuration

Synchronization Setup

Refer to **Synchronization Setup** in chapter 6 of the *VisualMotion 9 Application Manual* for details.

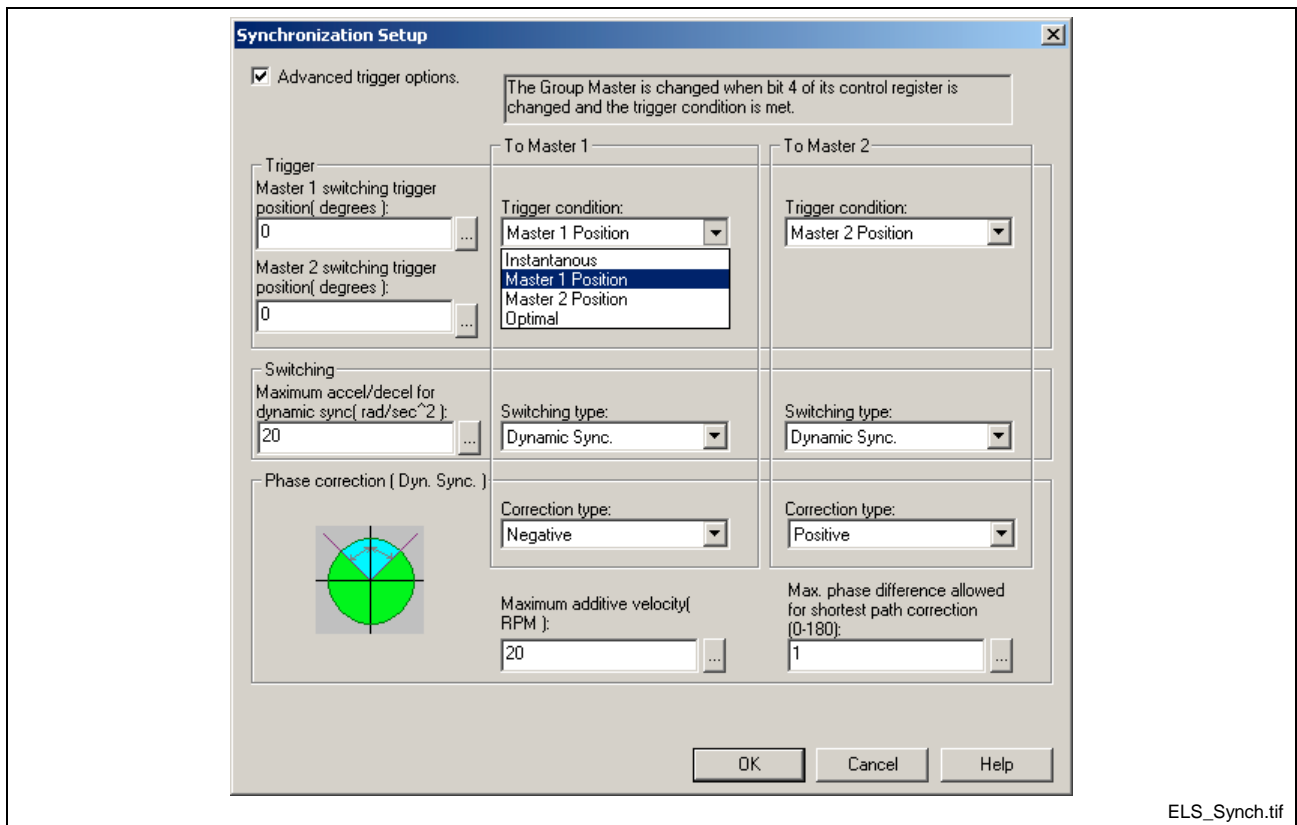
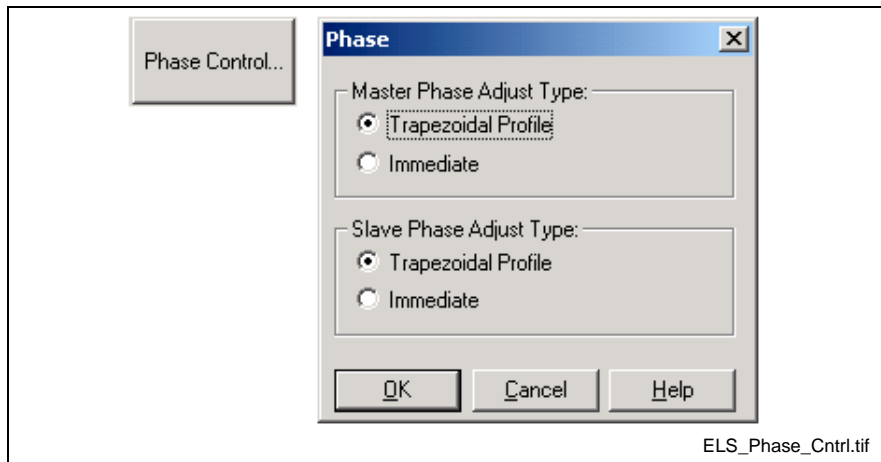


Fig. 3-55: ELS Synchronization Setup

Phase Control

Refer to **Phase Control** in chapter 6 of the *VisualMotion 9 Application Manual* for details.

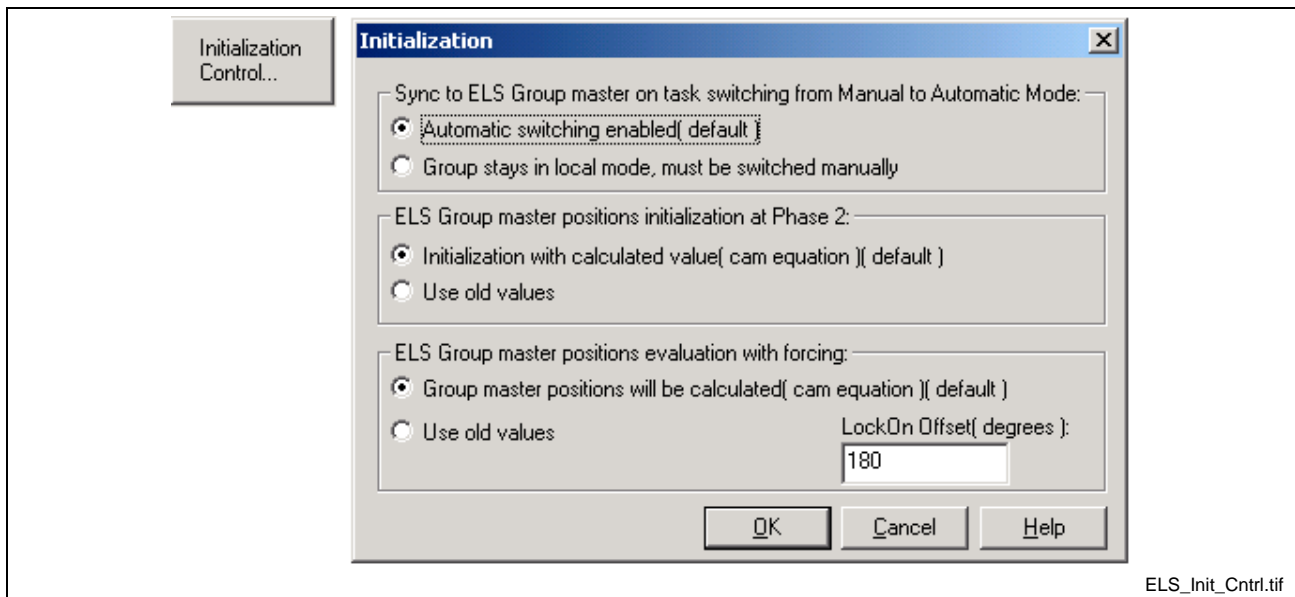


ELS_Phase_Cntrl.tif

Fig. 3-56: ELS Phase Control

Initialization Controls

Refer to **Initialization Control** in chapter 6 of the *VisualMotion 9 Application Manual* for details.



ELS_Init_Cntrl.tif

Fig. 3-57: ELS Initialization Control

ELSMstr1



The ELSMstr1 icon allows for assignment of up to six ELS System Masters. Each ELS Master is associated with a number from 1 to 6. Using software, this association creates an ELS Master connection box that uses the master's signal (commanded position) as an input to an ELS Group.

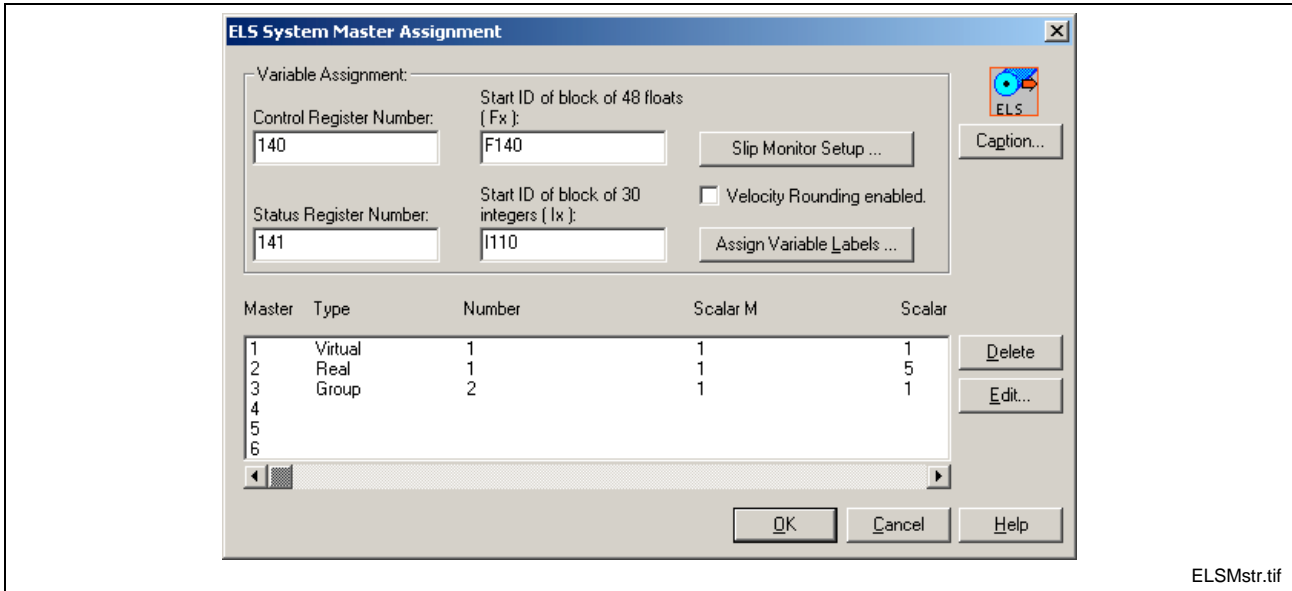


Fig. 3-58: ELS Master Assignment

Note: An ELS System Master is automatically assigned to Task A. All control and monitoring of ELS System Masters is performed by the control and status registers of Task A. All ELS Group axes that are Sync to Master and following an ELS System Master will stop if Task A is stopped.

The ELS System Master icon must be placed in the main level of the Initialization task. It should not be placed in an Initialization Subroutine or a system error will be issued when the program is compiled and downloaded to the control.

Slip Monitor Setup

The Slip Monitor Setup feature is used to monitor the difference in position (phase) between two ELS System Masters and initiates an error reaction when the difference is outside the allowable deviation window. Refer to Slip Monitoring in chapter 6 of the *VisualMotion 9 Application Manual* for details.

Velocity Rounding Enabled

Velocity Rounding sets the Virtual Master and Group jogging velocities down to the nearest ELS increment to eliminate cycle-to-cycle variations in drive velocity. This results in the velocities being slightly less than their commanded values. This feature can be enabled and disabled in bit 30 (ELS_MSTR_CONFIG) of the ELS Group Configuration Word.

Variable Assignment

The Variable Assignment section is used to define the start ID blocks for program variables that will be used for all ELS System Masters.

ELS System Master Program Variable Start ID Blocks

Start ID of block of 48 floats (Fx):

This number represents the first float in a block of 48 floats set aside for all 6 ELS System Masters. The float number must be preceded with an "F". **Example:** F140

Start ID of block of 30 integers (Ix):

This number represents the first integer in a block of 30 integers set aside for all 6 ELS System Masters. The integer number must be preceded with an "I". **Example:** I110

Default program variable labels and numbers for all 6 ELS System Masters are listed in the table below.

ELS System Master Program Variable	ELS System Master Program Variable						ELS System Master Program Variable Default Comment (80 character limit)	Update Mode
	1	2	3	4	5	6		
ELS_MSTR_FREQ#	F140	F141	F142	F143	F144	F145	ELS Master # filter cutoff frequency	Phase 2
ELS_MSTR_M#	F146	F147	F148	F149	F150	F151	ELS Master # M factor	Phase 2
ELS_MSTR_N#	F152	F153	F154	F155	F156	F157	ELS Master # N factor	Phase 2
ELS_MSTR_SLIP_WINDOW	F158						ELS Master max allowed slip deviation window	Captured on rising edge of capture bit in P4
ELS_MSTR_SLIP_OFFSET	F159						ELS Master position offset for slip monitoring	Phase 2/4
ELS_MSTR_SLIP_VELTHD	F160						ELS Master slip monitoring primary velocity threshold	Phase 4
ELS_MSTR_SLIP_PEAK	F161						ELS Master peak slip deviation	Phase 4 (read-only)
ELS_MSTR_SLIP_ACTUAL	F162						ELS Master current slip deviation (actual)	Phase 4 (read-only)
ELS_MSTR_STANDSTILL	F163						ELS Master Standstill Velocity Threshold	Phase 4
ELS_MSTR_POS#	F164	F165	F166	F167	F168	F169	ELS Master # output position	Phase 4 (read-only)
ELS_MSTR_VEL#	F170	F171	F172	F173	F174	F175	ELS Master # output velocity	Phase 4 (read-only)
ELS_MSTR_OFFSET#	F176	F177	F178	F179	F180	F181	ELS Master # real master offset	Phase 4
ELS_MSTR_REF_POS#	F182	F183	F184	F185	F186	F187	ELS Master # real master reference position	Phase 4
ELS_MSTR_A#	I110	I111	I112	I113	I114	I115	ELS Master # ID number	Phase 2
ELS_MSTR_EC#	I116	I117	I118	I119	I120	I121	ELS Master # encoder, Real Master only	Phase 2
ELS_MSTR_FLTR#	I122	I123	I124	I125	I126	I127	ELS Master # filter	Phase 2
ELS_MSTR_TYPE#	I128	I129	I130	I131	I132	I133	ELS Master # type	Phase 2
ELS_MSTR_SLIP_PRI	I134						ELS Master slip primary address	Phase 2
ELS_MSTR_SLIP_SEC	I135						ELS Master slip secondary address	Phase 2
ELS_MSTR_CONFIG	I136						ELS Master slip monitoring settings	Phase 2/4
ELS_MSTR_RSVD1	I137						reserved for ELS Master	Phase 2
ELS_MSTR_RSVD2	I138						reserved for ELS Master	Phase 2
ELS_MSTR_RSVD3	I139						reserved for ELS Master	Phase 2
Each # symbol represents the number of the ELS Master Shaded variables are read-only (You can overwrite the current value. However, ELS will overwrite your value if necessary)								

Table 3-22: ELS System Master Program Variables

Refer to [ELS Master Variable Definition](#) in chapter 6 of the *VisualMotion 9 Application Manual* for details.

Assign Variable Labels

Default variable labels and comments are added to Variables by clicking the **Add Default Labels** button. The **Add All Default Labels** button is grayed out since only variables can be assigned default labels for ELS System masters.

Note: Default values are automatically assigned for program variable blocks. It is strongly recommended that the programmer use the default values for program variables. This makes documentation and modifications to user program an easier task over the scope of the project.

ELS System Master Setup

Double clicking on one of the Master numbers opens the *Setup ELS System Master* window. The data entry fields of the *Setup ELS System Master* window are dependent upon the type of master type selected.

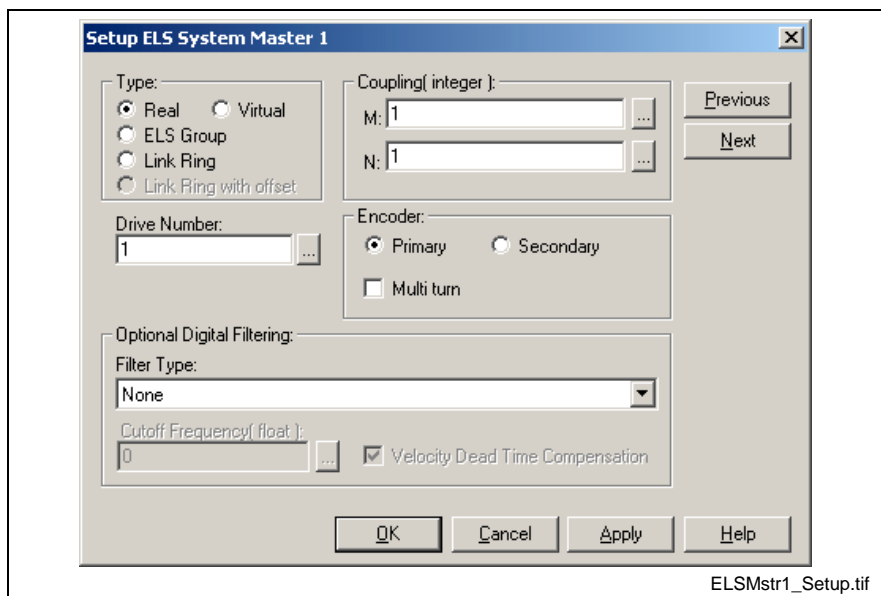


Fig. 3-59: ELS Master Setup

Type

- **Real** - A Real Master is either a primary (motor) or secondary encoder (position feedback) from a drive. Each drive in the system can potentially provide two Real Masters. The raw position value of the Real Master can be filtered and geared by a M/N ratio. A maximum of three Real Masters can be assigned.

Note: A Real Master can be primary encoder that are not a slave of an ELS Group or secondary encoder.

- **Virtual** - A Virtual Master is an internal motion engine with an independent set of control parameters. A maximum of two Virtual Masters can be assigned. Each Virtual Master can be used independently from the other. A Virtual Master is controlled by VisualMotion and/or a PLC using I/O registers and program variables.

Note: Virtual Masters are initialized using the Virtual Master icon before they are assigned a number.

- **ELS Group** - An ELS Group Master is the output of an ELS Group that can be used as an input master signal to a different ELS Group.
- **Link Ring** - This option sets the selected ELS System Master to receive the master position of the *External Number* Link Ring node.

Master Number

The **Master number** field displays different heading based on the ELS System Master selected.

- **Real Masters** - enter the SERCOS drive address of the drive containing the primary or secondary encoder.
- **Virtual Master** – enter the number (1 or 2) of the desired Virtual Master.
- **ELS Group** – enter the ELS Group number (1-8) whose output will be used as an ELS System Master.
- **Link Ring** – enter the Link Ring node number (1-32) whose output position will be used as an ELS System Master.

Coupling (float) (for Real Master only)

The M/N ratio is only available for Real Master outputs. The positional value for a Real Master output is multiplied by the M/N ratio and/or a Digital Filter and used as a new position value for an ELS Group input.

Example:

If a Real Master position is at 180° and a M/N value of 2/1 is used, then the ELS Group input will follow a position value of 360°.

If a Real Master position is at 180° and a M/N value of 1/2 is used, then the ELS Group input will follow a position value of 90°.

Encoder

Select the encoder type as either a **Primary** or a **Secondary** encoder. If the encoder type selected is an absolute or multi-turn encoder, place a check in the **Multi turn** box.

Optional Digital Filtering

Digital filtering is available for Real Masters and PID loops.

Filter Type:

- None
- First order low-pass, $G(s)=1/(s+1)$
- Second order low-pass, $G(s)=1/(s^2+2s+1)$
- Third order low-pass, $G(s)=1/(s^3+3s^2+3s+1)$
- Second order Butterworth, $G(s)=1/(s^2+2^{1/2}s+1)$
- Third order Butterworth, $G(s)=1/(s^3+2s^2+2s+1)$
- Modified 2nd order low-pass with velocity ramp tracking,
 $G(s)=(2s+1)/(s^2+2s+1)$
- Modified 3rd order low-pass with accel ramp tracking,
 $G(s)=(3s^2+3s+1)/(s^3+3s^2+3s+1)$

Cutoff Frequency (float):

When a filter type is chosen, a cutoff frequency for the filter must be entered. The cutoff frequency is the frequency where the signal is reduced by 3db. When set to 0 the filter is disabled.

To ensure a stable system, use the following calculation when entering a value for the Digital Filter Cutoff Frequency:

$$\text{Cutoff Frequency} \leq \frac{1}{2 * \text{Sampling Rate(sec.)}}$$

The sampling rate for a drive axis is the set phase 4 SERCOS cycle time (S-0-0002), entered in seconds.

Example: For a 2 ms SERCOS cycle time, the cutoff frequency is calculated as follows:

$$\text{Cutoff Frequency} \leq \frac{1}{2 * 0.002} = 250 \text{ Hz}$$

The following figure illustrates the frequency vs. degrees for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

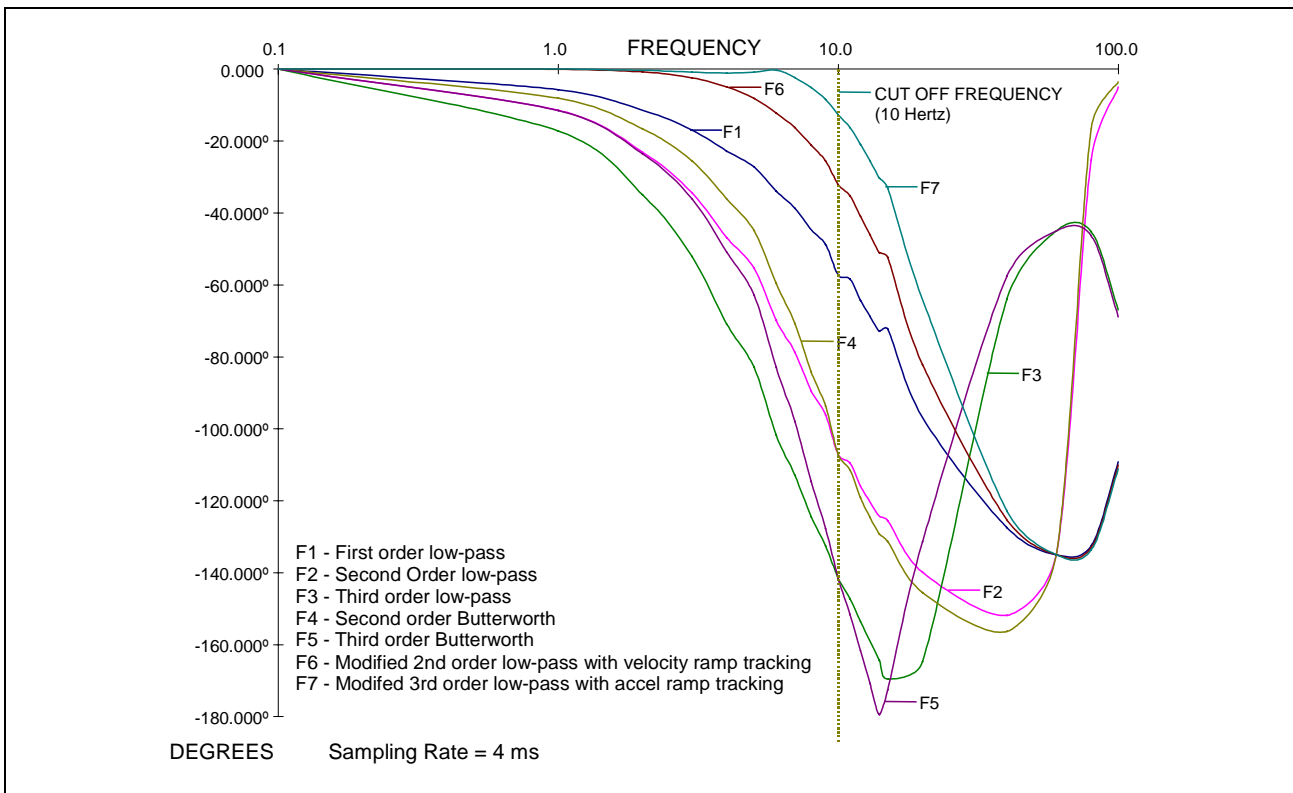


Fig. 3-60: Frequency vs. Degree Filter Chart

The following figure illustrates the gain vs. frequency for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

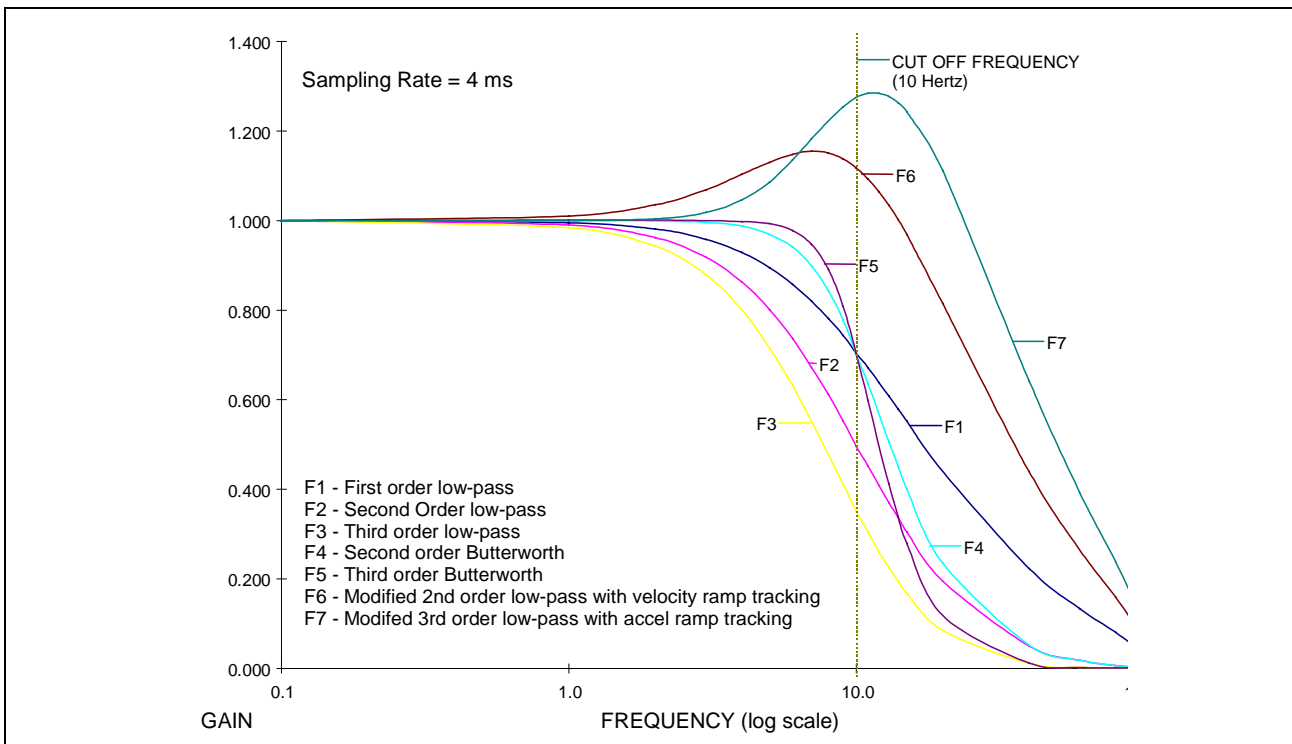


Fig. 3-61: Gain vs. Frequency Filter Chart

When a filter is chosen, the cutoff frequency for the filter must be entered. For example, the cutoff frequency for the First order low-pass filter is the frequency where the signal is reduced 3db [$.707$ gain, $db=20*\log(\text{gain})$].

Velocity Dead Time Compensation

Velocity Dead Time Compensation becomes active when a digital filter is applied to a Real Master. This feature allows the user to add 4 SERCOS cycles of velocity feed forward phase advance to an ELS Real Master to compensate for delays in control processing. The phase advance is performed where the Real Master position data is brought into the ELS System Masters such that the ELS System Master output user variables will reflect the phase advance. Since ELS propagates information from the Masters to the Groups via position and velocity (primarily by velocity), both the Real Master's position and velocity are advanced.

The Dead Time Compensation can be disabled for individual Real Masters for applications in which undershoot/overshoot during velocity changes could cause problems. The functionality of the feature is such that compensation is enabled by default. Users will need to disable it if they do not want to use it with their application.

Note: Dead Time Compensation ONLY compensates for the phase lag created by the 4 cycles of ELS processing / SERCOS delays. It DOES NOT compensate for additional dead time incurred by the various Real Master filters.

ELSMoDe



The ELSMoDe icon is used to switch between Single Axis, Velocity and Sync to Master Modes. Once switched into Single axis mode, the axis may be positioned independent of any ELS master/slave relationship (i.e., jogged into position), then returned to the master/slave condition.

Note: When an Mode Change icon is used to change an axis from Single Axis mode to Sync to Master mode, a second Mode Change icon must be used if the user program is to encounter a single axis icon, such as Home.

The ELSMoDe icon can also be used to switch an axis that is configured for Single Axis Mode into Velocity Mode. Axis parameter **A-0-0180** must be set to 36 to place command velocity into the cyclic data telegram. Axis parameter **A-0-0004, bit 7** must be set to 1 to enable acceleration. The Sync to Master mode is then ignored.

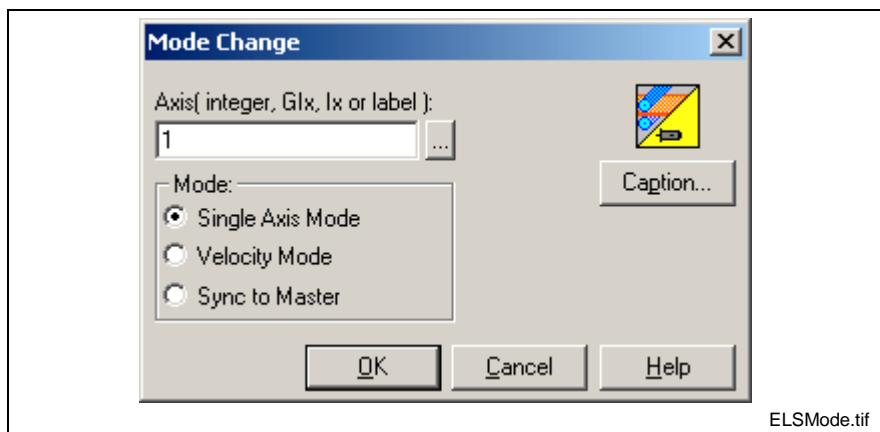


Fig. 3-62: ELS Mode Change

The axis is specified by an integer constant, variable, global variable or an equivalent label. Click the browse button to locate a data type from the VM Data Table. Entering “-1” in the axis box will send the mode change command to all ELS axes that were defined in the Task where the command is issued.

When an axis' Sync type is configured for Phase or Drive CAM mode is switched into sync mode, a relative phase offset is automatically initialized between the slave and the master. The drive does not move into absolute synchronization with the master.

Care should be taken when switching an axis into synchronization with a moving master. DIAX04 drives have the “ramp up and lock on” feature that assures smooth acceleration when synchronizing to a moving master.

When an axis is switched to single axis mode while the master is moving, it does not decel to a stop. It will stop at the last valid position command. To switch to single axis mode from following a moving master, first switch to velocity mode. The slave will continue moving at the last sampled master velocity (even in phase sync) and can be ramped down to a stop using the stop icons.

Event2



The Event2 icon allows the user to completely configure Events in project mode (Offline or Online) using a systematic process. Events are used to start an Event function (subroutine) when a specific condition (Event Trigger) is encountered in the program.

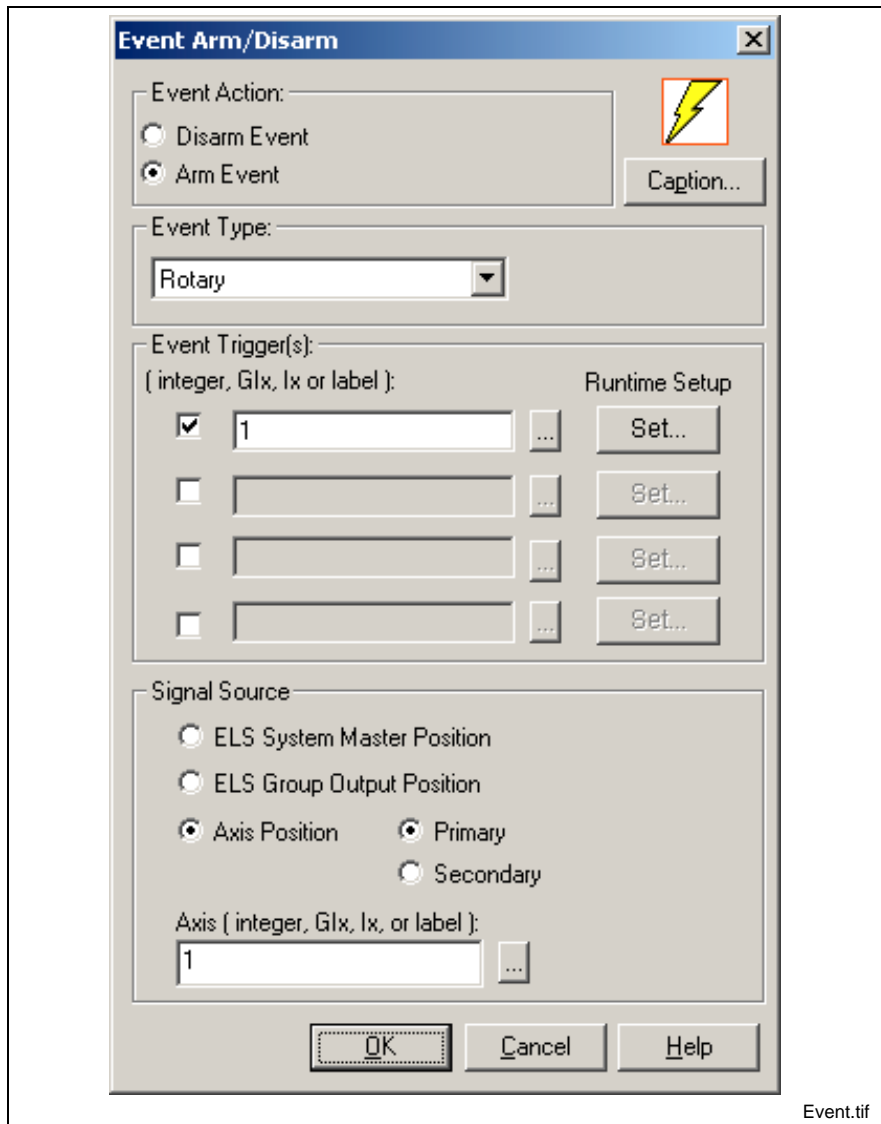


Fig. 3-63: Event Setup

Event Action:

Choose the event action to be performed by selecting one of two following radio buttons:

- **Disarm Event** - de-activates the specified Event trigger. If the event has already been made inactive, the icon has no effect.
- **Arm Event** – used to arm the 4 event types available from the Event Type drop-down list. Also used to when changing the argument of an event.

Note: A "Wait for Event Done" condition is performed by using the Wait icon.

Event Type

The event type drop-down list is available only when the **Arm Event** radio button is selected. The event types in the Event2 icon have been categorized into 4 types. Different event related fields of the *Event Arm/Disarm* window are displayed based on the selected Event Type. The following table lists the event types available in the Event2 icon along with their arming behavior, maximum number and description.

Event Type	Auto Rearming	Maximum Number	Description
Repeating Timer	Yes	16 per project (program)	A time-based event that executes an event function every programmed time interval.
Rotary	Yes	4 per axis, group or master	A position based event that executes an event function every time a programmed position is encountered.
I/O	No	Task Input Transition: 1 per task I/O Register: 16 per project (program) PPC-R Input: 3 per project (program)	An I/O based event that executes an event function when the programmed bit state of Input or Output is detected. The following I/O event types are available: - <i>Task Input Transition</i> - <i>I/O Register Event</i> - <i>PPC-R Input Event (0 -> 1)</i> - <i>PPC-R Input Event (1 -> 0)</i>
Probe	No	2 per drive	A drive-based probe event that executes an event function when a registration input is detected by the drive.

Table 3-23: Event Types

Note: For Single Axis and Coordinated Motion position-based events refer to the following icons for descriptions:

Move icon  for Single Axis

Path  and Circle  icon for Coordinated Motion


Event Triggers

Each event type must have a trigger (condition) that is used to start the event function (subroutine). Event triggers can be entered as an integer (Glx or lx) or label. The number of available event triggers varies based on the selected *Event Type*. The following table outlines the number of event triggers per *Event Type*.

Event Type	Number of Triggers
Repeating Timer	1
Rotary	4
I/O	1
Probe	1

Table 3-24: Number of Available Triggers

Selecting Events

Event triggers can be selected by clicking the browse button  within the Event2 icon. The browse button opens the VM Data Table where events are added, deleted or modified.

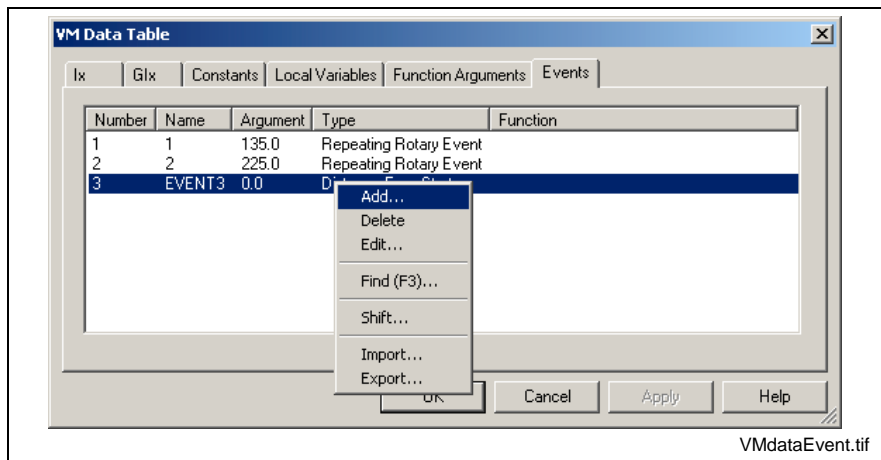



Fig. 3-64: Events in VM Data Table

Adding Events

Events are added to a VisualMotion project in the VM Data Table. The window can be opened by selecting the browse button to the right of the event trigger in the Event2 icon or by selecting the VM Data Table icon  from the toolbar.

Note: The configuration of events, including numbering, naming, arguments, assigning function, and adding messages can be performed both Offline and Online. However, the creation of Event Functions can only be performed Offline.

The following figure illustrates the different windows that are displayed when adding an event.

Note: When the VM Data Table is opened through the Event2 icon, only the allowable event data types are displayed. Opening the VM Data Table from the toolbar icon will display all of the available data types in VisualMotion.

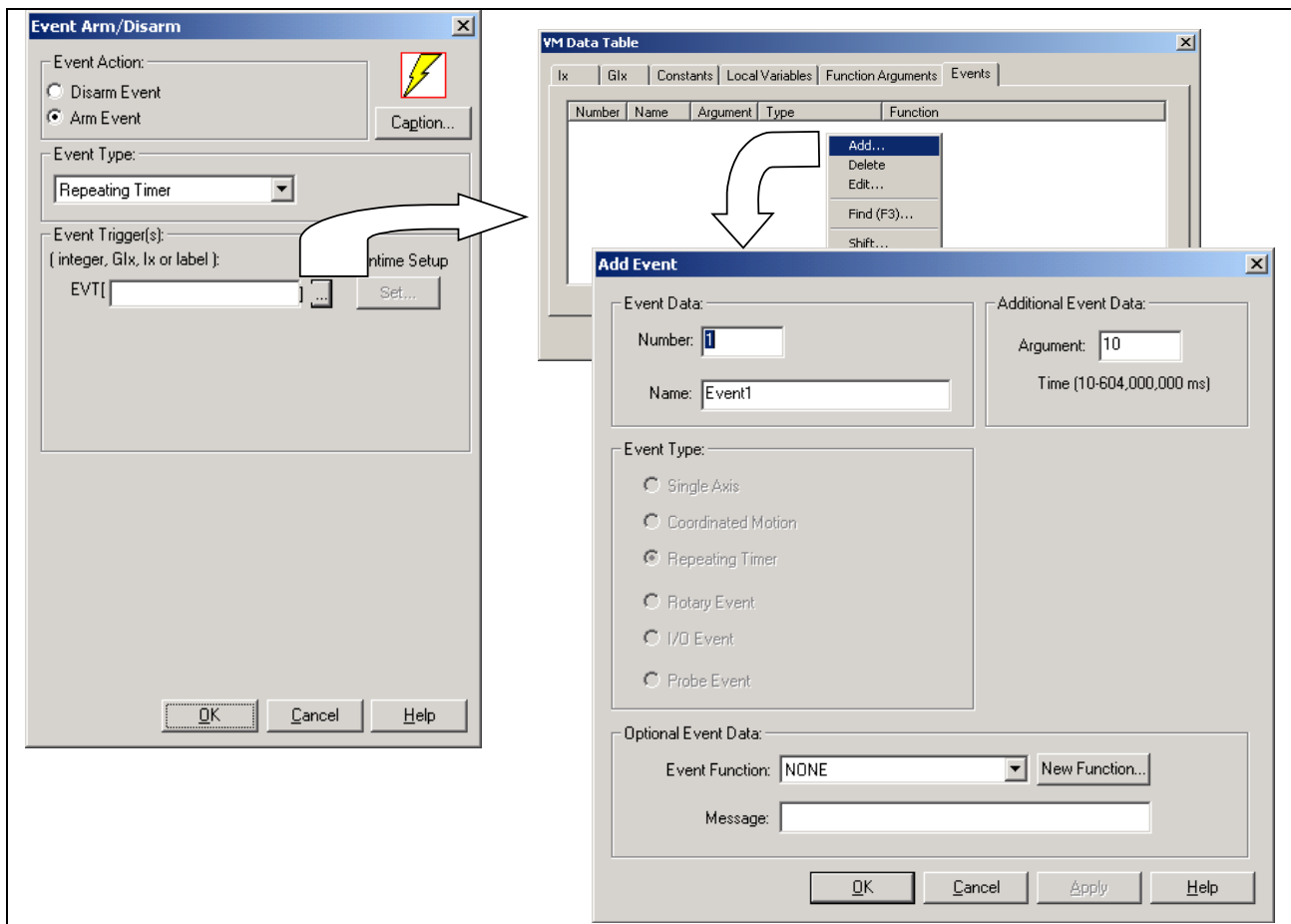


Fig. 3-65: Add an Event

Note: When an event is added through the Event2 icon, the Event Type section of the *Add Event* window is grayed out and the Event Type selected in the *Event Arm/Disarm* window is selected.

Right click and select **Add...** in the VM Data Table to open the *Add Event* window in Fig. 3-65. From this window, the user can configure the following items:

- **Number** the Event
- **Name** the Event
- Assign a value to the **Argument**
- Select an existing **Event Function** from a drop-down list
- Create a **New Function** from the *New Function* button
- Add a **Message** to the event

Note: When the *New Function* button is selected, the user can only assign a name to the function. The actual function is created in the VM Data Table.

Event functions can also be created by selecting *Insert* ⇒ **Event Function**, naming the event and creating the actual function. This method would make the function names available from the Event Function drop-down list in the *Add Event* window.

Runtime Setup

The **Runtime Setup** button is an advanced feature used to change the argument of an event trigger, the Event Function or both within the program flow during runtime. Refer to the *VisualMotion 9 Application manual, Programming Concept* chapter for details.

Note: Although the Calc2 icon can still be used to change the argument of an event trigger, the Event2 icon Runtime Setup makes it easier. When using the Calc2 icon, the user needs to know the correct syntax to use when changing the argument of an event trigger.

Repeating Timer Event Type

The *Event Arm/Disarm* window is displayed as follows when the Repeating Timer event type is selected. Only 1 event trigger can be assigned to a repeating timer.

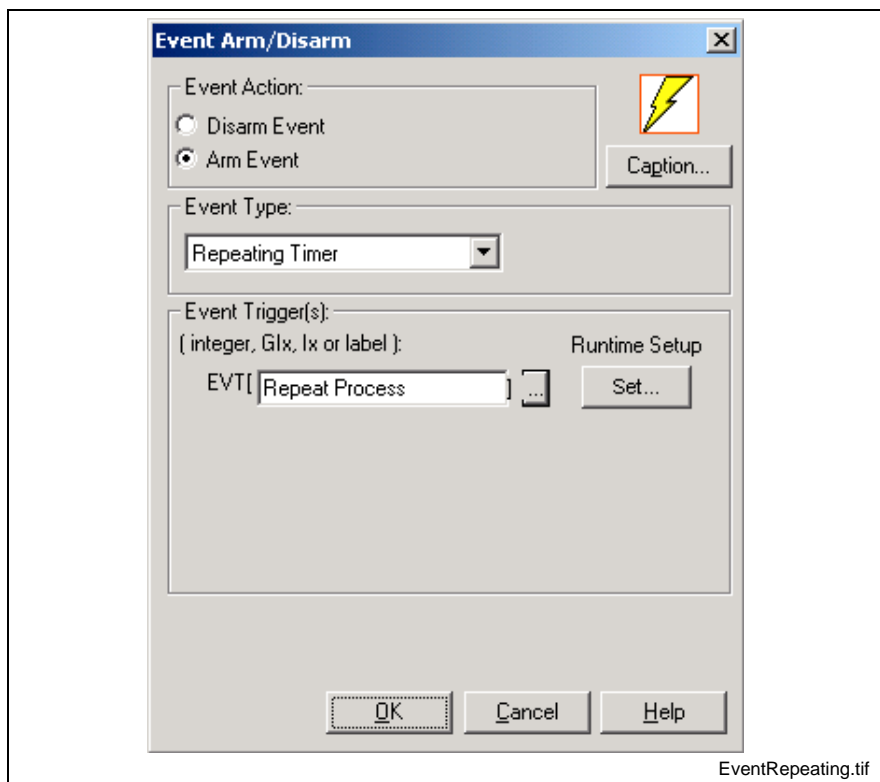


Fig. 3-66: Repeating Timer Event Type

Rotary Event Type

The *Event Arm/Disarm* window is displayed as follows when the Rotary event type is selected. Up to four event triggers can be assigned to any Signal Source of a Rotary event.

Note: When a rotary event is assigned to a main task, other than task A and the signal source is either ELS System Master or ELS Group, task A must be running in order for the rotary event to fire. The reason is that VisualMotion's ELS system is associated with task A. An ELS program will not function if task A is not running.

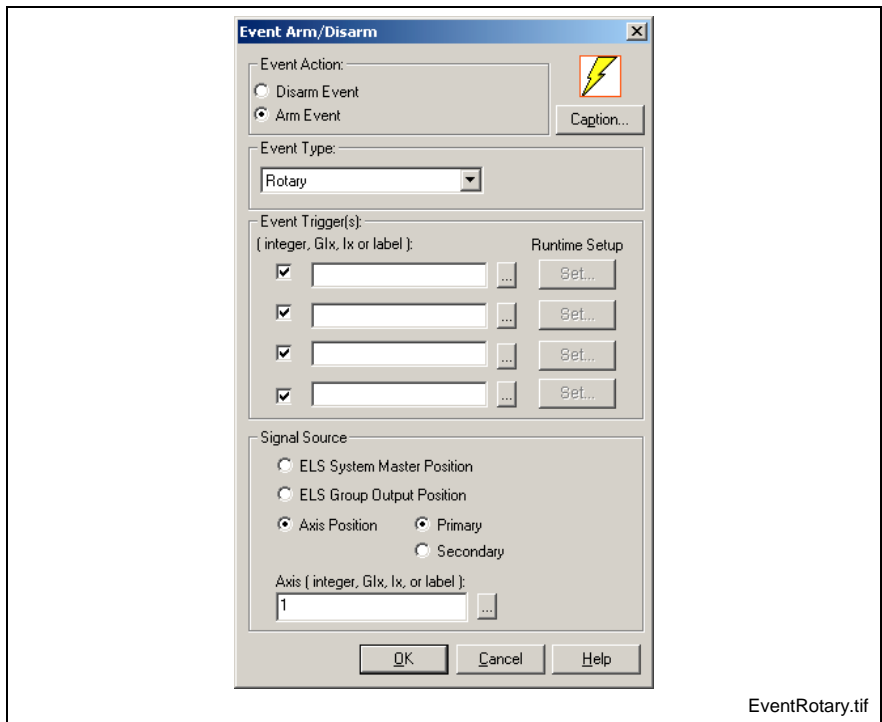


Fig. 3-67: Rotary Event Type

I/O Event Type

The *Event Arm/Disarm* window is displayed as follows when the I/O event type is selected. Only 1 Event Trigger can be assigned to an I/O event.

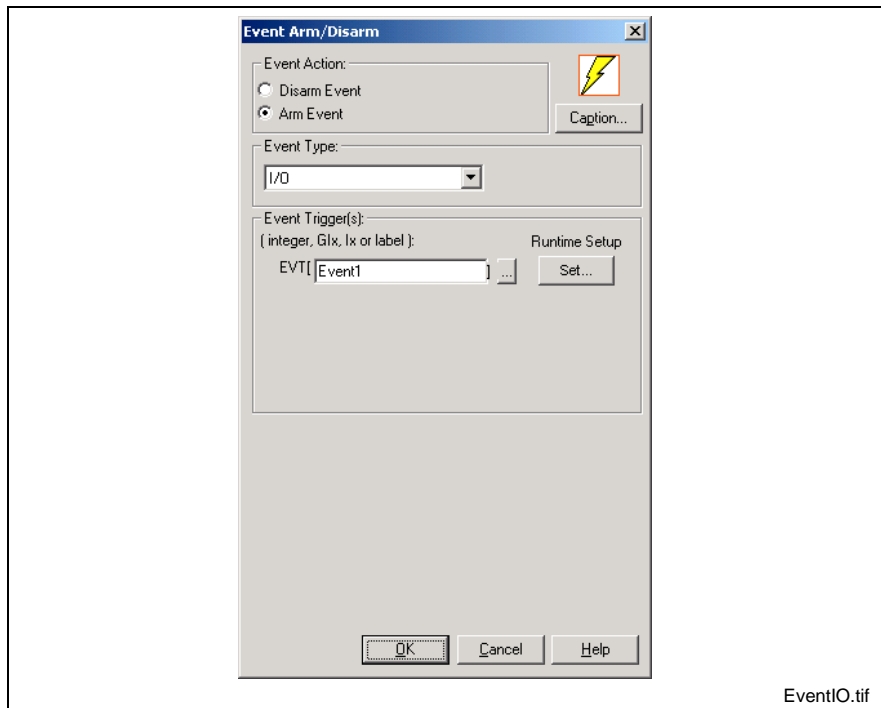


Fig. 3-68: I/O Event Type

The user can select from up to four different I/O events in the VM Data Table's *Add Event* window. The available types are listed in the figure below.

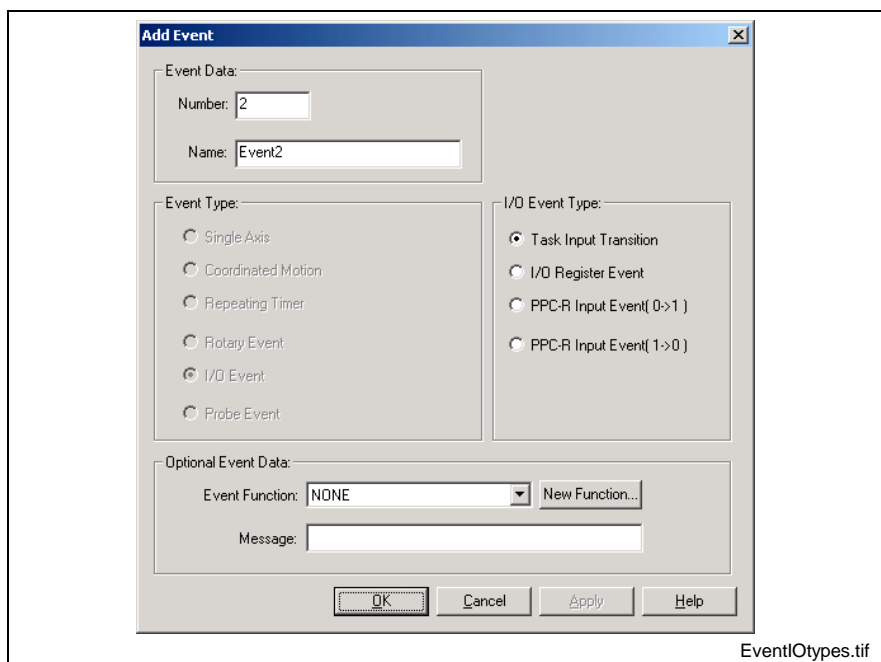


Fig. 3-69: Select I/O Event Type

Probe Event Type

The *Event Arm/Disarm* window is displayed as follows when the Probe event type is selected. Only 1 Event Trigger can be assigned to a Probe event. The Probe Trigger Selection in the Event2 icon will use the configured probe settings in the **Error! Reference source not found.** or Axis icons for the selected axis number.

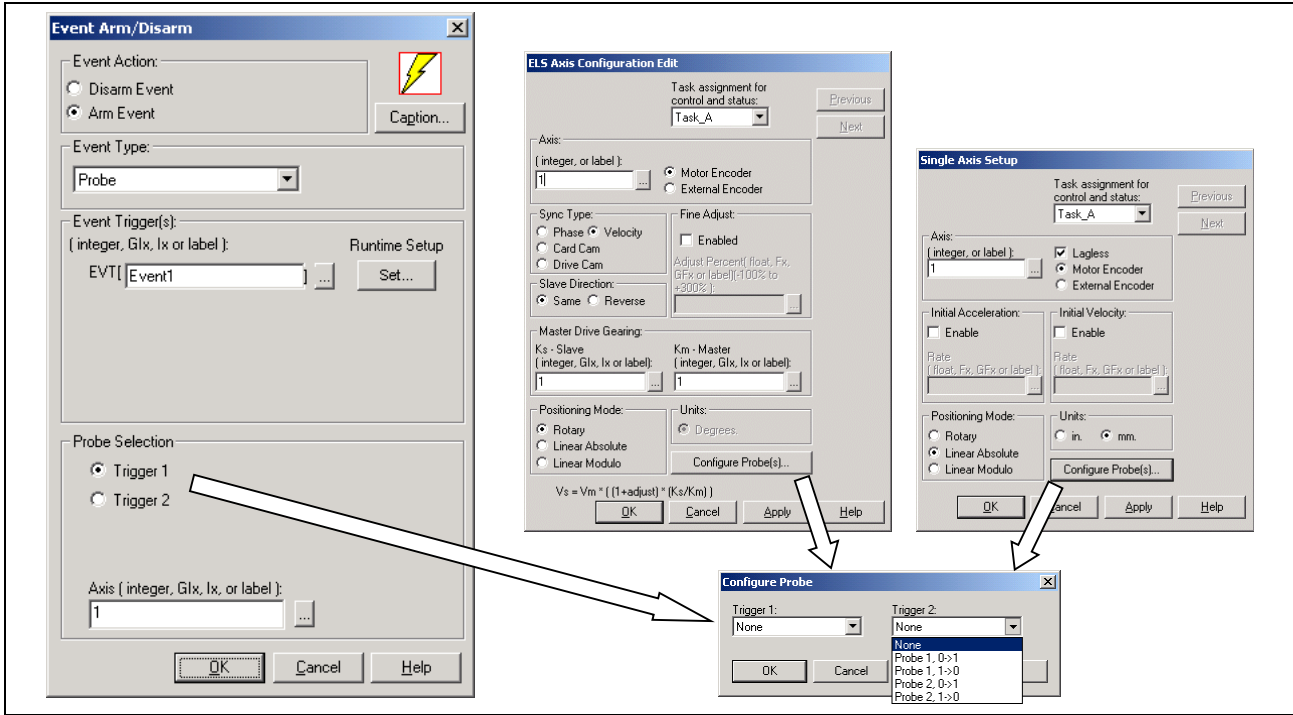
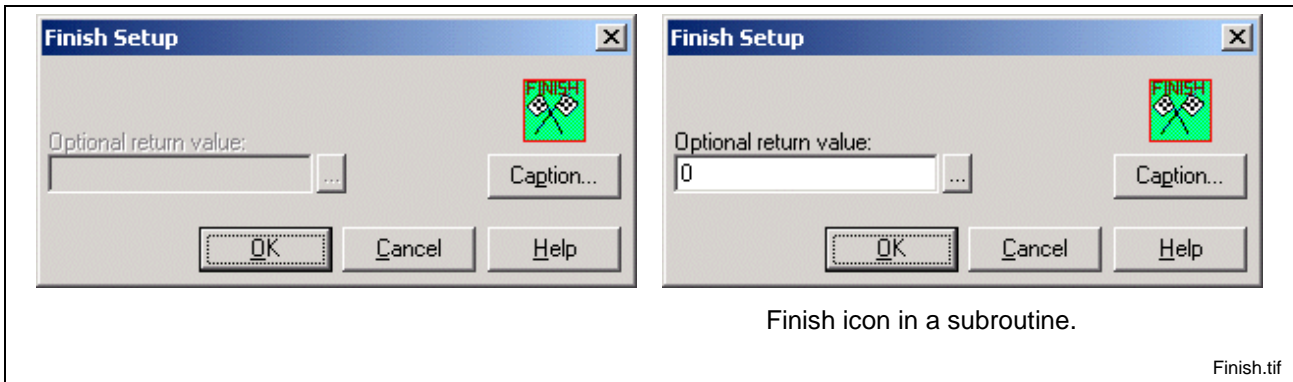


Fig. 3-70: Probe Event Type

Finish1



Each program task, subroutine and event function must end with a single Finish1 icon. Subroutines can return a single optional argument value to the calling function. The return argument may be a constant or a variable. A return argument is a convenient way to get position when using a common subroutine from more than one task. The **Optional return value** in the finish icon is only enabled from a finish icon in a subroutine. Task and event function cannot return an argument value. Refer to Optional Function Arguments (Sub1 icon) on page 3-116 for details.



Finish icon in a subroutine.

Finish.tif

Fig. 3-71: Finish Setup

Go1



The Go1 icon is used to enable one or all non-coordinated axes used in any task. It also enables the associated position, velocity or servo loops. It should be placed before an associated Move2 icon. The Go1 icon can also be used to resume multi-axis coordinated motion that has been stopped.

Placing a Go1 icon opens the *Go Setup* window. The **Motion Type** radio button specifies the type of motion to start. Choosing **Non-Coord** as a **Motion Type** requires an entry in the **Axis** field specifying the control axis to start. The axis is specified by a valid integer constant, variable, global variable or an equivalent label. Specifying "-1" in the **Axis** field, for a single axis program, enables all the single axes assigned to the current task in the Axis2 icon.

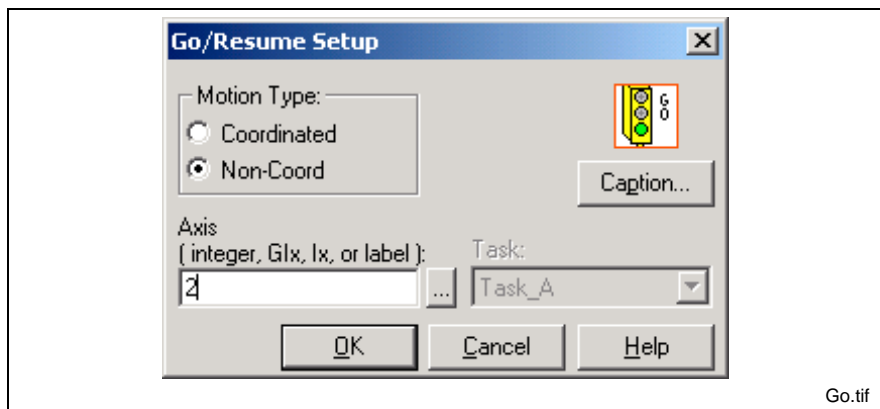


Fig. 3-72: Go/Resume Setup

Choosing **Coordinated** as a **Motion Type** resumes motion to a coordinated axis halted by a **Coordinated Stop**. The task containing the halt coordinated axis must be selected from the Task drop-down list. Aborted coordinated motion should not be simply resumed.

Note: In GPS6 firmware, the Go1 icon was used to start the single Virtual Master by placing a 0 in the **Axis** field. In GPP7/8/9 and GMP9 firmware, the two available Virtual Masters are set in motion using control registers.

Home



The Home icon is used to activate the drive's internal homing command for the specified axis in the **Axis to home** field. This is a single axis non-coordinated motion command. Placing a Home icon in a task or subroutine workspace automatically opens the *Homing Setup* window.

Note: The Home icon is used only for single-turn encoders. For multi-turn encoders, a move icon is used to move the axis to an absolute position, which will serve as the reference point for all moves.

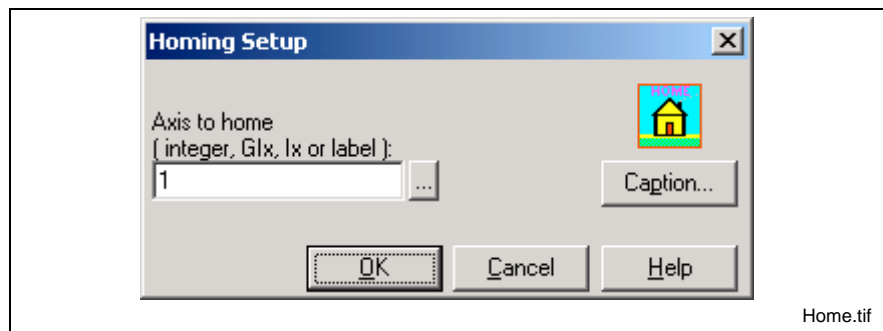


Fig. 3-73: Homing Setup

The axis to be home may be specified by a valid integer constant, variable (lx), global variable (Glx), or an equivalent label. Click the browse button to select an axis from the VM Data Table.

Once commanded to home, the drive homes the axis to the home position without further intervention from the control. Any errors in the drive's homing command are reported to the control. The program flow waits in this icon until the homing sequence is complete.

The Home icon uses the internal homing capability of the Bosch Rexroth's intelligent digital drive to perform the homing operation. For homing to occur, the homing parameters in the specified drive must have been setup before executing the Home icon. Refer to the relevant drive manual for details.

I_O (I/O Setup)



The I_O icon is used to control the state of I/O register bits. Placing an I_O icon in a task or subroutine workspace automatically opens an I/O Setup window.

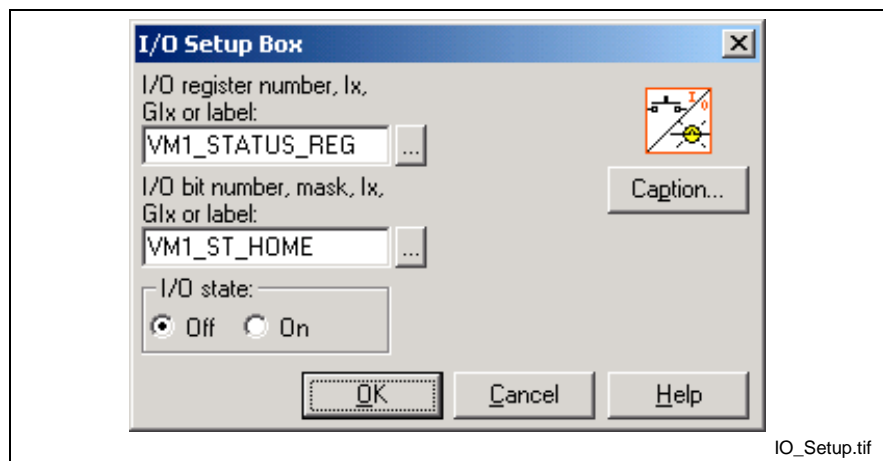


Fig. 3-74: I/O Setup

The **I/O register name** field specifies the target register number. The entry can be a number, an integer variable, or a defined register label.

The **I/O Bit number** field specifies the bit or bits to be controlled and permits entry of an integer constant or equivalent label, as an "and" mask for the target register. A single bit is specified by a number, an integer variable, or a defined bit label. Multiple bits in a single register are specified in hexadecimal by a bit mask (e.g., 0x21 would specify bits 5 and 0).

Multiple bits can be changed with a single icon by entering an I/O bit mask that specifies more than one bit (e.g. 0x21 or a label equivalent to the

desired mask). Clicking the browse button to the right of any field opens the VM Data Table.

The radio buttons for I/O state determine whether the target register bits, enabled by the I/O Bit mask, are cleared (logic zero, or off) or set (logic one, or on).

Clicking the browse button next to each field opens the VM Data Table which permits adding, editing or deleting register labels. The Register Labels window provides a scrolling list of default labels for the standard control system, axis, task, and digital drive I/O card control and status registers.

Join



A Join icon makes it possible to connect one line to another. VisualMotion icons have a maximum number of inputs; "join" overcomes this limitation by permitting many program flow paths to combine into a single path.

The Join icon is often the only method of completing a program flow, since VisualMotion cannot cross interconnecting lines. In addition, your program may require branching to several different calculations depending upon a certain condition. After the **Calc** icons you can use join icons to return to the main program flow before it enters the next icon.

Joint



The Joint icon is used for point-to-point movement and joint (elbow) positioning, typical to robotic motion. It changes one or more motor angles from the current set J0 to a new set J1 defined by a absolute(ABS[x]) point. This instruction is only for coordinated motion whose kinematics supports joint angles. It can only be used with a six axis robot with a second frame of reference (more than just x, y, and z).

A Joint move is an absolute point-to-point move, with only the endpoint of the move specified. It is the most efficient type of move because the path calculated by the path planner is optimized to minimize time. A Joint move uses the axis' maximum acceleration and deceleration rates, while line and circle coordinated motion commands use Path Maximum percentages (defined in Task parameters) and Maximum Acceleration and Deceleration rates (defined in Axis parameters). Rate limiting is based on the most efficient axis limiting without violating the axis.

The actual path taken to the specified point is not defined and may assume whatever form the path planner requires; however, once programmed, the path can be repeated.

The destination is specified as an entry in the absolute point table as an integer constant, variable, global variable or an equivalent label.

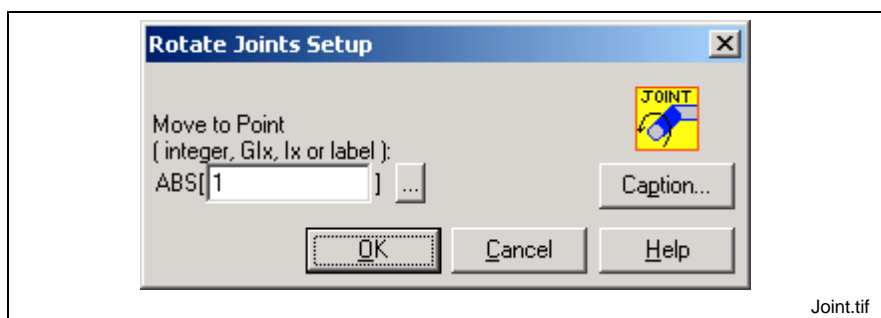


Fig. 3-75: Rotate Joints Setup

Line



All icons in each task, subroutine and event function must be connected. The line icon is used to draw a line indicating program flow from one icon to another. Clicking on the beginning icon surrounds the icon with a box. Clicking on the ending icon automatically draws a line from the first to the second icon, with an arrowhead indicating the direction of program flow.

Under some circumstances, VisualMotion may be unable to route a line and displays a *Connection could not be made, try connecting adjacent blocks* window. Lines may be manually routed by clicking adjacent empty squares on the invisible workspace grid from the first icon to the second. A manually placed line may not cross another line; attempting to do so displays an error box.

A line connecting two icons may be deleted by using the Scissors icon located next to the Line icon. Position the Scissors over the line and click the left mouse button.

Move2



The Move2 icon is used to program movement on any single non-coordinated axis from any task. The Move2 icon initiates motion only if the axis has been enabled previously with a "GO" icon. Placing a Move2 icon on a task or subroutine workspace automatically displays a *Single Axis Move Setup* window.

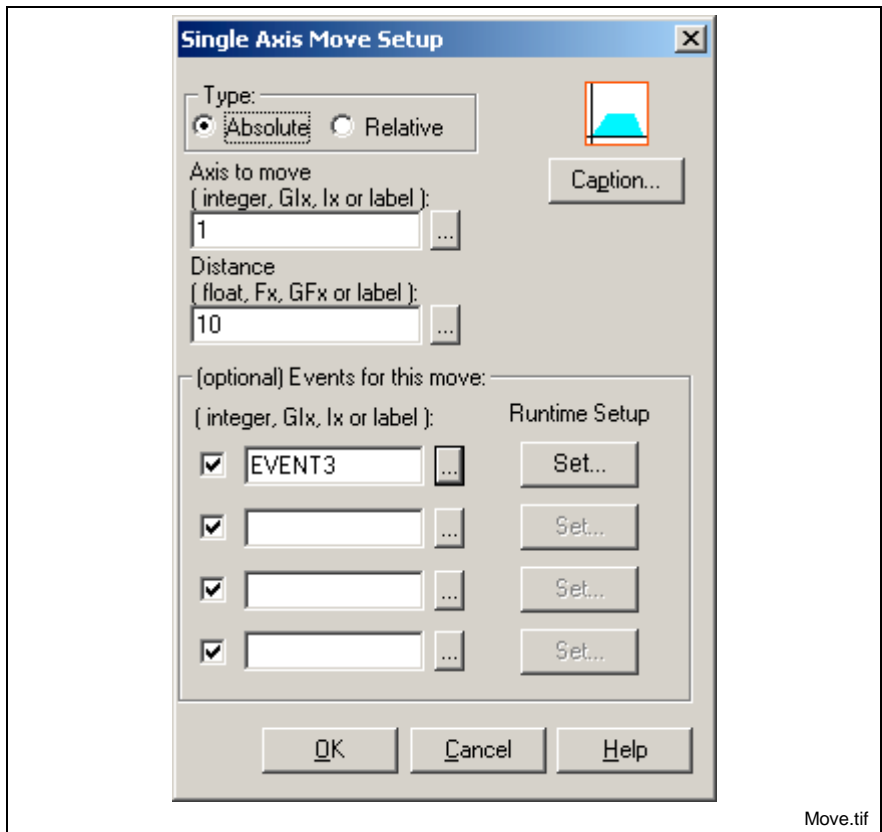


Fig. 3-76: Single Axis Move Setup

The **Type** radio buttons specify either an incremental move distance added to the current position (Relative) or a move to an absolute position (Absolute).

The **Axis to move** field specifies the axis to move. It accepts an integer number, constant or variable.

The **Distance** field specifies the incremental move distance or absolute target position. It accepts a float number, constant or variable.

Events

Events for single axis moves are armed using the Move2 icon. The user can configure up to 4 event triggers for each axis. The event triggers are enabled by placing a check in the checkbox and then clicking the browse button to the right of **EVT[]** field. The browse button opens that VM Data Table from where the user can selected a pre-configured event trigger or create one by right clicking and selecting **Add...**. Refer to the following figure for details.

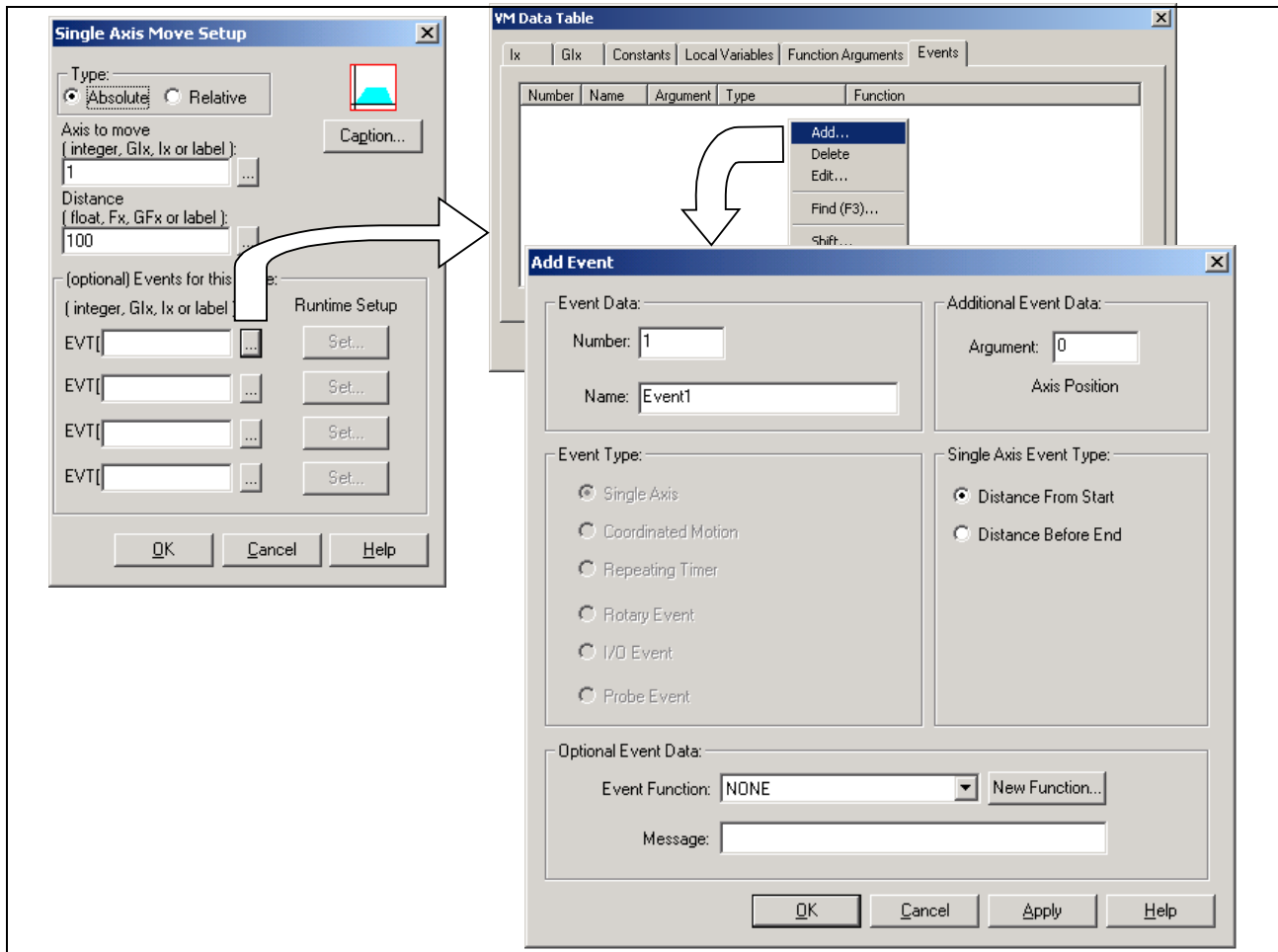


Fig. 3-77: Events in the Move Icon

Any event function assigned to an event trigger must be programmed Offline before running the program. The event types available for the "Move" icon are as follows:

- Single Axis Distance from Start
- Single Axis Distance from End

Runtime Setup

The **Runtime Setup** button is an advanced feature used to change the argument of an event trigger, the Event Function or both within the program flow during runtime. Refer to the *VisualMotion 9 Application manual*, for details.

Msg1



The Message icon is used to select a status or diagnostic message (up to 79 characters) at a specified point in the program flow. Placing a Msg1 icon on a task or subroutine workspace automatically displays the *Task Text Message Setup* window.

Messages are used to inform a user about the current state of the program through Task Parameters T-0-0122 and T-0-0123, or through the teach pendant. Systems using Direct ASCII Communication may obtain messages through the RS-232 port. Messages may also be sent to the top line of the pendant and to the serial ports. Messages are available after the icon is executed in the program and remain in effect until another message of the same type is executed.

Status messages tell a user or machine operator something about the ongoing process. At runtime, the current status and diagnostic messages can be viewed by selecting **Diagnosics** ⇒ **Tasks** from VisualMotion's main menu.

Diagnostic messages are typically used to provide information about the current state of the system, e.g., "412 No drives were found on ring." If an error occurs during task execution, this diagnostic message is overwritten with an error message.

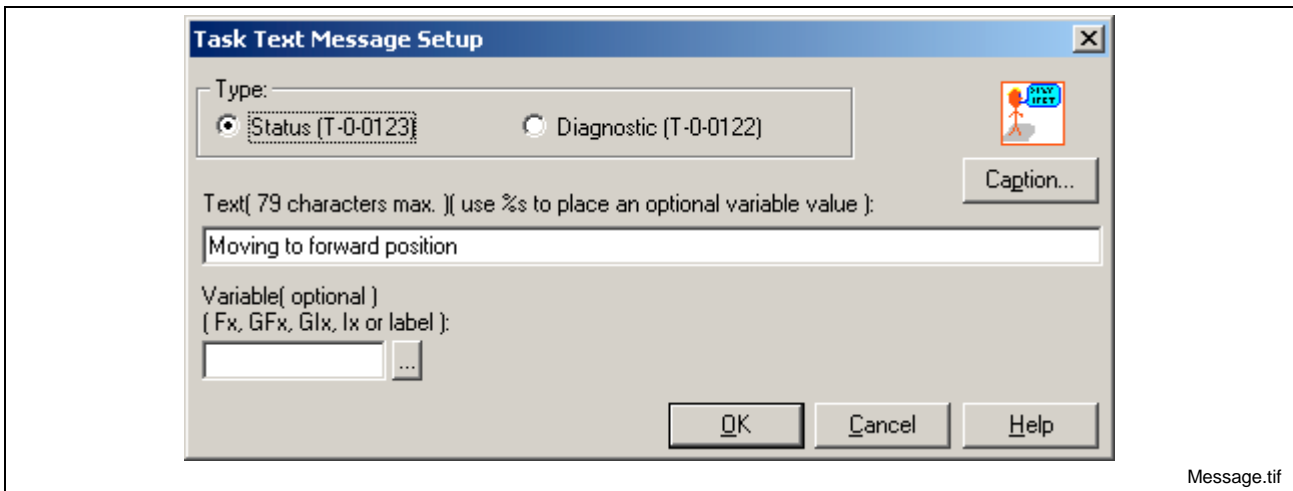


Fig. 3-78: Message Setup

An optional variable may also be displayed. This is useful for operator interface or debugging. A message may have one formatted variable in its string by using "%s" as a place holder for the variable entered in the **Variable (optional)** field.

Example Message = "The current count is %s." where %s equals variable I2 which has the label "current_count."

The displayed variable may also have a corresponding label. The VM Data Table is opened when clicking on the browse button to the right of the **Variable (optional)** field. It is also a good idea to copy the message into the Icon *Caption and Comment* window by clicking on the **Caption...** button. This message will appear when the cursor is held over the Msg1 icon in the program.

Param



The Param icon is used to transfer specified control or drive-related parameters or variables between the PC and an array of control variables at runtime. Transferring parameters to control-stored variables is the only way that programs can perform calculations and logical operations on parameter values.

Warning! Frequent changes of static drive parameters can cause premature failure of it's non-volatile memory.

The above warning message only applies to the following drives with all versions of firmware:

- ECODRIVE01
- DIAX04

It is possible to prevent damage to the EEPROM by changing the default setting of the drive parameter S-0-0269 (Buffer Mode) from 0 to 1. The change forces the SERCOS control to disable the parameter buffer on every power up sequence, limiting the number of times parameter changes are written to memory.

Note: No memory problem occurs in the following version of ECODRIVE03 firmwares:
ECODR3-SGP-01, ECODR3-SGP-03, ECODR3-SGP-20
ECODR3-SMT-02

Subsequently downloading the modified control variables permits dynamic modification of parameters during system operation. Placing a Param icon in a task or subroutine workspace automatically displays a *Parameter Transfer* window.



Fig. 3-79: Parameter to Variable Transfer

Example 1 To read the drive 1 modulo value (S-x-0103) into the variable "drive_1_modulo"

The radio buttons for **Transfer type** determine the direction of the transfer, from a specified control constant or variable to the parameter, or from the parameter to a control variable. The **Parameter type** radio buttons select from System, Axis, Task or Drive associated parameters.

Specify the parameter to be transferred in the **Parameter ID Number** field as an integer or equivalent label.

If the **Control** parameter type is selected, specify the source or target variable by entering an integer or float constant, variable (Ix or Fx), global variable (Glx or GFx), or an equivalent label in the appropriate data box.

If an **Axis** or **Drive** type parameter is selected, the additional field enabled requires a valid integer constant, variable (Ix), global variable (Glx), or an equivalent label specifying the axis or drive ID.

If the **Task** parameter type is selected, a pull-down pick list (with scroll buttons) permits selection of one of the four control tasks (A, B, C or D).

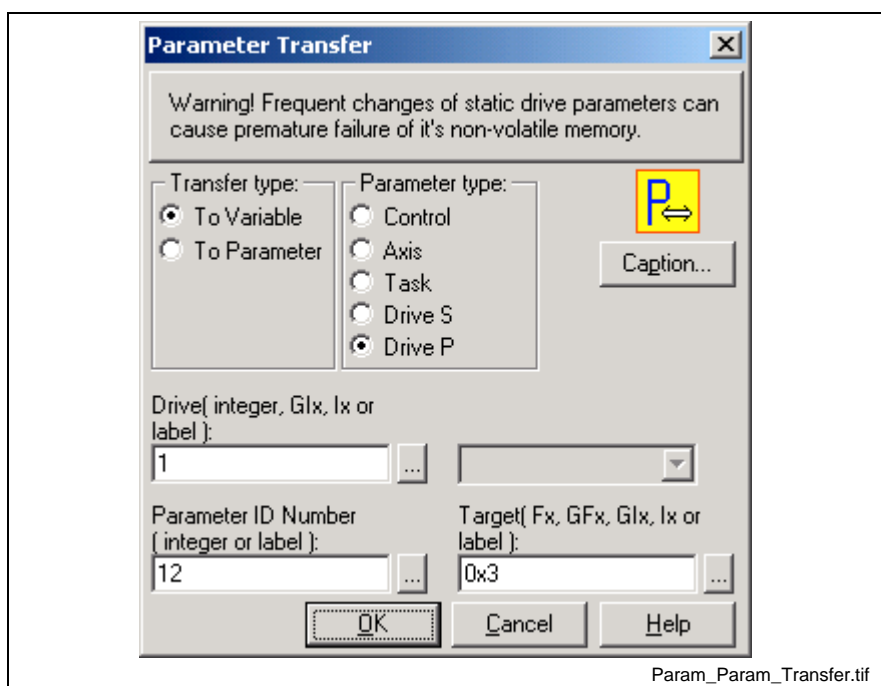


Fig. 3-80: Parameter to Parameter Transfer

Example To write to the drive 1 "Set Absolute Measuring Procedure Command" (P-x-0012)

Clicking the browse button to the right of any field opens the VM Data Table.

Note: Certain control parameters are read-only and cannot be changed, while others may be modified only with the control system in "Parameter" mode.

Note: To write or read to the drive parameter, select the type (S or P parameters), then enter the number in the Parameter ID Number. No offset is required for P parameters.

ParamBit



The ParamBit icon is used to change the state of a specific bit(s) of a binary parameter at runtime. The state of the bit is changed by creating a mask and writing a value of either 0 or 1. A bit mask can be created for single or multiple bits.

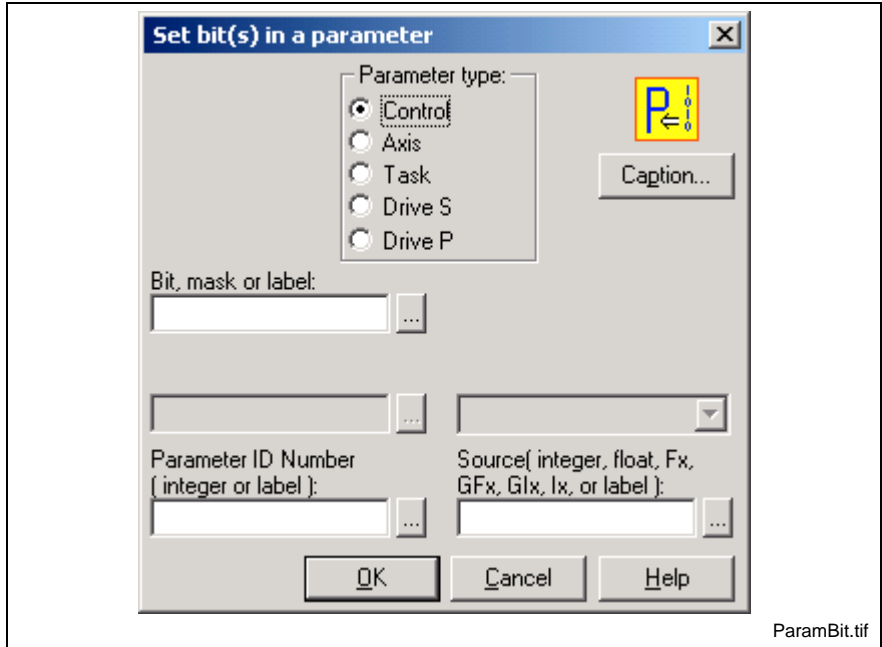


Fig. 3-81: Parameter Bit Icon

Note: Any read/write binary parameter can be used in the ParamBit icon. An error is issued if a read-only binary parameter is used.

Different entry fields in the *Set bit(s) in a parameter* window are displayed based on the selected **Parameter type** radio button.

Entry Fields	Function	Used with ...
Bit, mask or label	Enter a bit position value for single or multiple bit masking.	All Parameter Types
Axis	Enter an axis number as an integer, Glx, lx or label	Axis Parameter Type
Drive	Enter a drive number as an integer, Glx, lx or label	Drive S and Drive P SERCOS Parameter Types
Task ID	Select the user program control task for the drop-down list	Task Parameter Type
Parameter ID Number	Enter the parameter's significant numbers as an integer or label. For example: C-0-0010, enter a 10	All Parameter Types
Source	Enter a value of 0 or 1 as an integer, float, Fx, GFx, Glx, lx or label	All Parameter Types

Table 3-25: ParamBit Icon Entry Fields

Single Bit Masking

A single bit mask is created by entering the bit position (1-16) in the **Bit, mask or label** field. A parameter is selected by clicking on a radio button for the desired parameter type and entering its number in the **Parameter ID Number** field. The state of the bit is modified by entering a 0 or 1 in the **Source** field. The following figure illustrates the ParamBit icon used to change the state of bit 13 for control parameter C-0-0010 from its current state to 1.

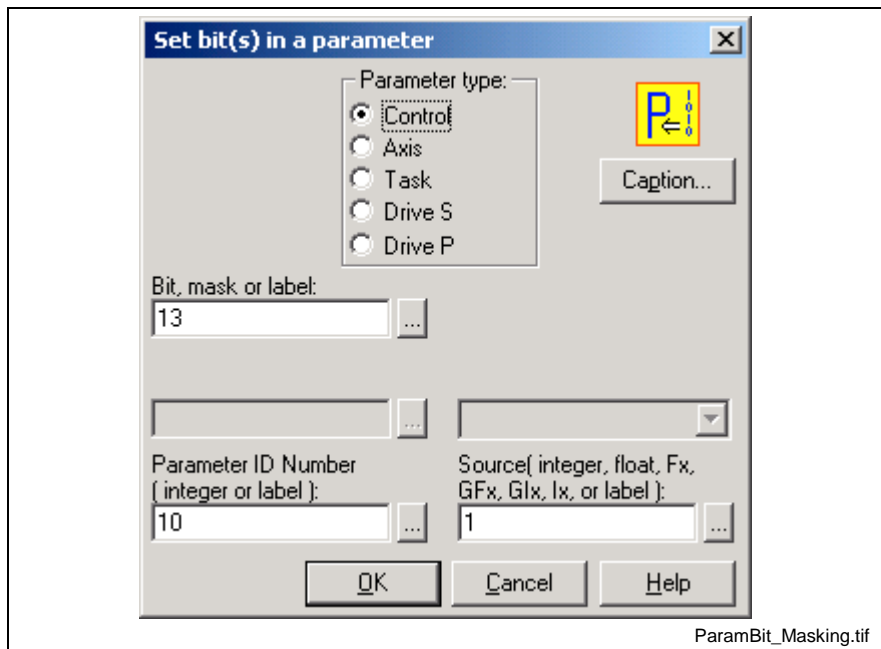


Fig. 3-82: Parameter Bit Setup for Single Bit Masking

Multiple Bit Masking

A multiple bit mask is created by entering a hexadecimal value (i.e., 0xF, 0x100) or label equivalent for the bits in the **Bit, mask or label** field.

Example:

Here's the hexadecimal value for the first 4 bits of a binary parameter.

000000000001111 (binary) = 0xF (hexadecimal)

The "0x" prefix must precede all hexadecimal values. If a decimal "15" is used as a mask, only bit 15 will be modified. If "0x15" is used, the following bits are modified: 000000000010101. Any scientific calculator can be used to convert between binary and hexadecimal.

A parameter is selected by clicking on the desired **Parameter type** radio button and entering its number in the **Parameter ID Number** field. The states of the bits are modified by entering a 0 or 1 in the **Source** field.

Note: The source field affects all bits in the mask. The bits in the mask cannot be selectively modified.

The following figure illustrates the ParamBit icon used to change the states of bits 1, 2, 3 and 4 for control parameter C-0-0010 from its current state to 1.

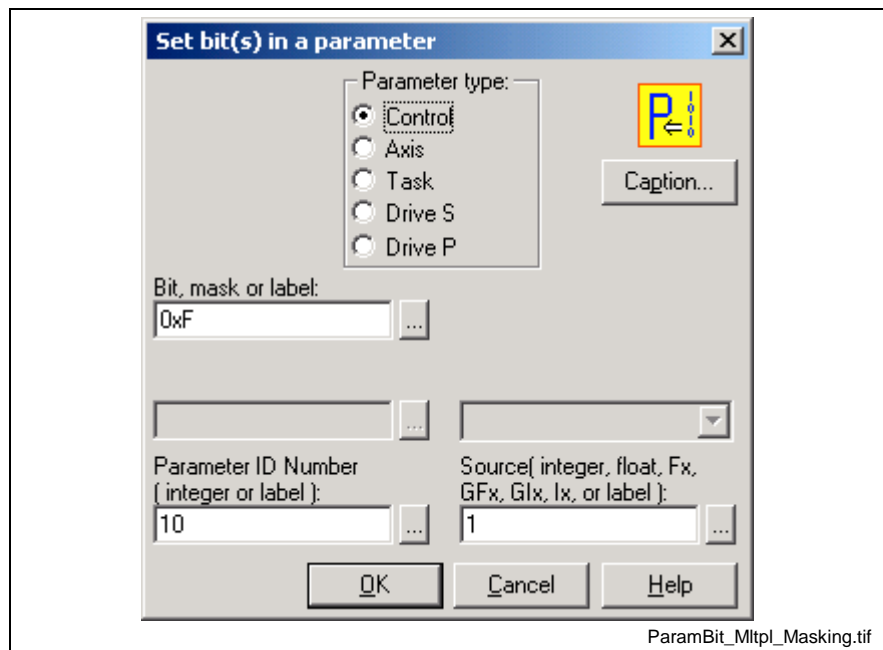


Fig. 3-83: Parameter Bit Setup for Multiple Bit Masking

Path



The Path icon is used to set up multi-axis coordinated straight line motion. Placing a Path icon on a task or subroutine workspace automatically displays a *Coordinated Line Setup* window.

Motion may be Absolute or Relative and is defined by the two endpoints of the line of motion. Points are specified by an index into the point table using an integer constant, variable, global variable or an equivalent label. Refer to the **Calc** icon. The second point, or target point, must be a point on the point table. The first point is the current position, which can be anywhere.

An absolute move begins from the endpoint of the previous path segment, or current position if the system is halted, and terminates at the absolute point specified.

The relative move begins at the endpoint of the previous path segment, or current position if the system is halted, and terminates at the relative offset point specified.

Clicking the browse button to the right of the **ABS** or **REL** field opens the VM Data Table.

A Go1 icon is not required when using the Path icon. When executed, the motion will immediately be sent to the path planner. Stepping to the next icon will take place immediately.

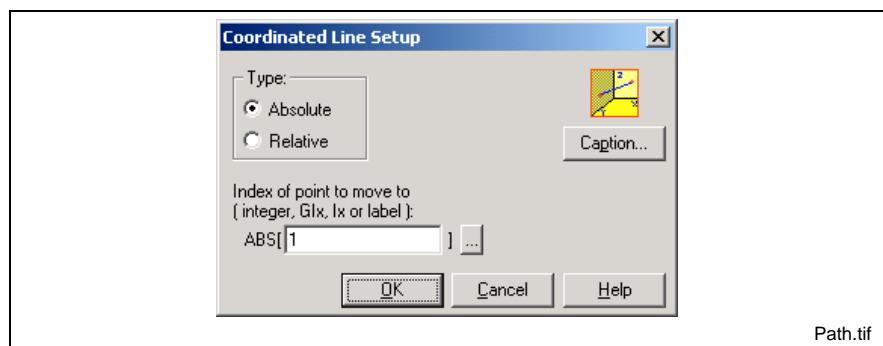


Fig. 3-84: Path Setup

Event Types for Coordinated Motion

Event triggers for coordinated motion are associated with the points table that defines the path. Up to 4 event triggers can be configured for each point in a coordinated motion path. The following table lists the event types available for the Path icon along with their arming behavior, maximum number and description.

Event Type	Auto Rearming	Maximum Number	Description
Percent from Start	No	4 events per point	A position-based event that executes an event function at the set percentage of the overall travel from the start.
Percent before End	No	4 events per point	A position-based event that executes an event function at the set percentage of the overall travel before the end.
Time from Start	No	4 events per point	A time-based event that executes an event function at a set time from the start of the move.
Time before End	No	4 events per point	A time-based event that executes an event function at a set time before the end of the move.

Table 3-26: Coordinated Motion Event Types

The following figure illustrates that different windows that are displayed when adding an event trigger to the coordinated motion point. Refer to Adding Events on page 3-82 for an example.

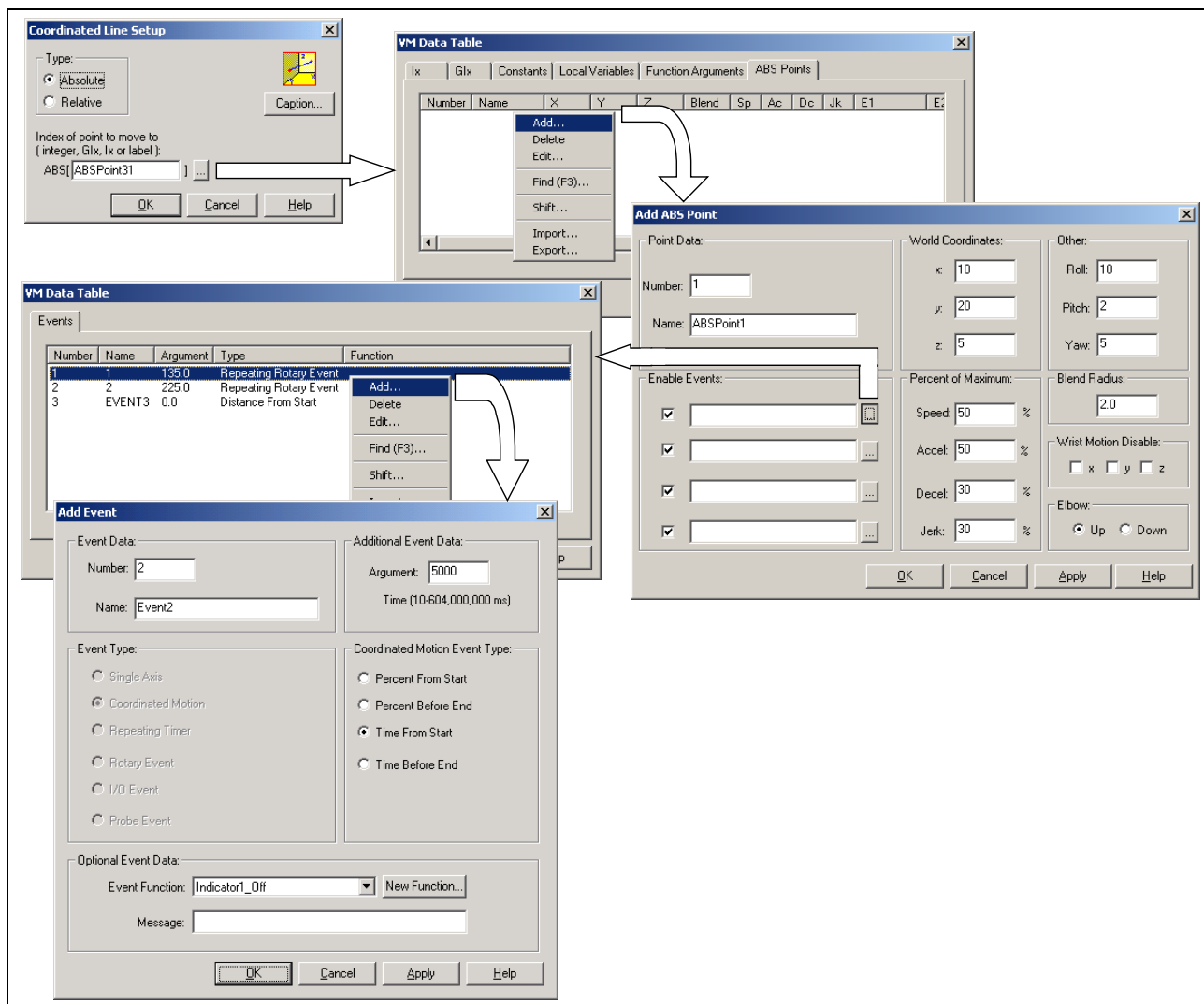


Fig. 3-85: Path Setup

Changing Arguments during Runtime

Any argument, event function or both can be modified during runtime by placing a Calc icon in the desired program flow location. Refer to the Calc2 icon, Events section on page 3-27 for the correct syntax when defining an argument for a coordinated motion point.

Note: The Event2 icon cannot be used to change the argument or event function of a coordinated motion point.

PID1



The PID1 icon can only be placed in the Initialization task and is used to initialize and install a proportional control loop on the control, up to 32 loops are possible. The location of the PID1 icon in the program is not important, it is only executed at program activation. The program flow will be slowed if this icon is continuously processed, so it is desirable to execute this icon only once per PID loop. The PID functions load on GPP resources is included in Load Due To I/O control parameter C-0-0202.

Loop Time is selectable in multiple of 8ms from 8 to 152 ms.

Set point can be a program or global variable (Fx, GFx, Glx, lx, or label), or signed or unsigned register. Internally, the value is converted to a float. It's offset is then subtracted and the resultant multiplied by it's scalar.

Feedback and **output** may be a program or global variable (Fx, GFx, Glx, lx, or label), a signed or unsigned register, or an axis parameter. Three axis parameters (position, velocity, and torque) are selectable in a list box. Other axis parameters can be added by selecting optional1 or optional2 and adding the axis parameter number.

The optional axis parameter data is added to the cyclic SERCOS data for update every SERCOS cycle time. If using the optional axis parameters, care must be taken to avoid using them for other purposes. Internally, the value is converted to float. It's offset is then subtracted and the resultant multiplied by it's scalar. A list of valid axis parameters can be viewed in drive parameter S-0-0188 "List of configurable Data in the MDT."

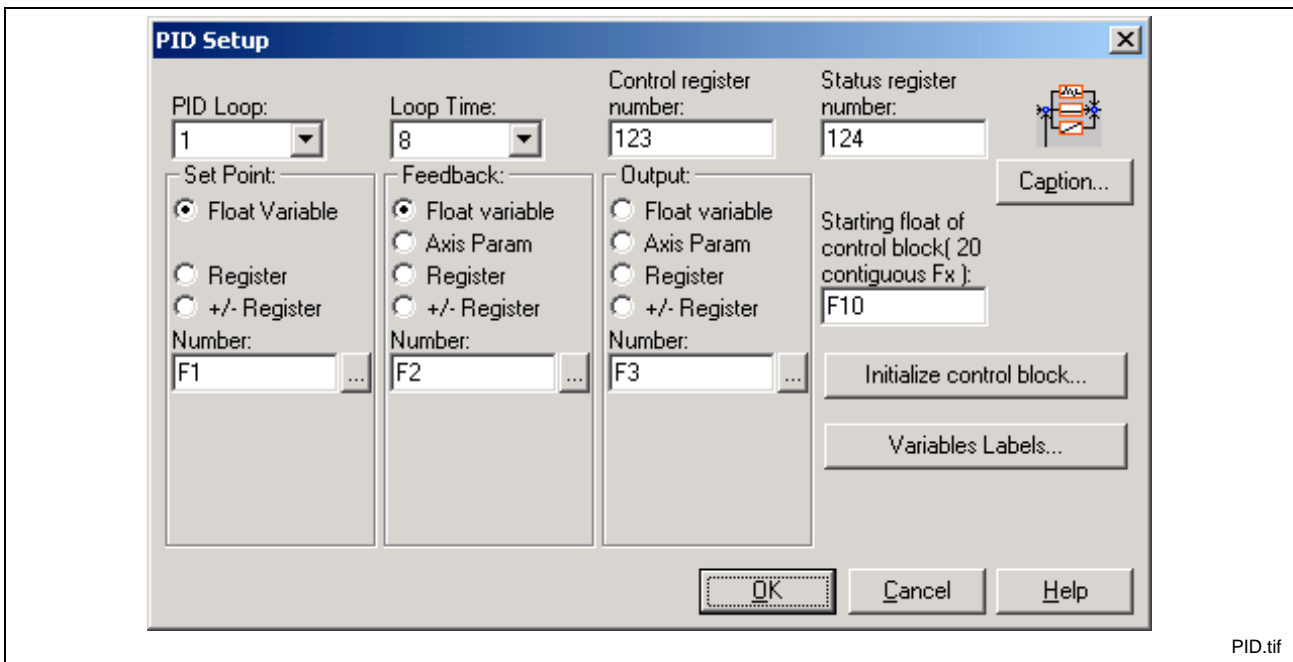


Fig. 3-86: PID Setup

The Starting float of the control block is the first variable used to store the initialization values. The remaining variables are sequentially assigned to each control block value. (F11, F12, F13, F14.....)

The **control block** is a group of 20 floats used for:

F(x)	Command scalar value	default 1.0.
F(x+1)	Command bias value	default 0.0.
F(x+2)	Feedback scalar value	default 1.0.
F(x+3)	Feedback bias value	default 0.0.
F(x+4)	Kp value	default 1.0.
F(x+5)	Ki value	default 0.0.
...	Kd value	default 0.0.
	Ki limit value	default 0.0.
	Minimum output value	default -10.0.
	Maximum output value	default 10.0.
	Preset value	default 0.0.
	Output scalar value	default 1.0.
F(x+12)	Output bias value	default 0.0.

Control block values, default or user selected, are added at compile time.

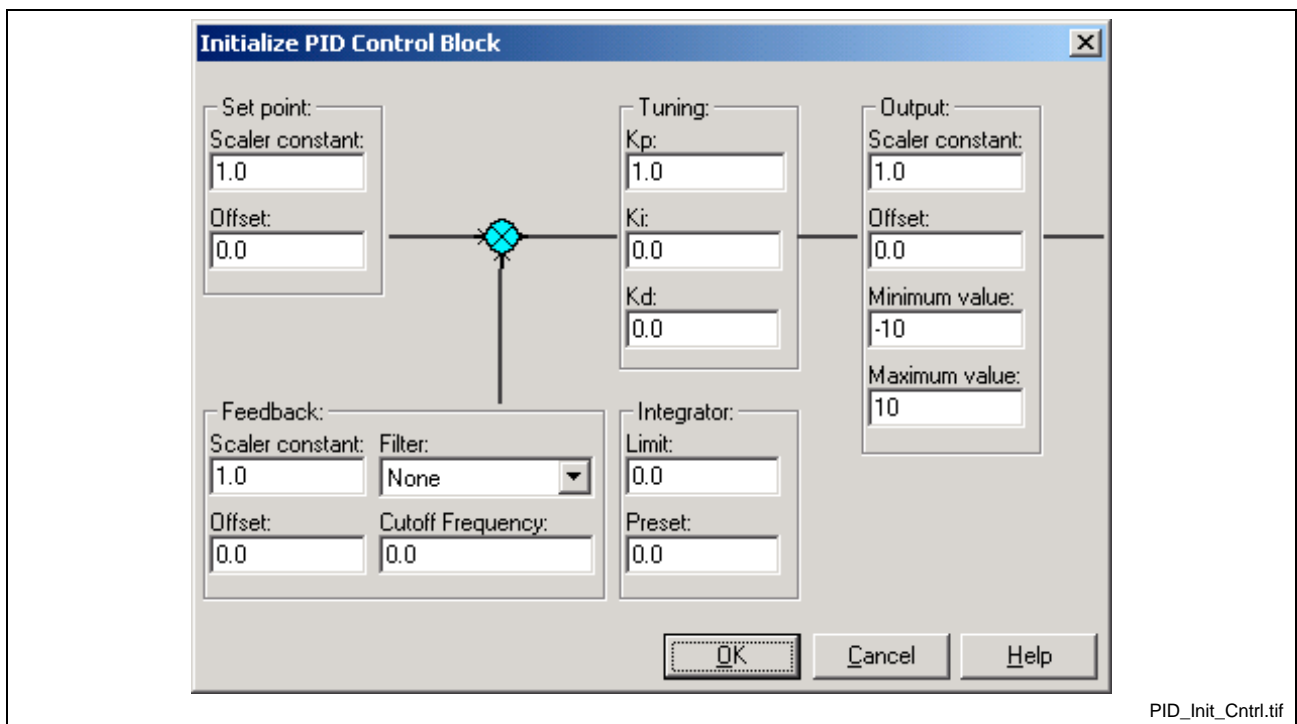


Fig. 3-87: Initialize PID Control Block

Feedback:

Digital filtering is available for PID loops and Real Masters.

Filter Type:

- None
- First order low-pass, $G(s)=1/(s+1)$
- Second order low-pass, $G(s)=1/(s^2 +2s +1)$
- Third order low-pass, $G(s)=1/(s^3+3s^2+3s+1)$
- Second order Butterworth, $G(s)=1/(s^2 +2^{1/2}s +1)$
- Third order Butterworth, $G(s)=1/(s^3+2s^2+2s+1)$
- Modified 2nd order low-pass with velocity ramp tracking,
 $G(s)=(2s+1)/(s^2 +2s +1)$
- Modified 3rd order low-pass with accel ramp tracking,
 $G(s)=(3s^2+3s+1)/(s^3+3s^2+3s+1)$

Cutoff Frequency(float):

When a filter type is chosen, a cutoff frequency for the filter must be entered. The cutoff frequency is the frequency where the signal is reduced by 3db. When set to 0 the filter is disabled.

To ensure a stable system, use the following calculation when entering a value for the Digital Filter Cutoff Frequency:

$$\text{Cutoff Frequency} \leq \frac{1}{2 * \text{Sampling Rate(sec.)}}$$

The sampling rate for a PID loop is the set PID Loop Time, entered in seconds.

Example: For a 8 ms PID Loop Time, the cutoff frequency is calculated as follows:

$$\text{Cutoff Frequency} \leq \frac{1}{2 * 0.008} = 62.5 \text{ Hz}$$

The following figure illustrates the frequency vs. degrees for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

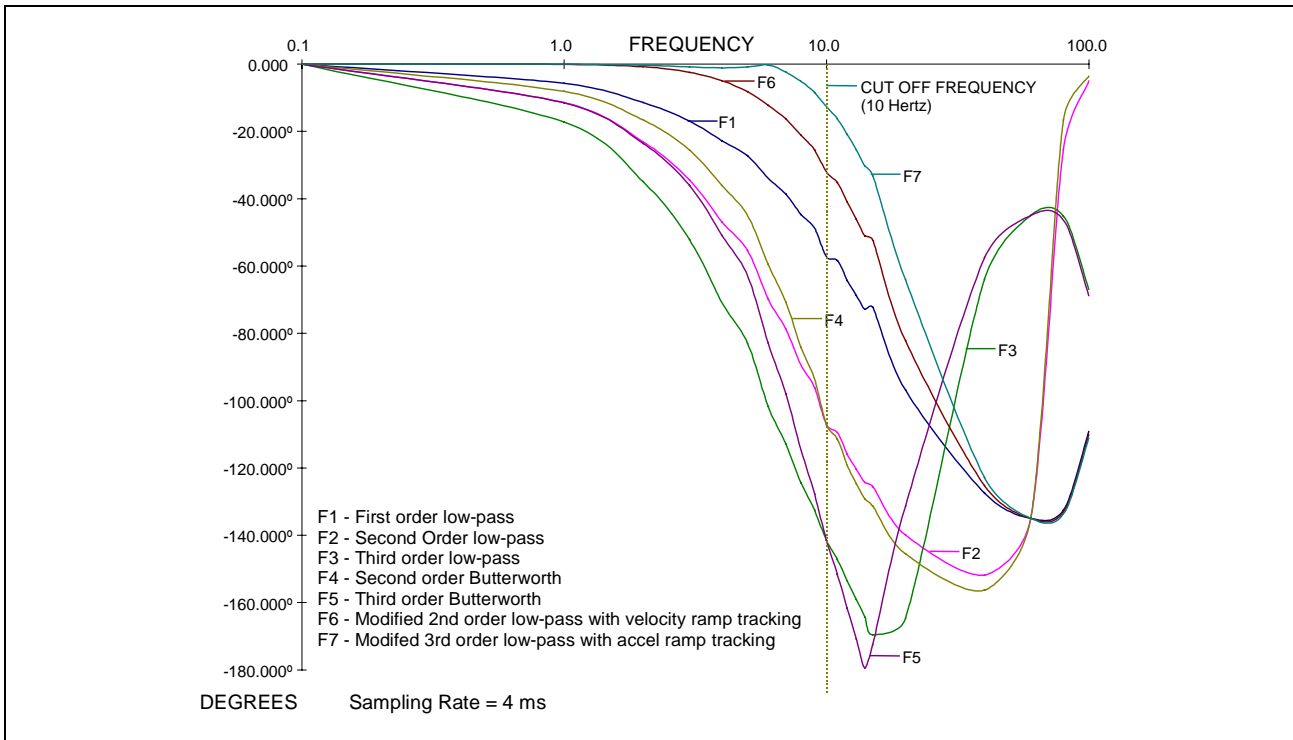


Fig. 3-88: Frequency vs. Degree Filter Chart

The following figure illustrates the gain vs. frequency for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

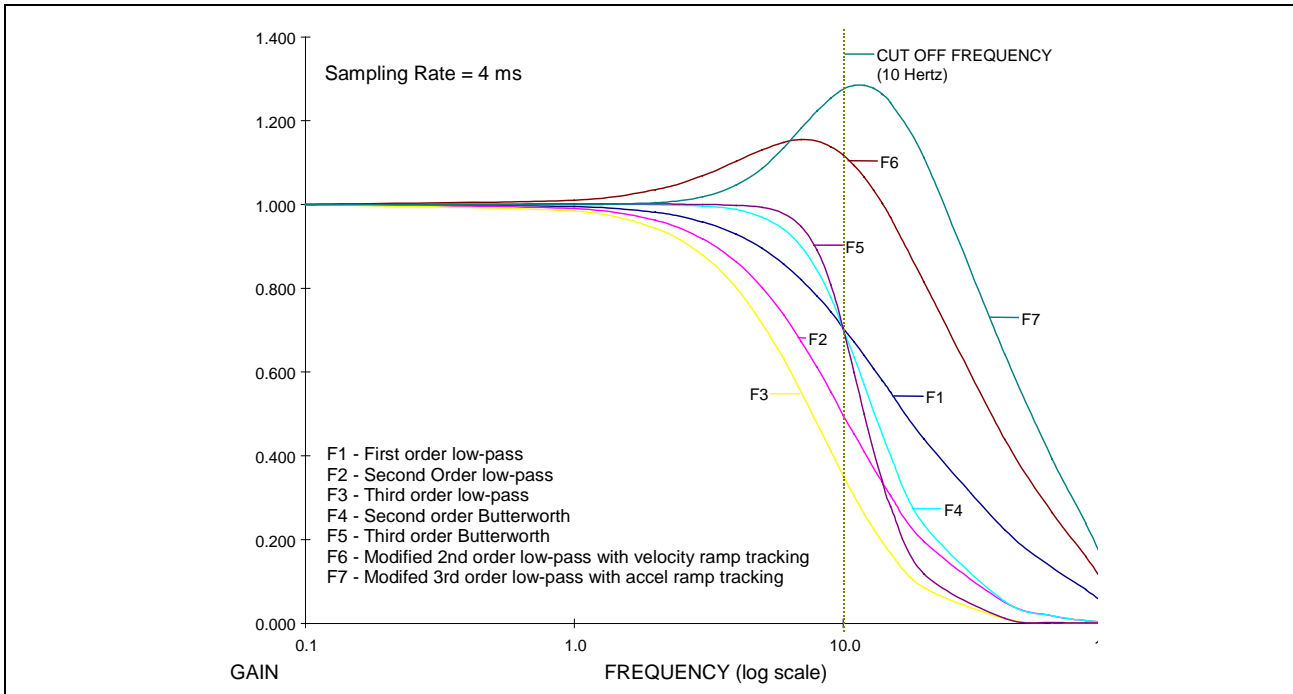


Fig. 3-89: Gain vs. Frequency Filter Chart

When a filter is chosen, the cutoff frequency for the filter must be entered. For example, the cutoff frequency for the First order low-pass filter is the frequency where the signal is reduced 3db [$.707$ gain, $db=20*\log(\text{gain})$].

Position



The Position icon is used to obtain the current position of one of the tasks from the path planner. Once this position is read, the value is stored in an absolute point table. This is useful when operating the system in a teach mode without a Teach Pendant. It also could be used to check axis position while running a program. For instance, a loop could be setup whereby the path position is constantly obtained and compared to a target position. When the path position matches the target position, a bit is toggled. The toggle causes a new branch test result, which alters program flow.

Once this icon captures the current position into a point, **Calc** icons can be used to copy the attributes of these points to new ones. Single axis motion can be created from coordinated motion data and offset or speed change adjustments can be made automatically.

Placing a Position icon on a task or subroutine workspace automatically displays a *Get Path Position* window. Four control tasks (A-D) can be selected from the Task menu. The absolute point table destination for the position data may be specified by an integer constant, variable, global variable or an equivalent label. Clicking the browse button to the right of the **Index of point** field opens the VM Data Table.

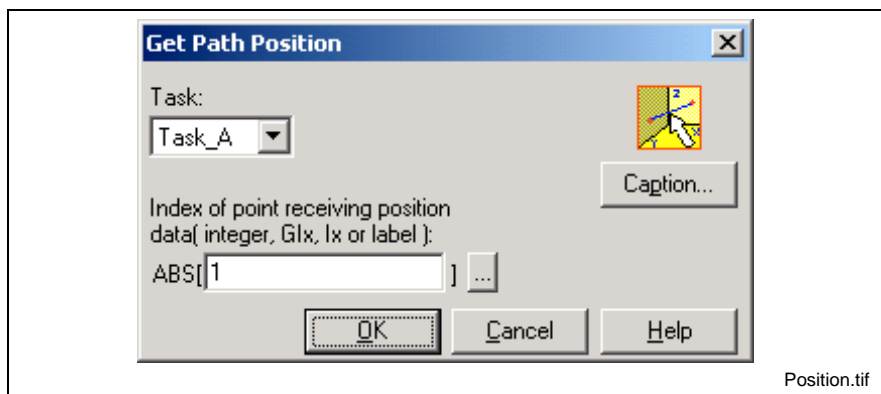


Fig. 3-90: Get Path Position

PrmBit



The PrmBit icon is used to initialize the state of a specific bit(s) of a binary control, axis, task, or drive parameter at program activation. The state of the bit is changed by creating a mask and writing a value of either 0 or 1. A bit mask can be created for single or multiple bits. Refer to the *ParamBit* icon on page 3-96 for details.

PrmInt



The PrmInt icon is used to initialize a control, axis, task, or drive parameter with a specific value at program activation.

Note: Any read/write parameter can be used in the PrmInt icon. An error is issued if a read-only parameter is used.

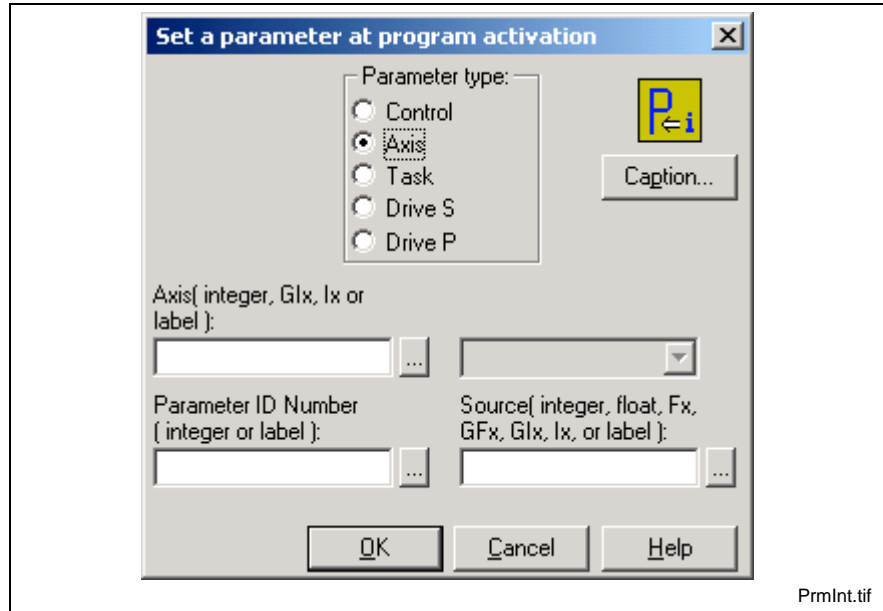


Fig. 3-91: Parameter Initialize

Different entry fields in the *Parameter Initialize* window are displayed based on the Parameter type radio button selected.

Entry Fields	Function	Used with ...
Axis	Enter an axis number as an integer, Glx, lx or label	Axis Parameter Type
Drive	Enter a drive number as an integer, Glx, lx or label	Drive S and Drive P SERCOS Parameter Types
Task ID	Select the user program control task for the drop-down list	Task Parameter Type
Parameter ID Number	Enter the parameter's significant numbers as an integer or label. For example: C-0-0010, enter a 10	All Parameter Types
Source	Enter the specific value for the parameter as an integer, float, Fx, GFx, Glx, lx or label	All Parameter Types

Table 3-27: PrmInt Icon Entry Fields

A parameter is selected by clicking on the desired parameter type radio button and entering it's number in the "*Parameter ID Number*" field. Enter the appropriate axis, task or drive number.

Selecting "*Axis*" or "*Drive*" enables a data entry field, permitting entry of an integer constant, variable (lx), global variable (Glx) or equivalent label specifying the axis or drive.

Selecting "*Task*" enables a drop-down list with selections for the four control tasks.

The "Source" data entry field specifies the value to be written to the parameter. The value type specified must match the parameter type.

For Example:

If a "Drive S" parameter type is selected, the value placed in the "Drive" data entry field must be of the same data type. For Drive, an integer Glx, lx or label must be used and not a different data type (such as Fx or a label created for a float).

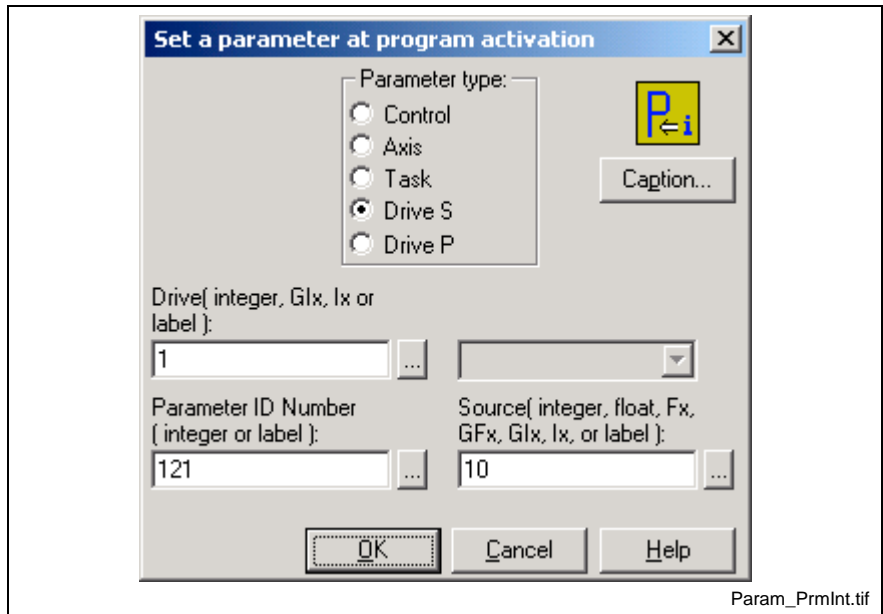


Fig. 3-92: Setting a Parameter at Program Activation

Example: *The gear ratio input revolutions of drive 1 is initialized to 10 at program activation.*

Ratio



The Ratio icon is used to set the ratio between two axes in a master/slave relationship, as when a gantry robot has a motor on each side of a supporting circular track. It can be used in Tasks which also contain coordinated axes.

The Ratio icon may also be used to link several axes to the same master axis or to chain several drives together. For example: if drive 1 is master to drive 2, drive 3 can be made a slave to drive 2, thereby linking drive 3 to drive 1 through drive 2. Note that the response time of drives chained in this manner is additive, at least one SERCOS cycle (approximately 2ms) must occur between each master to slave link.

Placing a Ratio icon on a task or subroutine workspace automatically displays a *Ratioed Axis Adjust* window. The Master and Slave Axes should already be assigned in the Axis2 icon as Ratioed axes. Refer to the **Axis Icon** for details.

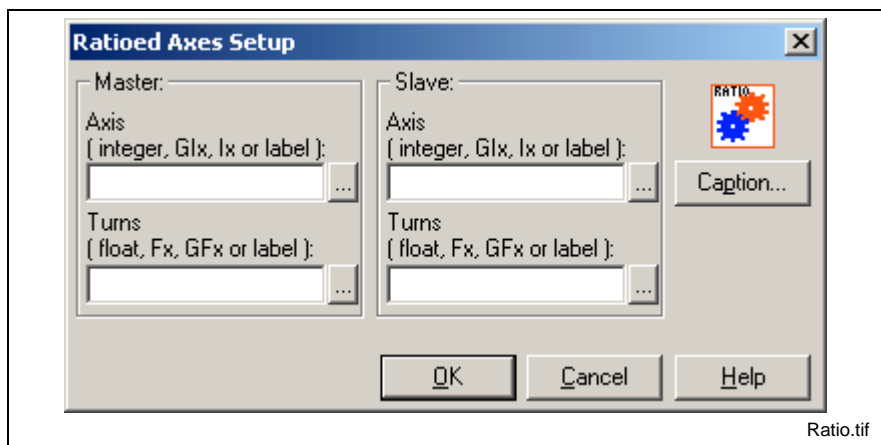


Fig. 3-93: Ratio Setup

The **Master** and **Slave** axes are selected by entering an integer constant, variable, global variable or an equivalent label in the **Axis** data fields.

The VM Data Table is available for all four fields. One axis is selected as a master axis. A slaved axis must not be assigned to any task other than the task containing the master axis.

Rotation of the master axis controls the proportional rotation of the selected slave axis according to the formula:

$$\text{Slave axis velocity} = \text{Master axis velocity} \times (\text{Slave ratio factor} \div \text{Master ratio factor})$$

The **Master** and **Slave Turns** fields permit simple entry of the ratio between the axes. The ratio factors may be a float constant, variable, global variable or an equivalent label. Individual data boxes for master and slave eliminate the need to normalize the ratio. For example, simply entering the number of teeth on each of two meshed gears allows VisualMotion to calculate the necessary coefficient. Each factor is in float format and is normalized before the division operation. This insures that the calculation maintains maximum precision with repeating decimals such as 2/3.

Entering the axis ratio factors using the Ratio icon automatically updates the master factor parameter A-0-0031 and slave factor parameter A-0-0032.

By default, the slave axis is maintained in the drive's position loop mode at all times, even when the user program is not running. The drive's shaft position remains locked to the master axis within the torque limits of the drive.

Reg (Register Transfer)



The Reg. icon is used to transfer data between the I/O registers and either the integer variable or global integer variable table. One to 1024 registers (16-bits each) may be transferred with a single command. Placing a Reg. icon on a task or subroutine workspace automatically displays a *Register Transfer* window.

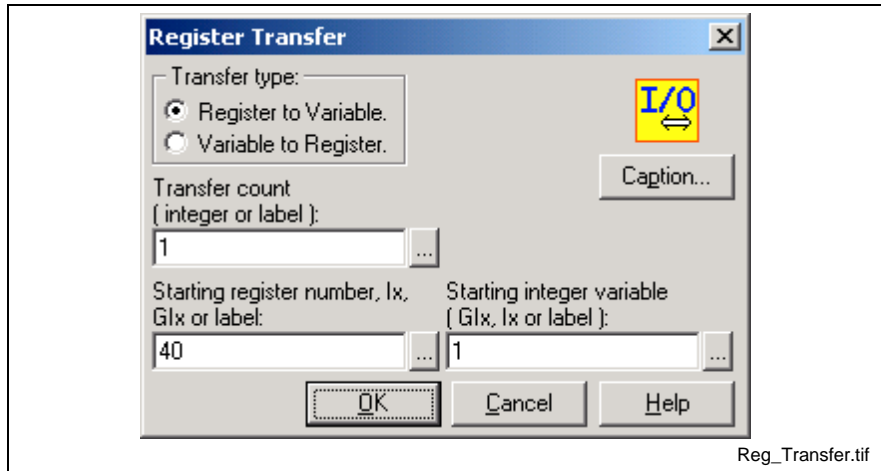


Fig. 3-94: Register Transfer

Example This transfers a 15 bit DEA card input register to a variable.

Transfer type selects the direction of data transfer between the I/O register(s) and a control integer variable table. Transfer count specifies the number of consecutive 16-bit words to be transferred. **Starting register number** specifies the base, or lowest address of the source for the start of the transfer. **Starting integer variable** specifies the base address of the target of the transfer.

The **Transfer count**, Starting register number and starting integer variable must be an integer constant, variable (Ix), global variable (GIx), or an equivalent label.

Scissors



The Scissors icon is used to delete a line between two icons. Select the Scissors on the palette, position the tip of the Scissors cursor over the line to delete, press the left mouse button to delete.

Start1



All four control tasks A-D, initialization task, subroutines and event functions must begin with a Start1 icon. The Start1 icon indicates the beginning of program flow to the control compiler and can also be used to declare function arguments and local variables.

Note: The **Define Function Arguments...** button is only available for subroutines (initialization and standard).
The **Properties...** button is only available for Task A-D.

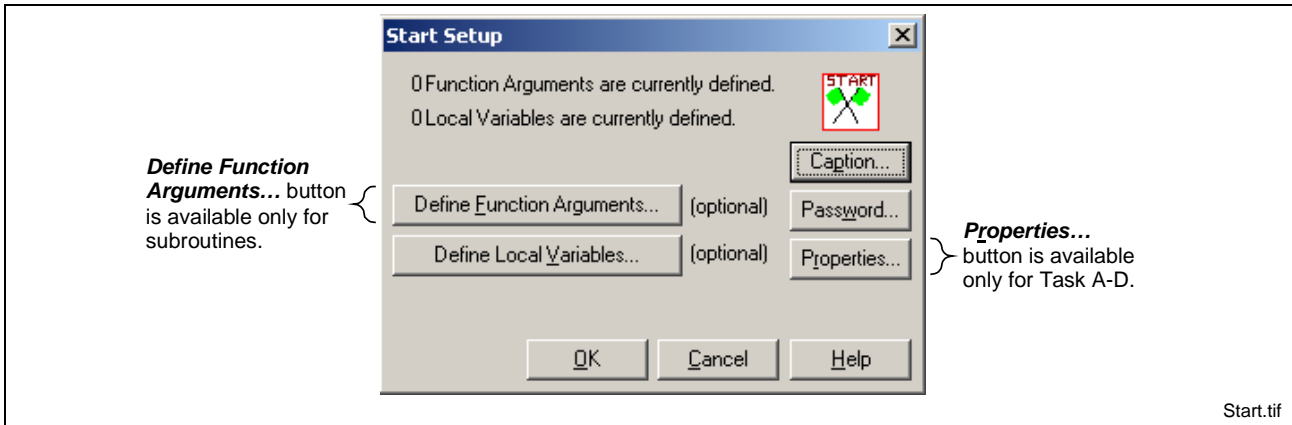


Fig. 3-95: Start Setup

Define Function Arguments

Function arguments are stack-based variables whose values exist only while in the subroutine (initialization or standard) where they are declared. The values of stack-based variables are placed in a memory location shared by function arguments and local variables. The total number of combined function arguments and local variables in a subroutine is limited to 16.

Selecting the **Define Function Argument...** button allows the user to define up to 5 function arguments of type **Float**, **Integer**, **ABS Index** and **REL Index**. Remember that the only thing that is being defined is the **Name** and **Type** for the function argument. The value that is passed to the defined function argument comes from the Sub1 icon of the calling task or subroutine.

Note: Duplicate function argument labels are not allowed within the same subroutine. A compiler error will be issued when the program is compiled. It is also recommended that the programmer not use VisualMotion keywords when naming function arguments. Refer to Using VisualMotion Keywords as Names on page 2-29 for details.

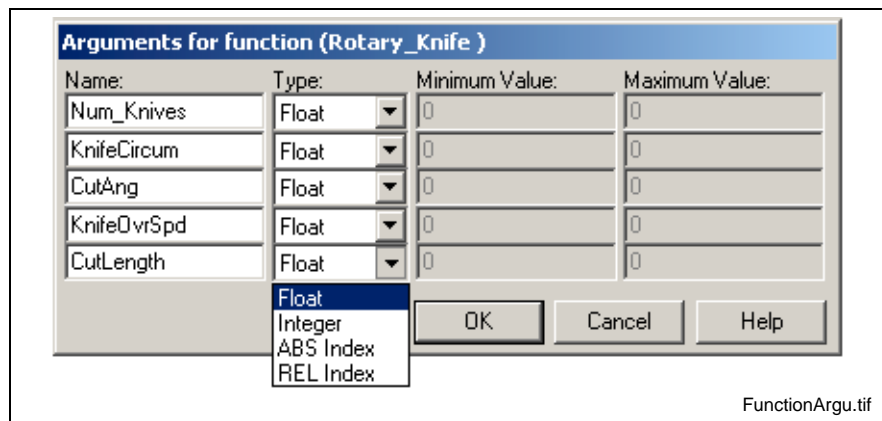


Fig. 3-96: Function Arguments

Function Argument Types

The available data types for function arguments are Float, Integer, ABS Index and REL Index. When **Float** and **Integer** are defined, the value passed by the caller (using the Sub1 icon) is lost when the subroutine ends.

Note: VisualMotion allows a single function argument value to be returned to the calling task or subroutine when using the Optional Return Value feature in the Sub1 and Finish icons. Refer to Optional Function Arguments (Sub1) on page 3-116 for details.

When **ABS Index** or **REL Index** are defined, the value that is passed is the index number of the absolute or relative points table and not an actual point value. The index number can be used within the subroutine and will not be saved when the subroutine ends. However, if any point data is modified, using an Calc icon, the value for that point will be modified in the project, even after the subroutine ends.

Define Local Variables

Local variables are stack-based variables whose values exist only while in the subroutine (initialization or standard) and/or event function where they are declared. Local variables are used within the subroutine for local data only and do not exist outside the subroutine. Local variables are useful for temporary results within the subroutine and their values are lost when the subroutine ends.

The total number of combined local variables and function arguments in a subroutine is limited to 16. The maximum number of local variables is dependent on the number of defined function arguments. If a maximum of 5 function arguments are defined, then only 11 local variables can be defined.

Notice that in the following figure, only 11 local variables are available, the rest of grayed out. This is because 5 function arguments were defined in Fig. 3-96 for the same subroutine.

Note: Duplicate local variable labels are not allowed within the same subroutine. A compiler error will be issued when the program is compiled.

It is also recommended that the programmer not use VisualMotion keywords when naming local variables. Refer to Using VisualMotion Keywords as Names on page 2-35 for details.

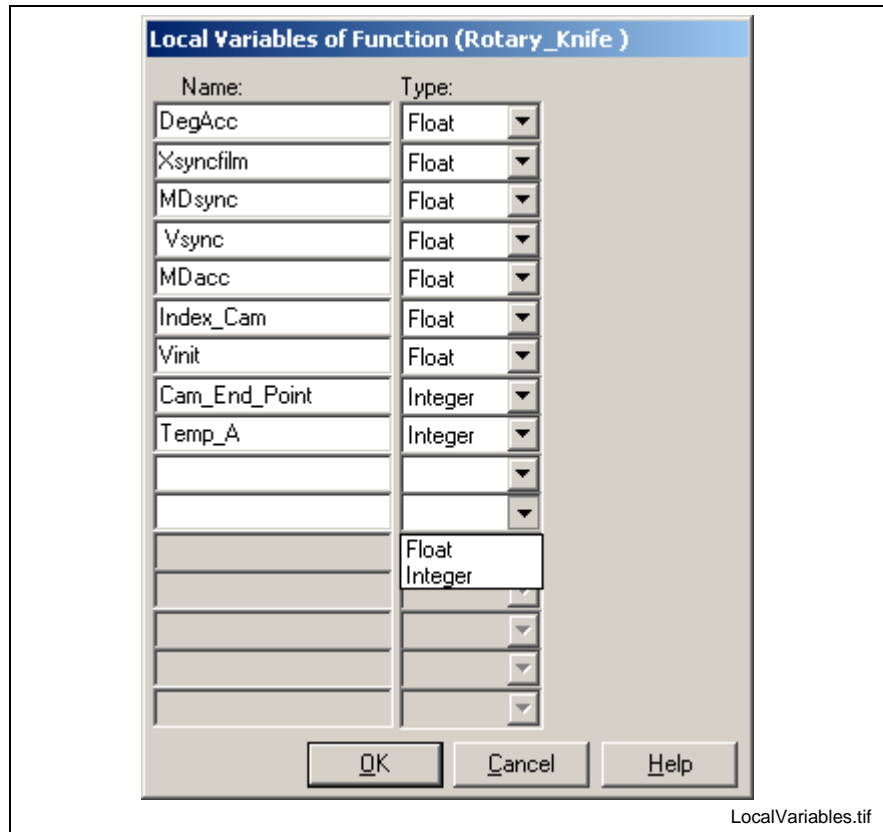


Fig. 3-97: Local Variables

Password

Any task, subroutine or event function can be password protected. The following *Password Setup* window is opened when the **Password...** button is clicked in the *Start Setup* window.

Important Note: Once a password is entered, it should be record and saved. The system does not allow entry to a program area without the correct password. If the password is forgotten for a subroutine or event function, the function must be deleted (right-click on name) and recreated. If the password is forgotten for a task (Initialization, A-D), the entire project must be recreated.

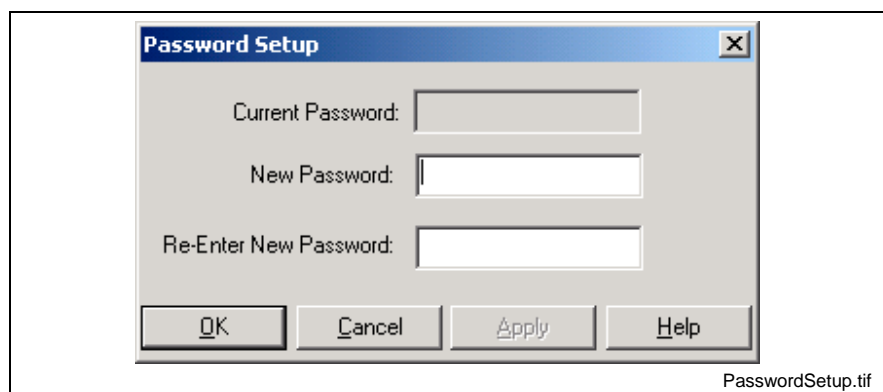


Fig. 3-98: Password Setup

The password can be any combination of letters and/or numbers up to a maximum of 20 characters. Once set, the password is request the first time the task, subroutine or event function is viewed. Afterwards, the user can switch between task without having to enter the password.

Password Protection

Once the password is set, the user must enter the valid password to gain access to desired workspace. If an invalid password is entered or the **Cancel** button is clicked, the following message is displayed in the icon workspace. Password protected workspace areas are indicated by a password icon in the Project Navigator window.

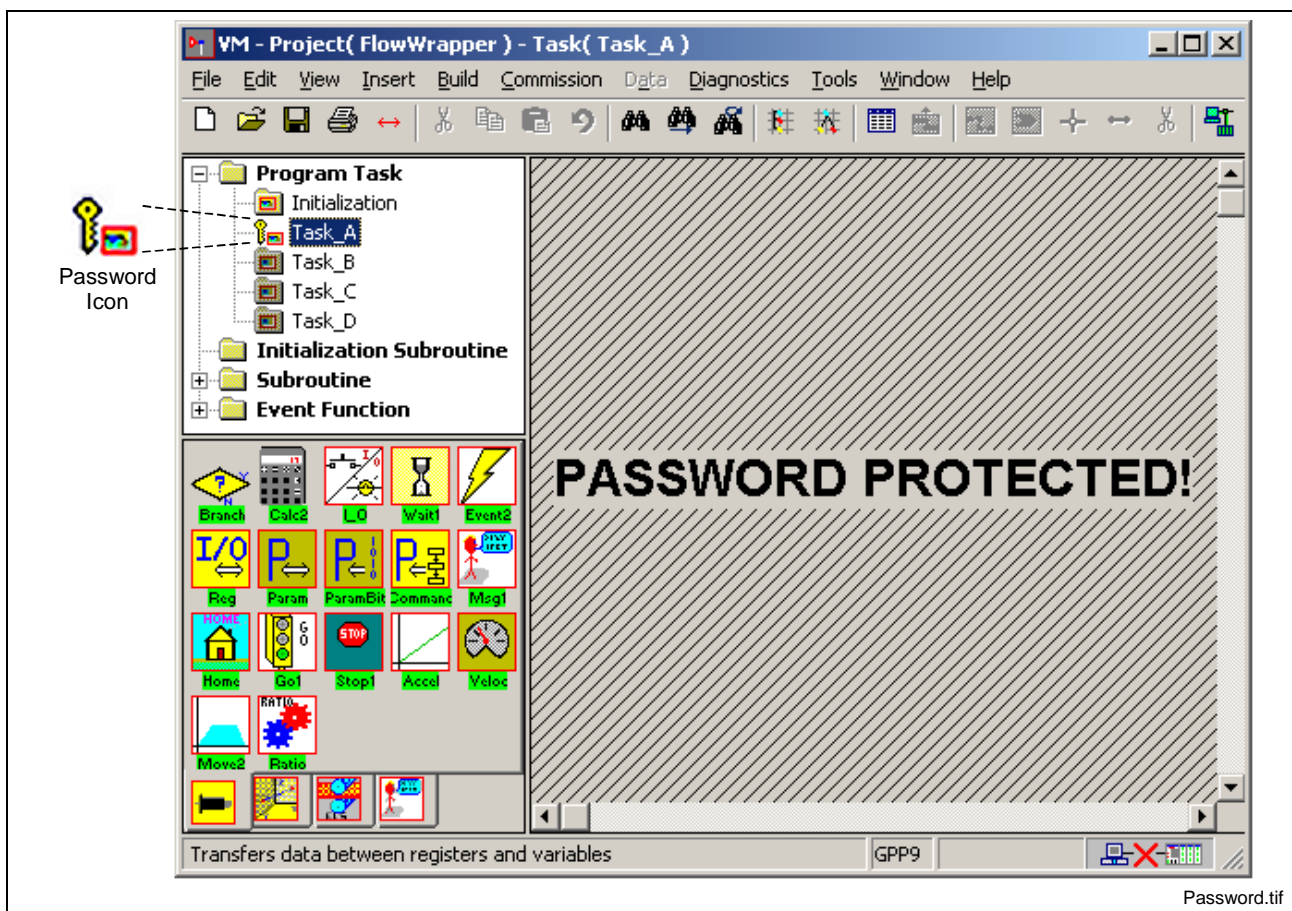


Fig. 3-99: Password Protected

Clear Password

Open the *Password Setup* window by clicking the **Password** button. The **Current Password** field will show the length of the set password as asterisks. To clear the set password, simply click on the **OK** button. Clicking the **Cancel** button will not clear the password.

Properties

The **Properties...** button is used to set task related options for Task A-D. The **Start** icon of each task (A-D) can have its own specific task options. Selecting any checkbox, set the corresponding bit in task parameter T-0-0002 for each task (A-D). Refer to task parameter T-0-0002 for a description of each option available.

Note: The *Task Properties* window can also be open by right clicking over a task letter in the Project Navigator window.

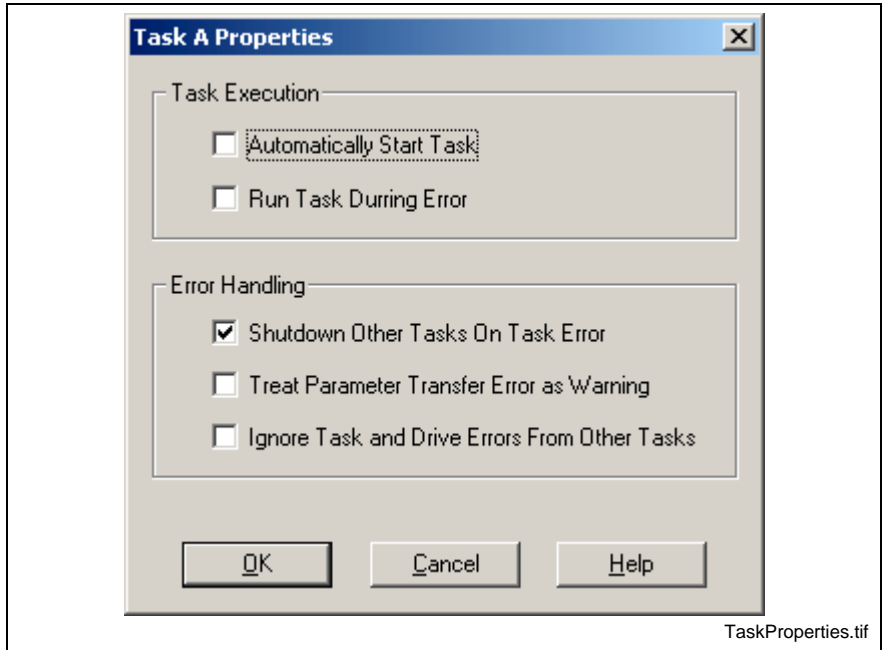


Fig. 3-100: Task Properties

Stop1



The Stop1 icon is used to halt motion on one or all axes used in a task. It will return the drives to an AH (Drive halt) state. Placing a Stop1 icon on a task or subroutine workspace automatically displays a *Stop Setup* window.

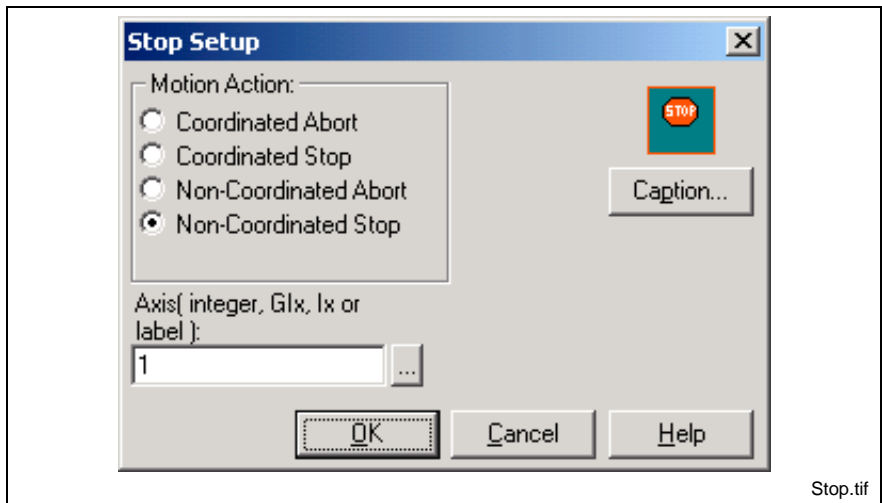


Fig. 3-101: Stop Setup

Integers and labels can be selected from the VM Data Table by clicking the browse button to the right of the **Axis** field.

Coordinated Abort and **Coordinated Stop** decelerate motion on path, stopping all axes associated with the coordinated motion. Selecting Coordinated Abort or Coordinated Stop enables a pop-down menu permitting selection of one of the four control tasks.

Restarting motion after a Coordinated Abort requires toggling the Cycle/Start bit of the associated Task Control register. Motion stopped using a Coordinated Stop may be resumed with a Go1 icon, although resuming timed events that are programmed for motion at operating speed may result in events occurring at unexpected times.

Non-Coordinated Abort and **Non-Coordinated Stop** stops motion on the specified axis by decelerating the axis to zero velocity using the currently programmed rate. The axis to stop may be specified by a valid integer constant, variable, global variable or an equivalent label. Specifying a "-1" stops all axes in the task.

Note: After an **Abort**, all queued events and the "look-ahead" motion calculated by the path planner are lost and the current move is aborted. The target position is set equal to the current feedback position (if A-0-0164, bit 3 = 0). After a **Stop**, both queued events and the calculated path are retained. The target position is not set equal to the feedback position. When the axis is again enabled, it will complete the last commanded move

Sub1



The Sub1 icon is used to invoke (call) a subroutine within the program flow of a task, other subroutine or event function. Placing the Sub1 icon in an icon program workspace opens the *Subroutines* window. The **Optional Function Argument** and **Optional Return Value** fields are initially grayed out.

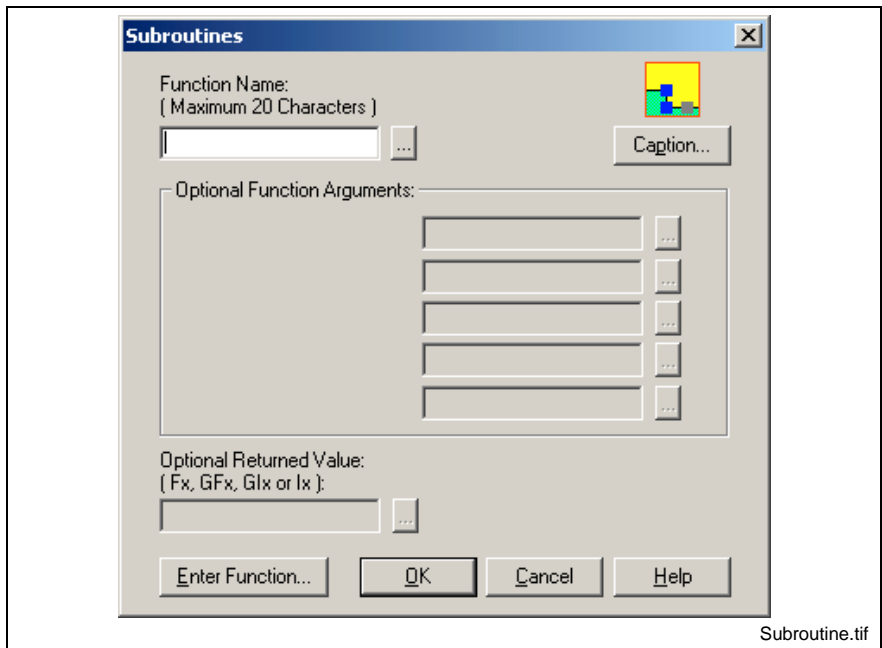


Fig. 3-102: Subroutine Setup

The user can assign a label in the **Function Name** field that will be used to identify the subroutine. Selecting the browse button to the right of the **Function Name** field allows the user to choose an existing subroutine.

To create the icon program for the subroutine, click the **Enter Function...** button after naming the subroutine. The *Subroutines* window closes and a new icon program workspace opens containing a Start and Finish1 icon. The user can then select and place the desired icons to create the program that will run when the Sub1 icon is encountered in the calling program flow. If the **OK** button is clicked before the **Enter Function...** button, then the *Subroutines* window closes and the Sub1 icon appear in the program flow. The user can then return at a future time to program the subroutine.

Note: Subroutine can be viewed by any of the following methods:

- Click the **Enter Function...** button in the Sub1 icon.
- Select **View** ⇒ **Subroutine** from VisualMotion Toolkit's main menu, highlight the desired subroutine and click the **View/Edit...** button.
- Select the subroutine name from the Project Navigator
- Browsing for Icon flow

The name assigned to the subroutine will appear in the *Project Navigator* window. Subroutines created for the Initialization task appear under **Initialization Task**. Subroutines created for a main task, other subroutines or event functions will all appear under **Subroutines**.

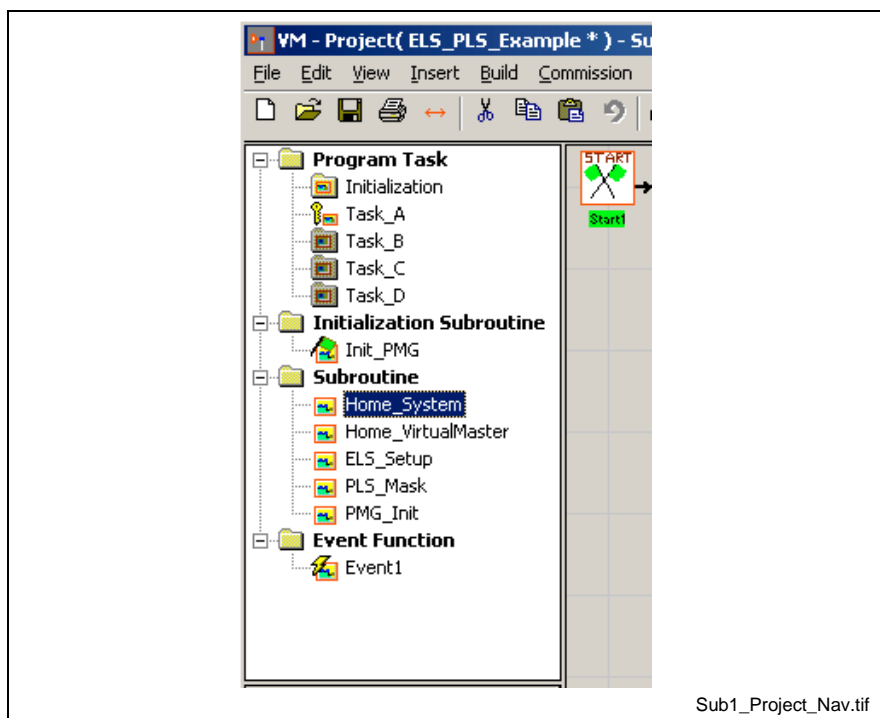


Fig. 3-103: Subroutine Names in Project Navigator Window

Note: VisualMotion 9 has an Initialization task separate from the four main tasks (A-D). Subroutines created for the Initialization task are unique to that task and should not be called from a main task, subroutine or event function. Likewise, main task subroutines should not be called from the Initialization task.

Optional Function Arguments

Function arguments are stack-based variables whose values exist only while in the subroutine (initialization or standard) where they are declared. The **Define Function Arguments** button in the Start1 icon allows the user to define up to 5 function argument names and assign data types to them. These function arguments are variables (float, integer, ABS index or REL index). Refer to Define Function Arguments (Start1 icon) on page 3-109 for details.

Passing Constants

When passing constant values to a subroutine with defined function arguments, the value can be used and/or modified. When the subroutine ends, the modified value is not returned to the caller unless the Optional Returned Value field is used.

Passing Variables

When passing a variable value to a subroutine with defined function arguments, a copy of the variable's current value is passed in to the subroutine. If the passed value is modified while in the subroutine, the original variable existing in the calling task or subroutine is not affected.

The **Optional Function Argument** fields become active in the *Subroutines* window when function arguments are defined in the Start1 icon of the subroutine. The function argument names used in the Start1 icon will appear in the *Subroutines* window after they are defined. The value that will be passed to the function argument is entered in the **Optional Function Arguments** fields.

The **Optional Returned Value** field becomes active in the *Subroutines* window when a value is entered in the **Optional return value** field in the Finish1 icon of the subroutine. The value returned from the Finish1 icon will be written to the variable entered in the **Optional Returned Value** field of the *Subroutines* window.

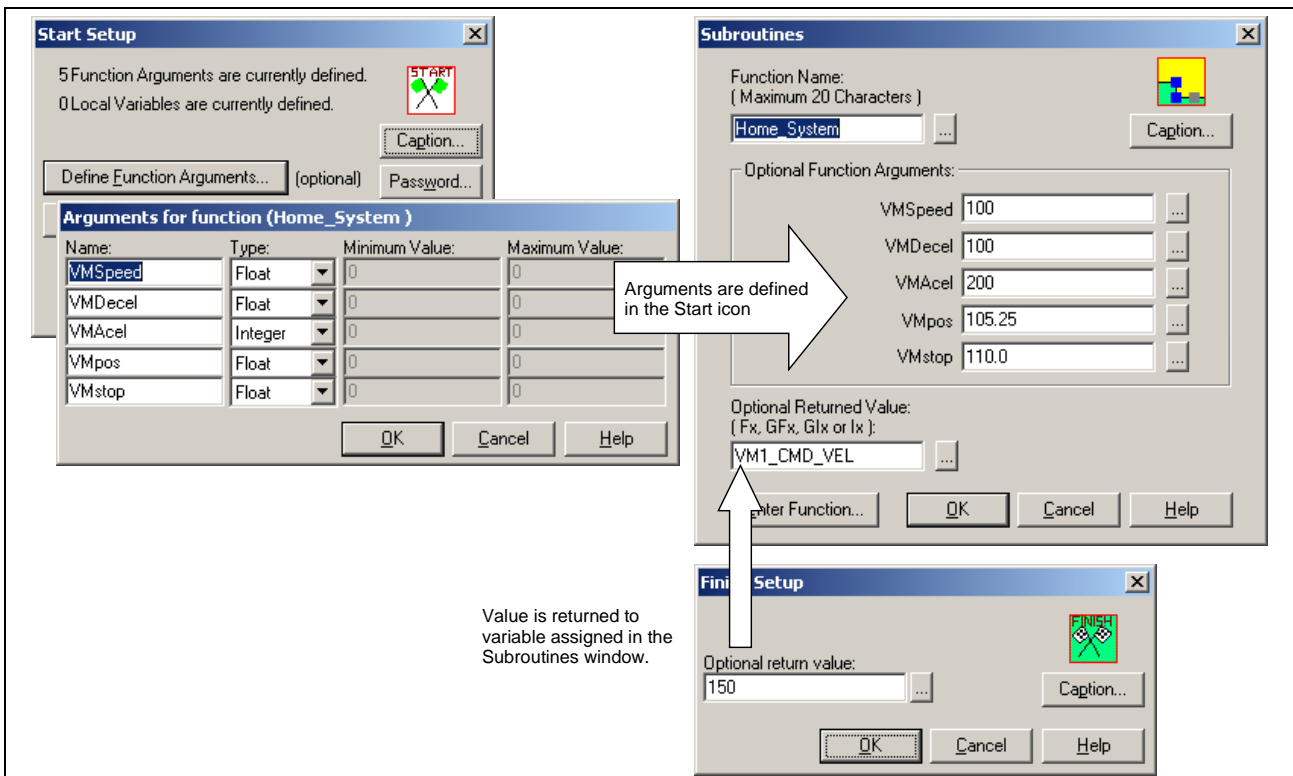


Fig. 3-104: Sub1 Icon – Optional Function Arguments

Function arguments are only allowed in the Start1 icon of a subroutine. Start1 icons in any task or event function will have this feature grayed out. However, an optional return value may be passed back to a task that calls the subroutine. The **Optional Return Value** in the Finish1 icon of a subroutine is a good way of getting a value from a subroutine.

Veloc (Velocity)



The Velocity icon is used to specify a rate for motion on a single non-coordinated axis. The axis may be specified by a valid integer constant, variable, global variable or an equivalent label. The velocity may be entered as a float constant, variable, global variable or an equivalent label. Clicking the browse button to the right of any field opens the VM Data Table.

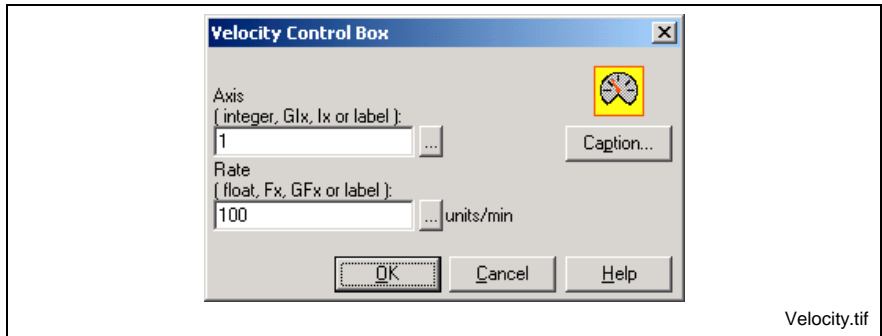


Fig. 3-105: Velocity Setup

VM (Virtual Master)



The VM (Virtual Master) icon is used to assign the control and status registers, and the variable (float and integer) start ID blocks for up to two virtual master axes. Fifteen floats and two integers are set aside for each Virtual Masters. The default register values and start ID blocks for the floats and integers are shown in the *Virtual Masters 1 & 2 Setup* window.

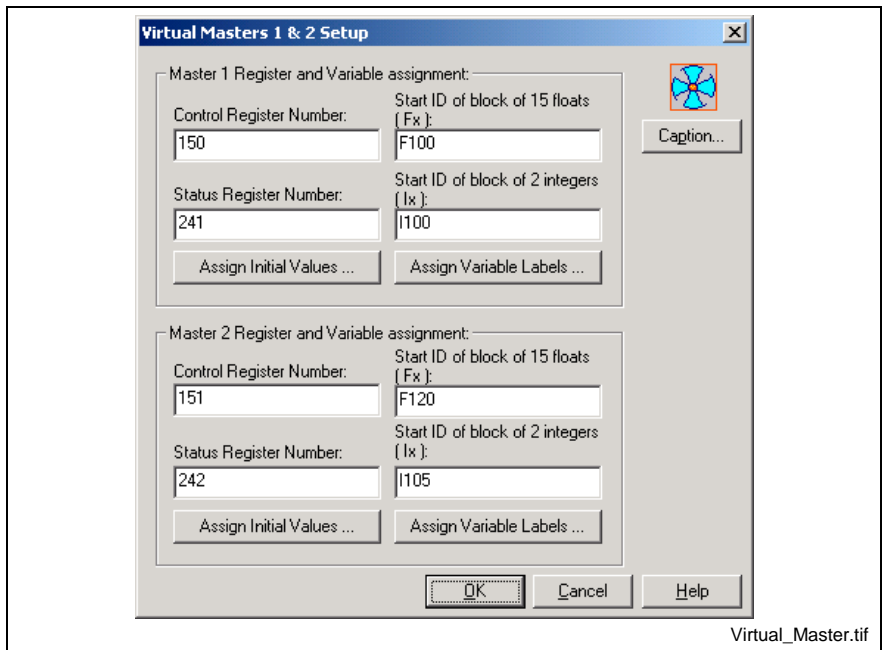


Fig. 3-106: Virtual Master Setup

Note: A Virtual Master is automatically assigned to Task A. All control and monitoring of the Virtual Master is performed by the control and status registers of Task A. All ELS Group axes that are Sync to Master and following the Virtual Master will stop if Task A is stopped.

The Virtual Master icon must be placed in the main level of the Initialization task. It should not be placed in an Initialization Subroutine or a system error will be issued when the program is compiled and downloaded to the control.

Register and Variable Assignment

The "Master Register and Variable Assignment" section is used to define the registers and program variable blocks that will be used for each Virtual Master.

Note: Default values are automatically assigned for registers and variable blocks. It is strongly recommended that the programmer use the default values for registers and variables. This makes documentation and modifications to user program an easier task over the scope of the project.

Control Register Number

This number represents the control register number assigned to each Virtual Master. Default control register labels and numbers for each Virtual Master are listed in the table below.

Virtual Master 1 & 2 Control Register Default Label	Virtual Master 1 Default Control Register-Bit	Virtual Master 2 Default Control Register-Bit	Virtual Master 1 & 2 Control Register Default Comment (80 character limit)
VM#_CT_FSTOP	150-1	151-1	VM # control, 0 → 1 triggers fast stop
VM#_CT_HOME	150-2	151-2	VM # control, 0 → 1 loads home position
VM#_CT_GO	150-3	151-3	VM # control, 0=stop, 1=go
VM#_CT_VMODE	150-4	151-4	VM # control, 0=position, 1=velocity mode
VM#_CT_RELMODE	150-5	151-5	VM # control, 0=absolute, 1=relative mode
VM#_CT_RELTRIG	150-6	151-6	VM # control, 0 → 1 triggers relative mode
Each # symbol represents an entry for the number of the Virtual Master			

Table 3-28: Virtual Master Control Registers

Status Register Number

This number represents the status register number assigned to each Virtual Master. Default status register labels and numbers for each Virtual Master are listed in the table below.

Virtual Master 1 & 2 Status Register Default Label	Virtual Master 1 Status Register-Bit	Virtual Master 2 Status Register-Bit	Virtual Master 1 & 2 Status Register Default Comment (80 character limit)
VM#_ST_FSTOP	241-1	242-1	VM # status, 1=fast stop active
VM#_ST_HOME	241-2	242-2	VM # status, 1=home complete
VM#_RESERVE3	241-3	242-3	
VM#_ST_VMODE	241-4	242-4	VM # status, 1=velocity mode
VM#_ST_RELMODE	241-5	242-5	VM # status, 1=relative mode
VM#_RESERVE6	241-6	242-6	
VM#_ST_ZEROVEL	241-7	242-7	VM # status, 1=standstill, 0=velocity
VM#_ST_INPOS	241-8	242-8	VM # status, 1=in position
Each # symbol represents an entry for the number of the Virtual Master			

Table 3-29: Virtual Master Status Registers

ELS System Master Program Variable Start ID Blocks

Start ID of block of 15 floats (Fx):

This number represents the first float in a block of 15 floats set aside for each Virtual Master. The float number must be preceded with an "F".

Example: F100

Start ID of block of 2 integers (Ix):

This number represents the first integer in a block of 2 integers set aside for each Virtual Master. The integer number must be preceded with an "I". **Example:** I100

Default program variable labels and numbers for all 6 ELS System Masters are listed in the table below.

Virtual Master 1 & 2 Program Variable Default label	Virtual Master 1 & 2		Virtual Master 1 & 2 Program Variable Default Comment (80 character limit)	Default Initial Value	Units	Update Mode
VM#_HOME_POS	F100	F120	Virtual Master # home position	0	Degrees	Phase 4
VM#_REL_MOVE_DIST	F101	F121	Virtual Master # relative move distance	1	Degrees	Phase 4
VM#_STOP_POS	F102	F122	Virtual Master # stop position	0	Degrees	Phase 4
VM#_CMD_ABS_POS	F103	F123	Virtual Master # commanded absolute position	0	Degrees	Phase 4
VM#_CMD_VEL	F104	F124	Virtual Master # commanded velocity	20	RPM	Phase 4
VM#_CMD_ACCEL	F105	F125	Virtual Master # commanded acceleration	100	Rad/sec ²	Phase 4
VM#_CMD_DECEL	F106	F126	Virtual Master # commanded deceleration	100	Rad/sec ²	Phase 4
VM#_E_STOP_DECEL	F107	F127	Virtual Master # E-Stop deceleration	500	Rad/sec ²	Phase 2
VM#_MAX_VEL	F108	F128	Virtual Master # maximum velocity	100	RPM	Phase 2
VM#_MAX_ACCEL	F109	F129	Virtual Master # maximum acceleration	500	Rad/sec ²	Phase 2
VM#_MAX_DECEL	F110	F130	Virtual Master # maximum deceleration	500	Rad/sec ²	Phase 2
VM#_JERK_ENABLE	F111	F131	Virtual Master # jerk limiting enable	1	Degrees	Phase 2
VM#_CUR_POS	F112	F132	Virtual Master # current position		Degrees	Phase 4
VM#_CUR_VEL	F113	F133	Virtual Master # current velocity		RPM	Phase 4
VM#_POS_WIN	F114	F134	Virtual Master # shortest path window	1	Degrees	Phase 2
VM#_POS_MODE	I100	I105	Virtual Master # positioning mode ^{1.)}	0		Phase 2
VM#_RESERVE_I1	I101	I106				

Each # symbol represents an entry for Virtual Masters 1 & 2
 Note 1.) Absolute Position Mode, 0=Positive, 1= Negative, 2= Shortest Path

Table 3-30: Virtual Master Program Variables

Assign Variable Labels

Default variable labels and comments can be added individually for Variable, Registers and Bits by selecting the appropriate **Data Type** radio button and clicking the **Add Default Labels** button. Clicking the **Add All Default Labels** button will add the default variable labels and comments to all Data Types at one time.

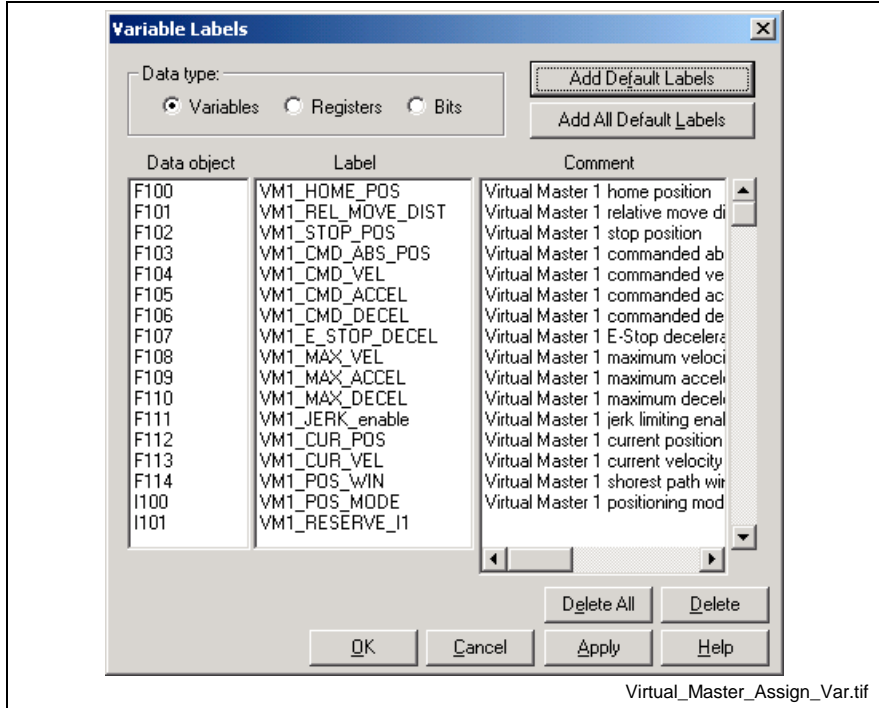


Fig. 3-107: Add Default Labels for Virtual Master

Assign Initial Values

Refer to Virtual Master Compile Time Initialization in chapter 6 of the *VisualMotion 9 Application Manual* for details.

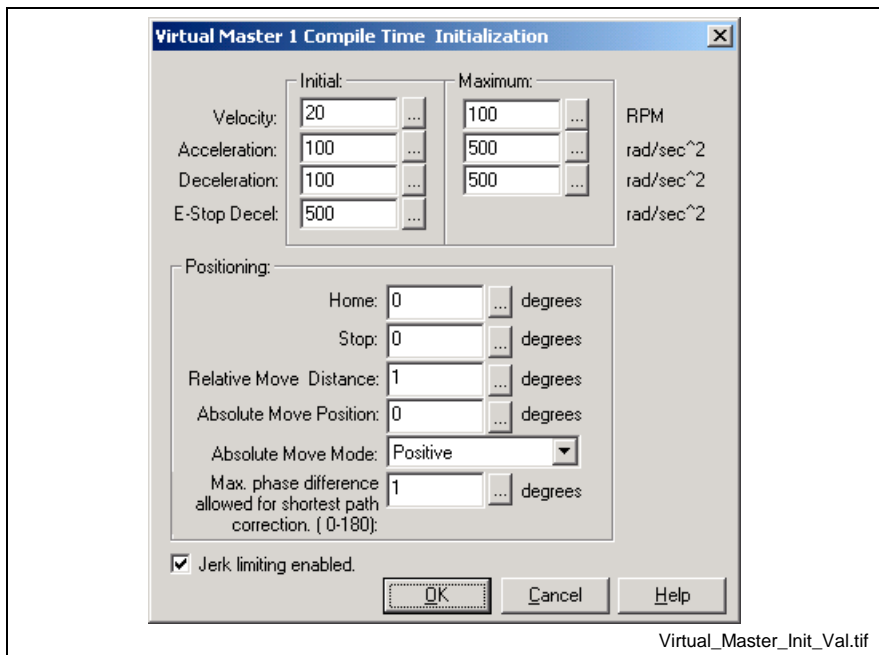


Fig. 3-108: Configure Initial Values for Virtual Master

Wait1



The Wait1 icon is used to hold the execution of the program flow at the Wait1 icon until a specified condition has been satisfied. The condition may be related to a single axis' position, time, I/O state, or path planner state (for coordinated motion). Placing a Wait1 icon on a task or subroutine workspace automatically displays a *Wait Control Box* window.

Axis in Position pauses program execution until the specified axis reaches the in position window of the associated drive. The axis may be specified by a valid integer constant variable (Ix) global variable (Gix) or an equivalent label.

If a "-1" is entered, program execution will wait until all axes assigned to the task are within their respective position windows before continuing.

Axis at Position pauses program execution until the specified axis reaches a specified position for the associated drive.

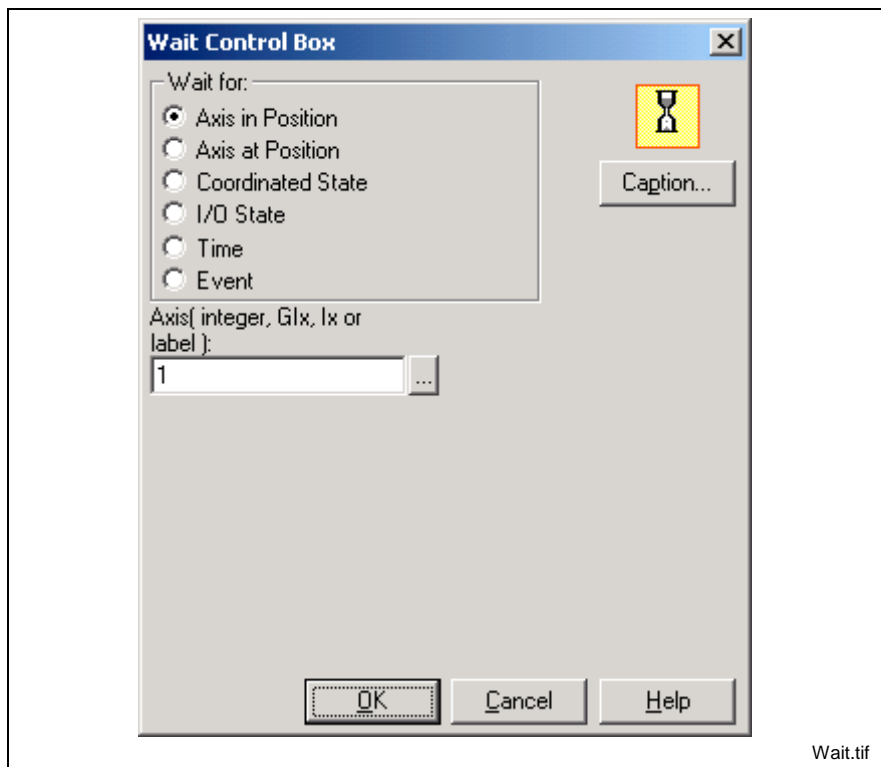


Fig. 3-109: Wait Setup

Selecting **Coordinated State** pauses program execution and tests the state of the path planner for the specified point until the planner enters the selected processing state. Coordinated Waits are specific to each task. You can wait in one task for the coordinated state of another

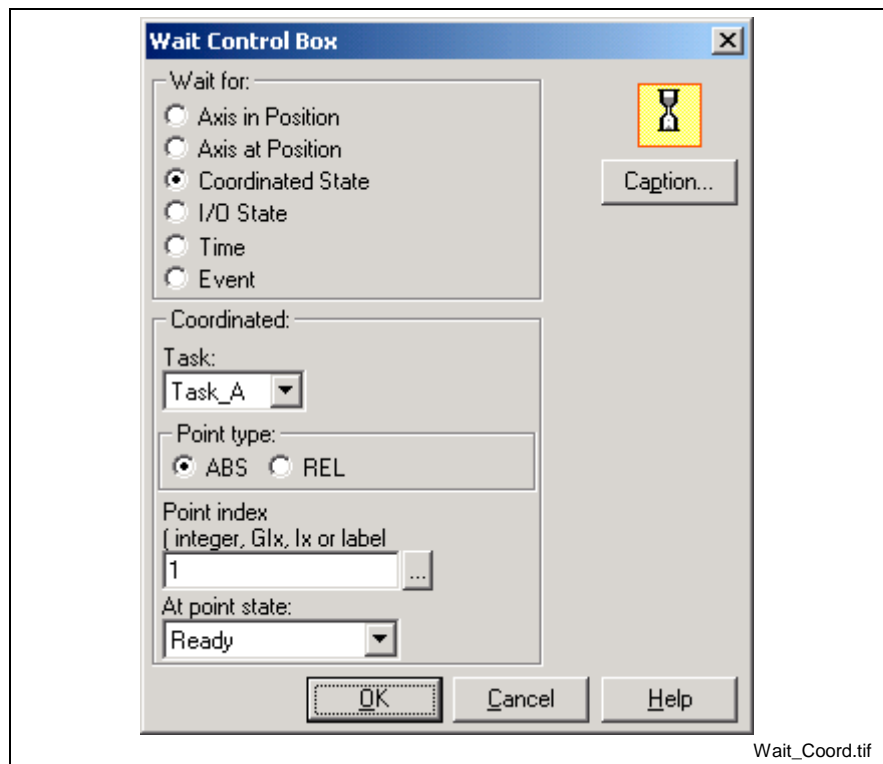


Fig. 3-110: Wait Icon Setup for Coordinated State

The **At point state** field provides the eight path planner test states listed below:

- **Segment Ready** - Path planner has processed and queued the specified point.
- **Acceleration** - Task's coordinated motion is accelerating into the segment ending at the specified point.
- **Constant Velocity** - Task's coordinated motion is traversing the segment ending at the specified point.
- **Blending** - Task's coordinated motion is traversing the blend segment calculated for the specified point.
- **Target Deceleration** - Task's coordinated motion is in deceleration in the segment ending at the specified point.
- **Controlled Stop** - Task's coordinated motion is decelerating on the segment ending at the specified point after a commanded stop.
- **Stopped** - Task's coordinated motion is stopped after a commanded stop.
- **At Target** - Task's coordinated motion is at the specified point.
- **Done** - Task's coordinated motion has completed for the specified point.

I/O State pauses program execution until the specified I/O condition for the specified I/O register is satisfied. The I/O register is specified by entering a valid I/O register ID number or equivalent label. Clicking the browse button to the right of any field opens the VM Data Table.

The contents of the specified register may be "anded" with a bit mask for the register contents. The bit mask allows "masking out" unwanted bits (by specifying "0" in the bit position in the mask), and permits the on/off condition to be effected by more than a single bit (by specifying "1" to enable the bit).

The **I/O State** radio buttons determine how the wait condition is satisfied. I/O State "on" (high or "1") requires that all the enabled bits are logical one. I/O State "Off" (low or "0") requires that all enabled bits are zero.

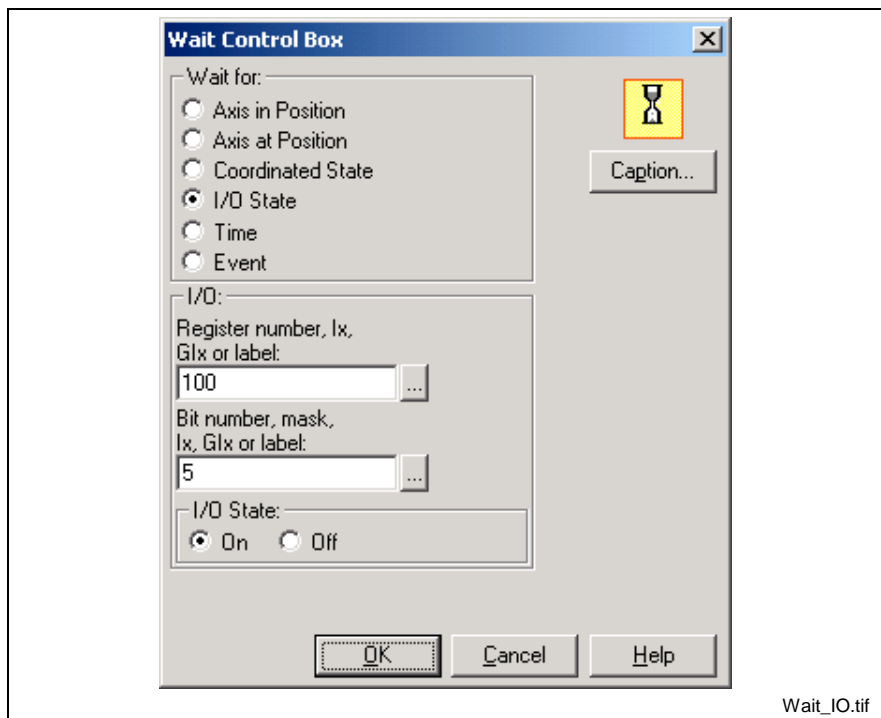


Fig. 3-111: Wait Setup for I/O state

Time pauses program execution for a specified time delay. Enter the number of milliseconds in the **Time Delay** field. The delay may be specified by an integer constant, variable (Ix), global variable (GIx), or an equivalent label. Clicking the browse button to the right of any field opens the VM Data Table.

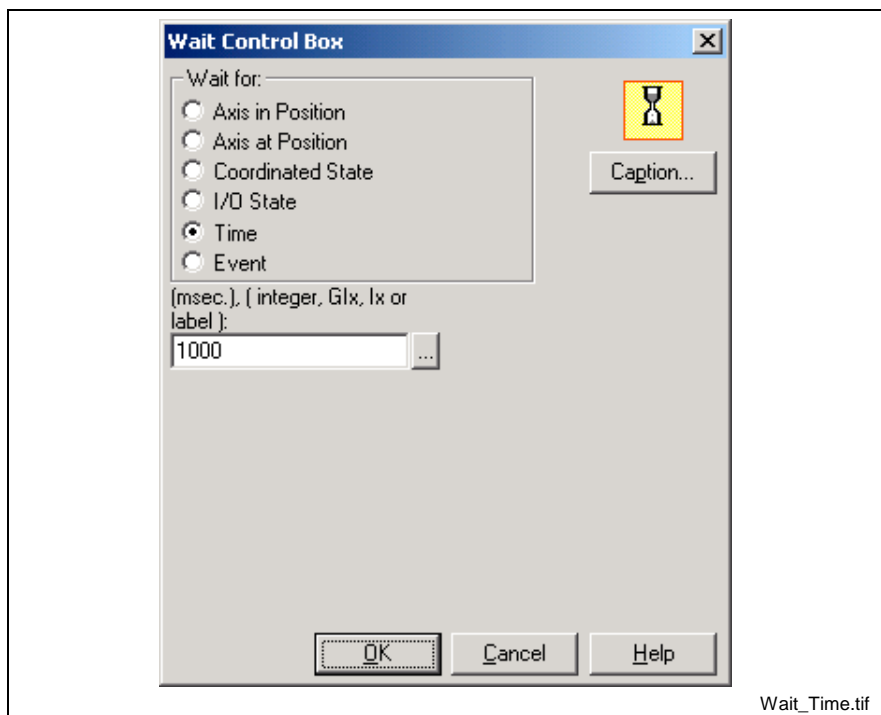


Fig. 3-112: Wait Setup for Time

Event pauses program execution until the specified event has completed. The event may be specified by an integer constant, variable (Ix), global variable (Glx), or an equivalent label. The VM Data Table can be displayed by clicking on the button to the right of the **Event ID** field.

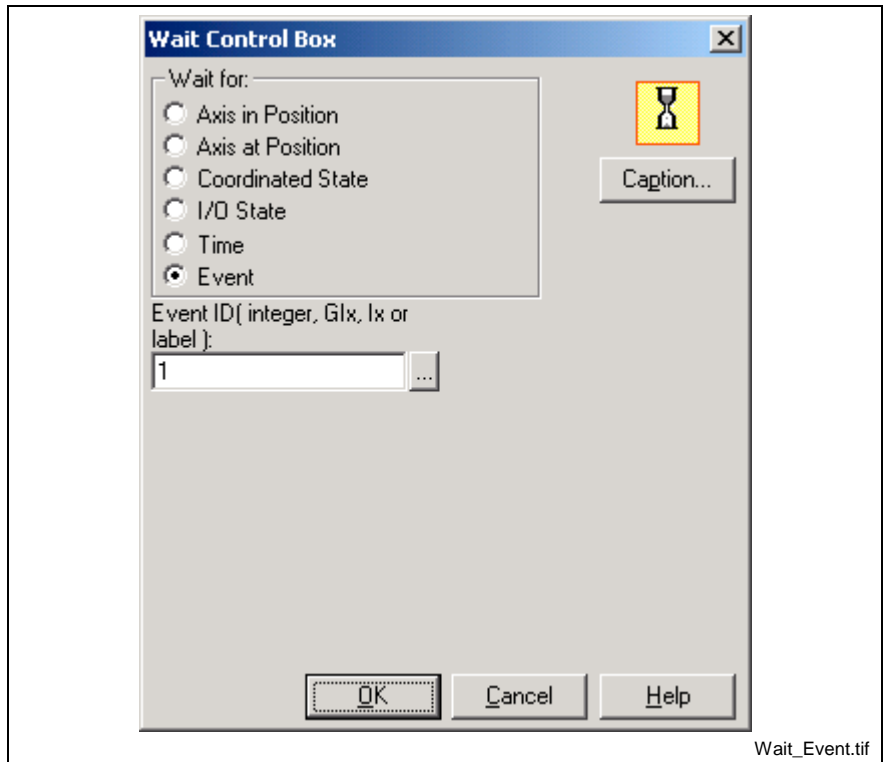


Fig. 3-113: Wait for Event

4 Registers

VisualMotion uses registers as a method for communication and system monitoring between the control (using I/O) and an external system. VisualMotion provides 1024 registers. Some registers are reserved for control and system functions and others are recommended as defaults. The remaining registers are available for use by the programmer. Registers can be viewed by selecting **Data** ⇒ **Registers** from VisualMotion Toolkit's main menu. A breakdown of VisualMotion registers can be found in Table 4-1.

Register	Register Label	Availability
001	System Control	Reserved for System
002-005	Task A-D Control	Reserved for System
006	System Diagnostic Code	Reserved for System
007-010	Task A-D Jog Control	Reserved for System
011-018	Axis Control 1-8	Reserved for System
019	Fieldbus Status	Reserved for System
020	Fieldbus Diagnose	Reserved for System
021	System Status	Reserved for System
022-025	Task A-D Status	Reserved for System
027	Initialization Task Control	Reserved for System
028	Initialization Task Status	Reserved for System
031-038	Axis Status 1-8	Reserved for System
040	Link Ring Status	Reserved for System
041-042	Link Ring Data	Reserved for System
050	Ethernet Status	Reserved for System
051	Standard Message Count	Reserved for System
052	Cyphered Message Count	Reserved for System
053	Invalid Count	Reserved for System
054	SIS Message Count	Reserved for System
086	PMG Control	Reserved for System
087	PMG Status	Reserved for System
088 and 089	Task A Extend Event Control	Reserved for System
090 and 091	Latch and Unlatch	Reserved for System
092-094	Mask BTC06 Key Functionality	Reserved for System
095-097	BTC06 Teach Pendant Status	Reserved for System
098 and 099	BTC06 Teach Pendant Control; Task A-B, C-D	Reserved for System
140	ELS Master Control	Reserved for System
141	ELS Master Status	Reserved for System
150 and 151	Virtual Master 1 & 2 Control	System Default
152-159	ELS Groups 1 – 8 Control	System Default
197	Coordinated Articulation Synchronized Mode Control	System Default
198	Coordinated Articulation Local Mode Control	System Default
209-240	Axis Control 9-40	Reserved for System
241 and 242	Virtual Master 1 & 2 Status	System Default
243-250	ELS Groups 1 – 8 Status	System Default
288	Coordinated Articulation Synchronized Mode Status	System Default
289	Coordinated Articulation Local Mode	System Default
309-340	Axis Status 9-40	Reserved for System

Table 4-1: Control Register

4.1 Register 001: System Control

System Control register bits are dedicated to system supervisory control functions.

Register	Register Label Name	Bit	Function
001	System_Control	1	Parameter Mode
		2	Not Used
		3	nEmergency Stop
		4	Not Used
		5	Clear All Errors
		6	Pendant Live-Man
		7	Rebuild Double Ring
		8	Activate Program
		9	Program Select LSB
		10	Program Select Bit_2
		11	Program Select Bit_3
		12	Program Select MSB
		13	Not Used
		14	Pendant Enable
		15	Pendant Level LSB
		16	Pendant Level MSB

Table 4-2: Register 001: System Control

Bit 1: Parameter Mode/ nRun Mode

A low-to-high (0-1) transition of this bit switches the system into Parameter Mode. All user tasks are immediately interrupted. The system is switched into parameter mode, and the drives are switched into SERCOS phase 2.

A high-to-low transition (1-0) re-initializes the system and switches it to Run Mode. Parameter initializations are performed and the drives are switched from phase 2 to phase 4. If there are no errors, the user tasks are ready to operate.

Bit 3: nEmergency Stop

This input is active low (0 = Emergency Stop). When set to (0), all user tasks are stopped except those selected to run during errors. All motion is stopped, and the drives are set to zero velocity and disabled. When set to (1), the emergency stop condition has been corrected, and the tasks can run if there are no other errors.

Bit 5: Clear All Errors

A low-to-high transition clears any existing system, task and drive errors as long as the condition that cause the error is resolved.

Bit 6: Pendant Live Man

This bit should be mapped to the teach pendant live man switch when a teach pendant is used. When set to (0), no motion can be initiated from the teach pendant and any motion in progress is immediately stopped. When set to (1), (*live-man closed*) the teach pendant can jog, start, and stop motion.

Bit 7: Rebuild Double Ring

This bit is used to reinitialize the double ring (primary and secondary) structure in a Link Ring. During a Redundancy Loss (register 40, bit 7) error at any node in a Link Ring configuration, hold this bit high (1) on ALL nodes until every node's Redundancy Loss bit goes low.

Next, set Clear ALL Errors (register 1, bit 5) on all nodes that have their Error output (register 21, bit 5) set.

Bit 8-12: Activate Program and Binary Program Select

The active program can optionally be changed using I/O bits 9-12 in the System Control register. A transition from (0) to (1) on Activate Program bit 8 will start the program based on bits 9-12. These bits correspond to the program number (from 1 to 10). If the Activate Program bit is high at power-up, the program selected with the Binary Program Select bits will be activated.

Note: User interface (VisualMotion, Teach Pendant, etc.) selections take precedence over these bits. The actual active program is acknowledged in System Status Register bits 9-12

Example To activate program 5, set the bits as follows:

Bit 9: 1

Bit 10: 0

Bit 11: 1

Bit 12: 0

Then Bit 8 requires a transition from 0 to 1.

Bit 14: Pendant Enable

This bit toggles control of tasks and jogging between the teach pendant and the I/O system. When set to (0), the system I/O Mapper and control registers are in control. When set to (1), the teach pendant assumes control of system functions, forcing all relevant bits in the control registers.

Bits 15-16: Pendant Access Level

These bits provide access protection for teach pendant menus. The protection levels are defined per-menu to provide restricted access to data. If a menu's protection level exceeds the value of these bits, the menu can be viewed but not edited.

4.2 Registers 002-005: Task Control

Task Control registers 002, 003, 004 and 005 are dedicated to task control for Tasks A, B, C and D respectively.

Register	Register Label Name	Bit	Function
002	Task A Control	1	Mode:Auto /nManual
003	Task B Control	2	Override Automatic Start
004	Task C Control	3	Not Used
005	Task D Control	4	Single Step
		5	Not Used
		6	Cycle Start /Resume
		7	nTask Stop
		8	Not Used
		9	Task Event Trigger
		10	Trace Enable
		11	Breakpoint Enable
		12	Sequencer Single Step
		13	Step Sequence Function
		14	Not Used
		15	Coordinate Fast Stop
		16	Not Used

Table 4-3: Registers 002-005: Task Control

Bit 1: Mode: Automatic/ nManual

This bit selects the mode of operation for a task. When set to (0), the task is in Manual Mode, the user program does not run, and manual jogging is enabled. When set to (1), the task is in Automatic Mode and is ready to begin execution.

Coordinated, Single Axis and Velocity Modes

**Bit 1 = 1
Switch to Automatic** The instruction pointer is reset to the beginning (Start Icon) of the program, and all events become inactive. At the next (0-1) transition of the cycle start bit when the cycle stop bit is (1), the program starts running.

**Bit 1 = 0
Switch to Manual** The user task and any events associated with it stop execution immediately. Any motion associated with the task is immediately decelerated to zero velocity. Coordinated, single-axis, and velocity axes are stopped using the maximum deceleration.

ELS Mode

**Bit 1 = 1
Switch to Automatic** VisualMotion's program flow resets at the beginning (Start Icon) of the program. Virtual Masters 1 and 2 are stopped at the programmed deceleration. All ELS Groups with active real or group input masters are switched to local mode.

**Bit 1 = 0
Switch to Manual** The ELS master is stopped using the E-Stop deceleration. The instruction pointer is set to where the program was stopped, but is reset to the beginning of the program when the task is returned to Auto Mode.

Bit 2: Override Automatic Start

Each task in a project can be individually configured to start immediately upon exiting parameter mode or after clearing an error when task parameter T-0-0002, bit 4 (Automatically Start Task) is set to 1.

When this bit is set to (1) and T-0-0002 = 1, the "Automatically Start Task" option is overridden and the task is disabled. The other task control register bits, such as Cycle Start/Resume, Single Step Select, etc., are enabled.

When this bit is set to (0) and T-0-0002 = 1, the "Automatically Start Task" option is enabled and the task control register bits are disabled.

Note: For running tasks, the setting of the Override Automatic Start bit requires a state change from one of the other task control register bits in order to enable or disable the function.

Bit 4: Single Step Select

When set to (1), the task is placed in Single Step Mode. Each positive (0 to 1) transition of Cycle Start bit executes one user task program instruction then pauses (providing that the system is in Automatic mode and nTask Stop is inactive). If this bit is set while a task is running the current instruction completes. The task then pauses and waits for a Cycle Start transition.

Event functions cannot be single-stepped. If one or more events have been started or queued by executing the current instruction, the events will always continue to completion before pausing the user task program.

When set to (0), Single Step mode is disabled. When Single Step is zero, normal cycling begins at the next positive transition of the Cycle Start bit (providing that automatic mode is true and nTask Stop is false).

Bit 6: Cycle Start/Resume

A low-to-high transition (0-1) start executing the user task starts executing at the current instruction, if the task is in automatic mode, the nTask Stop bit is (1), and there are no errors. It is also used to resume the task after a task stop and to restart the task after entering automatic mode. If single-stepping is enabled, the next instruction is executed with each positive transition. A low-to-high (0-1) transition is required to start or resume the task

Bit 7: nTask Stop

When Bit 7 = 0, the nTask is stopped. A high-to-low transition (1-0) stops the task program at the end of the current instruction. All types of motion are halted and can be resumed at the next Cycle Start. The nTask Stop bits function as a "Pause". When the bit is set to 0, the control pauses execution of the task and the instruction pointer remains at the current instruction. When the bit is again = 1 and the cycle start bit is toggled, the task resume execution at the current instruction.

All types of events will execute during a cycle stop state. Only the main program flow in tasks A, B, C and D is affected.

In **Coordinated motion**, the nTask Stop bit will decelerate the currently active segment point at the point's programmed deceleration percentage.



CAUTION

Coordinated motion segment points are programmed with individual acceleration and deceleration percentage rates. Using the nTask Stop bit might not decelerate the axis as quickly as desired. If an immediate stop condition is required, use the Coordinate Fast Stop function (bit 15).

Motion is then paused on the current segment. All distance and time-based events remain active. As long as the task is in automatic mode, the next 0 to 1 transition on the cycle start (bit 6) will resume motion and complete all pending segments.

Single axis motion halts each axis in the task by decelerating the axes to zero velocity while retaining target position. The previous state of the GO command is saved until the next cycle start. If the GO command was active, motion will be resumed at the next cycle start in automatic mode. All normal and repeating events remain active. All **Velocity** mode axes are decelerated to zero velocity if ramping is enabled, or set to zero velocity if step command is selected. All events remain active.

The **ELS** master axis is decelerated to zero velocity. The previous state of the GO command is saved until the next cycle start. If the GO command was active, the master will be commanded the last programmed velocity. All events on the ELS slaves remain active. The slave axes remain synchronized to the master if synchronization was enabled. Because the control has no control over a real master, motion of the slaves of a real master cannot be changed. The operation of **Torque** mode axes during a **nTask Stop** is not defined at this time.

Note: *nTask Stop does nothing to assure that the system is in a safe or known condition to stop. nTask Stop simply completes the current instruction, then ramps down motion in the task.*

Bit 9: Task Event Trigger

This bit is reserved as an Event Interrupt Input for each task. Each low-to-high (0 to 1) transition of this input will trigger an event to the corresponding task. This event type can be used to start a process or to respond to an external event.

In the event table, Type 6 selects an Interrupt Input event. The event/trigger (arm event) instruction enables the interrupt input. The event/done (disarm event) instruction is used to disable the input. The control scans the input every 4ms and starts an event upon a low-to-high transition. The event function will take priority over the user tasks, allowing quick response to an external input. The I/O Mapper can be used to reverse the logic of the interrupt input, or to direct other external inputs to it. Logic in the event function can then scan the multiple inputs to determine the source of the interrupt.

Bit 11: Breakpoint Enable

When this bit is set to (1), the breakpoint enabled in task parameter T-0-0137 is active. When program flow reaches the breakpoint, the task is stopped. When this bit is set to (0), the program executes normally, without breakpoints.

Bit 12: Sequencer Single Step

This bit places the control into Sequence Single Step mode. As long as this bit is (1), a low-to-high (0-1) transition on the cycle start bit causes the program to be stopped after each sequencer step is executed. If this bit is (0), the sequencer executes normally.

Bit 13: Step Sequence Function

This bit places the control into Function Single Step mode. As long as this bit is (1), a low-to-high (0-1) transition on the cycle start bit causes the program to be stopped after each sequencer function is executed. If this bit is (0), the sequencer executes normally.

Bit 15: Coordinate Fast Stop

This bit is used to perform a controlled stop of all task motion using the task's speed (T-0-0020), acceleration (T-0-0021) and deceleration

(T-0-0022) parameters. A low-to-high transition (0-1) activates this function, halting all task motion while maintaining positioning. A high-to-low transition (1-0) deactivates this function, continuing task motion from the stopped position.

4.3 Cycle Control Considerations

Cycle Stop in User Program

A cycle stop implies that a task's motion cycle has completed and the system is at a safe place to halt. The nTask Stop bit cannot always be used for this purpose, since it only stops task instruction execution and commands the drives to ramp down.

A nTask Stop signal may lose track of user task activity that is related to axis or segment position or time. In addition, the control's path planner may have several queued segments or events. Queued events always continue execution until completion. This may result in the system position and I/O losing synchronization with your programmed sequence of task instructions. If you attempt to simply restart motion, the results may not be predictable.

If each of your tasks require a cycle stop capability, a separate user task I/O bit should be user configured into the I/O system for each task needing a cycle stop. Your program then tests the associated I/O bit from within your task program. Use the condition of the I/O bit to branch to a program routine that halts motion and establishes a known system state. Since you are programming a unique system, only you can determine a safe system condition.

System Parameter Mode

A switch to Parameter Mode immediately disables all user tasks, and switches the control and axes to SERCOS Phase 2.

System Shutdown Errors

A shutdown error disables all drives, stops coordinated and single-axis motion, and puts the task into manual mode. All control bits are left at their current state.

Programmed End of Task

A task program that reaches and executes the task/end instruction or the Finish Icon has the same effect as activating the nTask Stop I/O line.

4.4 Register 006: System Diagnostic Code

This status register shows the current control diagnostic code in Motorola 16-bit format. This register is displayed as a 3 digit code when ***Decimal*** is the selected ***Format*** under ***Data*** ⇒ ***Registers***.

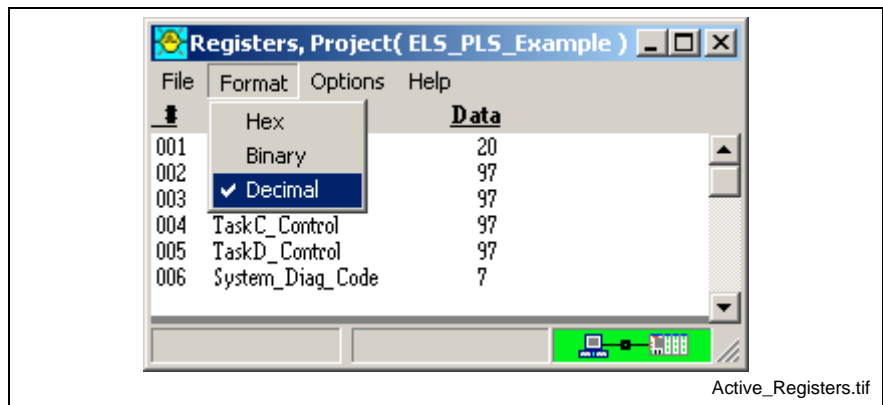


Fig. 4-1: Displaying Register 6

4.5 Registers 007-010: Task Jog Control

Registers 007, 008, 009 and 010 are dedicated for jogging control of Tasks A, B, C and D respectively.

Single-Axis and Velocity Mode Jogging

Any axis in single-axis or velocity mode can be jogged independently when its associated task is in manual mode. Single-axis and velocity mode axes are jogged using the Axis Control Registers. Motion is started with a low-to-high transition on either the jog forward or the jog reverse bit. Motion is stopped when both jog bits are low, when both jog bits are high, when the task mode selection changes, or when a travel limit or incremental distance has been reached. Jog mode (continuous or incremental) and jog speed or distance is selected in the Task Jog Register for the task associated with the axis.

Coordinated Jogging

Coordinated axes may be jogged in any direction while a task is in manual mode with the Task Jog Register. The coordinate to jog, type of jog, and parameters to use are selected using bits in this register.

Before jogging can begin, the coordinated axis must be selected with bits 9, 10, or 11 of the relevant task control register. Jogging begins when either bit 2 (Coordinated Jog Forward) or bit 3 (Coordinated Jog Reverse) is transitioned from 0 to 1. The axis will move the distance calculated by C-0-0042 and T-0-0025 with every 0-1 transition of the Coordinated Jog Forward or Reverse bits.

Register	Register Label Name	Bit	Function
007	Task A Jog	1	Continuous nStep
008	Task B Jog	2	Coordinate Jog Forward
009	Task C Jog	3	Coordinate Jog Reverse
010	Task D Jog	4	Jog Type LSB
		5	Jog Type MSB
		6	Distance Speed
		7 and 8	Not Used
		9	Jog X Coordinate
		10	Jog Y Coordinate
		11	Jog Z Coordinate

Register	Register Label Name	Bit	Function
		12	Jog Joint 4
		13	Jog Joint 5
		14	Jog Joint 6
		15 and 16	Not Used

Table 4-4: Registers 007-010: Task Jog

Bit 1: Mode: Continuous/nStep

This bit is used to select the jog mode for both coordinated and single-axis jogging. The mode takes effect when the next jog is started with a transition on the jog forward or jog reverse bit.

0 = nStep jogging. Motion stops after the large or small distance is reached, or when the jog bit is set to 0.

1 = continuous jogging. Motion stops when the travel limit is reached on an axis or when the jog bit is set to 0.

Bit 2 and 3: Coordinated Jog Forward and Reverse

Coordinated Motion

A low-to-high (0-1) transition on this bit while a task is in manual mode causes motion to start in the positive (Bit 2) or negative (Bit 3) direction for the coordinate selected in bits 9 to 14. A high-to-low (1-0) transition immediately stops the motion.

Bits 4 and 5: Jog Type

Bits 4 and 5 select the type of coordinated jogging for the next jog motion.

World Jog: jogs the axes in the world coordinate selected by bits 9, 10, and 11.

Bit 5	Bit 4	Jog Type Selected
0	0	World Jog
0	1	Joint Jog
1	1	not used

Fig. 4-2: Task Jog Type

Note: Selecting an invalid jog type will issue a warning message.

Bit 6: Distance/Speed (Large/nSmall, Fast/nSlow)

When bit 6 is set to 0 before a continuous jog, the slow jog speed is selected. For an incremental jog, the small distance and slow speed are selected.

When bit 6 is set to 1 before a continuous jog, the fast jog speed is selected. For an incremental jog, the large distance and fast speed are selected.

Bits 9-14: Jog Coordinate and Joint

These bits select the coordinate or joint that will be jogged when the jog forward or jog reverse bits are activated. Only one of these bits should be set, since jogging is allowed in only one coordinate at a time.

Bit number	World and Tool Jog	Joint Jog
9	X Coordinate	Joint 1
10	Y Coordinate	Joint 2
11	Z Coordinate	Joint 3
12		Joint 4
13		Joint 5
14		Joint 6

When jogging in world coordinates, motion will be generated parallel to the selected X, Y, or Z coordinate according to bits 9 through 11. For example, setting bit 9 high and bits 10 and 11 low enables jogging parallel to the X-axis.

For jogging of individual joints, the axis to jog is selected in bits 9 through 14. For example, if axes 2, 3, and 4 are used in coordinated motion, bit 9 selects axis 2, and bit 11 selects axis 4. The jog speed used is a percent of maximum axis velocity.

If an invalid jog type is selected, a warning message is issued.

4.6 Registers 011-018, 209-240: Axis Control

Registers 011-018 are used for Axes 1-8. Registers 209-240 are used for Axes 9-40. These Registers are reserved for axis control. These axes correspond to the SERCOS drive address (DSS02.1 for DIAX04 and programming module for ECODE03) on the fiber-optic communication loop. Any drive-controlled axis (single-axis, velocity) may be jogged independently when its associated task is in manual mode.

Jog mode (continuous or incremental) and jog speed or distance must be selected in the Task Jog Register for the task associated with the axis. Motion starts with a low-to-high transition on either the jog forward or the jog reverse bit. Motion stops when both jog bits are low, when both jog bits are high, when the task mode selection changes (e.g., from manual to automatic mode), or when a travel limit or incremental distance has been reached.

Register	Register Label Name	Bit	Function
011-018 and 209-240	Axis 1-8 and 9-40 Control	1	Disable Axis
		2	Jog Forward
		3	Jog Reverse
		4	Synchronized Jog
		5	reserved for future development
		:	:
:	:		
		16	

Table 4-5: Registers 011-018: Axis Control

Bit 1: Disable Axis

When set to (1), all axis motion is disabled based on the set *Drive Disable Method* of axis parameter A-0-0004, bit 12:

- When A-0-0004, bit 12 is set to (0), the drive stops the motor according to the value in P-0-0119. By default P-0-0119 = 0; the drive immediately commands the motor to zero velocity using full torque

(limited by S-0-0092) before disabling torque and applying the brake (if equipped).

- When A-0-0004, bit 12 is set to (1), the drive immediately disables torque, causing the motor to coast to a stop using its own inertia.

When set to (0), motion is enabled, provided other conditions allow it and there are no errors on the drive or the control. The axis resumes any motion commands from the control.

Bit 2 and 3: Jog Forward (bit 2) and Reverse (bit 3)

A low-to-high (0-1) transition on this bit while an axis is in manual mode causes motion to start in the positive (bit 2) or negative (bit 3) direction. A high-to-low (1-0) transition immediately stops motion.

Motion is stopped when both jog bits are low, when both jog bits are high, when the task mode selection changes, or when a travel limit or incremental distance has been reached. Jog mode (continuous or incremental) and jog speed or distance is selected in the Task Jog Register for the task associated with the axis.

Bit 4: Synchronized Jog

When set to (1) the following cases are enabled:

- For Ratio Mode, the axis will remain synchronized to its "master".
- For Torque Mode / Torque Following Mode, the axis will be switched to Velocity mode.

4.7 Register 019: Fieldbus/PLC Status

Register 019 holds the information for "Fieldbus Status." The register information can be referenced in a VisualMotion application program to respond to the status of each bit. The use of these bits is application-dependent. Table 4-6 below contains the bit assignment for the diagnostic object 5ff2.

Register	Register Label Name	Bit	Function
019	Fieldbus/PLC Status	1	FB Init. OK, LSB
		2	FB Init. OK, MSB
		3	Not Used
		4	FB Slave Ready
		5	Non Cyclic Ready
		6	Not Used
		7 -12	Not Used
		13	Not Used
		14	Register Data Valid
		15	Cyclic Data Valid
		16	Not Used

Table 4-6: Register 019: Fieldbus/PLC Status

Bit 1 and 2: Fieldbus Initialization OK; LSB and MSB

Status bits for the internal DPR (Dual-Port RAM) communication between the Fieldbus slave and the PPC-R:

Bit 1: FB Init. OK, LSB (least significant bit)

Bit 2: FB Init. OK, MSB (most significant bit)

Bit 2 (Control)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, or neither the PPC-R nor the fieldbus card has initialized the DPR.
0	1	The DPR is initialized by the fieldbus card, but not yet by the PPC-R.
1	0	The DPR initialization is complete. DPR has been initialized by the fieldbus card and PPC-R. Fieldbus to PPC-R communications system is ready.
1	1	Fieldbus to PPC-R communications system is ready.

Table 4-7: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

Bit 4: Fieldbus Slave Ready, LSB

Status bit for the active bus capabilities of the fieldbus slave (FB Slave Ready). This bit is monitored for the Fieldbus Error Reaction. Whenever this bit goes to 0 after a fieldbus card was initially found by the PPC-R, the selected Error Reaction (system shutdown, error message, or ignore) is initiated.

0 --> The fieldbus slave is not (yet) ready for data exchange.

1 --> The fieldbus slave can actively participate on the bus.

Bit 5: Non Cyclic Ready

Status bit for the non-cyclic channel (Parameter Channel):

0 --> The cyclic channel (Parameter Channel) cannot (yet) be used.

1 --> The non-cyclic channel (Parameter Channel) is ready for use by the fieldbus master.

Bit 14: Register Data Valid

Status bit for the register data output:

0 --> The register channel data (from PLC to PPC) are invalid.

1 --> The register channel data (from PLC to PPC) are valid.

Bit 15: Cyclic Data Valid

Status bit for the cyclic data output:

0 --> The cyclic data outputs (coming in to the PPC-R) are INVALID.

1 --> The cyclic data outputs (coming in to the PPC-R) are VALID. The system looks for this bit to be 1 before allowing data transfer.

4.8 Register 020: Fieldbus/PLC Diagnostics

Register 020 holds the information for "Fieldbus Diagnostics." Table 4-8 below contains the bit assignment for the diagnostic object 5ff0.

Register	Register Label Name	Bit	Function
020	Fieldbus/PLC Diagnostics	1	Not Used
		2	Not Used
		3	Not Used
		4	Not Used
		5	Not Used
		6	Not Used
		7	Not Used
		8	Not Used
		9-12	Not Used
		13	FB Type, LSB
		14	FB Type Bit 2
		15	FB Type Bit 3
		16	FB Type, MSB

Table 4-8: Register 020: Fieldbus Diagnostics

Bits 13- 16: Fieldbus/PLC Type

These bits identify the type of fieldbus interface card detected by VisualMotion. The bit combinations for Bits 13, 14, 15 and 16 are as listed in the table below.

Bit 16	Bit 15	Bit 14	Bit 13	Fieldbus Type
0	0	0	0	<NO CARD>
0	0	0	1	<Not Defined>
0	0	1	0	Interbus
0	0	1	1	DeviceNet
0	1	0	0	Profibus
0	1	0	1	ControlNet
0	1	1	0	<Not Defined>
0	1	1	1	Ethernet/IP (10 MB and 100MB)
1	1	1	1	Bosch Rexroth PLC Interface

Table 4-9: Identification of the Fieldbus Interface

4.9 Register 021: System Status

Register 021 is a read-only register dedicated to system status. The status is indicated by a high level in the appropriate bit.

Register	Register Label Name	Bit	Function
021	System Status	1	Parameter Mode
		2 and 3	Not Used
		4	Service Channel Ready
		5	Error
		6	Error Active
		7	Warning Active
		8	Not Used
		9	Active Program, LSB
		10	Active Program Bit 2
		11	Active Program Bit 3
		12	Active Program, MSB
		13	TP Password Active
		14	Teach Pendant
		15 and 16	Not Used

Table 4-10: Register 021: System Status

Bit 1: Parameter Mode/ Initializing

0 = Run Mode, 1 = Parameter Mode or Initializing System

If this bit is (1), the control is in parameter mode or the system is being initialized into run mode. If parameter mode is selected in System Control bit 1, the drives are in phase 2, access to restricted parameters is allowed, and the user programs are stopped.

If this bit is (0), the control is in run mode, and the user program is ready for operation if there are no errors.

Bit 4: Service Channel Ready

This bit can be checked by a user interface before initiating communication with a drive. When set to (0), the SERCOS ring is disconnected or phases are being switched. When set to (1), the drives are ready for service channel communication.

Bit 5: Error

This is the global error indicator for the VisualMotion system. If the system, any task, or any drive has an error, this bit is set to (1). If there are no errors present, it is set to (0).

Bit 6: Error Active

This bit is set to (1) when the current diagnostic is Fatal or Non-Fatal.

Bit 7: Warning Active

This bit is set to (1) when the current diagnostic is a Warning.

Bits 9-12: Active Program

These bits represent the currently active program as a binary program number. (Bit 12= most significant bit)

Example The bits below indicate that program 5 is active.

Bit 9: 1 Bit 10: 0
Bit 11: 1 Bit 12: 0

Bit 13: Teach Pendant Password Active

This bit is set if the teach pendant password is active. It allows the user program or I/O Mapper to disable functions while the corresponding functions are disabled in the teach pendant.

Bit 14: Teach Pendant Connected

This bit is set if the teach pendant is connected.

4.10 Registers 022-025: Task Status

Registers 022, 023, 024 and 025 are dedicated to task status for Tasks A, B, C and D respectively. The condition of the task status registers may be read by a user task program to determine the state of the system and the task's program execution.

Register	Register Label Name	Bit	Function
022	Task A Status	1	Mode Automatic nManual
023	Task B Status	2	Coordinated Running
024	Task C Status	3	Not Used
025	Task D Status	4	Single Step
		5	Task Error
		6	Task Running
		7	Not Used
		8	Coordinate In Position
		9	Not Used
		10	Trace Ready
		11	Breakpoint Reached
		12 – 14	Not Used
		15	Coordinated Fast Stop
		16	Not Used

Table 4-11: Register 022-025: Task Status

Bit 1: Mode: Automatic/Manual

This bit is set to (1) by the control when the task is in automatic mode and is ready for the program to be started. It is set to (0) when manual mode is selected, an error is preventing the task from starting, or the task has not been initialized into automatic mode.

Bit 2: Coordinated Running

This bit is set to (1) by the control when coordinated motion is ongoing in the task. When motion stops, the bit is reset to 0.

Bit 4: Single Stepping

When this bit is set to (1), the program flow has been stopped after an instruction, sequencer function, or sequencer step in single-step mode. To resume program flow, a (0-1) transition on the cycle start bit is required.

Bit 5: Task Error

This bit is set to (1) by the control when a task has an error or warning condition. An error is shown in parameter T-0-0122. The bit is (0) when no errors exist in this task.

Bit 6: Task Running

This bit is set to (1) by the control when the user task is executing program instructions, either from the main task or from an event. It is (0) when the task is not running.

Bit 8: Coordinate In Position

This bit is set to (1) when a coordinate segment or path has reached its end position and stops.

Bit 11: Breakpoint Reached

When this bit is set to (1), the breakpoint enabled in task parameter T-0-0137, activated with Task Control Register bit 11, has been reached. Program flow has been stopped. To resume program flow, a low-to-high (0-1) transition on the cycle start bit is required.

Bit 15: Coordinated Fast Stop

This bit is set to (1) after a coordinated fast stop is performed and the task motion comes to a complete stop using task parameter values.

4.11 Register 027: Initialization Task Control

The register is reserved for future use as a control register for the Initialization task.

4.12 Register 028: Initialization Task Status

The register is reserved for future use as a control register for the Initialization task.

4.13 Registers 031-038, 309-340: Axis Status

Registers 031-038 are for axes 1-8. Registers 309-340 (read only) for Axes 9-40. These registers are reserved for control axis status. These axes correspond to the SERCOS drive address on the fiber-optic communication loop.

Register	Register Label Name	Bit	Function
031-038 and 309-340	Axis 1-8 and 9-40 Status	1	Multiplex Channel Enabled
		2	Jogging Forward
		3	Jogging Reverse
		4	Phase Adjusted
		5	ELS Enabled
		6	ELS Secondary Mode
		7	Axis In-Position
		8	Axis Aligned
		9	Not Used
		10	Axis Stopped / Axis Present on Ring
		11	Axis Halted
		12	Class 3 Status
		13	Class 2 Warning
		14	Shutdown Error
		15	Drive Ready LSB
		16	Drive Ready MSB

Table 4-12: Registers 031-038: Axis Status

Bit 1: Multiplex Channel Enabled

This status bit is set to (1) if the SERCOS multiplex channel is enabled. The SERCOS multiplex channel can be enabled by either the selection of the Drive PLS Fast Write feature (axis parameter A-0-0004 Axis Options bit 8) or the automatic detection of the AT or MDT exceeding the 16 byte limit of the DKC2.3 drive.

Bits 2 and 3: Jogging Forward (Bit 2) and Reverse (Bit 3)

This status bit is set to (1) when the axis is jogging forward (Bit 2) or reverse (Bit 3) through either the teach pendant or the I/O bits. Otherwise, it is set to (0).

Bit 4: Phase Adjusted

For ELS and Cam axes, this bit indicates if a phase adjustment move has been completed. During the phase adjust, it is set to (0), indicating the adjustment is in progress. When the phase adjust is complete, or when first synchronizing, this bit is set to (1).

Bit 5: ELS Enabled

When this bit is set to (1), the axis is in ELS or cam mode. When set to (0), it is in velocity or single-axis mode, as indicated by bit 6.

Bit 6: ELS Secondary Mode

For ELS or cam axes, this indicates the current secondary mode that is enabled when ELS/cam is disabled. If it is set to (0), the axis is in single-axis mode. If it is set to (1), the axis is in velocity mode.

Bit 7: Single Axis in Position

Bit 7 is set to one (1) when the target position is reached and the axis is at zero velocity in single-axis mode. The drive associated with the axis must have its in-position window and zero velocity window parameters set correctly for this bit to function properly.

Bit 8: Axis Aligned (Control cam axes only)

This bit provides the status of control based cam alignment. It is set to (1) if the axis is aligned to the cam, and (0) if it is not aligned. This allows user program logic to determine if an alignment move or phase offset is needed. This bit is only for control cam axes. It is not checked for Drive Cams.

The following conditions set this bit to (0):

- The axis is not configured in the program to be a cam axis.
- A valid cam is not active for this axis.
- The absolute value of (position of the axis - slave position from cam equation) is greater than the in-position window (drive parameter S-0-0057).

The following conditions set this bit to (1):

- The axis is synchronized to the master
- The absolute value of (position of the axis - slave position from cam equation) is less than or equal to the in-position window (drive parameter S-0-0057).

Bit 10: Axis Stopped / Axis Present on Ring

This bit corresponds to Drive Parameter S-0-0182, bit 1.

During runtime (phase 4), this bit is set to (1) when the feedback velocity is less than the drive's zero velocity window.

In Parameter mode, this bit is set to (1) to indicate that the axis is present on the drive's SERCOS ring.

Bit 11: Axis Halted

Bit 11 is set to one (1) when the drive's restart/Inhalt bit is set to zero (0) (drive halted) and the axis is at zero velocity in single-axis mode.

Bit 12: Class 3 Status

This bit corresponds to the Change in Class 3 Diagnostics bit in the drive status word. When the diagnostic condition changes, the drive changes this bit from a (0-1). The program can then check this register bit for a (0-1) transition, instead of continually reading the status parameter through the service channel.

Bit 13: Class 2 Warning

This bit indicates that a Class 2 warning condition exists. It is set (1) when a warning occurs and is not cleared (0) until the warning is cleared. Warnings are often temporary conditions. This bit allows the program to latch and take action on a warning.

Bit 14: Drive Shutdown Error

Bit 14 is set to one (1) when there is a Class 1 Diagnostic Shutdown Error in the drive. This bit corresponds to the same bit in the drive's SERCOS status register.

Bit 15, 16: Ready to Operate

Bits 15 and 16 indicate when a drive is ready to operate. When both bits are one (1), the drive will respond to motion commands. These bits correspond to the drive's SERCOS status register. **See *Drive Parameter S-0-0135***.

Bit 16	Bit 15	Description	Digital Drive LED Code
0	0	Drive not ready for power up	error or P1 - P3
0	1	Drive ready for power up	bb
1	0	Drive control and power sections ready	Ab
1	1	Drive ready to operate	AH or AF

Table 4-13: Description of Bits 15 and 16 for Axis Status

4.14 Registers 040: Link Ring Status

This Link Ring status register monitors errors in Link Ring cross communication and fiber optic rings (primary and secondary) for the configured Link Ring node.

Register	Register Label Name	Bit	Function
040	Link Ring Status	1-3	Not used
		4	Link Error
		5	Error Primary Optic Ring
		6	Error Secondary Optic Ring
		7	Redundancy Loss
		8-16	Not Used

Table 4-14: Registers 040: Link Ring Status

Bit 4: Link Error

The Link Error status bit is set when a Link Ring error currently exists. All "541 Link Ring Error, see ext. diag" errors will set this bit until cleared.

Bit 5: Error in Primary Optic Ring

The error status bit is set when the node has detected a fiber optic cable break in the Link Ring's primary SERCOS ring of the node transmitting the signal. When using a double-ring structure, this status bit is set without the control being in an error state. In this situation, the node's DAQ03 card has routed around a damaged primary fiber optic cable by using the secondary ring. This bit is set to acknowledge the primary ring fiber optic break. In addition, all active Link Ring node's Redundancy Loss bits will be set to indicate that any further Link Ring fiber optic cable failures may result in a complete Link Ring failure.

Bit 6: Error in Secondary Optic Ring

The error status bit is set when the node has detected a fiber optic cable break in the Link Ring's secondary SERCOS ring of the node transmitting the signal. When using a double-ring structure, this status bit is set without the control being in an error state. In this situation, the Link Ring is most likely using the primary ring for communications so the fiber optic break that has occurred in the secondary ring does not immediately affect the Link Ring. This bit is set to acknowledge the secondary ring fiber optic break. In addition, all active Link Ring node's Redundancy Loss bits will be set to indicate that any further Link Ring fiber optic cable failures may result in a complete Link Ring failure. This bit is irrelevant on single-ring Link Rings.

Bit 7: Redundancy Loss

The Redundancy Loss bit is set to indicate that the redundant (backup) functionality of a double-ring Link Ring structure is no longer available. It infers that at least one primary or secondary fiber optic break exists in the double-ring structure. At least one node in the Link Ring will indicate a fiber optic break via its Error_Opt_Prim_Ring or Error_Opt_Sec_Ring status bits. When using a double-ring structure, this status bit is set without the control being in an error state. In this situation, the node's DAQ03 card has routed around a damaged fiber optic cable by using the double-ring's redundancy functionality. The Redundancy_Loss bit is set to acknowledge that a Link Ring fiber optic break exists and that redundancy is no longer available. It should be noted that the Redundancy_Loss bit is set simultaneously on all nodes. This bit is irrelevant on single-ring Link Rings.

4.15 Registers 041: Link Ring Data 1

This register provides a status bit for each configured Link Ring node (1-16 respectively) to verify that valid ELS Master position data is being generated in SERCOS phase 4 and is reliably communicating its data across the Link Ring. Only those nodes configured as active participants will have this functionality.

Register	Register Label Name	Bit	Function
041	Link Ring Data 1	1-16	Nodes 1-16 (respectively) Data Valid

Table 4-15: Registers 041: Link Ring Data 1

Bit 1-16: Node 1-16 Data Valid

When set to (1), the node is reliably communicating valid ELS Master position data across the Link Ring in SERCOS phase 4.

When set to (0), the node is no longer communicating valid data across the Link Ring or the node has drop from SERCOS phase 4.

Note: These bits ***do not*** verify the validity of the data being received from other Link Ring nodes. Only that each node is able to cross communicate with other nodes.

4.16 Registers 042: Link Ring Data 2

This register provides a status bit for each configured Link Ring node (17-32 respectively) to verify that valid ELS Master position data is being generated in SERCOS phase 4 and is reliably communicating its data across the Link Ring. Only those nodes configured as active participants will have this functionality.

Register	Register Label Name	Bit	Function
042	Link Ring Data 2	17-32	Nodes 17-32 (respectively) Data Valid

Table 4-16: Registers 042: Link Ring Data 2

Bit 17-32: Node 17-32 Data Valid

When set to (1), the node is reliably communicating valid ELS Master position data across the Link Ring in SERCOS phase 4.

When set to (0), the node is no longer communicating valid data across the Link Ring or the node has drop from SERCOS phase 4.

Note: These bits ***do not*** verify the validity of the data being received from other Link Ring nodes. Only that each node is able to cross communicate with other nodes.

4.17 Registers 050: Ethernet Status

This register monitors the status of the network communication of the Ethernet interface.

Register	Register Label Name	Bit	Function
050	Ethernet Status	1	Card Present
		2-8	Not Used
		9	Request Received
		10	Response Pending
		11	Response Done
		12	Response Sent
		13-15	Not Used
		16	Invalid Protocol

Table 4-17: Registers 050: Ethernet Status

Bit 1: Card Present

This bit monitors the presence of the Ethernet Interface.

0 = Ethernet interface is ***not*** present.

1 = Ethernet interface is present

Bit 9: Request Received

A (1) indicates that a request has been received and that the CIF driver function is executing.

Bit 10: Response Pending

A (1) indicates that the message received is currently being processed by the control. The response is pending on the completion of the message.

Bit 11: Response Done

A (1) indicates that the message has been processed by the control and is complete.

Bit 12: Response Sent

A (1) indicates that the response message has been sent to the DDE Server.

Bit 16: Invalid Protocol

A (1) indicates that an invalid protocol has been received (standard or encrypted ASCII).

4.18 Register 051: Standard Message Count

This register indicates the number of messages that have been received in standard ASCII protocol. The counter is incremented for any ASCII message received from any device that can communicate the ASCII protocol to the control. An ASCII message must start with a '>' and follow the standard ASCII protocol.

This register is initialized to zero on power up. Therefore, the counter will start incrementing once messages are received after the control has been powered up. If the counter is incremented, a message is processed by the control and returned to the sender. The maximum number of messages that can be counted is 65535 before the counter is reset.

4.19 Register 052: Cyphered Message Count

This register indicates the number of messages that have been received in encrypted ASCII protocol. This counter will only be incremented for the Ethernet because the encrypted ASCII protocol is only used with Ethernet. VisualMotion does not support encryption of SIS protocol messages over Ethernet.

This registers is initialized to zero on power up. Therefore, the counter will start incrementing once messages are received after the control has been powered up. If the counter is incremented, a message is processed by the control and returned to the sender. The maximum number of messages that can be counted is 65535 before the counter is reset.

4.20 Register 053: Invalid Protocol Count

This register indicates the number of messages that have been received with a protocol that can not be determined. Any device that attempts to communicate in an unknown communication protocol will cause this counter to increment. This can occur when the start character of a message did not start with a '>' (ASCII protocol) or a '02" (SIS protocol). This register is also incremented if an invalid SIS message is received. In this case the control will respond with the proper SIS error message. For all other cases, the control will not respond with an error, rather it will ignore all characters received and increment this counter accordingly.

Hardware errors will not cause this counter to increment. This counter will only increment if there has been no error detected from the device hardware and an invalid unrecognized message is received.

This registers is initialized to zero on power up. The maximum number of messages that can be counted is 65535 before the counter is reset and will start counting from zero again.

4.21 Register 054: SIS Message Count

This register indicates the number of messages that have been received in the SIS protocol. A SIS message must start with a '02' and follow the standard SIS protocol. This counter is only incremented if a valid SIS message is received by the control. All valid messages from any device that can communicate the SIS protocol to the control will increment the counter. If an invalid SIS message is received then register 53 will be incremented.

This registers is initialized to zero on power up. Therefore, the counter will start incrementing once messages are received after the control has been powered up. If the counter is incremented, a message is processed by the control and returned to the sender. The maximum number of messages that can be counted is 65535 before the counter is reset.

4.22 Registers 086: PMG Control

This register is used to enable the Position Monitoring Groups and set an offset for deviation monitoring between axis in a group.

Register	Register Label Name	Bit	Function
086	PMG Control	1-8	Enables Position Monitoring Group
		9-16	Calculates necessary offset of axis in groups 1-8 respectively to achieve 0 deviation.

Table 4-18: Registers 086: PMG Control

Bit 1-8: PMG#_ENABLE

These bits are used to enable the Position Monitoring feature for groups 1-8 respectively. A (0 ⇒ 1) transition, enables the PMG for the selected group.

Bit 9-16: PMG#_CALC_OFFSET

These bits are used to calculate the necessary position offset to achieve a 0 deviation between slave axes and the primary axis in groups 1-8, respectively. The calculated offset values are written to the following control parameters:

- C-0-3203 for group 1
- C-0-3213 for group 2
- C-0-3223 for group 3
- C-0-3233 for group 4
- C-0-3243 for group 5
- C-0-3253 for group 6
- C-0-3263 for group 7
- C-0-3273 for group 8

A (0 ⇒ 1) transition of these bits calculates the offset difference between the assigned slave axes and the primary axis. This value is stored in RAM memory until the PMG_ENABLE bit is set for the same group.

Note: If an offset value is modified, the PGM_ENABLE bit for the group must be set again in order for the offset to take affect.

4.23 Registers 087: PMG Status

This register displays the status of the PMG#_ENABLE and PMG#_ERROR condition.

Register	Register Label Name	Bit	Function
087	PMG Status	1-8	Position Monitoring Group Enabled
		9-16	Error reaction indication

Table 4-19: Registers 087: PMG Status

Bit 1-8: PMG#_ENABLED

These bits display the status of the PMG_ENABLE bits of control register 86. When set to 1, the Position Monitoring feature for the selected group has been enabled.

Bit 9-16: PMG#_ERROR

When set to 1, an error was generated from the Position Monitoring Group feature.

4.24 Registers 088 and 089: Task A Extend Event Control

Register 88 (USER_XI_REG) is used to trigger up to 16 events in Task A. Register 89 (USER_XO_REG) is used to monitor the status of events triggered by Register 88.

Together they provide a time critical way to control motion on the control without time wasting polling loops. These events are similar to the Task Input Transition, located in each task control register, but are limited to the task with the highest priority (Task A).

Register Operation

At power up and at system reset the output register USER_XO_REG is cleared.

The input register USER_XI_REG triggers an EUI on positive edge bit transitions. The event function assigned to the bit is executed at that time. Only positive going edges in USER_XI_REG can trigger an EUI.

The output register USER_XO_REG is effected by the Event Setup Box Icon arm and disarm functions as well as by bit transitions within USER_XI_REG.

When an EUI is armed through the Event Setup Box Icon, its bit is set in USER_XO_REG. This provides an external output that indicates that the EUI is armed and ready for operation. Likewise, it is reset if disarmed through the icon.

When an EUI is triggered, its corresponding output bit in USER_XO_REG is reset (normally it is set) providing an external output indicating that the event is active. It remains reset so long as no high-to-low (1-0) transition on the corresponding input bit in USER_XI_REG detected. When a high-to-low (1-0) transition on the input bit is detected in USER_XI_REG, the output bit in USER_XO_REG is again set indicating the event is armed and ready for another positive going edge.

4.25 Registers 090 and 091: Latch and Unlatch

Register 090 provides 16 latches that can be set to (1) directly or by the I/O Mapper. The bits in register 090 can be reset to (0) only by writing a (1) to the corresponding bit in register 091. Register 091 immediately clears its bits to (0), so that a transition is not needed on the unlatch bits.

4.26 Registers 092-094: Mask Pendant Key Functionality

The bits in registers 092-094 “mask” the functionality of the corresponding bits in registers 095-097. This means that if a bit is set to 1 (on) in register 092, 093 or 094, the BTC06 key controlled by the corresponding bit in register 095, 096 or 097 is not operational. (Register 092 masks register 095, register 093 masks register 096 and register 094 masks register 097.) This masking feature allows the programmer to redefine the action of a key or prevent the action normally mapped to that key.

Examples Register 092, bit 1 is set to 1. Pressing the F1 key (status given in register 095, bit 1) does not result in the expected action.

Register 093, bit 7 is set to 1. Pressing the X+ key (status given in register 096, bit 7) does not result in jogging in the Jog Menu.

Note: When a key on the BTC06 is masked, a key press still sends an acknowledgement to the status register bit, although no action occurs. This means that the key can be mapped to perform a different function, or the usual function can be prevented.

Note: If an F-key is masked off, the action assigned to it will not appear at the bottom of the BTC06 screen of the Control Menu.

4.27 Registers 095-097: BTC06 Teach Pendant Status

The bits in these registers are set when the corresponding keys are pressed on the teach pendant. They can be scanned in the user program or I/O Mapper to detect system operations or to extend teach pendant functionality to control the user program.

Note: These bits are all read-only.

The BTC06 keyboard is mapped to register 095, 096 and 097. The figure below and to the right outlines the register and bit location in the following format:

Register - Bit

Example: **95 – 01**, key is mapped to register 095, bit 01

When a key is pressed its corresponding bit turns on and remains on for as long as the key is pressed.

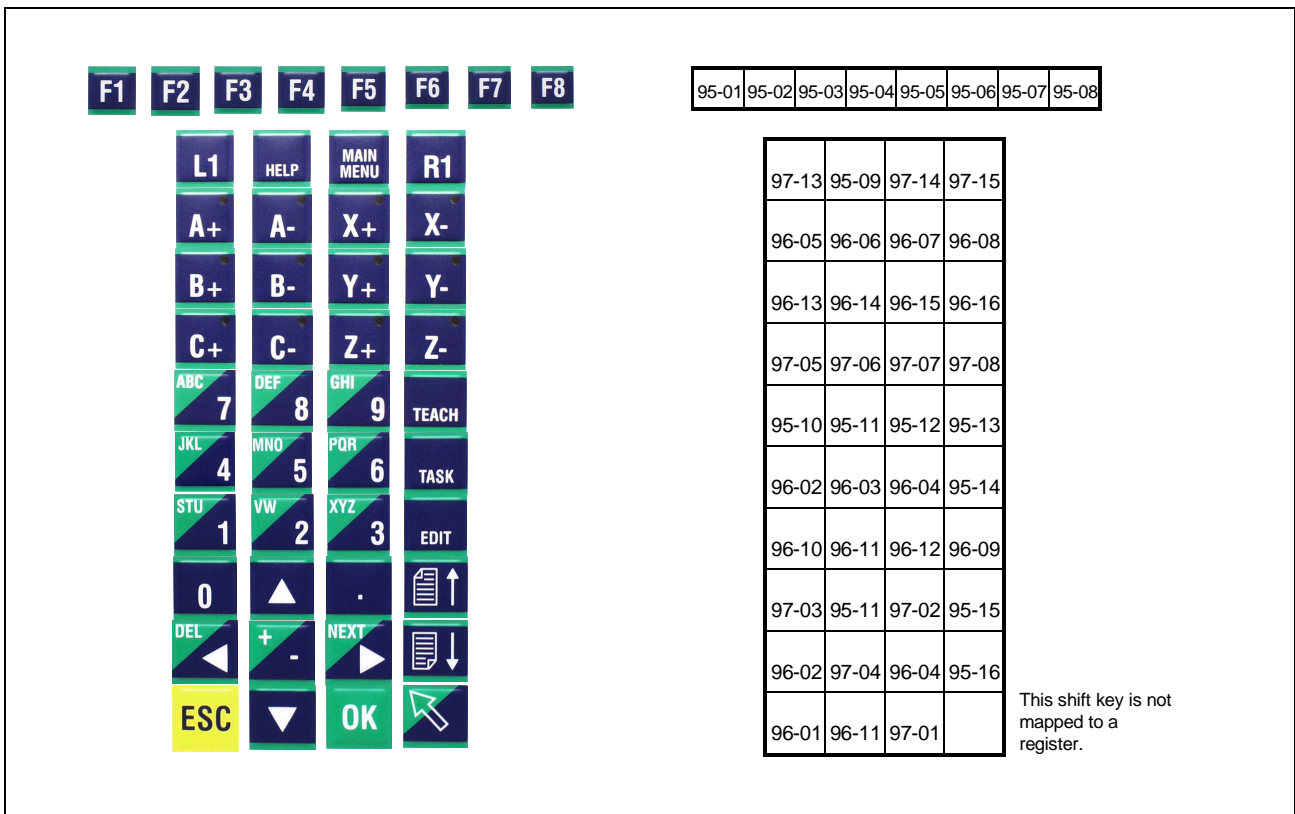


Fig. 4-3: Teach Pendant: Function Keys to Register Map

4.28 Registers 098: Pendant Control - Task A, B

The bits in this register can disable teach pendant control of the selected function for the corresponding Tasks A-B.

Register	Register Label Name	Bit	Function
098	Pendant Control – Task A-B	1	Block Task A Manual
		2	Block Task A Auto
		3	Block Task A Step
		4	Block Task A Jog
		5	Block Task A Entry
		6	Block Task A Teach
		7 and 8	Not Used
		9	Block Task B Manual
		10	Block Task B Auto
		11	Block Task B Step
		12	Block Task B Jog
		13	Block Task B Entry
		14	Block Task B Teach
		15 and 16	Not Used

Table 4-20: Register 098: Pendant Control – Task A-B

4.29 Registers 099: Pendant Control - Task C-D

The bits in this register can disable teach pendant control of the selected function for the corresponding Tasks C-D.

Register	Register Label Name	Bit	Function
099	Pendant Control – Task C-D	1	Block Task C Manual
		2	Block Task C Auto
		3	Block Task C Step
		4	Block Task C Jog
		5	Block Task C Entry
		6	Block Task C Teach
		7 and 8	Not Used
		9	Block Task D Manual
		10	Block Task D Auto
		11	Block Task D Step
		12	Block Task D Jog
		13	Block Task D Entry
		14	Block Task D Teach
		15 and 16	Not Used

Table 4-21: Register 099: Pendant Control – Task C-D

4.30 Register 140 ELS Master Control

This register is used to enable slip monitoring between two ELS System Masters. Slip monitoring is configured in the *Slip Monitor Setup* window that is activated by selecting the **Slip Monitor Setup...** button in the ELS System Master icon.

Note: This register number can be assigned by VisualMotion as the default system register when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default number. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
140	ELS Master Control	1-6	Reserved
		7	Set ELS Master 1 Reference Position
		8	Set ELS Master 2 Reference Position
		9	Set ELS Master 3 Reference Position
		10	Set ELS Master 4 Reference Position
		11	Set ELS Master 5 Reference Position
		12	Set ELS Master 6 Reference Position
		13-14	Reserved
		15	Capture Slip Monitoring
		16	Enable Slip Monitoring

Table 4-22: Register 140: ELS Mater Control

Bits 7- 12: Set ELS Master (1-6) Reference Position

When a secondary feedback device is used as an ELS Master (Real Master), it is sometimes necessary to reference the ELS Master output position (ELS_MSTR_POS#) using the ELS Master reference position (ELS_MSTR_REF_POS#). A low-to-high (0-1) transition of this bit, while in phase 4, copies the ELS Master reference position to the ELS Master output position.

Note: To ensure accuracy and avoid any sudden movement of an ELS Group, following the ELS Master, this command should only be activated when no motion is present.

Bit 15: Capture Slip Monitoring

This bit is used to capture the value of the maximum allowed deviation window variable (ELS_MSTR_SLIP_WINDOW), and the master position offset variable (ELS_MSTR_SLIP_OFFSET) when two ELS System Masters are configured for slip monitoring. The value of ELS_MSTR_SLIP_OFFSET is calculated if the **Master Position Offset** selection is set to **Dynamically reset phase offset on system phase up** in the *Slip Monitor Setup* window. This is the default setting.

A low-to-high (0-1) transition of this bit captures and writes the maximum allowed deviation window and master position offset variable values to the control's memory during run-time.

Note: If the ELS_MSTR_SLIP_WINDOW variable is modified from its initial value, this bit must transitioned from 0-1 for the new value to take effect.

Bit 16: Enable Slip Monitoring

This bit enables or disables slip monitoring between two ELS System Masters.

0 = Disable slip monitoring

1 = Enable slip monitoring

4.31 Register 141 ELS Master Status

This register monitors the maximum position difference using two master encoder signals.

Note: This register number can be assigned by VisualMotion as the default system register when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default number. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
141	ELS Master Status	1	Master 1 at Standstill
		2	Master 2 at Standstill
		3	Master 3 at Standstill
		4	Master 4 at Standstill
		5	Master 5 at Standstill
		6	Master 6 at Standstill
		7	ELS Master 1 Position Referenced
		8	ELS Master 2 Position Referenced
		9	ELS Master 3 Position Referenced
		10	ELS Master 4 Position Referenced
		11	ELS Master 5 Position Referenced
		12	ELS Master 6 Position Referenced
		13	Reserved
		14	Monitoring ERROR Active
		15	Lead Encoder
		16	Slip Monitoring Enabled

Table 4-23: Register 141: ELS Master Status

Bit 1-6: Master at Standstill

The *Master at Standstill* bit is set to 1 when the master's output velocity value is at or below the Slip Masters Velocity Threshold variable (ELS_MSTR_STANSTILL) for two SERCOS cycles.

Bits 7- 8: ELS Master (1-6) Position Referenced

This bit is set to 1 when the ELS Master reference position has been copied to the ELS Master output position. The set command is activated using control register 140, ELS Master Control, bits 1-6.

0 = ELS Master **not** referenced

1 = ELS Master referenced

Bit 14: Monitoring ERROR Active

This bit is set to 1 when a slip monitoring error is encountered. Perform the following steps to clear this error:

1. Disable slip monitoring (ELS Master control register, bit 16 = 0).
2. Transition the Clear All Error bit from 0-1(Register 001, Bit 5).

Bit 15: Lead Encoder

This bit is used when the primary and secondary masters used in slip monitoring are Real Masters. Its used to indicate which master caused a slip monitoring error.

0 = Primary Master

1 = Secondary Master

Bit 16: Enable Slip Monitoring

This bit monitors the status of the Enable Slip Monitoring bit.

0 = Slip monitoring disabled

1 = Slip monitoring enabled

4.32 Registers 150 and 151: Virtual Master 1 & 2 Control

GPP 9 supports two Virtual Masters. A Virtual Master is an internal motion engine with an independent set of control parameters. Each Virtual Master can be used independently from each other.

A Virtual Master is controlled by the VisualMotion user program created with VisualMotion Toolkit (VMT) and/or a PLC using I/O registers and program variables. The initialization of these registers and program variables is defined in the Virtual Master icon.

Note: These register numbers can be assigned by VisualMotion as default system registers when creating an icon program or can be changed to any register number not reserved by VisualMotion. It is strongly recommend that the programmer use the default numbers. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
150 and 151	Virtual Master 1and 2 Control	1	VM#_CT_FSTOP
		2	VM#_CT_HOME
		3	VM#_CT_GO
		4	VM#_CT_VMODE
		5	VM#_CT_RELMODE
		6	VM#_CT_RELTRIG
		7-16	Not Used

Table 4-24: Registers 150-151: Virtual Master 1 & 2 Control

Bit 1: Virtual Master 1or 2 Fast Stop

A low-to-high (0-1) transition decelerates the Virtual Master using the programmed E-Stop deceleration variable (VM#_E_STOP_DECEL). No motion is possible while this bit remains high (1).

Bit 2: Virtual Master Home Position

A low-to-high (0-1) transition returns the Virtual Master to the programmed home position in the VM#_HOME_POS variable.

Bit 3: Virtual Master Go Command

A low-to-high (0-1) transition starts the selected Virtual Master. A high-to-low (1-0) transition stops the Virtual Master.

Bit 4: Virtual Master Mode of Operation

Each Virtual Master supports two modes of operation:

- 0 = Positioning Mode
- 1 = Velocity Mode

Velocity Mode

In velocity mode, the Virtual Master accelerates to the programmed value in velocity variable (VM#_CMD_VEL). A negative value in the velocity variable will cause the Virtual Master to move in the opposite direction. Additionally, it can stop at a predetermined stop position between 0 and 360 degree set by the VM#_STOP_POS variable.

When bit 4 (VM#_CT_VMODE) is set to 1, a low-to-high (0-1) transition of bit 3 (VM#_CT_GO) will trigger the Virtual Master to accelerate to the programmed velocity variable (VM#_CMD_VEL). A high-to-low (1-0) transition of bit 3 (VM#_CT_GO) will cause an immediate stop using the programmed deceleration variable (VM#_CMD_DEC).

A high-to-low (1-0) transition of bit 4 (VM#_CT_VMODE) with bit 5 (VM#_CT_RELMODE) set to 0 and bit 3 (VM#_CT_GO) set to 1 will cause a stop at the stop position variable (VM#_STOP_POS). VM#_STOP_POS will then overwrite VM#_CMD_ABS_POS.

A high-to-low (1-0) transition of bit 4 (VM#_CT_VMODE) with bit 5 (VM#_CT_RELMODE) set to 0 and bit 3 (VM#_CT_GO) set to 0 will initialize the absolute positioning mode to the current position. It will replace the value in variable VM#_CMD_ABS_POS with the value of variable VM#_STOP_POS. A subsequent transition from 0-1 of bit 3 (VM#_CT_GO) will complete the programmed move of variable VM#_CMD_ABS_POS.

A high-to-low (1-0) transition of bit 4 (VM#_CT_VMODE) with bit 5 (VM#_CT_RELMODE) set to 1 will cause an immediate stop with the programmed deceleration variable VM#_CMD_DEC (random stop position). In this case, the state of VM#_CT_GO does not matter.

Positioning Mode

Relative Positioning

A relative positioning move of a Virtual Master is used when a specific distance is required relative to the Virtual Master's current stop position variable VM#_CUR_POS. This distance (VM#_REL_MOVE_DIST) can be less than or greater than modulo (0-360 degrees). For example, if an indexing move requires that the Virtual Master travel 100 degrees from a stopped position of 350 degrees, enter a value of 100 for VM#_REL_MOVE_DIST and set bit 6 (VM#_CT_RELTRIG) to 1.

Note: A negative value for VM#_REL_MOVE_DIST will move the Virtual Master counter clockwise.

The following bit state will increment the current position of the Virtual Master by the value in variable VM#_REL_MOVE_DIST for every low-to-high transition of bit 6.

Bit 3 VM#_CT_GO	Bit 5 VM#_CT_RELMODE	Bit 6 VM#_CT_RELTRIG
1	1	0 ⇒ 1

Fig. 4-4: Relative Positioning for Virtual Master

If during a relative move, bit 3 (VM#_CT_GO) is switched to low (1-0), the relative move will stop. The move will be continued with a low-to-high (0-1) transition of bit 3 (VM#_CT_GO).

Absolute Positioning Absolute positioning is used to position the Virtual Master between 0 and 360 degrees. The maximum travel distance with absolute positioning is 180 degree using shortest path or 359.9999 degree for positive or negative direction.

When bit 5 (VM#_CT_RELMODE) is set to 0, bit 3 (VM#_CT_GO) is set to 1 and bit 4 (VM#_CT_VMODE) is set to 0, the absolute target position in variable VM#_CMD_ABS_POS is used and every change of the target position will immediately trigger the positioning movement. If during an absolute move, bit 3 (VM#_CT_GO) is switched to 0 (1-0), the absolute move will be stopped. The move will be continued with a low-to-high (0-1) transition of bit 3 (VM#_CT_GO).

Bit 5: Virtual Master Absolute/Relative Mode

Absolute or relative position mode of the Virtual Master is set with this bit. Positioning mode of a Virtual Master is set with the following combination of bits 4 and 5.

Active Position Mode	Bit 4	Bit 5
Absolute Mode	0	0
Relative Mode	0	1

Table 4-25: Setting Virtual Master Position Mode

Bit 6: Virtual Master Relative Trigger Mode

A low-to-high (0-1) transition triggers the relative move bit. This bit is only functional when a relative mode setup is configured using bits 4 and 5. The Virtual Master will move in increments of the value in variable VM#_REL_MOVE_DIST every time this bit transitions from low-to-high.

4.33 Registers 152-159: ELS Groups 1-8 Control

Note: These register numbers can be assigned by VisualMotion as default system registers when creating an icon program or can be changed to any register number not reserved by VisualMotion. It is strongly recommend that the programmer use the default numbers. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
152-159	ELS Group 1-8 Control	1	G#_CT_LOCK_OFF
		2	G#_CT_M_REL_PH
		3	G#_CT_S_REL_PH
		4	G#_CT_MSTR_SEL
		5	G#_CT_VAR_CLK
		6	G#_CT_LOCAL
		7	G#_CT_JOG_INC
		8	G#_CT_JOG_ABS
		9	G#_CT_JOG_PLUS
		10	G#_CT_JOG_MINS
		11	G#_CT_M_ABS_PH
		12	G#_CT_S_ABS_PH
		13-16	Not Used

Table 4-26: Registers 152-159: ELS Group Control

Bit 1: Group Lock Off/ Lock on Cams

A low-to-high (0-1) transition activates the lock off cam profile. This cam profile decelerates to a stop over one cycle of the master. If the group's master requires additional cycles to stop, the cam's profile H factor is set to zero.

A high-to-low (1-0) transition activates the lock on cam profile. This profile accelerates from a stopped position to match the master's velocity over one cycle of the master (360 degrees). Once the velocity is matched, a one-to-one cam profile is active and follows the master.

Bit 2: Group Master Relative Phase Adjust

A low-to-high (0-1) transition of this bit adds a relative offset, prior to an optional group CAM profile, to the group's output position. The relative move value is taken from program variable G#_REL_M_PH. Each additional master phase adjust is added to the prior. The total or absolute master phase adjust, from the master input position, is written to program variable G#_ABS_M_PH.

Bit 3: Group Slave Relative Phase Adjust

A low-to-high (0-1) transition of this bit adds a relative offset, after an optional group master phase adjust and CAM profile, to the group's output position. The relative move value is taken from program variable G#_REL_S_PH. Each additional slave phase adjust is added to the prior.

The total or absolute slave phase adjust, from the master input position, is written to program variable G#_ABS_S_PH.

Bit 4: Group Master Input Select

This bit allows the ELS Master to be switched between the two configured masters (Virtual, Real or Group). The changeover can be done while the master is running.

When set to 0, the ELS Group and slave axes follow master 1 as selected in the user program and in the integer block variable (G#_MSTR1).

When set to 1, the ELS Group and slave axes follow master 2 as selected in the user program and in integer block variable (G#_MSTR2).

Bit 5: Group Forcing

When an ELS group is switched to local mode and the group master is at standstill, the internal variables of the group can be forced to a user defined value with a low-to-high (0-1) transition input of this forcing bit. The internal variables that can be changed through forcing are:

- internal group input master position
- group cam table input position
- ELS Group master position (only when bit 9 in ELS group configuration word is set to 1)
- state of the state machine for lock on / lock off
- absolute group master and group slave offset
- lock on, lock off and 1:1 cam profile ID number
- H-factor for lock on, lock off and 1:1 cam profile
- M and N factor

In the ELS Group configuration word bit 1, (enable jerk limiting), can only be changed in Phase 2 or through forcing. Bit 9 of the ELS Group configuration has influence on how forcing is done. The completion of forcing is indicated with bit 5, (forcing completed), of the group's status register.

Bit 6: Group Local Mode Select

A low-to-high (0-1) transition switches the group into local mode. First, the group's input master will be stopped using a stop ramp deceleration variable (G#_STOP_DECEL). After the group's input master is at standstill, the group can be jogged using bits 7-10 (group's jog engine) or group variables can be forced to a different value.

Bit 6, Group # Local Mode Status, of the group's status register indicates that forcing or jogging is possible.

A high-to-low (1-0) transition switches the group back to the selected group input master.

Bit 7: Group Jog Continuous/Incremental

This bit controls the group's ability to jog either continuous or incremental. The group must be in local mode. Bits 9 and 10 control the direction of jog.

0 = Continuous Jog

1 = Incremental Jog

Bit 8: Group Jog Absolute

This bit controls the group's ability to jog to the absolute position variable (G#_JOG_ABS). The group must be in local mode. Bits 9 and 10 control the direction of jog.

0 = no function

1 = Absolute Jog

Bit 9: Group Jog in Positive Direction

This bit jogs a group in the positive direction for the configured jog type in bits 7 or 8.

Bit 9	Bit 8	Bit 7	Description
0 ⇒ 1	0	0	Continuous jogging in the positive direction at a rate of velocity variable G#_JOG_VEL.
0 ⇒ 1	0	1	Incremental (G#_JOG_INC) jogging in the positive direction at a rate of velocity variable G#_JOG_VEL for every 0 ⇒ 1 transition of bit 9.
0 ⇒ 1	1	0	Absolute jogging to position variable G#_JOG_ABS in the positive direction at a rate of velocity variable G#_JOG_VEL. Additional 0 ⇒ 1 transitions of bit 9 will not change position unless position variable G#_JOG_ABS changes.

Table 4-27: Bit 9: Group Jog in Positive Direction.

Bit 10: Group Jog in Negative Direction

The function of this bit is the same as bit 9 but in the negative direction.

Bit 11: Group Master Absolute Phase Adjust

A low-to-high (0-1) transition of this bit adds an absolute offset (before an optional group CAM profile) to the group's output position. The absolute move value is taken from program variable G#_REL_M_PH. Additional transitions of this bit will not trigger another phase adjust unless the value in G#_REL_M_PH is modified.

The value in program variable G#_REL_M_PH is written to G#_ABS_M_PH after the phase adjust is complete.

Bit 12: Group Slave Absolute Phase Adjust

A low-to-high (0-1) transition of this bit adds an absolute offset (after an optional group master phase adjust and CAM profile) to the group's output position. The absolute move value is taken from program variable G#_REL_S_PH. Additional transitions of this bit will not trigger another phase adjust unless the value in G#_REL_S_PH is modified.

The value in program variable G#_REL_S_PH is written to G#_ABS_S_PH after the phase adjust is complete.

4.34 Register 197: Coordinated Articulation Synchronized Mode Control

A Coordinated Articulation configuration uses two control registers. This first register is used as the control register for synchronized mode. Refer to Register 198: Coordinated Articulation Local Mode Control for details

on the second register. The following table lists the functions used in this control register.

Note: This register number can be assigned by VisualMotion as a default system register when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default number. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
197	Coordinated Articulation Control	1	CA#_X_SYNC
		2	CA#_Y_SYNC
		3	CA#_Z_SYNC
		4	CA#_ROLL_SYNC
		5	CA#_PITCH_SYNC
		6	CA#_YAW_SYNC
		7	CA#_X_REL_MOD
		8	CA#_Y_REL_MOD
		9	CA#_Z_REL_MOD
		10	CA#_ROLL_REL_MOD
		11	CA#_PITCH_REL_MOD
		12	CA#_YAW_REL_MOD

Table 4-28: Register 197: Coordinated Articulation Control

Bits 1-6: Synchronized Mode

These bits are used to transition the Coordinated Articulated configuration axes between Sync and Local modes.

Sync Mode

A low-to-high (0-1) transition enables each axis (X, Y, Z, ROLL, PITCH and YAW) to Sync mode. This transition is delayed until the ramp generator's velocity is 0.

Local Mode

A high-to-low (1-0) transition enables each axis (X, Y, Z, ROLL, PITCH and YAW) to Local mode. Local mode is used for the manual positioning of each axis.

Bit 7-12: Positioning Mode

These bits are used to enable the absolute or relative offset calculation of the Coordinated Articulation CAM output section.

0 = Absolute Mode

1 = Relative Mode

4.35 Register 198: Coordinated Articulation Local Mode Control

A Coordinated Articulation configuration uses two control registers. This second register is used as the control register for local mode. Local mode allows the user to manually position the world coordinate in-position value to the current CAM output position before synchronizing. Refer to Register 197: Coordinated Articulation Synchronized Mode Control for details on the first register. The following table lists the functions used in this control register.

Note: This register number can be assigned by VisualMotion as a default system register when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default number. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
198	Coordinated Articulation Control	1	CA#_X_MAN_EN
		2	CA#_Y_MAN_EN
		3	CA#_Z_MAN_EN
		4	CA#_ROLL_MAN_EN
		5	CA#_PITCH_MAN_EN
		6	CA#_YAW_MAN_EN
		7	CA#_X_MAN_IMD
		8	CA#_Y_MAN_IMD
		9	CA#_Z_MAN_IMD
		10	CA#_ROLL_MAN_IMD
		11	CA#_PITCH_MAN_IMD
		12	CA#_YAW_MAN_IMD

Table 4-29: Register 198: Coordinated Articulation Control

Bits 1-6: Normal or Immediate Local Mode

These bits are used to transition the Coordinated Articulated configuration axes between Normal and Immediate local modes. Normal and Immediate local mode is set in bits 6-12 for each respective axes. These bits are functional when bits 1-6 of control register 197 are transitioned from 0 to 1 (Local Mode).

Normal Local Mode

A low-to-high (0-1) transition enables the ramp generator, creates a move profile for variables CA#_%_TAR_POS value to variable CA#_%_IN_POS (where # = Unit number and % = X, Y, Z, ROLL, PITCH or YAW).

A high-to-low (1-0) transition disables the ramp generator. If a move is at velocity, the velocity is ramped down using the value in variable CA#_LIN_DECEL.

Immediate Local Mode

A low-to-high (0-1) transition stores the value of variable CA#_%_TAR_POS to variable CA#_%_IN_POS.

A high-to-low (1-0) transition has no effect.

Bits 7-12: Immediate Mode

These bits are used to set a Coordinated Articulated axes' local mode positioning to Normal or Immediate.

0 = Normal Local Mode

1 = Immediate Local Mode

4.36 Registers 241 and 242: Virtual Master 1 & 2 Status

Note: These register numbers can be assigned by VisualMotion as default system registers when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default numbers. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
241 and 242	Virtual Master 1 and 2 Status	1	VM#_ST_FSTOP
		2	VM#_ST_HOME
		3	Reserve3
		4	VM#_ST_VMODE
		5	VM#_ST_RELMODE
		6	Reserve6
		7	VM#_ST_ZEROVEL
		8	VM#_ST_INPOS
		9-16	Not used

Table 4-30: Registers 241-242: Virtual Master 1 & 2 Status

Bit 1: Virtual Master 1or 2 Fast Stop Active

An active high (1) is an indication that the fast stop function for this Virtual Master is active in bit 1 of the Virtual Master control register.

Bit 2: Virtual Master Home Position

An active high (1) is an indication that the homing process for this Virtual Master is complete and at the command position in variable VM#_HOME_POS.

Bit 4: Virtual Master Mode of Operation Active

This status bit indicates which of the following modes of operation are active.

0 = Positioning Mode

1 = Velocity Mode

Bit 5: Virtual Master Absolute/Relative Mode Active

This status bit indicates whether absolute or relative position mode of the Virtual Master is active.

0 = Absolute Mode

1 = Relative Mode

Bit 7: Virtual Master at Zero Velocity

This status bit is active high (1) anytime the Virtual Master is at zero velocity.

Bit 8: Virtual Master in Position

This status bit is active high (1) when the Virtual Master has reached its commanded position. This can occur when...

- Virtual Master stop position variable VM#_STOP_POS is reached.
- Virtual Master absolute position variable VM#_CMD_ABS_POS is reached.
- Virtual Master relative distance variable VM#_REL_MOVE_DIST is reached.

4.37 Registers 243 - 250: ELS Groups 1-8 Status

Note: These register numbers can be assigned by VisualMotion as default system registers when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default numbers. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
243-250	ELS Group 1-8 Status	1	G#_ST_LOCK_ON
		2	G#_ST_MSTR_PH
		3	G#_ST_SLV_PH
		4	G#_ST_MSTR_SEL
		5	G#_ST_VAR_ACK
		6	G#_ST_LOCAL
		7 and 8	Not Used
		9	G#_ST_MOTION
		10	G#_ST_JOG_POS
		11	G#_ST_M_ABS_PH
		12	G#_ST_S_ABS_PH
		13-16	Not Used

Table 4-31: Registers 243-250: ELS Group 1-8 Status

Bit 1: Group Lock Off/ Lock On Cams Status

An active high (1) is an indication that the lock off cam profile feature in ELS is active. This cam profile decelerates to a stop over one cycle of the master.

A low (0) is an indication that the lock on cam feature is active. This profile accelerates from a stopped position to match the master's velocity over one cycle of the master (360 degrees). Once the velocity is matched, a one-to-one cam profile is active and follows the master.

Bit 2: Group Master Phase Adjust Status

A low-to-high (0-1) transition of the ELS Group control register bit 2 (G#_CT_M_REL_PH) cause a momentary transition (0-1-0) of this group master phase adjust status bit.

Bit 3: Group Slave Phase Adjust Status

A low-to-high (0-1) transition of the ELS Group control register bit 3 (G#_CT_S_REL_PH) cause a momentary transition (0-1-0) of this group master slave adjust status bit.

Bit 4: Group Master Input Select Status

A low-to-high (0-1) transition of the ELS Group control register bit 4 (G#_CT_MSTR_SEL) cause an active high of this status bit.

A 0 is an indication that the ELS Group and slave axes are following the master 1 input to the group as selected in the user program and in the integer block variable (G#_MSTR1).

A 1 is an indication that the ELS Group and slave axes are following the master 2 input to the group as selected in the user program and in integer block variable (G#_MSTR2).

Bit 5: Group Forcing Status

This status bit is active high (1) when the ELS group is switched to local mode and the group forcing bit (G#_CT_VAR_CLK) is active. This status bit is an indication that one of the following internal variables is being forced to a different value.

- internal group input master position
- group cam table input position
- ELS Group master position (only when bit 9 in ELS group configuration word is set to 1)
- state of the state machine for lock on / lock off
- absolute group master and group slave offset
- lock on, lock off and 1:1 cam profile ID number
- H-factor for lock on, lock off and 1:1 cam profile
- M and N factor

Bit 6: Group Local Mode Select

This status bit is active high (1) when the ELS Group is switched to local mode and the master input velocity is at zero. Once active high, the group's internal variables can be forced or jugged.

A high-to-low (1-0) transition of the group's control register bit 6 (G#_CT_LOCAL) switches the group back to the selected group input master and cause a high-to-low transition of this status bit.

Bit 8: Group Jog Position Status

This bit is active high (1) after the group is jogged using the absolute jog bit (G#_CT_JOG_ABS) and the position in the absolute jog variable (G#_JOG_ABS) is reached.

Bit 9: Group Motion Status

This bit is active high (1) whenever the group is in motion, whether following a master input or being jogged.

Bit 11: Group Master Absolute Phase Adjust Status

A low-to-high (0-1) transition of the ELS Group control register bit 11 (G#_CT_MSTR_PH_ABS) cause a momentary transition (0-1-0) of this group master phase adjust status bit.

Bit 12: Group Slave Absolute Phase Adjust Status

A low-to-high (0-1) transition of the ELS Group control register bit 12 (G#_CT_SLV_PH_ABS) cause a momentary transition (0-1-0) of this group slave phase adjust status bit.

4.38 Register 288: Coordinated Articulation Synchronized Mode Status

A Coordinated Articulation configuration uses two status registers. This first register is used to monitor the status of each axis enabled to synchronized mode. It also monitors the equivalence between variables CA#_%_IN_POS and CA#_%_TAR_POS (where # = Unit number and % = X, Y, Z, ROLL, PITCH or YAW). Refer to Register 289: Coordinated Articulation Local Mode Status for details on the second register. The following table lists the functions used in this status register.

Note: This register number can be assigned by VisualMotion as a default system register when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default number. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
288	Coordinated Articulation Status	1	CA#_X_SYNCED
		2	CA#_Y_SYNCED
		3	CA#_Z_SYNCED
		4	CA#_ROLL_SYNCED
		5	CA#_PITCH_SYNCED
		6	CA#_YAW_SYNCED
		7	CA#_X_READY
		8	CA#_Y_READY
		9	CA#_Z_READY
		10	CA#_ROLL_READY
		11	CA#_PITCH_READY
		12	CA#_YAW_READY

Table 4-32: Register 288: Coordinated Articulation Status

Bits 1-6: Synchronized Mode

These bits are used to indicate if a Coordinated Articulation axis is enabled to Sync or Local modes.

0 = Local Mode (using target position for world coordinated input value)

1 = Sync Mode (using CAM output world coordinated input value)

Bit 7-12: Positioning Mode

These bits are used to indicate if a Coordinated Articulation axis world coordinated input value equals the CAM output value.

0 = Not ready for synchronization

1 = CAM output value (CA#_%_CAM_OUT) equals world coordinated input value (CA#_%_IN_POS) and that the ramp generator is at 0 velocity.

4.39 Register 289: Coordinated Articulation Local Mode Status

A Coordinated Articulation configuration uses two status registers. This second register is used to monitor if a Coordinated Articulation axis is at position when moving the axis in local mode. Refer to Register 288: Coordinated Articulation Synchronized Mode Status for details on the first register. The following table lists the functions used in this status register.

Note: This register number can be assigned by VisualMotion as a default system register when creating an icon program or can be changed to any number not reserved by VisualMotion. It is strongly recommend that the programmer use the default number. This makes documentation and modification to user programs an easier task over the scope of the project.

Register	Register Label Name	Bit	Function
289	Coordinated Articulation Status	1	CA#_X_AT_TAR_POS
		2	CA#_Y_AT_TAR_POS
		3	CA#_Z_AT_TAR_POS
		4	CA#_ROLL_AT_TAR_POS
		5	CA#_PITCH_AT_TAR_POS
		6	CA#_YAW_AT_TAR_POS

Table 4-33: Register 289: Coordinated Articulation Status

Bits 1-6: Local Mode At Target Position

These bits are set to 1 when a Coordinated Articulated axis target position (CA#_%_TAR_POS) variable is equal to its respective in position (CA#_%_IN_POS) variable.

0 = Refer to Note

1 = CA#_%_IN_POS at CA#_%_TAR_POS position and ramp generator is at 0 velocity.

Note: This bit is not set to 1 if the ramp generator move is interrupted (dropped Enable, user task stopped, etc.) before the target position is reached. In such an instance, it is not possible to 'force' the bit to 1 by setting the ramp generator's target value equal to the current input value. Rather, another ramp generator move must be completed before this bit is set to 1.

5 Parameters

5.1 Overview

Bosch Rexroth's VisualMotion controls provide parameters to adapt to a specific application. Basic communication and initialization parameters for the control and digital drives must be entered before it is possible to operate or program a VisualMotion system.

System-builders must modify certain parameters describing the mechanical characteristics of their unique application. Parameters specify machine limitations, such as maximum velocity and acceleration. Other parameters are used to specify the mechanical characteristics of the system, such as the ratio between motor revolutions and shaft rotation and slide travel.

VisualMotion structures parameters into four classes:

- Control (C-parameters)
- Task (T-parameters)
- Axis (A-parameters)
- Drive (S-standard SERCOS and P-product specific)

Control parameters contain setup and status parameters for the control. Task and Axis parameters are associated with the user program. Digital drive parameters are stored in each SERCOS-compatible digital drive and are necessary for configuring a motion system.

Note: Only VisualMotion control, task and axis parameters will be described in this chapter. Bosch Rexroth digital drives uses S (SERCOS) and P (product specific) parameters for setting up and configuring each drive. Refer to the respective *Digital Drive* documentation or *Drive Help System* for a description of these drive parameters.

Control, task, axis, and drive parameters can be accessed via a serial, Ethernet, or PCI interface using any of the following methods:

- VisualMotion Toolkit
- Windows™ based HMI software
- PLC

Some parameters may be accessed by the user program through a **Param** icon. VisualMotion also allows access to specific groups of parameters for editing and archiving.

Refer to *Chapter 3, Icon Programming*, and *Chapter 13, Communication Protocol*, for details.

5.2 Parameter Identification

All parameters in VisualMotion have a unique **IDentification Number (IDN)** and are displayed using the SERCOS format:

SERCOS Format:

```
## X-s-nnnn
|   |   |   |   |
|   |   |   |   | 4 digit parameter number
|   |   |   |   | Set: 0 or 7
|   |   |   |   | Class: Parameter type
|   |   |   |   | Class Number
```

Example: 01 S-0-0001

Description: NC cycle time for drive 1

Legend:

- ##:** Class Number
 (01) fixed for C parameters
 (01-40) SERCOS drive address for A,S, and P parameters
 (01-04)Task:A=01,B=02,C=03,D=04
- X :** Class: Parameter type
 C – Control
 T – Task
 A – Axis
 S – SERCOS drive parameter
 P – Product specific parameter
- s :** Certain S and P drive parameters belong to two different sets
 0:set parameters can be modified
 7:set parameters are read only
- n :** 4 digit parameter number
 (0000 – 4095)

Parameter Attributes

The following attributes for each parameter are provided:

- Data length
- Data type
- Display format
- Units
- Minimum value
- Maximum value
- Default value
- Access (read / write)

Class "C" Parameters

Class "C" control parameter types include the system setup and status parameters for VisualMotion. Certain status parameters are stored on the control in non-volatile memory, while other status parameters are only temporary indicators and are lost when the system is powered off.

Class "T" Parameters

Class "T" task parameter types include setup and status information for each task. Tasks A-D are represented by a corresponding ASCII numbers 01- 04 respectively.

Class "A" Parameters

Class "A" axis parameter types include the parameters used to configure and display information about the each axis. A maximum of 40 different sets of parameters is possible. A set corresponding to one of the eight axes is identified by a single digit in the range 01-40.

Class "S" and "P" Parameters

Class "S" (SERCOS) drive parameters are accessed through the SERCOS Service Channel.

Class "P" (Product specific) drive parameters include the parameters required to configure and operate Bosch Rexroth's digital drives. These parameters are unique to Bosch Rexroth drives and are necessary for proper communication between the drives and external devices.

Most servo and position loop parameters are contained in SERCOS compatible digital drives. These parameters include Feed Constant, Kv Factor, In-Position Window, Monitor Window, and all homing parameters. Acceleration, Deceleration, and Jerk are defined in the user program and limits for these are set in the drive.

5.3 Parameter Transfer

Parameter transfer capability is built into VisualMotion's programming language. The **Param** icon is used for parameter transfers. It allows the monitoring of system values and SERCOS command execution.

All control and drive parameters of the float type may be read into a VisualMotion float variable. All other parameters, unless they are a String or List type, can be read into VisualMotion integer variable. VisualMotion issues a runtime error if values are transferred to/from the wrong type of variable. The parameters with "read / write in any phase" access may be changed by a user program.

5.4 SERCOS Drive Telegram Utility

VisualMotion's *Parameter Overview* tool contains a SERCOS Telegram Utility used to configure a drive's AT (drive telegram) and MDT (master data telegram). The AT and MDT are used to cyclically exchange parameter data between drives and control every SERCOS cycle. The AT is sent from each drive to the control and the MDT is sent from the control to each active drive on the system.

This utility provides the user with a convenient and comprehensive interface for viewing the AT and MDT.

The AT and MDT are comprised of parameter numbers (IDNs) from various drive parameters. Some IDNs displayed in the AT and MDT are automatically configured based on the control's primary and secondary modes of operation. User configurable IDNs can be added to each telegram. User configurable IDNs and data are placed within axis parameters. The user configurable IDNs are labeled as *Optional Feedback ID* for the AT and *Optional Command ID* for the MDT.

Refer to *Axis Parameters used by AT and MDT* on page 5-8 for details.

AT (Drive Telegram)

The AT (Drive Telegram) is a block of drive data transmitted cyclically to the control every SERCOS cycle. The configurable data block of the AT contains a 2-byte drive status word, variable length drive dependent parameters that are automatically configured based on the mode of operation and a user configurable data block.

Note: The AT's configurable data block size limitation is dependent upon the connected drive and drive firmware. Refer to the relevant *Digital Drive* documentation for the actual size limitation of the AT.

VisualMotion controls allow only 2 variable length drive dependent parameters to be added to the user configurable data block of the AT.

ECODRIVE03 Drives:

DKC02.3 drives using SGP/SMT firmware have a 16-byte (8 word) limit for the configurable data block. If the length of the data that the user adds exceeds this limit, the cyclic multiplex channel is automatically activated. Refer to *Multiplex (MUX) Channel (DKC 2.3 only)* on page 5-9 for details.

DIAX04 Drives:

DIAX04 drives using SSE/ELS firmware have a 24-byte (12 word) limit and do not have a multiplex channel. Since DIAX04 drives have a much larger configurable data block for the AT, adding 2 drive parameters will not exceed the 24-byte limit.

Drive Status Word

The Drive Status Word (S-0-0135) is part of the AT's configurable data block and contains drive status information transmitted to the control. The following table describes the bits of the drive status word.

Bit	Description
2 – 0	Control Information for SERCOS service channel
5	Bit change command
7 – 6	Real-time status bits 1 and 2
9 – 8	actual type of operation 00: primary mode of operation 01: secondary mode of operation
11	Bit change class 3 diagnostic
12	Bit change class 2 diagnostic
13	Drive lock, error in class 1 diagnostic
15 – 14	Ready to operate: 00: Drive not ready for power to be switched on because internal checks not positively connected 10: Control and power supply ready for operation, torque free 01: Ready to switch on power 11: In Operation, under torque

Table 5-1: S-0-0135 Drive Status Word

Configurable AT Data Block

The contents of the AT telegram is automatically set by the control based on axis configuration options and the settings in the SERCOS drive telegram utility.

MDT (Master Data Telegram)

The MDT (Master Data Telegram) is a block of control data transmitted cyclically to the drives every SERCOS cycle. The configurable data block of the MDT contains a 2-byte master control word, variable length drive dependent parameters that are automatically configured based on the mode of operation and a user configurable data block.

Note: The MDT's configurable data block size limitation is dependent upon the connected drive and drive firmware. Refer to the relevant *Digital Drive* documentation for the actual size limitation of the MDT.

VisualMotion controls allow only 3 variable length drive dependent parameters to be added to the user configurable data block of the MDT.

ECODRIVE03 Drives:

DKC02.3 drives using SGP/SMT firmware have a 16-byte (8 word) limit for the configurable data block. If the length of the data that the user adds exceeds this limit, the cyclic multiplex channel is automatically activated. Refer to *Multiplex (MUX) Channel (DKC 2.3 only)* on page 5-9 for details.

DIAX04 Drives:

DIAX04 drives using SSE/ELS firmware have a 32-byte (16 word) limit and do not have a multiplex channel. Since DIAX04 drives have a much larger configurable data block for the AT, adding 3 drive parameters will not exceed the 32-byte limit.

Master Control Word

The Master Control Word (S-0-0134) is part of the MDT's configurable data block and contains control data transmitted to the drives. The following table describes the bits in the master control word.

Bit	Description
5 – 0	Control Information for Service Channel
7 – 6	Real-time Control Bits 1 and 2
9 – 8	Command Mode 00: primary mode 01: auxiliary modes
10	IPOSYNC: interpolator pulse, toggles if new command values to be transmitted
13	Drive HALT, 1⇒0 transition, standstill of drive while maintaining max. acceleration(S-0-0138 (only possible if bits 14 and 15 set to 1)
14	Drive ENABLE, 1⇒0 transition, torque off without delay (independent of bit 15 or 13)
15	Drive ON, 1⇒0 transition, best possible standstill (only possible if bit 14 set to 1)

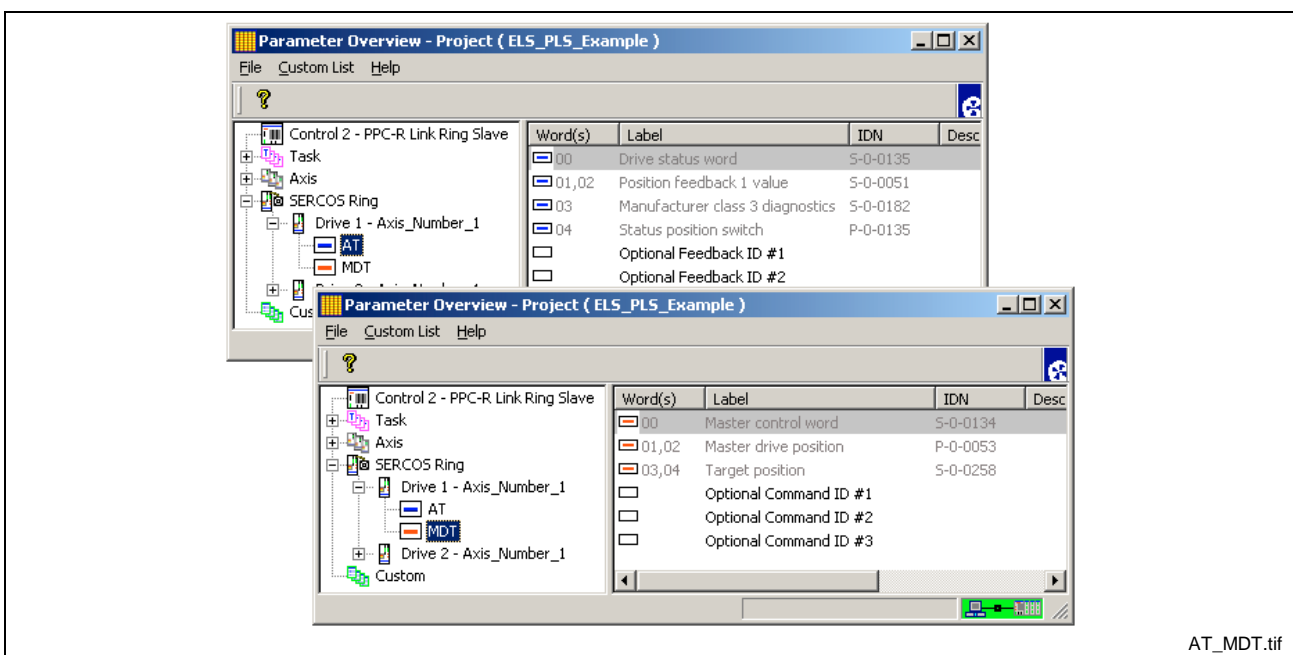
Table 5-2: S-0-0134 Master Control Word

Configurable MDT Data Block

The contents of the MDT is automatically set by the control based on axis configuration options and the settings in the SERCOS drive telegram utility.

Displaying the Contents of the AT and MDT

To launch the SERCOS Drive Telegram Utility, select **Data** ⇒ **Parameters**. Using the parameter tree structure, expand the SERCOS Ring to display the active drives. Expanding each drive icon displays the AT and MDT selections. Parameters that are operational mode specific are displayed in gray and user configurable data blocks in black.



AT_MDT.tif

Fig. 5-1: SERCOS Drive AT and MDT

Configuring the AT and MDT

Use the following steps to add parameter IDNs to the AT or MDT.

Note: The control must be switched to parameter mode to add or modify user configurable IDNs.

1. Open either the AT or MDT to display the contents of the telegram.
2. Double click a user configurable IDN or right click and select *Edit* to open the "List of configurable data" window of the selected telegram.

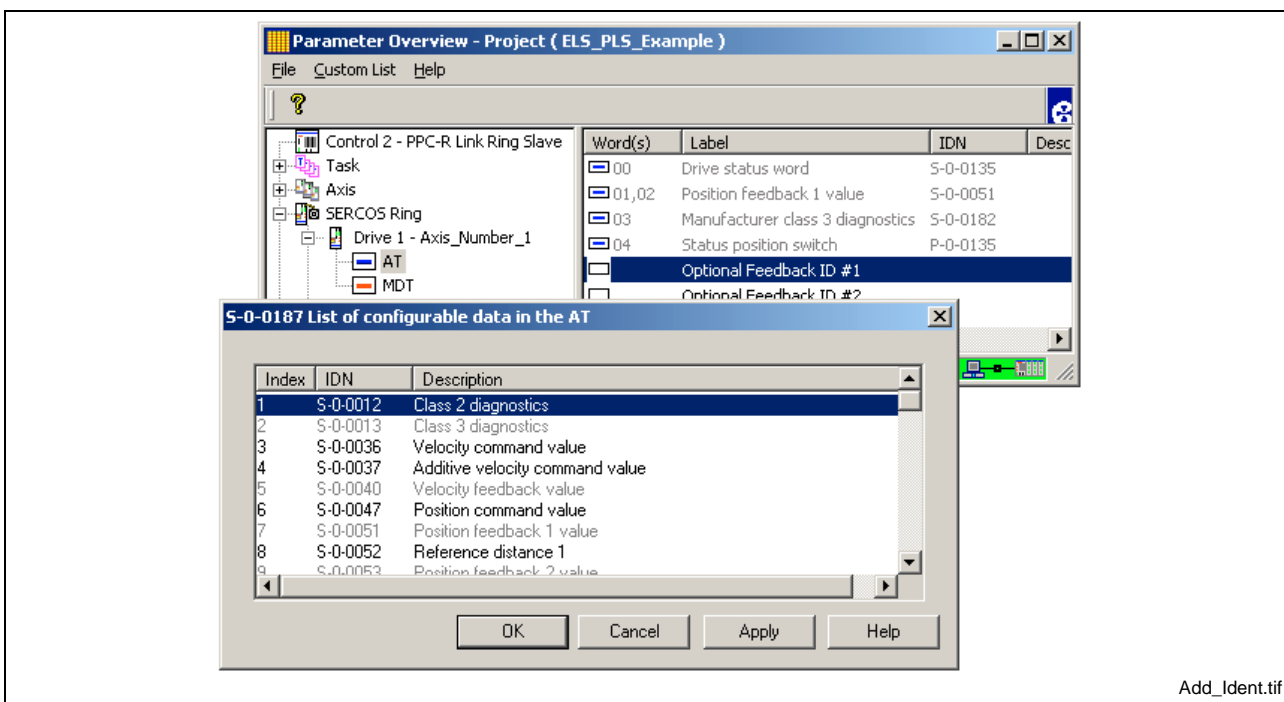


Fig. 5-2: Adding a Parameter IDN

Note: The list of configurable data for each telegram is provide in the following drive parameters:

- S-0-0187 for the AT
- S-0-0188 for the MDT

3. Select the desired parameter from the list and click the *OK* button. A message will appear notifying the user that changes take effect in phase 4.
4. Select the next user configurable IDN and repeat the process.
5. Switch the control to phase 4, right click any parameter IDN in the telegram, select *Refresh* or press the *F5* key to display the configured telegram.

Fig. 5-3 displays the configured telegram with user configured IDNs in black.

Note: To remove a user configured IDN, set the control to parameter mode, right click over the IDN and select *Remove*. Refresh the window to display the optional default IDN.

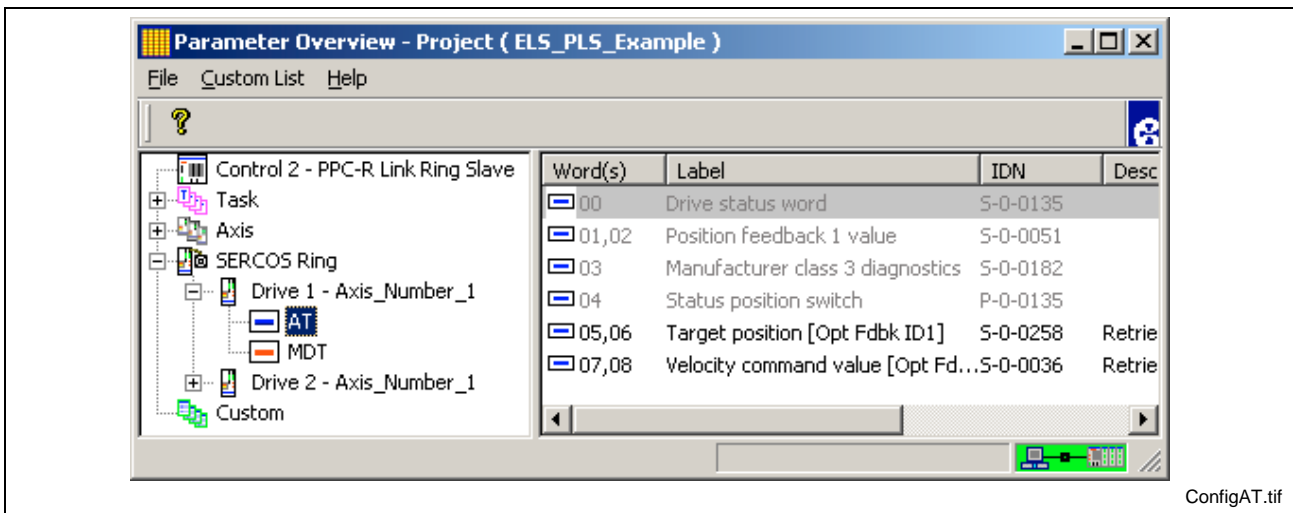


Fig. 5-3: Configured Telegram

Axis Parameters used by AT and MDT

A user configured parameter IDN is placed in an axis parameter. The data for the configured parameter IDN is accessed from a different axis parameter. The SERCOS telegram displays the axis parameter from where the data is retrieved under the *Description* column.

User Configured AT

The user can place up to 2 additional parameters into the user configurable data block of the AT. The corresponding axis parameters containing the parameter's IDN and data are listed in the following table.

User Configured AT (Drive Telegram) Data Location		
User Configured Data	Parameter Ident	Data Accessed
Optional Feedback ID #1	A-0-0185	A-0-0195
Optional Feedback ID #2	A-0-0186	A-0-0196

Table 5-3: User Configured AT Data

User Configured MDT

The user can place up to 3 additional parameters into the user configurable data block of the MDT. The corresponding axis parameters containing the parameter's IDN and data are listed in the following table.

User Configured MDT (Drive Telegram) Data Location		
User Configured Data	Parameter Ident	Data Accessed
Optional Command ID #1	A-0-0180	A-0-0190
Optional Command ID #2	A-0-0181	A-0-0191
Optional Command ID #3	A-0-0182	A-0-0192

Table 5-4: User Configured AT Data

Multiplex (MUX) Channel (DKC 2.3 only)

The MUX channel is a feature of DKC02.3 drive firmware and was designed to overcome the limited 16-byte data container length of the AT and MDT. The MUX channel is enabled by either the selection of the Drive PLS Fast Write Feature or enabled automatically when the AT or MDT exceed the 16-byte limit. Once enabled, the MUX channel reconstructs the AT and MDT to place predetermined parameter IDNs into the user configurable portion of the telegram and the remaining IDNs into the MUX channel.

The MUX IDNs are placed in a circular queue. Every SERCOS cycle, the index for each queue is incremented and one IDN and corresponding data from the MDT MUX queue is written to the drive and one IDN from the AT MUX queue is requested from the drive. The following AT read cycle updates the data location for that AT MUX IDN. The number of IDNs in the circular queue will determine how many SERCOS cycles occur before the data is updated.

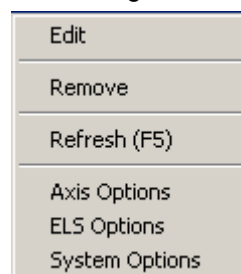
The following parameters and registers are associated with the MUX channel:

Parameter or Register	Description
A-0-0004, Bit 8 = 1	enables Drive PLS Fast Write
A-0-0200 A-0-0201	read-only lists of all IDNs that can be placed into the MUX for AT and MDT
A-0-0202 A-0-0203	holds the circular MUX queues for the AT and MDT
S-0-0360 S-0-0364	holds the data for the MUX channel parameter for the AT and MDT
S-0-0362 S-0-0366	holds the list of indexes for MUX channel list parameters
S-0-0368, Bits 0-5 S-0-0368, Bits 8-13	holds the index into A-0-0202 and A-0-0203 for the AT and MDT
S-0-0370 S-0-0371	holds copies for A-0-0200 and A-0-0201 for the AT and MDT
Registers 31-38, 309-340	bit 1 shows the status of the MUX channel enabled for axes 1-8 and 9-40

Table 5-5: Multiplexing Parameters

Telegram Options

The following telegram options are available when right clicking on any user configurable data:



The *Edit* and *Remove* selections are only visible when right clicking on user configurable data while in parameter mode. The telegram option selections are only available in parameter mode and require a system P2 ⇒ P4 transition for the modifications to take effect.

Edit

This selection is used to open the telegram's list of configurable data for adding an allowable parameter IDN to the telegram's user configurable data block.

Remove

This selection is used to remove a user configured parameter IDN from the selected telegram.

Refresh (F5)

This selection refreshes the contents of the telegram window to display any recent changes.

Axis Options

The *Axis Options* window is used to add a specific parameter IDN ,regardless of the mode of operation, to the cyclic data as well as enable features such as the Multiplex channel. Refer to axis parameter A-0-0004 for details.

Axis Option	Function
Optional Cyclic Data	sets A-0-0004, Bit 6 to 1
Drive PLS Fast Write	sets A-0-0004, Bit 8 to 1
Position Using Secondary Encoder	sets A-0-0004, Bit 11 to 1
Configure Minimum Cyclic Data	sets A-0-0004, Bit 14 to 1

Table 5-6: Axis Options

ELS Options

The *ELS Options* window is used to set several options for the ELS or CAM motion. It is also used to remove specific parameter IDNs from the cyclic data. Refer to axis parameters A-0-0004 and A-0-0164 for details.

ELS Option	Function
Remove Dynamic Phase Offset	sets A-0-0164, Bit 2 to 1
Remove MFG Class 3 Diagnostics	sets A-0-0164, Bit 3 to 1
Remove Target Position	sets A-0-0164, Bit 4 to 1
ELS Secondary Mode	sets A-0-0164, Bit 9 to 1

Table 5-7: ELS Options

System Options

The *System Options* window is used to enable a 4Mbps SERCOS transmission rate, set a SERCOS refresh delay, and set the minimum SERCOS cycle time. Refer to control parameters C-0-0010, C-0-0098 and C-0-0099 for details.

5.5 Parameter Types

The information provided in this chapter will focus on VisualMotion control, task, and axis parameter types. Drive parameters vary based on the type of digital drive and firmware used. Refer to the relevant *Digital Drive* manual for specific drive parameter descriptions.

Control Parameters - Class C

This section provides a list of all control parameters that are available in the system. They are grouped by functionality and indicate the supported control firmware type.

System Setup Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0001	Language Selection	X	X
C-0-0002	Device Address	X	X
C-0-0003	Serial Port A Setup	X	X
C-0-0004	Serial Port B Setup	X	X
C-0-0005	Communication Protocol Selection	X	X
C-0-0009	Error Reaction Mode	X	X
C-0-0010	System Options	X	X
C-0-0012	Serial Port B Device Type	X	X
C-0-0013	Serial Port A Mode	X	X
C-0-0014	Serial Port B Mode	X	X
C-0-0016	Communication Time-out Period	X	X
C-0-0017	Serial Port A Password	X	X
C-0-0018	Serial Port B Password	X	X
C-0-0020	Transmitter Fiber Optic Length	X	X
C-0-0021	User Watchdog Timer	X	X
C-0-0022	User Watchdog Task ID	X	X

Table 5-8: System Setup Parameters

Jogging and Display Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0042	World Large Increment	X	X
C-0-0043	World Small Increment	X	X
C-0-0045	World Fast Jog Speed	X	X
C-0-0046	World Slow Jog Speed	X	X
C-0-0052	Axis Large Increment	X	X
C-0-0053	Axis Small Increment	X	X
C-0-0055	Axis Fast Jog Velocity	X	X
C-0-0056	Axis Slow Jog Velocity	X	X

Table 5-9: Jogging and Display Parameters

Global Variable Command Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0080	Maximum Number of Global Integers	X	X
C-0-0081	Maximum Number of Global Floats	X	X
C-0-0082	Save Global Variables Command	X	X
C-0-0083	Save Global Variables Status	X	X

Table 5-10: Global Variable Command Parameters

Program Management Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0090	Download Block Size	X	X
C-0-0091	Total Program Memory	X	X
C-0-0092	Available Program Memory	X	X
C-0-0093	Contiguous Program Memory	X	X
C-0-0094	Maximum Executable Program Size	X	X
C-0-0098	Initialization Delay	X	X
C-0-0099	Minimum SERCOS Cycle Time	X	X

Table 5-11: Program Management Parameters

System Status Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0100	Control Firmware Version	X	X
C-0-0101	Control Hardware Version	X	X
C-0-0102	Control Version Date	X	X
C-0-0103	Allowable Drive Address Range	X	X
C-0-0104	Bootloader Firmware Version	X	X
C-0-0120	Operating Mode	X	X
C-0-0121	SERCOS Communication Phase	X	X
C-0-0122	Diagnostic Message	X	X
C-0-0123	Diagnostic Code	X	X
C-0-0124	Extended Diagnostic	X	X
C-0-0125	System Timer Value	X	X
C-0-0126	Date and Time	X	X
C-0-0127	Current PPC-R Temperature	X	X
C-0-0128	Elapsed Time Operational Counter	X	X
C-0-0142	Card Label String	X	X

Table 5-12: System Status Parameters

Save CAM Built to Flash Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0166	Save Built CAM to Flash ID	X	X
C-0-0167	Save CAM Built to Flash Command	X	X
C-0-0168	Save Built CAM to Flash Status	X	X

Table 5-13: Save CAM Built to Flash Parameters

Control Processor Usage Status Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0200	Current Load Due To Motion	X	X
C-0-0201	Peak Load Due To Motion	X	X
C-0-0202	Current Load Due To IO	X	X
C-0-0203	Peak Load Due To IO	X	X

Table 5-14: Control Processor Usage Status Parameters

Link Ring Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0300	Link Ring Control Word	X	X
C-0-0301	Link Ring Primary Fiber Optic Length	X	X
C-0-0302	Link Ring Secondary Fiber Optic Length	X	X
C-0-0303	Link Ring MDT Error Counter	X	X

Table 5-15: Link Ring Parameters

Ethernet Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0400	Ethernet Card IP Address	X	
C-0-0401	Ethernet Card Subnet Mask	X	
C-0-0402	Ethernet Card Gateway IP Address	X	
C-0-0403	Ethernet Card CIF Network Control	X	
C-0-0404	Ethernet Card Network Access Control	X	
C-0-0405	Ethernet Card Network Password	X	
C-0-0406	CIF Ethernet Card Hardware ID	X	
C-0-0407	CIF Ethernet Card Firmware Version	X	
C-0-0408	CIF Ethernet Driver Version	X	

Table 5-16: Ethernet Parameters

Initialization Task Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0522	Init. Task Diagnostic Message	X	X
C-0-0523	Init. Task Status Message	X	X
C-0-0530	Init. Task Current Instr. Pointer	X	X
C-0-0531	Init. Task Current Instruction	X	X
C-0-0532	Init. Task Instr. Pointer at Error	X	X
C-0-0533	Init. Task Composite Instr. Pointer	X	X
C-0-0535	Init. Task Current Subroutine	X	X
C-0-0536	Init. Task Stack Variable Data	X	X
C-0-0537	Init. Task Subroutine Breakpoint	X	X

Table 5-17: Initialization Task Parameters

BTC06 Teach Pendant Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0801	Pendant Protection Level 1 Password	X	X
C-0-0802	Pendant Protection Level 2 Password	X	X
C-0-0803	Pendant User Accessible Floats Section	X	X
C-0-0804	Pendant User Accessible Integers Section	X	X
C-0-0805	Pendant Start of User Accessible Registers	X	X
C-0-0806	Pendant End of User Accessible Registers	X	X
C-0-0807	Pendant Password Timeout	X	X
C-0-0810	TPT message and prompt control word	X	X
C-0-0811	User Task Controlled Menu ID for TPT	X	X
C-0-0812	User Task Controlled Task ID for TPT	X	X
C-0-0813	User Task Controlled Axis Number for TPT	X	X
C-0-0814	TPT Data Transaction Word	X	X

Table 5-18: Pendant Parameters

Internal System Monitoring

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0990	Exit to Monitor Prompt	X	X

Table 5-19: Internal System Monitoring

System Memory Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0993	Software Reset for Control	X	X
C-0-0994	Shutdown Command for Flash Programming	X	X
C-0-0996	Clear Program and Data Memory	X	X
C-0-0997	Clear Diagnostic Log	X	X

Table 5-20: System Memory Parameters

System Parameter Lists

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-2000	List of All Parameters	X	X
C-0-2001	List of Required Parameters	X	X
C-0-2002	List of invalid A-, C- and T- parameters	X	X
C-0-2010	List of SERCOS Devices	X	X
C-0-2011	List of SERCOS Drives	X	X
C-0-2012	List of SERCOS I/O Stations	X	X
C-0-2013	I/O Configuration List	X	X
C-0-2016	List of Virtual axes	X	X
C-0-2017	I/O User Configuration List	X	X
C-0-2020	Diagnostic Log List	X	X
C-0-2021	Diagnostic Log Options	X	X
C-0-2022	Probe Exception Handler	X	X

Table 5-21: System Parameter Lists

Oscilloscope Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-2501	Oscilloscope signal 1 type	X	X
C-0-2502	Oscilloscope signal 2 type	X	X
C-0-2503	Oscilloscope signal 3 type	X	X
C-0-2524	Oscilloscope signal 4 type	X	X
C-0-2504	Oscilloscope signal 1 id number	X	X
C-0-2505	Oscilloscope signal 2 id number	X	X
C-0-2506	Oscilloscope signal 3 id number	X	X
C-0-2525	Oscilloscope signal 4 id number	X	X
C-0-2507	Oscilloscope signal 1 axis number	X	X
C-0-2508	Oscilloscope signal 2 axis number	X	X
C-0-2509	Oscilloscope signal 3 axis number	X	X
C-0-2526	Oscilloscope signal 4 axis number	X	X
C-0-2510	Oscilloscope sample rate	X	X

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-2511	Oscilloscope signal 1 list	X	X
C-0-2512	Oscilloscope signal 2 list	X	X
C-0-2513	Oscilloscope signal 3 list	X	X
C-0-2527	Oscilloscope signal 4 list	X	X
C-0-2514	Oscilloscope sample count	X	X
C-0-2515	Oscilloscope trigger post-count	X	X
C-0-2516	Oscilloscope trigger type	X	X
C-0-2517	Oscilloscope trigger id number	X	X
C-0-2518	Oscilloscope trigger axis or mask	X	X
C-0-2519	Oscilloscope trigger level or mask	X	X
C-0-2520	Oscilloscope trigger mode	X	X
C-0-2521	Oscilloscope trigger source	X	X
C-0-2522	Oscilloscope trigger control word	X	X
C-0-2523	Oscilloscope trigger status word	X	X

Table 5-22: Oscilloscope Parameters

Fieldbus/PLC Interface Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-2600	Fieldbus/PLC Mapper (cyclic channel) To PLC	X	X
C-0-2601	Fieldbus/PLC Mapper (cyclic channel) From PLC	X	X
C-0-2607	Multiplex Control Word	X	X
C-0-2608	Multiplex Status Word	X	X
C-0-2611	Fieldbus/PLC Cyclic Channel: Current number of misses	X	X
C-0-2612	Fieldbus/PLC Cyclic Channel: Peak number of misses	X	X
C-0-2613	Fieldbus/PLC Cyclic Channel: Timeout counter	X	X
C-0-2630	Fieldbus Slave Device Address	X	
C-0-2631	Fieldbus Parameter/PCP Channel Length	X	
C-0-2632	Fieldbus/PLC Multiplex Method	X	
C-0-2633	Fieldbus Baud Rate (DeviceNet only)	X	
C-0-2635	Fieldbus/PLC Error Reaction	X	X
C-0-2636	Fieldbus/PLC Word Swap	X	X
C-0-2637	Fieldbus Slave/PLC Firmware Version	X	X
C-0-2638	Fieldbus Available Cyclic IN Parameters	X	X
C-0-2639	Fieldbus Available Cyclic OUT Parameters	X	X

Table 5-23: Fieldbus Interface Parameters

PLC Interface Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-0035	PLC Communication Option	X	
C-0-2640	PLC Connection Options		X
C-0-2641	PLC Input Register List	X	X
C-0-2642	PLC Output Register List	X	X
C-0-2643	PLC Lifecounter Check: Number of Retries	X	X
C-0-2644	PLC Lifecounter Check: Current Number of Misses	X	X
C-0-2645	PLC Lifecounter Check: Peak Number of Misses	X	X
C-0-2646	PLC Lifecounter Check: Number of Timeouts	X	X
C-0-2647	ISP Function Block Timeout	X	
C-0-2651	PLC Register Channel: Current number of misses	X	X
C-0-2652	PLC Register Channel: Peak number of misses	X	X
C-0-2653	PLC Register Channel: Timeout counter	X	X

Table 5-24: PLC Interface Parameters

Option Card PLS Interface Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-2901	PLS1 Start Output Register	X	X
C-0-2902	PLS1 Start Mask Register	X	X
C-0-2903	PLS1 Build Table Command	X	X
C-0-2904	PLS1 Build Table Status	X	X
C-0-2905	PLS1 Activate Command	X	X
C-0-2906	PLS1 Activate Status	X	X
C-0-2907	PLS1 Error Code	X	X
C-0-2908	PLS1 Extended Error Code	X	X
C-0-2909	PLS1 Hardware ID	X	X
C-0-2910	PLS1 Software ID	X	X
C-0-2920	PLS1 Switch On List	X	X
C-0-2921	PLS1 Switch Off List	X	X
C-0-2922	PLS1 Switch Output List	X	X
C-0-2930	PLS1 Output Master List	X	X
C-0-2931	PLS1 Output Lead Time List	X	X
C-0-2932	PLS1 Output Lag Time List	X	X
C-0-2933	PLS1 Output One Shot (PT) List	X	X
C-0-2934	PLS1 Output Mode List	X	X

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-2935	PLS1 Output Direction List	X	X
C-0-2936	PLS1 Output Hysteresis List	X	X
C-0-2940	PLS1 Master Type List	X	X
C-0-2941	PLS1 Master Number List	X	X
C-0-2942	PLS1 Master Encoder List	X	X
C-0-2943	PLS1 Master Phase Offset List	X	X

Table 5-25: Option Card PLS Interface Parameters

I/O Mapper Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-3000	I/O Mapper Program	X	X
C-0-3001	I/O Mapper Options	X	X
C-0-3003	I/O Mapper Total Operations	X	X
C-0-3004	I/O Mapper File Size	X	X
C-0-3005	I/O Mapper Executable Size	X	X

Table 5-26: I/O Mapper Parameters

CAM Table Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
C-0-3100	CAM Tags	X	X
C-0-3101	CAM Table 1	X	X
C-0-3102	CAM Table 2	X	X
C-0-3103	CAM Table 3	X	X
C-0-3104	CAM Table 4	X	X
C-0-3105	CAM Table 5	X	X
C-0-3106	CAM Table 6	X	X
C-0-3107	CAM Table 7	X	X
C-0-3108	CAM Table 8	X	X
C-0-3109	CAM Table 9	X	X
consecutively through			
C-0-3140	CAM Table 40	X	X
C-0-3141	CAM Type	X	X

Table 5-27: CAM Table Parameters

Position Monitor Group Parameters

Parameter Number		Description	Valid for	
			GPP 9	GMP 9
C-0-3201	C-0-3211	PMG # Maximum Allowed Deviation Window where # = group numbers 1-8	X	X
C-0-3221	C-0-3231			
C-0-3241	C-0-3251			
C-0-3261	C-0-3271			
C-0-3202	C-0-3212	PMG # List of Axes	X	X
C-0-3222	C-0-3232			
C-0-3242	C-0-3252			
C-0-3262	C-0-3272			
C-0-3203	C-0-3213	PMG # List of Position Offsets	X	X
C-0-3223	C-0-3233			
C-0-3243	C-0-3253			
C-0-3263	C-0-3273			
C-0-3204	C-0-3214	PMG # Current Peak Group Deviation	X	X
C-0-3224	C-0-3234			
C-0-3244	C-0-3254			
C-0-3264	C-0-3274			
C-0-3205	C-0-3215	PMG # Maximum Deviation	X	X
C-0-3225	C-0-3235			
C-0-3245	C-0-3255			
C-0-3265	C-0-3275			
C-0-3206	C-0-3216	PMG # Configuration	X	X
C-0-3226	C-0-3236			
C-0-3246	C-0-3256			
C-0-3266	C-0-3276			

Table 5-28: Position Monitoring Group Parameters

Task Parameters - Class T

This section provides a list of all task parameters that are available in the system. They are grouped by functionality and indicate the supported control firmware type.

Task Setup Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
T-0-0001	Task Motion Type	X	X
T-0-0002	Task Options	X	X

Table 5-29: Task Setup Parameters

Coordinated Motion

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
T-0-0005	World Position Units	X	X
T-0-0010	Kinematics Number	X	X
T-0-0011	Coordinated X-axis	X	X
T-0-0012	Coordinated Y-axis	X	X
T-0-0013	Coordinated Z-axis	X	X
T-0-0020	Maximum Path Speed	X	X
T-0-0021	Maximum Acceleration	X	X
T-0-0022	Maximum Deceleration	X	X
T-0-0023	Look Ahead Distance	X	X
T-0-0024	Velocity Override	X	X
T-0-0025	Maximum Jog Increment	X	X
T-0-0026	Maximum Jog Velocity	X	X
T-0-0027	Path Smoothing Filter Constant	X	X
T-0-0050	Kinematics Value 1	X	X
T-0-0051	Kinematics Value 2	X	X
T-0-0052	Kinematics Value 3	X	X
T-0-0053	Kinematics Value 4	X	X
T-0-0054	Kinematics Value 5	X	X
T-0-0055	Kinematics Value 6	X	X
T-0-0056	Kinematics Value 7	X	X
T-0-0057	Kinematics Value 8	X	X
T-0-0058	Kinematics Value 9	X	X
T-0-0059	Kinematics Value 10	X	X

Table 5-30: Coordinated Motion Parameters

Coordinated Motion Status

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
T-0-0100	Target Point Number	X	X
T-0-0101	Segment Status	X	X
T-0-0111	Current X Position	X	X
T-0-0112	Current Y Position	X	X
T-0-0113	Current Z Position	X	X

Table 5-31: Coordinated Motion Status Parameters

Task Status

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
T-0-0120	Task Operating Mode	X	X
T-0-0122	Task Diagnostic Message	X	X
T-0-0123	Task Status Message	X	X
T-0-0130	Current Instruction Pointer	X	X
T-0-0131	Current Instruction	X	X
T-0-0132	Instruction Pointer at Error	X	X
T-0-0133	Composite Instruction Pointer	X	X
T-0-0135	Current Subroutine	X	X
T-0-0136	Stack Variable Data	X	X
T-0-0137	Subroutine Breakpoint	X	X
T-0-0138	Sequencer Information	X	X
T-0-0200	Last Active Event Number	X	X

Table 5-32: Task Status Parameters

Task Parameter Lists

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
T-0-2000	List of All Parameters	X	X
T-0-2001	List of Required Parameters	X	X

Table 5-33: Task Parameters Lists

Axis Parameters - Class A

This section provides a list of all axis parameters that are available in the system. They are grouped by functionality and indicate the supported control firmware type.

Axis Setup Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-0001	Task Assignment	X	X
A-0-0002	Type of Positioning	X	X
A-0-0003	Axis Motion Type	X	X
A-0-0004	Axis Options	X	X
A-0-0005	Linear Position Units	X	X
A-0-0006	Reference Options	X	X
A-0-0007	Configuration Mode	X	X
A-0-0009	Drive PLS Register	X	X
A-0-0020	Maximum Velocity	X	X
A-0-0021	Maximum Acceleration	X	X
A-0-0022	Maximum Deceleration	X	X
A-0-0023	Jog Acceleration	X	X
A-0-0025	Maximum Jog Increment	X	X
A-0-0026	Maximum Jog Velocity	X	X

Table 5-34: Axis Setup Parameters

Axis Runtime Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-0100	Target Position	X	X
A-0-0101	Commanded Position	X	X
A-0-0102	Feedback Position	X	X
A-0-0110	Programmed Velocity	X	X
A-0-0111	Commanded Velocity	X	X
A-0-0112	Feedback Velocity	X	X
A-0-0120	Programmed Acceleration	X	X
A-0-0131	SERCOS Control Word	X	X
A-0-0132	SERCOS Status Word	X	X
A-0-0133	AT Error Count	X	X
A-0-0140	Mfg. Class 3 Status Word	X	X
A-0-0141	Torque Mode Commanded Torque	X	X
A-0-0142	Torque Feedback (cyclic)	X	X
A-0-0145	Current Motion Type	X	X

Table 5-35: Axis Status Parameters

Axis Electronic Line Shafting Parameters

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-0030	Ratio Mode Master Axis	X	X
A-0-0031	Control CAM/Ratio Master Factor (N)	X	X
A-0-0032	Control CAM/Ratio Slave Factor (M)	X	X
A-0-0033	Control CAM Stretch Factor (H)	X	X
A-0-0034	Control CAM Currently Active	X	X
A-0-0035	Control CAM Position Constant (L)	X	X
A-0-0036	Ratio Mode Encoder Type	X	X
A-0-0037	Ratio Mode Step Rate	X	X
A-0-0038	Ratio Mode Options	X	X
A-0-0150	Programmed Ratio Adjust	X	X
A-0-0151	Programmed Phase Offset	X	X
A-0-0153	Control Phase Adjust Average Velocity	X	X
A-0-0155	Control Phase Adjust Time Constant	X	X
A-0-0156	Phase Offset Velocity Feedback	X	X
A-0-0157	Current Phase/ Control CAM Master Offset	X	X
A-0-0158	Relative Phase Offset Distance Remaining	X	X
A-0-0159	Ratio Adjust Step Rate	X	X
A-0-0160	Commanded Ratio Adjust	X	X
A-0-0161	Control CAM Programmed Slave Adjust	X	X
A-0-0162	Control CAM Current Slave Adjust	X	X
A-0-0163	Control CAM Output Position	X	X
A-0-0164	ELS Options	X	X

Table 5-36: Axis Electronic Line Shafting Parameters

Axis Feedback Capture (Registration)

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-0170	Probe Configuration Status	X	X
A-0-0171	Probe 1 Positive Captured Value	X	X
A-0-0172	Probe 1 Negative Captured Value	X	X
A-0-0173	Probe 2 Positive Captured Value	X	X
A-0-0174	Probe 2 Negative Captured Value	X	X

Table 5-37: Axis Feedback Capture (Registration)

Optional SERCOS Data

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-0180	Optional Command ID #1	X	X
A-0-0181	Optional Command ID #2	X	X
A-0-0182	Optional Command ID #3	X	X
A-0-0185	Optional Feedback ID #1	X	X
A-0-0186	Optional Feedback ID #2	X	X
A-0-0190	Command Data #1	X	X
A-0-0191	Command Data #2	X	X
A-0-0192	Command Data #3	X	X
A-0-0195	Feedback Data #1	X	X
A-0-0196	Feedback Data #2	X	X

Table 5-38: Optional SERCOS Data

Multiplexing Parameters (DKC 2.3 only)

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-0200	MDT Multiplex Selection List	X	X
A-0-0201	AT Multiplex Selection List	X	X
A-0-0202	MDT Multiplex Ident List	X	X
A-0-0203	AT Multiplex Ident List	X	X

Table 5-39: Multiplexing Parameters

Axis Parameter Lists

Parameter Number	Description	Valid for	
		GPP 9	GMP 9
A-0-2000	List of All Parameters	X	X
A-0-2001	List of Required Parameters	X	X

Table 5-40: Axis Parameters Lists

5.6 Control Parameters – Class C

Control parameters include setup parameters for system configuration, program options, serial interface, and I/O options. Also included are status parameters such as operating mode, and diagnostic messages.

Note: The system parameters that appear in this section are **not** all valid for both GPP 9 and GMP 9 firmware. For this reason, any system parameter that is used specifically for GPP 9 or GMP 9 will be indicated at the end of the description.

For example:

"C-0-2640 PLC Connection Options (**GMP 9 only**)"

No specification to a firmware version indicates that the message is valid for both GPP 9 and GMP 9. Refer to the tables on page 5-11 for a complete listing of parameters by firmware type.

System Setup (C-0-0001 through C-0-0035)

C-0-0001 Language Selection

All parameter names, units and diagnostic warning messages within the control are stored in multiple languages. This parameter determines the output language for the text. The following two languages are supported:

- 0 = German
- 1 = English

C-0-0001 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	4
Default value:	1
Access:	read / write in any phase

C-0-0002 Device Address

This parameter is used to set the control's device address for the different interface scenarios listed in the table below. The default device address is 128.

Interface Scenario	Valid Range
Control Address	0-126 and 128 for SIS and ASCII (127 is invalid)
RS-232 (RS-422) Serial Interface	0-126 and 128 for SIS and ASCII (127 is invalid)
RS-485 Serial Interface	0-126 and 128 for SIS (127 is invalid) 0-31 for ASCII
Link Ring	1 to 32

Table 5-41: Device Address

The device address can be set in VisualMotion Toolkit by selecting **Tools** ⇒ **Control Settings** and changing the **Address** field under the *General* tab in offline or service mode.

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

Control Address

Communication via Ethernet TCP/IP, Fieldbus (non-cyclic channel) and PCI/DPR (non-cyclic/programming channels) is supported for both ASCII and SIS protocols.

ASCII Protocol When using ASCII protocol, the control's address is not used when data is processed.

SIS Protocol When using SIS protocol, the data received by the control must contain the same device address that is set in this parameter.

Serial Interface

RS-232, RS-422 and RS-485 serial interfaces are supported using either ASCII or SIS protocol.

ASCII Protocol When using RS-232 or RS-422 serial interface, this parameter allows a single device address to be set. When using RS-485 serial interface, up to 32 controls can be accessed across a Multi-Drop connection. Each control using RS-485 must have its own unique device address.

SIS Protocol: The data received by the control must contain the same device address set in this parameter for both RS-232, RS-422 or RS-485 serial interface. Address 127 is reserved for a SIS master and cannot be entered as a device address.

Link Ring Address

Up to 32 controls can be used in a Link Ring configuration. Link Ring does not use ASCII or SIS protocols for communication. This parameter sets the device address for each control used in a Link Ring configuration. Each control must have its own unique device address.

C-0-0002 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	128
Default value:	128
Access:	read / write in any phase

C-0-0003 Serial Port A Setup

This parameter sets the baud rate, checksum, and parity for serial port X10 on the control. The port can communicate with a PC, a PLC, HMI or any device that uses ASCII or SIS protocols. The default settings of the port are 8 data bits, 1 stop bit, and no parity.

Checksum verification can be enabled by adding 1 to the baud rate (i.e. $9600 + 1 = 9601$).

The parity can be changed by adding 0 for no parity, 4 for even parity, and 8 for odd parity to the baud rate/checksum value.

For example:

A port setting of 9600 baud rate with checksum enabled and even parity would equal 9605 ($9600 + 1 + 4$).

The supported baud rates are 9600, 19200, 38400, 57600 and 115200.

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

C-0-0003 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	115203
Default value:	9601 (baud rate + checksum on)
Access:	read / write in any phase

C-0-0004 Serial Port B Setup

This parameter sets the baud rate, checksum, and status message for serial port X16 on the control. The port can communicate with a PC, a PLC, HMI or any device that uses ASCII or SIS protocols. The default settings of the port are 8 data bits, 1 stop bit, and no parity. The device connected to this port is selected in parameter C-0-0012 Serial Port B Device Type.

Checksum verification and status message can be enabled by adding 1 for checksum and 2 for status message to the baud rate (i.e. $9600 + 1 + 2 = 9603$). The supported baud rates are 9600, 19200, 38400, 57600 and 115200.

The parity can be changed by adding 0 for no parity, 4 for even parity, and 8 for odd parity to the baud rate/checksum/status message value.

For example:

A port setting of 9600 baud rate with checksum and status message enabled and even parity would equal 9607 ($9600 + 1 + 2 + 4$).

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

C-0-0004 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	115203
Default value:	9601 (baud rate + checksum on)
Access:	read / write in any phase

C-0-0005 Communication Protocol Selection

This parameter is read-only and displays the current communication protocol recognized by VisualMotion for system communication. The allowable types are listed in the following table:

Value	Description
1	autosense for SIS or ASCII protocols (set by VisualMotion)
2	reserved for future
3	reserved for future

Table 5-42: Communication Protocol Selection

C-0-0005 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	1
Maximum value:	3
Default value:	1
Access:	read-only

C-0-0009 Error Reaction Mode

This parameter sets the reaction of the control to fatal errors as either immediate or controlled. When a fatal error occurs, the control determines which user task will be shutdown. Axes, associated with a user task that is shutdown, are stopped based on the error reaction of the drive. Refer to *Chapter 12, Error Reaction*, for details.

0 = Immediate error reaction to fatal errors

By default, all axes, regardless of their operating mode, are immediately disabled and stopped by the drive.

1 = Controlled error reaction to fatal errors

All motion comes to a controlled stop before the axes are disabled. ELS axes stop synchronized and coordinated motion axes stop on path. Axes in single axis mode and velocity mode are switched to drive halt.

C-0-0009 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	1
Default value:	0
Access:	read in any phase / write in phase 2

C-0-0010 System Options

This parameter sets several options for the VisualMotion system and for the SERCOS ring.

0000000000000000	_ Bit 1
Bit 1:	Simulation mode
Bits 2-3:	Phase 2 SERCOS I/O Updates
Bit 4:	not used
Bit 5:	Enable 4Mbps SERCOS transmission
Bits 6-7:	RECO I/O Error Reaction Bits
Bits 8-9:	Reserved for future development
Bit 10:	not used
Bit 11:	Jog in auto mode
Bit 12:	Prioritized service channel
Bit 13:	Ignore drive warnings
Bit 14:	Ignore axis ready status in program commands
Bit 16:	Disable AT timing check

Fig. 5-4: Bit Description C-0-0010

C-0-0010 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read in any phase / write in phase 2

Bit 1: Simulation Mode

When set to 1, the presence of axes will be simulated. The SERCOS ring will not be scanned for drives, and the drive enable bits will be ignored. This mode is useful for simulating coordinated motion when the control is not connected to the actual system, or when a program does not contain any axes. All axis and task status parameters are simulated. Drive parameters and I/O are not simulated since they require a SERCOS drive. Any drive-controlled motion (homing, single-axis, etc.) is also not simulated. ELS and single-axis modes do not support this feature.

0 = normal drive operation

1 = simulation mode, do not scan for drives

Bits 2-3: Phase 2 SERCOS I-O Update

In SERCOS phase 2, all drive-resident I/O and SERCOS I/O modules are scanned every 500ms by default (bits 2, 3 = 0). This can slow communications when downloading parameter lists from the drives, depending upon how much I/O is visible. If fast updates of I/O in phase 2 are not critical, set bits 2 and 3 to a nonzero value. The I/O will be scanned every 10 seconds or not at all, which speeds up communication with the user interface.

Bit 3	Bit 2	Update Time
0	0	500 ms update
0	1	10 second update
1	N/A	no update

Bit 5: Enable 4Mbps SERCOS Transmission

This parameter sets the control's SERCOS transmission rate to either 2 or 4 Mbits/s.

0 = 2 Mbits/s SERCOS transmission rate (default)

1 = 4 Mbits/s SERCOS transmission rate

The transmission rate has no effect on the control or drive performance unless a smaller SERCOS cycle time is needed. The minimum SERCOS cycle time in control parameter C-0-0099 is 2ms.

Bit 6-7: RECO I/O Error Reaction Bits

Bits 6 and 7 are used to specify how the control responds to errors reported by RECO I/O modules. The bit settings are as follows:

Bit 6	Bit 7	Error
0	0	ignore
1	0	warning
x	1	fatal error (default)

Table 5-43: RECO I/O Error Reaction Bits (C-0-0010)

Ignore

The control ignores any errors reported by Local and SERCOS RECO I/O modules. This reaction is selected if the user program is to handle RECO I/O errors. In this case, the I/O Configuration tool can be used to map the RECO I/O modules to VisualMotion registers. Refer to *Chapter 6, VisualMotion I/O System*, for details.

Note: This reaction provides default backwards compatibility to older versions of GPP firmware that are not capable of directly responding to RECO I/O errors.

Warning

The control responds to errors reported by the Local and SERCOS RECO I/O modules by issuing a "215 RECO I/O Failure" warning. This error reaction is selected if the user is to be notified of any RECO I/O errors, while still allowing the user program to continue executing.

Fatal Error

The control responds to errors reported by the Local and SERCOS RECO I/O modules by issuing a "544 RECO I/O Failure" error, stopping program execution and motion. This is the system's default error reaction. This error reaction is selected if the application requires program execution and motion to be stopped as soon as a RECO I/O error is detected.

Bit 11: Jog in Auto Mode

Any axis that is currently in single-axis, velocity, or coordinated motion mode can be jogged with the axis jog bits.

When set to 1, jogging can be performed in auto mode or when a task is running. This allows continuous or incremental axis motion to be started and stopped by simply setting its jog bits and parameters.

Bit 12: Prioritized Service Channel

Only one task or user interface can access a drive's SERCOS service channel at a time. When drive parameter lists, long text strings, or oscilloscope data is transferred from one service channel access, other service channel access could be suspended for several seconds.

Set this bit to 1 if the timing for user tasks access, via the SERCOS service channel, is critical. The user tasks will suspend any SERCOS transmission of any text strings or lists from the user interface, and the "!78 Service channel in use" communication error will be issued.

If user tasks access is not critical, parameter lists and oscilloscope are seldom used during normal operation, or "!78" errors occur while viewing parameters, this bit should be set to 0.

0 = Don't prioritize the service channel (default)

1 = User tasks have priority over to user interface

Note: Even with prioritization, service channel access can vary between 10 to 100 ms. Therefore, drive parameters which transfer time critical data should be configured in the cyclic data channel.

Refer to the *SERCOS Drive Telegram Utility* on page 5-4 for details.

Bit 13: Ignore Drive Warnings

When set to 0, the control reacts to drive warnings by shutting down associated user tasks. A drive warning sets bit 13 (Class_2_Warning) in the relevant axis status register. The default reaction of the control to a warning is to issue the error "498 Drive Shutdown Warning".

When set to 1, drive warnings are ignored by the control and associated user tasks are not shutdown.

The setting in this parameter affects all axis in the system.

0 = Drive warnings issue shutdown (default)

1 = Drive warning are ignored by the control

Bit 14: Ignore Axis Ready Status in Program Commands

When set to 0, the control issues an error if a start command is instructed before the drive is ready (AH or AF). These commands include the start command, the homing command, and the operation mode switch. This is the control's default setting.

When set to 1, the control does not issue an error if the axis is not ready.

0 = Error if axis is not ready (default)

1 = Ignore error if axis is not ready

Bit 16: Disable AT Timing Check

When set to 0, the SERCOS AT (Drive Telegram) time checking is enabled. This is the control's default setting.

When set to 1, SERCOS AT time checking is disabled.

0 = Check SERCOS AT timing (Default)

1 = Do not check AT timing

C-0-0012 Serial Port B Device Type

This parameter selects the type of device that is connected to X16 on the control. Modifications to this parameter require that power to the control be cycled in order for the changes to take effect. The baud rate and options for these ports are configured in parameter C-0-0004 Serial Port B Setup. The allowable values are:

- 0 = off
- 1 = Host protocol (default)
- 2 = BTC06 teach pendant using VT100 mode

C-0-0012 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	1
Access:	read in any phase / write in phase 2

C-0-0013 Serial Port A Mode

This parameter selects the communication mode for X10 on the control. This selection takes effect immediately. The baud rate and options for this port is configured in parameter C-0-0003 Serial Port A Setup. The allowable values are:

- 232 = RS232 (default)
- 422 = RS422
- 485 = RS485

C-0-0013 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	232
Maximum value:	485
Default value:	232
Access:	read / write in any phase

RS-232

Connect to one device at a time according to RS-232 standard. Maximum cable length is 20 meters.

RS-422

Connect to one device at a time according to RS-422 standard. Maximum cable length is 200 meters.

RS-485

The control is a slave on a multi-drop ring with a host and up to 32 slaves, using the RS-485 standard. Maximum cable length is 200 meters.

C-0-0014 Serial Port B Mode

This parameter selects the communication mode for X16 on the control. This selection takes effect immediately. The baud rate and options for this port is configured in parameter C-0-0004 Serial Port B Setup.

C-0-0014 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	232
Maximum value:	485
Default value:	232
Access:	read / write in any phase

C-0-0016 Communication Time-out Period

This parameter adjusts the communication time-out period. The state of the communication error timer is set to enabled/disabled by start/stop commands from the serial device. The communication timer is reset by both a Timer Reset command and a change of state from disabled to enabled.

C-0-0016 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	milliseconds
Minimum value:	50
Maximum value:	2000
Default value:	2000
Access:	read / write in any phase

C-0-0017 Serial Port A Password

This parameter displays the current access level to the control over the X10 serial port. The available access levels are as follows:

C-0-0017	Access Level	Description
007	not defined (default)	no password is assigned and the user has full access to all control functions
###	Read/Write	password is assigned and the user has full access to all control functions
***	Read-only	password is assigned and the user only has read access

If this parameter displays "###" or "***", a password has been set and the user must enter the password in order to change the access level.

Note: Entering a string other than the correct password or cycling power to the control will change the control's current access level to "read-only".

Password Requirements:

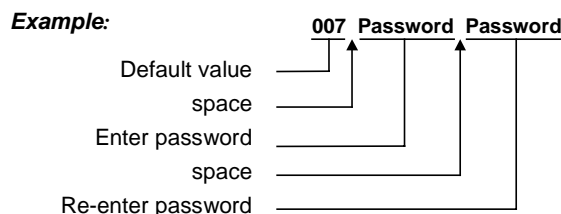
The password can contain 3 to 10 alpha and/or numeric characters. The password is not case sensitive and special characters such as "\$" or "%" are not allowed. Once set, the password is used to toggle between the different access levels.

Every time the password is entered, the access level will toggle between "###" (Read/Write) and "***" (Read-only).

Note: The password for serial port X10 can only be set and accessed while connected to the port.

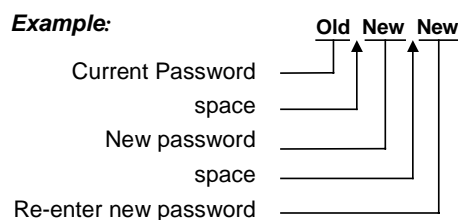
Setting a Password from the Default Value

When "007" is displayed in this parameter, use the following syntax to enter a password:



Changing an Existing Password

To change an existing password, enter the current password followed by the new password twice.



C-0-0017 Attributes

Data length:	10 character max
Data type:	unsigned character
Display format:	string
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	007 (not defined)
Access:	read / write in Phase 2

C-0-0018 Serial Port B Password

This parameter displays the current access level to the control over the X16 serial port. Refer to *C-0-0017 Serial Port A Password* for details.

C-0-0020 Transmitter Fiber Optic Length

This parameter adjusts the intensity of the output from the control's SERCOS transmitter, based on the length of the cable in meters. When using plastic fiber optic cable assemblies, set this parameter to match the length of the cable that is used between the control and the first drive's receiver (RX). For glass fiber, set this parameter to 50 m. Modifications to this parameter take effect when the control is in switch in and out of parameter mode.

C-0-0020 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	meters
Minimum value:	+0.0
Maximum value:	+2000.0
Default value:	+0.0
Access:	read / write in any phase

C-0-0021 User Watchdog Timer

The user watchdog timer enforces a time constraint on a user task specified in parameter C-0-0022.

Every time a nonzero timeout value is written to this parameter, a timer is triggered on the control. The task specified in parameter C-0-0022 must be completed within this time or error "508 User Watchdog Timeout" is issued. The timer is checked by the control every 50ms.

If a nonzero value is used, this parameter becomes active when the control is in SERCOS phase 4, there are no errors, and the task specified in C-0-0022 is running.

C-0-0021 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	milliseconds
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read / write in any phase

C-0-0022 User Watchdog Task ID

This parameter selects the task that must be running in order for the watchdog timer in C-0-0021 to be active. If the watchdog function is not used, set parameter C-0-0021 to 0. The allowable values are:

- 0 = no task
- 1 = task A
- 2 = task B
- 3 = task C
- 4 = task D

C-0-0022 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	4
Default value:	1
Access:	read / write in any phase

C-0-0035 PLC Communication Option (GPP only)

This parameter is used to initialize the handshaking between the PPC-R control and the MTS-R PLC interface. The control monitors the state of this parameter **only** at power up. To initialize handshaking between the control, set C-0-0035 to 1 and cycle power to the system.

C-0-0035 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	1
Default value:	0
Access:	read in any phase / write in phase 2

Jogging and Display Parameters (C-0-0042 through C-0-0056)

C-0-0042 World Large Increment

This parameter is a scalar, for all tasks, that is multiplied with *T-0-0025 Maximum Jog Increment* to set the world large increment distance used when jogging a coordinated motion axis. The world large increment distance is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Jog Control	Setting	Description
Registers 7-10	bit 1 = 0	nStep jogging (incremental)
	bit 6 = 1	large increment

Table 5-44: Register Setting for World Large Increment Jog Distance

Refer to *Registers 7- 10: Task Jog Control* in chapter 4 for details on jogging a coordinated axis.

C-0-0042 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	50
Access:	read / write in any phase

C-0-0043 World Small Increment

This parameter is a scalar, for all tasks, that is multiplied with *T-0-0025 Maximum Jog Increment* to set the world small increment distance used when jogging a coordinated motion axis. This parameter is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Control	Setting	Description
Registers 7-10	bit 1 = 0	nStep jogging (incremental)
	bit 6 = 0	small increment

Table 5-45: Register Setting for World Small Increment Jog Distance

Refer to *Registers 7- 10: Task Jog Control* in chapter 4 for details on jogging a coordinated axis.

C-0-0043 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	10
Access:	read / write in any phase

C-0-0045 World Fast Jog Speed

This parameter is a scalar, for all tasks, that is multiplied with *T-0-0026 Maximum Jog Velocity* to set the world fast jog speed when jogging a coordinated motion axis. This parameter is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Control	Setting	Description
Registers 7-10	bit 1 = 1	continuous
	bit 6 = 1	fast jog speed

Table 5-46: Register Setting for World Fast Jog Speed

Refer to *Registers 7- 10: Task Jog Control* in chapter 4 for details on jogging a coordinated axis.

C-0-0045 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	50
Access:	read / write in any phase

C-0-0046 World Slow Jog Speed

This parameter is a scalar, for all tasks, that is multiplied with *T-0-0026 Maximum Jog Velocity* to set the world slow jog speed when jogging a coordinated motion axis. This parameter is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Control	Setting	Description
Registers 7-10	bit 1 = 1	continuous
	bit 6 = 0	slow jog speed

Table 5-47: Register Setting for World Slow Jog Speed

Refer to *Registers 7- 10: Task Jog Control* in chapter 4 for details on jogging a coordinated axis.

C-0-0046 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	10
Access:	read / write in any phase

C-0-0052 Axis Large Increment

This parameter is a scalar that is multiplied with *A-0-0025 Maximum Jog Increment* to set the axis large increment distance used when jogging an axis in single-axis mode. The axis large increment distance is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Jog Control	Setting	Description
Registers 7-10	bit 1 = 0	nStep jogging (incremental)
	bit 6 = 1	large increment

Table 5-48: Register Setting for Axis Large Increment Jog Distance

Note: Axes in single-axis mode are jogged using their relevant axis control registers 11-18 and 209-240.

Refer to *Registers 7- 10: Task Jog Control* and *Registers 11-18: Axis Control* in chapter 4 for details on jogging an axis in single-axis mode.

C-0-0052 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	50
Access:	read / write in any phase

C-0-0053 Axis Small Increment

This parameter is a scalar that is multiplied with *A-0-0025 Maximum Jog Increment* to set the axis small increment distance used when jogging an axis in single-axis mode. The axis small increment distance is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Jog Control	Setting	Description
Registers 7-10	bit 1 = 0	nStep jogging (incremental)
	bit 6 = 0	small increment

Table 5-49: Register Setting for Axis Small Increment Jog Distance

Note: Axes in single-axis mode are jogged using their relevant axis control registers 11-18 and 209-240.

Refer to *Registers 7- 10: Task Jog Control* and *Registers 11-18: Axis Control* in chapter 4 for details on jogging an axis in single-axis mode.

C-0-0053 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	10
Access:	read / write in any phase

C-0-0055 Axis Fast Jog Velocity

This parameter is a scalar that is multiplied with *A-0-0026 Maximum Jog Velocity* to set the axis fast jog velocity used when jogging a an axis in single-axis mode. The axis fast jog velocity is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Jog Control	Setting	Description
Registers 7-10	bit 1 = 1	continuous
	bit 6 = 1	fast jog velocity

Table 5-50: Register Setting for Axis Fast Jog Velocity

Note: Axes in single-axis mode are jogged using their relevant axis control registers 11-18 and 209-240.

Refer to *Registers 7- 10: Task Jog Control* and *Registers 11-18: Axis Control* in chapter 4 for details on jogging an axis in single-axis mode.

C-0-0055 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	50
Access:	read / write in any phase

C-0-0056 Axis Slow Jog Velocity

This parameter is a scalar that is multiplied with *A-0-0026 Maximum Jog Velocity* to set the axis slow jog velocity used when jogging a an axis in single-axis mode. The axis slow jog velocity is used when bits 1 and 6 of the relevant task control register are set as follows:

Task Jog Control	Setting	Description
Registers 7-10	bit 1 = 1	continuous
	bit 6 = 0	slow jog velocity

Table 5-51: Register Setting for Axis Slow Jog Velocity

Note: Axes in single-axis mode are jogged using their relevant axis control registers 11-18 and 209-240.

Refer to *Registers 7- 10: Task Jog Control, Registers 11-18: Axis Control, and Registers 209-240: Axis Control* in chapter 4 for details on jogging an axis in single-axis mode.

C-0-0056 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percent
Minimum value:	1
Maximum value:	100
Default value:	10
Access:	read / write in any phase

C-0-0080 Maximum Number of Global Integers

This parameter defines the maximum number of global integers that will be created and accessible to the VisualMotion system. Modifications to this parameter require that power to the control be cycled in order for the changes to take effect. Global variables can be viewed using VisualMotion Toolkit by selecting **Data ⇒ Variables...**

C-0-0080 Attributes

Data length:	4 byte data
Data type:	unsigned long
Display format:	signed decimal
Units:	--
Minimum value:	512
Maximum value:	32767
Default value:	512
Access:	read / write in any phase

C-0-0081 Maximum Number of Global Floats

This parameter defines the maximum number of global floats that will be created and accessible to the VisualMotion system. Modifications to this parameter require that power to the control be cycled in order for the changes to take effect. Global variables can be viewed using VisualMotion Toolkit by selecting **Data** ⇒ **Variables...**

C-0-0081 Attributes

Data length:	4 byte data
Data type:	unsigned long
Display format:	signed decimal
Units:	--
Minimum value:	256
Maximum value:	32767
Default value:	256
Access:	read / write in any phase

C-0-0082 Save Global Variables Command

This parameter is used to save global integers and float variables to flash memory. When the control is powered, the values of all global variables are initialized to zero unless they were saved to flash memory before the control was powered down. The number of allowable global variables is determined by control parameters C-0-0080 (Global Integers) and C-0-0081 (Global Floats).

To save global variables...

1. Switch the control to parameter mode.
2. Edit C-0-0082 and set bits 1 and 2 to 1. C-0-0083, bits 1 and 2 display a 1 indicating a successful flash.
3. When completed, reset bits 1 and 2 to 0 and exit parameter mode.

Global variables are now stored in flash memory and can be reinitialized with saved values.

Note: Global variables can also be flash using the Data Editor in VisualMotion Toolkit.

C-0-0082 Attributes

Data length:	2 byte data
Data type:	unsigned short
Display format:	binary
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	0
Access:	read in any phase / write in phase 2

C-0-0083 Save Global Variables Status

This parameter is a status indicator for the *Save Global Variables Command* parameter C-0-0082. When the Save Global Variables Command is set, this parameter will indicate the communication status of the flash process. The status indications are described in the following table.

Bit 4	Bit 3	Bit 2	Bit 1	Description
0	0	0	0	No command to flash global variables
0	0	1	1	Command to flash global variables was successful
0	1	1	1	Command to flash global variables is processing.
1	1	1	1	Communication error, flash was not successful

C-0-0083 Attributes

Data length:	2 byte data
Data type:	unsigned short
Display format:	binary
Units:	--
Minimum value:	0
Maximum value:	15
Default value:	0
Access:	read-only

Program Management (C-0-0090 through C-0-0099)**C-0-0090 Download Block Size**

This parameter is used to set the block size that is used for user program downloads to the control. Refer to the "PD" command in *Chapter 13, Communication Protocols*, for details.

C-0-0090 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	115
Default value:	64
Access:	read / write in any phase

C-0-0091 Total Program Memory

This displays the total file memory on the control that can be used for user programs. This includes compiled programs and allocated points, variables, event and zone tables. The amount of total program memory varies based on the firmware version installed.

C-0-0091 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	bytes
Minimum value:	--
Maximum value:	--
Default value:	total available program memory of installed firmware
Access:	read-only

C-0-0092 Available Program Memory

This parameter displays the amount of memory the control currently has available for storage of user programs. This does not include fragmented memory. Additional memory can be freed by clicking the **Defragment** button in the *Program Management* window.

C-0-0092 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	bytes
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-0093 Contiguous Program Memory

This parameter displays the amount of memory the control currently has available for storage of user programs. This does not include fragmented memory. Additional memory can be freed by clicking the **Defragment** button in the *Program Management* window.

C-0-0093 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	bytes
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-0094 Maximum Executable Program Size

VisualMotion reserves a fixed amount of memory to store the currently active executable program. The executable program contains the instructions but not the points, events, variables, and labels. This parameter indicates the largest executable program that can be stored on the control. The program size can be checked against this number before the program is downloaded. If the control receives a user program that exceeds this number, communication error “!60 Executable program is too large” is displayed.

C-0-0094 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	bytes
Minimum value:	0
Maximum value:	0
Default value:	varies on firmware
Access:	read-only

C-0-0098 Initialization Delay

This parameter causes the control to delay for the specified number of seconds before it initializes the SERCOS ring after power-up. This prevents the control from issuing a "No drives were found on ring" error if I/O stations or drives take a long time to initialize the SERCOS ring.

C-0-0098 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	seconds
Minimum value:	0
Maximum value:	255
Default value:	4
Access:	read in any phase / write in phase 2

C-0-0099 Minimum SERCOS Cycle Time

This parameter sets the minimum SERCOS cycle time used by the control. During system initialization, the control uses this time as a starting value for the internal SERCOS cycle time layer. For applications that require more calculating time, such as with high axis counts, lower SERCOS baud rates, DAQ cards, or complex control motion such as coordinated motion, coordinated articulation or CAM Indexer, the control will perform an internal check and determine if this value is sufficient. If the time in this parameter is too small for performing the necessary calculations, the control will automatically change the internal SERCOS cycle time and modify C-0-0099, all drive's SERCOS cycle time parameter S-0-0002 and NC cycle time parameter S-0-0001 to match. If the value in this parameter is greater then the internal check performed by the control, the value in C-0-0099 will be used by the control and drives.

C-0-0099 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	milliseconds
Minimum value:	2000
Maximum value:	16000
Default value:	2000
Access:	read in any phase / write in phase 2

Note: If larger SERCOS cycle times are used and communication errors continue, increase the DDE Server's *Response Timeout* under **Setting** ⇒ **Server Configuration** menu selection.

System Status (C-0-0100 through C-0-0127)

System status messages are available through the communication ports and the user program, and provide the current status of the VisualMotion system.

C-0-0100 Control Firmware Version

This parameter displays the firmware version number issued by the control.

Format:

```
PSM01*-GPP-09V28-MS
|         |         |_____ Firmware version
|         |_____ Firmware type
|_____ Hardware type
```

C-0-0100 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	installed firmware version
Access:	read-only

C-0-0101 Control Hardware Version

This parameter displays the control's hardware version.

Format:

```
PPC-R0x.2N-N-NN-NN-NN-FW Rev:3.0
|         |         |_____ |_____ Board revision
|         |         |_____ Configuration
|         |_____ Hardware revision no.
|_____ Hardware platform
```


C-0-0101 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	installed hardware configuration
Access:	read-only

C-0-0102 Control Version Date

This parameter displays the release date for the firmware detected by the control.

C-0-0102 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	installed firmware release date
Access:	read-only

C-0-0103 Allowable Drive Address Range

This parameter displays the maximum SERCOS drive address that can be set in any SERCOS device within the control's fiber optic ring. The SERCOS drive address is set using the S2 and S3 rotary selector switches. Refer to *SERCOS Drive Address Settings* in chapter 10 of the *VisualMotion Project Planning* manual for details.

C-0-0103 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	40
Maximum value:	40
Default value:	40
Access:	read-only

C-0-0104 Bootloader Firmware Version

This parameter displays the firmware version of the installed Bootloader in the PSM memory card.

C-0-0104 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	installed Bootloader firmware
Access:	read-only

C-0-0120 Operating Mode

This parameter displays the control's current mode of operation. The allowable values are:

- 0 = Initializing control
- 1 = Parameter Mode
- 2 = Manual / Automatic / Run Mode
- 3 = reserved

C-0-0120 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	0
Access:	read-only

C-0-0121 SERCOS Communication Phase

This parameter displays the current SERCOS initializing phase of the control.

Note: The drives can be at a lower SERCOS phase than the control if an error exists at the drive level.

C-0-0121 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	4
Default value:	current SERCOS phase is displayed
Access:	read-only

C-0-0122 Diagnostic Message

This parameter displays the current system status or error message issued by the control. During initialization, a **Msg1** icon in the initialization task sets this message.

For example:

400 Emergency Stop

C-0-0122 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	Highest priority message is displayed
Access:	read-only

Refer to the *VisualMotion 9 Trouble Shooting Guide* for a complete listing of all system status and error codes.

C-0-0123 Diagnostic Code

This parameter displays the current system status or error message number issued by the control.

For example:

4 for 004 Emergency Stop

C-0-0123 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	Highest priority code is displayed
Access:	read-only

Refer to the *VisualMotion 9 Trouble Shooting Guide* for a complete listing of all system status and error codes.

C-0-0124 Extended Diagnostic

This is a dynamic system message used to provide additional diagnostic information for a status warning or error message C-0-0122 Diagnostic Message.

C-0-0124 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	dependent on C-0-0122
Access:	read-only

Refer to the *VisualMotion 9 Trouble Shooting Guide* for a complete listing of all system status and error codes.

C-0-0125 System Timer Value

This general-purpose timer continuously counts in milliseconds while the control is running. It can be read into an integer variable via the **Parameter Transfer** icon to provide timing for a section of a VisualMotion user program, or its incremental value can be used to time a process. It is a 31 bit counter with a maximum count of 2,147,483,647 (2^{31}), after which it rolls over to 0 and continues counting. It can be set to any value by the user program or the user interface.

C-0-0125 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	milliseconds
Minimum value:	0
Maximum value:	2^{31}
Default value:	increments by 1 every msec
Access:	read / write in any phase

C-0-0126 Date and Time

This parameter contains the date and time used by the control for the diagnostic log. VisualMotion controls do **not** contain a real-time clock. For this reason, the host device must set the date and time after the control is powered up.

When using VisualMotion Toolkit, the date and time can be set by clicking on the *Set Time* button under menu selection **Diagnosics** ⇒ **System** or by directly modifying this parameter.

When using Direct ASCII Communication, the control requires 'SET' before the new date and time:

```
SET 06-22-2000 15:33:00
```

Format:

Month-Day-Year Hour-Minute-Second in 24 hour format
06-22-2000 15:33:00

C-0-0126 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	01-01-1980 00:00:00 (at power up)
Access:	read / write in any phase

C-0-0127 Current Control Temperature (GPP only)

This parameter displays the current internal temperature for the control hardware in degrees Celsius.

C-0-0127 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	degrees
Minimum value:	-32768
Maximum value:	32768
Default value:	current operating temperature
Access:	read-only

Note: The maximum allowable internal temperature for the control is 70°C (158°F) when operating at a maximum ambient temperature of 45°C (113°F).

C-0-0128 Elapsed Time Operational Counter

This parameter displays the value of the Elapsed Time Counter which continuously counts the number of system operation hours. The value in this parameter is updated 10,000 times every hour. Elapsed Time Counter values are stored in the control's EEPROM every 10 minutes so that the counter is not disrupted when the control is powered off or when firmware is changed, if the new firmware is the same or a later version. Versions of firmware earlier than GPP 9 do not contain this parameter.

C-0-0128 Attributes

Data length:	4 byte data
Data type:	float
Display format:	signed decimal
Units:	hours
Minimum value:	0
Maximum value:	1,080,000.00
Default value:	--
Access:	read-only

C-0-0142 Card Label String

An alpha-numeric descriptive name, up to a maximum of 80 characters, assigned to the control is stored in this parameter (e.g., Master PPC). It has no functional significance.

C-0-0142 Attributes

Data length:	variable 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	<undefined>
Access:	read / write in any phase

C-0-0166 Save Built CAM to Flash ID

This parameter specify a valid id number, between 1 and 37, for a built control CAM that will be flashed when C-0-0167 Save Built CAM to Flash Command is used.

C-0-0166 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	--
Units:	--
Minimum value:	0
Maximum value:	37
Default value:	0
Access:	read / write in any phase

C-0-0167 Save Built CAM to Flash Command

This parameter sets the command to flash the built control CAM specified in C-0-0166 Save Built CAM to Flash ID.

To save a built control CAM...

1. Switch the control to parameter mode.
2. Edit C-0-0167 and set bits 1 and 2 to 1.
C-0-0168, bits 1 and 2 display a 1 indicating a successful flash.
3. When completed, reset bits 1 and 2 to 0 and exit parameter mode.

Note: The **Command** icon can be used to execute and reset this command.

C-0-0167 Attributes

Data length:	2 byte data
Data type:	unsigned short
Display format:	binary
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	0
Access:	read in any phase / write in phase 2

C-0-0168 Save Built CAM to Flash Status

This parameter is a status indicator for the *Save Built CAM to Flash Command C-0-0167*. When the command is set, this parameter will indicate the status of the flash process and the ID number of the control CAM that was flashed. The status indications are described in the following table.

Bit 4	Bit 3	Bit 2	Bit 1	Description
0	0	0	0	No command to flash CAM
0	0	1	1	Command to flash CAM was successful and CAM ID has been encoded in the upper-byte
0	1	1	1	Command to flash CAM is processing
1	1	1	1	Communication error, flash was not successful

Table 5-52: Lower-byte Flash Status

The upper-byte (bits 9-16) of this parameter will display the CAM's ID number upon a successful flash. If the command to flash fails, the lower-byte will display a binary 15 and an error code will be displayed in the upper-byte. The following table lists the error codes that can occur:

Error Code	Description
23	"Insufficient Program Space" There is not enough free memory in the flash file system to save the specified CAM.
74	"Error in command execution" A general error occurred during the control's attempt to save the specified CAM to flash.
104	"Invalid CAM ID" The CAM ID written to C-0-0166 must be between 1 and 37.
105	"CAM Does Not Exist" The CAM specified in C-0-0166 does not exist, it has not been built or downloaded.
106	"CAM is Not Ready" The CAM specified in C-0-0166 is not in a ready state. Its build process may not have yet completed.
107	"CAM is In Use" The CAM specified in C-0-0166 is currently in use.

Table 5-53: Upper-byte Flash Status

C-0-0168 Attributes

Data length:	2 byte data
Data type:	unsigned short
Display format:	binary
Units:	--
Minimum value:	0
Maximum value:	--
Default value:	0
Access:	read-only

Control Processor Usage Status (C-0-0200 through C-0-0203)**C-0-0200 Current Load due to Motion**

This parameter displays the current amount of time required by the control's processor to process high priority motion task as a percentage of parameter C-0-0099(SERCOS Cycle Time).

For example:

If C-0-0200 = 10% and C-0-0099 = 2 ms, then the amount of time to process high priority motion task is 200 µs.

C-0-0200 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percent
Minimum value:	+0.0
Maximum value:	+100.0
Default value:	current processing time is displayed
Access:	read-only

C-0-0201 Peak Load due to Motion

This parameter displays the peak amount of time encountered by the control for processing high priority motion task as a percentage of parameter C-0-0099 (SERCOS Cycle Time). This parameter is written to by the control during Phase 4. This parameter is reset at power-down or when switching to Phase 2.

C-0-0201 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percent
Minimum value:	+0.0
Maximum value:	+100.0
Default value:	last peak processing time encountered
Access:	read-only

C-0-0202 Current Load due to I/O

This parameter displays the current amount of time required by the control's processor to process all configured Inputs and Outputs as a percentage of parameter C-0-3001, bit 5 (I/O Mapper scan time).

C-0-0202 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percent
Minimum value:	+0.0
Maximum value:	+100.0
Default value:	current processing time is displayed
Access:	read-only

C-0-0203 Peak Load due to I/O

This parameter displays the peak amount of time encountered by the control for processing all configured Inputs and Outputs as a percentage of parameter C-0-3001, bit 5 (I/O Mapper scan time).

C-0-0203 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percent
Minimum value:	+0.0
Maximum value:	+100.0
Default value:	last peak processing time encountered
Access:	read-only

Link Ring Parameters (C-0-0300 through C-0-0303)

C-0-0300 Link Ring Control Word

This parameter is used for configuring a control as a Link Ring participant. The type of ring structure is also set in the control word.

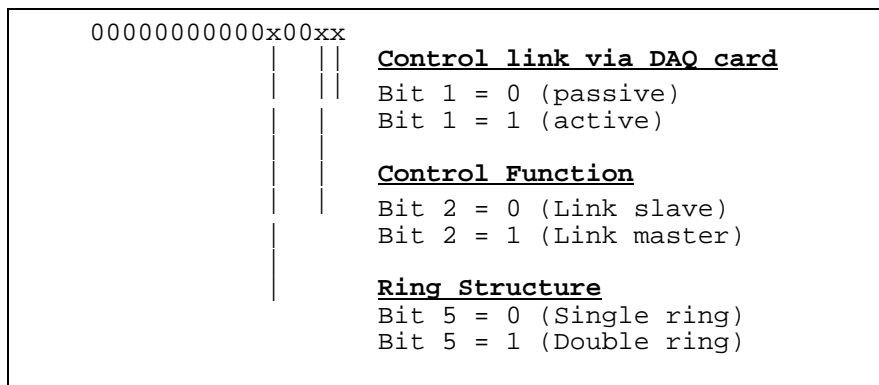


Fig. 5-5: Bit Description C-0-0300

The corresponding bits are listed in the tables below.

C-0-0300 Bit 2	C-0-0300 Bit 1	Control behavior in the link	Control function in the link
1	1	Active participant	Link master
0	1	Active participant	Link slave
x	0	Passive participant	repeater

C-0-0300 Bit 5	Ring structure
0	Single ring
1	Double ring

C-0-0300 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read in any phase / write in phase 2

C-0-0301 Link Ring Primary Fiber Optic Length

This parameter is used to adjust the output power of the DAQ card to the length of the connected primary fiber optic cable. The length indicated relates to the fiber optic cable from X52 (TX) to the next connected DAQ, X53 (RX).

C-0-0301 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	meters
Minimum value:	+0.0
Maximum value:	+2000.0
Default value:	0.0
Access:	read / write in any phase

C-0-0302 Link Ring Secondary Fiber Optic Length

This parameter is used to adjust the output power of the DAQ card to the length of the connected secondary fiber optic cable. The length indicated relates to the fiber optic cable from X70 (TX) to the next connected DAQ, X71 (RX).

C-0-0302 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	meters
Minimum value:	+0.0
Maximum value:	+2000.0
Default value:	0.0
Access:	read / write in any phase

C-0-0303 Link Ring MDT Error Counter

Every Link Ring slave counts the number of invalid master data telegrams (MDT). The control will issue error "541 Link Ring Error" and react with the parameterized error reaction if more than one MDT is invalid.

With a double MDT failure, the "MDT error counter" stops counting. The error counter stops at 65535. In the case of a severely disrupted transmission, the value 65535 will be displayed after an extended period of time.

C-0-0303 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read-only

Ethernet Parameters (C-0-0400 through C-0-0408)

C-0-0400 Ethernet Card IP Address (GPP only)

This parameter contains a unique IP Address, specified in dot notation i.e., 172.16.11.200, assigned to the optional Ethernet card. This parameter is valid for standard Ethernet TCP/IP and Ethernet/IP communication using 10mb or 100mb cards. The IP address in this parameter must match the IP address configured in the DDE server.

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

Note: An IP address is provided by the IT department.

C-0-0400 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	
Access:	read / write in any phase

C-0-0401 Ethernet Card Subnet Mask (GPP only)

This parameter contains a subnet mask, specified in dot notation, used to determine what subnet is assigned to the IP address in parameter C-0-0400. This parameter is valid for standard Ethernet TCP/IP and Ethernet/IP communication using 10mb or 100mb cards.

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

C-0-0401 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	
Access:	read / write in any phase

C-0-0402 Ethernet Card Gateway IP Address (GPP only)

This parameter contains the Gateway IP Address, specified in dot notation i.e., 172.16.1.1, assigned to the optional Ethernet card. This parameter is valid for standard Ethernet TCP/IP and Ethernet/IP communication using 10mb or 100mb cards. A Gateway IP address identifies the router to which the IP Address is assigned. This parameter can be configured in VisualMotion Toolkit by selecting **Tools** ⇒ **Control Settings** and selecting the **Network Card** tab.

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

C-0-0402 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	
Access:	read / write in any phase

C-0-0403 Ethernet Card CIF Network Control (GPP only)

This parameter specifies whether the control is connected to a half or full duplex switch. To change the duplexing mode for network communication, the user must enter the following text using all caps.

- HALF
- FULL

This parameter is valid for standard Ethernet TCP/IP and Ethernet/IP communication using the 10mb card **only**. The 100mb card supports auto-negotiation and does not require setting the duplexing mode.

C-0-0403 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	last configured duplex mode
Access:	read / write in any phase

C-0-0404 Ethernet Card Network Access Control (GPP only)

This parameter is used to enable the control's network access level (Bit 1) and to enable the optional Ethernet card as either standard Ethernet TCP/IP or Ethernet/IP (Bit 2). This parameter is valid for standard Ethernet TCP/IP and Ethernet/IP communication using 10mb or 100mb cards.

Modifications to this parameter require that power to the control be cycled in order for the changes to take effect.

Network Access Level (Bit 1)

Bit 1 of this parameter works in conjunction with C-0-0405 Ethernet Card Network Password to enable the control's network access level. The available access levels are Read-only, Read/Write or No Access.

Note: Before enabling the control's network access level in this parameter, a password should be set in C-0-0405. If no password is set, the network access level can not be set to "No Access".
The user will simply access the control in either the Read/Write or Read-only levels (Determined by C-0-0405).

C-0-0405 password levels:
 007 = Not defined (default)
 ### = Read/Write
 *** = Read-only

Refer to *C-0-0405 Ethernet Card Network Password* for details.

The following table shows the combinations that are required for both control parameters C-0-0404 and C-0-0405.

C-0-0404 (Bit 1)	C-0-0405	Access Level	Description
0000000000000000x	007	not defined (default)	user has full access to all control functions
00000000000000000	***	Read-only	only read access has been allowed
0000000000000000x	###	Read/Write	the user has full access to all control functions
00000000000000001	***	No Access	the control can not be access via Ethernet unless the correct password is entered in C-0-0405

Table 5-54: Enable Network Access Level C-0-0404, Bit 1

Once C-0-0404, bit 1 is set to 1, the network access level can be toggle, using a serial or Ethernet connection, between "****" (No Access) or "###" (Read/Write) by entering the correct password in C-0-0405.

Note: Entering a string other than the correct password or cycling power to the control will change the control's network access to "No Access". Entering the password again will change the control's network access back to "Read/Write".

Enable Ethernet/IP (Bit 2)

Bit 2 of this parameters enables the optional Ethernet card as either standard Ethernet TCP/IP or Ethernet/IP communication.

C-0-0404 (Bit 2)	Description
0000000000000000	standard Ethernet TCP/IP enabled (default)
0000000000000010	Ethernet/IP enabled

Table 5-55: Enable Ethernet C-0-0404, Bit 2

C-0-0404 Attributes

Data length:	2 byte data
Data type:	unsigned short
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read in any phase / write in phase 2

C-0-0405 Ethernet Card Network Password (GPP only)

This parameter displays the current access level to the control over an Ethernet connection. The available access levels are as follows:

C-0-0405	Access Level	Description
007	not defined (default)	no password is assigned and the user has full access to all control functions
###	Read/Write	password is assigned and the user has full access to all control functions
***	Read-only	password is assigned and the user only has read access

If this parameter displays "###" or "***", a password has been set and the user must enter the password in order to change the access level.

Note: Entering a string other than the correct password or cycling power to the control will change the control's access level to "Read-only".

Password Requirements:

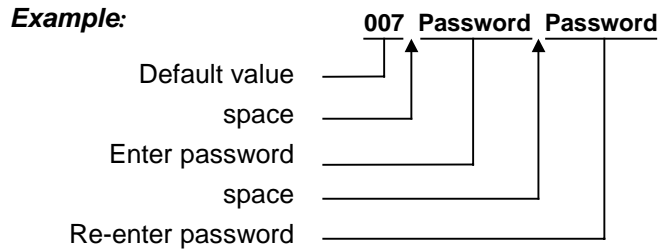
The password can contain 3 to 10 alpha and/or numeric characters. The password is not case sensitive and special characters such as "\$" or "%" are not allowed. Once set, the password is used to toggle between the different network access levels.

Every time the password is entered, the access level will toggle between "###" (Read/Write) and "***" (Read-only).

Note: If parameter C-0-0404 is set to 1 and C-0-0405 is set to "Read-only", the control will be set to "No Access" for all users connected to the control via a network connection. Refer to *C-0-0404 Ethernet Card Network Access Control* for details.

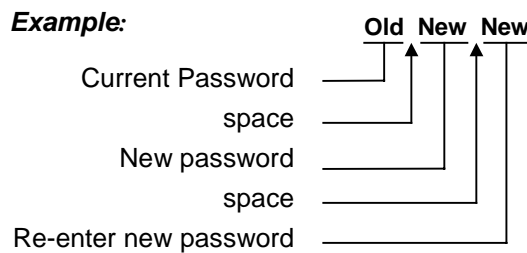
Setting a Password from the Default Value

When "007" is displayed in this parameter, use the following syntax to enter a password:



Changing an Existing Password

To change an existing password, enter the current password followed by the new password twice.



C-0-0405 Attributes

Data length:	10 character max
Data type:	unsigned character
Display format:	string
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	007 (not defined)
Access:	read / write in any phase 2

C-0-0406 CIF Ethernet Card Hardware ID (GPP only)

This parameter displays the current optional Ethernet card hardware typecode. When no Ethernet card is installed, the parameter displays *<no card present>*.

C-0-0406 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	current Ethernet hardware typecode if installed
Access:	read-only

C-0-0407 CIF Ethernet Card Firmware Version (GPP only)

This parameter displays the current optional Ethernet card firmware version. When no Ethernet card is installed, the parameter displays *<no card present>*.

C-0-0407 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	current Ethernet firmware version if installed
Access:	read-only

C-0-0408 CIF Ethernet Driver Version (GPP only)

This parameter displays the current optional Ethernet card interface driver version. If no Ethernet card is installed, the parameter displays *<no card present>*.

C-0-0408 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	current Ethernet CIF driver if installed
Access:	read-only

Initialization Task Parameters (C-0-0522 through C-0-0537)

C-0-0522 Init. Task Diagnostic Message

This parameter displays the current diagnostic message and/or code for the Initialization task. During normal operation, a **Msg1** icon in the user program sets this message. If an error occurs during task execution, this diagnostic message is overwritten with an error message. Refer to the *VisualMotion Troubleshooting Guide* for error messages.

C-0-0522 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

C-0-0523 Init. Task Status Message

This parameter displays the current status message for the initialization task. A **Msg1** icon in the user program sets this message as an aid to the operator or for debugging purposes. This message is not overwritten with an error message, allowing debugging of an error condition set in the Init. Task Diagnostic Message.

C-0-0523 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

C-0-0530 Init. Task Current Instruction Pointer

This parameter returns a hexadecimal value equal to the initialization task's execution address (i.e. the instruction pointer). The hex value is an offset from the start of the program. This parameter is primarily used for debugging and troubleshooting programs.

For example:

"0x000000F0" indicates that the program counter is at 0xF0, or 240 bytes from the start of the program.

C-0-0530 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x00000000
Maximum value:	0xFFFFFFFF
Default value:	read by control
Access:	read-only

C-0-0531 Init. Task Current Instruction

This parameter displays the mnemonic for the current instruction and the first 2 arguments of the instruction. The mnemonic is in the base code format generated by the control's compiler. This parameter is primarily used for debugging and troubleshooting programs.

C-0-0531 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

C-0-0532 Init. Task Pointer at Error

This status parameter displays the instruction pointer where the last task error occurred.

C-0-0532 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x00000000
Maximum value:	-0xFFFFFFFF
Default value:	read by control
Access:	read-only

C-0-0533 Init. Task Composite Instruction Pointer

This parameter dynamically displays a flag and an instruction pointer indicating the relative memory address where a program instruction is being executed. This parameter is primarily used for debugging and troubleshooting programs.

C-0-0533 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

C-0-0535 Init. Task Current Subroutine

This parameter indicates the current subroutine being executed with the function number and name. If function number and name information is not included in the user program file, the string "NONE" is returned.

C-0-0535 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

C-0-0536 Init. Task Stack Variable Data

This parameter displays the current stack local variable data. Stack variables are valid only while the program flow is within a task or subroutine. Maximum number of stack variables is 16. If there are no arguments or local variables in a task or function, the string "NONE" is returned.

C-0-0536 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	NONE
Access:	read-only

C-0-0537 Init. Task Subroutine Breakpoint

This parameter is reserved for future development.

BTC06 Teach Pendant (C-0-0801 through C-0-0814)

C-0-0801 Pendant Protection Level 1 Password

This parameter defines a four-digit numeric password that prevents entry into protected menus. If set to 0, the password is disabled.

Note: If the password is used, the pendant protection level bits (Register 1, bits 15 and 16) do not function.

C-0-0801 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	9999
Default value:	0
Access:	read / write in any phase

C-0-0802 Pendant Protection Level 2 Password

This parameter is reserved for future development.

C-0-0803 Pendant User Accessible Floats Section

This parameter defines the maximum allowable range for program floats to be user accessible from the BTC06. The operator can view all the program floats, but the operator can only access the program floats, up to the number set in this parameter. If the operator needs to change a program float greater than the number in this parameter, then the operator can either enter a password or set the pendant level protection bits (System Control Register 1, bits 15 & 16).

For example:

User Accessible Program Float Section = 10

When the operator selects Table Edit Menu/Float Table Menu, the operator can only access the first ten floats. The programmer is responsible for structuring the program floats properly. The allowable selections are as follows:

- 0 = no program floats are accessible
- -1 = all program floats are accessible
- n = number of defined program floats

C-0-0803 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	-1
Maximum value:	32767
Default value:	-1
Access:	read / write in any phase

C-0-0804 Pendant User Accessible Integers Section

This parameter defines the maximum allowable range for program integers to be user accessible from the BTC06. The operator can view all the program integers, but the operator can only access the program integers, up to number set in this parameter. If the operator needs to change a program integer greater than the number in this parameter then the operator can either enter a password or set the pendant level protection bits (System Control Register 1, bits 15 & 16).

For example:

User Accessible Program Integer Section = 10

When the operator selects Table Edit Menu/Integer Table Menu, the operator can only access the first ten integers. The programmer is responsible for structuring the program integers properly. The allowable selections are as follows:

- 0 = no program integers are accessible
- -1 = all program integers are accessible
- n = number of defined program integers

C-0-0804 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	-1
Maximum value:	32767
Default value:	-1
Access:	read / write in any phase

C-0-0805 Pendant Start of User Accessible Registers

This parameter defines the starting register that is accessible to the operator. The operator can view all the registers, but the operator can only access the registers beginning with C-0-0805 and ending with C-0-0806. If the operator needs to change a bit in a register outside the window of this parameter, then the operator can either enter a password or set the pendant level protection bits (System Control Register 1, bits 15 & 16). When the Register I/O Menu is selected on the BTC06, the first register to be displayed is the number stored in the Start of User Accessible Registers parameter.

C-0-0805 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	1024
Default value:	1
Access:	read / write in any phase

C-0-0806 Pendant End of User Accessible Registers

Refer to the description of *C-0-0806 Pendant End of User Accessible Registers* for details.

C-0-0806 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	1024
Default value:	1024
Access:	read / write in any phase

C-0-0807 Pendant Password Timeout

This parameter sets a timeout on the BTC06 password when using VT100 terminal firmware. After the password is entered (C-0-0801 Pendant Protection Level 1 Password), the user can access any screens requiring the password for the time set in this parameter. When a key is pressed, the timer is reset. After the timer expires, the password is again required. If the timeout is set to 0, the password is always required.

C-0-0807 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	seconds
Minimum value:	0 (disabled)
Maximum value:	3600 (one hour)
Default value:	30
Access:	read / write in any phase

C-0-0810 TPT Message and Prompt Control Word

This parameter is used to display a user task status message in the top two lines of the BTC06 and it can prompt the user for data entry into a variable. This parameter is used for the BTC06 when using VT100 terminal firmware.

00000000000000000000
Bit 1
Bits 1-8: Variable ID number
Bits 9-11: Variable type
Bits 12-14: Task ID
Bits 15-16: Control and Status

Fig. 5-6: Bit Description C-0-0810

C-0-0810 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read / write in any phase

Bits 1-8: Variable ID Number

This ID number identifies the variable as a binary bit number between 0 and 255.

For example:

175 (decimal) is set as 10101111 (binary)

Bits 9-11: Variable Type

These bits identify the variable type for the number set in bits: 1-8. The following variable types are supported:

Bits 9-11	Value	Description
000	0	Integer Variable
001	1	Float Variable
010	2	Global Integer Variable
011	3	Global Float Variable

Bits 12-14: Task ID

These bits identify which VisualMotion task is associated with the selected variables. The following selections are available:

Bit 12-14	Value	Description
000	0	Inactive (no messaging)
001	1	Use task A status message
010	2	Use task B status message
011	3	Use task C status message
100	4	Use task D status message

Bits 15-16: Control and Status

These bits set the control and status displayed on the BTC06.

Bit 15-16	Value	Description
00	0	Done (status)
01	1	New prompt request for input (prompt for data entry) or data entry abort (when bits 1-8 = 0 and bits 9-11 = 0)
10	2	Operator input active (status)
11	3	Operator input error (status)

VisualMotion Programming

Use the “Message” icon to create the message to be displayed on the first two lines of the BTC06.

Use the VisualMotion “Calculator” icon to build an integer that corresponds to the desired bit settings for parameter C-0-0810.

Use the “Parameter Transfer” icon to transfer the integer to parameter C-0-0810. The integer will be converted to a 16-bit binary word and processed in this parameter.

Note: The programmer should set up the entire word before writing it to the parameter.

The task’s status/prompt message appears on the first two lines of the BTC06 display. The programmer is responsible for formatting the message and accounting wrap-around. The BTC06 task keeps track of message changes and updates the display accordingly.

A task’s status/prompt message may contain only one data entry field. Field size and location are determined by the following conventions:

Data entry fields are indicated by one or more ‘#’ signs. Field size is determined by the number of consecutive ‘#’ signs used.

Data entry fields may contain a decimal sub-field.

For example:

#####.##

Searching left to right, the first ‘#’ sign found determines the field’s location.

For example:

“Please enter part #: #####”. The first ‘#’ sign (“part #”) would erroneously be used for data entry.

The absence of ‘#’ signs in a prompt message forces a default field of 12 characters at the end of a message.

In the displayed message, ‘#’ signs are blanked out and the cursor is placed left justified in the field.

When writing the message in the task, the first line can be terminated by using the ‘~’ character. This makes for easy formatting of a message with a prompt on the second line.

Note: The functionality of the VisualMotion Message Setup Box remains unchanged. A variable can still be displayed in the message by using ‘%s’ in a message. A variable that requires data entry must be entered as a series of ‘#’ signs.

Note: The variable defined by the values in bits 1-8 and bits 9-11 is displayed in the task status message. If the variable number and type are not defined, a formatting string (consisting of one or more ‘#’ signs) that is entered in the VisualMotion Message Icon will be displayed as ‘#’ sign(s) instead of as the variable.

Control and Status**Prompt for user input**

When the BTC06 task sees the new prompt status (binary 01 in bits 15 and 16), it prompts the operator for data entry. As soon as the operator begins to enter a value, the BTC06 sets the status to a binary 10 in bits 15 and 16 indicating that the operator is in the process of entering numerical data. No additional messages can be placed until the data entry is complete (press the OK or ESC key) or a Data entry abort is commanded. When the operator presses the OK or ESC key, the BTC06 task sets the status to a binary 00, (Done in bits 15 and 16), indicating that data input is

complete and successful (no input error). The programmer should then check the input against internal maximum and minimum values.

Data entry abort If desired, the programmer can use the VisualMotion “Parameter Transfer” icon to force an Abort Message by setting a binary 01 status in bits 15 and 16 with the variable number and type set to 0. This abort can override the user input when a certain condition is met in the system (e.g. a hazardous condition).

Data input error When a data input error occurs, the BTC06 flashes the data field for 3 seconds. Bits 15 and 16 are set to a binary 11, indicating the data input error, and the variable remains unchanged.

C-0-0811 User Task Controlled Menu ID for TPT

This parameter allows the programmer to control or “force” which menus the user can access during a task. This parameter is used for the BTC06 when using VT100 terminal firmware. The allowable selections are as follows:

0 = Inactive	10 = Global Float Table Edit Menu
1 = Main menu	11 = Jog Menu
2 = Program Menu	12 = Control Menu
3 = Table Edit Menu	13 = Register I/O Menu
4 = Absolute Table Edit Menu	14 = Parameter Menu
5 = Relative Table Edit Menu	15 = Card Parameter Menu
6 = Event Table Edit Menu	16 = Task Parameter Menu
7 = Integer Table Edit Menu	17 = Axis Parameter Menu
8 = Float Table Edit Menu	18 = Drive Parameter Menu
9 = Global Integer Table Edit Menu	19 = Diagnostic Menu

C-0-0811 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	19
Default value:	0
Access:	read / write in any phase

The BTC06 task continually scans parameter C-0-0811 for a non-zero value. As long as this parameter remains, zero, VisualMotion user tasks have no menu controls over the BTC06. However, when this parameter is set to a non-zero value, VisualMotion user tasks have complete control over BTC06 menus.

Writing a zero into C-0-0811 relinquishes user task control over BTC06 menus. The BTC06 continues to display the last active menu, but now the operator can freely select menus.

The programmer selects the desired active menu in a user task by writing a menu ID number into C-0-0811. At this point, the user can move only to adjoining menus lower in hierarchy, but not back to the higher menus.

For example:

If the control menu is “forced”, the user can move from there to the diagnostic menu, but not back to the main menu. To prevent movement

between menus, the programmer can use Register 92-94 to mask the functionality of the F-keys.

VisualMotion Programming

For each menu to be “forced” during a task, the programmer writes the corresponding menu ID number into C-0-0811, using the VisualMotion “Parameter Transfer” icon. The BTC06 task knows that a new menu request has been made by seeing the transition of C-0-0811 from its current value to a different value. The transition triggers the menu to change.

Note: A menu that may cause motion, such as the jog menu, checks that the selected task is stopped before motion is allowed. The error: “User task must be stopped” is issued if this is not the case (e.g., jogging is allowed only during manual or auto mode, but not while the task is running).

Motion menus check whether the task in the control has any motion queued to the path planner. For example, if a cycle stop is executed while motion is active, any attempt to jog results in the following error: “User task has motion pending.” To allow jogging, all motion must be cleared from the path planner by switching to manual mode or performing a coordinated abort in the VisualMotion Stop Icon.

IMPORTANT: Parameter C-0-0811 must be active ($\neq 0$) for parameters C-0-0812, C-0-0813, and C-0-0814 to be functional.

IMPORTANT: System errors are handled in the same way, regardless of the user’s ability to control menu selection. System errors automatically transfer control to the diagnostic screen.

C-0-0812 User Task Controlled Task ID for TPT

This parameter allows the programmer to control or “force” which task motion system is displayed on the BTC06 for all axes and instructions defined in the active task. The user can also select the displayed task motion system by choosing it from the task menu. Using parameter C-0-0812, the programmer chooses the task for the user. This parameter is used for the BTC06 when using VT100 terminal firmware.

C-0-0812 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	0
Access:	read / write in any phase

Note: Parameter C-0-0811 must be active ($\neq 0$) for parameter C-0-0812 to be functional.

C-0-0813 User Task Controlled Axis Number for TPT

This parameter allows the programmer to control or “force” which single axis (defined by that axis’ SERCOS address) is to be jogged. (The user can also select the axis by choosing it from the Jog Menu.) Using parameter C-0-0813, the programmer chooses the axis for the user. This parameter is used for the BTC06 when using VT100 terminal firmware.

If an invalid axis or point number is found, the BTC06 responds by issuing an error message. It is the programmer’s responsibility to ensure these parameters are set appropriately before taking control of a menu.

C-0-0813 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	100
Default value:	1
Access:	read / write in any phase

Note: Parameter C-0-0811 must be active (≠0) for parameter C-0-0813 to be functional.

C-0-0814 TPT Data Transaction Word

This parameter allows the programmer to:

- monitor the user’s index value in a particular BTC06 menu (regardless of the user’s permission to change the field value).
- direct the user’s data entry in a menu (e.g., to direct the point to be taught in the Jog Menu, the register number in the Register Menu, the current point number in the Float Table Menu, etc.) with the ability to assign the user read-only (“lock”) or write (“set”) privileges.

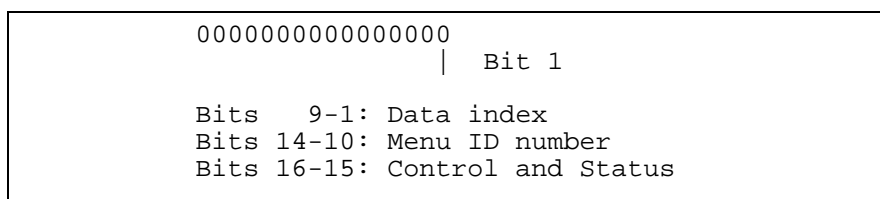


Fig. 5-7: Bit Description C-0-0814

C-0-0814 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read / write in any phase

Bits 9-1: Data Index

The data index is dependent on the selected menu ID number. These bits can display the value for a selected menu.

For example:

If 13, Register menu is selected for bits 10-14, then bit 1-9 will display the register number being requested or being written in as a 16-bit binary word.

Bits 14-10: Menu ID Number

The following table lists the available menu selections along with its index type.

Bits 14-10	Menu No.	Menu Name	Index Type
00001	1	Main Menu	not defined
00010	2	Program Menu	line number
00011	3	Table Edit Menu	not defined
00100	4	Absolute Table Menu	point number
00101	5	Relative Table Menu	point number
00110	6	Event Table Menu	event number
00111	7	Integer Table Menu	integer number
01000	8	Float Table Menu	float number
01001	9	Global Integer Menu	global integer number
01010	10	Global Float Menu	global float number
01011	11	Jog Menu ABS Menu	point number
01100	12	Control Menu	not defined
01101	13	Register Menu	register number
01110	14	Parameter Table Select Menu	not defined
01111	15	Card Parameter	not defined
10000	16	Task Parameter	not defined
10001	17	Axis Parameter	not defined
10010	18	Drive Parameter	not defined
10011	19	Diagnostic Menu	not defined

Bits 16-15: Status and Control

These bits set the status and control for the BTC06 display.

Bit 16-15	Description
00	Done (status)
01	Read Request
10	Set/Unlock Command (write a value in a specific screen, editable by the user)
11	Lock Command (write a value in a specific screen, not editable by the user)

VisualMotion Programming

Use the VisualMotion “Calculator” icon to build an integer that corresponds to the desired bit settings for parameter C-0-0814. Use the “Parameter Transfer” icon to transfer the integer to parameter C-0-0814. The integer will be converted to a 16-bit binary word and processed in this parameter.

Note: The programmer should set up the entire word before writing it to the parameter.

The user task builds this word and sets the transaction request in bits 15 and 16. The transaction is complete when the Done status (bits 15-16 = 00) is set by the BTC06.

Note: If the absolute point number is "locked" in the Jog Menu after teaching a point, the BTC06 does not automatically advance the point number. The point number is always dictated by parameter C-0-0814.

Note: Parameter C-0-0811 must be active (≠0) for parameter C-0-0814 to be functional.

Generic Cases The following table is a list of generic cases for this parameter.

Bits 16-15	Bits 14-10	Bits 9-1	Action
10	0	0	clears all locks in all menus
10	≠0	0	clears locks in a specific menu
11	0	0	locks all data indexes in all menus
11	≠0	≠0	locks data index in a specific menu
11	0	≠0	locks data index in the current menu
10	≠0	≠0	sets data index in a specific menu
10	0	≠0	sets data index in the current menu
01	0	0	requests data index from current menu the user is viewing
01	≠0	0	requests data index from a specific menu

Internal System Monitoring (C-0-0990)

C-0-0990 Exit to Monitor Prompt

This parameter halts control firmware processing and exits to a monitor prompt in order to upgrade the firmware via the serial port. To exit to the prompt, change the "NO" string value to "PROBE" or "YES" while the control is in parameter mode. The following screen will appear:



Fig. 5-8: VM DDE Server Error Message

Note: Two dots at the bottom of the H1 display are an indication that the control is in the PROBE monitor. To exit the PROBE monitor function in the control, cycle power to the control.

Most system errors that would cause an exit to the probe prompt are now reported as shutdown error "514 Control System Error Code #".

C-0-0990 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	NO
Access:	read in any phase / write in phase 2

C-0-0993 Software Reset for Control

This parameter is used to reset power to the control. Entering "YES" in this parameter will cause the control to reset (cycle power) and reinitialize after a 2-second delay. The control must first be switched to parameter mode before this parameter can be accessed.

Note: Although the PPC-R has a hardware reset button (S2), this parameter resets the control while in parameter mode. Resetting the PPC-R using the S2 button will cause an immediate reset regardless of motion. The PPC-P11.1 does not have a hardware button and can only be reset using this parameter.

C-0-0993 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	---
Minimum value:	---
Maximum value:	---
Default value:	NO
Access:	read in any Phase/ write in Phase 2

System Memory Parameters (C-0-0994 and C-0-0996)

C-0-0994 Shutdown Command for Flash Programming

This parameter is used to switch the control to download mode (DL). This mode is necessary when upgrading control or drive firmware using Dolphi software via a RS232 serial interface. Entering "YES" while in parameter mode, switches the control to download mode and displays the characters DL in the control's H1 display. Cycle power to the control to restore it to normal operating mode.

Note: The control can also be switched to DL mode by cycling power to the control while holding the S1 button until the bootloader firmware version is displayed.

C-0-0994 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	NO
Access:	read in any phase / write in phase 2

C-0-0996 Clear Program and Data Memory

When a "YES" is written to this parameter, the control's non-volatile memory will be cleared, including all parameters, programs, and data.

Note: Use VisualMotion's archive function, under menu selection **Commission** ⇒ **Archive**, before clearing the control's memory. Once program and data memory is cleared, it can no longer be retrieved.

C-0-0996 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	NO
Access:	read in any phase / write in phase 2

C-0-0997 Clear Diagnostic Log

This parameter is used to clear the diagnostic log in VisualMotion. When "YES" is written to C-0-0997, the Diagnostic log in VisualMotion Toolkit is cleared.

C-0-0997 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	NO
Access:	read in any phase / write in phase 2

System Parameter Lists (C-0-2000 through C-0-2021)

C-0-2000 List of All Parameters

This parameter contains a list of all control parameters that are part of the current firmware version. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

C-0-2000 Attributes

Data length:	variable length 4 byte data
Data type:	unsigned integer
Display format:	IDN (parameter number)
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2001 List of Required Parameters

This parameter contains a list of all required control parameters that are part of the archive / restore function. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

C-0-2001 Attributes

Data length:	variable length 4 byte data
Data type:	unsigned integer
Display format:	IDN (parameter number)
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2002 List of Invalid A-, C- and T- Parameters

This parameter displays a list of all parameters with an invalid CRC (Cyclic Redundancy Check). The maximum number of entries is fixed to 200. If more than 200 parameters are invalid, they will not be displayed. When switching to phase 4, an error is issued until all invalid parameters in the list have been corrected. When the user clears the error, another message is displayed: "009 Select Parameter Mode to Continue", indicating that the user has to switch back to P2. The list contains entries to identify the wrong parameters in the following format:

pxx: p-0-yyyy where p is the parameter type (A-, C, or T-parameters),
 xx is the drive address (1 to 40), task number (1 to 4) or
 in case of a C-parameter it is always "00"
 yyyy is the parameter number.

For example:

"A03: A-0-0005" specifies drive number 3, axis parameter A-0-0005.

C-0-2002 Attributes

Data length:	max 200 string entries
Data type:	string array
Display format:	string
Units:	--
Minimum value:	0
Maximum value:	200
Default value:	0
Access:	read-only

C-0-2010 List of SERCOS Devices

During SERCOS phase 1 initialization, the control scans the SERCOS ring for all connected SERCOS devices. These devices include all drives that are connected via the SERCOS ring and any SERCOS I/O stations. The devices found will be listed in sequential order by SERCOS address. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

C-0-2010 Attributes

Data length:	variable length 2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2011 List of SERCOS Drives

During SERCOS phase 1 initialization, the control scans the SERCOS ring for all connected SERCOS digital drives. These digital drives include any Bosch Rexroth drives that are connected via the SERCOS ring. The digital drives found will be listed in sequential order by SERCOS address. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

C-0-2011 Attributes

Data length:	variable length 2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2012 List of SERCOS I/O Stations

During SERCOS phase 1 initialization, the control scans the SERCOS ring for all connected SERCOS I/O devices. The devices found will be listed in sequential order by SERCOS address. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

C-0-2017 I/O User Configuration List

This parameter is used to store I/O devices configured using VisualMotion's I/O Configuration utility. RECO I/O modules (Local and SERCOS) stored in parameter C-0-2013 I/O Configuration List as well as any drive I/O configured using VisualMotion's I/O Configuration tool are stored in this parameter. Refer to *Chapter 6, VisualMotion I/O System*, for details.

C-0-2017 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read in any phase / write in phase 2

C-0-2020 Diagnostic Log List

VisualMotion maintains a log of the last 100 diagnostic errors. Each diagnostic log string includes the date and time that the error occurred, the shutdown error code, and any other error codes. The most recent error is listed first.

Note: The date and time is rest if no external battery is connected and the PPC-R is powered off. Refer to the *VisualMotion 9 Project Planning* manual for details.

Format:

```

11-03 12:15:47 420 1 28 0
|         |         |         |         |
|         |         |         |         |__ extended secondary error code
|         |         |         |         |__ secondary error code
|         |         |         |         |   (28 = excessive deviation)
|         |         |         |         |__ extended error code (1=drive)
|         |         |         |         |__ error code
|         |         |         |         |   (420=Drive D shutdown error)
|         |         |         |         |__ time error occurred
|         |         |         |         |   (refer to C-0-0126 for time
|         |         |         |         |   and date)
|         |         |         |         |__ date error occurred

```

The extended secondary code is the data that varies in a message, such as the drive number in "Drive D Shutdown Error." If there is a task or drive error, it is printed as a secondary error code. Error codes are not supported on all versions of drives.

Emergency Stop and warnings are normally not included in this list. To log these errors, set options in parameter C-0-2021 Diagnostic Log Options.

C-0-2020 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	current diagnostic log list
Access:	read-only

C-0-2021 Diagnostic Log Options

This parameter sets which errors should be log and displayed in parameter C-0-2020 Diagnostic Log List. The options that can be active in this parameter are list in the following 16-bit word.

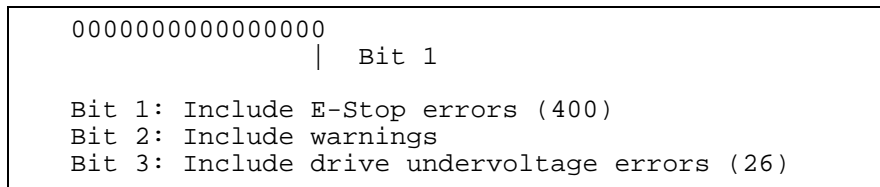


Fig. 5-9: Bit Description C-0-2021

C-0-2021 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000100 (drive undervoltage error)
Access:	read / write in any phase

C-0-2022 Probe Exception Handler

This parameter is used for internal purposes and does not provide any functionality to the user.

Oscilloscope Parameters (C-0-2501 through C-0-2523)

The oscilloscope tool is launched in VisualMotion Toolkit by selecting **Diagnostics ⇒ Oscilloscope**. The parameterization of the oscilloscope function is found in this section.

- Signal selection parameterization
- Signal timing
- Oscilloscope trigger and control
- Oscilloscope data

C-0-2501 Oscilloscope Signal 1 Type

Parameters C-0-2501, C-0-2502, C-0-2503, and C-0-2524 identify the VisualMotion data type (e.g., axis parameter) for the 4 oscilloscope signals. The following table lists the supported data types:

Value in C-0-2501 *	Data Type	Valid for	
		GPP 9	GMP 9
0	None	X	X
1	Program float variable (Fx)	X	X
2	Program integer variable (Ix)	X	X
3	Global float variable (GFx)	X	X
4	Global integer variable (GIx)	X	X
5	Axis parameter	X	X
6	Register bit	X	X
7	+/- Register	X	X
8	Control parameter	X	X
9	Task parameter	X	X

* also valid for C-0-2502, C-0-2503, and C-0-2524

Table 5-56: Oscilloscope Signal Type

C-0-2501 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	9
Default value:	0
Access:	read / write in any phase

C-0-2502 Oscilloscope Signal 2 Type

Refer to *C-0-2501 Oscilloscope Signal 1 Type* for a description of this parameter.

C-0-2503 Oscilloscope Signal 3 Type

Refer to *C-0-2501 Oscilloscope Signal 1 Type* for a description of this parameter.

C-0-2504 Oscilloscope Signal 1 ID Number

This parameter identifies the ID number for the data type written in parameter C-0-2501 Oscilloscope Signal 1 Type. The value for this parameter is written using only the parameter number's significant digits (e.g., C-0-0083 would be written as 83 and F4 would be written as 4). This description is also valid for C-0-2505, C-0-2506, and C-0-2525.

The following table contains a list of valid control parameter ID numbers when signal type 8 is written in C-0-2501.

Valid Control Parameters		Valid for	
Value	Description	GPP9	GMP9
200	Current Load Due to Motion	X	X
201	Peak load Due to Motion	X	X
202	Current Load Due to I/O	X	X
203	Peak Load Due to I/O	X	X
3204	PMG 1 Current Peak Group Deviation	X	X
3205	PMG 1 Maximum Deviation	X	X
3214	PMG 2 Current Peak Group Deviation	X	X
3215	PMG 2 Maximum Deviation	X	X
3224	PMG 3 Current Peak Group Deviation	X	X
3225	PMG 3 Maximum Deviation	X	X
3234	PMG 4 Current Peak Group Deviation	X	X
3235	PMG 4 Maximum Deviation	X	X
3244	PMG 5 Current Peak Group Deviation	X	X
3245	PMG 5 Maximum Deviation	X	X
3254	PMG 6 Current Peak Group Deviation	X	X
3255	PMG 6 Maximum Deviation	X	X
3264	PMG 7 Current Peak Group Deviation	X	X
3265	PMG 7 Maximum Deviation	X	X
3274	PMG 8 Current Peak Group Deviation	X	X
3275	PMG 8 Maximum Deviation	X	X

Table 5-57: List of Valid Control Parameters for C-0-2504

The following table contains a list of valid task parameter ID numbers when signal type 9 is written in C-0-2501.

Valid Task Parameters		Valid for	
Value	Description	GPP9	GMP9
111	Current X Position	X	X
112	Current Y Position	X	X
113	Current Z Position	X	X

Table 5-58: List of Valid Task Parameters for C-0-2504

The following table contains a list of valid axis parameter ID numbers when signal type 5 is written in C-0-2501.

Valid Axis Parameters		Valid for	
Value	Description	GPP9	GMP9
100	Target Position	X	X
101	Commanded Position	X	X
102	Feedback Position	X	X
111	Commanded Velocity	X	X
112	Feedback Velocity	X	X
120	Programmed Acceleration	X	X
141	Torque Mode Commanded Torque	X	X
142	Torque Feedback (Cyclic)	X	X
156	Phase Offset Velocity Feedback	X	X
158	Relative Phase Offset Distance Remaining	X	X
190	Optional MDT Command 1	X	X
191	Optional MDT Command 2	X	X
192	Optional MDT Command 3	X	X
195	Optional AT Feedback 1	X	X
196	Optional AT Feedback 2	X	X

Table 5-59: List of Valid Axis Parameters for C-0-2504

C-0-2504 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read / write in any phase

C-0-2505 Oscilloscope Signal 2 ID Number

This parameter identifies the ID number for the data type written in parameter C-0-2502 Oscilloscope Signal 2 Type. Refer to the tables in parameter C-0-2504 Oscilloscope Signal 1 ID Number for valid control, task, and axis parameters.

C-0-2506 Oscilloscope Signal 3 ID Number

This parameter identifies the ID number for the data type written in parameter C-0-2503 Oscilloscope Signal 3 Type. Refer to the tables in parameter C-0-2504 Oscilloscope Signal 1 ID Number for valid control, task, and axis parameters.

C-0-2507 Oscilloscope Signal 1 Axis Number

This parameter is only applicable when the signal type for C-0-2501, C-0-2502, C-0-2503, or C-0-2524 is set to 5, 6, or 9.

The following table lists the allowable values for this parameter based on supported signal type:

C-0-2501 Signal Type *	Allowable Value in 2507 **	Description
5 (axis number)	1 - 40	value equals SERCOS address for selected axis parameter in C-0-2504, C-0-2505, C-0-2506, or C-0-2525
6 (register bit)	1 -16	value equals bit number for selected register number in C-0-2504, C-0-2505, C-0-2506, or C-0-2525
9 (task)	1 = A 2 = B 3 = C 4 = D	value equals task letter for the selected task parameter in C-0-2504, C-0-2505, C-0-2506, or C-0-2525
* also valid for C-0-2502, C-0-2503, and C-0-2524		
** also valid for C-0-2508, C-0-2509, and C-0-2526		

Table 5-60: Oscilloscope Signal Axis Number

C-0-2507 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	1
Maximum value:	40
Default value:	1
Access:	read / write in any phase

Note: A value of 1 is written in this parameter when a signal type other than 5, 6, or 9 is used.

C-0-2508 Oscilloscope Signal 2 Axis Number

Refer to *C-0-2507 Oscilloscope Signal 1 Axis Number* for a description of this parameter.

C-0-2509 Oscilloscope Signal 3 Axis Number

Refer to *C-0-2507 Oscilloscope Signal 1 Axis Number* for a description of this parameter.

C-0-2510 Oscilloscope Sampling Rate

This parameter sets the sampling rate in milliseconds to determine how often a trace is captured. This parameter is used in conjunction with C-0-2514 Oscilloscope Sample Count to determine the capture duration for the entry trace. Sampling rate is set in the *Oscilloscope Options* window by selecting *Timing* from the oscilloscope main menu. Sample rates available when using the Oscilloscope are 2, 4, 8, 16, 32 and 64.

Note: The smallest sampling rate is limited by C-0-0099 Minimum SERCOS Cycle Time.

For example:

Capture duration = [(C-0-2510 = 8ms) * (C-0-2514 = 500)] = 4000 ms

C-0-2510 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	ms
Minimum value:	1
Maximum value:	64
Default value:	2
Access:	read / write in any phase

C-0-2511 Oscilloscope Signal 1 List

This parameter stores all captured data for the configured signal 1. This information is uploaded from the control after the capture is complete.

C-0-2511 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2512 Oscilloscope Signal 2 List

This parameter stores all captured data for the configured signal 2. This information is uploaded from the control after the capture is complete. Refer to *C-0-2511 Oscilloscope Signal 1 List* for the parameters attributes.

C-0-2513 Oscilloscope Signal 3 List

This parameter stores all captured data for the configured signal 3. This information is uploaded from the control after the capture is complete. Refer to *C-0-2511 Oscilloscope Signal 1 List* for the parameters attributes.

C-0-2514 Oscilloscope Sample Count

This parameter sets the quantity of data captured for each signal. This parameter is used in conjunction with C-0-2510 to determine the capture duration for the entry trace. Sample count is set in the *Oscilloscope Options* window by selecting *Timing* from the oscilloscope main menu. The available sample count when using the Oscilloscope are 100, 200, 500, 1000, 2000 and 4000.

C-0-2514 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	10
Maximum value:	4096
Default value:	10
Access:	read / write in any phase

C-0-2515 Oscilloscope Trigger Post-count

The oscilloscope utility can use an optional pretrigger value to display a percentage of the total sample count (C-0-2514 Oscilloscope Sample Count) before the actual configured signals are captured. A pretrigger can be set in the *Oscilloscope Options* window by selecting *Timing* from the oscilloscope main menu.

This parameter displays the remainder of the total sample count after the pretrigger percentage.

For example:

Trigger post count = (C-0-2514) * (1 - (pretrigger/100))

Pretrigger = 100 * (1 - (C-0-2515/C-0-2514))

C-0-2515 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	4096
Default value:	0
Access:	read / write in any phase

C-0-2516 Oscilloscope Trigger Type

When using the oscilloscope utility, a trigger can be user initiated or internally initiated. This parameter sets the signal type that will be used as the trigger enable. Refer to *C-0-2501 Oscilloscope Signal 1 Type* for available types. The signal type set in this parameter will trigger the oscilloscope after the polarity and threshold settings are true in the *Card Signal Setup* window. Select **Signal Selection** from the oscilloscope's main menu to display the *Card Signal Setup* window.

C-0-2516 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	9
Default value:	0
Access:	read / write in any phase

C-0-2517 Oscilloscope Trigger ID Number

This parameter identifies the numeric value that corresponds to the selected signal type in parameter C-0-2516 Oscilloscope Trigger Type. Refer to *C-0-2504 Oscilloscope Signal 1 ID Number* for details.

C-0-2517 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read / write in any phase

C-0-2518 Oscilloscope Trigger Axis or Mask

This parameter is only applicable when C-0-2516 is set with one of the following trigger types:

C-0-2516 Signal Type	Allowable Value in 2518	Description
5 (axis number)	1 - 40	value equals SERCOS address for selected axis parameter in C-0-2516
6 (register bit)	1 -16	value equals bit number for selected register number in C-0-2516
9 (task)	1 = A 2 = B 3 = C 4 = D	value equals task letter for the selected task parameter in C-0-2516

Table 5-61: Oscilloscope Signal Axis Number

C-0-2518 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	40
Default value:	0
Access:	read / write in any phase

Note: A value of 1 is defaulted in this parameter when a signal type other than 5, 6, or 9 is used.

C-0-2519 Oscilloscope Trigger Level or Mask

This parameter sets the level or threshold to which the oscilloscope will use its internal trigger and capture the configured signals in C-0-2516 Oscilloscope Trigger Type, C-0-2517 Oscilloscope Trigger ID Number and C-0-2518 Oscilloscope Trigger Axis or Mask. The trigger level can be a variable value, state of a register bit, float value, etc.

C-0-2519 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	--
Minimum value:	+ 0.0
Maximum value:	dependent on C-0-2516 Oscilloscope Trigger Type, C-0-2517, C-0-2518
Default value:	+0.0
Access:	read / write in any phase

C-0-2520 Oscilloscope Trigger Mode


This parameter sets the polarity or direction in which the trigger level set in parameter *C-0-2519 Oscilloscope Trigger Level or Mask* will be detected. The selections used in this parameter are:

- 1 = Positive
- 2 = Negative
- 3 = Positive or Negative

C-0-2520 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	1
Maximum value:	3
Default value:	1
Access:	read / write in any phase

C-0-2521 Oscilloscope Trigger Source

This parameter sets the trigger source that will be used by the oscilloscope utility for capturing signal traces. A trigger source can be initiated either by the user or internally. A user-initiated trigger is only active when the operator presses the enable trigger button  in the oscilloscope window. An internally initiated trigger is configured through oscilloscope parameters and active when all parameter requirements are met. The selections used in this parameter are:

- 1 = User Initiated
- 2 = Internally Initiated

C-0-2521 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	1
Maximum value:	2
Default value:	1
Access:	read / write in any phase

C-0-2522 Oscilloscope Trigger Control Word

This parameter is a 16-bit control word used to enable and trigger a capture of configured signal types. This parameter is activated for both user initiated or internally initiated captures. Configured signal data, for up to 4 signals, are captured and stored in parameters C-0-2511 through C-0-2513 and C-0-2527, respectively. This data can then be uploaded from the control to the relevant oscilloscope signal list parameter. The following figure describes the functions of bits in this parameter:

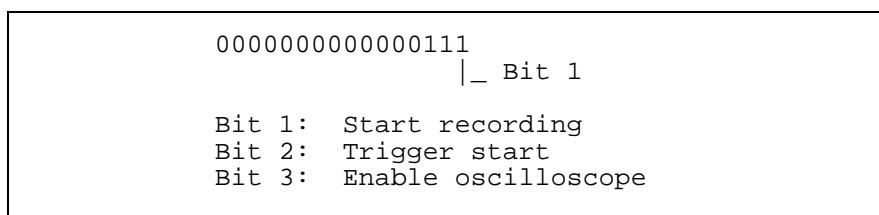


Fig. 5-10: Bit Description C-0-2522

To initiate a capture manually, bits 1-3 of this parameter are set according to the following table:

C-0-2522 Control	Description
0000000000000000	initial state of oscilloscope
0000000000000100	enable oscilloscope
0000000000000111	enable trigger and start recording
0000000000000100	oscilloscope is enabled and waiting for trigger

Table 5-62: Oscilloscope Control Word

C-0-2522 Attributes

Data length:	2 byte data
Data type:	binary word
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read / write in any phase

C-0-2523 Oscilloscope Trigger Status Word

This parameter is a 16-bit status word that displays the current process being performed by parameter *C-0-2522 Oscilloscope Trigger Control Word*. The following figure describes the functions of the bits in this parameter:

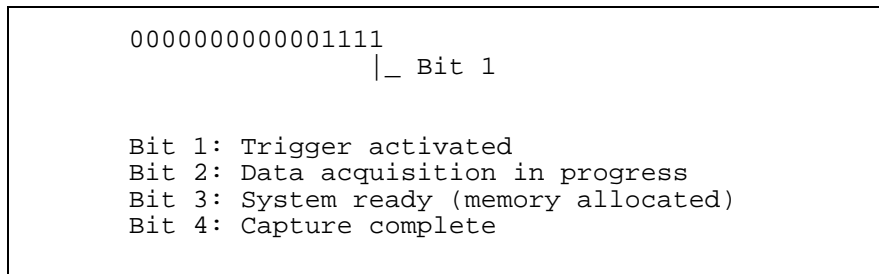


Fig. 5-11: Bit Description C-0-2523

The following table describes the status of this parameter based on the settings in C-0-2522.

C-0-2523 Status	Description
0000000000000000	initial state of oscilloscope
0000000000000100	oscilloscope is ready and waiting for trigger
0000000000000111	data acquisition in progress
0000000000001101	capture complete

Table 5-63: Oscilloscope Control and Status Words

C-0-2523 Attributes

Data length:	2 byte data
Data type:	binary word
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read-only

C-0-2524 Oscilloscope Signal 4 Type

Refer to *C-0-2501 Oscilloscope Signal 1 Type* for a description of this parameter.

C-0-2525 Oscilloscope Signal 4 ID Number

Refer to *C-0-2504 Oscilloscope Signal 1 ID Number* for a description of this parameter.

C-0-2526 Oscilloscope Signal 4 Axis Number

Refer to *C-0-2507 Oscilloscope Signal 1 Axis Number* for a description of this parameter.

C-0-2527 Oscilloscope Signal 4 List

This parameter stores all captured data for the configured signal 4. This information is uploaded from the control after the capture is complete. Refer to *C-0-2511 Oscilloscope Signal 1 List* for the parameters attributes.

Fieldbus/PLC Interface Parameters (C-0-2600 through C-0-2653)

The parameters in this section support the following interfaces:

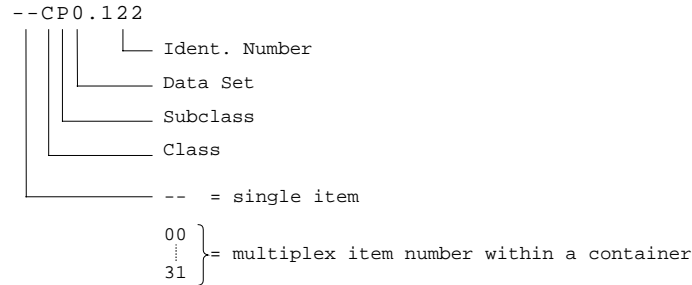
- Profibus
- DeviceNet
- ControlNet
- Ethernet/IP
- Interbus
- PLC (MTS-R)
- PCI Bus

Refer to the relevant fieldbus interface chapters for details on the specific fieldbus slave interface.

C-0-2600 Fieldbus/PLC Mapper (cyclic channel) To PLC

This parameter defines the fieldbus object-mapping list transmitted from the PPC-R to the PLC via the cyclic channel. The parameter can be configured using the Fieldbus Mapper utility in VisualMotion. The data is a series of order identifiers of VisualMotion data types.

Format:



Class	Subclass	Data Set	Ident. Number
F (float) I (integer) G (global integer) H (global float)	P	0	variable #
A (axis)	P	1-40 axis #	parameter #
T (task)	P	1-4 task #	parameter #
C (control)	P	0	parameter #
R (register)	X	0	register #

Table 5-64: Mapping List Format

C-0-2600 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of configured objects
Access:	read in any phase / write in phase 2

Note: Data from this parameter is transmitted to the PLC in the order in which it appears.

C-0-2601 Fieldbus/PLC Mapper (cyclic channel) From PLC

This parameter defines the fieldbus object-mapping list transmitted from the PLC to the PPC-R via the cyclic channel. This parameter can be configured using the Fieldbus Mapper utility in VisualMotion Toolkit. Refer to *C-0-2600 Fieldbus/PLC Mapper (cyclic channel) To PLC* for details.

C-0-2601 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of configured objects
Access:	read in any phase / write in phase 2

Note: Data from this parameter is transmitted to the PPC-R control in the order in which it appears.

C-0-2607 Multiplex Control Word

The multiplex control word is used to command the transfer of multiplex data between the PLC and the PPC-R control. Refer to the relevant fieldbus chapter under the section entitled *Information for the PLC Programmer* for multiplex control word details.

C-0-2607 Attributes

Data length:	2 bytes
Data type:	Hex display format
Display format:	Hex
Units:	--
Minimum value:	0x0000
Maximum value:	0xFFFF
Default value:	0X0000
Access:	read / write in any phase

C-0-2608 Multiplex Status Word

The multiplex status word is used to acknowledge the transfer of multiplex data between the PLC and the PPC-R control. Refer to the relevant fieldbus chapter under the section entitled *Information for the PLC Programmer* for multiplex status word details.

C-0-2608 Attributes

Data length:	2 bytes
Data type:	Hex display format
Display format:	Hex
Units:	--
Minimum value:	0x0000
Maximum value:	0xFFFF
Default value:	0X0000
Access:	read / write in any phase

C-0-2611 Fieldbus/PLC Cyclic Channel: Current Number of Misses

This parameter supports the Fieldbus and PCI Bus interfaces.

PCI Bus Support

This parameter displays the current number of missed transfers to/from the cyclic channel. Since the PLC and the control access the DPR completely unsynchronized, it is possible that the control can fail to copy data multiple times. The cyclic channel uses a handshake to ensure that the PLC or the control are not writing and reading from the DPR at the same time. When the control tries to access the DPR, it checks first to see if the handshake bits are set by the PLC. If so, the DPR is not allowed to be accessed. When 10 consecutive missed transfers occur, control parameter C-0-2613, timeout counter, is increment by 1.

PCI Bus and Fieldbus Support

If the control can not copy all the data within one cycle due to resource limitations, the counter is incremented by 1.

C-0-2611 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	10
Default value:	0
Access:	read / write in any phase

C-0-2612 Fieldbus/PLC Cyclic Channel: Peak Number of Misses

This parameter supports the Fieldbus and PCI Bus interfaces.

PCI Bus Support

This parameter displays the maximum number of missed transfers to/from the cyclic channel. Since the PLC and the control access the DPR completely unsynchronized, it is possible that the control fails multiple times to copy data. It represents the maximum number stored in control parameter C-0-2611.

PCI Bus and Fieldbus Support

If the control can not copy all the data within one cycle due to resource limitations, this counter is incremented by 1.

C-0-2612 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	10
Default value:	0
Access:	read / write in any phase

C-0-2613 Fieldbus/PLC Cyclic Channel: Timeout Counter

This parameter displays the number of timeouts in the cyclic channel. A timeout is a condition when the control was not able to copy data from/to the cyclic channel for 10 consecutive cycles. Each time that C-0-2611 counts up to 10, this parameter will be incremented by 1. The value will stay at the maximum value if another timeout occurs.

C-0-2613 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	255
Default value:	0
Access:	read / write in any phase

C-0-2630 Fieldbus Slave Device Address (GPP only)

The hexadecimal value in this parameter identifies the fieldbus slave device address for either Profibus, DeviceNet, ControlNet. The allowable range for fieldbus slave drive address varies based on the configured fieldbus device, as follows:

Fieldbus Interface	Allowable Device Address
Profibus	1-125
DeviceNet	1-63
ControlNet	1-99

Table 5-65: Fieldbus Slave Device Address

C-0-2630 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x0002
Maximum value:	0x00FF
Default value:	0x0063
Access:	read in any phase / write in phase 2

C-0-2631 Fieldbus Parameter/PCP Channel Length (GPP only)

When using a VisualMotion Profibus or Interbus fieldbus interface, a subset of the cyclic DP (Decentralized Peripheral) channel can be allocated for non-cyclic communications. This subset of the cyclic channel is called the parameter channel for Profibus and the PCP channel for Interbus. This parameter allocates the first 2 to 6 words of the cyclic DP channel as follows:

Channel Length	Profibus (Parameter Channel)	Interbus (PCP Channel)	Description
0x0000	X	X	No parameter channel
0x0004		X	allocates the first 2 words (4 bytes allocated)
0x0008	X		allocates the first 4 words (8 bytes allocated)
0x000C	X		allocates the first 6 words (12 bytes allocated)

Table 5-66: Fieldbus Parameter/PCP Channel Length

C-0-2631 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	bytes
Minimum value:	0x0000
Maximum value:	0x000C
Default value:	0x0000
Access:	read in any phase / write in phase 2

C-0-2632 Fieldbus/PLC Multiplex Method (GPP only)

This parameter sets the primary or secondary multiplex method for fieldbus interfaces. The settings in this parameter vary based on the method configured in VisualMotion Toolkit's Fieldbus utility. Refer to the section on multiplexing methods in the *VisualMotion 9 Application manual*. The settings are as follows:

- 0x0000 = primary method
- 0x0001 = secondary method

C-0-2632 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x0000
Maximum value:	0x0001
Default value:	0x0000
Access:	read in any phase / write in phase 2

C-0-2633 Fieldbus Baud Rate (DeviceNet only) (GPP only)

This parameter displays the configured baud rate in hexadecimal for the installed DeviceNet slave card. The baud rate is set using VisualMotion Toolkit's Fieldbus utility when configuring a DeviceNet slave card. The allowable baud rates are:

- 0x0001E848 = 125 KBaud
- 0x0003D090 = 250 KBaud
- 0x0007A120 = 500 KBaud

C-0-2633 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x00000000
Maximum value:	0x0007A121
Default value:	0x0001E848
Access:	read in any phase / write in phase 2

C-0-2635 Fieldbus/PLC Error Reaction

This parameter determines how the control will react to a fieldbus error. Fieldbus error reaction can be configured using the Fieldbus Mapper Utility in VisualMotion Toolkit. The valid error reaction settings are as follows:

- 0x0000 = shutdown (default)
- 0x0001 = warning
- 0x0002 = ignore

C-0-2635 Attributes

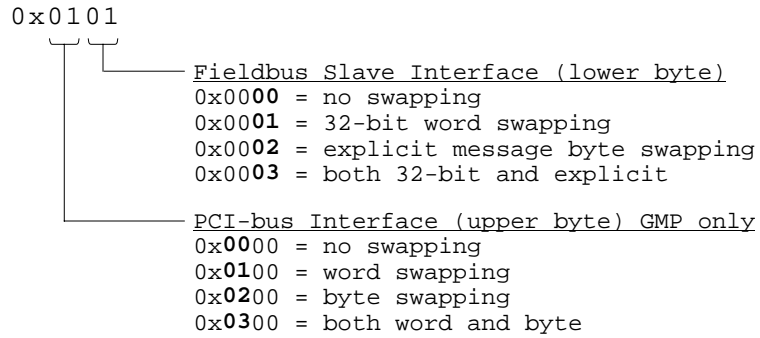
Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x0000
Maximum value:	0x0002
Default value:	0x0000
Access:	read in any phase / write in phase 2

C-0-2636 Fieldbus/PLC Word Swap

This parameter determines the order in which fieldbus slave interface or PCI-bus interface non-cyclic and/or cyclic data are transmitted between VisualMotion controls and an external PLC. The lower byte of this parameter is used for fieldbus slave interface, while the upper byte is used for the PCI-bus interface.

The following format is used:

Format:



The following table shows the different swapping types available:

Type	Word 1	Word 2
No swapping (original order of data)	byte 1 byte 2	byte 3 byte 4
32-bit word swapping for fieldbus & Word swapping for PCI-bus	byte 3 byte 4	byte 1 byte 2
Explicit message byte swapping for fieldbus & Byte swapping for PCI-bus	byte 2 byte 1	byte 4 byte 3
Both 32-bit and explicit & Both word and byte	byte 4 byte 3	byte 2 byte 1

Table 5-67: Word and Byte Swapping

32-bit Object Word Swapping (Fieldbus Slave) and Word Swapping (PCI-bus)

The setting of this option determines the order in which the two data words in any 32-bit (double word) cyclic or non-cyclic mapped object are transmitted.

Explicit Message Byte Swapping (Fieldbus Slave)

The setting of this option determines the order in which the bytes of non-cyclic data >4 bytes long are transmitted.

Byte Swapping (PCI-bus)

The setting of this option determines the order in which the bytes of any cyclic mapped object are transmitted.

C-0-2636 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x0000
Maximum value:	0x0003
Default value:	0x0000
Access:	read in any phase / write in phase 2

C-0-2637 Fieldbus/PLC Slave Firmware Version

This parameter displays the current firmware version and release date of the installed and configured fieldbus or PLC interface. If no fieldbus or PLC interface is detected, this parameter contains no value.

Format:

```
Firmware version  Date released  Driver version
V01.034           08.10.99       V04T17__24022003
```

C-0-2637 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	39 characters
Default value:	firmware version of installed fieldbus interface
Access:	read-only

C-0-2638 Fieldbus/PLC Available Cyclic IN Parameters

This parameter contains a list of allowable control, axis and task parameters that can be used as cyclic input data for parameter C-0-2600 Fieldbus/PLC Mapper (cyclic channel) To PLC.

Format:

```
C-0-0002
A-0-0020
T-0-0002
```

C-0-2638 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of supported parameters
Access:	read-only

C-0-2639 Fieldbus/PLC Available Cyclic OUT Parameters

This parameter contains a list of allowable control, axis and task parameters that can be used as cyclic output data for parameter C-0-2601 Fieldbus/PLC Mapper (cyclic channel) From PLC.

Format:

```
C-0-0002
A-0-0020
T-0-0002
```

C-0-2639 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of supported parameters
Access:	read-only

C-0-2640 PLC Connection Options (GMP only)

This parameter is used to establish communication between the PLC and PPC-P11.1 control. If set to 0, the PLC interface will not be initialized in the firmware and no communication will be established. If bit 1 (the least significant bit) is set to 1 (default) the interface will be initialized and communication will take place. Cycle power to the control for the change to take effect.

C-0-2640 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000 0000 0000 0000
Maximum value:	0000 0000 0000 0010
Default value:	0000 0000 0000 0001
Access:	read in any phase / write in phase 2

C-0-2641 PLC Input Register List

This parameter contains the configured list of registers that are placed in the register channel of the Dual Port RAM for transfer to the PLC. The register channel is used to cyclically transfer register data between the VisualMotion controls and a PLC at the current I/O Mapper cycle (2 or 4ms). The register channel is limited to 128 registers (256 bytes) in one direction (input or output).

The list can be configured using the *Parameter Overview* as follows:

1. Select **Data** ⇒ **Parameters**.
2. Double click on parameter C-0-2641 to open the parameter list edit window.
3. Right click and select **Insert Item (INS)**.
4. Enter the register number to add it to the list (e.g. 1 for reg. 1).
5. Click on **OK** to return to the *Parameter Overview* window.

C-0-2641 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	unsigned integer
Units:	--
Minimum value:	0
Maximum value:	128
Default value:	0
Access:	read in any phase / write in phase 2

C-0-2642 PLC Output Register List

This parameter contains the configured list of registers that are retrieved from the register channel of the Dual Port RAM. Refer to *C-0-2641 PLC Input Register List* for a description of the register channel.

Note: It is not advisable to list status registers in this parameter. These registers are read-only and no data will be transmitted across the DPR memory.

C-0-2642 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	unsigned integer
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of supported parameters
Access:	read in any phase / write in phase 2

C-0-2643 PLC Lifecounter Check: Number of Retries

VisualMotion PLC and controls check the life counter every SERCOS cycle in the I/O task. If the value is the same as the previous cycle, the PPC will recheck the life counter before issuing an error, if C-0-2643 is non-zero. The value is a multiple of retries.

Currently, the PLC interface checks every I/O Mapper cycle (2 or 4ms) with a retry rate fixed in firmware to 10ms, giving a life count timeout of 20-40ms.

C-0-2643 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned Integer
Units:	--
Minimum value:	0
Maximum value:	255
Default value:	40
Access:	read / write in any phase

C-0-2644 PLC Lifecounter Check: Current Number of Misses

The PLC life counter is checked every SERCOS cycle. If the value has not changed since the last SERCOS cycle, this parameter is incremented by 1. The count runs from 0 to the maximum value set in C-0-2643 (PLC life counter check: # of Retries). When the life counter changes between two consecutive SERCOS cycles, this parameter is reset to 0. This parameter is used for PCI-bus and PLC/MTS-R interfaces.

C-0-2644 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned Integer
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read / write in any phase

C-0-2645 PLC Lifecounter Check: Peak Number of Misses

This parameter monitors C-0-2644 (PLC life counter check: Current # of misses), peak count between 0 and the value set in C-0-2643 (PLC life counter check: # of Retries), and holds that value until a larger value is encountered. This parameter is only used for the MTS-R to PLC interface.

C-0-2645 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned Integer
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read / write in any phase

C-0-2646 PLC Lifecounter Check: Number of Timeouts

This counter increments by one every time C-0-2645 (PLC life counter check: Peak # of misses) encounters the number of missed life counter updates specified in C-0-2643 (PLC life counter check: # of Retries). A count incremented of one represents a PLC life counter update failure and processed by the control according to the selected error reaction specified in C-0-2635. This parameter is used for PCI-bus and PLC/MTS-R interfaces.

C-0-2646 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned Integer
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read / write in any phase

C-0-2647 ISP Function Block Timeout (GPP only)

This parameter sets the non-cyclic channel communication timeout value (in ms) in the DPRAM of the MTS-R. The non-cyclic channel is used for WinPCL C-code function blocks. This value can be increased from the default value to ensure sufficient time for non-cyclic communication when uploading (for example) a CAM list from a digital drive. If the value in this parameter is exceeded, an error (055, Maximum PLC cycle time exceeded) will appear in the MTS-R H1 7-segment display. This parameter is only used for the MTS-R to PLC interface.

C-0-2647 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned integer
Units:	ms
Minimum value:	0
Maximum value:	65535
Default value:	100
Access:	read / write in any phase

C-0-2651 PLC Register Channel: Current Number of Misses

This parameter displays the current number of missed transfers to/from the register channel. The register channel uses a handshake to ensure that either the PLC or the control are not writing and reading from the DPR at the same time. When the PPC tries to access the DPR, it checks first to see if the handshake bits are set by the PLC. If so, it is not allowed to access the DPR. Since the PLC and the control access the DPR completely unsynchronized, it is possible that the control fails multiple times to copy data. In addition, if the control firmware can not copy all the

data within one cycle due to resource limitations, the counter is increased. This parameter is used for PCI-bus and PLC/MTS-R interfaces.

C-0-2651 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	10
Default value:	0
Access:	read / write in any phase

C-0-2652 PLC Register Channel: Peak Number of Misses

This parameter displays the maximum number of missed transfers to/from the register channel. It represents the maximum number stored in control C-0-2651. The register channel uses a handshake to ensure that either the PLC or the control are not writing and reading from the DPR at the same time. When the control tries to access the DPR, it checks first to see if the handshake bits are set by the PLC. If so, it is not allowed to access the DPR. Since the PLC and the control access the DPR completely unsynchronized, it is possible that the control fails multiple times to copy data. In addition, if the control firmware can not copy all the data within one cycle due to resource limitations, the counter is increased. This parameter is used for PCI-bus and PLC/MTS-R interfaces.

C-0-2652 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	10
Default value:	0
Access:	read / write in any phase

C-0-2653 PLC Register Channel: Timeout Counter

This parameter displays the number of timeouts in the register channel. A timeout is a condition when the control was not able to copy data from/to the register channel for 10 consecutive cycles. Each time that C-0-2651 counts up to 10, this parameter will be incremented by 1. The value will stay at the maximum value if another timeout occurs. This parameter is used for PCI-bus and PLC/MTS-R interfaces.

C-0-2653 Attributes

Data length:	2 byte data
Data type:	unsigned Integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	255
Default value:	0
Access:	read / write in any phase

Option Card PLS Interface Parameters (C-0-2901 through C-0-2943)**C-0-2901 PLS1 Start Output Register**

This parameter defines the start (first) output register assigned to the Option Card PLS's output modules. Two consecutive registers are used to display the status of the Option Card PLS outputs. The bits of the displayed register number represent outputs 1-16, respectively. The bits of the next register represent optional outputs 17-32. These registers are updated every SERCOS cycle.

C-0-2901 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	1024
Default value:	70
Access:	read / write in any phase

C-0-2902 PLS1 Start Mask Register

This parameter defines the start (first) mask register assigned to the start output register in parameter C-0-2901. The next consecutive mask registers is assigned the next consecutive output register. The bits of the displayed register number mask outputs 1-16, respectively. The bits of the next register, mask outputs 17-32.

C-0-2902 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	1024
Default value:	74
Access:	read / write in any phase

C-0-2903 PLS1 Build Command

While operating in phase 4, the user can modify existing Option Card PLS parameters (for example, change settings for on / "off" positions). This parameter is used to issue a build command for the Option Card PLS table, based on the new settings. The new settings are calculated by the control and a new PLS table is generated. The new settings are not effective immediately and require an activation command from parameter C-0-2905. The status of the build command is monitored in parameter C-0-2904.

Bit 4	Bit 3	Bit 2	Bit 1	Description
0	0	0	0	No command to build PLS
0	0	1	1	Command to build PLS

Table 5-68: PLS1 Build Command

Note: This build command is not necessary if modifications to the Option Card PLS are made while in parameter mode (P2). The PLS table is calculated and activated every time a phase transition from Phase 2 to 3 is performed.

C-0-2903 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000000011
Default value:	0000000000000000
Access:	read / write in any phase

C-0-2904 PLS1 Build Status

This parameter displays the status of a build command issued in C-0-2903. The status indications are described in the following table:

Bit 4	Bit 3	Bit 2	Bit 1	Description
0	0	0	0	No command to build PLS
0	0	1	1	Command to build PLS was successful
0	1	1	1	Command to build PLS is processing.
1	1	1	1	Communication error, build command was not successful

Table 5-69: PLS1 Build Status

C-0-2904 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000001111
Default value:	0000000000000000
Access:	read-only

C-0-2905 PLS1 Activate Command

This parameter is used to activate the Option Card PLS table built with C-0-2903. If this command is issued before the build command C-0-2903 is done, an error is issued.

C-0-2905 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000000011
Default value:	0000000000000000
Access:	read / write in any phase

C-0-2906 PLS1 Activate Status

This parameter displays the status of a activate command issued in C-0-2905. If the activate command had been executed successfully, this parameter displays a binary 3. If an error is encountered, a binary 15 will be displayed. A binary 7, indicates that the control is still processing the command.

C-0-2906 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000001111
Default value:	0000000000000000
Access:	read-only

C-0-2907 PLS1 Error Code

This parameter displays an internal diagnostic error code for the Option Card PLS that is used for internal purposes.

C-0-2907 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	512
Default value:	0
Access:	read-only

C-0-2908 PLS1 Extended Error Code

This parameter displays an extended error code for the error displayed in C-0-2907 PLS1 Error Code.

C-0-2908 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	512
Default value:	0
Access:	read-only

C-0-2909 PLS1 Hardware ID

This parameter displays the hardware ID for the installed Option Card PLS. The hardware ID is displayed as a hexadecimal value. The high byte represents the hardware ID for the Option Card PLS. The lower byte represents the hardware revision for the Option Card PLS.

C-0-2909 Attributes

Data length:	2 byte data
Data type:	unsigned decimal
Display format:	hexadecimal
Units:	--
Minimum value:	0x0000
Maximum value:	0xFFFF
Default value:	0x0000
Access:	read-only

C-0-2910 PLS1 Software ID

This parameter displays the firmware version installed on the Option Card PLS. Option Card PLS firmware is included with the VisualMotion firmware.

C-0-2910 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2920 PLS1 Switch On List

This parameter displays a list of "on" positions of all 96 Option Card PLS switches. The units for the value are based on the assigned master (degrees, mm, or inch). If the switch's on and "off" position have the same value, the output will not turn on. Any modifications to "on" positions require a build (C-0-2903) and activate (C-0-2905) command.

C-0-2920 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 96 switches with a 0 value
Access:	read / write in any phase

C-0-2921 PLS1 Switch Off List

This parameter displays a list of the "off" positions of all 96 Option Card PLS switches. The units for the value are based on the assigned master (degrees, mm, or inch). Any modifications to "off" positions require a build (C-0-2903) and activate (C-0-2905) command.

C-0-2921 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 96 switches with a 0 value
Access:	read / write in any phase

C-0-2922 PLS1 Switch Output List

This parameter displays a list of the Option Card PLS's output assignments for all 96 switches. If a switch is not used, the value is 0. Otherwise, the value indicates the output number (1-32). Any modifications to the switch output list requires a build (C-0-2903) and activate (C-0-2905) command.

C-0-2922 Attributes

Data length:	variable length 2 byte data
Data type:	integer
Display format:	unsigned integer
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 96 switches with a 0 value
Access:	read / write in any phase

C-0-2930 PLS1 Output Master List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the number of the master assigned to that output. An Option Card PLS can have up to 8 masters of type ELS Master, ELS Group, and drive.

C-0-2930 Attributes

Data length:	variable length 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a 0 value
Access:	read in any phase / write in phase 2

C-0-2931 PLS1 Output Lead Time List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the lead time, in μs , assigned to each output. Lead time is the amount of time that the output is enabled prior to reaching the switch's "on" position. This value is used by the control to calculate a position based on the switch's "on" position and the current speed of the PLS master. Any modifications to the output lead time list requires a build (C-0-2903) and activate (C-0-2905) command.

The assigned lead time is always active, regardless of any additional functions, such as a one shot. Lead times are written in increments of 250 μs . The allowable range is 0 - 500000 μs (for a total of 200 ms). When using the PLS tool, the user will be prompt when entering a value that is not in increments of 250 μs .

Note: There are no individual lead times for switches. All switches assign to an output will use the output's lead time.

C-0-2931 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	μs
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a 0 value
Access:	read / write in any phase

C-0-2932 PLS1 Output Lag Time List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the lag time, in μs , assigned to each output. Lag time is the amount of time that the output is disabled prior to reaching the switch's "off" position. This value is used by the control to calculate a position based on the switch's "off" position and the current speed of the PLS master. This parameter is only available when C-0-2934 PLS1 Output Mode List is set to 0. Any modifications to the output lag time list requires a build (C-0-2903) and activate (C-0-2905) command.

The assigned lag time is always active, regardless of any additional functions, such as a one shot. Lag times are written in increments of 250 μs . The allowable range is 0 - 500000 μs (for a total of 200 ms). When using the PLS tool, the user will be prompt when entering a value that is not in increments of 250 μs .

Note: There is no individual lag times for switches. All switches assign to an output will use the output's lag time.

C-0-2932 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a 0 value
Access:	read / write in any phase

C-0-2933 PLS1 Output One Shot List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the on time (μs) assigned to that output, when configured as a one shot output. On times for one shots are written in increments of 250 μs . The allowable range is 0-1000000 μs (for a total of 1 s). When using the PLS tool, the user will be prompt when entering a value that is not in increments of 250 μs . Any modifications to the output one shot list requires a build (C-0-2903) and activate (C-0-2905) command.

Note: There are no individual one shot times for switches. All switches assign to an output will use the output's one shot time.

C-0-2933 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a 0 value
Access:	read / write in any phase

C-0-2934 PLS1 Output Mode List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the Mode assigned to that output. The mode determines which functions can be assigned to an output. Any modifications to the output mode list requires a build (C-0-2903) and activate (C-0-2905) command.

The available mode setting are:

- 0 - output lead and lag time are available.
- 1 - output lead and one shot are available, no lag time.

C-0-2934 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	unsigned integer
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a 0 value
Access:	read / write in any phase

C-0-2935 PLS1 Output Direction List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the direction in which each switch's On and "off" position will be detected for the assigned output. If the outputs are only active in one direction, the outputs will be turned off if the direction changes, independent of the previous state of the output. If the outputs are controlled via a oneshot and the direction is changed, the outputs will be on for the specified oneshot time. However, during the time where the direction has changed, but the oneshot is still on, the oneshot is not retriggerable. Any modifications to the output direction list requires a build (C-0-2903) and activate (C-0-2905) command.

The allowable direction values within the list are:

- 0 - Positive
- 1 - Negative

- 2 - Positive and Negative

C-0-2935 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	integer
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a value of 2
Access:	read / write in any phase

C-0-2936 PLS1 Output Hysteresis List

This parameter displays a list of 32 Option Card PLS outputs. The value displayed for each output represents the hysteresis assigned to that output.

C-0-2936 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 32 outputs with a value of 0
Access:	read in any phase / write in phase 2

C-0-2940 PLS1 Master Type List

This parameter displays a list of 8 Option Card PLS masters. The value displayed for each master represents the input type assigned to that PLS master. The available master input types are as follows:

- 0 – not assigned
- 1 – ELS Master
- 2 – ELS Group
- 3 – Drive (primary or secondary encoder)
- 4 - Used for internal purposes
- 5 - Used for internal purposes

C-0-2940 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	integer
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	list of 8 PLS masters with a 0 value (not configured)
Access:	read in any phase / write in phase 2

C-0-2941 PLS1 Master Number List

This parameter displays a list of 8 Option Card PLS masters. The value displayed for each master represents the number that is assigned to the corresponding input master type in C-0-2940. The available input type numbers for the master types assigned in C-0-2940 are as follows:

- 1-6 for master type 1 (ELS Master)
- 1-8 for master type 2 (ELS Group)
- 1-40 for master type 3 (Drive)

Note: The minimum, maximum and default values listed in the attributes table is the allowable range for the data contained in the list and not for the number of items (elements) contained in the list.

C-0-2941 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	unsigned integer
Units:	--
Minimum value:	0 (value of data in list)
Maximum value:	40 (value of data in list)
Default value:	0
Access:	read in any phase / write in phase 2

C-0-2942 PLS1 Master Encoder List

This parameter displays a list of 8 Option Card PLS masters. The value displayed for each master represents the encoder type for the master type (Drive) assigned in C-0-2940. The available encoder types are as follows:

- 1 – primary encoder
- 2 – secondary encoder

Note: The minimum, maximum and default values listed in the attributes table is the allowable range for the data contained in the list and not for the number of items (elements) contained in the list.

C-0-2942 Attributes

Data length:	variable length 2 byte data
Data type:	integer array
Display format:	unsigned integer
Units:	--
Minimum value:	0 (value of data in list)
Maximum value:	2 (value of data in list)
Default value:	0
Access:	read in any phase / write in phase 2

C-0-2943 PLS1 Master Phase Offset List

This parameter displays a list of 8 Option Card PLS masters. The value displayed for each master represents the relative phase offset assigned to that PLS master. The offsets can be set in any Phase and have an immediate effect. No build or activation is required.

Note: The minimum, maximum and default values listed in the attributes table is the allowable range for the data contained in the list and not for the number of items (elements) contained in the list.

C-0-2943 Attributes

Data length:	variable length 4 byte data
Data type:	float array
Display format:	float
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read / write in any phase

I/O Mapper Parameters (C-0-3000 through C-0-3005)

C-0-3000 I/O Mapper Program

This parameter contains a listing of all mapped I/O Boolean strings, timers, counters, etc. At the start of the list, the number of Boolean strings is sent to/from the control. This parameter is executed every I/O Mapper scan time (2 or 4ms). If a count of zero is sent, the mapping is deleted.

C-0-3000 Attributes

Data length:	variable length 1 byte data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	1000 entries
Default value:	0
Access:	read in any phase / write in phase 2

C-0-3001 I/O Mapper Options

This parameter selects I/O Mapper compilation options. Changing these options causes the control to recompile the object code.

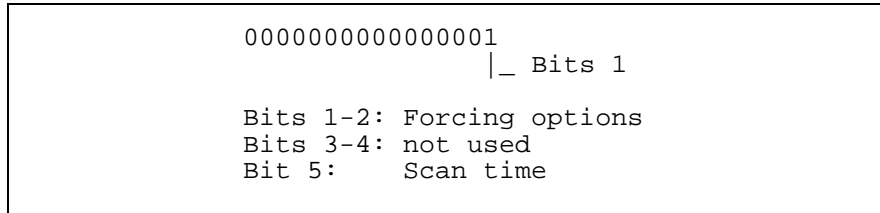


Fig. 5-12: Bit Description C-0-3001

Bits 1-2: Forcing Options

Bit 1	Bit 2	Description
0	0	Disable forcing
1	0	Force only reserved control registers (default)
1	1	Force all registers

I/O forcing allows the results of I/O Mapper equations to be ignored if forcing is enabled. Forcing can cause the I/O Mapper to execute twice as slow. As options, the control can disable forcing, enable forcing for all I/O registers, or enable forcing only for the reserved control registers. Control registers include System control register 1, Axis control register 11, etc.

Note: If the BTC06 is used, forcing of the control registers must be enabled.

Bit 5: Scan Time

The allowable I/O Mapper scan times are shown in the following table:

Bit 5	Description
0	I/O Mapper scanned every 2 ms.
1	I/O Mapper scanned every 4 ms.

C-0-3001 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000010011
Default value:	0000000000000001 (Force only reserved control registers)
Access:	read in any phase / write in phase 2

C-0-3003 I/O Mapper Total Operations

This is the total number of operations used by the I/O Mapper currently executing on the control. Each operand in a Boolean equation or each contact in a ladder diagram corresponds to one operation.

C-0-3003 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	5000
Default value:	0 (no I/O Mapper downloaded to control)
Access:	read-only

C-0-3004 I/O Mapper File Size

This is the space consumed in bytes by the I/O Mapper text strings on the control.

C-0-3004 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	bytes
Minimum value:	0
Maximum value:	131072
Default value:	0 (no I/O Mapper downloaded to control)
Access:	read-only

C-0-3005 I/O Mapper Executable Size

This is the space used in bytes by the compiled I/O Mapper currently executing on the control.

C-0-3005 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	bytes
Minimum value:	0
Maximum value:	131072
Default value:	0 (no I/O Mapper downloaded to control)
Access:	read-only

CAM Table Parameters (C-0-3100 through C-0-3140)

C-0-3100 CAM Tags

This parameter is a list of name(s) given to each configured and downloaded CAM table to the control. Each CAM name can be up to 20 characters in length. The order in which the names are displayed identifies each CAM in ascending order, starting with 1.

C-0-3100 Attributes

Data length:	variable length 1 bytes data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read / write in any phase

Note: This parameter can be modified in any Phase as long as the CAM is not active in the program.

C-0-3101 CAM Table 1 through C-0-3137 CAM Table 37

Each parameter is a list that contains all the points of a built CAM table for CAM's 1-37. CAMs are built and transferred in VisualMotion Toolkit under menu select **Tools** ⇒ **CAM Builder** or via an icon program.

C-0-3101 Attributes

Data length:	variable length 1 bytes data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read / write in any phase

Note: This parameter can be modified in any Phase as long as the CAM is not active in the program.

C-0-3138 CAM Table 38 through C-0-3140 CAM Table 40

Each parameter is a list that contains all the points of a built CAM table for CAM's 38-40. These CAMs are pre-defined for the ELS lock on / lock off feature. Refer to *Chapter 6, Electronic Line Shafting*, in the VisualMotion 9 Application Manual for details.

C-0-3138 Attributes

Data length:	variable length 1 bytes data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read / write in any phase

Note: This parameter can be modified in any Phase as long as the CAM is not active in the program.

C-0-3141 CAM Type

Bit 1 of this parameter stores the current CAM type used in the system. This parameter is written at program activation. The following table lists the CAM types supported:

Bit 1	CAM Type
0	Degree format
1	Percentage format

Table 5-70: CAM Type

Note: Writing to this parameter builds the default cams 38-40. If these cams are in use, the error message "Cannot store CAM: already active for axis 0" is displayed.

C-0-3140 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000000001
Default value:	0000000000000001
Access:	read in any phase / write in phase 2

Position Monitoring Group Parameters

The position monitoring group functionality in VisualMotion 9 allows the system to monitor the deviation between a primary slave position and up to 5 slave axes. Up to 8 groups can be commissioned, each containing 6 parameters. The following parameters are used to store the pertinent data from the Position Monitoring Group tool in VisualMotion. Refer to the *Position Monitoring Group* section under the **Commission** menu for details.

C-0-32x1 PMG # Maximum Allowed Deviation Window

The value in this parameter is scanned each time a group's enable bit (Reg. 86 bits 1-8) is set. It represents the maximum positional deviation allowed in a grouping of axes. The two methods used for positional deviation monitoring are as follows:

- *Deviation from Primary Signal* – deviation between the primary signal and slave axes.
- *Min/Max Group Deviation Window* - deviation between a minimum and maximum axes positions in a group.

The following table lists the 8 similar parameters in the PMG functionality used to store the maximum deviation window.

Parameter	Description
C-0-3201	PMG 1 Maximum Allowed Deviation Window
C-0-3211	PMG 2 Maximum Allowed Deviation Window
C-0-3221	PMG 3 Maximum Allowed Deviation Window
C-0-3231	PMG 4 Maximum Allowed Deviation Window
C-0-3241	PMG 5 Maximum Allowed Deviation Window
C-0-3251	PMG 6 Maximum Allowed Deviation Window
C-0-3261	PMG 7 Maximum Allowed Deviation Window
C-0-3271	PMG 8 Maximum Allowed Deviation Window

Table 5-71: Maximum Allowed Deviation Window Parameters

Note: Changes to the maximum deviation window, while a group is enabled, has no immediate effect. The group's control register enable bit (Reg. 86 bits 1-8) must be disabled and then set before a deviation window change can take effect.

C-0-32x1 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	+0.0
Maximum value:	+90.0
Default value:	+0.0
Access:	read / write in any phase

C-0-32x2 PMG # List of Axes

This list parameter displays the axes contained in a Position Monitoring Group. The first axis in the list is the primary slave. The following table lists the 8 similar parameters in the PMG functionality used to store the List of Axes per group.

Parameter	Description
C-0-3202	PMG 1 List of Axes
C-0-3212	PMG 2 List of Axes
C-0-3222	PMG 3 List of Axes
C-0-3232	PMG 4 List of Axes
C-0-3242	PMG 5 List of Axes
C-0-3252	PMG 6 List of Axes
C-0-3262	PMG 7 List of Axes
C-0-3272	PMG 8 List of Axes

Table 5-72: List of Axes Parameters

C-0-32x2 Attributes

Data length:	variable length 2 bytes data
Data type:	integer array
Display format:	unsigned integer
Units:	--
Minimum value:	1
Maximum value:	40
Default value:	--
Access:	read in any phase / write in phase 2

C-0-32x3 PMG # List of Position Offsets

This list parameter displays the offsets for each axis contained in the PMG group. All axes in control parameter C-0-32x2 PMG # List of Axes must be defined in this list (e.g., the item count of both lists must match). If the list is empty then the default value of 0.0 is initialized for each axis in the group. The offset value is added to the position feedback (A-0-0102) value. This combined value is then used for deviation detection. The following table lists the 8 similar parameters in the PMG functionality used to store the List of Position Offsets per group.

Parameter	Description
C-0-3203	PMG 1 List of Position Offsets
C-0-3213	PMG 2 List of Position Offsets
C-0-3223	PMG 3 List of Position Offsets
C-0-3233	PMG 4 List of Position Offsets
C-0-3243	PMG 5 List of Position Offsets
C-0-3253	PMG 6 List of Position Offsets
C-0-3263	PMG 7 List of Position Offsets
C-0-3273	PMG 8 List of Position Offsets

Table 5-73: List of Position Offsets Parameters

C-0-32x3 Attributes

Data length:	variable length 4 bytes data
Data type:	float array
Display format:	unsigned decimal
Units:	--
Minimum value:	0.0
Maximum value:	--
Default value:	0.0
Access:	read / write in any phase

C-0-32x4 PMG # Current Peak Group Deviation

This parameter displays the current peak position deviation between the furthestmost axis position in the group and the primary slave. Every SERCOS cycle, the system recalculates this value and stores it in this parameter. If a deviation error is detected, the system no longer updates this parameter and stores the maximum deviation position. The following table lists the 8 similar parameters in the PMG functionality used to store the Current Peak Group Deviation per group.

Parameter	Description
C-0-3204	PMG 1 Current Peak Group Deviation
C-0-3214	PMG 2 Current Peak Group Deviation
C-0-3224	PMG 3 Current Peak Group Deviation
C-0-3234	PMG 4 Current Peak Group Deviation
C-0-3244	PMG 5 Current Peak Group Deviation
C-0-3254	PMG 6 Current Peak Group Deviation
C-0-3264	PMG 7 Current Peak Group Deviation
C-0-3274	PMG 8 Current Peak Group Deviation

Table 5-74: Current Peak Group Deviation Parameters

C-0-32x4 Attributes

Data length:	4 byte
Data type:	float number
Display format:	signed decimal
Units:	A-0-0005 of primary slave
Minimum value:	0.0
Maximum value:	--
Default value:	current peak group deviation
Access:	read-only

C-0-32x5 PMG # Maximum Deviation

This parameter displays the peak position deviation encountered between the furthestmost axis position in a group from the primary slave after it was enabled. Every SERCOS cycle, the system checks the current value in C-0-32x4 PMG # Current Peak Group Deviation. If the value of C-0-32x4 is greater, than the new value is stored this parameter. If a deviation error is detected, the system no longer updates this parameter and stores the maximum deviation position.

Note: Writing a 0 to this parameter will reset the value and display the new current maximum deviation in the group.

The following table lists the 8 similar parameters in the PMG functionality used to store the Maximum Deviation per group.

Parameter	Description
C-0-3205	PMG 1 Maximum Deviation
C-0-3215	PMG 2 Maximum Deviation
C-0-3225	PMG 3 Maximum Deviation
C-0-3235	PMG 4 Maximum Deviation
C-0-3245	PMG 5 Maximum Deviation
C-0-3255	PMG 6 Maximum Deviation
C-0-3265	PMG 7 Maximum Deviation
C-0-3275	PMG 8 Maximum Deviation

Table 5-75: Maximum Deviation Parameters

C-0-32x5 Attributes

Data length:	4 byte
Data type:	float number
Display format:	signed decimal
Units:	A-0-0005 of primary slave
Minimum value:	0.0
Maximum value:	current maximum group deviation
Default value:	current maximum group deviation
Access:	read / write in any phase

C-0-32x6 PMG # Configuration

This parameter displays the current Position Monitoring Group configuration. This following figure describes the bit functions used in this binary parameter.

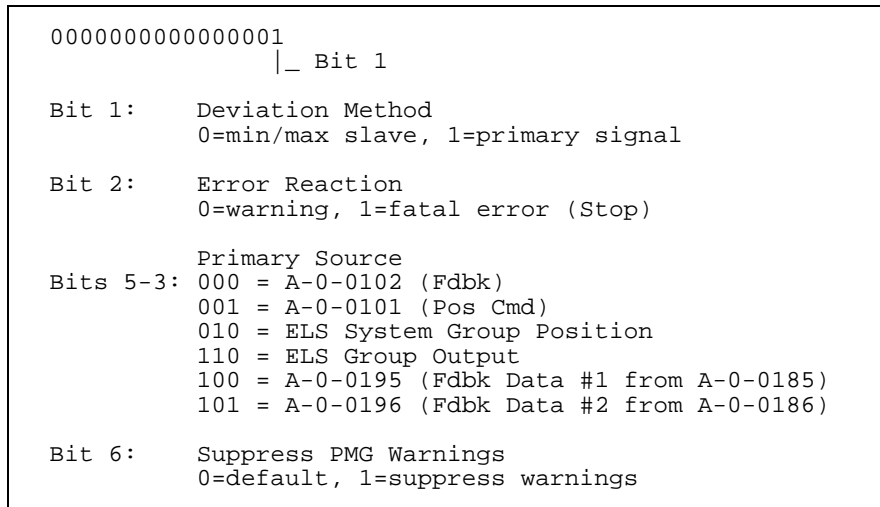


Fig. 5-13: PMG # Configuration Bit Description

The following table lists the 8 similar parameters in the PMG functionality used to store the PMG Configuration per group.

Parameter	Description
C-0-3206	PMG 1 Configuration
C-0-3216	PMG 2 Configuration
C-0-3226	PMG 3 Configuration
C-0-3236	PMG 4 Configuration
C-0-3246	PMG 5 Configuration
C-0-3256	PMG 6 Configuration
C-0-3266	PMG 7 Configuration
C-0-3276	PMG 8 Configuration

Table 5-76: PMG Configuration Parameters

C-0-32x6 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000 0000 0000 0000
Maximum value:	--
Default value:	current PMG configuration
Access:	read in any phase / write in phase 2

5.7 Task Parameters - Class T

Each task (A-D) in VisualMotion has a set of parameters that selects options and displays status information. This following section describes these parameters for tasks and coordinated motion.

Task Setup (T-0-0001 and T-0-0002)

T-0-0001 Task Motion Type

This parameter is set automatically by the user program and is set to 1 when coordinated motion is used. Until a valid program is activated, all tasks are non-coordinated by default. The following values define the task motion type as follows:

- 0 = no coordinated motion
- 1 = coordinated motion

T-0-0001 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	1
Default value:	0
Access:	read-only

T-0-0002 Task Options

This parameter specifies an automatic start option for the task and how the task will react to errors.

0000000000000001	
	_ Bit 1
Bit 1:	Run task during error
Bit 2:	Task errors do not shutdown other task
Bit 3:	Parameter transfer errors are warning
Bit 4:	Automatically start tasks
Bit 5:	Ignore task and drive errors from other tasks

Fig. 5-14: Bit Description T-0-0002

Bit 1: Run Task during Errors

The setting in this bit determines whether or not this task will run during errors. The following settings are available:

- 0 = Shutdown task on errors (default)
- 1 = Run task during errors

Refer to *Chapter 12, Error Reaction*, for details.

Bit 2: Task Errors Do Not Shutdown Other Tasks

This bit is used to determine if task errors that occur in this task should shutdown other tasks. The following settings are available:

- 0 = Shutdown other tasks on task error (default)
- 1 = Task errors do not shutdown other tasks

Refer to *Chapter 12, Error Reaction*, for details.

Bit 3: Parameter Transfer Errors are Warnings

This bit is used to determine if a parameter transfer error should be processed as an error or a warning. The following settings are available:

- 0 = Parameter transfer errors can shutdown task (default)
- 1 = Parameter transfer errors are warnings

Setting this bit to 1 allows the current task to continue running if a parameter transfer error occurs. The message "205 Parameter Transfer Warning: see Task diag." is issued. If a parameter value is critical to the operation of the task, the task error bit (bit 5 of task status register) can be tested after the parameter transfer, or set this bit to 0. Refer to task status registers 22-25 in chapter 4 for details.

Bit 4: Automatically Start Tasks

This option automatically starts tasks and keeps them running. All task control register bits are ignored. The task is switched to automatic mode and started when exiting parameter mode or when clearing an error. The task is stopped when parameter mode is selected. This option can be used for supervisory or communications tasks, or to allow the system to start at power-up without any operator intervention.

The `Override_Auto_Start` bit (bit 2 in the task control register) can be used to temporarily disable this function. Refer to task control registers 2-5 in chapter 4 for details.

Bit 5: Ignore Task and Drive Errors from Other Tasks

This bit is used to determine whether or not this task will ignore task and drive errors from other tasks. The following settings are available:

- 0 = Do not ignore task and drive errors from other tasks (default)
- 1 = Ignore task and drive errors from other tasks

Refer to *Chapter 12, Error Reaction*, for details.

T-0-0002 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	--
Default value:	0000000000000000
Access:	read in any phase / write in phase 2

Coordinated Motion (T-0-0005 through T-0-0026)

T-0-0005 World Position Units

This parameter selects the display units for position, speed, and acceleration data for coordinated motion. No unit conversions are performed when changing this parameter. The following settings are available:

- 0 = inches
- 1 = millimeters
- 2 = radians

T-0-0005 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	2
Default value:	1
Access:	read in any phase / write in phase 2

T-0-0010 Kinematic Number

The kinematic number represents a library routine that identifies the kinematic to be used for coordinated motion. Kinematic routines are application specific and unique to hardware configurations.

T-0-0010 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	17
Default value:	0
Access:	read-only

T-0-0011 Coordinated X Axis

This parameter sets the axis number corresponding to the x-axis of this task. An axis number is assigned in the **Axis** icon. If a 0 is set in this parameter, no coordinated x-axis is assigned to this task.

T-0-0011 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	40
Default value:	0
Access:	read-only

T-0-0012 Coordinated Y Axis

This parameter sets the axis number corresponding to the y-axis of this task. An axis number is assigned in the **Axis** icon. If a 0 is set in this parameter, no coordinated y-axis is assigned to this task.

T-0-0012 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	40
Default value:	0
Access:	read-only

T-0-0013 Coordinated Z Axis

This parameter sets the axis number corresponding to the z-axis of this task. An axis number is assigned in the **Axis** icon. If a 0 is set in this parameter, no coordinated z-axis is assigned to this task.

T-0-0013 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	40
Default value:	0
Access:	read-only

T-0-0020 Maximum Path Speed

This parameter sets the maximum path speed allowed for all coordinated motion axes assigned to this task. The speed entry in the absolute and relative point tables is a percentage of this maximum path speed.

The maximum path speed can also be limited by the maximum velocity parameter (A-0-0020) of each axis and each drive's bipolar velocity limit parameter (S-0-0091).

T-0-0020 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches/min or mm/min
Minimum value:	+0.001
Maximum value:	1e+07
Default value:	+1000.0
Access:	read / write in any phase

T-0-0021 Maximum Acceleration

This parameter sets the maximum acceleration allowed for all coordinated motion axes assigned to this task. The acceleration entry in the absolute and relative point tables is a percentage of this maximum acceleration.

The maximum acceleration can also be limited by the maximum acceleration parameter (A-0-0021) of each axis.

T-0-0021 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches/sec ² or mm/sec ²
Minimum value:	+0.01
Maximum value:	1e+07
Default value:	+200.0
Access:	read / write in any phase

T-0-0022 Maximum Deceleration

This parameter sets the maximum deceleration allowed for all coordinated motion axes assigned to this task. The deceleration entry in the absolute and relative point tables is a percentage of this maximum deceleration.

T-0-0022 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005 / sec ²
Minimum value:	+0.01
Maximum value:	1e+07
Default value:	+200.0
Access:	read / write in any phase

T-0-0023 Look Ahead Distance

This parameter sets the minimum look ahead distance that the control's path planner uses to calculate a path. The system will never have a smaller pre-calculated distance in the system queue than the value in this parameter until the path is completed. The Look Ahead Distance is a mechanism used for preventing the path motion instructions from reading more points from the point table.

Setting the look ahead distance to a large value can improve the overall system performance. The length of the Look Ahead Distance determines how many path segments that are to be processed by the path planner. Of course, if the parameter is set too large, the path planner may process many statements before physical motion takes place. For segments that are queued up, the path planner cannot identify potential real-time actions that may stop motion or require a program branch and the action may not happen with as fast of a response.

Decreasing the Look Ahead Distance value causes the path planner to process fewer coordinated motion program statements ahead of the current commanded position. This results in a lower potential for wasted calculations in specific cases in which the path may be aborted earlier than expected.

There is a minimum of two geometry segments that are automatically placed in the queue. Even if the Look Ahead Distance is set to 0, there will be at least the next two unprocessed segments regardless of the Look Ahead Distance value. If the segments are blended together then there is a minimum of three segments on the system queue.

In order to provide a smooth motion for a dynamic path the Look Ahead Distance should be set to the size of the smallest segment. When dynamically adding segments to the path, the decision of adding another segment must be finalized when the current position is at least two segments before the newer segments that are to be added. Blending between segments can increase the number of segments in the path planner to be processed. Generally, the Look Ahead Distance should be twice the length of the longest blend distance. If all blend distances are zero, the look ahead distance should be equal to the shortest geometry segment.

T-0-0023 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005
Minimum value:	+0.0
Maximum value:	+10000.0
Default value:	+10.0
Access:	read / write in any phase

T-0-0024 Velocity Override

The velocity override provides a method to equally slow all motion in a task. When a velocity override factor is specified for coordinated motion, all velocities in the point table are multiplied by this factor as they are used. When a velocity override is specified for non-coordinated motion that is generated by the digital drive, each velocity command is multiplied by this factor before the command is executed.

T-0-0024 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percentage
Minimum value:	+0.099
Maximum value:	+100.0
Default value:	+100.0
Access:	read / write in any phase

T-0-0025 Maximum Jog Increment

This parameter defines the maximum distance that is used for incremental coordinated jogging. C-0-0042 World Large Increment and C-0-0043 World Small Increment percent parameters are based on this value.

T-0-0025 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005
Minimum value:	+0.0001
Maximum value:	+1000.0
Default value:	+1.0
Access:	read / write in any phase

T-0-0026 Maximum Jog Velocity

This parameter defines the maximum velocity used for coordinated jogging. The world fast (C-0-0045) and world slow (C-0-0046) percent parameters are based on this value.

T-0-0026 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005 / min
Minimum value:	+0.0001
Maximum value:	+100000.0
Default value:	+100.0
Access:	read / write in any phase

T-0-0027 Path Smoothing Filter Constant

This parameter defines the filter constant used to smooth the path profile with coordinated moves and coordinated jogging. A percentage of this filter constant value can be adjusted in the point table under the % Jerk column for user programs. This percent value will be read from the point table only at the end of a geometry segment that is not blended when the path velocity is at zero at the time the Visual Motion icon program executes a coordinated motion move icon. If this is not the case, the previously read percent value will remain active. When the control is jogging it will use this value for the path profile.

T-0-0027 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	ms
Minimum value:	+0.0
Maximum value:	+200.0
Default value:	+40.0
Access:	read / write in any phase

T-0-0050 Kinematic Value 1 through T-0-0059 Kinematic Value 10

Each segment in a robotic arm is represented as a length using parameters T-0-0050 through T-0-0059. These parameters are a coefficient used in the kinematic equation of coordinated motion. Kinematics is unique to each application. Refer to *Chapter 10, Coordinated Motion*, for details.

T-0-0050 through T-0-0059 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	--
Minimum value:	+0.0
Maximum value:	--
Default value:	dependent on Kinematic
Access:	read in any phase / write in phase 2

Coordinated Motion Status (T-0-0100 through T-0-0113)

Coordinated motion statuses are read-only dynamically updated parameters that provide status values for each task.

T-0-0100 Target Point Number

This parameter displays the current target point.

For example:

If 10 is displayed here, motion to point ABS [10] or REL [10] is taking place, or the machine is currently at this point.

**CAUTION**

The point displayed in T-0-0100 is the target point set in the path planner but necessarily the actual position. Depending on value in parameter T-0-0023 Look Ahead Distance, the actual position of the drive might well be at the previous segment moving towards the next.

T-0-0100 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	--
Default value:	current target point in path planner
Access:	read-only

T-0-0101 Segment Status

This parameter displays the status of the current segment. The segment status codes are listed in the table below. The codes are valid for the current segment, except for codes 0 and 1 that are transitional or do not apply to the current segment. Use code 6 to check if motion is halted due to a path/stop. The following codes are available:

Code	Segment Status	Activity
0	Segment ready	Path Planner is currently not running
1	Acceleration	Acceleration in progress
2	Constant velocity	Constant velocity in progress (at target velocity)
3	Blending	Blending in progress
4	Target deceleration	Deceleration to target position in progress
5	Controlled stop	Controlled stop taking place (error, jog or path/stop)
6	Stopped	Motion has stopped (error, jog or path/stop)

T-0-0101 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	6
Default value:	dynamically updated by control
Access:	read-only

T-0-0111 Current X Position

This parameter displays the current commanded position, in world coordinates, for the x-axis set in T-0-0011.

T-0-0111 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

T-0-0112 Current Y Position

This parameter displays the current commanded position, in world coordinates, for the y-axis set in T-0-0012.

T-0-0112 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

T-0-0113 Current Z Position

This parameter displays the current commanded position, in world coordinates, for the z-axis set in T-0-0013.

T-0-0013 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	T-0-0005
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

Task Status (T-0-0120 through T-0-0200)

T-0-0120 Task Operating Mode

This parameter displays the current operating mode of the task. Information about the task's state, etc., is available in the control and status I/O registers. The following settings are available:

- 0 = initialization
- 1 = parameter
- 2 = manual
- 3 = automatic

T-0-0120 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	read by control
Access:	read-only

T-0-0122 Task Diagnostic Message

This parameter displays the current diagnostic message for the task. During normal operation, a **Msg1** icon in the user program sets this message. If an error occurs during task execution, this diagnostic message is overwritten with an error message.

T-0-0122 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0123 Task Status Message

This parameter displays the current status message for this task. A **Msg1** icon in the user program sets this message and aids the operator when debugging. This message is not overwritten with an error message, allowing debugging of an error condition set in the Task Diagnostic Message. The following strings are available:

- 0 = initialization
- 1 = parameter
- 2 = manual
- 3 = automatic

T-0-0123 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0130 Current Instruction Pointer

This parameter returns a hexadecimal value equal to the current task's execution address (i.e. the instruction pointer). The hex value is an offset from the start of the program.

For example:

"0x000000F0" indicates that the program counter is at 0xF0, or 240 bytes from the start of the program.

T-0-0130 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x00000000
Maximum value:	0xFFFFFFFF
Default value:	read by control
Access:	read-only

T-0-0131 Current Instruction

This parameter displays the current instruction pointer and mnemonic of the current instruction. The mnemonic is in the base code format generated by the control's compiler. This parameter is used for debugging and troubleshooting programs.

Format :

```
0028 START
|         |_____ current mnemonic
|_____  current instruction pointer
```


T-0-0131 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0132 Instruction Pointer at Error

This status parameter displays the instruction pointer where the last task error occurred. This parameter is used for debugging and troubleshooting programs.

T-0-0132 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x00000000
Maximum value:	0xFFFFFFFF
Default value:	read by control
Access:	read-only

T-0-0133 Composite Instruction Pointer

This parameter dynamically displays a flag and a current instruction pointer indicating the relative memory address where a program instruction is executed. This parameter is used for debugging and troubleshooting programs.

The first number (flag) identifies where the instruction resides in the program as well as how many pointers identify the relative memory address where the instruction is being executed. The numbers, in hexadecimal format that follow the flag are the relative memory address where the instruction is being executed.

Flag (First Number)	Description
1	Instruction is located in the main task
2 to 11	Instruction is located in a subroutine
(-2) to (-11)	Instruction is in an event

T-0-0133 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0135 Current Subroutine

This parameter indicates the current subroutine being executed with the function number and name. This includes tasks, non-accessible functions, functions, and subroutines. If function number and name information is not included in the user program file, the string "NONE" is returned.

T-0-0135 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0136 Stack Variable Data

This parameter displays the current stack variable data for function arguments and local variables defined in the **Start** icon. Stack variables are valid only while the program flow is within a task, subroutine, or event function. The maximum number of stack variables is 16. If there are no defined arguments or local variables in a task, subroutine, or event function, this parameter displays "NONE".

T-0-0136 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0137 Subroutine Breakpoint

This task parameter specifies the index number of the subroutine to which program execution will halt when task breakpoint is enabled. To enable Task breakpoint set bit 11 of the Task_Control register to (1). Task program flow continues with a (0-1) transition of bit 6 (Cycle_Start) in the Task_Control register.

T-0-0137 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	65535
Default value:	0
Access:	read / write in any phase

T-0-0138 Sequencer Information

This task parameter displays information about the currently running sequencer in both index and name format. The index indicates the current row in the sequence list or step list. If no sequencer is running, this parameter displays "NONE".

T-0-0138 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	NONE
Access:	read-only

T-0-0200 Last Active Event Number

This parameter displays the index of the current or last active event in the event (EVT) table. The value can be used to access other information (message, status, function, etc.) contained in the event table.

T-0-0200 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read-only

Task Parameter Lists (T-0-2000 and T-0-2001)

T-0-2000 List of All Parameters

This parameter contains a list of all task parameters that are part of the current firmware version.

T-0-2000 Attributes

Data length:	variable length 4 byte data
Data type:	unsigned integer
Display format:	IDN (parameter number)
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

T-0-2001 List of Required Parameters

This parameter contains a list of all required task parameters that are part of the archive / restore function.

T-0-2001 Attributes

Data length:	variable length 4 byte data
Data type:	unsigned integer
Display format:	IDN (parameter number)
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

5.8 Axis Parameters – Class A

Axis parameters are used to configure the axis and provide limits for coordinated motion. Some parameters only apply to a specific axis mode (coordinated or single-axis).

Axis Setup (A-0-0001 through A-0-0038)

A-0-0001 Task Assignment

This parameter associates an axis with a user task. The **Axis2** and **ELSGroup1** icons automatically set the parameter when a program is activated. The following settings are available:

0 = No task selected

1 = Task A

2 = Task B

3 = Task C

4 = Task D

A-0-0001 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	4
Default value:	0
Access:	read-only

A-0-0002 Type of Positioning

This parameter defines the type of positioning for an axis as either normal positioning or lagless positioning. This is relevant for an axis configured as single-axis, coordinated, ratio, ELS phase synch, control CAM or drive CAM. Lagless positioning helps reduce the following error during the acceleration or deceleration of an axis. The following settings are available:

0 = Normal positioning

1 = Lagless positioning

A-0-0002 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	1
Default value:	0
Access:	read in any phase / write in phase 2

A-0-0003 Axis Motion Type

This parameter defines the operating mode of the axis and configures data in the AT and MDT SERCOS telegrams. The **Axis2** and **ELSGroup1** icons are used to define the operating mode.

Motion Type	Description
0 = Disabled	no active motion type
1 = Single Axis	The axis moves independently of other axis using a target position with defined velocity and acceleration / deceleration ramps.
2 = Coordinated Axis	The path planner in the control calculates the position of axis to produce a coordinated move using predefined ABS or REL point tables and kinematic.
3 = Velocity Mode	This motion type operates the axis at constant velocity without a position control loop. The control sends only velocity commands to the digital drive. The digital drive maintains the velocity profile.
4 = Ratio Slave	This motion type designates the axis as a slave to the master axis designated in Parameter A-0-0030 Ratio Mode Master Axis.
5 = ELS Slave	This motion type designates the axis as an Electronic Line Shafting (ELS) slave.
6 = Torque Mode	This motion type designates the axis the run in torque mode, with torque commands sent from the control and no velocity or position loop.
7 = Control CAM Axis	This motion type designates the axis to run from a CAM table stored on the control and uses an ELS Virtual or Real Master.
8 = Torque Following Mode	This motion type designates the axis to run with **-torque following mode.
9 = Coordinated Articulation	This motion type designates the axis as a control CAM with coordinate transformation.

Table 5-77: Axis Motion Type

A-0-0003 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	9
Default value:	0
Access:	read-only

A-0-0004 Axis Options

This parameters sets all the necessary options for a Bosch Rexroth digital drive.

0000000000000001	_ Bit 1
Bit 1:	Position initialization
Bit 2:	Positioning mode
Bit 3&4:	ELS synchronization mode
Bit 5:	Enable Measuring Wheel Services
Bit 6:	Optional cyclic data
Bit 7:	Not used
Bit 8:	Drive PLS Fast Write (DKC only)
Bit 9:	ELS secondary mode
Bit 10:	Disable ELS shortest path
Bit 11:	Positioning using secondary encoder
Bit 12:	Drive disable method
Bit 13:	Linear axis modulo positioning
Bit 14:	Configure minimum cyclic data
Bit 15:	Disable modulo positioning for rotary axis
Bit 16:	Disable automatic scaling

Fig. 5-15: Bit Description A-0-0004

Bit 1: Position Initialization

This option is used for non-absolute measurement systems, such as single-turn or incremental encoder. During phase 2 initialization, the feedback position value can remain unchanged (bit 1 = 0) or be initialized with the "Starting Position Value" in P-0-0019 (bit 1 = 1).

0 = Keep feedback value in Phase 2 (default)

1 = Reset feedback in Phase 2

Keep feedback value in Phase 2

When set to 0, a different starting position value can be entered through the SERCOS service channel or the feedback value will be reset to the default one-revolution measurement.

Reset Feedback in Phase 2

When set to 1, the control overwrites the feedback position A-0-0102 with the value in P-0-0019 Starting Position Value after the system is switch out of parameter mode.

Bit 2: Positioning Mode

0 = Linear Positioning Mode (default)

1 = Rotary Positioning Mode

The positioning mode is valid for all axis types. All relevant scaling parameters are automatically set in the drive. If an axis is an ELS slave, its mode is automatically set to rotary positioning. This parameter can be modified in a VisualMotion program using the **PrmBit** icon.

When rotary positioning mode is enabled in the drive, position is in degrees, velocity in RPM, and acceleration in radians/sec. Single-axis and velocity mode values are entered in these units. Events for single-axis and velocity mode work with both rotary and linear positioning.

Coordinated motion and events are compatible with rotary positioning only if motion takes place within the modulo value and does not rollover. If an axis is coordinated, velocity is in linear units/sec and acceleration in units/sec.

Linear Positioning Mode

When set to 0, linear positioning is selected. The units and scaling are specified with parameter A-0-0005 Linear Position Units. Absolute positioning is enabled in the drive if bit 13 (Linear Axis Modulo Positioning) is set 0. Setting bit 13 to 1 enables linear axis using modulo position.

Rotary Positioning Mode When set to 1, rotary positioning is enabled. The position, velocity, and acceleration units and scaling are fixed at the drive. Modulo positioning is enabled by default, with a rollover value specified in drive parameter S-0-0103. Setting bit 15 (Disable Modulo Positioning for Rotary Axis) to 1 disables modulo positioning and sets absolute positioning.

Bits 4 and 3: ELS Synchronization Mode

This option sets the type of synchronization for an electronic line shafting axis. These bits are set automatically at program activation by the ELS/INIT instruction.

Bit 4	Bit 3	Mode
0	0	Velocity Synchronization. The axis runs in velocity mode, with its velocity equal to the master's velocity times the translation ratio. The ratio can be fine adjusted at run time.
0	1	Phase Synchronization. The axis runs in the operating mode selected in parameter A-0-0002 Type of Positioning, and maintains a phase relationship according to the phase offset and translation ratio parameters. The phase offset can be adjusted at run time.
1	1	CAM Table. The axis is linked to a CAM and synchronized to a master. Position relationship is maintained according to the CAM table, ratio and stretch factors. Position offset can be adjusted at run time.

Bit 5: Enable Measuring Wheel Services

When set to 1, bit 5 enables measuring wheel services. The AT is configured as follows:

- S-0-0386 (active feedback) is written to the AT
- S-0-0051 (feedback 1) and S-0-0052 (feedback 2) are not written to the AT

Drive parameter S-0-0386 provides the control with an active position feedback value. When using the measuring wheel services (bit 5), all feedback position values (S-0-0386) are used by the control for all internal functions (jog mode, stop icon, etc.), instead of S-0-0051 or S-0-0053.

Note: When using Measuring Wheel Services (Bit 5=1), Bit 11 should be set to 0. Measuring wheel services is not possible together with Positioning Using Secondary Encoder (Bit 11=1).

Parameter S-0-0386 is supported by the following drive firmware:
DIAX04 (ELS05V(>31)) and ECODRIVE03 (SGP03V22 and SGP20V11)

Bit 6: Optional Cyclic Data

0 = Use smallest cyclic data configuration (default)

1 = Include optional cyclic data (velocity feedback, programmed acceleration)

Use Smallest Cyclic Data Configuration

When set to 0, the smallest amount of cyclic data is used to increase the number of drives on the ring and increase cycle time. Velocity feedback is normally read through the service channel, which can take up to 50ms using a program command.

Include Optional Cyclic Data

To include velocity feedback in the cyclic data for all axis modes, set this bit to 1. In single-axis mode, programmed acceleration is also sent cyclically instead of through the service channel.

Up to five optional, IDNs can be specified with parameters A-0-0180 Optional Command ID #1 through A-0-0186 Optional Feedback ID #2 in addition to or exclusive of this option bit.

Bit 7: DCP Configuration

0 = Drive PLS Fast Write disabled (default)

1 = Drive PLS Fast Write enabled

When set to 1, the Drive PLS fast write is enabled. Enabling this feature

Bit 8: Drive PLS Fast Write (DKC2.3 only)

0 = Drive PLS Fast Write disabled (default)

1 = Drive PLS Fast Write enabled

When set to 1, the Drive PLS fast write is enabled. Enabling this feature will force the SERCOS multiplex channel on. The PLS fast write applies when using the Calc Drive PLS icon.

Bit 9: ELS Secondary Mode

0 = Secondary mode is single axis mode (default)

1 = Secondary mode is velocity mode

For an ELS axis, this bit selects the secondary, non-synchronized mode for a digital drive. This is the default mode until the **ELSmode** icon switches the axis into ELS velocity or phase synchronous mode.

Secondary Mode is Single Axis Mode

When set to 0, the default secondary mode is set to single axis positioning mode. Velocity mode may also be selected when using the **ELSmode** icon, if it is configured in the cyclic data using parameters A-0-0180 Optional Command ID #1 to A-0-0196.

Secondary Mode is Velocity Mode

When set to 1, the secondary mode is set to velocity mode. On drives that don't support single axis mode, the secondary mode is always velocity, regardless of this bit.

Bit 10: Disable ELS Shortest Path

0 = Shortest path positioning for ELS phase adjust (default)

1 = Shortest path disabled

This bit selects the positioning method used for the ELS phase adjust move.

Shortest Path Positioning for ELS Phase Adjust

When set to 0, the axis takes the shortest path within a revolution. If the difference in position is between 180 and 360 degrees, the axis travels counterclockwise.

Shortest Path Disabled

When set to 1, shortest path is disabled. A move within one revolution travels in the positive direction if the programmed position is positive, and in the negative if, it is negative.

Bit 11: Positioning Using Secondary Encoder

0 = Use primary feedback (encoder 1) (default)

1 = Use secondary feedback (encoder 2)

This bit configures the digital drive to use Encoder 2 to close the position loop and provide cyclic feedback from drive parameter S-0-0053. This option must be set if the drive is using linear motor firmware.

Bit 12: Drive Disable Method

0 = Stop axis immediately (default)

1 = Coast to a stop

This bit changes the response of the drive after a fatal error or when the disable bit in the axis control register is set. It configures the way the control sets the enable bits (14 and 15) in the Master control word S-0-0134.

Stop Axis Immediately

When set to 0, the drive immediately commands the motor to zero velocity before disabling torque and applying the brake. This option should be used for coordinated motion or linear motion, where coasting can cause damage or injury.

Coast to a Stop

When set to 1, the drive immediately disables torque, causing the motor to coast, stopping with its own inertia. This should be used in some types of line shafting applications, where immediate disabling of the drive could cause damage.

Bit 13: Linear Axis Modulo Positioning

0 = Linear axis uses absolute positioning (default)

1 = Linear axis uses modulo positioning

Linear Axis Uses Absolute Positioning

When set to 0, absolute motion is enabled, with signed positions and no modulo. This option should be used for coordinated motion and most absolute positioning applications.

Linear Axis Uses modulo Positioning

When set to 1, the modulo value in S-0-0103 is used. When the axis position reaches the modulo value, it resets to 0. There are no negative position values. Unlike rotary mode, the scaling of position, velocity, and acceleration is linear (inches or mm). This option should be used for continuous indexing operations.

Bit 14: Configure Minimum Cyclic Data

0 = Default or maximum cyclic data (default)

1 = Minimum cyclic data

ELS or CAM Axes**Default or Maximum Cyclic Data**

When set to 0, parameter S-0-0036 (velocity command value) is included in the cyclic data, which allows real-time update of velocity when the axis is in its secondary mode (A-0-0004, bit 9=1).

Minimum Cyclic Data

When set to 1, parameter S-0-0036 (velocity command value) is removed from the cyclic data. Now that the velocity feedback value is read through the service channel only, no ramping or real-time control can be done on the axis. Therefore, the axis should only be run in ELS synchronization mode.

Single-axis mode**Default or Maximum Cyclic Data**

When set to 0, parameter S-0-0259 (position velocity value) is included in the cyclic data for applications where the velocity is changed often in the user program.

Minimum Cyclic Data When set to 1, parameter S-0-0259 (position velocity value) is removed from the cyclic data to allow the maximum number of drives and options and the minimum SERCOS cycle time. The velocity is then read through the service channel. This option is not recommended in applications where the velocity is changed often in the user program.

Velocity mode

Minimum Cyclic Data When set to 1, parameter S-0-0182 Manufacturer Class 3 Diagnostics is removed from the AT. This is required when using a REFU drive, since they do not support S-0-0182.

Ratio Axis Master

Default or Maximum Cyclic Data When set to 0, parameter S-0-0040 (velocity feedback value) is included in the cyclic data for applications where the velocity is changed often in the user program.

Minimum Cyclic Data When set to 1, parameter S-0-0040 (velocity feedback value) is removed from the cyclic data.

Bit 15: Disable Modulo Positioning for Rotary Axis

0 = Absolute positioning (default)

1 = Modulo positioning

Absolute Positioning When set to 0 (default), the rotary axis position resets to the modulo value for every revolution.

Modulo Positioning When set to 1, modulo positioning is disabled for a rotary axis. Positions less than zero are negative, and the modulo parameter is not used. This should be used only with positioning applications, not with indexing, ELS, or continuous motion.

Bit 16: Disable Automatic Scaling

0 = Control sets SERCOS scaling parameters (RECOMMENDED) (default)

1 = Control does not set SERCOS scaling parameters

For most drives, the control sets SERCOS scaling parameters such as S-0-0044 (velocity data scaling type) and S-0-0076 (position data scaling type). If a drive does not accept the control's parameter settings, these parameters can be set manually.

Note: This bit should only be set at the recommendation of the Bosch Rexroth applications department.

A-0-0004 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	displays current axis options
Access:	read in any phase / write in phase 2

A-0-0005 Linear Position Units

This parameter sets the units of measurement that will be used for linear axis positioning, speed, and acceleration data. No unit conversions are performed when changing this parameter to a different unit of measure. The following units are available:

- 0 = inches
- 1 = millimeters (default)
- 2 = degrees

The drive's display and scaling parameters are automatically set by the control. All data is transmitted in decimal format with the same resolution as the data transmitted by the drive. Velocity, acceleration, and jerk data are always transmitted with an accuracy of three-decimal places. The following decimal places are used for position data:

Units	Decimal Places	Maximum Value
inches	5	21474.83648
millimeters	4	214748.3648
radians	6	2147.483648

Table 5-78: Decimal Accuracy for A-0-0005

If an axis is in rotary mode, the drive automatically sets the position to degrees, the velocity to RPM, and the acceleration to rads/sec/sec. In rotary mode, the scaling and display units in this parameter do not apply.

Note: Changing this parameter affects the scaling of drive parameters. If you need to retain such constants, you must save each required drive mechanical parameter before modifying this parameter.

A-0-0005 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	inches, millimeters or degrees
Minimum value:	0
Maximum value:	2
Default value:	1
Access:	read in any phase / write in phase 2

A-0-0006 Reference Options

This parameter selects options for reference position monitoring and homing.

0000000000000001	_ Bit 1
Bit 1:	Issue error when drive is enabled
Bit 16-2:	Reserved for future development

Fig. 5-16: Bit Description A-0-0006

Bit 1: Issue Error when Drive is Enabled

When set to 1, the control immediately issues the error "500 Axis D is not referenced" if the drive is enabled with no referenced position. The control reads drive parameter S-0-0403, Position Feedback Status, to determine if the position is referenced.

While the drive is disabled, the Set Absolute Measurement command (P-0-0012) is used to set the reference position. This option should only be set when an absolute encoder is used. It prevents an incremental homing procedure from being used.

A-0-0006 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read in any phase / write in phase 2

A-0-0007 Configuration Mode

This parameter allows drives to be excluded from the user program and initialized to single-axis or velocity mode. An error will not be issued if the drive is not found on the SERCOS ring. The following settings are available:

Setting	Description
0 (default)	the axis can be used in the program with its defined axis type and its presence on the ring will be verified
1	the drive is excluded from the program, but can be jogged using default values
2	the drive is not configured and is put into a torque-free mode after its initialized

Table 5-79: Settings for A-0-0007

A-0-0007 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	2
Default value:	0
Access:	read in any phase / write in phase 2

A-0-0009 Drive PLS Register

This parameter displays the register assigned to the Drive PLS. The register selected here can be read by the user program, the I/O Mapper, or the user interface as a status of the current PLS outputs.

A-0-0009 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	512
Default value:	0
Access:	read / write in any phase

A-0-0020 Maximum Velocity

This parameter sets the maximum programmable velocity for this axis in coordinated, single-axis or velocity mode.

If a coordinated axis is commanded to a value greater than this parameter, the path planner will scale the velocity of this and all coordinated axes to produce a valid coordinated move.

If a single-axis or velocity mode axis is commanded to a value greater than this parameter, error *470 Axis # velocity > maximum* is issued and the axis is stopped based on its parameterized error reaction.

A-0-0020 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min
Minimum value:	+0.001
Maximum value:	1e+07
Default value:	+1000.0
Access:	read / write in any phase

Note: The drive's bipolar velocity limit parameter (S-0-0091) is the maximum velocity allowed for the motor connected to the drive. Parameter A-0-0020 should always be set to a value less than S-0-0091.

A-0-0021 Maximum Acceleration

This parameter sets the maximum programmable acceleration for this axis in coordinated, single-axis or velocity mode.

If a coordinated axis is commanded to a value greater than this parameter, the path planner will scale the acceleration of this and all coordinated axes to produce a valid coordinated move. This parameter is also used as the maximum acceleration when jogging a coordinated axis.

For single-axis or velocity mode, this parameter is only used for jogging and sets the maximum acceleration.

A-0-0021 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / sec ²
Minimum value:	+0.01
Maximum value:	1e+07
Default value:	+200.0
Access:	read / write in any phase

A-0-0022 Maximum Deceleration

This parameter sets the maximum programmable deceleration for a coordinated axis. It is not used for single-axis or velocity mode.

If a coordinated axis is commanded to a value greater than this parameter, the path planner will scale the deceleration of this and all coordinated axes to produce a valid coordinated move.

This parameter is also used as the maximum deceleration when jogging a coordinated axis.

A-0-0022 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / sec ²
Minimum value:	+0.01
Maximum value:	1e+07
Default value:	+200.0
Access:	read / write in any phase

A-0-0023 Jog Acceleration

This parameter sets the acceleration and deceleration rate for axis jogging in single-axis or velocity mode. The value entered is a percentage of the maximum acceleration (A-0-0021 Maximum Acceleration).

A-0-0023 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	percentage
Minimum value:	1
Maximum value:	100
Default value:	100
Access:	read / write in any phase

A-0-0025 Maximum Jog Increment

This parameter sets the maximum distance used for incremental single-axis jogging. Control parameters C-0-0042 World Large Increment and C-0-0043 World Small Increment are percentages that are multiplied with the value in this parameter to produce an incremental jogging distance.

A-0-0025 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	+0.0001
Maximum value:	+1000.0
Default value:	+1.0
Access:	read / write in any phase

A-0-0026 Maximum Jog Velocity

This parameter sets the maximum velocity used for jogging an axis in single-axis mode, velocity mode, or as a joint. Control parameters C-0-0055 Axis Fast Jog Velocity and C-0-0056 Axis Slow Jog Velocity are percentages that are multiplied with the value in this parameter to produce a jogging velocity.

A-0-0026 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min
Minimum value:	+0.0001
Maximum value:	+100000.0
Default value:	+100.0
Access:	read / write in any phase

A-0-0030 Ratio Mode Master Axis

This parameter defines the axis that will be used as the master for ratio mode. For ELS programs, setting this parameter to 0 assigns ELS Group 1 as the master.

A-0-0030 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	40
Default value:	0
Access:	read-only

A-0-0031 Control CAM/Ratio Master Factor (N)

The slave-to-master ratio is defined using two parameters, A-0-0031 and A-0-0032 Control CAM/Ratio Slave Factor (M). This allows the control to normalize the ratio calculation, preserving full system accuracy for repeating-decimal ratios such as 2/3. Both parameters must be set or a run-time error may occur.

In ratio mode, the velocity of the slave is determined by:

$$V_{\text{slave}} = V_{\text{master}} * (K_{\text{slave}} / K_{\text{master}})$$

Where:

V_{slave} = Velocity of the slave axis

V_{master} = Velocity of the master axis

K_{slave} = Slave factor set by Parameter A-0-0032

K_{master} = Master factor set by Parameter A-0-0031

A-0-0031 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	--
Minimum value:	-100000.0
Maximum value:	+100000.0
Default value:	+1.0
Access:	read / write in any phase

A-0-0032 Control CAM/Ratio Slave Factor (M)

Refer to parameter *A-0-0031 Control CAM/Ratio Master Factor (N)* for a description of this parameter.

A-0-0033 Control CAM Stretch Factor (H)

This is the stretch factor (H) for CAM profiles on the control. Every position at the output of the CAM is multiplied by this value.

A-0-0033 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	--
Minimum value:	-100000.0
Maximum value:	+100000.0
Default value:	+1.0
Access:	read / write in any phase

A-0-0034 Control CAM Currently Active

This is the CAM number that is currently active for this axis. If the CAM is set to 0, the axis directly follows the master axis. CAM activation only takes affect after the master has passed zero degrees or when the master is stopped.

A-0-0034 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	40
Default value:	1
Access:	read / write in any phase

A-0-0035 Control CAM Position Constant (L)

The value in this parameter is multiplied to the master position according to the following equation.

$$Scmd = CAM + L * Mcmd$$

A-0-0035 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	--
Minimum value:	-100000.0
Maximum value:	+100000.0
Default value:	+1.0
Access:	read / write in any phase

A-0-0036 Ratio Mode Encoder Type

This parameter sets the type of master used for ratio mode. The primary encoder or secondary encoder can be used. The following settings are available:

Setting	Description
0 = Not used or primary feedback is used	selects the master drive's primary feedback, which is the value read from drive parameter S-0-0051
1 = Primary feedback is used	selects the master drive's primary feedback, which is the value read from drive parameter S-0-0051
2 = Secondary feedback is used	selects the secondary feedback, read cyclically from drive parameter S-0-0053 (feedback 2)

Table 5-80: Ratio Mode Encoder Type

A-0-0036 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	2
Default value:	0
Access:	read-only

A-0-0037 Ratio Mode Step Rate

This parameter sets the rate used in control ratio mode when the ratio parameters A-0-0031 Control CAM/Ratio Master Factor (N) A-0-0032 Control CAM/Ratio Slave Factor (M) are changed, either directly or through the **Axis** icon. If the step rate is set to 0, the ratio will be changed immediately without a ramp.

For example:

The current ratio is 0, the programmed ratio is 10:1, and the step rate parameter is set to 10 units/sec. The ratio will be ramped for one second until it reaches the target value.

A-0-0037 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	units/sec
Minimum value:	+0.0
Maximum value:	1e+07
Default value:	+0.0
Access:	read / write in any phase

A-0-0038 Ratio Mode Options

In a ratio mode configuration, a master's signal (position or velocity) is evaluated by VisualMotion, taking into account any gear ratio calculations, and a resultant signal is sent to the ratioed slave axis. Any axis in the system can be used as a ratioed axis. ELS Virtual Masters cannot be used in the ratioed axis configuration.

Note: The master number, type and master-to-slave ratio are initially setup in the **Axis** icon. The master-to-slave ratio can be modified in the user program by using the **Ratio** icon. Ratioed slave axes can be switched off a master by writing 0 turns for the slave in the Ratio icon.

Bit 1 (Control Mode) of axis parameter A-0-0038 sets up the drive's mode of operation for a ratioed slave axis. Bit 2 (Feedback Mode) determines which master signal will be sent to VisualMotion. The values of both bits are evaluated by VisualMotion every SERCOS cycle to determine the type of signal (position or velocity command) that will be sent to the ratioed slave axis every SERCOS cycle.

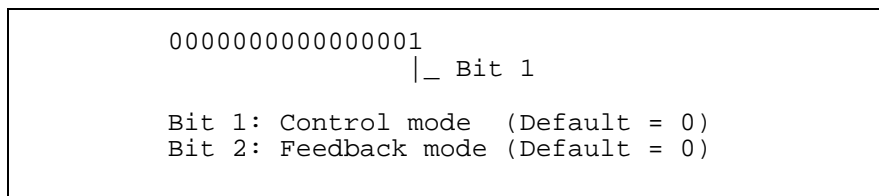


Fig. 5-17: Bit Description A-0-0038

Bit 1: Control Mode

The Control Mode bit is used to setup the drive's mode of operation (position or velocity) for a ratioed slave axis. The following bit states are defined for bit 1:

- 0 = Sets the ratioed slave axis in Position Mode (default)
- 1 = Sets the ratioed slave axis in Velocity Mode

Position Mode (Bit 1=0)

The drive is operating in a closed position loop, with lag or lagless control selected in parameter A-0-0002 Type of Positioning. Master commands sent to the ratioed slave axis by VisualMotion will be in position. When the slave axis is switched into ratio mode, it maintains a relative position to the master. A constant phase difference is maintained between the master and the ratioed slave axis.

Note: Axis parameter A-0-0004 Axis Options bit 11 configures the digital drive's primary or secondary feedback device to close the position loop and provide cyclic feedback from drive parameter S-0-0053.

Velocity Mode (Bit 1=1)

The drive is operating in velocity mode. Master commands sent to the ratioed slave axis by VisualMotion will be in velocity. When the slave axis is switched to ratio mode, it follows the velocity of the master. Any position offsets that occur when the slave axis is switched to ratio mode may not be maintained, since no position loop is being closed.

This option is useful for applications where the velocity of the slave needs to be adjusted. The slave velocity may be adjusted in response to a tension loop by changing drive parameter S-0-0037, *Additive Velocity Command Value*.

Bit 2: Feedback Mode

The Feedback Mode bit is used to determine which master signal (position or velocity) is evaluated by VisualMotion before it's sent to the ratioed slave axis. The following bit states are defined for bit 2:

0 = The master's position is evaluated by VisualMotion (default)

1 = The master's velocity is evaluated by VisualMotion

Follow Position (Bit 2=0)

The master's position is evaluated by VisualMotion, including any master-to-slave ratio calculations. The resultant is sent to the ratioed slave axis.

If the slave's Control Mode (bit 1) is set to 0 (position mode), the ratioed slave axis follows the master's resultant position.

Note: Bits 1 and 2 of axis parameter A-0-0038 should be set to 0 when relative positioning between the master and the slave is critical.

If the slave's Control Mode (bit 1) is set to 1 (velocity mode), VisualMotion calculates a velocity for the ratioed slave axis to follow based on the master's resultant position.

Note: Velocity following is not as accurate as position following due to conversion errors and drive implementation.

Follow Velocity Bit (2=1)

The master's velocity is evaluated by VisualMotion, including any master-to-slave ratio calculations. The resultant is sent to the ratioed slave axis.

If the slave's Control Mode (bit 1) is set to 0 (position mode), VisualMotion calculates a position for the ratioed slave axis to follow based on the master's resultant velocity.

If the slave's Control Mode (bit 1) is set to 1 (velocity mode), the ratioed slave axis follows the master's resultant velocity.

Note: These options should be selected when problems with position initialization are experienced, or when relative accuracy of position is not required.

A-0-0038 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000000
Access:	read in any phase / write in phase 2

Axis Status (A-0-0100 through A-0-0145)

Parameters A-0-0100 through A-0-0145 provide status values for each configured axis. The values from these parameters are both generated from a running VisualMotion program or in respond, from the drive, to a program command.

Note: Feedback values are obtained from the drive through the cyclic SERCOS telegram rather than through the service channel.

A-0-0100 Target Position

The value in this parameter is the programmed position used by the drive in single-axis mode. Target positions are generated every time a **Move** icon is encountered in a running VisualMotion program. VisualMotion writes the programmed target position to this parameter and to drive parameter S-0-0258. Drive parameter S-0-0258 is then used by the drive to move the motor to the target position.

A-0-0100 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 setting
Minimum value:	-100000.0
Maximum value:	+100000.0
Default value:	generated from running program
Access:	read / write in any phase

A-0-0101 Commanded Position

The value in this parameter is the commanded position used by the drive in coordinated or ratio mode. Commanded positions are generated in a running VisualMotion program. VisualMotion updates the commanded position in this parameter and in drive parameter S-0-0047. Drive parameter S-0-0047 is then used by the drive to move the motor to the commanded position.

A-0-0101 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 setting
Minimum value:	--
Maximum value:	--
Default value:	generated from running program
Access:	read-only

A-0-0102 Feedback Position

This parameter displays the current feedback position value from the drive. The value in this parameter is written to this parameter and drive parameter S-0-0051 every SERCOS cycle.

A-0-0102 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 setting
Minimum value:	--
Maximum value:	--
Default value:	current feedback value of configured axis
Access:	read-only

A-0-0110 Programmed Velocity

The value in this parameter is the programmed velocity used by the drive in single-axis mode. Programmed velocities are generated every time a **Velocity** icon is encountered in a running VisualMotion program. VisualMotion writes the programmed velocity in this parameter and in drive parameter S-0-0259. Drive parameter S-0-0259 is then used by the drive to accelerate the motor at the programmed velocity.

A-0-0110 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min
Minimum value:	+0.0
Maximum value:	1e+07
Default value:	generated from running program
Access:	read / write in any phase

A-0-0111 Commanded Velocity

The value in this parameter is the commanded velocity used by the drive in velocity mode. Commanded velocities are generated in a running VisualMotion program. VisualMotion updates the commanded velocity in this parameter and in drive parameter S-0-0036. Drive parameter S-0-0036 is then used by the drive to accelerate the motor at the commanded velocity.

A-0-0111 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min
Minimum value:	--
Maximum value:	--
Default value:	generated from running program
Access:	read-only

A-0-0112 Feedback Velocity

This parameter displays the current feedback velocity value from the drive. The value in this parameter is written to this parameter and drive parameter S-0-0040 every SERCOS cycle.

A-0-0112 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min
Minimum value:	--
Maximum value:	--
Default value:	generated from running program
Access:	read-only

Note: If A-0-0112 is not configured in the cyclic telegram, the control reads this value from the SERCOS service channel.

A-0-0120 Programmed Acceleration

The value in this parameter is the programmed acceleration used by the drive in single-axis or velocity mode. Programmed accelerations are generated every time an **Accel** icon is encountered in a running VisualMotion program. VisualMotion writes the programmed acceleration in this parameter and in drive parameter S-0-0260. Drive parameter S-0-0260 is then used by the drive to accelerate the motor to the programmed acceleration.

A-0-0120 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / sec ²
Minimum value:	+0.0
Maximum value:	1e+07
Default value:	generated from running program
Access:	read / write in any phase

A-0-0131 SERCOS Control Word

The SERCOS control word is automatically configured based on the compiled user program. The SERCOS control word is read from parameter S-0-0134 (Master Control Word) and is transmitted cyclically to the control. Refer to S-0-0134 in the relevant *Digital Drive Functional Description* for details.

A-0-0131 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	generated by control
Access:	read-only

A-0-0132 SERCOS Status Word

The SERCOS status word is read from parameter S-0-0135 (Drive Status Word) and is transmitted cyclically to the control. Refer to S-0-0135 in the relevant *Digital Drive Functional Description* for details.

A-0-0132 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	generated by control
Access:	read-only

A-0-0133 AT Error Count

The AT (drive telegram) error counter is used for troubleshooting of SERCOS connections. If the value of this parameter is increasing while being displayed, there may be a noisy SERCOS connection or a faulty communication interface on the drive associated with this axis. If two consecutive ATs are invalid, the control issues error 409.

A-0-0133 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read-only

A-0-0140 Mfg. Class 3 Status Word

The function of this parameter will vary based on the connected drive. The following Bosch Rexroth digital drives indicate the supported SERCOS parameter used for Manufacturer class 3 diagnostics:

- DIAX04 and ECODRIVE03 (supports S-0-0182)
- IndraDrive (supports S-0-0144)

The following table shows when SERCOS parameter S-0-0182 or S-0-0144 is automatically placed in the AT:

Operating Mode	Condition
Single-axis	always placed in the AT
Velocity	when A-0-0004, bit 14 = 0
ELS	when A-0-0164, bit 3 = 0
Ratio	when A-0-0164, bit 3 = 0
Control CAM	when A-0-0004, bit 9 = 0 and A-0-0164, bit 3 = 0
Coordinated Articulation	when A-0-0164, bit 3 = 0

Table 5-81: Auto Placement of S-0-0182 or S-0-0140 in to the AT

DIAX04 and ECODRIVE03

For DIAX04 and ECODRIVE, this parameter is the cyclic equivalent of parameter S-0-0182. The control sets the axis status register based on the settings of these bits. Refer to S-0-0182 in the relevant *Digital Drive Functional Description* for details.

IndraDrive

IndraDrive digital drives do not support parameter S-0-0182. For this reason, parameter S-0-0144 will be placed in the AT. However, only four bits will be configured in A-0-0140. The following table shows how the relevant bits in S-0-0144 are configured in A-0-0140:

S-0-0144	A-0-0140	Description
Bit 12	Bit 1	feedback velocity < S-0-0124
Bit 13	Bit 6	IZP (position reached)
Bit 14	Bit 8	In_Sych
Bit 15	Bit 11	drive halt and feedback velocity < S-0-0124

Table 5-82: S-0-0144 to A-0-0140 Configuration

Bits 0-11 of S-0-0144 are available for configuration by the user.

Note: If S-0-0144 is configured and placed in the AT by the user, only the four bits mentioned above will be configured.

Refer to S-0-0144 in the relevant *Digital Drive Functional Description* for details.

A-0-0140 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	generated by control
Access:	read-only

A-0-0141 Torque Mode Commanded Torque

This is the cyclic equivalent of drive parameter S-0-0080, Torque Command. This parameter is set from the user program via a parameter transfer. Refer to S-0-0080 in the relevant *Digital Drive Functional Description* for details.

A-0-0141 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percentage
Minimum value:	--
Maximum value:	--
Default value:	+0.0
Access:	read / write in any phase

A-0-0142 Torque Feedback (cyclic)

This is the cyclic equivalent of drive parameter S-0-0084, Torque Feedback. This parameter is updated only when an axis is in Torque Mode. Otherwise, the drive's service channel is used. Refer to S-0-0084 in the relevant *Digital Drive Functional Description* for details.

A-0-0142 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percentage
Minimum value:	--
Maximum value:	--
Default value:	+0.0
Access:	read-only

A-0-0145 Current Motion Type

This parameter displays the current motion type corresponding to the drive's operating mode. This motion type can be the drive's primary or secondary mode of operation. The following motion type are available:

- 0 = disabled
- 1 = single axis
- 2 = coordinated axis
- 3 = velocity mode
- 4 = ratio slave
- 5 = ELS slave
- 6 = torque mode / torque following
- 7 = control CAM axis

A-0-0145 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	7
Default value:	0 (axis selected in program)
Access:	read-only

Electronic Line Shafting (A-0-0150 through A-0-0164)

ELS parameters A-0-0150 through A-0-0164 are written to by an active ELS VisualMotion user program. The various icons that are used are identified in each parameter.

A-0-0150 Programmed Ratio Adjust

This parameter contains the ratio fine adjust used to adjust the velocity or phase of an ELS configured axis to compensate for mechanical variations between master and slave axes. This value corresponds to drive parameter P-0-0083. It is adjusted from -100 to 300 percent using the **ELSAAdj1** icon. If fine adjust is included in the cyclic data, it is updated every SERCOS cycle and may be adjusted using a ramp (A-0-0159 Ratio Adjust Step Rate). Otherwise, the drive's service channel is used.

A-0-0150 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percentage
Minimum value:	-327.0
Maximum value:	+327.0
Default value:	+0.0
Access:	read / write in any phase

A-0-0151 Programmed Phase Offset

This parameter contains the programmed phase offset value used in position synchronization and drive CAM modes. The drive executes a position profile with acceleration (P-0-0142) and velocity limit (P-0-0143) when this parameter is changed. This value corresponds to drive parameter S-0-0048, in the cyclic data, set through the **ELSA_{adj1}** or **CAM_{adj}** icon.

A-0-0151 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	--
Maximum value:	--
Default value:	+0.0
Access:	read / write in any phase

Note: When dynamic synchronization is enabled in the drive, parameters A-0-0153 and A-0-0155 are not used.

A-0-0153 Control Phase Adjust Average Velocity

This parameter sets the average velocity of the phase offset move. When a phase offset is changed, the control performs an absolute positioning move in addition to the ELS master command. This is the target velocity used for the phase offset adjustment move. The time constant parameter A-0-0155 Control Phase Adjust Time Constant can be used to automatically set the velocity for the phase offset based on the time of the move.

Note: Peak velocity will be up to twice as large as this value.

A-0-0153 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min or RPM
Minimum value:	+0.0
Maximum value:	1e+06
Default value:	+0.0
Access:	read / write in any phase

A-0-0155 Control Phase Adjust Time Constant

The control uses a filter to implement a jerk limited profile for the phase adjust. This parameter sets the amount of time that the move will require, regardless of the position. This can be calculated in the user program so that the phase adjust is always distributed for the length of one part.

For example:

If parameter A-0-0155 is set to 0, the control uses parameter A-0-0153 Control Phase Adjust Average Velocity, to set the average velocity of the phase offset move.

Note: Peak velocity will be up to twice as large as this value.

A-0-0155 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	seconds
Minimum value:	+0.0
Maximum value:	+1000.0
Default value:	+0.0
Access:	read / write in any phase

A-0-0156 Phase Offset Velocity Feedback

This parameter displays the feedback velocity of the phase offset. The axis must be configured as ELS phase or ELS CAM slave in the ELSGrp1 icon.

A-0-0156 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 / min or RPM
Minimum value:	--
Maximum value:	--
Default value:	+0.0
Access:	read-only

A-0-0157 Current Phase/ Control CAM Master Offset

This parameter displays the current cyclic phase offset command sent from the control to the drive.

A-0-0157 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	--
Maximum value:	--
Default value:	read from control
Access:	read / write in any phase

A-0-0158 Relative Phase Offset Distance Remaining

This parameter displays the distance remaining on the relative phase offset. When a continuous phase offset is in progress, this parameter displays +/- 0.4 * Modulo based on the direction.

A-0-0158 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	--
Maximum value:	--
Default value:	+0.0
Access:	read-only

A-0-0159 Ratio Adjust Step Rate

This parameter sets the rate used when the ELS programmed ratio adjust, A-0-0150 Programmed Ratio Adjust, is changed.

If the step rate is set to 0, the ratio adjust will be changed immediately without a ramp.

If ratio adjust is not included in the cyclic data for the drive, the value is not ramped. Refer to *A-0-0004 Axis Options* and *A-0-0180 Optional Command ID #1* for data configuration.

For example:

The current ratio adjust is 0%, the programmed adjust is 10%, and the step rate parameter is set to 10 % / sec. The ratio adjust will be ramped for one second until it reaches the target value.

A-0-0159 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percentage per second
Minimum value:	+0.0
Maximum value:	+10000.0
Default value:	+0.0
Access:	read / write in any phase

A-0-0160 Commanded Ratio Adjust

This parameter sets the currently commanded value for ratio adjust, which is updated either gradually or immediately to the programmed ratio adjust value, A-0-0150 Programmed Ratio Adjust.

A-0-0160 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percentage
Minimum value:	-327.0
Maximum value:	+327.0
Default value:	+0.00
Access:	read / write in any phase

A-0-0161 Control CAM Programmed Slave Adjust

This parameter sets the target value for the slave phase adjust (**Sph** in the CAM equation), which is set using the CAMAdj icon.

A-0-0161 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	--
Maximum value:	--
Default value:	read from active program
Access:	read / write in any phase

A-0-0162 Control CAM Current Slave Adjust (Sph)

This is the currently commanded value of the slave phase adjust (**Sph** in CAM equation).

A-0-0162 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	--
Maximum value:	--
Default value:	read from active program
Access:	read / write in any phase

A-0-0163 Control CAM Output Position

This parameter displays the slave position, which is the output of the CAM equation for this axis. It provides a status of the CAM position even when the axis is not synchronized to the CAM. An initial move or phase offset can be performed with the value read from this parameter before switching into CAM synchronization mode.

A-0-0163 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005
Minimum value:	--
Maximum value:	--
Default value:	read from active program
Access:	read / write in any phase

A-0-0164 ELS Options

This parameter sets several options for ELS or CAM motion. It's also used to remove certain parameters from the cyclic data. For dynamic synchronization and ramp, set bits 1 and 6.

<pre> 0000000000000001 _ Bit 1 Bit 1: Use drive internal phase offset Bit 2: Remove S-0-0048 "Position command value additional" from cyclic data Bit 3: Remove S-0-0182 "Mfg Class 3 Diag" from cyclic data Bit 4: Remove S-0-0258 "Target Position" from cyclic data Bit 5: Use service channel for Drive Cam control/status Bit 6: Do not automatically set phase offset Bit 7: Enable Electronic Pattern Control </pre>

Fig. 5-18: Bit Description A-0-0164

Bit 1: Use drive internal phase offset

0 = Phase offset profile is generated by the control

1 = Phase offset profile is generated by the drive (default)

On Bosch Rexroth digital drives, the phase offset for ELS and drive-based CAMs can be generated using parameters P-0-0142, P-0-0143, P-0-0153, and P-0-0155.

Enabling the phase offset in the drive frees control resources and automatically places parameter S-0-0182 into the cyclic data. Bit 4 (Phase_Adjust) of the relevant axis status register is set to 1 when bit 8 of S-0-0182 is set to 1. Refer to S-0-0182 in the relevant *Digital Drive Functional Description* for details.

Bit 2: Remove S-0-0048 "Position command value additional" (phase offset) from cyclic data

0 = S-0-0048 is transmitted via the MDT (default)

1 = S-0-0048 is removed from the MDT and transmitted via the service channel

By default, S-0-0048 is placed in the MDT of the cyclic data for ELS slave axes using phase synchronization. If the phase offset is never changed while the user program is running, cycle time in the SERCOS ring can be conserved by eliminating this parameter from the cyclic data. When the phase offset is sent through the service channel, the **ELSA_{adj1}** or **CAM_{adj}** icon will take up to 50ms to execute. If the control is generating the phase offset profile, the value will change instantly.

Bit 3: Remove S-0-0182 "Mfg Class 3 Diag" from cyclic data

0 = S-0-0182 is transmitted via the AT (default)

1 = S-0-0182 is removed from the AT

Note: Removing S-0-0182 from the cyclic data has an adverse affect on single-axis or velocity mode when using the non-coordinated abort in the **Stop** icon. Since axis status is no longer available, the control can not set the target position equal to the current feedback position upon a non-coordinated abort.

Bit 4: Remove S-0-0258 "Target position" from cyclic data

0 = S-0-0258 is transmitted via the MDT (default)

1 = S-0-0258 is removed from the MDT

Bit 5: Use service channel for Drive CAM control/status

0 = Drive CAM control and status uses drive real time bit (default)

1 = Drive CAM control and status is through service channel

In most applications, this bit should be set to 0. The CAM can then be changed in the following SERCOS cycle through the drive real time bits. The drive has two real time bits that are used for CAMs and probe functions. If the CAM does not need to be changed on the fly and more than one probe is needed for registration, this parameter bit can be set.

Bit 6: Do not automatically set phase offset

0 = Sets phase offset automatically at synchronization and stores value in A-0-0157 (default)

1 = Phase offset can be initialized with programmed offset (A-0-0151 Programmed Phase Offset) before synchronization.

If this bit is set to 1, the phase offset can be initialized to any value before the drive is switched into ELS mode. If the bit is set to 0, the control automatically establishes relative phase synchronization.

Bit 7: Enable Electronic Pattern Control

If this bit is set to 1, the primary operating mode for axes configured as Drive CAMs is enable to Electronic Pattern Control. This is only available for DIAX04 drives using ELS05VRS firmware.

A-0-0164 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000001
Access:	read in any phase / write in phase 2

Axis Feedback Capture (Registration) (A-0-0170 through A-0-0174)

A-0-0170 Probe Configuration Status

This parameter displays the status of the drive feedback capture setup in VisualMotion at program activation or execution. The control uses the SERCOS probe functions and real-time bits along with the event system to allow user programs to perform registration functions. Bosch Rexroth digital drives provide two probe inputs that can be used for capturing the feedback position.

```

0000000000000001
                    |_ Bit 1

Bit 1: Probe 1 positive edge enabled
Bit 2: Probe 1 negative edge enabled
Bit 3: Probe 2 positive edge enabled
Bit 4: Probe 2 negative edge enabled
    
```

Fig. 5-19: Bit Description A-0-0170

A-0-0170 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0000000000000001
Access:	read-only

A-0-0171 Probe 1 Positive Captured Value

This parameter displays the last captured value for the probe 1 positive edge (0 ⇒1) input on the drive. Upon either a positive or a negative transition of a probe input, the drive captures the position into the cyclic data. Since the captured feedback positions must be included in the SERCOS cyclic data telegram, the probe setup icon must be included in the user program for each drive that will use the probe function.

A-0-0171 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0.0000
Access:	read-only

A-0-0172 Probe 1 Negative Captured Value

Refer to *A-0-0171 Probe 1 Positive Captured Value* for a description of this parameter.

A-0-0173 Probe 2 Positive Captured Value

Refer to *A-0-0171 Probe 1 Positive Captured Value* for a description of this parameter.

A-0-0174 Probe 2 Negative Captured Value

Refer to *A-0-0171 Probe 1 Positive Captured Value* for a description of this parameter.

Optional SERCOS Data (A-0-0180 through A-0-0196)**A-0-0180 Optional Command ID #1**

VisualMotion automatically configures the MDT (S-0-0024, Configuration List of the Master Data Telegram) based on the mode of operation specified in the user program and the axis parameter settings during system initialization (SERCOS Phase 2 to phase 3).

In addition to the configured MDT, this parameter allows the user to add an additional drive parameter from S-0-0188 to the MDT by entering the parameter IDN number.

For example:

S-0-0040 is entered as **40**

P-0-0053 is entered as 53 + 32768 or **32821**

Note: Since the MDT is configured based on the mode of operation and axis parameters, there might not be enough room in the telegram for an additional drive parameter. If this is the case, the control will issue an error.

Parameter A-0-0180 works in conjunction with parameter A-0-0190 Command Data #1. A-0-0180 identifies which drive parameter is being added to the MDT and A-0-0190 stores the value for that parameter.

For example:

To update the torque limit in real-time, set **A-0-0180** to 92 (Torque Limit). While the drive is in Phase 4, the value in Parameter **A-0-0190** is sent cyclically to the drive, and can be written using the **Param** icon in the user program.

A-0-0180 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read in any phase / write in phase 2

A-0-0181 Optional Command ID #2

Refer to *A-0-0180 Optional Command ID #1* for a description of this parameter.

A-0-0182 Optional Command ID #3

Refer to *A-0-0180 Optional Command ID #1* for a description of this parameter.

A-0-0185 Optional Feedback ID #1

VisualMotion automatically configures the AT (S-0-0016, Custom Amplifier Telegram Configuration List) based on the mode of operation specified in the user program and the axis parameter settings during system initialization (SERCOS Phase 2 to phase 3).

In addition to the configured AT, this parameter allows the user to add an additional drive parameter from S-0-0187 to the AT by entering the parameter IDN number.

For example:

S-0-0040 is entered as **40**

P-0-0052 is entered as 52 + 32768 or **32820**

Note: Since the AT is configured based on the mode of operation and axis parameters, there might not be enough room in the telegram for an additional drive parameter. If this is the case, the control will issue an error.

Parameter A-0-0185 works in conjunction with parameter A-0-0195 Feedback Data #1. A-0-0185 identifies which drive parameter is being added to the AT and A-0-0195 displays the current value for that parameter.

For example:

By default, feedback velocity is received through the service channel. To obtain the feedback velocity in real-time, set A-0-0185 to 40 (Feedback Velocity). While the drive is in Phase 4, the value in Parameter A-0-0195 is updated, and can be read using the **Param** icon in the user program.

A-0-0185 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read in any phase / write in phase 2

A-0-0186 Optional Feedback ID #2

Refer to *A-0-0185 Optional Feedback ID #1* for a description of this parameter.

A-0-0190 Command Data #1

This parameter displays the real-time value that corresponds to the parameter identified in A-0-0180 Optional Command ID #1. Changes to this parameter will affect the value of the drive parameter set in A-0-0180.

Note: When using the *Parameter Transfer* icon in VisualMotion to transfer a value for the parameter stored in A-0-0180, be aware of the attribute for that parameter. If the parameter identified in A-0-0180 is of type float, transfer a float value. For any other types (integer, hexadecimal, binary), transfer an integer. A simple way of identifying this type is to view the drive parameter via the Parameter Overview Tool.

A-0-0190 Attributes

Data length:	4 byte data
Data type:	attribute of parameter in A-0-0180
Display format:	attribute of parameter in A-0-0180
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read / write in any phase

A-0-0191 Command Data #2

Refer to *A-0-0190 Command Data #1* for a description of this parameter.

A-0-0192 Command Data #3

Refer to *A-0-0190 Command Data #1* for a description of this parameter.

A-0-0195 Feedback Data #1

This parameter displays the real-time value that corresponds to the parameter identified in *A-0-0185 Optional Feedback ID #1*.

A-0-0195 Attributes

Data length:	4 byte data
Data type:	attribute of parameter in A-0-0185
Display format:	attribute of parameter in A-0-0185
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0.0000
Access:	read-only

A-0-0196 Feedback Data #2

Refer to *A-0-0195 Feedback Data #1* for a description of this parameter.

Multiplexing Parameters (A-0-0200 through A-0-0203) (DKC 2.3 only)

The VisualMotion system configures the MDT and AT based on the modes of operation specified in the application program and Axis parameter settings during system initialization (SERCOS Phase 2 to phase 3).

In DKC 2.3 drives, if the maximum telegram length is exceeded the system will automatically enable the multiplex (mux) channel. The system will then populate the mux with up to 5 IDNs base on the system and user selections. The system will also populate the cyclic portion of the telegram with up to 3 IDNs (8 bytes) that must be cyclically transferred. This is based on the system and user selections.

A-0-0200 MDT Multiplex Selection List (DKC 2.3 only)

This parameter contains a list of all supported MDT data identification numbers (IDN) usable within the VisualMotion system. It is automatically transferred to each DKC2.3 drive (S-0-0370) when mux support is required. This is a read only list and will be uploaded by VisualMotion to display valid selections for programming the Optional Command Data channel (parameters A-0-0180...182).

Note: Multiplex parameters are only support with DKC2.3 digital drives.

The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

A-0-0200 Attributes

Data length:	variable length 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

A-0-0201 AT Multiplex Selection List (DKC 2.3 only)

This parameter contains a list of all supported AT data identification numbers (IDN) usable within the VisualMotion system. It is automatically transferred to each DKC2.3 drive (S-0-0371) when mux support is required. This is a read only list and will be uploaded by VisualMotion to display valid selections for programming the Optional Command Data channel (parameters A-0-0190...191).

Note: Multiplex parameters only support with DKC2.3 digital drives.

The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

A-0-0201 Attributes

Data length:	variable length 1 byte data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

A-0-0202 MDT Multiplex Ident List (DKC 2.3 only)

This parameter list contains the Idents that the system has automatically placed in the MDT mux circular queue.

A-0-0202 Attributes

Data length:	variable length 1 byte data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

A-0-0203 AT Multiplex Ident List (DKC 2.3 only)

This parameter list contains the Idents that the system has automatically placed in the AT mux circular queue.

A-0-0203 Attributes

Data length:	variable length 1 byte data
Data type:	string array
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

Axis Parameter Lists (A-0-2000 and A-0-2001)

A-0-2000 List of All Parameters

This parameter contains a list of all axis parameters that are part of the current firmware version. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

A-0-2000 Attributes

Data length:	variable length 4 byte data
Data type:	unsigned integer
Display format:	IDN (parameter number)
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

A-0-2001 List of Required Parameters

This parameter contains a list of all required axis parameters that are part of the archive / restore function. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

A-0-2001 Attributes

Data length:	variable length 4 byte data
Data type:	unsigned integer
Display format:	IDN (parameter number)
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

6 VisualMotion's I/O System

6.1 Overview

VisualMotion's I/O system supports the following I/O devices. These I/O devices communicate with VisualMotion through user assigned registers. VisualMotion GPP 9 and GMP 9 firmwares provide 1,024 16-bit registers.

Type	Firmware Support
Local RECO02	GPP 9
Remote SERCOS RECO02	GPP 9 / GMP 9
DIAX03/04 Digital Drive I/O Modules	GPP 9 / GMP 9
ECODRIVE03 EcoX Bus System	GPP 9 / GMP 9

Table 6-1: VisualMotion I/O Devices

Refer to the *VisualMotion 9 Project Planning* manual for I/O device details.

6.2 I/O Configuration Tool

The I/O Configuration tool is a Windows™ based editor used to program, monitor and document all control and drive I/O devices in a VisualMotion system. This tool is available in project or service mode. Online editing of an I/O configuration is supported.

The I/O Configuration Tool is launched by selecting **Commission** ⇒ **I/O Setup**.

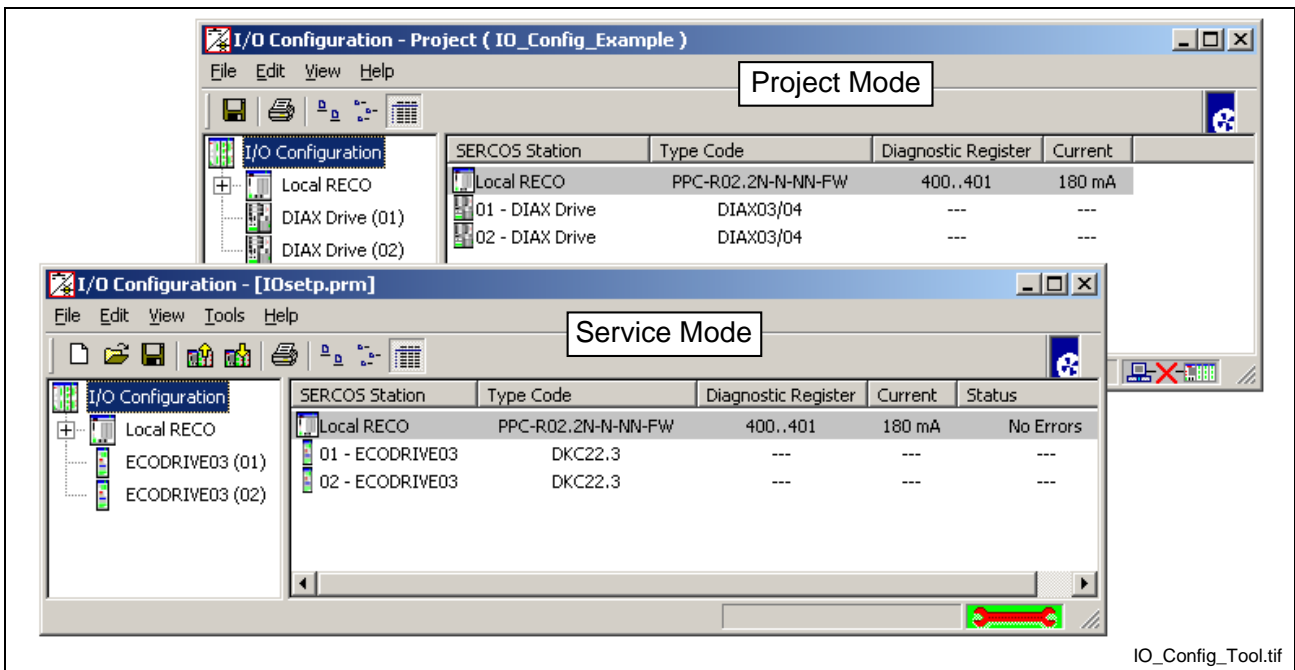


Fig. 6-1: I/O Configuration Tool

RECO02 Error Detection

I/O Configurations consists of a SERCOS device (Local RECO, RMK controller or digital drive) and individual I/O modules. Two diagnostic registers are assigned to the Local RECO02 and each RMK controller for use as a 32-bit status word for monitoring the status of each configured module. RECO02 I/O status words are scanned every I/O Mapper scan time (2 or 4 ms).

The following figure shows the bit structure for reporting errors and indicating the module at fault. The lower 16-bits specify the RECO rack slot number in which the I/O module error exists. Bit 0 refers to RECO rack slot 0 and bit 15 refers RECO rack slot 15. Refer to **Slot Addressing of the RMB02.2 Racks** in the *VisualMotion 9 Project Planning* manual for details.

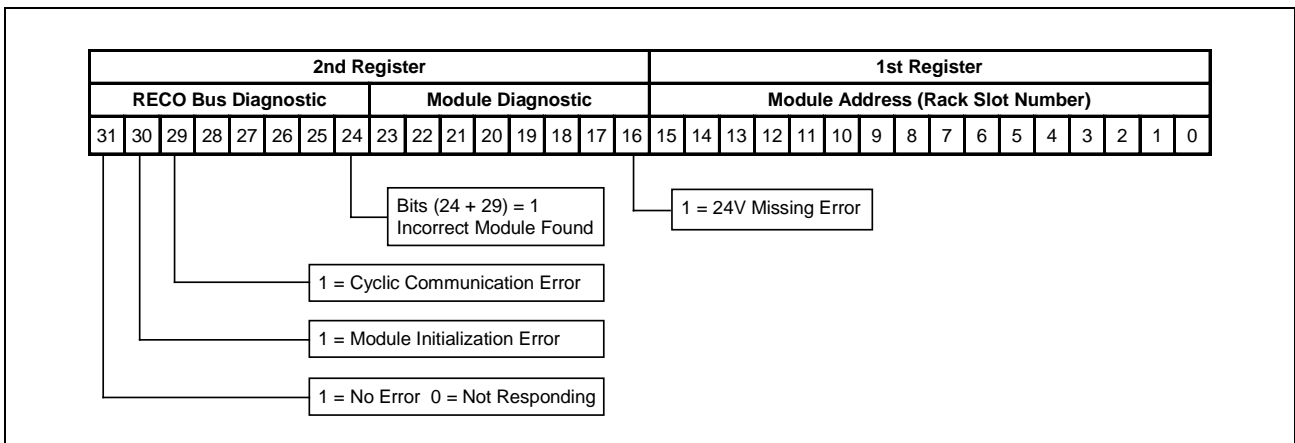


Fig. 6-2: 32-bit RECO Status Word

The following table describes all supported errors for RECO02 modules. RECO02 I/O errors, other than the 24V Missing error, can only be cleared/reset by cycling power to the control after correcting the cause of the problem.

RECO02 Error	Description
No Error	All configured RECO modules were found and working properly.
Not Responding	This error is normally generated at the RECO controller level.
24V Missing	Output module is missing 24V supply. Verify that all output module connectors are properly connected to 24V.
Incorrect Module Found	The I/O module identified does not match the original I/O Configuration.
Cyclic Communication	A cyclic communication error occurred in the I/O module identified in the <i>Module Address</i> .
Module Initialization	An error occurred during the initialization of the I/O module. Contact Bosch Rexroth Service.

Table 6-2: RECO02 Error Description

When a RECO error is issued, the SERCOS device and I/O module name are displayed as bold text in the left window and as red text in the right window.

Note: Digital drive I/O module errors are not monitored by VisualMotion. I/O module errors are reported directly to the digital drive. Refer to the appropriate *Digital Drive* manual for parallel I/O module errors.

6.3 Menu Selection

The availability of menu selections varies based on the current mode of communication (project/service). Inactive menu selections are grayed out.

The File Menu

The **File** menu contains standard Windows™ commands, such as Save and Print. The Upload and Download Configuration are specific I/O Setup functions and are only available in service mode.

New Ctrl+N

This selection is available in service mode and is used to create a blank I/O Configuration work area.

Open... Ctrl+O

This selection is available in service mode and is used to open existing I/O configuration (*.prm) file.

Upload Configuration

This selection is available in service mode and is used to upload the current I/O configuration from the control's memory to the I/O Configuration Tool.

Download Configuration

This selection is available in service mode and is used to download any modifications made to the I/O configuration to the control's memory.

The Edit Menu

The Edit menu is used to make additions and modifications to the User I/O configuration. All edit menu selections are available in both project and service mode. Selections are made available or gray-out based on the I/O device icon selected in the tree structure. For example, if a user clicks on the "I/O Configuration" tree icon, the "Add SERCOS Device" would be the only selection available. The user can select either from the edit menu or by right clicking over any I/O Device icon. The following table lists the edit selections available for each I/O device tree icon.









Tree Icon	Edit Menu Selection
 I/O Configuration	Add SERCOS Device
 Local RECO or  RMK I/O Controller	Add I/O Module Remove I/O Station Modify I/O Station
 DIAX Drive (04) or  ECODRIVE03	Add I/O Module Remove Drive Modify Drive
RECO I/O Module  2 - 32 Inputs @ 24VDC DIAX I/O Module  1 - 15 Inputs/16 Outputs @ 24VDC ECODRIVE I/O Module  1 - 16 Inputs/16 Outputs @ 24VDC	Remove I/O Module Modify I/O Module

Table 6-3: Edit Menu Selections

Add SERCOS Device

Select the appropriate SERCOS device to add to the I/O Configuration. Only one PPC-R control can be selected in an I/O configuration. The following selections are available:

- PPC-R01 (Single Slot)
- PPC-R02 (Double Slot)
- RMK I/O Controller
- DIAX Drive with I/O Module
- ECODRIVE with Eco-X I/O Module

Note: When using a PPC-P11.1 control, the PPC-R selections are not available. The only available SERCOS devices are RMK controllers and digital drives.

I/O Station Setup

Select the SERCOS address to match the address that will be used for the device. For PPC-R controls, select Local RECO. RMK I/O Controllers and digital drives use the selected SERCOS address set on the front of the unit.

Note: Registers can not be assigned to digital drives for status and diagnostic purposes. Only I/O modules installed in digital drives can be assigned registers.

The **Diagnostic Register** section is used to assign a register for monitoring the status of the I/O station. Click on the **Select/View Register** button to add a register number to the I/O station.

The *Register Assignment* window displays the available registers in the current project. Select the desired register number and click the **OK** button. The number of required registers will be automatically selected (highlighted).

Note: The assignment of I/O registers to SERCOS devices and I/O modules can be performed while adding the devices or by selecting **Edit** ⇒ **Auto-Assign Registers** after all the devices are configured.

Add I/O Module

Select the type of I/O module to configure for a Local RECO, RMK Controller or digital drive. The **Add I/O Module** selection is only available when a Local RECO, RMK I/O Controller or digital drive is selected in the tree structure. The number of assigned registers in a VisualMotion I/O system is based on the configured I/O modules.

Local RECO and RMK Controller I/O Modules

The following I/O modules are available for Local RECO and RMK controllers.

Type (radio button)	Selection	Description	Required Registers
Analog	RMC02.2-2E-1A	RECO02 Analog Module (2 Inputs / 1 output)	4
Digital Input	RME02.2-16-DC024	RECO02 Digital 16 Input Module	1
	RME02.2-16-AC115	RECO02 Digital 16 Input Module	1
	RME02.2-32-DC024	RECO02 Digital 32 Input Module	2
Digital Output	RMA02.2-16-DC024-200	RECO02 Digital 16 Output Module	1
	RMA02.2-16-DC024-100	RECO02 Digital 16 Output Module	1
	RMA02.2-32-DC024-050	RECO02 Digital 32 Output Module	2
	RMA02.2-16-RE230-200	RECO02 Digital 16 Output Module	1
	RMA02.2-16-AC230-200	RECO02 Digital 16 Output Module	1

Table 6-4: Local RECO and RMK Controller I/O Modules

DIAX03/04 Digital Drive I/O Modules

The following I/O modules are available for DIAX03/04 digital drives.

Type (radio button)	Selection	Description	Required Registers
Analog Input	DAE02.1	Analog Input Module (2 inputs – 14 bit @ 10V)	2
	DRF02.1	Analog Input Module (2 inputs – 12 bit @ 10V)	2
Digital (Input/Output)	DEA4.1	Digital Drive 15 Inputs/16 Outputs Module	2
	DEA5.1	Digital Drive 15 Inputs/16 Outputs Module	2
	DEA6.1	Digital Drive 15 Inputs/16 Outputs Module	2
	DEA8.1	Digital Drive 32 Inputs/24 Outputs Module	4
	DEA9.1	Digital Drive 32 Inputs/24 Outputs Module	4
	DEA10.1	Digital Drive 32 Inputs/24 Outputs Module	4

Table 6-5: DIAX03/04 Digital Drive I/O Modules

ECODRIVE03 Eco-X I/O Module

The EMD module is an expansion I/O module that interfaces with the DKC22.3 using the EcoX bus system. EcoX communication is designed for a DKC22.3 digital drive using SGP20 firmware. Refer to the *VisualMotion 9 Project Planning* manual for details.

Remove I/O Station (Remove Drive)

Select the I/O station (Local RECO, RMK Controller or Digital Drive) to be removed from the I/O configuration. The *Remove I/O Station or Drive* selection is only available when a Local RECO, RMK I/O Controller or a digital drive is selected in the tree structure.

Note: Any I/O modules configured under the selected I/O station will also be removed.

Modify I/O Station (Modify Drive)

Select the I/O station (Local RECO, RMK Controller or Digital Drive) to modify. The **Modify I/O Station or Drive** selection is only available when a Local RECO, RMK I/O Controller or a digital drive is selected in the tree structure. Only the address and register assignment of an I/O station can be modified.

Remove I/O Module

Select the I/O module to be removed from an I/O station or digital drive. The **Remove I/O Module** selection is only available when a RECO or digital drive I/O module is selected in the tree structure.

Modify I/O Module

Select the I/O module to be modified in the I/O station or digital drive. The **Modify I/O Module** selection is only available when a RECO or digital drive module is selected in the tree structure. Only the I/O module's slot and register assignment can be modified.

Auto-Assign Registers

This menu selection allows the user to add default starting registers to all configured I/O devices. Used registers are displayed with an icon within each register number.

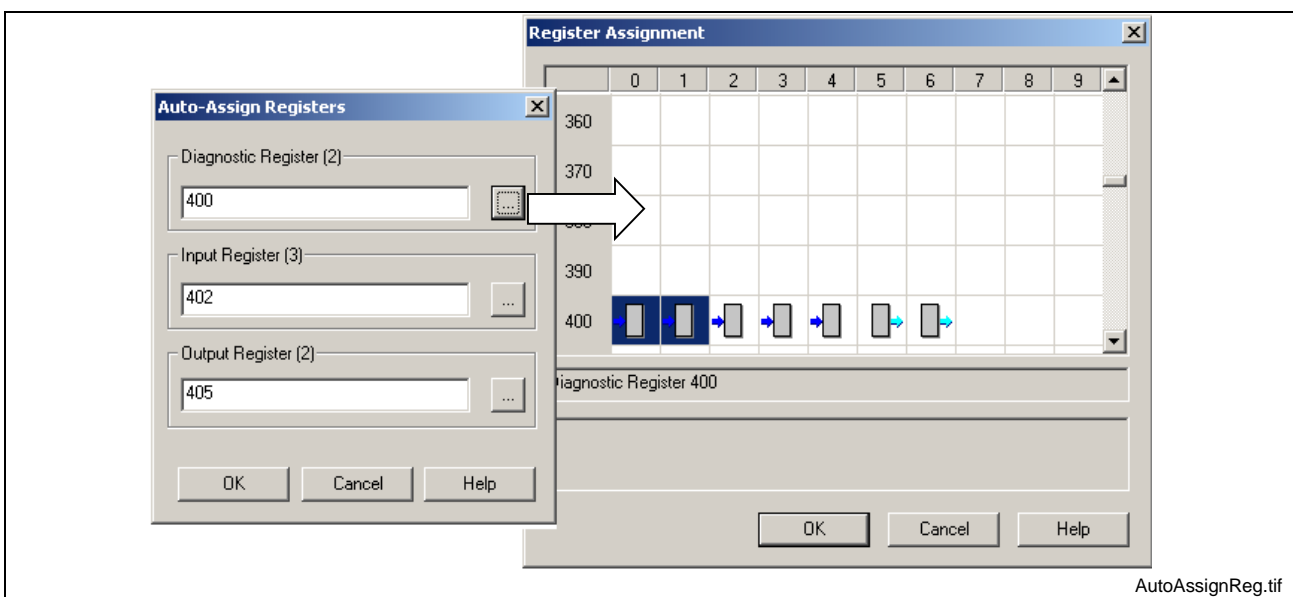


Fig. 6-3: Auto Assign Registers to I/O

RECO I/O Error Reaction

This menu selection allows the user to select one of the following error reactions for all RECO I/O modules in a system. The error codes displayed on the control dependent on the set RECO error reaction.

Error Reaction	Description
Ignore Errors	This setting will not stop motion to the system, but indicate a condition in the I/O Configuration Tool. No errors are displayed on the control.
Generate a Warning	This setting will not stop motion, but error diagnostic "215 RECO I/O Failure, see ext. diag" is displayed on the control.
Generate Fatal Error (Default Setting)	If an error is detected on any RECO I/O module, all motion to the system is stopped and error code diagnostic "544 RECO I/O Failure, see ext. diag" is displayed on the control until the error is corrected and the system is restarted.

Table 6-6: RECO I/O Error Reaction Settings

The View Menu

The View menu is used to increase the overall size of the work area by allowing the user to toggle the Toolbar and Status area on or off. A check to the right of the name indicates that the item is active.

Toolbar

Selecting **View ⇒ Toolbar** turns the icon toolbar on or off. The icon toolbar contains standard Windows™ commands. The icons available vary based on the mode of communication, as illustrated in Fig. 6-4.

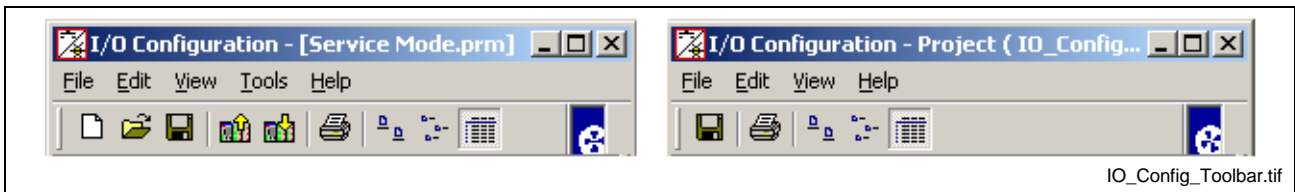


Fig. 6-4: I/O Configuration Toolbar

Status Bar

Selecting **View ⇒ Status Bar** turns the status indicator at the bottom of the window on or off. The status bar contains icon description text to the left, downloading and uploading percentage (when in service mode) and communication mode to the right.

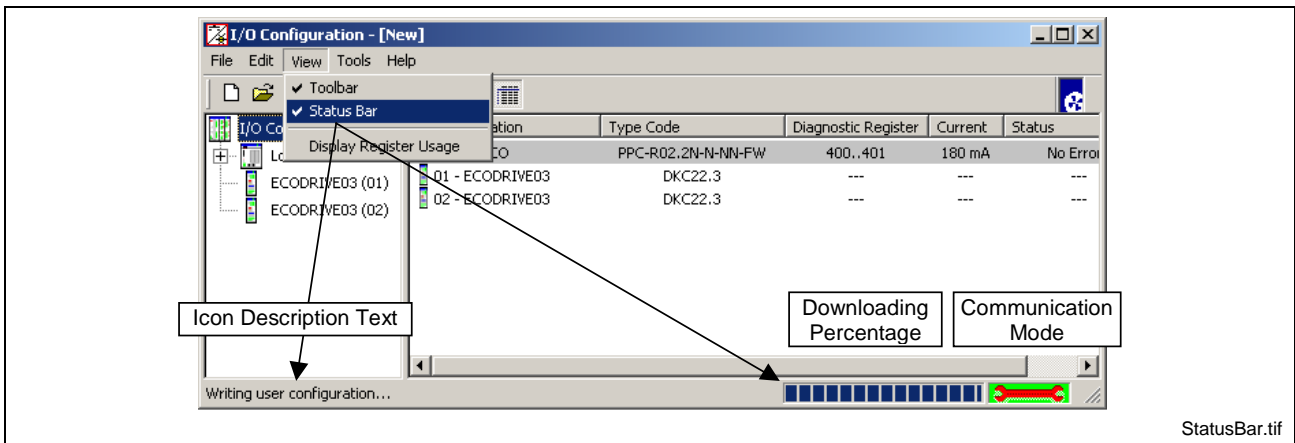


Fig. 6-5: I/O Configuration Status Bar

Display Register Usage

Each register assigned to an I/O device can have a maximum of 16 input or output points. Some registers are reserved for system functions, while others are recommended as defaults.

Selecting **View** ⇒ **Display Register Usage** opens a window indicating what system registers contain labels. Any registers that have been assigned to any RECO or Drive I/O will be displayed as input or output. System reserved registers display red crosshatches behind each I/O icon. Label names appear in the field just below the icons for the selected register, as shown in Fig. 6-6. Register labels can be assigned and edited by selecting **Edit** ⇒ **VM Data** from VisualMotion Toolkit's main menu.

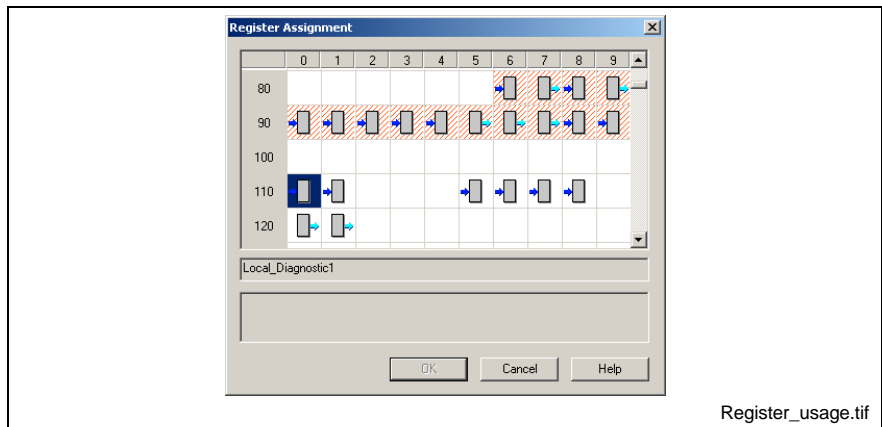


Fig. 6-6: I/O configuration Register Usage

The Tools Menu

Control Selection

Selecting **Tools** ⇒ **Control Selection** opens the window in Fig. 6-7. The user can select the method of communication as Serial or Network and set the appropriate target SERCOS address.

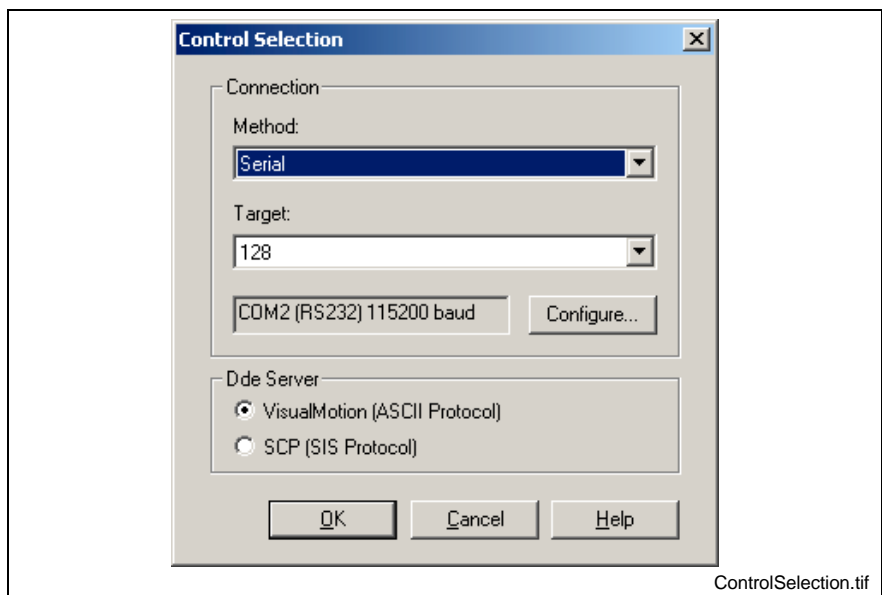


Fig. 6-7: I/O Configuration Control Selection

The Help Menu

Selecting **Help** ⇒ **Contents F1** opens the I/O Configuration portion in the VisualMotion help system.

Selecting **Help** ⇒ **About...** displays VisualMotion Toolkit version, licensed and contact information. For a listing of Bosch Rexroth Service and Support locations throughout the World, click the **Support** button.

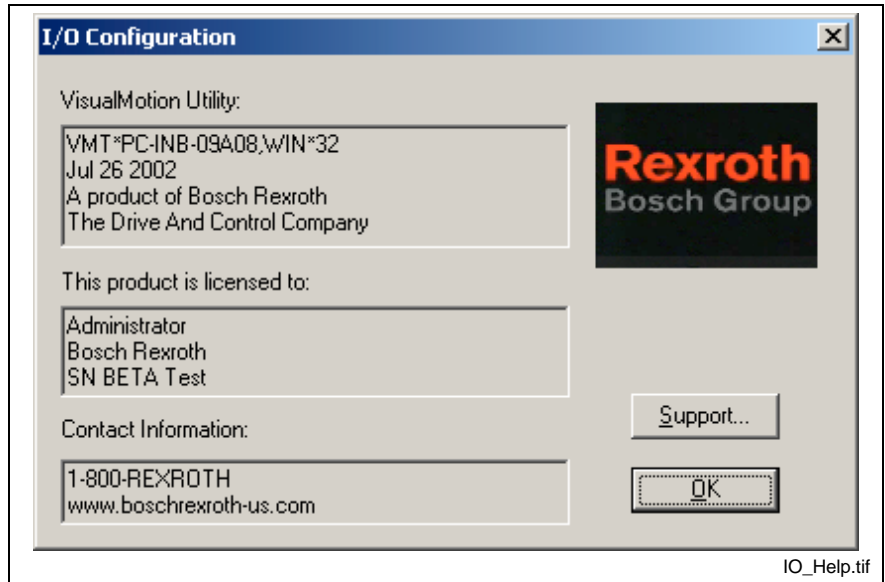


Fig. 6-8: About I/O Configuration Window

I/O Configuration in Project Mode

RECO02 and digital drive I/O modules can be configuration for a project in offline or online mode. Offline programming does not require the user to be connected to the control. While online mode, requires communication with the control.

Offline Programming

Use the following steps to create an I/O Configuration in offline mode.

1. Select **Commission** ⇒ **I/O Setup** to open the I/O Configuration Tool.
2. Right click the I/O configuration tree selection, select “Add SERCOS Device” and begin by adding a PPC-R control (Single or Double Slot) to the Local RECO.
3. To add RECO02 modules, refer to Add I/O Module on page 6-4.
4. To add a remote RECO02 I/O rack, follow step 2 and select “RMK I/O Controller” as the SERCOS device. Select a SERCOS address that will not be used by any other SERCOS device or drive. Set the SERCOS address on the RMK to match your selection. To add RECO02 I/O modules to the remote I/O station, refer to Add I/O Module on page 6-4.
5. To add a digital drive (DIAX04/ECODRIVE), follow step 2 and select one of the following digital drives:
 - DIAX Drive with I/O Module
 - ECODRIVE with Eco-X I/O ModuleSelect a SERCOS address that will not be used by any other SERCOS device or drive. Set the SERCOS address on the drive to match your selection. To add I/O modules to the drive, refer to Add I/O Module on page 6-4.
6. Save your I/O configuration to the offline project file.

Switching an I/O Configuration to Online Mode

To synchronize an I/O configuration file commissioned offline, open the project containing the I/O configuration and switch the control to online mode (*File* ⇒ *Online*). VisualMotion detects any changes to the current project on the control (if downloaded previously) and informs the user of the components in the project that have changed. The user can accept the changes and download them to the control or establish a connection to the control without downloading any data by selecting “Go online Unsynchronized”.

Note: The system must in parameter mode before downloading the I/O configuration to the control.

Online Programming (Editing)

VisualMotion automatically detects connected RECO I/O (Local or Remote) modules and connected digital drives in online mode. All drive-based digital and analog I/O modules cannot be detected by the I/O Configuration Tool and must be configured manually. Online editing of configured I/O modules is supported. Refer to Synchronize Project Components on page 2-16 for details.

Importing an I/O Configuration into a Project

An I/O configuration downloaded to the control can be imported into the project while in online mode, or from another project or file when in offline mode.

Use the following steps to import an I/O configuration from data stored on the control.

1. Start VisualMotion Toolkit and open the target project.
2. Switch VisualMotion to online mode.

Note: To transfer control data, previously downloaded to the control, switch VisualMotion to online mode.

To import data from another offline project or file, switch VisualMotion to offline mode.

3. Select **File** ⇒ **Import Project Component** from VisualMotion Toolkit's main menu.
4. From the "Transfer Control Data to Project" window, select the I/O Setup checkbox. By default, both control parameters C-0-0010 and C-0-2017 are checked. To transfer only the I/O User Configuration List, uncheck control parameter C-0-0010.

Note: The system must be in parameter mode before transferring data from the control.

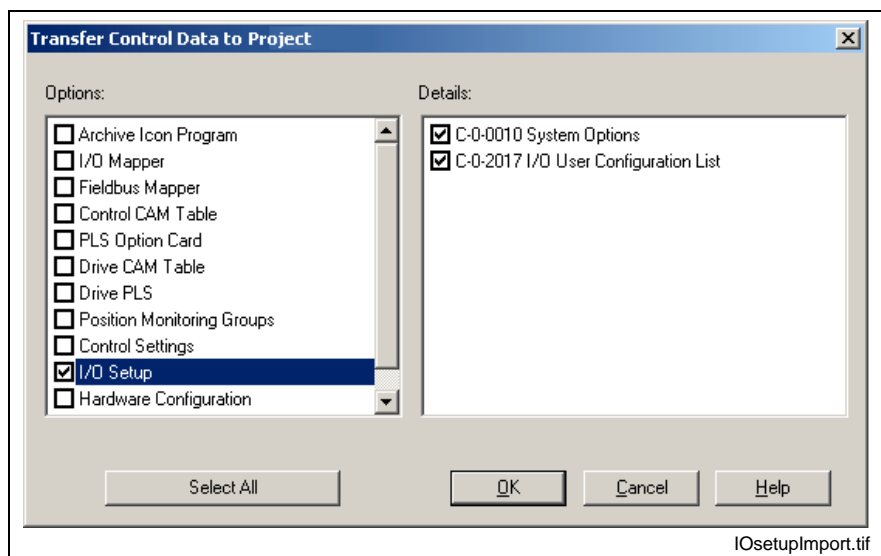


Fig. 6-9: I/O Setup Transfer from Control



5. Press the OK button to transfer the I/O Setup to the current project. The I/O configuration data is now synchronized with the current project.

Note: When importing data from another project or file in offline mode, the data is not synchronized with the project until VisualMotion is switched to online mode and the data is downloaded to the control.

I/O Configuration in Service Mode

VisualMotion's service mode allows the user to make modifications to an I/O configuration stored in the control's memory. A serial or Ethernet connection, with established communication, to the control is required before proceeding.

The following steps outline the procedure for uploading and downloading an existing I/O configuration from and to the control for modifications.

1. Start VisualMotion and select the "Service" mode radio button. Next, select **Commission** ⇒ **I/O Setup** to open the I/O Configuration Tool.
2. Switch the control to parameter mode.
3. Upload the I/O configuration by selecting **File** ⇒ **Upload Configuration** or click the upload icon ().
4. Make the necessary modifications to the I/O configuration. Save the file and download the modifications to the control by clicking the download icon ().

Note: I/O configurations downloaded to the control or saved to a file in service mode are not synchronized with a project's offline data. It is the responsibility of the project manager to ensure that I/O configurations modified in service mode are imported into the appropriate VisualMotion project.

7 I/O Mapper

7.1 Overview

The I/O Mapper is a Windows based editor used to program, monitor and document PLC logic functionality on a GPP/GMP control. PLC logic functionality can be edited in graphical Ladder logic or textual Boolean equation. The following table lists the supported PLC functionality.

Logic	Description	Quantity
AND	TRUE state of connected logic must exist to set output	5000 total operators
OR	TRUE state of either connected logic must exist to set output	
NOT	inverse of logic is considered TRUE	
One Shot Relay	re-triggered	32 each
Latch Relay	latch or unlatch	
Binary Shift Register	cascadable 32 bit left or right shifting	
Timer	Time on delay	
Counter	cascadable Up/Down with preset	

Table 7-1: PLC Functionality

Four windows are provided for the navigation, creation and cross referencing of an I/O Mapper file.

- Rung Select - navigate between rungs
- Ladder Editor - create graphic Ladder logic
- Boolean Equations Editor - create and modify Boolean equations
- Cross Reference - available when cross reference is requested. Refer to Cross Reference on page 7-8.

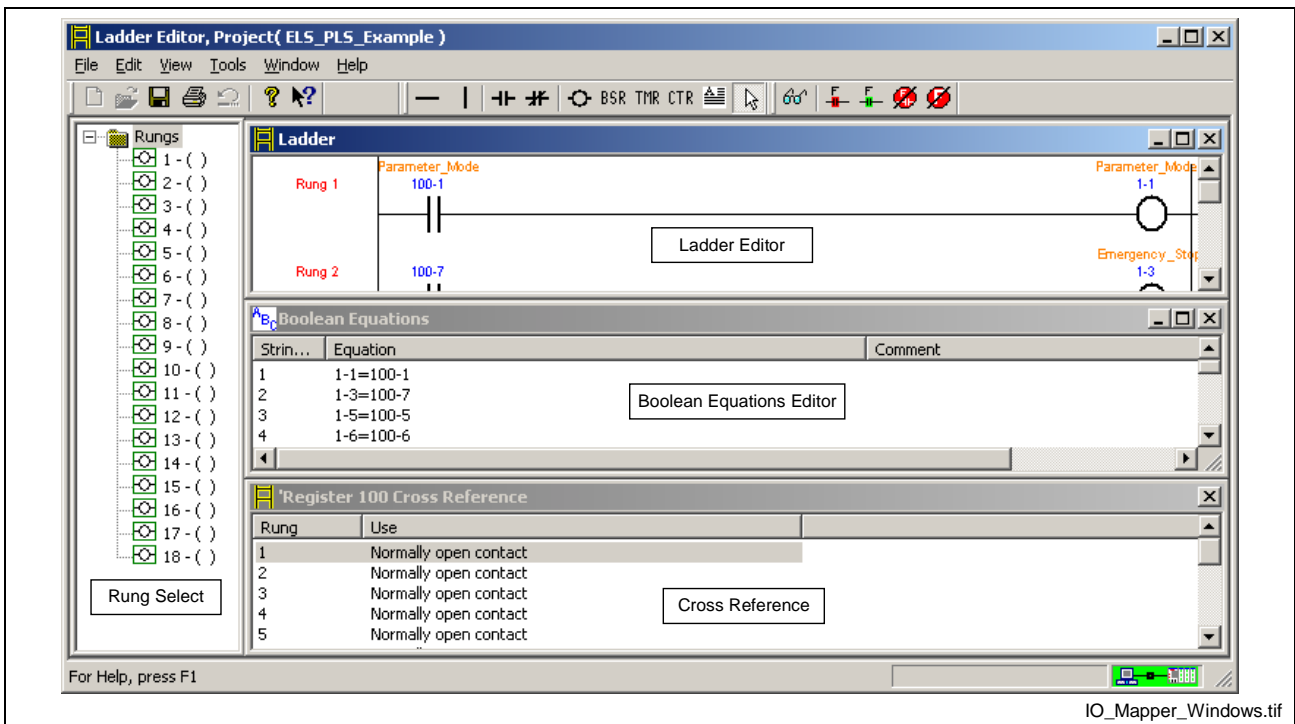


Fig. 7-1: I/O Mapper Windows

Ladder logic is converted and saved as Boolean strings. The Boolean strings are downloaded to the control in phase 2 (Parameter Mode), converted to native microprocessor code and run to completion every 2 or 4 ms (I/O Mapper scan time). On power-up, scanning begins as soon as the control reaches phase 2. On a download, scanning begins upon successful conversion and storage.

An I/O Mapper can be created in project or service mode. Refer to I/O Mapper in Project Mode on pages 7-20 and I/O Mapper in Service Mode on page 7-21 for details.

Online editing of an I/O Mapper file is supported. Refer to Synchronize Project Components on page 2-16 for details.

7.2 Specifications

The I/O Mapper specifications are provided as a guide to ensure successful programming. The maximum numbers listed in the different tables exist to guarantee that the entire I/O Mapper file runs to completion every I/O Mapper scan time.

Ladder Window

The total number of rungs that can be placed in the Ladder window is limited by the maximum number of rungs and/or operators used. Refer to Boolean Equations on page 7-3 for operator details.

Type	Max. Number
Rungs	1000

Table 7-2: Ladder Window Limitations

Rung Design

A rung is defined as a single or collection of contact(s) connected to a single coil or function. A rung displayed in the Ladder window is limited by the following.

Rung Component	Max. Number
Rows	7 per rung
Contacts	7 per row
Coils or function	1 per rung

Table 7-3: Rung Specifications

Although the Ladder window can display a rung containing 7 rows with 7 contacts on each row, the number of characters required to create the Boolean equation would exceed the maximum. An error is issued when the maximum number of characters is exceeded. Refer to Boolean Equations for details.

Boolean Equations

Boolean equations (strings) are displayed by selecting **Windows** ⇒ **Boolean Equations**. The following limits should be observed when creating an I/O Mapper file using the *Boolean Equation* editor.

Boolean Equation Components	Max. Number
Boolean string length	160 characters
Strings	1000
Operators	5000

Table 7-4: Boolean Equation Specifications

Boolean String example:

1-3=((!100-1&100-2&100-3)|100-8|100-11|100-13)) ;Rung comment

The example above contains the following:

- 48 characters in Boolean string
- 11 operators
- 14 characters in the comment, including a semicolon and null termination

Note: A warning is issued if a complex Boolean equation is created but cannot be represented as Ladder logic. If the complex Boolean equation is used, the Ladder window is disabled.

Boolean String Operators

Boolean strings consist of register bits, functions and operators. Comments are not counted towards to maximum Boolean string length. These operators are listed in the following table.

Operator	Description
(open parenthesis
)	closed parenthesis
=	equal to
&	AND statement
!	NOT statement
	OR statement

Table 7-5: Boolean Operators

Memory Usage

To check the I/O Mapper memory status, right click in the Ladder window and select *Check rungs and convert to Boolean strings*. The following VisualMotion Message displays a summary of the number of rungs, Boolean strings and operations with percentage of memory used.

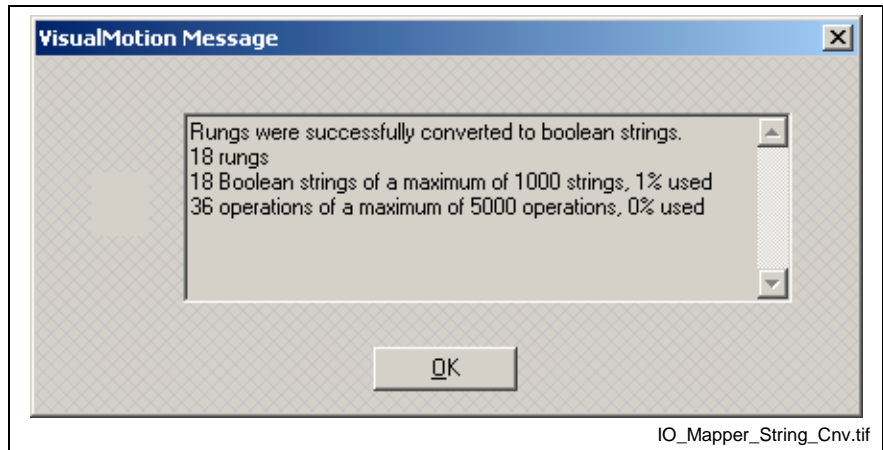


Fig. 7-2: Memory Usage

7.3 Menu Selection

Many of the menu selections used in the I/O Mapper have standard Windows™ functionality. Only those menu selections that are unique to the I/O Mapper will be described.

The availability of menu selections varies based on the current mode of communication (project/service). Inactive menu selections are grayed out.

The File Menu

The **File** menu contains commands for standard Windows functions, such as Open, Save, Print, etc. The *Get Ladder from control* and *Send Ladder to control* are I/O Mapper specific functions and available only in service mode.

New

This menu selection is only available in service mode and used to open a blank Ladder editor window for creating a new I/O Mapper file.


Open

This menu selection is only available in service mode and used to open an I/O Mapper file (*.iom). The file can be modified, save or downloaded to the control.


Import

This menu item has been disabled. Refer to Import Project Component... on page 2-12 for details.

Get Ladder from control

This menu selection is only available in service mode and is use to upload an I/O Mapper file previously downloaded to the control. This function can also be performed by clicking on the upload icon .

Send Ladder to control


This menu selection is only available in service mode and is used to download the Boolean string equivalent of the Ladder program to the control. If a change is made to the Ladder, the user will be prompted to check the rungs before the Boolean strings are downloaded. This function can also be performed by clicking on the download icon .

Printing an I/O Mapper File

Selecting **Print...** **Ctrl P** will print the entry Ladder logic with the Boolean equation directly above each rung. If a change has been made to the Ladder, the user will be prompted to check the rungs before the Ladder is printed.

The Edit Menu

The **Edit** menu contains Windows™ editing features for both the Ladder and Boolean Equations windows.

Note: The **Edit** menu is not available when the status icon  is selected. By default, the status icon is selected to show the status of the I/O Mapper when online.

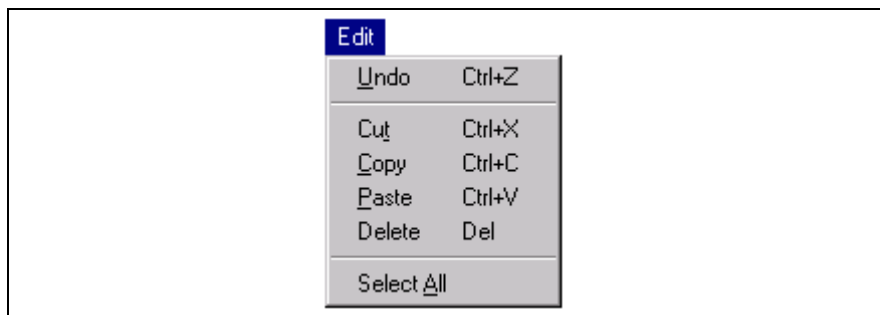


Fig. 7-3: The Edit Menu

The View Menu

The **View** menu is used to activate the toolbar and status bar within the I/O Mapper. In addition, the user can switch the Ladder logic window to status mode by selecting Status, pressing F7 on the keyboard or clicking on the Status icon toolbar. A check represents an active selection. To increase the overall size of the other windows, uncheck the desired views.

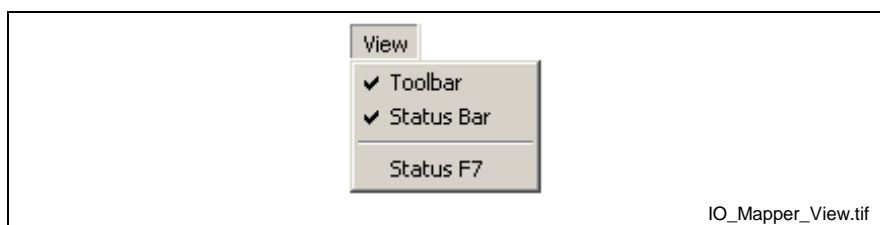



Fig. 7-4: The View Menu

The Tools Menu

The **Tools** menu commands are used to set the Forcing Options for logic functions, set the Mapper scan time and set the method of communication and number for the control.

Note: The **Tools** menu is not available when the status icon  is selected. By default, the status icon is selected to show the status of the I/O Mapper when online.

Forcing Options

Selecting **Tools** ⇒ **Forcing Options** allows the user to set forcing options for the I/O Mapper. If the *Disabled* option is selected, the forcing icon in the Ladder window will not function. Refer to *Forcing Options* for details. Cycle power to control for changes to take effect.

Scan Time

Selecting **Tools** ⇒ **Scan Time** allows the user to set the I/O Mapper scan time to either 2ms or 4 ms.

Control Selection

The *Card Selection Setup* window is only available in service mode and used to setup communications with the control. Refer to Control Selection on page 2-157 for details.

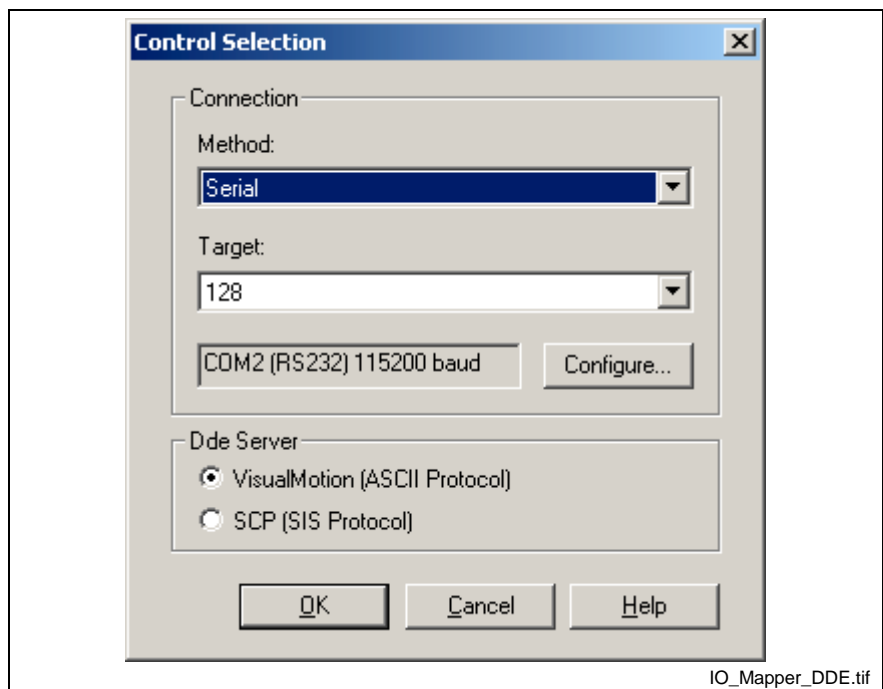


Fig. 7-5: Control Selection

The Window Menu

The **Window** menu is used to arrange the I/O Mapper's windows. The windows can be Cascaded or Tiled. The currently active window has a check mark to the left of the window name.

The Cross Reference selection is visible when requested. Refer to Cross Reference on page 7-8 for details

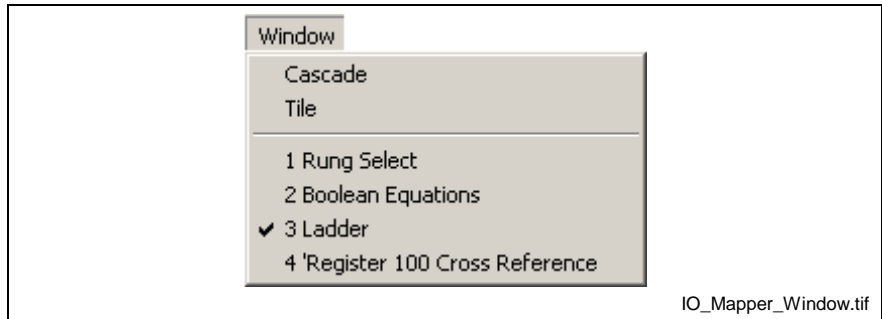


Fig. 7-6: The Window

The Help Menu

Selecting **Help ⇒ Contents F1** opens the I/O Configuration portion in the VisualMotion help system.

Selecting **Help ⇒ About...** displays VisualMotion Toolkit version, licensed and contact information. For a listing of Bosch Rexroth Service and Support locations throughout the World, click the **Support** button.

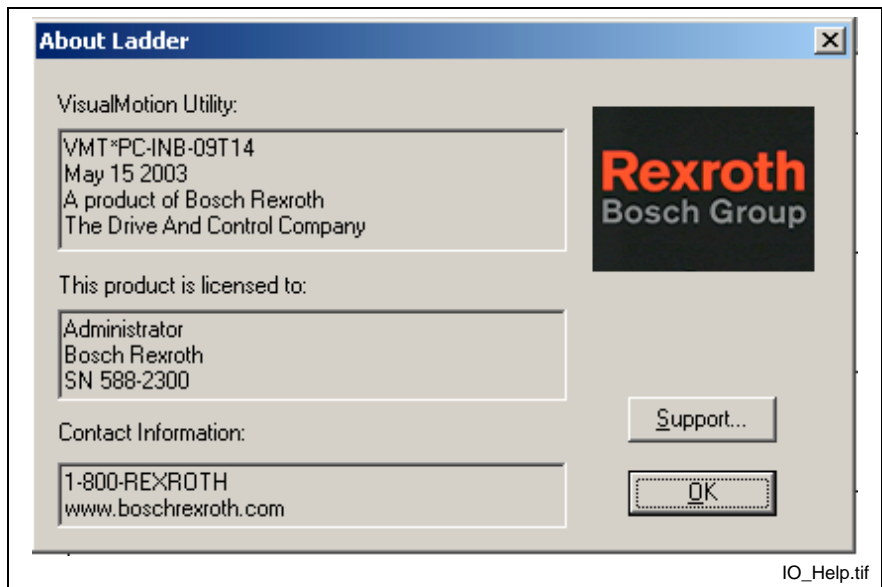
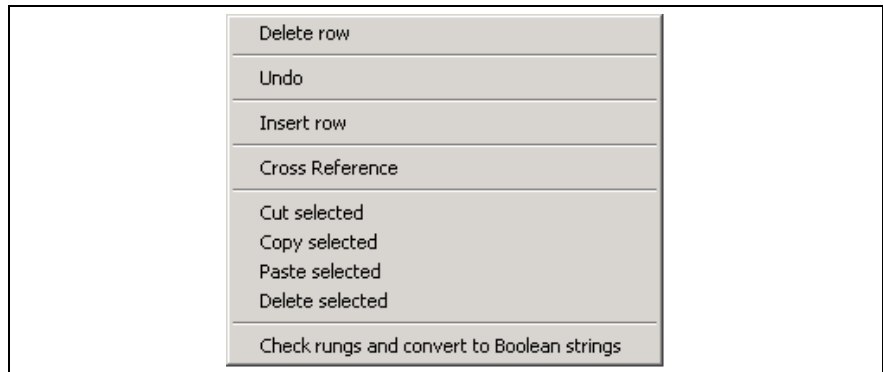


Fig. 7-7: About I/O Configuration Window

7.4 Additional Menu Selections

The following additional menu selections are available in the Ladder view area by right clicking anywhere in the view area.



Delete row

To delete a row (rung), select a point anywhere on the row, right click and select *Delete row*. VisualMotion warns before deleting the row. All rungs below the deleted one move up.

Undo

This function will undo the last function performed in the Ladder view area.

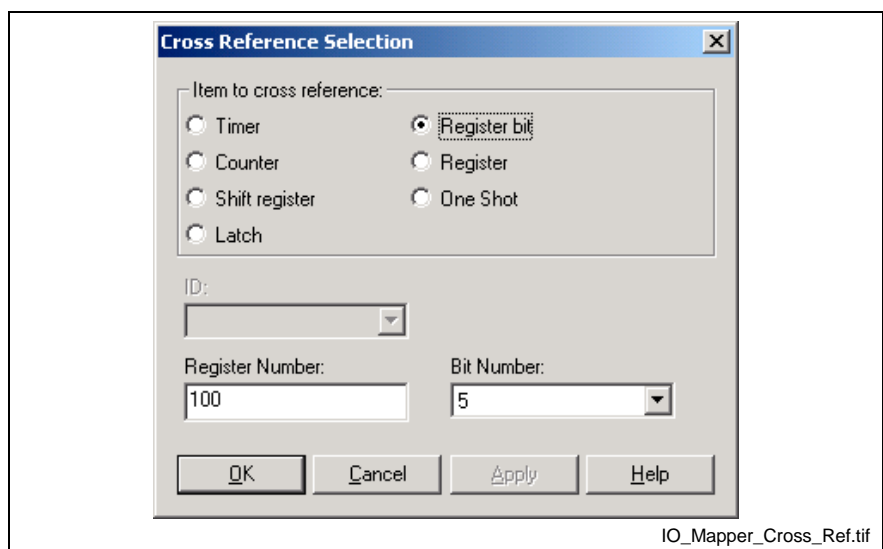
Insert row

This function adds a new blank rung at the selected location.

Cross Reference

The cross reference function is used to quickly view every location where a contact(s) or output is used in the I/O Mapper.

To view a cross reference for a contact or an output, click on the desired logic with the arrow selection cursor, right click and select Cross Reference. The Cross Reference Selection window opens with the properties of the selection logic.



IO_Mapper_Cross_Ref.tif

Fig. 7-8: Cross Reference Selection

Select the item to cross reference to and click on OK. A Cross Reference window will open listing every rung where the contact is used, including any output functions, such as timers or counters. Once open, this window can be cascaded or tiled from the *Window* menu.

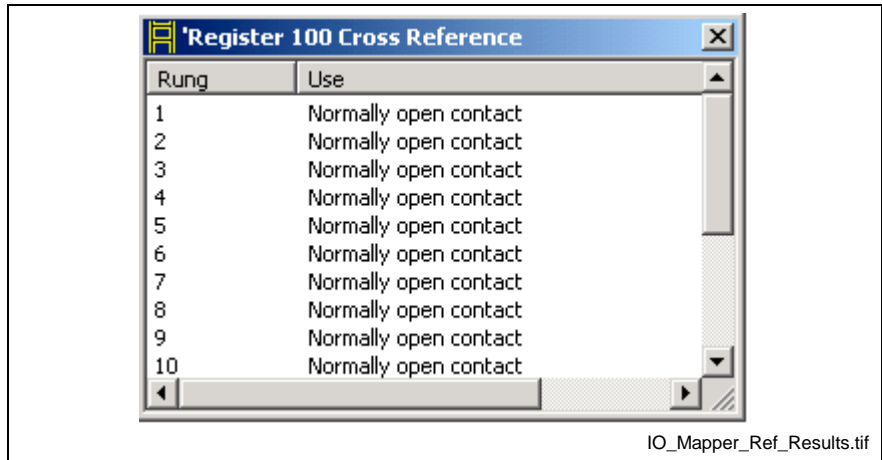


Fig. 7-9: Cross Reference Results

Double click the mouse on any location to display the rung at the top of the *Ladder* window.

Check rungs and convert to Boolean strings

This function verifies all rungs for connectivity, not necessarily functionality, and displays a summary of the number of rungs, Boolean strings and operations with percentage of memory used.

7.5 Toolbar Buttons

The Ladder and Forcing toolbar buttons are accessible when the Ladder window is active.

Ladder Logic Toolbar Buttons

The Ladder logic toolbar buttons are used for selecting and placing Ladder logic, connecting the logic and placing comments for individual rungs.

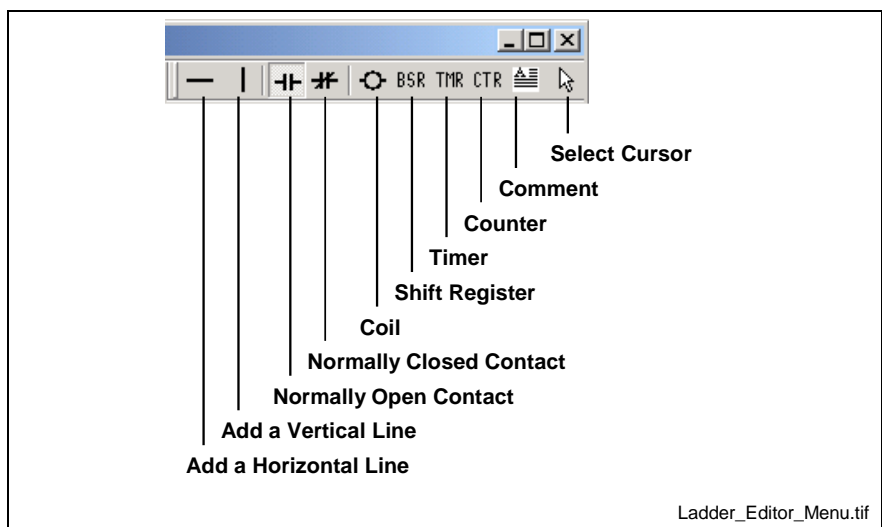


Fig. 7-10: Ladder Logic Toolbar Buttons

Contacts

Normally open and *closed* contacts are selected and placed to the left of a rung (row) the Ladder window. A contact can be overwritten by placing a new contact in the same position. Contact types are edited by double clicking the contact.

Lines

The *horizontal* and *vertical lines* are used for connecting multiple contacts to each other and to coils and functions. Unwanted lines can be erased by placing the same line type in the same position.

Coils and Functions

Coils and *functions* (Bit Shift Register, Timer, Counter) are selected and placed to the right of a rung (row) in the Ladder window.

Comment

Comments are placed above the desired rung. The maximum number of characters for a comment is 80. Comments are erased by selecting the comment, using the Select Cursor, and pressing the delete key.

Select Cursor

The *Select Cursor* is used for selecting a single or multiple area in the Ladder window. Multiple areas are selected by using a standard click and drag mouse function. Selected areas are shown with a green crosshatch.

Forcing Icon Toolbar Buttons

Forcing toolbar buttons are used to change the state of a contact regardless of how it may be mapped to other register bits. These toolbar buttons are only available in online and service modes.

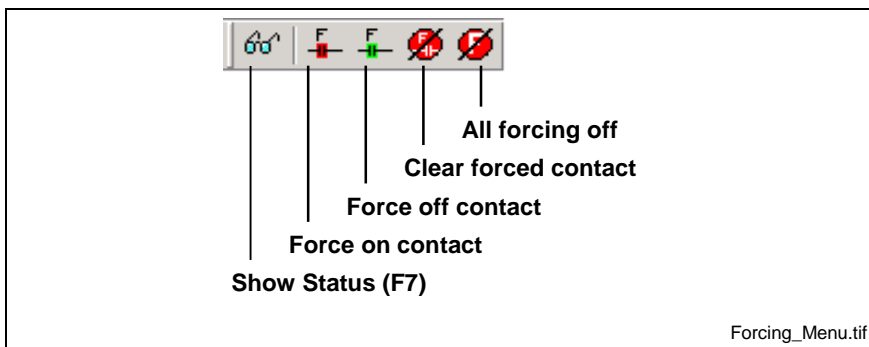



Fig. 7-11: Forcing Toolbar Buttons

Show Status (F7)

The *Show Status* toolbar button is active by default when the I/O Mapper is opened in online mode. When active the states of all contacts and functions can be monitored. Depress the Status (F7)  toolbar button to view contact state changes when using the Forcing toolbar buttons.

Forcing Options

Before the forcing toolbar buttons are functional, the forcing options must be set by selecting **Tools** ⇒ **Forcing Options**. The control must be in Parameter mode before changing forcing options.

**CAUTION**

I/O forcing is provided as a system configuration and debugging tool. Forcing can modify control register bits affecting the safe operation of the system. A failure of serial communication between the Host and VisualMotion will prevent the Host from being able to clear a forcing mask, possibly during active motion in the system.

Note: Timers, Counters, Shift Registers, Latches and One Shots along with their respective output contacts (Q) **cannot** be forced.

Disabled

When this option is selected, the state of a contact or coil **cannot** be forced.

Only Control Registers (Default)

When this option is selected, the state of a contact or coil for those assigned to a control register bit can be forced.


All Registers Allowed

When this option is selected, the state of all contacts and coils can be forced.


Using the Forcing Toolbar buttons

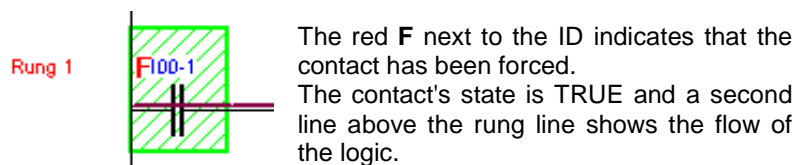
By default, only contacts and coils in the Ladder window that have a register bit assigned, can be forced. To disable this function or select all registers, modify the options under **Tools** ⇒ **Forcing Options**.


To force the state of a contact or a coil, follow the steps below:

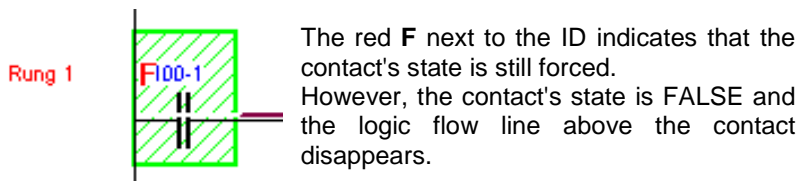
1. Select  and click on the desired contact or coil using the left mouse button. A green crosshatched area will cover the object.
2. From the Forcing toolbar buttons, select one of the following choices:





3. If you select ( Force on) and the forcing options are set to *All registers*, the contact or coil will be forced TRUE and appear as follows:



4. Now, if you select ( Force off) for the same contact in step 3, the contact's state will be forced to FALSE and appear as follows:



5. Select ( Clear force) for the selected contact to clear the forced states of steps 3 and 4.
6. To clear the forced state of the entire Ladder, select ( Forcing off).

VisualMotion Messages for Forcing

The following messages appear when the forcing toolbar buttons are used incorrectly:

Message	Description
Please select the area!	This message appears if a contact or coil is not selected in the Ladder window and the <i>Force on</i> , <i>Force off</i> or <i>Clear Force</i> toolbar buttons are selected.
Only registers can be forced	This message appears if a contact belonging to a function or a function is selected in the Ladder window and the <i>Force on</i> , <i>Force off</i> or <i>Clear Force</i> toolbar buttons are selected.

Table 7-6: VisualMotion Messages for Forcing

7.6 Input Logic Functions

The I/O Mapper provides Normally Open (**—|**) contacts and a Normally Closed (**—|/**) contacts as input logic. Placing a contact in the Ladder window opens the *Properties* window.

The *Properties* window is used for assigning a relationship between a contact and a VisualMotion register bit. It can also be used to select the type of contact. The default contact type is a Register bit. This can be modified by selecting the *Contact Setup* tab and choosing the desired type as shown in the figure below.

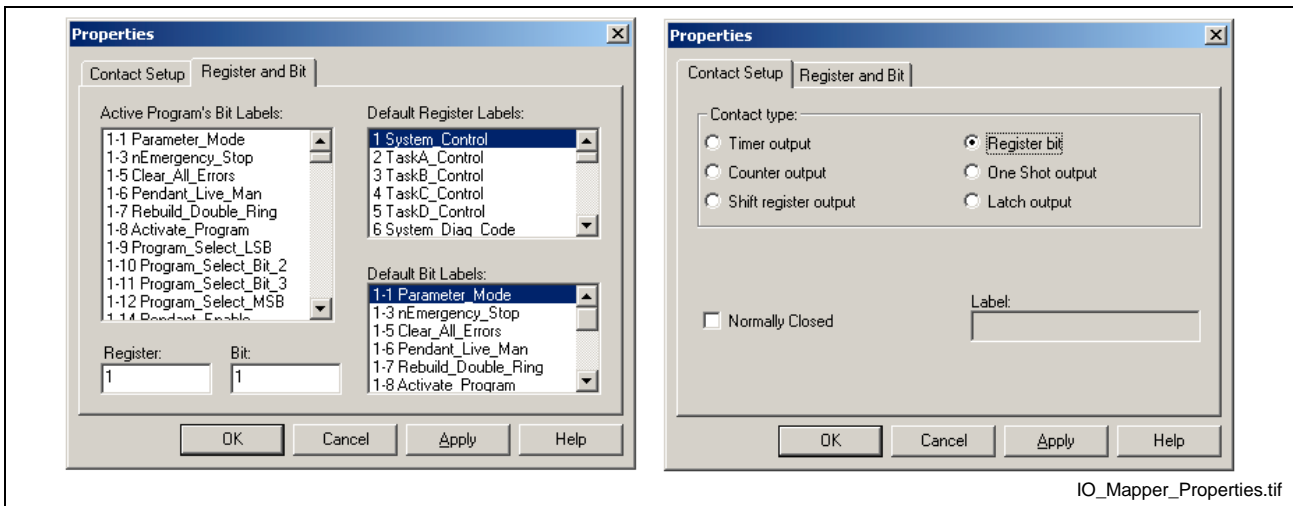


Fig. 7-12: Properties Window for Contacts

Register and Bit

The Register and Bit tab is used to select a VisualMotion register and bit. They can be entered directly, selected from a Default Register Bit Label or selected from a label in the Active Program. The default register bit labels are displayed in two separate sections. The user first selects the desired *Default Register Label* and then the *Default Bit Label*. The *Register* and *Bit* fields display the current selection. Once the desired label is selected, click on **Apply** to continue to the Contact Setup tab or **OK** to close and apply the label.

Contact Setup

The Contact Setup tab is used to select the contact type. The I/O Mapper provides the following 6 output contact types:

- |**T**| Timer output
- | | Register bit
- |**C**| Counter output
- |**S**| One Shot output
- |**B**| Shift Register output
- |**L**| Latch output

With the exception of the Register bit contact, the other output contact types are output signals from the functions provided with the I/O Mapper. Refer to Output Logic Functions on page 7-14 for details.

Note: When a contact type other than Register Bit is selected, the *Register and Bit* tab changes to *Contact Selection*.

Contact Selection for Functions

The *Contact Selection* tab displays the ID output names for the corresponding output contacts in the *Contact Setup* tab. The following table shows the range of ID names for the output contacts.


Contact Type	ID of Output Contact	Output Function
One Shot	OSR01Q - OSR32Q	OSR01 - OSR32
Latch	LAT01Q - LAT32Q	LAT01SET - LAT32SET and LAT01RESET - LAT32RESET
Timer	TON01Q - TON32Q	TON01 - TON32
Counter	UDC01Q - UDC32Q	UDC01 - UDC32
Binary Shift Register	BSR01Q - BSR32Q	BSR01 - BSR32

Table 7-7: Contact Selection ID

Note: When a Binary Shift Register output contact is selected, the user must also select the *Shift Register Bit* number. This number identifies which bit will set the contact when a 1 is shifted into that bit position using the Binary Shift Register function.

The ID format displayed in blue will have the following format:

BSR01Q12

Refer to Binary Shift Register (BSR)  on page 7-17 for details.

7.7 Output Logic Functions

The I/O Mapper provides coils, binary shift registers, timers and counters as output logic functions. With the exception of a normal coil, an output logic function's inputs and outputs use global integers to store their values. Global integer values are initialized to 0 on power-up unless their values are flashed to memory using the *Save Global Variables Command*. Refer to Saving Global Variables to Flash on page 2-108 for details.

Coil Relay

A coil is used to represent an output state that when TRUE, sets the corresponding assigned register bit, causing a response in the user program.

Normal Coil

A normal type coil is assigned to a register bit and is said to be TRUE when all contact states on the rung are also TRUE. A normal coil can be used as part of a simple output function or to complement a more complex function, such as a Binary Shift Register, Timer or Counter.

Fig. 7-13 is an example of a coil being used as a simple output function. Refer to Fig. 7-16, Fig. 7-17 and Fig. 7-18 for examples of a normal coil function complementing a more complex function.

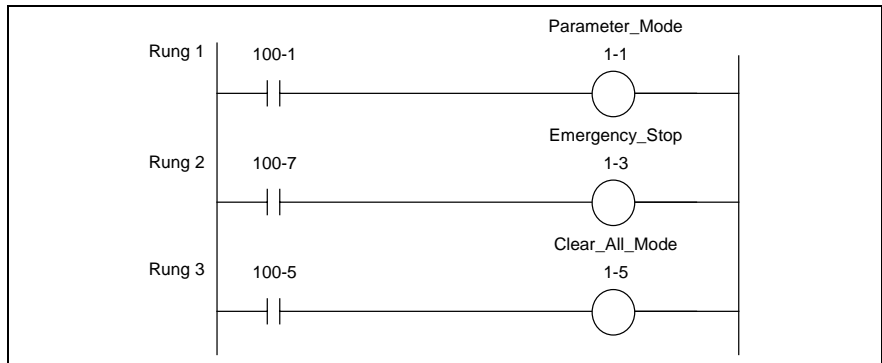


Fig. 7-13: Coil used as a Simple Output Function

One Shot Coil (OSR)

A **One Shot Relay** sets its Q output for one I/O Mapper scan time when the rung conditions are TRUE. The Q output is set for the duration of the set I/O Mapper scan time and then reset even though the rung conditions might still be TRUE. A One Shot will reinitialize when the next raising edge is detected.

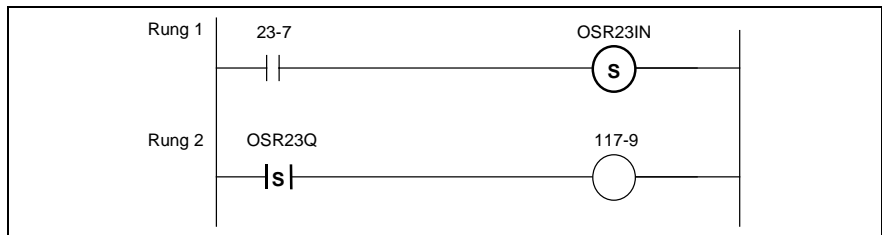


Fig. 7-14: One Shot Example

In the One Shot example above, when contact 23-7 is TRUE and the rung is scan for the first time, the Q output will set closing contact OSR23Q and energizing relay coil 117-9. The next time the rung is scanned, the One Shot output is inactive. The following table describes a One Shot's one input (IN) and one output (Q).

Signal Name	Description
OSRxxIN	a rising edge sets Q TRUE for one scan
OSRxxQ	output is set to 1 for one scan when rung is TRUE

Table 7-8: One Shot Functions

The One Shot input and output functions use the following predefined Global Integers (GIx) for storing their values.

OSR ID	Global Integer	Description
OSR xx where, xx represents counters 01-32	GI425	One Shots Q state, 32 bits
	GI426	One Shots IN , 32 bits
	GI427-GI428	Reserved

Table 7-9: One Shot Global Integers (GIx)

Latched and Unlatched (LAT)

A Latched coil allows momentary conditions to be maintained until a condition exists to reset (Unlatched) the function. The Latched and Unlatched outputs are used together to create a latch function. On a 0 to 1 transition of the SET input, to Q output is set to 1. On a 0 to 1 transition of the RESET input, the Q output is set to 0. The SET and RESET inputs are observed by a level and not by their transitional edge. Regardless of the order of execution, the RESET input always has a higher priority. All latches are reset on power-up.



Fig. 7-15: Latched and Unlatched Example

In the Latched and Unlatched example above, when contact 23-1 is closed, the SET input of the Latch output sets the Q output and energizes the relay coil 334-9. When contact 55-2 is closed, the RESET input of the Unlatched output resets the Q output and de-energizes the relay coil 334-9. The following table describes the Latch input (SET), the Unlatch input (RESET) and the function's one output (Q).

Signal Name	Description
LAT xx SET	a 1 sets the latch Q output to 1
LAT xx RESET	a 1 resets the latch Q output to 0
LAT xx Q	Latch output

Table 7-10: Latch and Unlatch Functions

The Latch and Unlatch input and output functions use the following predefined Global Integers (GIx) for storing their values.

LAT ID	Global Integer	Description
LAT xx where, xx represents counters 01-32	GI465	Latches Q state, 32 bits
	GI466	Latches history state, 32 bits
	GI467-GI468	Reserved

Table 7-11: Latch and Unlatch Global Integers (GIx)

Binary Shift Register (BSR) BSR

Binary Shift Registers are used to store the status of previous events and react to the status later. Each new change in status gets stored in the first bit and the remaining bits are shifted over.

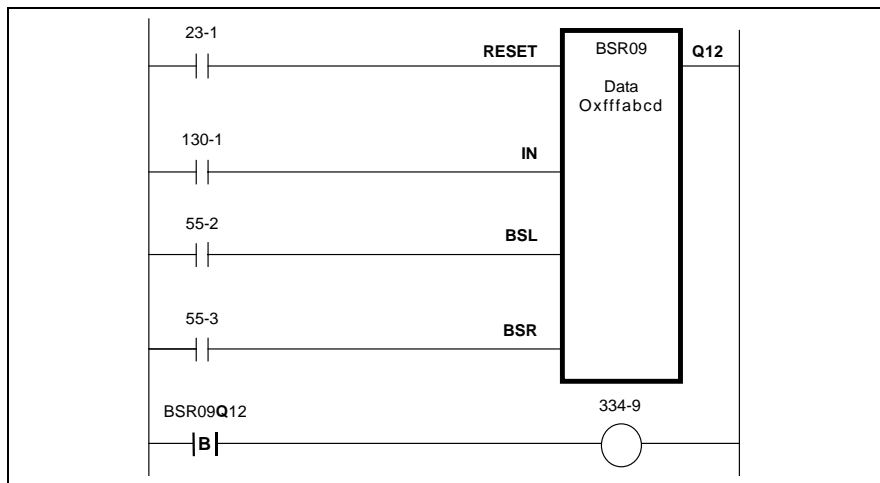


Fig. 7-16: Binary Shift Register Example

The bits in the shift register are shifted each time a raising edge is detected on either the **BSL** or the **BSR** inputs. Each Binary Shift Register has four inputs (RESET, IN, BSL, and BSR) and 32 output contacts. On a 0 to 1 transition of the BSL or BSR input, all bits are shifted left or right one bit and the state of input IN is set. All bits are available for evaluation. The shift register is initialized to 0 on power-up.

Signal Name	Description
BSRxxRESET	a 0 to 1 transition sets all shift register bits to 0
BSRxxIN	input data for bit 1 when <i>BSL</i> is set, bit 32 when <i>BSR</i> is set
BSRxxBSL	0->1 transition moves shift register data 1 bit to left, sets <i>BSRxxIN</i> value into bit 1
BSRxxBSR	0->1 transition moves shift register data 1 bit to right, sets <i>BSRxxIN</i> value into bit 32
BSRxxQyy	the number "yy" in the Q output contact refers to the bit number within the shift register (1 to 32).

Table 7-12: Binary Shift Register Functions

The Binary Shift Register's input and output functions use the predefined Global Integers (GIx) for storing their values.

BSR ID	Global Integer	Description
BSRxx where, xx represents counters 01-32	GI429	Shift Registers RESET history state, 32 bits
	GI430	Shift Registers IN history state, 32 bits
	GI431	Shift Registers BSL (Binary Shift Left) history state, 32 bits
	GI432	Shift Registers BSR (Binary Shift Right) history state, 32 bits
	GI433-GI464	Shift Registers Q, integer

Table 7-13: Binary Shift Register Global Integers (GIx)

Timer (TON) TMR

The **Timer ON** delay function counts in ms with the restriction that it can only count in multiples of the set I/O Mapper scan time. If a value other than a multiple of the scan time is used as *PT*, the timer will count until the next multiple of the scan time.

Example:

The scan time is set to 4 ms and the value of *PT* = 30. The *Timer* will count in multiples of 4 and set output *Q* at 32.

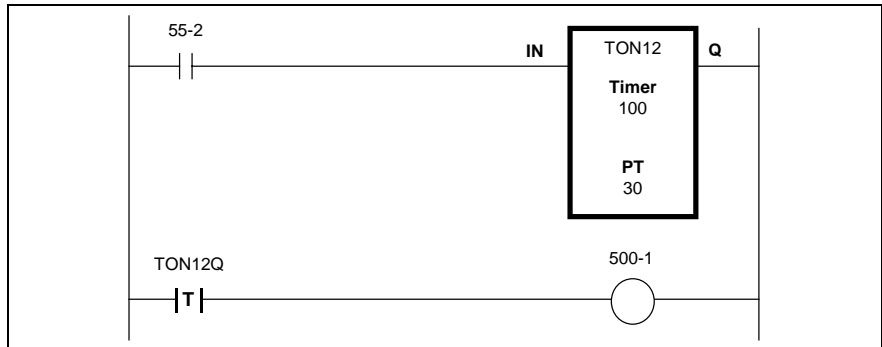


Fig. 7-17: Timer Example

In the Timer example above, the **Time** starts when contact 55-2 is closed (0 to 1 transition of **IN**) and stop when the contact opens. When the **Time** is equal to or greater than **PT**, the **Q** output will set closing contact TON12Q and energize relay coil 500-1. The following table describes a timer's two inputs (IN and PT) and two outputs (Q and TIME). The TIME value is initialized to 0 on power-up.

Signal Name	Description
TONxxIN	If a rising edge is detected, the on-delay timing is started. A falling edge resets ET to 0.
TONxxPT	Value at which the output will become high.
TONxxQ	Output is set to TRUE when the ET >= PT
TONxxTIME	Elapsed time

Table 7-14: Timer Functions

The Timer's input and output functions use the predefined Global Integers (Glx) for storing their values.

Timer ID	Global Integer	Description
TONxx	GI357	Timers Q state, 32 bits
where, xx	GI358	Timers IN history state, 32 bits
represents	GI359-GI360	Reserved
timers 01-32	GI361-GI392	Timers TIME, integer
	GI393-GI424	Timers PT, integer

Table 7-15: Timer Global Integers (Glx)

Counter (UDC) CTR

An **Up/Down Counter** is a function used for totaling the number of transitions in a process, or triggering a behavior when a target count is reached. A counter can be incremented up or down and sets its output **Q** when the **Count** value matches the **PV** value.

The counter's range is -2^{31} to $(2^{31}-1)$. It will rollover at $(2^{31}-1)$ to -2^{31} on a count up.

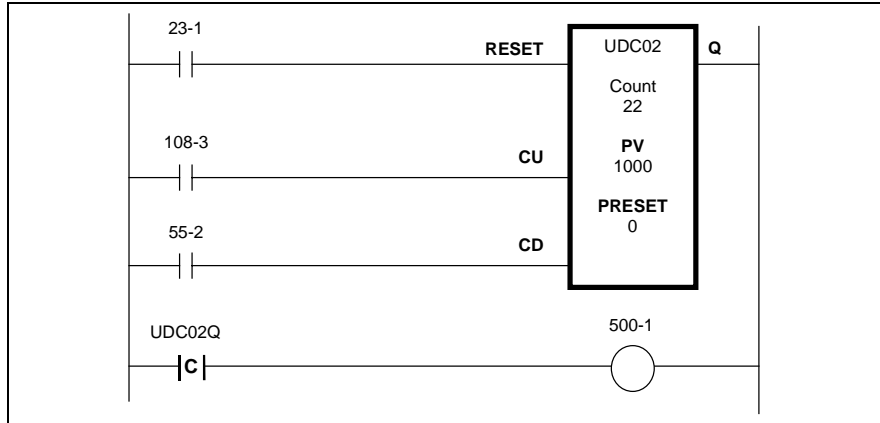


Fig. 7-18: Counter Example

In the counter example in Fig. 7-18, the **Count** value increments by 1 every time contact 108-3 is closed and decrements by 1 every time contact 55-2 is closed. Once the **Count** value is equal to or greater than the **PV** value, contact UDC02Q will close energizing relay coil 500-1. The following table describes a counter's five inputs (RESET, CU, CD, PV, PRESET) and one output contact (Q).

Signal Name	Description
UDCxxRESET	a 0 to 1 transition sets the <i>Count</i> value to the <i>PRESET</i> value
UDCxxCU	a 0 to 1 transition increments the <i>Count</i> by one
UDCxxCD	a 0 to 1 transition decrements the <i>Count</i> by one
UDCxxPV	the target value used to set the counter's corresponding contact (UDCxxQ)
UDCxxPRESET	the initial value that is used by the counter when activated or after a <i>RESET</i>
UDCxxQ	the counter's output contact is set to 1 when the <i>Count</i> is equal to or greater than the <i>PV</i> .

Table 7-16: Counter Functions

The counter's inputs and output functions use the predefined Global Integers (GIx) for storing their values.

Counter ID	Global Integer	Description
UDCxx	GI257	Counters Q state, 32 bits
where, xx represents counters 01-32	GI258	Counters CU history state, 32 bits
	GI259	Counters CD history state, 32 bits
	GI260	Counters RESET history state, 32 bits
	GI261-GI292	Counters CV, integer
	GI293-GI324	Counters PV, integer
	GI325-GI356	Counters PRESET, integer

Table 7-17: Counter Global Integers (GIx)

7.8 I/O Mapper in Project Mode

Offline programming does not require communication with the control. Online programming requires communication with the control.

Use the following steps to create an I/O Mapper.

1. From VisualMotion Toolkit's main menu, select **Commission** ⇒ **I/O Mapper**.
2. From the Ladder icon toolbar, click on the desired icon and move the cursor to the Ladder window.
3. Place contacts to the left of the Ladder window and coils or functions to the right. This symbol (⊗) appears when attempting to place an icon in the wrong location.
4. From the Properties window, choose the contact type under the Contact Setup tab. Refer to Input Logic Functions on page 7-13 for details.
5. Save the I/O Mapper.

Switching to Online Mode

To synchronize an I/O Mapper file commissioned offline, open the project containing the I/O Mapper and switch the control to online mode (**File** ⇒ **Online**). VisualMotion detects any changes to the current project on the control (if downloaded previously) and informs the user of the components in the project that have changed. The user can accept the changes and download them to the control or establish a connection to the control without downloading any data by selecting "Go online Unsynchronized".

Note: The system must in parameter mode before downloading the I/O Mapper to the control.

Importing an I/O Mapper into a Project

An I/O Mapper downloaded to the control can be imported into a new project while in online mode, or from another project or file when in offline mode.

Use the following steps to import an I/O Mapper from data stored on the control.

1. Start VisualMotion Toolkit and open the target project.
2. Switch VisualMotion Toolkit to online mode.
3. Switch the control to parameter mode.

Note: To transfer control data, previously downloaded to the control, switch VisualMotion to online mode.
To import data from another offline project or file, switch VisualMotion to offline mode.

4. From VisualMotion Toolkit's main menu, select **File** ⇒ **Import Project Component**.
5. From the "Transfer Control Data to Project" window, select the I/O Mapper checkbox. By default, both control parameters C-0-3000 and C-0-3001 are checked. To transfer only the I/O Mapper, uncheck control parameter C-0-3001.

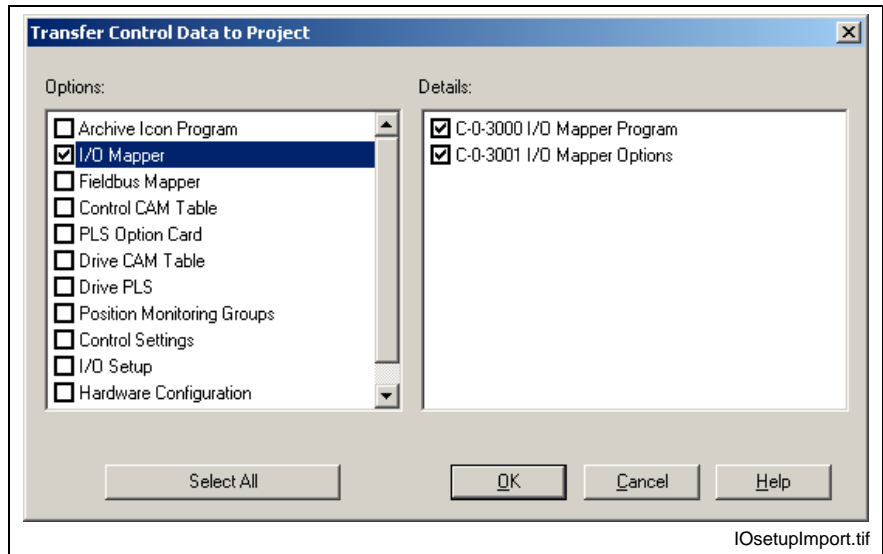




Fig. 7-19: I/O Mapper Transfer from Control

Note: When importing data from another project or file in offline mode, the data is not synchronized with the project until VisualMotion is switched to online mode and the data is downloaded to the control.

7.9 I/O Mapper in Service Mode

Service mode allows the user to make modifications to an I/O Mapper stored in the control's memory. I/O Mapper files can also be opened, modified and downloaded to the control. A serial or Ethernet connection, with active communication, is required before proceeding with the following steps.

1. Start VisualMotion Toolkit and select the "Service" mode radio button.
2. Select **Commission** ⇒ **I/O Mapper**.
3. Switch the control to parameter mode.
4. Upload the I/O Mapper by selecting **File** ⇒ **Get Ladder from control** or click the upload icon ().
5. Make the necessary modifications to the I/O Mapper and save the file. Download the modifications to the control by clicking the download icon ().

Note: I/O Mapper configurations that are downloaded to the control or saved to a file in service mode are not synchronized with a project's offline data. It is the responsibility of the project manager to ensure that I/O Mapper configurations modified in service mode are imported into the appropriate VisualMotion project.

7.10 Import Default I/O Mapper

During the installation of VisualMotion, a default I/O Mapper file (*Def100.iom*) is installed under the "Param" subfolder. The default I/O Mapper file is intended as an example and in support of the IoBox Windows based HMI software.

Default I/O Mapper in Project Mode (Offline)

Use the following steps to import the default I/O Mapper in to the current project.

1. From VisualMotion Toolkit's main menu, select **File** ⇒ **Import Project Component** while in offline mode.
2. Switch the control to parameter mode.
3. From the *Import a program and/or data into the project* window, select **File** from the *Transfer data from* section.
4. Click on the browse button, change the *Files of type:* to **Old I/O Mapper Files (*.iom)** and locate the *Def100.iom* file.
5. From the *Components:* section, place a check next to **I/O Mapper** and click on the **OK** button.

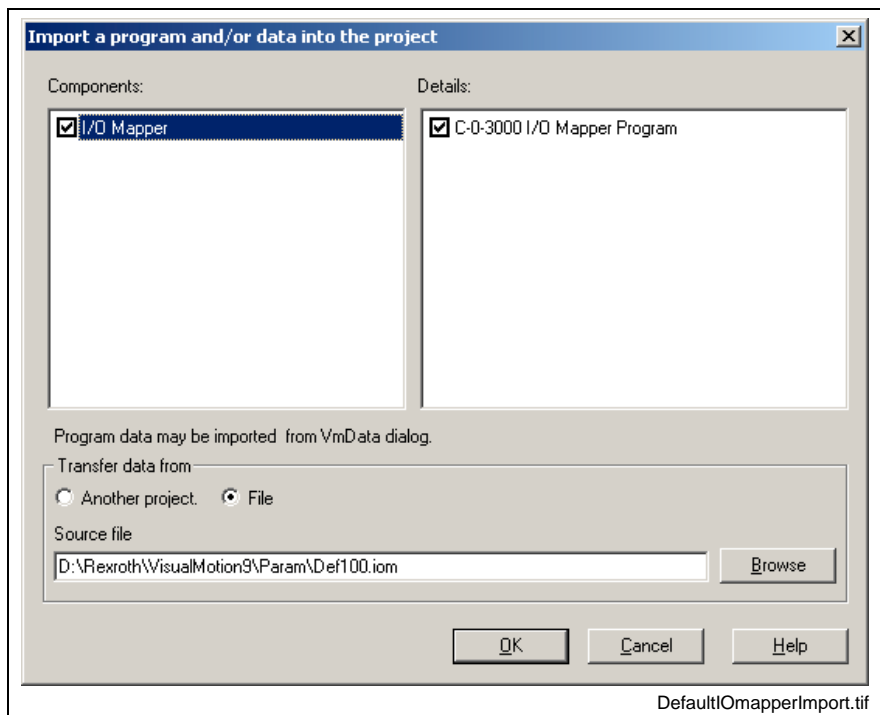



Fig. 7-20: Import Default I/O Mapper

6. Switch the project to online mode by selecting **File** ⇒ **Online** from VisualMotion Toolkit's main menu.

Default I/O Mapper in Service Mode

Follow these steps to open and download the default I/O Mapper to the control in Service mode.

1. Open the I/O Mapper and select **File** ⇒ **Open**, locate the *Def100.iom* file and click on **O**pen.
2. Switch the control to parameter mode.
3. To download the I/O Mapper to the control, select **File** ⇒ **Send Ladder to control** or click on the download icon ().

8 Programmable Limit Switch Functionality

8.1 PLS Description

A Programmable Limit Switch is used to switch on and off digital outputs based on the input position of an associated axis or master. The basic component of a PLS will be referred to as a PLS object.

PLS Object

A PLS object is a process that receives an input position from one associated axis or master and switches on and off digital outputs based on the programmed on and off positions. The following graphic illustrates a basic PLS object.

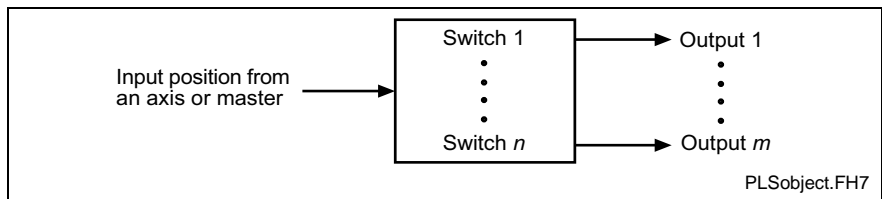


Fig. 8-1: Basic PLS Object

VisualMotion 9 supports three PLS types (Control, Drive and Option Card PLS). The number of supported PLS objects by type are:

- 2 for Control PLS
- 1 for each Drive PLS
- Up to 8 for Option Card PLS

A Drive PLS is considered as one PLS object. A Control PLS can have a maximum of 2 PLS objects in a VisualMotion user program. For example, a PLS icon in a user program is considered as one PLS object. On the other hand, an Option Card PLS can have up to 8 PLS objects.

The number of available switches and outputs varies on the PLS type. The following table describes the three PLS types supported in VisualMotion 9.

PLS Type	Description	Number of Outputs	Number of Switches
Control PLS	Maximum of two PLS objects per VisualMotion user program	16 per PLS object	1 switch per output
Drive PLS	One PLS object per drive	Number of outputs is based on drive firmware 8 (DIAX04) 16 (ECODRIVE03)	1 switch per output
Option Card PLS	Optional high-speed PLS card for PPC-R control User definable PLS objects 1-8	16/32 outputs (16 outputs per output module)	96 switches for 16/32 outputs a maximum of 96 switches can be assigned to 16/32 outputs

Table 8-1: PLS Types

Control PLS

A *Control PLS* data structure exists in every VisualMotion user program. A VisualMotion user program can have a maximum of 2 Control PLS objects.

Control PLS Specifications

A Control PLS has the following specifications:

- 16 switches assigned to one 16 bit VisualMotion register (one bit per output)
- 16 Outputs updated every SERCOS scan time
- One individual lead time for each switch
- One PLS input type from the following:
 - ELS Master (1-6 System Masters),
 - ELS Group (1-8) or
 - Drive (1-40)

Control PLS in GPP8 Firmware

When accessing a VisualMotion control using GPP8 firmware, Control PLS settings can be configured from the PLS tool or by placing a Control PLS icon in the user program. Control PLS data is only stored with the compiled user program in the control's memory. The active user program must contain a configured Control PLS, otherwise, only Drive and Option Card PLSs, if configured, will be displayed in the PLS tool.

Note: Control PLSs settings which were created using the PLS icon (compiled and downloaded) can be uploaded into the PLS tool. However, any modifications to Control PLS settings, using the PLS tool, will not be reflected in the PLS icon of the source program.

Communication Protocol for Control PLS Data

Unlike a Drive and Option Card PLS that use parameters for data storage, a Control PLS is stored as data in the compiled VisualMotion user program. The VisualMotion DDE server (ASCII protocol) or the SCP server (SIS protocol) can be used to access Control PLS data.

The following tables list each instruction with a brief description.

ASCII Protocol Instructions

ASCII Protocol	Description
WH n.X.m	Control PLS Switch Data
WR X.1	Control PLS Output Register
WM X.1	Control PLS Mask Register
WO X.1	Control PLS Master Phase Offset
WT X.1	Control PLS Master Type
WA X.1	Control PLS Master Number

Table 8-2: ASCII Protocol Instructions for a Control PLS

SIS Protocol Instructions

The table below indicates the mapping from the ASCII protocol to the corresponding fields in the SIS protocol "User Data Header." The class and subclass values are mapped directly from the ASCII protocol.

SIS Protocol		Description
Class	Subclass	
0x57	0x52	Control PLS Output Register
	0x4D	Control PLS Mask Register
	0x4F	Control PLS Master Phase Offset
	0x54	Control PLS Master Type
	0x41	Control PLS Master Number
	0x48	List of On, Off, and Lead Time Values
	0x45	List of On Values
	0x46	List of Off Values
	0x47	List of Lead Time Values

Table 8-3: SIS Class and Subclass for a Control PLS

Drive PLS

Drive PLS parameter support is dependent on the version of drive firmware installed on the drive. Drive PLS parameters can also be written to within the icon program and activated during runtime.

DIAX04 Drive PLS Specifications

Note: DIAX04 supported firmware version is as follows:

- FWA-DIAX04-ELS-05VRS-MS

Each DIAX04 digital drive can have one Drive PLS object. A DIAX04 Drive PLS has the following specifications:

- 8 switches assigned to one 16 bit VisualMotion register (one bit per output for bits 1-8)
- 8 Outputs
- An individual lead time for each switch
- One PLS input type from the following:
 - Motor encoder (S-0-0051) or
 - External encoder (S-0-0053)

Drive based I/O (i.e., DEA4.2M card) can be configured for PLS output. Refer to Configuring Drive based I/O Cards for PLS Output on page 8-17 for details.

ECODRIVE03 Drive PLS Specifications

Note: ECODRIVE03 supported firmware versions are as follows:

- FWA-ECODR3-SGP-03VRS-MS
- FWA-ECODR3-SMT-02VRS-MS

Each ECODRIVE03 digital drive can have one Drive PLS object. An ECODRIVE03 PLS has the following specifications:

- 16 switches assigned to one 16 bit VisualMotion register (one bit per output)
- 16 Outputs
- An individual lead time for each switch
- One PLS input type from the following:
 - Motor encoder (S-0-0051) or
 - External encoder (S-0-0053)

Parameters related to a Drive PLS

The following parameters are valid for both DIAX04 and ECODRIVE03 drive based PLSs. The following tables will list each parameter with a brief description. Refer to the respective Digital Drive Functional Description manual for details.

Drive Parameters

Parameter	Description	Updated
P-0-0131	Signal Select Position Switch	Phase 4
P-0-0132	Switch On Threshold Position Switch	Phase 4
P-0-0133	Switch Off Threshold Position Switch	Phase 4
P-0-0134	Position Switch Lead Time	Phase 4
P-0-0135	Status Position Switch	Phase 4

Control Parameter

A-0-0009	Output Register Value	Phase 4
----------	-----------------------	---------

Table 8-4: Drive PLS Parameters

Note: Refer to the respective Digital Drive Functional Description manual for details about the drive PLS feature.

Option Card PLS

The *Option Card PLS* is an option card for the PPC-R and PPC-P11.1 that can be used with one or two PLS output modules (NSW01.1R). Each module has 16 digital outputs. A PPC-R can be ordered configured with an Option Card PLS and either 16 or 32 digital outputs for programmable limit switches.

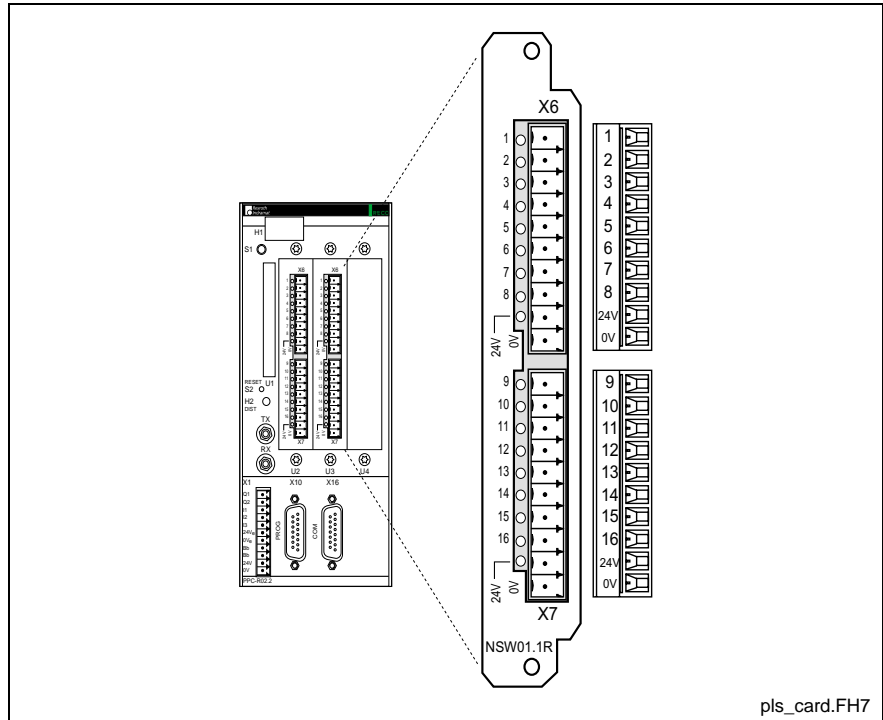


Fig. 8-2: NSW01.1R Option Card PLS

Option Card PLS Specifications

- Up to 32 digital outputs with a high-speed update rate of 250 μ s at SERCOS cycle time from 2 ms to 16 ms.
- Up to 96 limit switches, for all 16/32 outputs
 - Any combination of limit switches can be configured to the outputs.
- Up to 8 input types from the following:
 - ELS Masters (1-6 System Masters),
 - ELS Groups (1-8) or
 - Drives (1-40)

Note: The maximum speed for any Option Card PLS input master is 3500 rpm.

Parameters related to the Option Card PLS

The parameters used for an Option Card PLS can be grouped into four categories:

- General Parameters (C-0-2901 through C-0-2910)
- Switch Settings (C-0-2920 through C-0-2922)
- Output Settings (C-0-2930 through C-0-2935)
- PLS Input Settings (C-0-2940 through C-0-2943)

The following tables will list each parameter with a brief description.

General Parameters

Parameter	Description	Updated
C-0-2901	PLS1 Start Output Register	Phase 4 (read/write)
C-0-2902	PLS1 Start Mask Register	Phase 4 (read/write)
C-0-2903	PLS1 Build Table Command	Phase 4 (read/write)
C-0-2904	PLS1 Build Table Status	Phase 4 (read)
C-0-2905	PLS1 Switch Table Command	Phase 4 (read/write)
C-0-2906	PLS1 Switch Table Status	Phase 4 (read)
C-0-2907	PLS1 Error Code	Phase 2 (read) / Phase 4
C-0-2908	PLS1 Extended Error Code	Phase 2 (read) / Phase 4
C-0-2909	PLS1 Hardware ID	Phase 2 (read) during power up
C-0-2910	PLS1 Software ID	Phase 2 (read) during power up

Table 8-5: General Parameters for an Option Card PLS

Switch Parameters

Parameter	Description	Updated
C-0-2920	PLS1 Switch On List	Phase 4 (read/write)
C-0-2921	PLS1 Switch Off List	Phase 4 (read/write)
C-0-2922	PLS1 Switch Output List	Phase 4 (read/write)

Table 8-6: Switch Parameters for an Option Card PLS

Output Parameters

Parameter	Description	Updated
C-0-2930	PLS1 Output Master List	Phase 2 (read/write)
C-0-2931	PLS1 Output Lead Time	Phase 4 (read/write)
C-0-2932	PLS1 Output Lag Time	Phase 4 (read/write)
C-0-2933	PLS1 Output One Shot (PT) List	Phase 4 (read/write)
C-0-2934	PLS1 Output Mode List	Phase 4 (read/write)
C-0-2935	PLS1 Output Direction List	Phase 4 (read/write)
C-0-2936	PLS1 Output Hysteresis	Phase 2 (read/write)

Table 8-7: Output Parameters for an Option Card PLS

PLS Master Parameters

Parameter	Description	Updated
C-0-2940	PLS1 Master Type List	Phase 2 (read/write)
C-0-2941	PLS1 Master Number List	Phase 2 (read/write)
C-0-2942	PLS1 Master encoder List	Phase 2 (read/write)
C-0-2943	PLS1 Master Phase Offset List	Phase 4 (read/write)

Table 8-8: PLS Master Parameters for an Option Card PLS

VisualMotion PLS Tool

The PLS tool is a Windows™ based editor used to program, monitor and document Control, Drive and Option Card programmable limit switches. PLS configurations can be programmed or edited in project or service mode. Online editing of PLS configuration is supported. Refer to Synchronize Project Components on page 2-16 for details.

To launch the PLS tool, select **Commission** ⇒ **PLS** from VisualMotion's main menu.

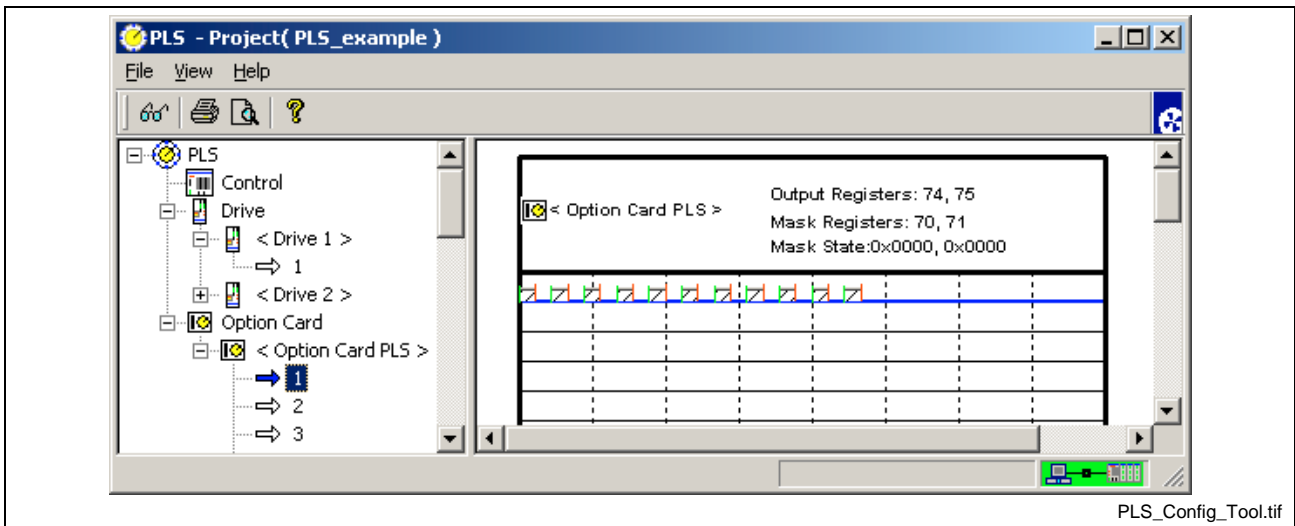


Fig. 8-3: PLS Configuration Tool

PLS Tool Communication Modes

Project Mode

If a PLS configuration is being parameterized for a new project, the PLS tool opens displaying only the basic icon structure, as illustrated in Fig. 8-4. Otherwise, all configured Control, Drive and Option Card PLSs will be read and displayed.

Offline Editing

When a PLS is configured or edited in offline mode, the PLS data is read and saved to the project's offline data. Offline mode does not require the user to be physically connected to the control.

Online Editing

Online editing allows the user to edit configured PLS data while synchronized with the control. When the PLS tool is opened, any configured PLS data is read from the control and displayed in the PLS tool window. To edit a PLS in online mode, refer to Edit PLS Configuration in Project Mode on page 8-31 for details.

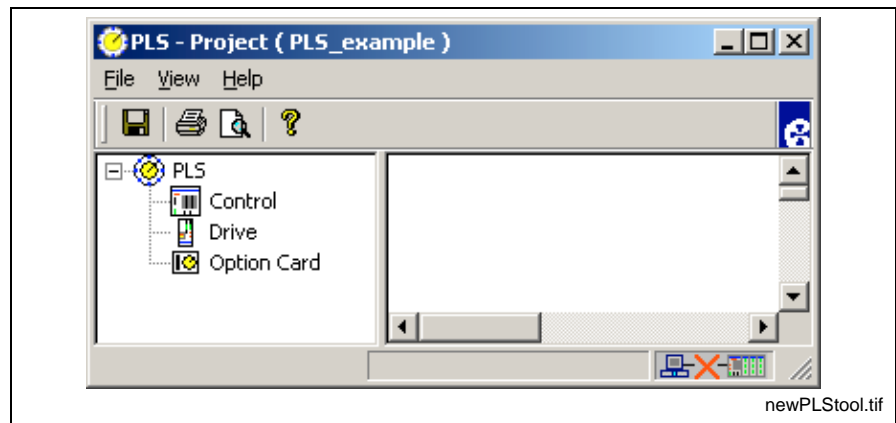


Fig. 8-4: PLS Tool in Project Mode

Service Mode

Although service mode can be used to parameterize a PLS configuration and download it to the control, service mode should be used to make any necessary modifications when the project data files are not available.

PLS configurations are uploaded from the control by selecting the upload icon or by selecting **File** ⇒ **Get PLS Configuration from PPC Control**. To edit a PLS in service mode, refer to Edit PLS Configuration in Service Mode on page 8-34 for details.

Note: PLS configurations edited in service mode will become unsynchronized with a project unless the PLS data is imported in to the project. Refer to Importing a PLS Configuration into a Project on page 8-33 for details.

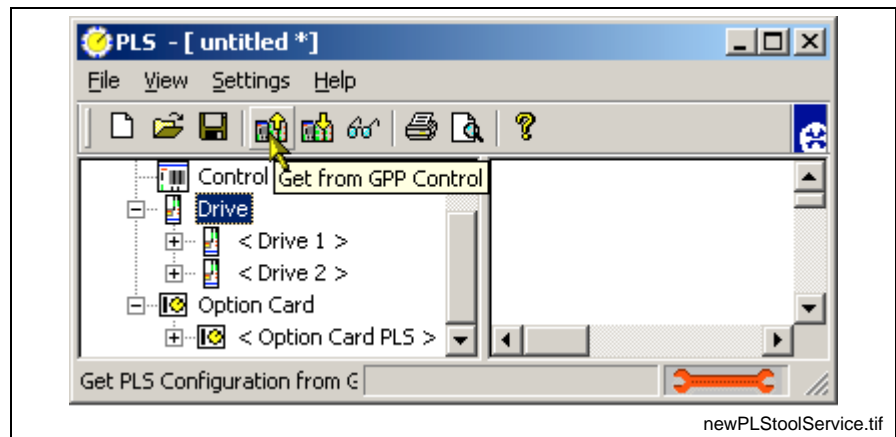


Fig. 8-5: PLS Tool in Service Mode

8.2 Configure a Control PLS

Use the following steps to configure a Control PLS in offline project mode:

1. Open the PLS Configuration tool by selecting **Commission** ⇒ **PLS...** from VisualMotion's main menu.
2. Right click the Control icon and select **Add Control PLS...** Select **Control** from the drop-down list and select **Control # 1** or 2.

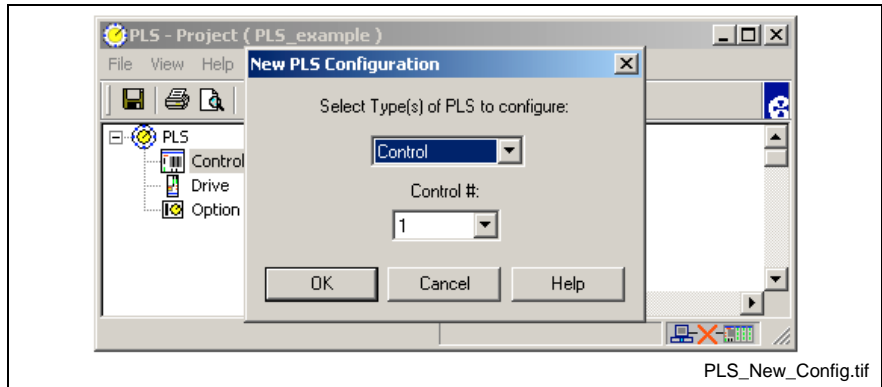


Fig. 8-6: New PLS Configuration

3. From the *PLS Switches* window in Fig. 8-7. Assign a switch's On/Off position and lead time by double clicking on the desired switch number or highlight the switch number and press the **Edit** button.

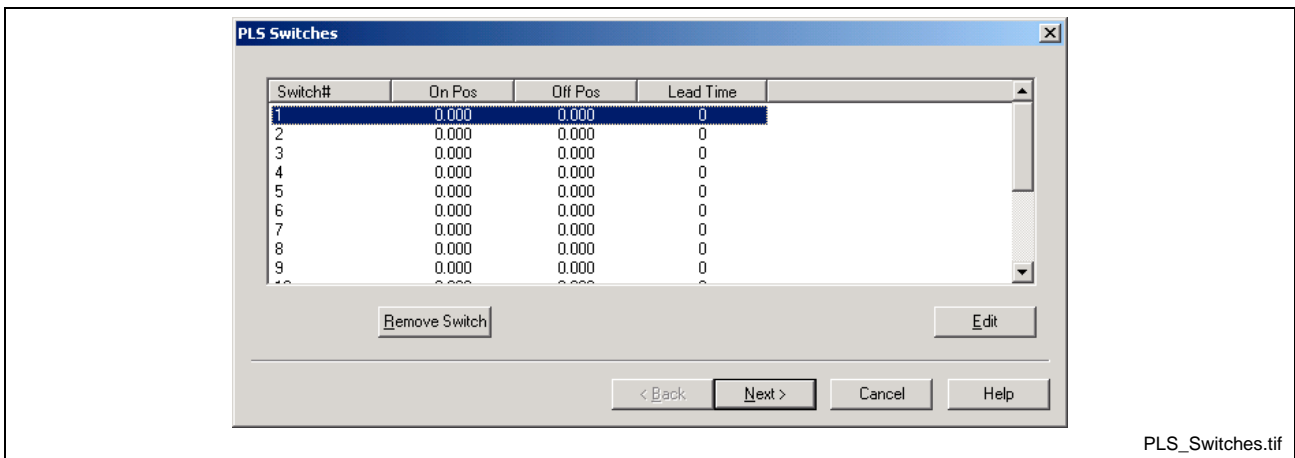


Fig. 8-7: PLS Switches Window

Switch Configuration for a Control PLS

The following steps configure a switch's On / Off position and lead time.

1. Select the Switch Index number.

Note: Once a switch is configured and the **Apply** button is pressed, a new Switch Index number must be selected to configure a different switch. Pressing the **OK** button accepts the values and closes the window. The current switch number is displayed in the header portion of the window.

Example: Switch 1 Configuration

2. Enter the On and Off Positions and Lead Time.

Note: The On and Off positions for each switch is relative to the units of measurement in axis parameter A-0-0005 and task parameter T-0-0005.

Example: If these parameters are set to 1 (mm), then the unit of measurement for the switch's On and Off positions is mm.

Note: A Control PLS can have a maximum of 16 switches configured to a VisualMotion register.

3. When all the switches are configured, press the **OK** button to close the *Switch # Configuration* window and return to the *PLS Switches* window.
4. Press the **Next >** button to continue to the *PLS Master Configuration* window.

VisualMotion Error for Switch Configuration

VisualMotion issues the following error when a different Switch Index number is selected without first applying any changes to an already configured switch.

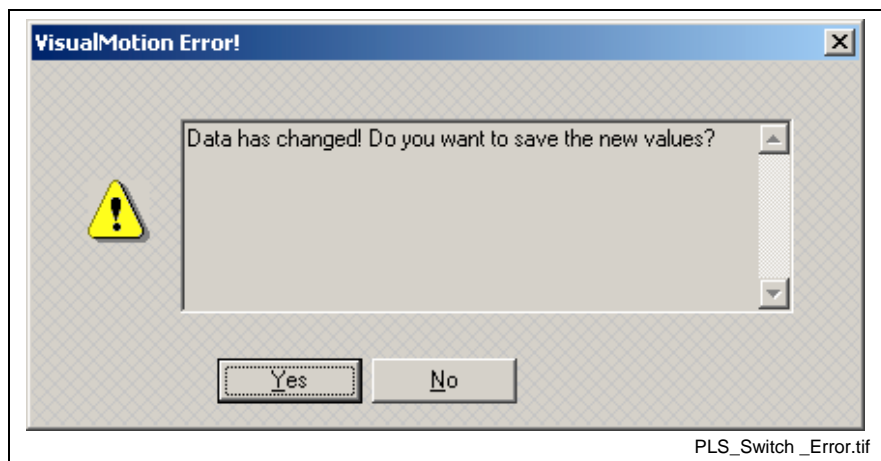


Fig. 8-8: PLS Switch Configuration Error Message

Example:

Switch 1 is configured and the **Apply** button is pressed. Now, the user makes a change to an On/Off position or the Output and then selects a different Switch Index number. VisualMotion assumes that you want to continue without first applying any changes to the current switch.

PLS Master Configuration for a Control PLS

The *PLS Master* window is used to configure 1 master for each Control PLS. The available PLS types are:

- ELS Master (System Master)
- ELS Group
- Drive (DIAX04 and ECODRIVE03)

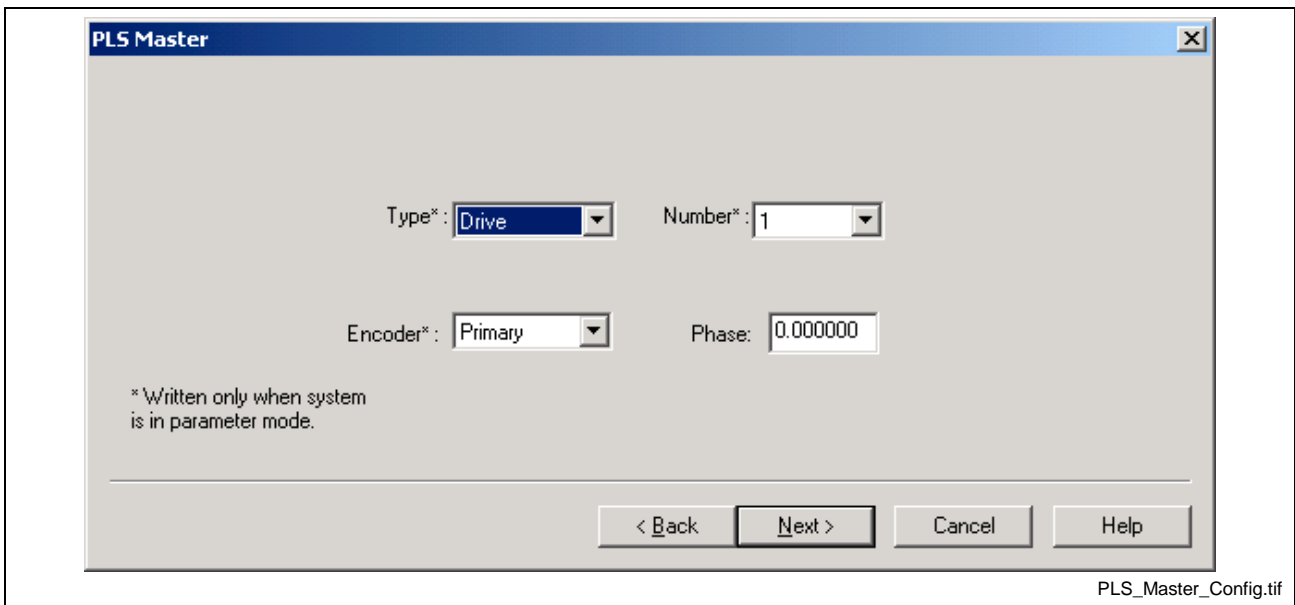


Fig. 8-9: PLS Master Configuration for a Control PLS

1. Select a PLS *Type*:
 - ELS Master
 - ELS Group
 - Drive
2. Select a Number from the drop down list corresponding to the input type selected.

Type	Allowable Number Range	Encoder
ELS Master	1-6	N/A
ELS Group	1-8	N/A
Drive	1-40 (SERCOS address)	Primary or Secondary

Note: When a Drive is selected, the allowable number range is 1-40 for up to 40 axes. Then the user must specify the *Encoder* for each axis as either the primary feedback or a secondary feedback device.

3. Enter a Phase offset if applicable.
The *Phase* field is an offset that is added to the output of the PLS master.

Note: ELS Master

When selected as a PLS Master type, the ELS Master's position value is used as an input to the PLS. In GPP 9, an ELS Master can be a Virtual Master, the output of an ELS Group, a Real Master or the output of a second PPC-R cross-communicating in a Link Ring.

ELS Group

In an ELS multiple master configuration, the output (position) of an ELS Group can be used as an input to the PLS. If a *Phase* value is added to the PLS Master Configuration window in Fig. 8-9, then the value is added to the output of the ELS Group and does not affect any phase offsets that might have been configured in the ELS Group.

- When the PLS Masters is configured, press the **Next >** button to continue to the *PLS Register Assignment* window.

PLS Register Assignment for a Control PLS

The *PLS Register Assignment* window is used to assign a VisualMotion register to each Control PLS. The user can also set the range of motion of the graphical representation of the PLS switches. The following steps assign a register to the Control PLS and sets the graphical limits of the PLS main window.

- Accept the default register or set a different *Output Register*.
- Accept the default register or set a different *Mask Register*.

Note: During the initial setup, the Mask register bits cannot be set to (1). After the PLS is configured and downloaded to the control, the user must manually set the state of each Mask Register bit to (1) by selecting **Data ⇒ Registers** from VisualMotion main menu. Only the Output Register whose Mask Register is set to (1) will output a signal in the Control PLS's register.

- Set the *Graphs Limit's* minimum and maximum value.
- Press the **Finish** button to end the configuration wizard.

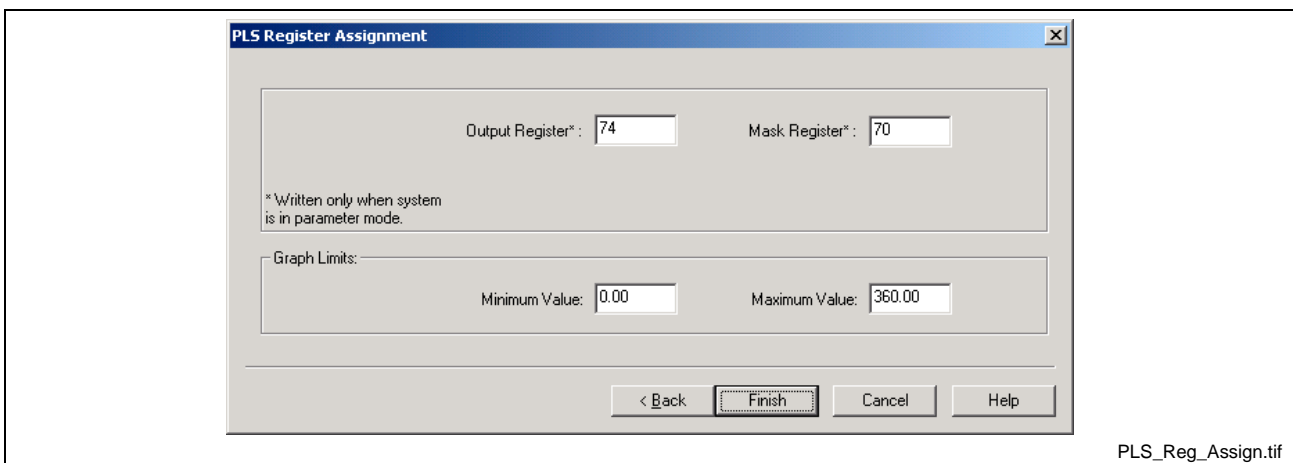


Fig. 8-10: PLS Register Assignment for a Control PLS

Output Register

The Output register is assigned to the Control PLS for monitoring the status of each switch. The 16 Control PLS switches are assigned to bits 01-16, respectively.

Mask Registers

The Mask register is used to force the state of the output register bits. At power-up, the bits in the mask register are set to 0 to prevent any unwanted outputs from enabling. The Control PLS switches assigned to a output register will not function until the state of the register's complementary mask register bits are set to (1).

Note: The default state of the Output Register is zero (0). The user must set the corresponding mask register bits each time after power-up.

Assigning Mask Register Numbers

Mask register numbers can only be assigned to the first Control PLS in a project. If a second Control PLS is configured, the mask register number will be automatically assigned one consecutive number greater than the first Control PLS.

Example:

Mask register for Control PLS 1 is set to 80.

Mask register for Control PLS 2, if configured, will be set to 81.

Note: Future edits of the mask register number for Control PLS 1 will automatically change the mask register number of Control PLS 2.

Graph Limits

Set the minimum and maximum values for the graphical representation of all Control PLS switches.

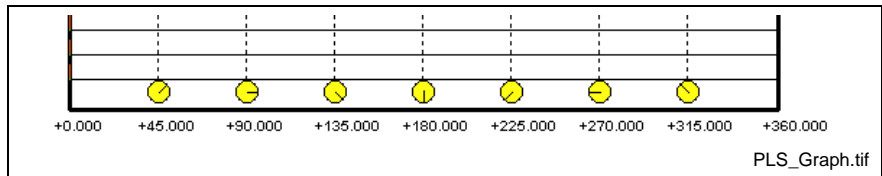


Fig. 8-11: Graph Limits

Example:

For rotary applications, the user can set the limits from 0.00 to 360.00 degrees.

For linear applications, the user can set the limits to match the actual range.

Example: -100.00 to 100.00

8.3 Configure a Drive PLS

Use the following steps to configure a Drive PLS in offline project mode:

1. Open the PLS Configuration tool by selecting **Commission** ⇒ **PLS...** from VisualMotion's main menu.
2. Right click the Drive icon and select **Add Drive PLS...** Select **DIAX04** or **ECODRIVE** from the drop-down list and select a **Drive #** from 1 to 40.

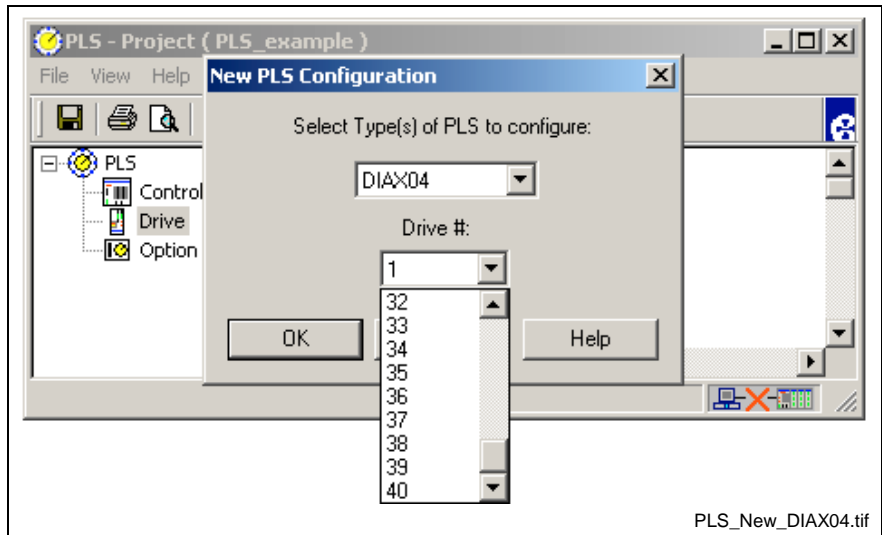


Fig. 8-12: New PLS Configuration

3. From the *PLS Switches* window in Fig. 8-13. Assign a switch's On/Off position and lead time by double clicking on the desired switch number or highlight the switch number and press **Edit**.

Note: DIAX04 drives will display 8 switches and ECODRIVE03 drives will display 16 switches. Any number of switches can be removed from the configuration by selecting the switch and pressing the **Remove Switch** button.

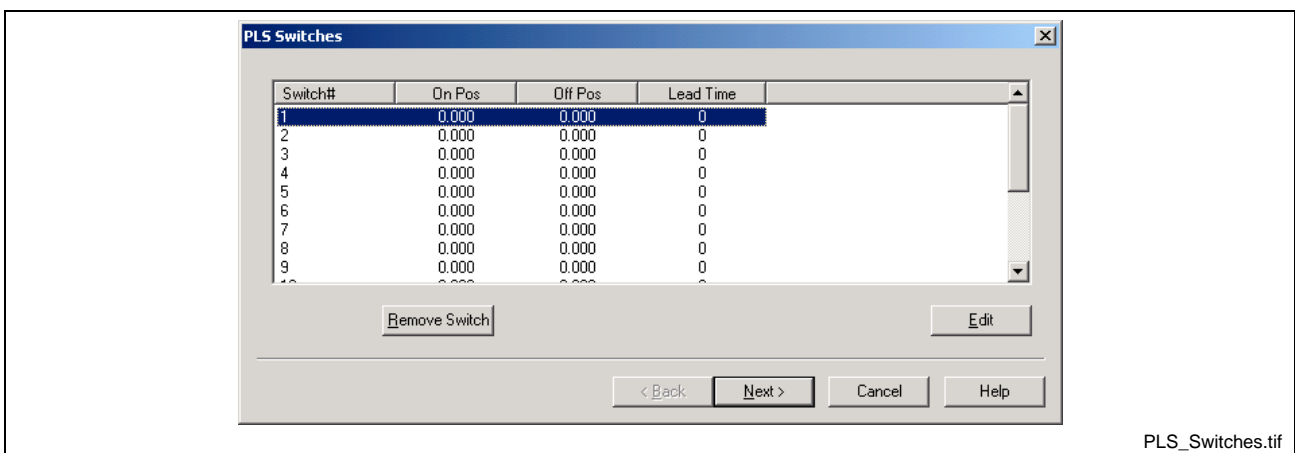


Fig. 8-13: PLS Switches Window

Switch Configuration for a Drive PLS

The following steps configure a switch's On / Off position and lead time.

1. Select the Switch Index number.

Note: Once a switch is configured and the **Apply** button is pressed, a new Switch Index number must be selected to configure a different switch. Pressing the **OK** button accepts the values and closes the window. The current switch number is displayed in the header portion of the window.

Example: Switch 1 Configuration

2. Enter the On and Off Positions and Lead Time.

Note: The On and Off positions for each switch is relative to the units of measurement in axis parameter A-0-0005 and task parameter T-0-0005.

Example: If these parameters are set to 1 (mm), then the unit of measurement for the switch's On and Off positions is mm.

Note: A Drive PLS can have a maximum of 16 switches configured to a VisualMotion register.

3. When all the switches are configured, press the **OK** button to close the *Switch # Configuration* window and return to the *PLS Switches* window.
4. Press the **Next >** button to continue to the *PLS Master Configuration* window.

VisualMotion Error for Switch Configuration

VisualMotion issues the following error when a different Switch Index number is selected without first applying any changes to an already configured switch.

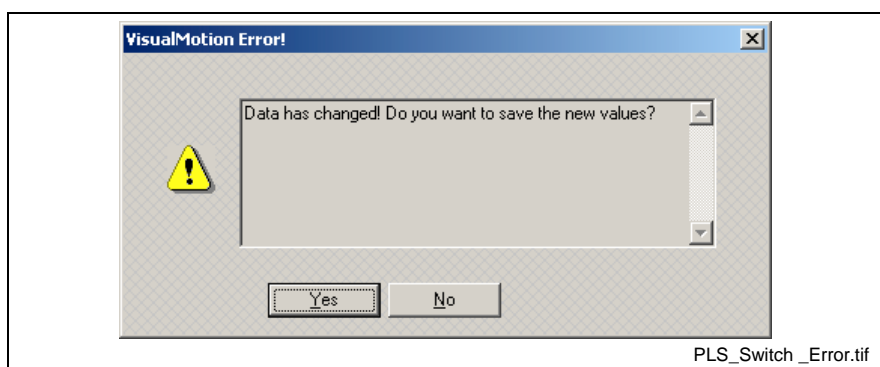


Fig. 8-14: PLS Switch Error Message

Example:

Switch 1 is configured and the **Apply** button is pressed. Now, the user makes a change to an On/Off position or the Output and then selects a different Switch Index number. VisualMotion assumes that you want to continue without first applying any changes to the current switch.

PLS Master Configuration for a Drive PLS

The *PLS Master* window is used to configure the drive's Encoder type. The available Encoder types to use as a PLS Master are:

- Primary
- Secondary

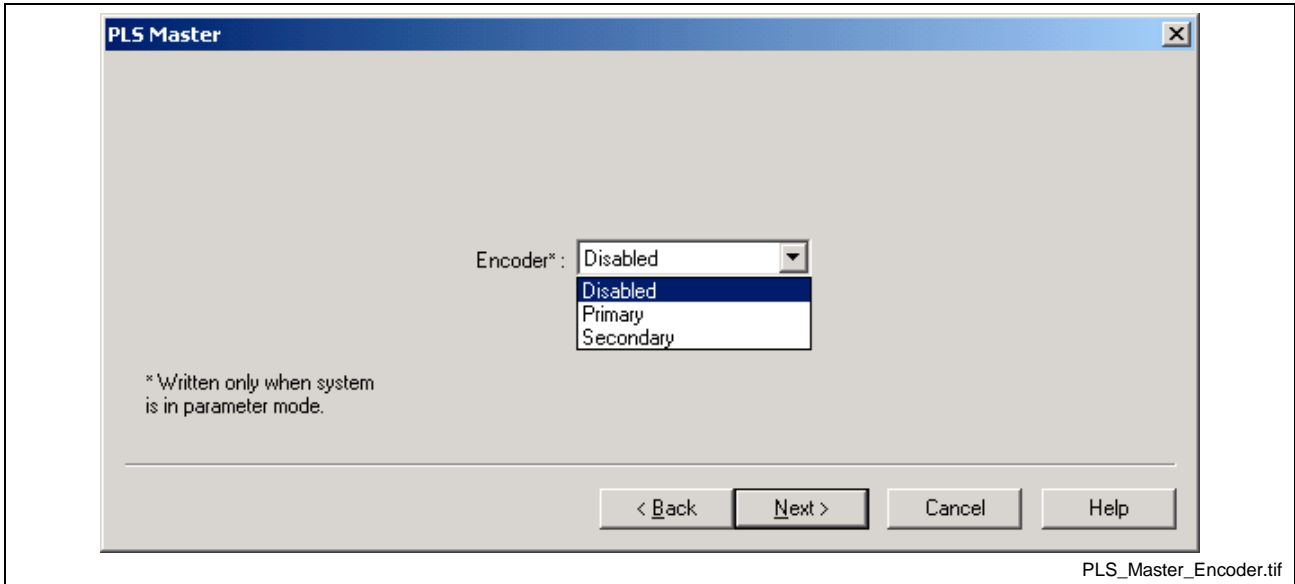


Fig. 8-15: PLS Master Configuration for a Control PLS

When the PLS Masters is configured, press the **Next >** button to continue to the *PLS Register Assignment* window.

PLS Register Assignment for a Drive PLS

The *PLS Register Assignment* window is used to assign a VisualMotion register to each Drive PLS. The user can also set the range of motion of the graphical representation of the PLS switches. The following steps assign a register to the Drive PLS and sets the graphical limits of the PLS main window.

1. Accept the default register or set a different *Output Register*.
2. Set the *Graphs Limit's* minimum and maximum value.
3. Press the **Finish** button to end the configuration wizard.

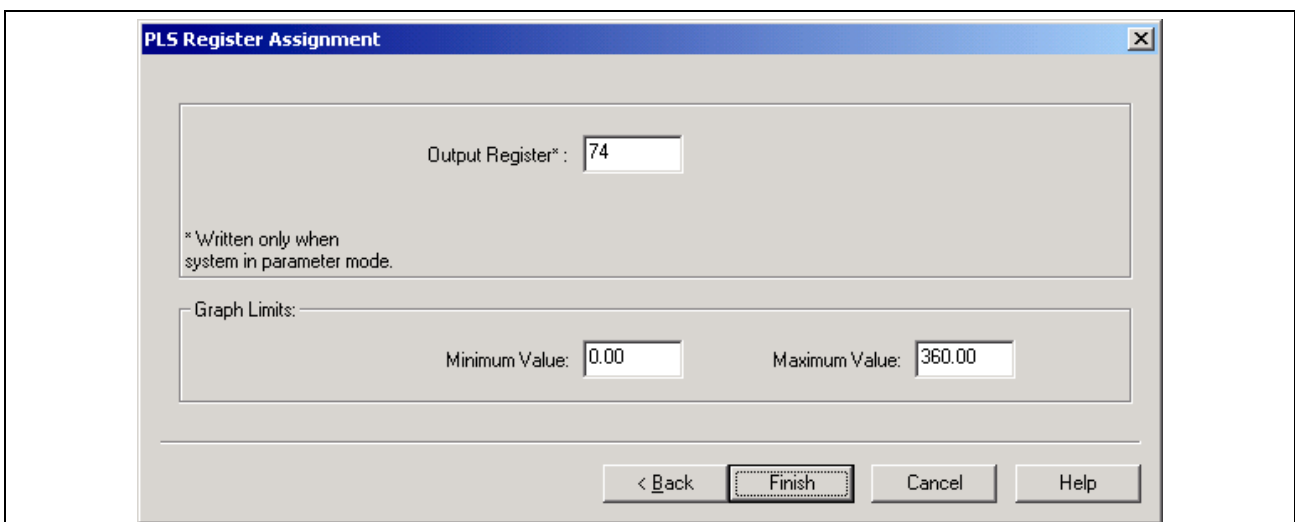


Fig. 8-16: PLS Register Assignment for a Control PLS

Output Register

The Output register is assigned to the Drive PLS for monitoring the status of each switch. A DIAX04 drive's switches are assigned to bits 01-08, respectively. An ECODRIVE03 drive's switches are assigned to bits 01-16, respectively.

Note: No Mask Register is required for a Drive PLS. The Drive PLS output register is always active.

Graph Limits

Set the minimum and maximum values for the graphical representation of all Control PLS switches.

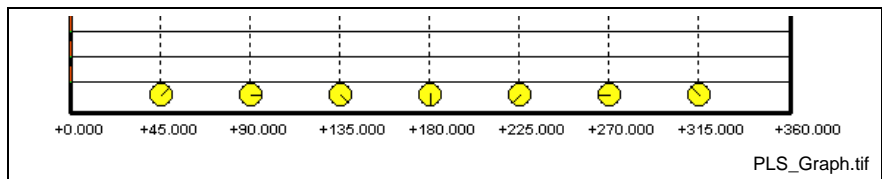


Fig. 8-17: Graph Limits

Example:

For rotary applications, the user can set the limits from 0.00 to 360.00 degrees.

For linear applications, the user can set the limits to match the actual range.

Example: -100.00 to 100.00

Configuring Drive based I/O Cards for PLS Output

VisualMotion can assist the user in automatically configuring a DIAX04 Drive PLS's output signal directly to a DEA I/O card. The following data is required by VisualMotion.

- Output register for Drive PLS

The supported DIAX04 digital drive I/O cards that can be used to output Drive PLS signals are:

- DEA04.2M
- DEA05.2M
- DEA06.2M

To have VisualMotion automatically configure these I/O cards to output Drive PLS signals, follow these steps.

1. Using the PLS Tool, configure a DIAX04 Drive PLS for axis (n) and assign a VisualMotion register (for example, register 70) for the PLS *Output Register*. Refer to PLS Register Assignment for a Drive PLS on page 8-16.
2. Downloaded the configured PLS to the control in parameter mode.

Note: The PLS Output Register assignment is written to axis parameter A-0-0009.

3. Using the I/O Configuration Tool, configure axis (n) with a DEA04.2M, 5.2M or 6.2M I/O card and assign the outputs to the same register number that was used in the PLS tool.
4. Download the I/O configuration to the control in parameter mode.

How it works

During phase initialization (P2 to P3), VisualMotion reads and compares the register numbers assigned to the DIAX04 Drive PLS and the DEA4.2M, 5.2M or 6.2M I/O card. If the register numbers match (in this example both are 70) then drive parameter P-0-0124 is automatically configured with the IDN number of P-0-0135 and the type of I/O card configured.

During runtime (P4), the Drive PLS outputs (P-0-0135) are sent directly to the physical outputs on the drive's DEA4.2M, 5.2M or 6.2M I/O cards. At the same time, the Drive PLS outputs (P-0-0135) are also written to register 70, across the SERCOS AT Cyclic Telegram, for use in the VisualMotion user program. The following figure shows the runtime sequence of the configured Drive PLS outputs.

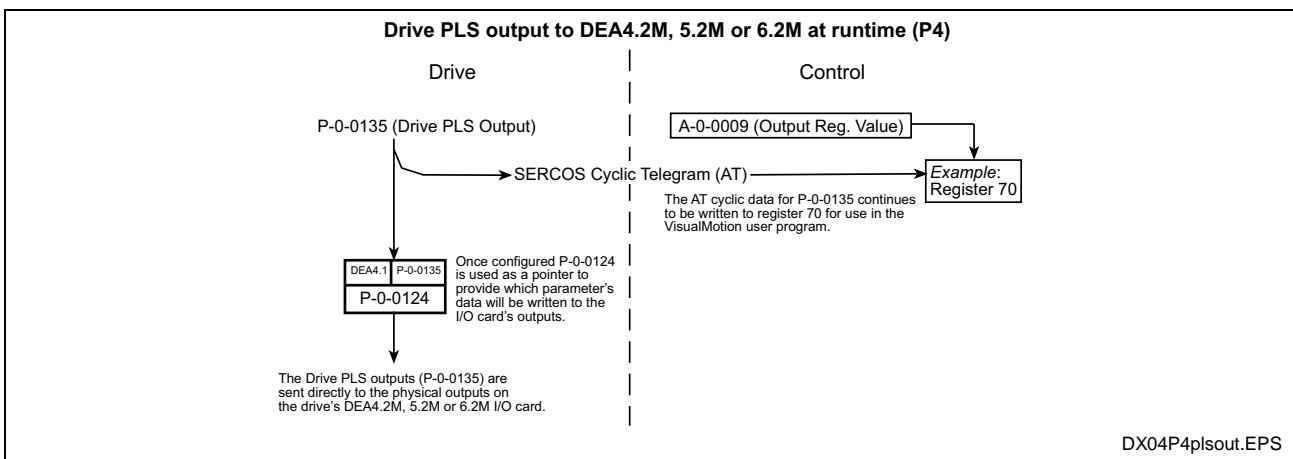


Fig. 8-18: DEA4 I/O Configured for Drive PLS Output

8.4 Configure an Option Card PLS

Use the following steps to configure a Option Card PLS in offline project mode:

1. Open the PLS Configuration tool by selecting **Commission** ⇒ **PLS...** from VisualMotion's main menu.
2. Right click the Option Card icon and select **Add Option Card PLS...** Select **Option Card** from the drop-down list and click on the OK button.

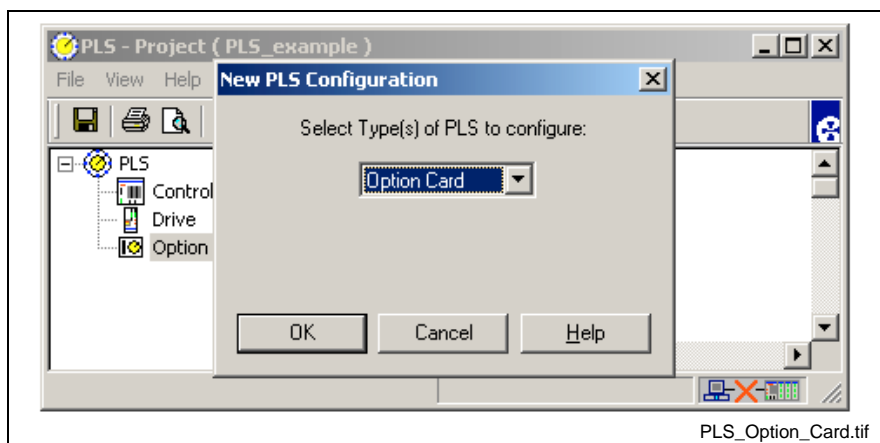


Fig. 8-19: New PLS Configuration

- After the type is selected, the PLS Configuration wizard will guide the user through the initial setup beginning with the *PLS Switches* window in Fig. 8-20. Assign a switch's On/Off position to an output by double clicking on the desired switch number or highlight the switch number and press the **Edit** button.

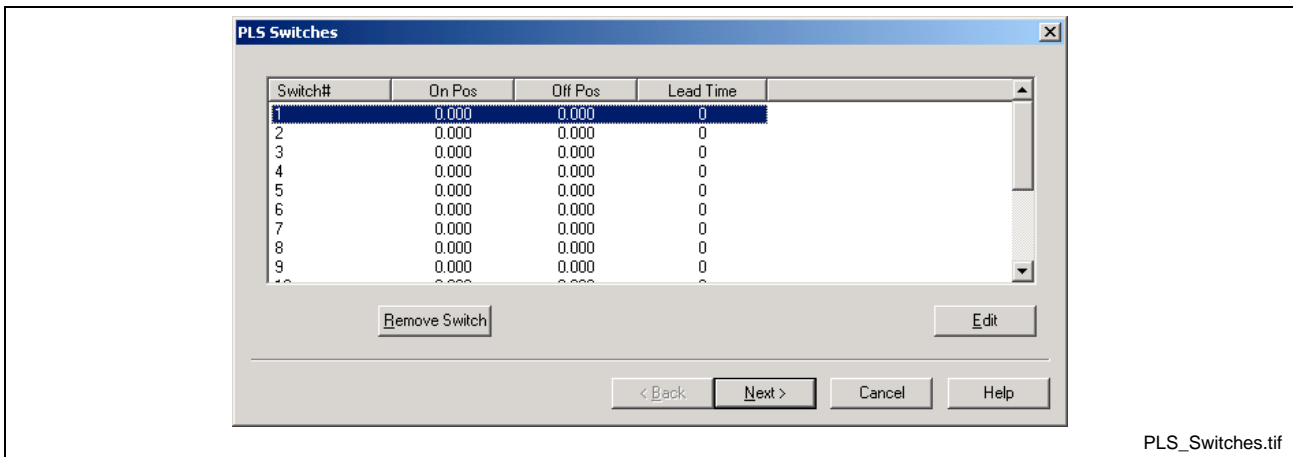


Fig. 8-20: PLS Switches Window

Switch Configuration for an Option Card PLS

The following steps configure a switch's On / Off position and assigns it to an output.

- Select the Switch Index number.

Note: Once a switch is configured and the **Apply** button is pressed, a new Switch Index number must be selected to configure a different switch. Pressing the **OK** button accepts the values and closes the window. The current switch number is displayed in the header portion of the window.

Example:

Switch 1 Configuration

- Enter the On and Off Positions.

Note: The On and Off positions for each switch is relative to the units of measurement in axis parameter A-0-0005 and task parameter T-0-0005.

Example:

If these parameters are set to 1 (mm), then the unit of measurement for the switch's On and Off positions is mm.

- Enter an Output number for the switch.

Note: The user can assign anywhere from 1 to 96 switches to one output or distribute them among the available 16 or 32 outputs. The total number of switches available for an Option Card PLS is 96.

- When all the switches are configured, press the **OK** button to close the *Switch # Configuration* window and return to the *PLS Switches* window.

- Press the **Next >** button to continue to the *PLS Master(s) Configuration* window.

VisualMotion Error for Switch Configuration

VisualMotion issues the following error when a different Switch Index number is selected without first applying any changes to an already configured switch.

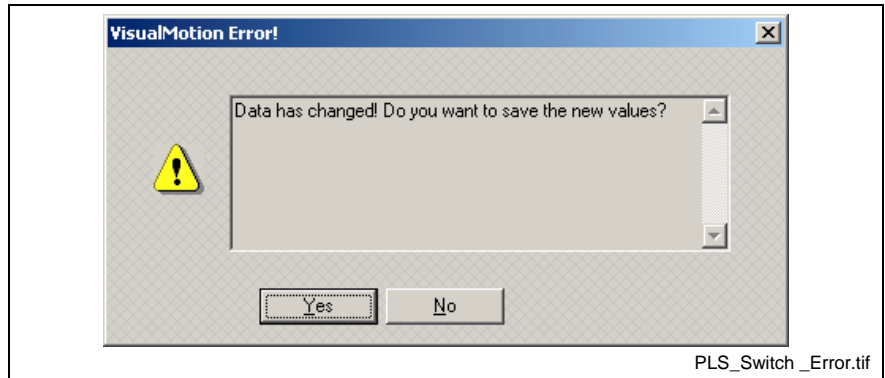


Fig. 8-21: PLS Switch Configuration Error Message

Example:

Switch 1 is configured and the **Apply** button is pressed. Now, the user makes a change to an On/Off position or the Output and then selects a different Switch Index number. VisualMotion assumes that you want to continue without first applying any changes to the current switch.

PLS Master(s) Configuration for an Option Card PLS

The *PLS Master* window is used to configure up to 8 PLS Masters. The available PLS types are:

- ELS Master (System Master)
- ELS Group
- Drive (DIAX04 and ECODRIVE03)

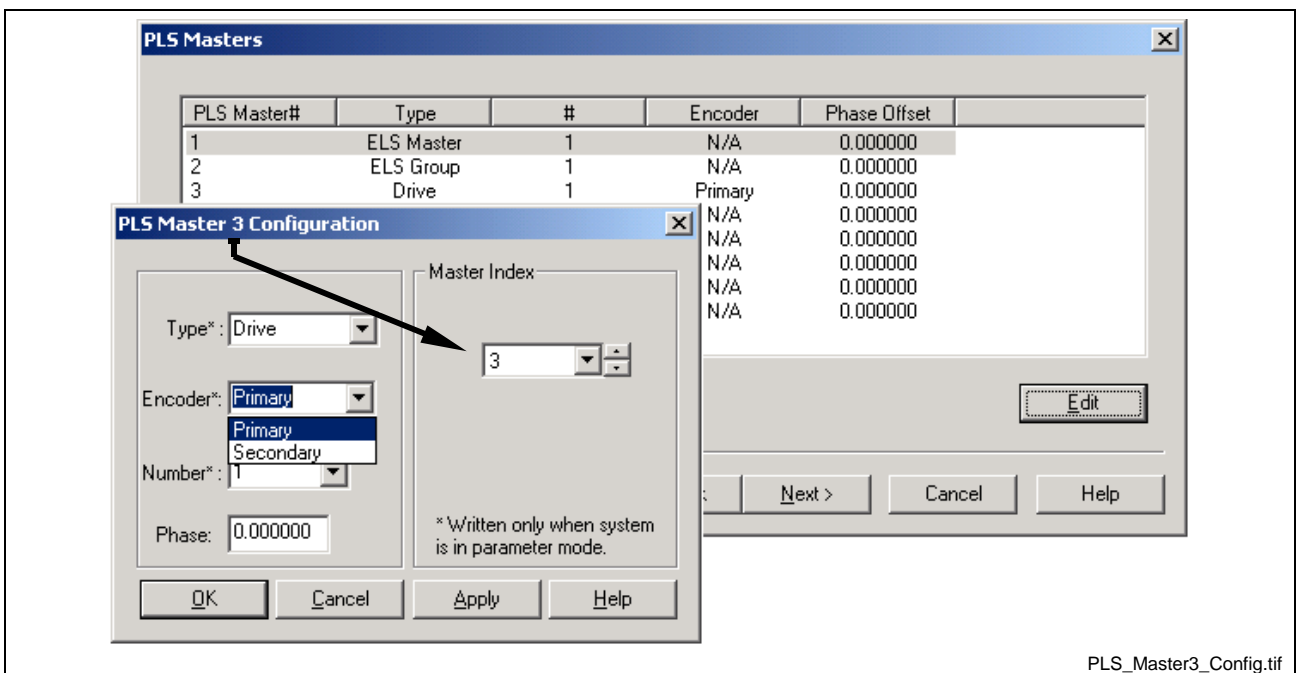


Fig. 8-22: PLS Master Configuration

1. Double click on the desired *PLS Master #* to open the *PLS Master # Configuration* window.
2. Select a *Master Index* number.

Note: Once a PLS Master is configured and the **Apply** button is pressed, a new *Master Index* number must be selected to configure a different PLS Master. Pressing the **OK** button accepts the values and closes the window. The current PLS Master number is displayed in the header portion of the window.

Example:

PLS Master 1 Configuration

3. Select an input *Type*:

- ELS Master
- ELS Group
- Drive

Note: ELS Master

When selected as a PLS Master type, the ELS Master's position value is used as an input to the PLS. In GPP 9, an ELS Master can be a Virtual Master, the output of an ELS Group, a Real Master or the output of a second PPC-R cross-communicating in a Link Ring.

ELS Group

In an ELS multiple master configuration, the output (position) of an ELS Group can be used as an input to the PLS. If a *Phase* value is added to the PLS Master Configuration window Fig. 8-22, then the value is added to the output of the ELS Group and does not affect any phase offsets that might have been configured in the ELS Group.

4. Select a Number from the drop down list corresponding to the input type selected.

Type	Allowable Number Range	Encoder
ELS Master	1-6	N/A
ELS Group	1-8	N/A
Drive	1-40 (SERCOS address)	Primary or Secondary

Note: When a Drive is selected, the allowable number range is 1-40 for up to 40 axes. Then the user must specify the *Encoder* for each axis as either the primary feedback or a secondary feedback device.

5. Enter a Phase offset if applicable. The *Phase* field is an offset that is added to the output of the PLS master.
6. When all PLS Masters are configured, press the **OK** button to close the *PLS Master Configuration* window and return to the *PLS Masters* window.
7. Press the **Next >** button to continue to the *PLS Register Assignment* window.

VisualMotion Error for PLS Master Configuration

VisualMotion issues the following error within the *PLS Master Configuration* window when a different Master Index number is selected without first applying any changes to an already configured PLS Master.

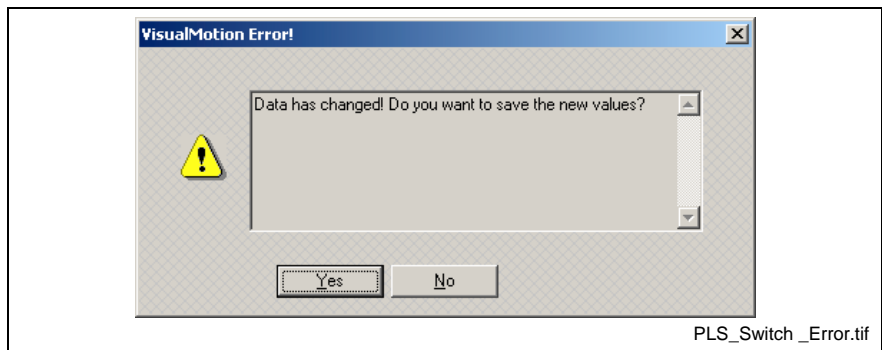


Fig. 8-23: PLS Switch Configuration Error Message

Example:

PLS Master 1 is configured and the **Apply** button is pressed. Now, the user makes a change to the Type, Encoder, Number or Phase fields and then selects a different Master Index number. VisualMotion assumes that you want to continue without first applying any changes to the current PLS Master.

PLS Register Assignment for an Option Card PLS

The PLS Register Assignment window is used to assign a VisualMotion register to both NSW01.1R Option Card PLSs installed in the control. The user can also set the range of motion of the graphical representation for the PLS switches. The following steps assign a register to the Option Card PLS outputs and sets the graphical limits of the PLS main window.

1. Accept the default register or set a different *Start Output Register*. The *End Output Register* is set automatically to the next consecutive register.
2. Accept the default register or set a different *Start Mask Register*. The *End Mask Register* is set automatically to the next consecutive register.

Note: During the initial setup, the Mask register bits cannot be set to (1). After the PLS is configured and downloaded to the control, the user must manually set the state of each Mask Register bit to (1) by selecting **Data ⇒ Registers** from VisualMotion main menu. Only the Output Register whose Mask Register is set to (1) will output a signal on the Option Card PLS.

3. Set the *Graphs Limits* minimum and maximum value.
4. Press the **Next >** button to continue to the PLS Outputs window.

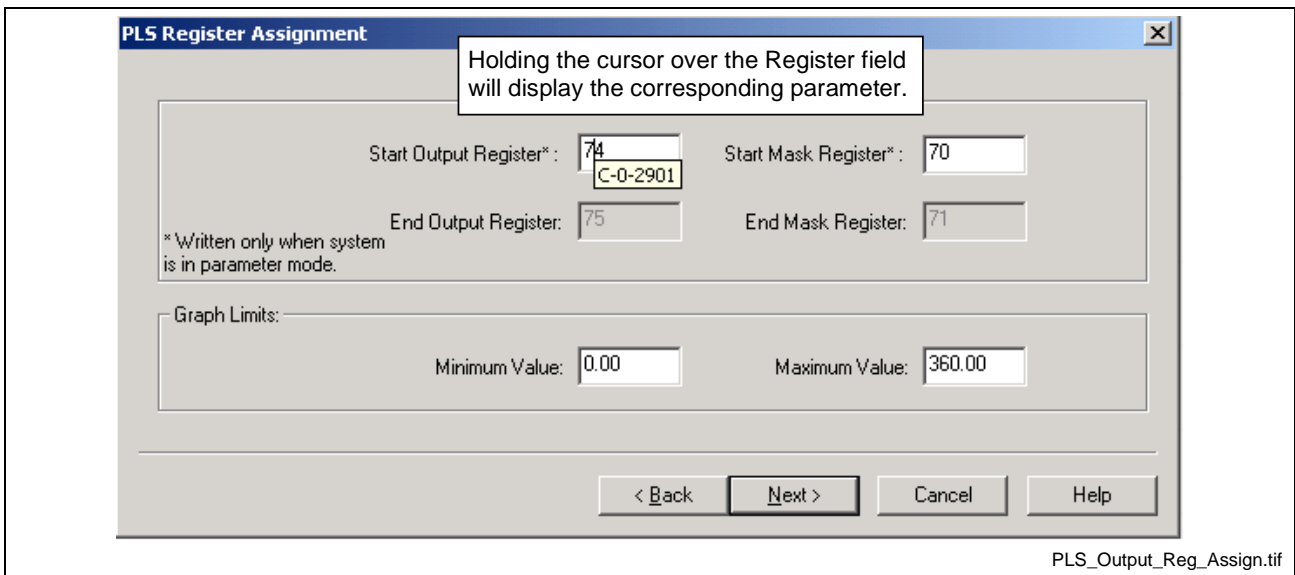


Fig. 8-24: PLS Register Assignment

Output Registers

The Start and End registers are assigned to the PLS outputs for monitoring the status of each output. The Start Output Register is assigned to outputs 01-16 on the first NSW01.1R installed in slot U2, where the outputs match the bit numbers respectively. The End Output Register is automatically set to the next consecutive register and assigned to outputs 17-32 of the second NSW01.1R whether or not it is installed in slot U3.

Mask Registers

The Start and End Mask registers are used to force the state of the output registers. On power-up, the bits in the mask registers are set to 0 to prevent any unwanted outputs from enabling. The PLS switches assigned to a start output register will not function unless the state of the register's complementary mask register bits are set to 1.

Example:

The 16 outputs of the first NSW01.1R card are assigned to register 74 and each output contains a PLS switch. In order for the outputs to function based on the On / Off positions of each switch, the bits in mask register 70 must be set to 1. Otherwise, the On / Off positions of the switch will be reached but the corresponding output register bit will not enable.

Note: The default state of the Start and End Registers is zero (0). The user must set the corresponding mask register each time after power-up.

Graph Limits

Set the minimum and maximum values for the graphical representation of all PLS switches.

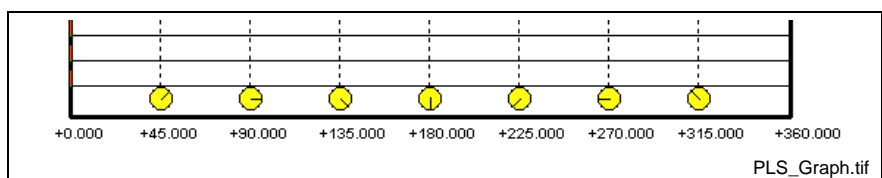


Fig. 8-25: Graph Limits

Example:

For rotary applications, the user can set the limits from 0.00 to 360.00 degrees.

For linear applications, the user can set the limits to match the actual range.

Example: -100.00 to 100.00

PLS Outputs

The *PLS Outputs* window is used to assign PLS Masters to configured Outputs. The final process in the configuration of a PLS is to assign a PLS Master to each output. From the window in Fig. 8-26, the user can scroll through the overall configuration of all 32 outputs.

Note: Although all 32 outputs are displayed and can be configured, only the first 16 outputs will have a physical output for PPC-R configurations having only one NSW01.1R installed.

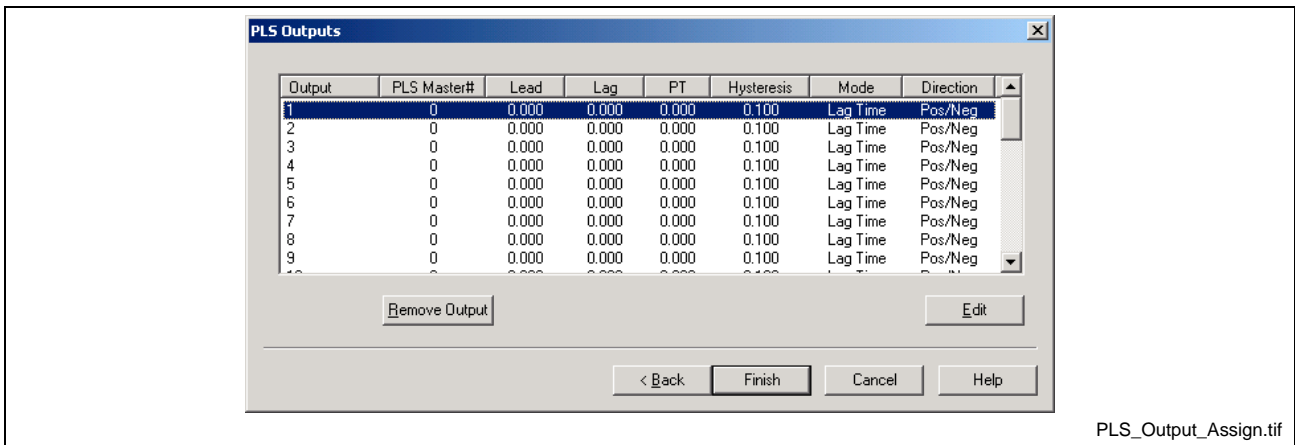


Fig. 8-26: Output Assignment

1. Double click on an output or highlight the output and press the **Edit** button to open the *Output # Configuration* window.
2. After all the outputs are configured, press the **Finish** button to end the configuration wizard.

Output Configuration

The *Output Configuration* window is used to fine tune each PLS output. From the *Output # Data* section, the user can make timing adjustments and set the direction in which the switch is recognized. The *Output Index* section is used to navigate between all 32 outputs. From the *Output # Switches* section, the user can view the current switch(es) configured to an output, add or remove any switches or edit a switch's On/Off position. The *Output # PLS Master* section is used to assign a PLS Master to the output.

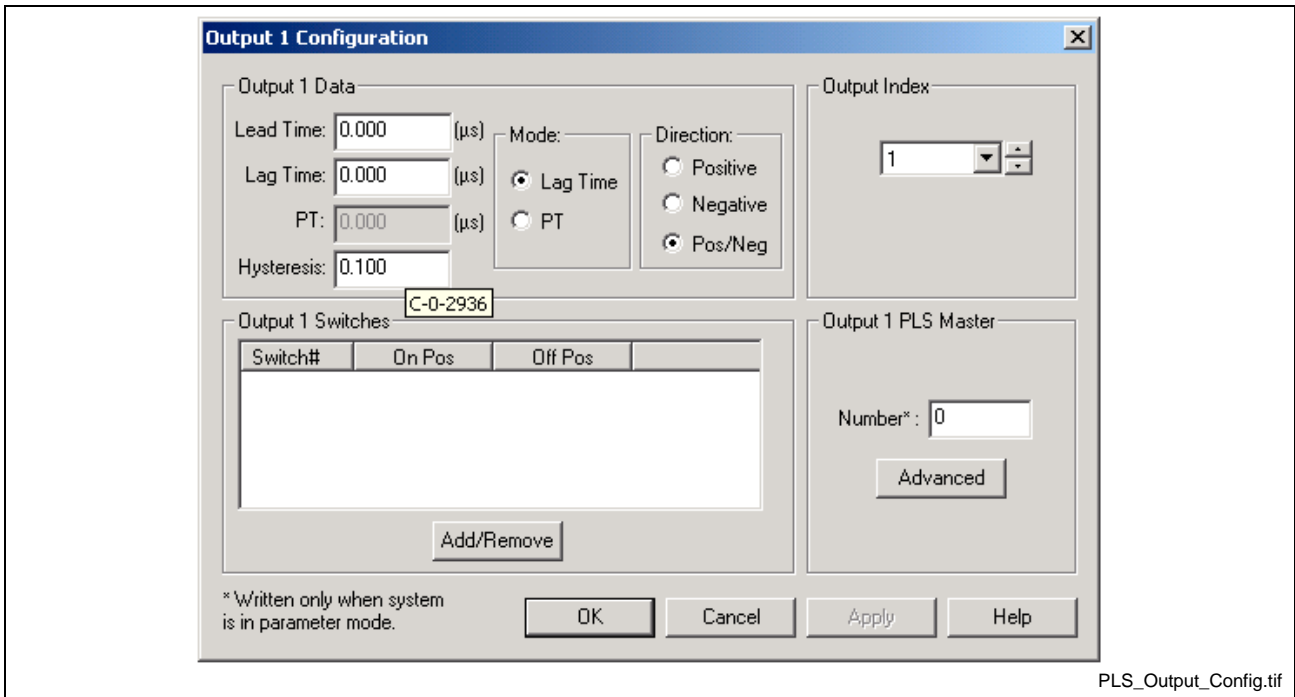


Fig. 8-27: Output Configuration Window

Holding the cursor over the following fields will display the corresponding parameter to which the configured value is stored.

Field Name	Parameter	Condition
Lead Time	C-0-2931	Always active
Lag Time	C-0-2932	Active when Mode = Lag Time
PT (Time Duration)	C-0-2933	Active when Mode = PT Mode
Hysteresis	C-0-2936	Always active
Mode	C-0-2934	Always active
Direction	C-0-2935	Always active
Number (Output # PLS Master)	C-0-2941	Always active

Table 8-9: Output Configuration Parameters

From the Output Configuration window, the user has the following configuration options:

Output # Data

The output data options are used to fine tune a PLS output. The associated control parameter is displayed when the cursor is held over an available field.

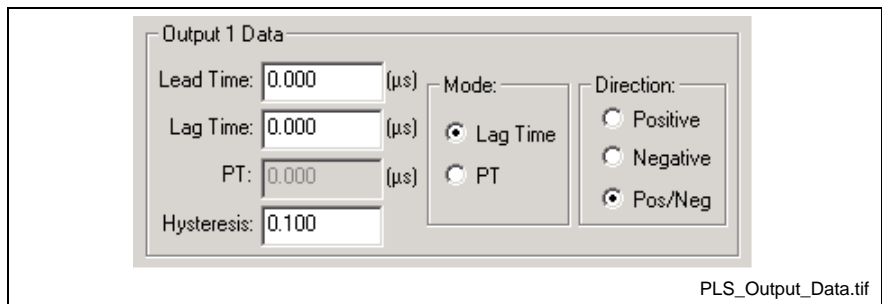


Fig. 8-28: Output Configuration Window Output Data Field

The following options are available:

- **Lead Time (C-0-2931)** - The amount of time that the output is enabled prior to reaching the switch's On position. This value is used by the control to calculate a position based on the switch's On position and the current speed of the PLS Master.
- **Lag Time (C-0-2932)** - The amount of time that the output is disabled prior to reaching the switch's Off position. This value is used by the control to calculate a position based on the switch's Off position and the current speed of the PLS Master. This option is only available when the *Mode* is set to Lag Time.

The time entered for both Lead and Lag Time is in increments of 250μs, up to a maximum of 500,000μs.

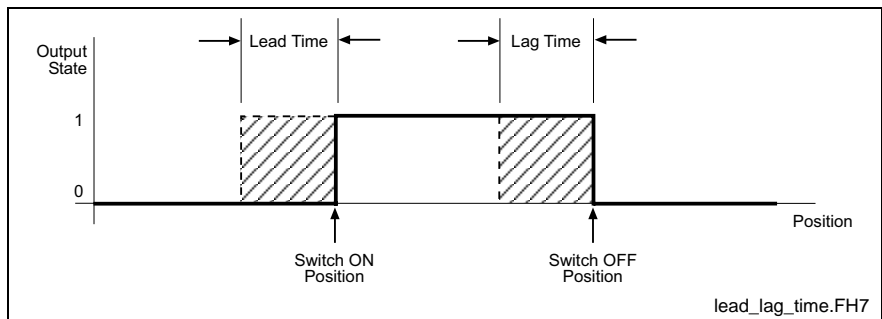


Fig. 8-29: Lead and Lag Time

- **PT (Time Duration) (C-0-2933)** - The amount of time that the output is maintained enabled every time the switch's On position (rising edge) is reached. The time is entered in increments of 250μs, up to a maximum of 1,000,000 μs.

Note: When the Output Mode is set to PT and a value of 0 is used, the switch's on position will be ignored.

Note: If after a rising edge is encountered and before the falling edge is reached a change of direction occurs and a PT time is set, the timer will run for the set time duration. This time duration can be reset and is only triggered with a rising edge.

Note: If a non-incremental time is entered, VisualMotion will issue an error and set the time to the next closest 250μs increment.

- Hysteresis (C-0-2936)** - This value can be either *positive or negative* and sets the tolerance distance used by the control to determine whether or not a switch is in position. This is used to prevent dithering of the output. The value entered for the hysteresis will affect all switches in the *Output Switches* section. A PLS Master that is either a Real Master or a Drive uses a feedback device to output a positional value. Due to small positional value corrections from the drive trying to hold or maintain a position, the feedback output data will experience small positional changes. The illustration in Fig. 8-30 shows the different reactions for positive and negative hysteresis values.

Note: Positive Hysteresis

In the positive direction, the switch's On and Off positions enables and disables the output.

In the negative direction, the switch's Off and On positions minus the hysteresis value enables and disables the output.

Negative Hysteresis

In the positive direction, the switch's On and Off positions plus the hysteresis value enables and disables the output.

In the negative direction, the switch's Off and On positions enables and disables the output.

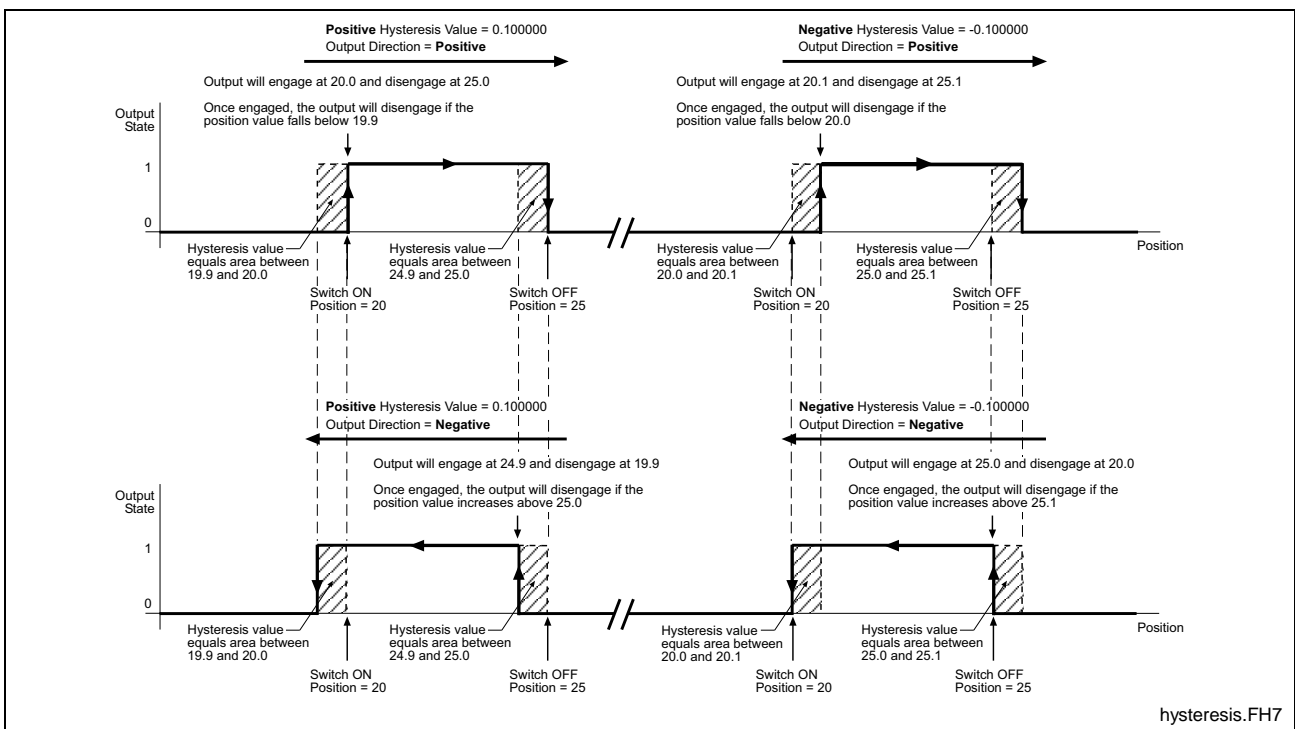


Fig. 8-30: Positive and Negative Hysteresis Value Reaction

Mode (C-0-2934)

The mode radio buttons enable options for Lag Time and PT.

Lag Time When selected, the Lead Time, Lag Time and Hysteresis fields are available for data entry.

PT When selected, the Lead Time, PT and Hysteresis fields are available for data entry.

Direction (C-0-2935)

This option sets the direction of the switches configured for the output in the Output # Switches section.

Positive When set to positive direction, all switches associated with the output will enable the output **only** in the positive direction with the On switch position and disable the output with the Off switch position.

Negative When set to negative direction, all switches associated with the output will enable the output **only** in the negative direction with the Off switch position and disable the output with the On switch position.

Note: Once the On position is reached for either positive or negative direction, any change in direction (while the output is enable) will immediately disable the output.

Positive/Negative This setting is a combination of the positive and negative direction settings. The outputs will react as described for both the positive and negative direction settings.

Output Index

The *Output Index* section is used to navigate between all 32 outputs.

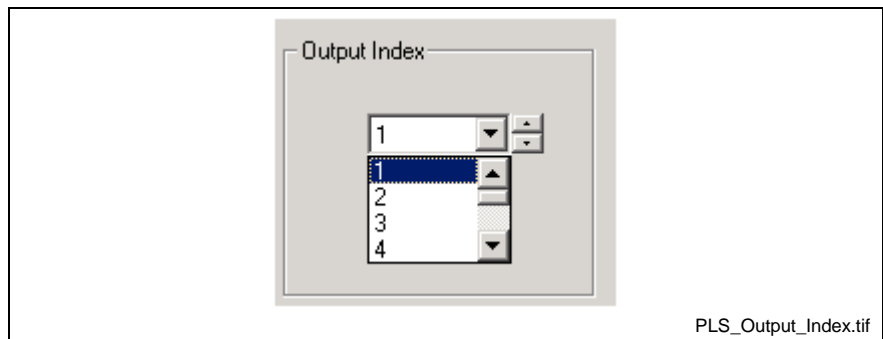


Fig. 8-31: Output Index

VisualMotion issues the following error within the *Output # Configuration* window when a different Output Index number is selected without first applying any changes to an already configured Output.

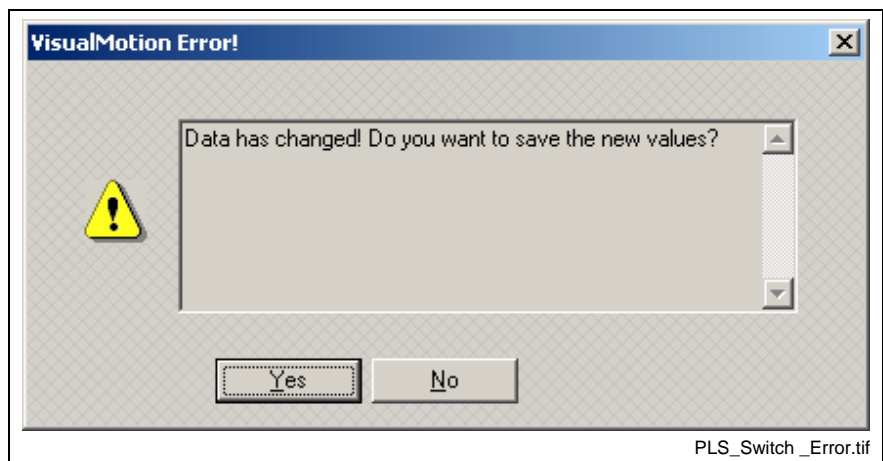


Fig. 8-32: PLS Output Index Error Message

Example:

Output 1 is configured and the **Apply** button is pressed. Now, the user makes a change to any field and then selects a different Output Index number. VisualMotion assumes that you want to continue without first applying any changes to the current Output.

Output 1 PLS Master (C-0-2941)

The Number field is used to specify the PLS Master that will be used by the system to provide positional data to the output.

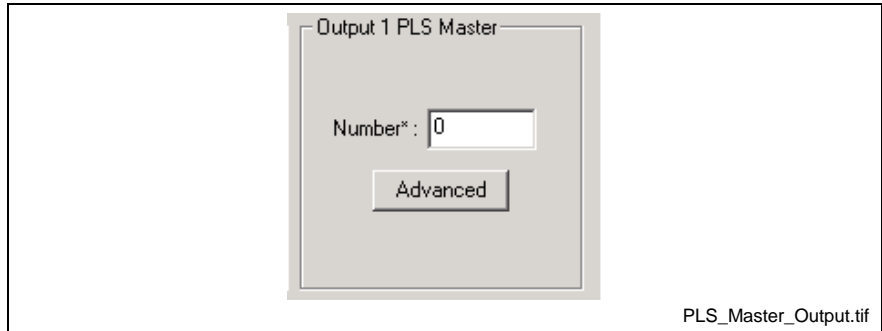


Fig. 8-33: Output PLS Master

The **Advanced** button can be used to select, modify an existing master or configure a new master. Refer to **PLS Master(s) Configuration** on page 8-20 for details. Every configured output must have a PLS Master assigned to function.

Note: The PLS Output Master Number field can be updated in offline project mode or in service mode. Updated in Phase 2

Output 1 Switches

The *Output # Switches* section displays all the configured PLS switches that were assigned to the output.

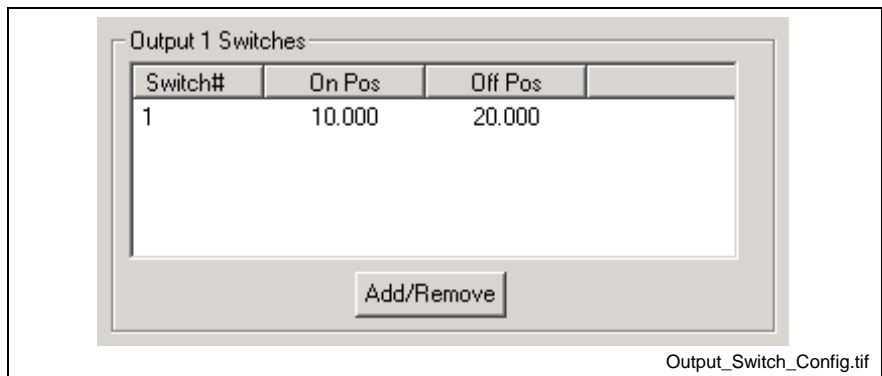


Fig. 8-34: PLS Output Switch Configuration

In addition to these switches, the user can double click on a switch or press the **Add/Remove** button to open the *Output Switch Configuration* window.

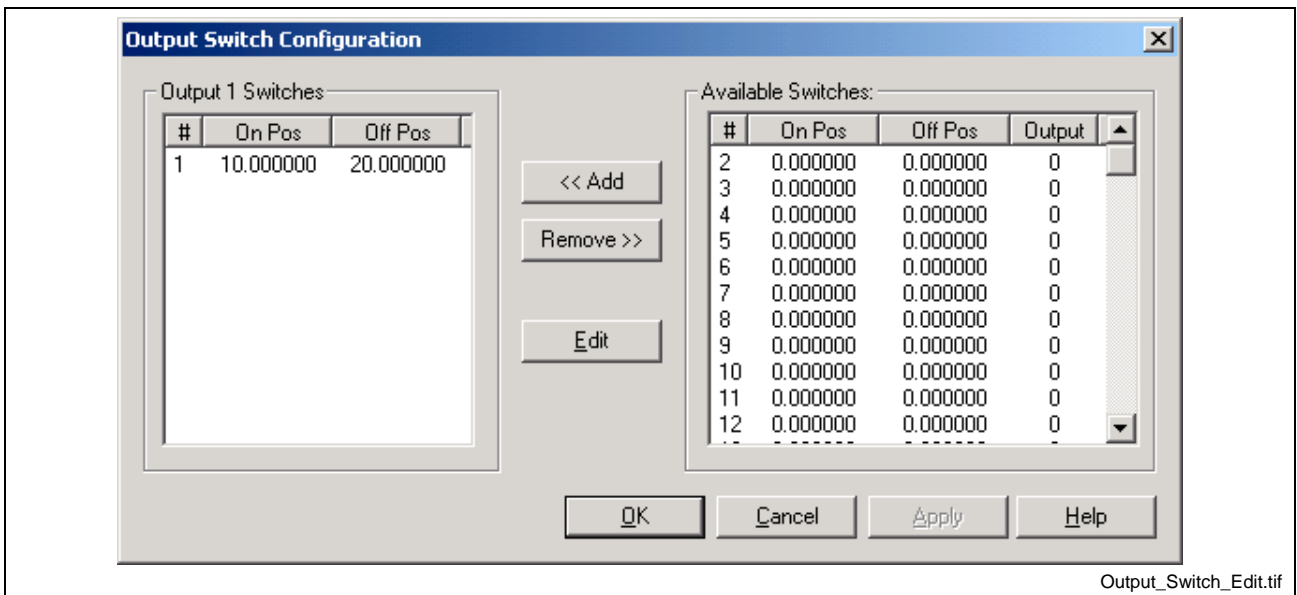


Fig. 8-35: PLS Output Switch Configuration

From the window in Fig. 8-35, the user can **<< Add** or **Remove >>** switches between *Available Switches* and *Output # Switches* by first selecting the switch.

Edit a Switch's On/Off Position

A switch's On/Off position can be edited in either the *Available Switches* or *Output # Switches* locations. Double clicking on a switch opens the *Switch # Configuration* window. Refer to Switch Configuration for a Control PLS on page 8-19 for details.

Note: The Output field can not be modified from this window. To change the output of a switch, **<< Add** it to the desired output in the *Output # Switch* section.

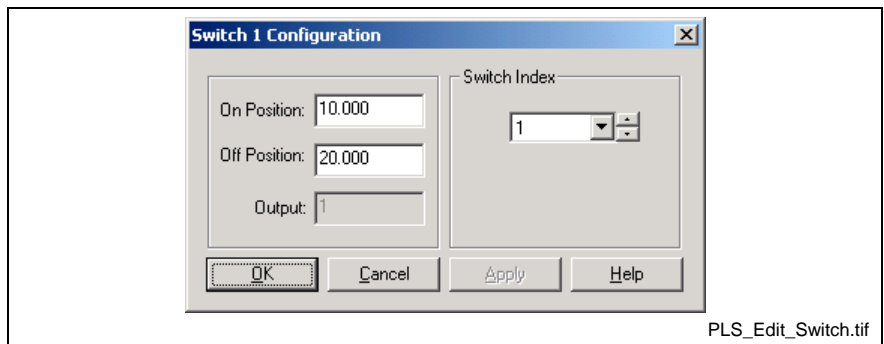


Fig. 8-36: Edit Switch

VisualMotion Message for Output Switch Configuration

When adding a switch from the *Available Switches* (assigned to a different output than the output number in the *Output # Switches*), the following VisualMotion Message will appear.

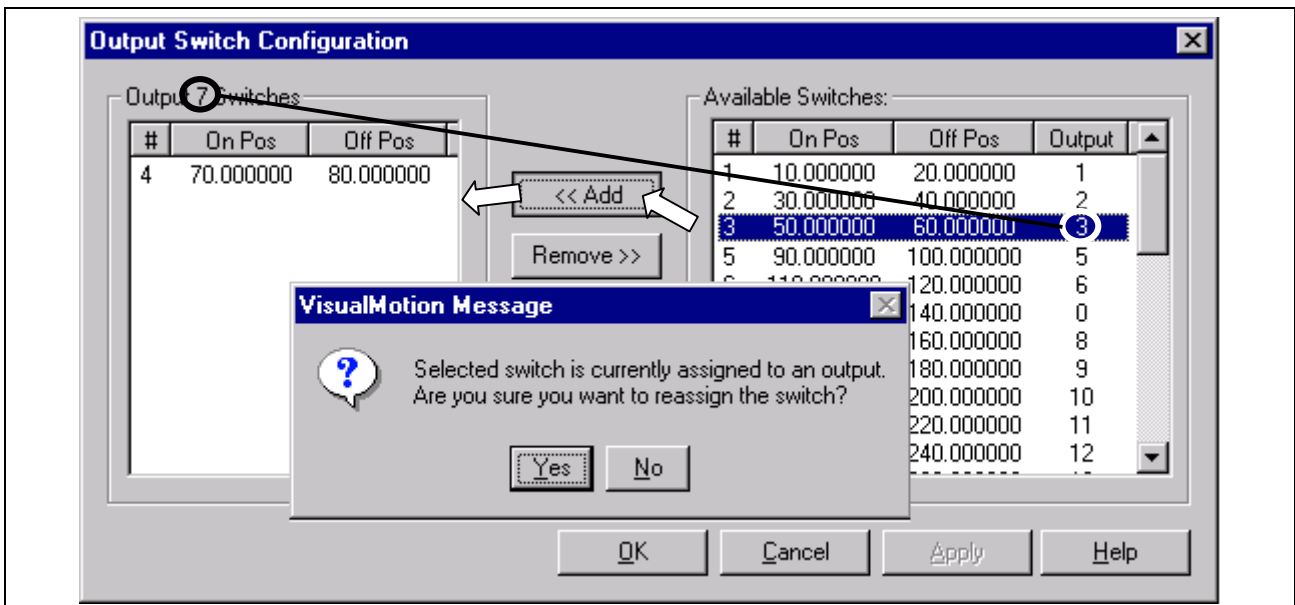


Fig. 8-37: VisualMotion Message for Output Switch Configuration

Note: If a switch is removed from the *Output # Switches*, the On and Off positions will be maintained but the output will be set to 0. The switch will appear in *Available Switches* with the output number set to 0.

8.5 Editing PLS Configurations

A PLS configuration created as part of a VisualMotion 9 project should be modified in offline project mode. This guarantees that the PLS configuration is saved in the correct format for the project and that the data is synchronized with the project.

Note: Modifying a PLS configuration in service mode only modifies the data on the control's memory. Offline project data is not updated when modifying PLS configurations in service mode.

Edit PLS Configuration in Project Mode

The editing of a PLS configuration is the same whether offline or online. The only difference is the source of the data and what happens to the project's state (synchronized or unsynchronized) when data is modified and saved in online mode.

Offline Editing

PLS data edited in offline mode is read and saved from/to the project's offline data. Afterwards, the new PLS project data can be synchronized with the control by selecting the *Online* toolbar button or selecting **File** ⇒ **Online**.

Online Editing

PLS data edited in online mode is read from the control's memory. The synchronization of project data edited and saved in online mode is dependent on whether or not the data can be written to in the control's current operating phase (phase 2 or phase 4).

If the control is in parameter mode (phase 2), all data that is edited in online mode is saved to both the project's offline data and to the control's memory. The control remains in its synchronized online state.

However, if the control is in phase 4, the project remains synchronized if the edited data being saved can be written to in phase 4. Otherwise the project becomes unsynchronized and the user is prompted to save the data to the offline project or discard the changes.

Refer to Synchronize Project Components on page 2-16 for details.

Editing PLS Configurations

To edit a project's PLS configuration, open the project "*.vmj" file and select **Commission** ⇒ **PLS**.

Any control, drive or Option Card PLS saved to the offline data or downloaded to the control will be displayed in the PLS tool. Refer to the following figure as an example.

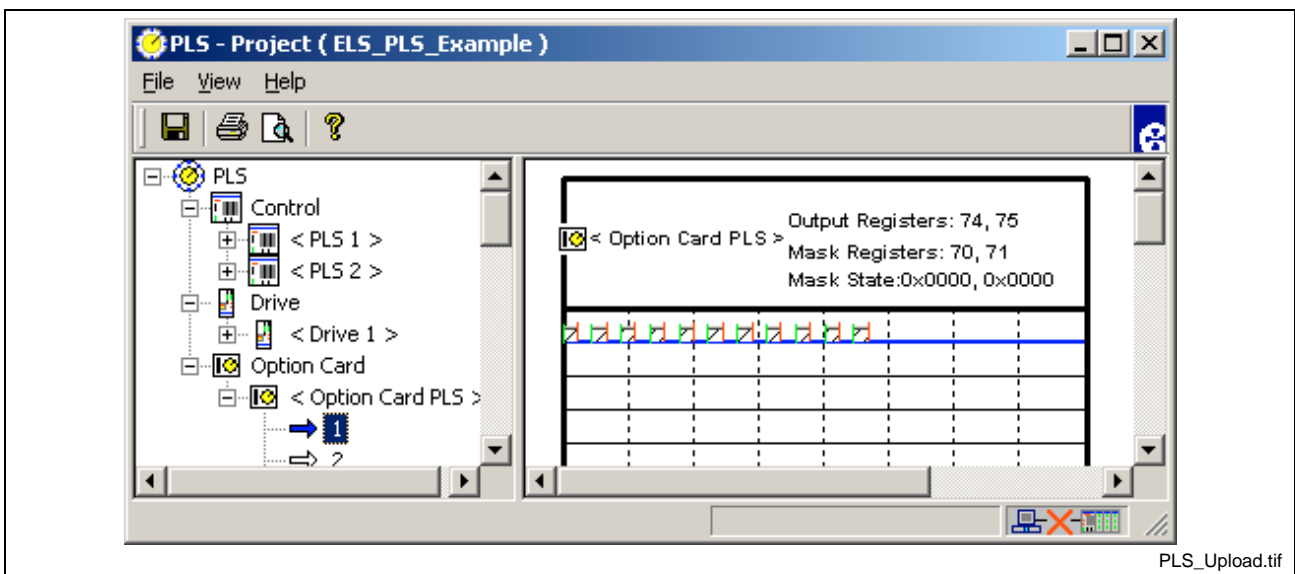
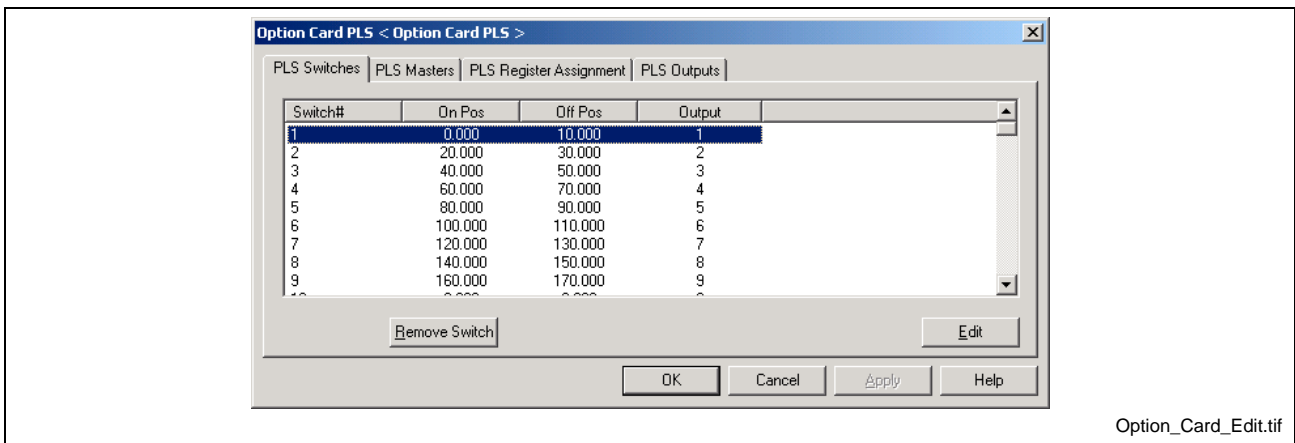


Fig. 8-38: Uploaded PLS Configurations

Select the type of PLS to edit from the PLS tree structure in the left window. Selecting the + symbol to the left of the desired PLS type will expand the tree structure to display the individual switches. Double clicking on an individual switch number in the tree structure or on the graphical switch representation will open an edit window from which any of the original data can be modified. Fig. 8-39 displays the edit window for a high speed Option Card PLS. Each PLS type has its own unique edit window displayed with a series of tabs corresponding to the necessary configuration elements for the selected PLS type.

Refer to the following sections for details:

- Configure a Control PLS on page 8-9
- Configure a Drive PLS on page 8-14
- Configure an Option Card PLS on page 8-18



Option_Card_Edit.tif

Fig. 8-39: Option Card PLS Edit Window

Importing a PLS Configuration into a Project

A PLS configuration downloaded to the control can be imported into a project while in online mode, or from another project or file when in offline mode.

Use the following steps to import a PLS configuration from data stored on the control.

1. Start VisualMotion and open the target project for importing the PLS configuration.
2. Switch VisualMotion to online mode.

Note: To transfer control data, previously downloaded to the control, switch VisualMotion to online mode.
To import data from another offline project or file, switch VisualMotion to offline mode.

3. Select **File** ⇒ **Import Project Component** from VisualMotion Toolkit's main menu.
4. From the "Transfer Control Data to Project" window, select from the following PLS configurations:
 - Option Card PLS
 - Drive PLS
 - Control PLS (located under the Program Data checkbox)

Note: The system must be in parameter mode before transferring data from the control.

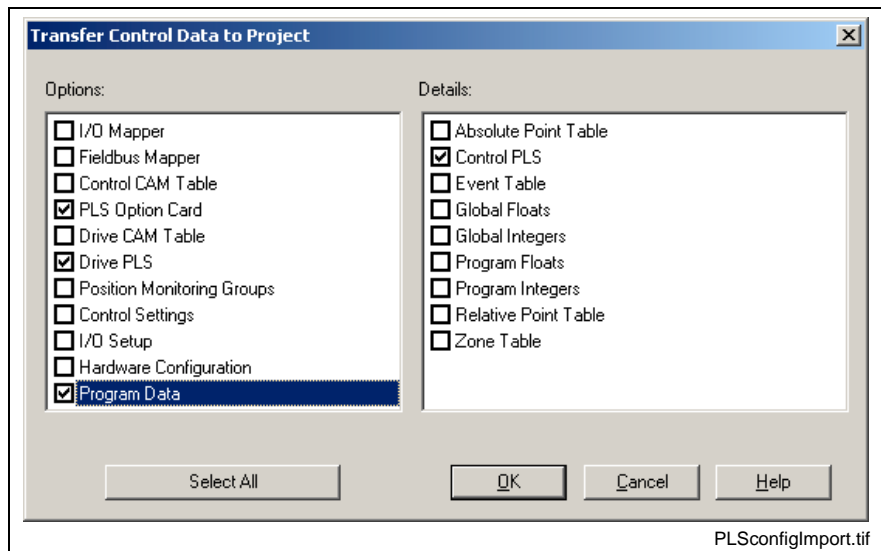


Fig. 8-40: PLS Configuration Transfer from Control

5. Press the OK button to transfer the selected PLS configurations to the current project. The PLS configuration data is now synchronized with the current project.

Note: When importing data from another project or file in offline mode, the data is not synchronized with the project until VisualMotion is switched to online mode and the data is downloaded to the control.



Edit PLS Configuration in Service Mode

VisualMotion's service mode allows the user to make modifications to a PLS configuration stored in the control's memory. PLS configurations downloaded to the control or saved to a file in service mode are not synchronized with a project's offline data.

Note: It is the responsibility of the project manager to ensure that PLS configurations modified in service mode are imported into the appropriate VisualMotion project.

Note: A serial or Ethernet connection, with established communication to the control, is required before proceeding.

The following steps outline the procedure for uploading and downloading an existing PLS configuration from and to the control for modifications.

1. Start VisualMotion and select the "Service" mode radio button. Next, select **Commission** ⇒ **PLS** to open the PLS Tool.
2. Switch the control to parameter mode.
3. Upload the PLS configuration by selecting **File** ⇒ **Get PLS Configuration from GPP Control** or click the upload icon ().
4. Make the necessary modifications to the PLS configuration. Save the file and download the modifications to the control by clicking the download icon ().

Editing with a Right Mouse Click

An existing PLS configuration can be modified by right clicking the mouse over any PLS tree icon. PLS configurations can be added, edited or removed.

Right clicking over the main PLS tree icon allows the user to add a new PLS configuration of the following types:

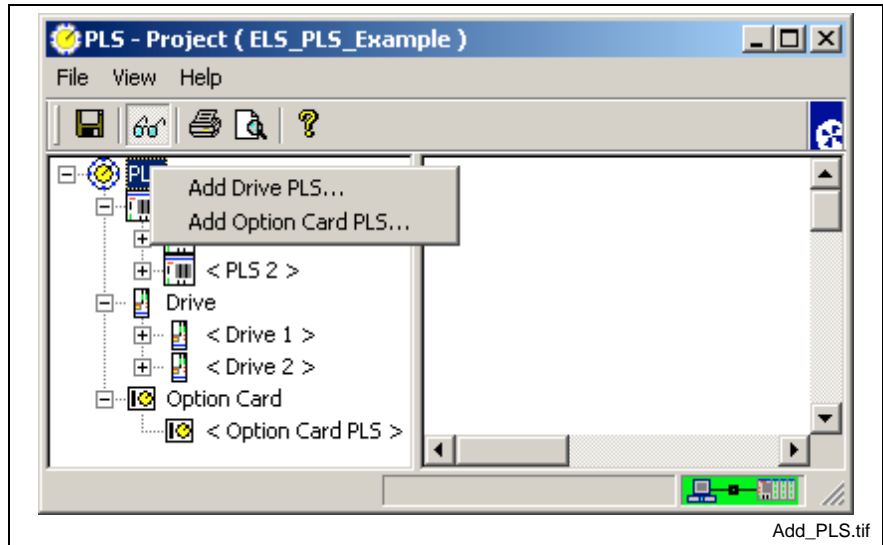


Fig. 8-41: Adding Additional PLSs

Right clicking over an existing PLS type tree icon (Drive or Option Card PLS) allows the user to add a second or third (if allowed) configuration of that type.

Note: Only Drive and Option Card PLSs can be added to the system. Only two Control PLSs are allowed. For this reason, both Control PLSs will be displayed.

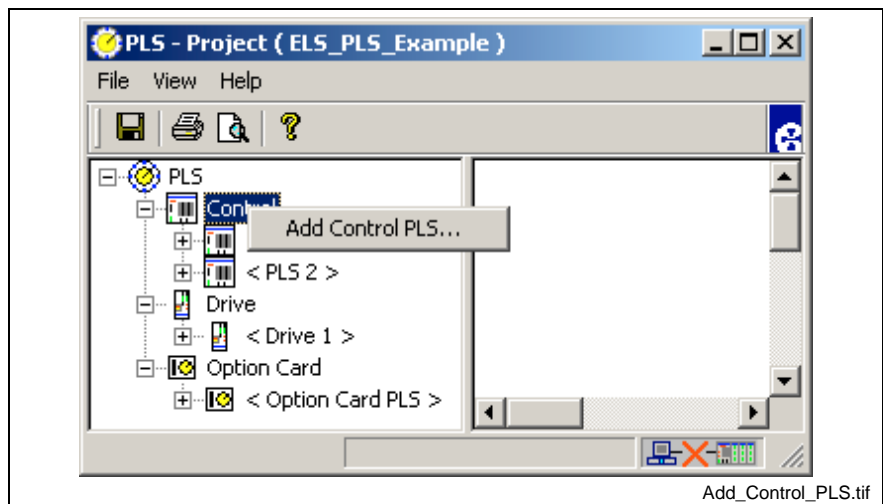


Fig. 8-42: Add_Control_PLS

Right clicking over an existing PLS configuration allows the user to edit or remove the selected configuration.

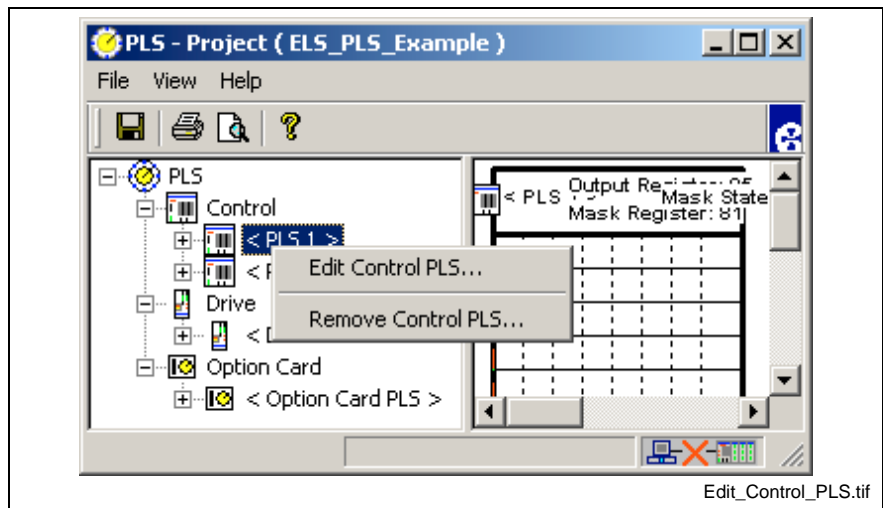


Fig. 8-43: Edit or Remove PLS Configuration

Note: The above mentioned mouse shortcuts launch the PLS configuration wizard for the corresponding PLS type.

8.6 Saving and Downloading PLS Configurations


PLS configurations parameterized in offline or online project modes can be saved to offline data or synchronized with the control. PLS configurations parameterized or modified in service mode are not synchronized with any project. The data is stored either in the control's memory or as a parameter (*.prm) file on the hard drive.

Saving a PLS Configuration

The saving of a PLS configuration varies slightly between project and service mode. In project mode, all configured PLS types (Control, Drive and Option Card) can be saved to a project in offline mode or synchronized with the control if edited and saved while online. In service mode, Drive and Option Card PLSs can be saved to a file on the PC's hard drive. However, a Control PLS is saved only with the user program and not as a file on a PC's hard drive.


Save in Project Mode

Project mode provides an offline and online editing

In a project, PLS configurations can be saved to offline data files in the project folder structure. The PLS configuration is saved either by selecting the Save icon  or by switching VisualMotion Toolkit to online mode.

When switching to online mode, all PLS configuration parameters are downloaded to the control.

Save in Service Mode

PLS configurations can be saved to the hard drive by selecting **File** ⇒ **Save As...** or by pressing the Save icon . The file is saved to the VisualMotion 9 project folder with a *.prm extension.

Note: Individual PLS configurations can be saved only when configured using **File** ⇒ **New** and saving a single configuration type.

Downloading a PLS to the Control

Downloading a PLS configuration varies between project and service modes. PLS configurations downloaded in project mode are synchronized with the project in which they were created. Service mode is intended for modifying PLS configurations when the project data is not available to the user.

Note: After a PLS configuration is downloaded to the control, the assigned PLS mask register must be set before the actual PLS output can fire.


Download in Project Mode

In project mode, a PLS configuration is downloaded when switching to online mode. The user is asked to save any modified data. The *Synchronize Project Data* window allows the user to selectively download PLS types to the control and drives.

A configured Option Card PLS downloaded to the control is automatically built and activated before the VisualMotion switches to online mode.

Note: The system must be in parameter mode to download PLS parameters to the control. Any parameter that can only be downloaded in parameter mode will be indicated by a VisualMotion error.

Download in Service Mode

In service mode, PLS configurations are downloaded to the control and drives by selecting **File** ⇒ **Send PLS Configuration to GPP Control** or by pressing the download icon ().

After a successful download, the user has the option to build and activate the new Option Card PLS if present in the system. Otherwise, any configured Control and Drive PLS are activated automatically.

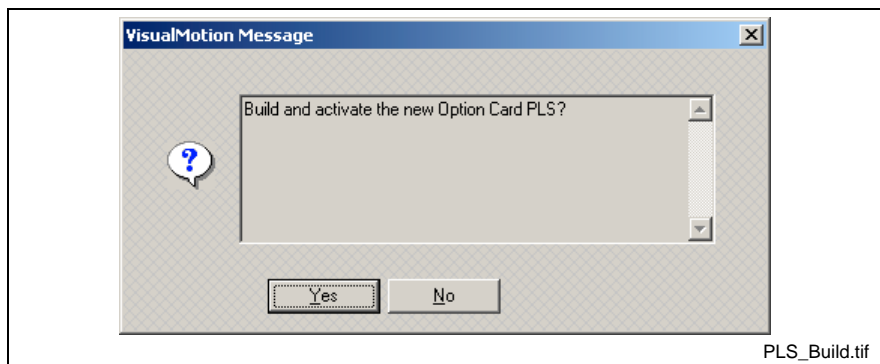


Fig. 8-44: PLS Build New Option Card Message

Note: The system must be in parameter mode to download all the elements of every configured PLS.

If the system is not in parameter mode, VisualMotion will issue an error indicating to the user that certain values will not be saved and offer an option to continue. If the user continues, additional errors will be issued indicating which parameter lists will not be written.

8.7 Uploading PLS Configurations


The uploading of a PLS configuration can be performed in project or service mode.

Upload in Project Mode

In project mode, PLS configurations that were saved to the project's offline data are automatically uploaded when selecting **Commission** ⇒ **PLS** in offline mode. The PLS tool reads all pertinent PLS parameters from offline data and displays them in the PLS tool. If the project's offline data does not contain a configured PLS, then the PLS tool display only the basic PLS icons.

Note: To upload a PLS configuration from the control's memory and save it as part of the project's offline data, refer to Importing a PLS Configuration into a Project on page 8-33.

Upload in Service Mode

In service mode, the user must initiate the upload by selecting **File** ⇒ **Get PLS Configuration from GPP Control** or by pressing the upload icon (). PLS configurations can also be uploaded from a file by selecting **File** ⇒ **Open...** and selecting the correct file with the *.prm extension.

Note: When opening a file, if a PLS configuration is currently opened, a VisualMotion Message will warn the user that "Data will be lost! Are you sure you want to continue?"

Service mode is primarily use to upload PLS configurations from a file or from the control's memory when the user does not have access the offline project data and modifications are required. Refer to Edit PLS Configuration in Service Mode on page 8-34 for details.

8.8 Monitoring a PLS Status

PLS configurations can be monitored in project mode or service mode. The monitoring feature of the PLS tool allows the user to monitor the status of the output registers that are assigned to each PLS type.

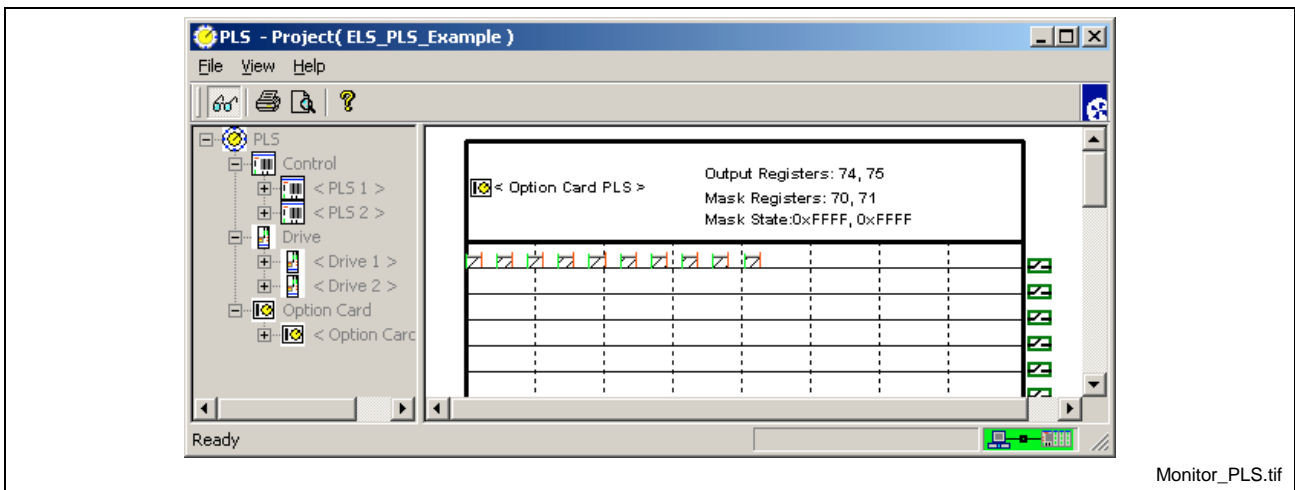



Fig. 8-45: Monitoring a PLS Status

Monitor in Project Mode

To monitor a PLS configuration in project mode, follow these steps:



1. Open the VisualMotion project file of the active program to monitor and switch to online mode.
2. Select the PLS type to monitor (Control, Drive or Option Card PLS) and press the Monitor Status icon () or the F7 key.

The PLS tool icon tree structure grays out and a small switch is displayed to the right of each PLS switch line. Each small switch changes from an open switch to a closed switch each time the PLS switch output is active.

To monitor a different PLS type, deselect the monitor status icon, select a different PLS type and press the monitor status icon again.

Monitor in Service Mode


To monitor a PLS configuration in service mode, follow these steps:

1. Open VisualMotion Toolkit in service mode and open the PLS tool by selecting **Commission** ⇒ **PLS**.
2. Upload the active PLS configuration on the control by selecting **File** ⇒ **Get PLS Configuration from GPP Control** or by pressing the upload icon ().
3. Select the PLS type to monitor (Control, Drive or Option Card PLS) and press the Monitor Status icon () or the F7 key.

The PLS tool icon tree structure grays out and a small switch is displayed to the right of each PLS switch line. Each small switch changes from an open switch to a closed switch each time the PLS switch output is active.

To monitor a different PLS type, deselect the monitor status icon, select a different PLS type and press the monitor status icon again.

PLS Message

The PLS Message is only available in service mode. The PLS Configuration tool issues the following *PLS Message* window if a PLS configuration is uploaded from the control, modified, downloaded and then the Monitor Status icon () or the F7 key is pressed.

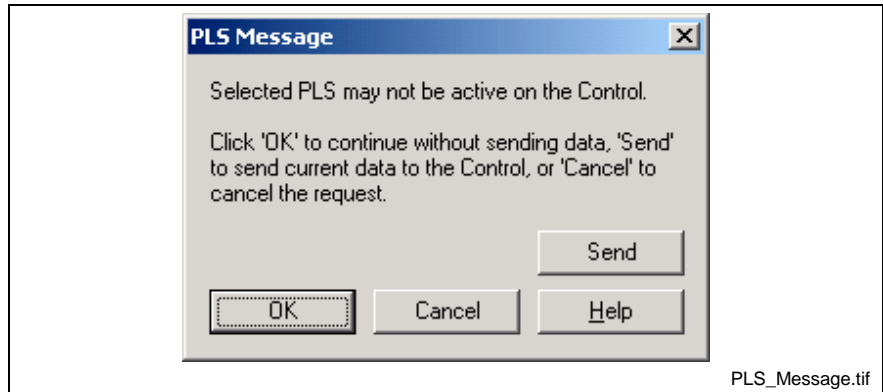



Fig. 8-46: PLS Message in Service Mode

Note: If the download  button is pressed, the control should be in parameter mode to send all data relevant to the PLS configuration.

8.9 Access PLS Data via the Calc Icon

After a PLS is configured and downloaded to the control, the user can make modifications to the elements of a PLS in the user program using the **Calc** icon. All three PLS types (Control, Drive and Card) can be accessed using the Calc icon.

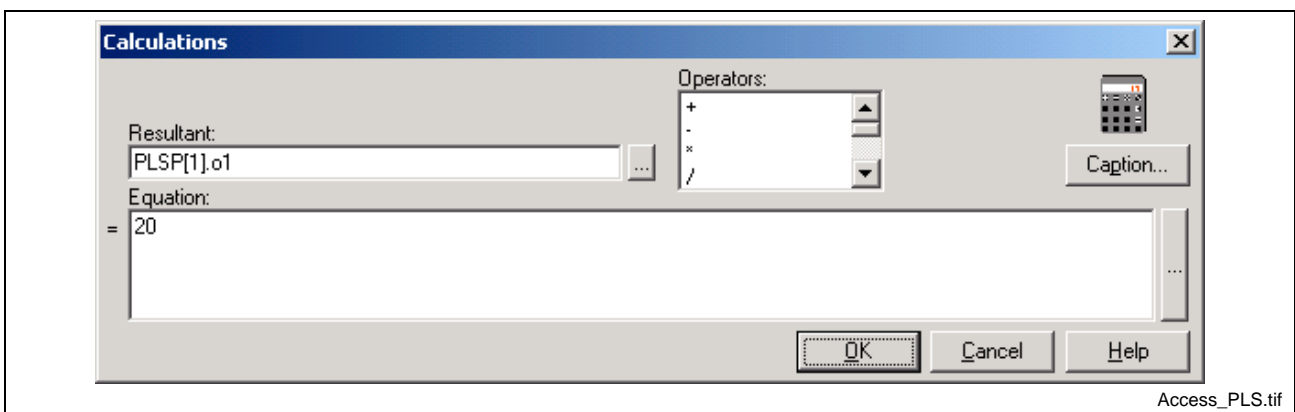


Fig. 8-47: Calc Icon Accessing a PLS

Note: Any modifications to PLS elements will not be retained after recycling power.

Control PLS

A VisualMotion Control PLS is stored with the user program as a separate table, and can be assigned to an ELS Master, ELS Group or Real Master. A user program can contain a maximum of two active Control PLS. The following table lists the elements of an Option Card PLS that can be easily modified using the Calc icon.

Syntax	Description	Variable Range	Example	Comment
PLS[x].a	Number	1-40 when t=3 or 4, Drive No. 1-6 when t=5, ELS Master 1-8 when t=6, ELS Group	PLS[1].a=2	set drive number of PLS 1 to 2
PLS[x].o	Phase Offset		PLS[1].o=20	add an offset of 20 to PLS 1
PLS[x].r	Output Register		PLS[1].r=70	set output register of PLS 1 to 70
PLS[x].t	PLS Input Type	3 = Drive's primary feedback 4 = Drive's secondary feedback 5 = ELS Master 6 = ELS Group	PLS[1].t=3	set input type of PLS 1 to 3 (drive's primary feedback)
PLS[x].on1-on16	On Position		PLS[1].on1=10	switch 1 On position is 10
PLS[x].off1-off16	Off Position		PLS[1].off1=20	switch 1 Off position is 20
PLS[x].lt1-lt16	Lead Time		PLS[1].lt1=5	switch 1 lead time is 5
x = PLS number 1-2				

Table 8-10: Elements of a Control PLS

Drive PLS

VisualMotion 9 supports drive based PLS configurations for DIAX04 and ECODRIVE03 digital drives. The drive based PLS is stored on the drive in parameter lists. The elements of this list cannot be modified individually. Modifying one element of the list requires sending the entire list over the service channel.

Syntax	Description	Example	Comment	Parameter
PLSD[x].r	Output Register	PLSD[1].r=72	set output register of drive 1 to 72	A-0-0009
PLSD[x].t	PLS Input Type	PLSD[1].t=1	set input type of drive 1 to 1 (0=disable, 1=motor encoder, 2=external encoder)	P-0-0131
PLSD[x].on1-on16	On Position	PLSD[1].on1=10	switch 1 On position for drive 1 is 10	P-0-0132
PLSD[x].off1-off16	Off Position	PLSD[1].off1=20	switch 1 Off position for drive 1 is 20	P-0-0133
PLSD[x].lt1-lt16	Lead Time	PLSD[1].lt1=5	switch 1 lead time for drive 1 is 5	P-0-0134
x = drive number range 1- 40				

Table 8-11: Elements of a Drive PLS

Option Card PLS

A user program can contain only one active Option Card PLS. The following tables list the elements of an Option Card PLS.

Master Elements

The following Master elements are available for modification using the Calc icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].ox	Phase Offset	PLSP[1].o1=20	add an offset of 20 to PLS Master 1	C-0-2943
x = Master range 1-8				

Table 8-12: Master Elements of an Option Card PLS

Switch Elements

The following switch elements are available for modification using the Calc icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].onx	On Position	PLSP[1].on3=40	switch 3 On position is 40	C-0-2920
PLSP[1].offx	Off Position	PLSP[1].on3=60	switch 3 Off position is 60	C-0-2921
PLSP[1].outx	Assigned Output	PLSP[1].out3=2	switch 3 is assigned to output 2	C-0-2922
x = switch range 1- 96				

Table 8-13: Switch Elements of an Option Card PLS

Output Elements

The following Output elements are available for modification using the Calc icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].ltx	Lead Time	PLSP[1].lt2=500	add a lead time of 500µs to output 2	C-0-2931
PLSP[1].lgx	Lag Time	PLSP[1].lg2=500	add a lag time of 500µs to output 2	C-0-2932
PLSP[1].ssx	PT (Time Duration)	PLSP[1].ss2=1000	maintain output 2 On for 1000µs	C-0-2933
PLSP[1].hxx	Hysteresis	PLSP[1].h2=0.1	add a hysteresis of 0.1 to the output 2	C-0-2936
PLSP[1].mdx	Mode	PLSP[1].md2=0	set mode of output 2 to Lag Time (0=Lag Time, 1=PT)	C-0-2934
PLSP[1].drx	Direction	PLSP[1].dr2=0	set direction of output 2 to positive (0=pos, 1=neg, 2=pos/neg)	C-0-2935
x = output range 1-32				

Table 8-14: Output Elements of an Option Card PLS

9 PPC-P11.1 Control Functionality

9.1 Overview

The PPC-P11.1 is a PCI motion control card designed to work with multiple application-specific firmware. This chapter describes how the PPC-P11.1, using GMP9 firmware, works as part of a complete PLC / Motion Control system solution. The PPC-P11.1 can function as a stand-alone motion control or be combined with additional system components, such as:

- Logic controller (PLC or SoftPLC)
- Fieldbus master
- HMI package (WinHMI or WonderWare)
- I/O devices
- Digital drives and motors

Communication between the PPC-P11.1 and system components is performed via a Dual Port RAM (DPR) / PCI interface.

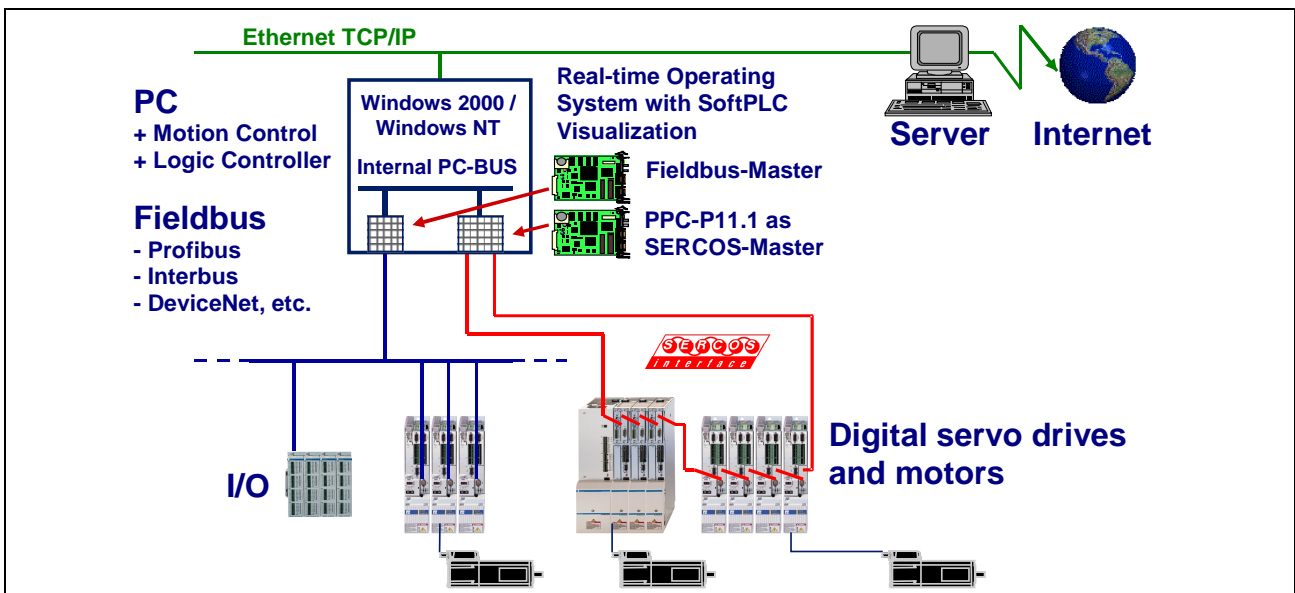


Fig. 9-1: PPC-P11.1 and PLC System

Fieldbus and I/O Support

The following table describes fieldbus and I/O supported by GMP9 firmware:

Interface	Supported	Not Supported
Fieldbus Interface	No firmware support ^{Note 1:}	VisualMotion fieldbus slave interface cards
I/O Interface	SERCOS RECO02 modules ECO-X module DIAX04 I/O cards	Local RECO02 modules
Note 1: If a fieldbus interface is required, the logic controller must be equipped to communicate with a fieldbus master card.		

Table 9-1: Fieldbus and I/O Support

Bosch Rexroth Interfaces

The following PC interfaces are available from Bosch Rexroth.

Windows™ Environment

- VisualMotion Toolkit's Data Mapper (Fieldbus Mapper) for configuring the cyclic channel.
- VisualMotion DDE (VM_DDE) Server for communicating with VisualMotion Toolkit 9 via the DPR.
- Scalable Communication Platform (SCP) for communication with HMI products and VisualMotion Toolkit 9 via the DPR.

RTX (Real-time Operating System from VentureCom)

- Real-time Dynamic Linking Libraries (RTDLL) for interfacing with the Siemens WinAC PLC running in an RTX environment.

PCI Bus Memory

The PPC-P11.1 is identified on the PCI bus with the following vendor and device ID.

PCI	Value	Description
Vendor ID	0x16F2	Bosch Rexroth
Device ID	0x0001	PPC-P11.1
Base Address ID	0	used by PPC-P11.1 PCI memory

Table 9-2: PCI Identification

During power up, the PCI bus allocates a 32K memory block used by the following PPC-P11.1 memory structure. The listed address offset is added to the assigned PCI base address (0).

Memory	Address Offset	Length
Reserved	0x0000 – 0x03FF	1 KB
Operations Registers	0x0400 – 0x07FF	1 KB
Reserved	0x0800 – 0x3FFF	14 KB
Dual Port RAM	0x4000 – 0x5FFF	8 KB
Reserved	0x6000 – 0x7FFF	8 KB

Table 9-3: PCI Bus Address for the PPC-P11.1

9.2 DPR Interface

The DPR interface provides a common memory area accessible to system components for sharing and transmitting system data and status.

DPR Memory	Details
Register Channel	128 Input / 128 Output registers accessible to the PLC and PPC-P11.1
Cyclic Channel	64 Word Input / 64 Word Output, with optional multiplexing
Non-Cyclic Channel	Supports both modified Short Format 3 ^{Note 1.} (SF3a) and SIS ^{Note 2.}
Status and Control	Provides status and control of the PLC and PPC-P11.1 Interface
PPC-P11.1 Diagnostic Information	Provides diagnostics and current SERCOS phase
Operation Registers	Interrupt and mailbox registers used between the host (PC) and local (PPC-P11.1).
Programming Channel	Com channel used by Windows application through SCP or VM DDE Server

Note 1. Refer to the Fieldbus Interface chapters in the VisualMotion 9 Application Manual for Short Format 3 details.
 Note 2. Refer to chapter 12.4 for SIS details

Table 9-4: DPR Memory Structure

The following figure illustrates the standard communication paths between the PC and the PPC-P11.1.

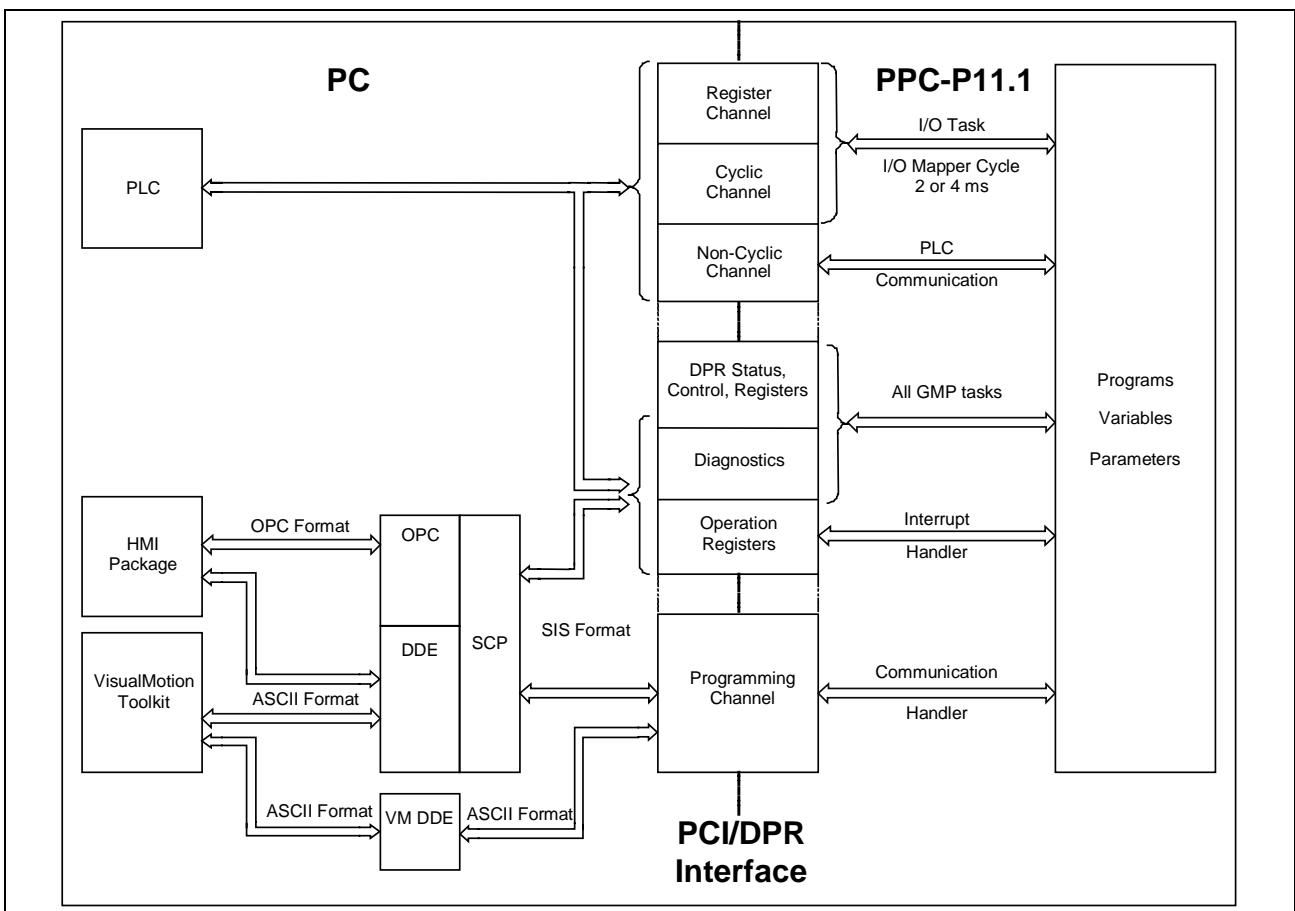


Fig. 9-2: Dual Port RAM Interface

DPR Parameters

The following tables list the parameters used to enable, configure and monitor the DPR interface between the PPC-P11.1 and the PC. Refer to chapter 5 for a detailed description of VisualMotion parameters.

PCI Communication

The following parameters enable communication across the PCI bus.

Parameter	Description
C-0-2635	PLC Error Reaction
C-0-2640	PLC Connection Options

Table 9-5: PCI Communication Parameters

Cyclic Channel

Parameter	Description
C-0-2600	PLC Mapper (cyclic channel) to PLC
C-0-2601	PLC Mapper (cyclic channel) from PLC
C-0-2632	PLC Multiplex Method
C-0-2636	PLC Word Swap
C-0-2638*	Fieldbus Available Cyclic IN Parameters
C-0-2639*	Fieldbus Available Cyclic OUT Parameters
*Note: Read-only parameters list of valid parameters that can be used in C-0-2600/2601	

Table 9-6: Cyclic Channel Parameters

Register Channel

Parameter	Description
C-0-2641	PLC Input Register List
C-0-2642	PLC Output Register List

Table 9-7: Register Channel Parameters

PLC Monitoring

Parameter	Description
C-0-2637	PLC Firmware Version
C-0-2643	PLC Lifecounter Check: Number of Retries
C-0-2644	PLC Lifecounter Check: Current Number of Misses
C-0-2645	PLC Lifecounter Check: Peak Number of Misses
C-0-2646	PLC Lifecounter Check: Number of Timeouts

Table 9-8: PLC Monitoring Parameters

Cyclic Channel Monitoring

Parameter	Description
C-0-2611	PLC Cyclic Channel: Current number of misses
C-0-2612	PLC Cyclic Channel: Peak number of misses
C-0-2613	PLC Cyclic Channel: Timeout counter

Table 9-9: Cyclic Channel Monitoring Parameters

Register Channel Monitoring

Parameter	Description
C-0-2651	PLC Register Channel: Current number of misses
C-0-2652	PLC Register Channel: Peak number of misses
C-0-2653	PLC Register Channel: Timeout counter

Table 9-10: Register Channel Monitoring Parameters

Shared DPR Memory Map

The following table describes the shared DPR memory area for communication between a PLC and the PPC-P11.1. All listed addresses are offset from the assigned PCI base address.

DPR Data		Address		Length	Direction		Comments
		Start (Hex)	End (Hex)	Bytes	PPC-P11.1	PLC	
PLC Cyclic Outputs	PPC Read Registers	0x4000	0x40FF	256	Read	Write	Defined by C-0-2642
	Cyclic Data Channel	0x4400	0x447F	128	Read	Write	Defined by C-0-2601; 64 words with multiplexing
PLC Cyclic Inputs	PPC Write Registers	0x4B00	0x4BFF	256	Write	Read	Defined by C-0-2641
	Cyclic Data Channel	0x4F00	0x4F7F	128	Write	Read	Defined by C-0-2600; 64 words with multiplexing
Non-Cyclic Channel	PLC Request	0x5800	0x590F	272	Read	Write	Modified SF3 or SIS
	PPC Response	0x5A00	0x5B0F	272	Write	Read	
PPC-P11.1 Diagnostic Information	Error_Code	0x5E74	0x5E75	2	Write	Read	C-0-0123
	Diagnostic_Text	0x5E7C	0x5EB7	60	Write	Read	C-0-0122 ; C-0-0124
	MC_Mode	0x5EB8	0x5EB9	2	Write	Read	C-0-0121 (SERCOS phase)
Status, Control Handshaking	PLC_Stat	0x5F00	0x5F01	2	Read	Write	
	PLC_Cmd	0x5F02	0x5F03	2	Read	Write	
	PLC_Count	0x5F04	0x5F05	2	Read	Write	
	Reserved	0x5F06	0x5F09	4	Read	Write	
	PLC_Clock	0x5F0A	0x5F0D	4	Read	Write	writes to C-0-0126 when value changes PLC can set date/time on PPC-P11.1
	MC_Stat	0x5F20	0x5F21	2	Write	Read	
	MC_Response	0x5F22	0x5F23	2	Write	Read	
	MC_Count	0x5F24	0x5F25	2	Write	Read	
	Out_PLC	0x5F40	0x5F40	1	Read	Write	
	Out_MC	0x5F41	0x5F41	1	Write	Read	
	In_PLC	0x5F42	0x5F42	1	Read	Write	
	In_MC	0x5F43	0x5F43	1	Write	Read	
	INT_REQ_OS	0x5F4C	0x5F4C	1	No access	Write	
	INT_REQ_RTOS	0x5F4D	0x5F4D	1	No access	Write	
	INT_RES_OS	0x5F4E	0x5F4E	1	No access	Write	
	INT_RES_RTOS	0x5F4F	0x5F4F	1	No access	Write	
	PLC_Ident	0x5FA0	0x5FC7	40	Read	Write	
	MC_Ident	0x5FC8	0x5FEF	40	Write	Read	
	PLC_Phase	0x5FF0	0x5FF1	2	Read	Write	
	MC_Phase	0x5FF2	0x5FF3	2	Write	Read	
PLC_Result	0x5FF4	0x5FF4	1	Read	Write		

Table 9-11: Shared DPR Memory

9.3 Register Channel

The register channel of the DPR is used to transmit VisualMotion register bits between the PLC and the PPC-P11.1. GMP9 firmware supports 1024 registers. Up to 128 registers can be written to each PLC Input (C-0-2641) and Output (C-0-2642) Register List. Each register is 16 bits (2 bytes) in length. Data consistency of all configured registers is transmitted to the DPR (I/O Task) every I/O Mapper cycle (2 or 4 ms).

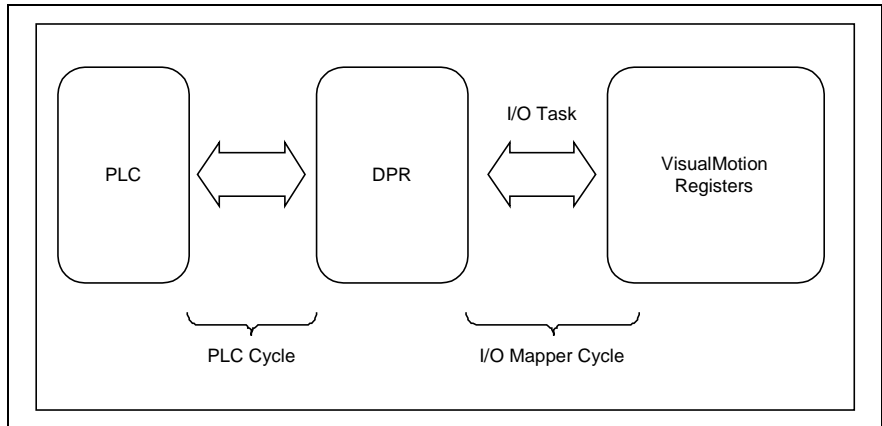


Fig. 9-3: Register Channel

Register Channel Transmission

The following table shows register transmission across the DPR.

Access		Direction	Length
PLC Input (Read)	PPC-P11.1 Output (Write)	PLC ← PPC-P11.1	256 Bytes
PLC Output (Write)	PPC-P11.1 Input (Read)	PLC → PPC-P11.1	256 Bytes

Table 9-12: PLC Input and Output Register List

Register Channel Configuration

The register channel is configured using VisualMotion Toolkit's Parameter Overview Tool. The registers configured in the Register Channel are transmitted in the order in which they appear in the PLC Input and Output Register Lists. Use the following steps to add registers:

1. Launch VisualMotion Toolkit in online or service mode, switch the control to parameter mode and select **Data ⇒ Parameters...**
2. Double click on either register list (C-0-2641 or C-0-2642).
3. Right click in the *Edit* window and select **Append Item (CTRL+INS)**
4. Add register numbers in the desired order of transmission.

Note: Read-only status registers must be added to the PLC Input Register List C-0-2641.

9.4 Cyclic Channel

The cyclic channel is used to transmit system parameters and program variables between the PLC and PPC-P11.1. Data transfer is limited to 64 words, with optional multiplexing for up to 32 levels. Each parameter and variable is considered a 32-bit double word in length. Each register is 16 bits (2 bytes) in length. Data consistency of all configured parameters and variables is transmitted to the DPR every I/O Mapper cycle (2 or 4 ms).

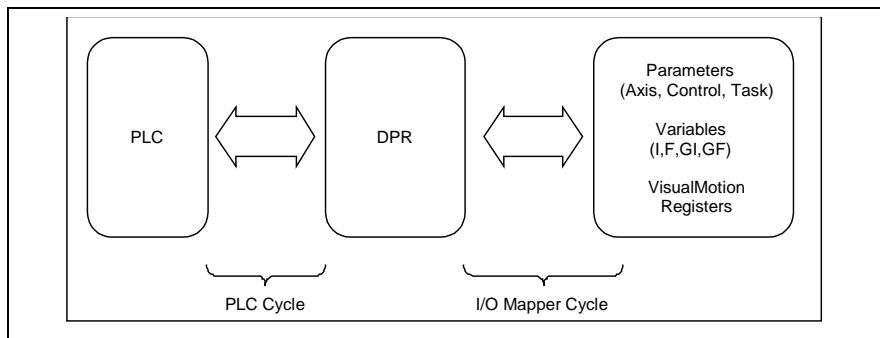


Fig. 9-4: Cyclic Channel Overview

Accessing Drive Parameters

Drive parameter (S and P) should be transmitted non-cyclically due to the inherent delay of parameter access over the SERCOS service channel. However, if a drive parameter is mapped to an axis parameter, then the axis parameter can be transmitted as cyclic data. Control parameters C-0-2638 and C-0-2639 are lists of allowable parameters that can be mapped. Refer to axis parameters A-0-0180 through A-0-0196 in chapter 5 for details.

Cyclic Channel Transmission

The following table shows data access across the DPR.

Access		Direction	Max. Length
PLC Input (Read)	PPC-P11.1 Output (Write)	PLC ← PPC-P11.1	128 Bytes
PLC Output (Write)	PPC-P11.1 Input (Read)	PLC → PPC-P11.1	128 Bytes

Table 9-13: Cyclic Channel DPR

Cyclic Channel Configuration

The cyclic channel is configured using VisualMotion Toolkit's Data Mapper (Fieldbus Mapper). The parameters and variables configured in the cyclic channel are transmitted in the order in which they appear in C-0-2600 and C-0-2601. Refer to the fieldbus interface chapters in the VisualMotion 9 Application Manual for *Cyclic Data Configuration* details.

Note: When multiplexing, a maximum of 31 parameters and/or variables is allowed per level. Status and control words are 16 bits in length and are located at the end of configured data. Refer to the Profibus fieldbus interface chapter in the *VisualMotion 9 Application manual* for multiplexing details.

Handshaking

The DPR interface between the PLC and the PPC-P11.1 verifies access to cyclic channel memory areas before reading or writing data (handshaking).

The PLC sets its request register before checking to see if the PPC-P11.1 is using the memory area. If the DPR is busy, it keeps its request active until the memory area is available and then accesses the DPR.

The PPC-P11.1 will always check to see if the DPR is busy before setting its request bit for the DPR. If the PLC is currently accessing the DPR, the PPC-P11.1 waits and tries once more in the same I/O Mapper cycle. If the PPC-P11.1 is locked out, a missed cycle counter is incremented. After 10 consecutive missed cycles, an error or warning is issued based on the configured PLC Error Reaction parameter C-0-2635. The following control parameters are used to monitor handshaking conflicts.

Parameter	Description
C-0-2611	PLC Cyclic Channel: Current # of misses
C-0-2612	PLC Cyclic Channel: Peak # of misses
C-0-2613	PLC Cyclic Channel: Timeout Counter

Table 9-14: Handshaking Monitor Parameters

Note: If the error reaction is set to ignore errors, then no error or warning is issued, but the timeout counter (C-0-2613) is incremented.

Handshaking State Machine

The following tables contain the PLC and PPC-P11.1 (MC) registers used for cyclic channel handshaking.

Name	Set By	Memory Area	Address Offset	Length
OUT_PLC	PLC	PLC Outputs	0x5F40	1 Byte
OUT_MC	PPC-P11.1		0x5F41	1 Byte
IN_PLC	PLC	PLC Inputs	0x5F42	1 Byte
IN_MC	PPC-P11.1		0x5F43	1 Byte

Table 9-15: Handshaking Registers

Handshaking Register Value	Description
0x00	Free area
0x80	Busy area

Table 9-16: Handshaking Register Value

The following figure flowcharts an example of the PLC Input Area.

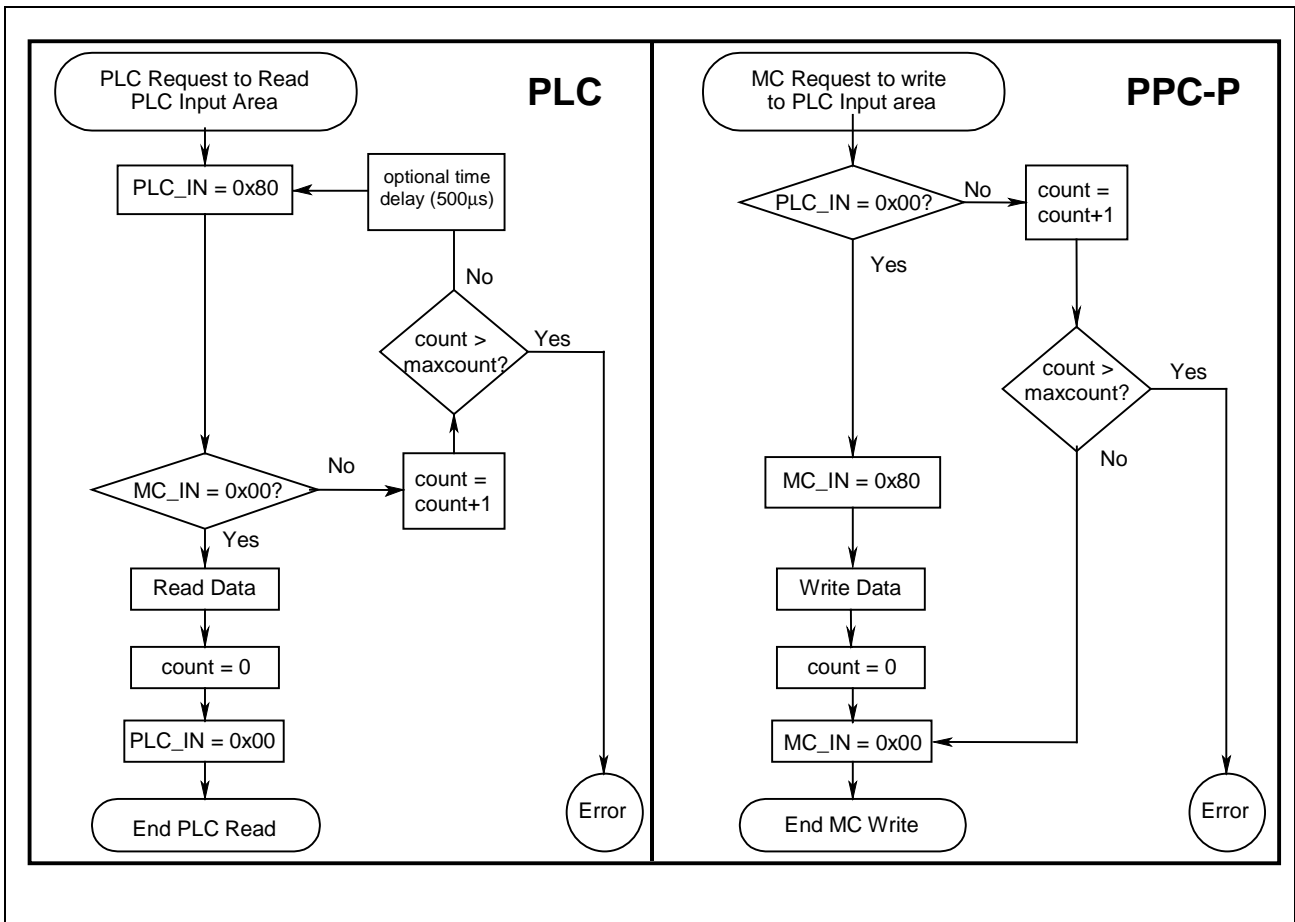


Fig. 9-5: Handshaking State Machine

The PLC and PPC-P11.1 counters, in the above figure, are internal to each processor and increment every time the processor tries to access the DPR while in use. The counters are reset to 0 every time the read or write action is successfully completed.

Note: The PLC should allow a time delay for the PPC-P11.1 to release the PLC Input memory area by setting IN_MC = 0x00, or set the loop *maxcount* to a large value.

9.5 Non-Cyclic Channel

The non-cyclic channel is used to transmit data that is not required every cycle, such as:

- parameter lists
- parameterization of axes or programs
- non-cyclically mapped data

List parameters must be transmitted non-cyclically due to their format. Drive parameter (S and P) must be transmitted non-cyclically due to the inherent delay of parameter access over the SERCOS service channel.

The PPC-P11.1 is notified of a non-cyclic channel request with an interrupt and mailbox message from the PLC. A response is returned from the PPC-P11.1 with another interrupt and mailbox message.

The non-cyclic channel supports the following two protocols:

Protocol	Description
SIS	standard Bosch Rexroth binary protocol using the Scalable Communication Platform (SCP)
Short Format 3 (SF3a)	modified version of Short Format 3

Table 9-17: Supported Protocols

Refer to the fieldbus interface chapters in the VisualMotion 9 Application Manual for Short Format 3 details. Refer to chapter 12.4 for SIS details

Protocol Identification

SIS allows access to all data. SF3a allows direct access to all non-list parameter values and can access lists, attributes, max/min values, etc., with ASCII protocol embedded in the data exchange objects. The first byte of the PLC Request identifies the protocol used.

Protocol	Value of First Byte
SIS	0x02
SF3a	0x20 - 0x5E

Table 9-18: Protocol Identification

Non-Cyclic Channel Transmission

The following table shows data access across the DPR.

DPR Data		Address	Direction	Length
PLC Request	(PPC-P11.1 Read)	0x5800	PLC → PPC-P11.1	272 Bytes
PPC Response	(PPC-P11.1 Write)	0x5A00	PLC ← PPC-P11.1	272 Bytes

Table 9-19: Non-Cyclic Channel DPR Location

Operating Procedure of the Non-Cyclic Channel

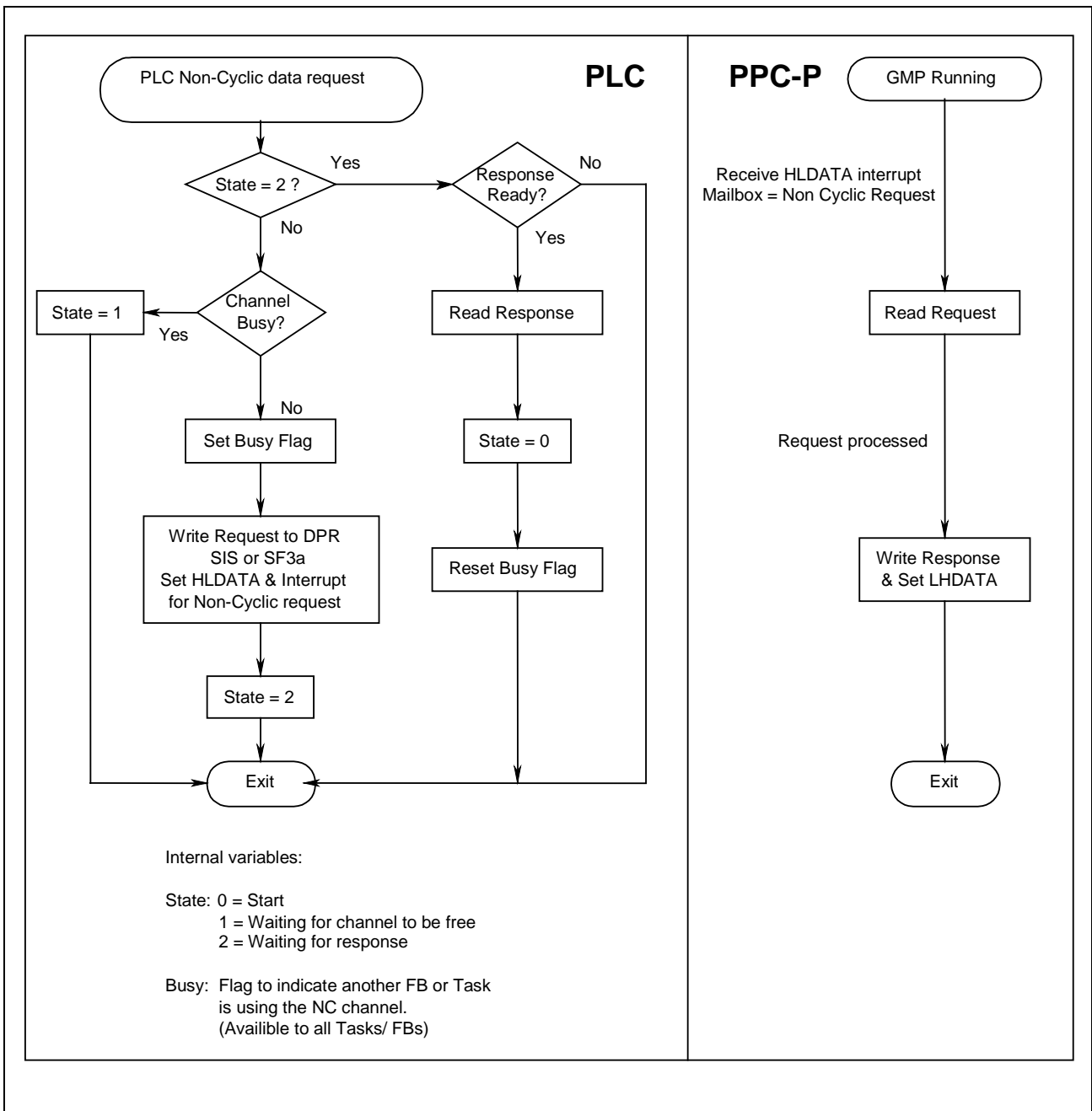


Fig. 9-6: Non-Cyclic Channel Operating Procedure

9.6 PPC-P11.1 Diagnostic Information

Error_Code

This parameter contains the current system status or error message number issued by the control.

Control parameter C-0-0123, Diagnostic Code is displayed.

If the PLC interface is initialized, the update rate is based on the I/O task (2 or 4 ms).

Diagnostic_Text

Control parameters C-0-0122 Diagnostic Message, and C-0-0124 Extended Diagnostic Message are separated by a semicolon. The combined message is truncated to 60 bytes.

Diagnostic_Text is generated by the following rules:

1. If no extended message exists, C-0-0124 = 0, then Diagnostic_Text contains C-0-0122.
2. If an extended message exists, the first 30 characters in Diagnostic_text contain the first 30 characters of C-0-0122, followed by a semicolon (;), then followed by 29 characters of the extended message. In this case the message might not be complete.

For example: 412 No drives found on ring; CP0: Ring not closed

If the PLC interface is initialized, the update rate is based on the I/O task (2 or 4 ms).

MC_Mode

This parameter displays the current SERCOS initializing phase of the control. The PPC-P11.1 mirrors C-0-0121 Current SERCOS Phase in the DPR location.

If the PLC interface is initialized, the update rate is based on the I/O task (2 or 4 ms).

9.7 Status and Control Registers

The following descriptions are for the Status and Control register of the DPR. Refer to Table 9-11: Shared DPR Memory for address offsets.

PLC_Stat

The following table describes the PLC status register.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	not used							Run	not used					Error	Ready	

Table 9-20: PLC Status Register

Bit	Name	Description	Active	Actions
0	Ready	Indicates hardware/Firmware ready for startup	1	
1	Error	Indicates error on PLC system	1	Clear Ready and Running bits
7	Run	Indicates phase up complete; PLC Running	1	Start PLC program; Start life counter

Table 9-21: PLC Status Register Bit Description

PLC_Cmd

Following a request for PLC initialization, the PLC indicates which initialization sequence to startup. After a correct response is received from the PPC-P11.1 in MC_Response, this register is reset to 0x0000.

Value	Description
0x0000	No active command
0x0001	Request interface shutdown for new PLC program download
0x0002	Request MC startup initialization sequence

Table 9-22: PLC Command Register

PLC_Count

This 16-bit PLC life counter is checked by the PPC-P11.1 every I/O Mapper cycle (In the I/O Task) to see if the PLC is active. This counter is written to on every PLC cycle.

PLC_Clock

The PLC_Clock register is written to by the PLC. The control reads this value and writes it to control parameter C-0-0126. The following table describes the PLC clock register.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	Hours					Minutes						Seconds in 2s increments				

Table 9-23: Low Word Bit Description

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Description	Day					Month					Year, relative to 1980					

Table 9-24: High Word Bit Description

MC_Stat

The following table describes the motion control status register.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	not used							Run	not used			/Download	Error	Ready		

Table 9-25: Motion Control Status Register

Bit	Name	Description	Active	Actions
0	Ready	Indicates hardware/firmware ready for further startup	1	
1	Error	Indicates error on PPC-P11.1 (Mirror of PPC-P11.1 Error Bit)	1	Clear Ready and Running bits
2	/Download	1 = Download not possible in Phase 3,4 0 = Allow PLC to download new programs when the PPC-P11.1 is in P2.	1 0	
7	Run	Phase up complete; PPC-P11.1 is Running	1	Start life counter

Table 9-26: Motion Control Status Register Bit Description

MC_Response

During the handshaking sequence, the PLC indicates which Handshaking sequence to start by writing 0x0001 or 0x0002 to the PLC_Cmd register after an interrupt and the initialization mailbox message.

The PPC-P11.1 acknowledges that it is following the correct Handshaking sequence by displaying the same value in MC_Response that the PLC enters in PLC_Cmd.

Value	Description
0x0000	No active command
0x0001	Request interface shutdown for new PLC program download
0x0002	Request MC startup initialization sequence

Table 9-27: MC_Response

MC_Count

This 16 bit counter is incremented every I/O Mapper cycle by the PPC-P11.1. The PLC uses this counter as verification that the motion control is capable of communication. The counter starts after a successful initialization sequence by the PLC.

Out_PLC

This Handshaking register is written to by the PLC when requesting to write to the PLC_Output area (Both Register and Cyclic Data Channels). This register output is read by PPC-P11.1

Value	Description
0x00	Input/output area of DPR free
0x80	PLC accessing DPR

Table 9-28: PLC Output Handshaking Register

Out_MC

Handshaking register written by the PPC-P11.1 when requesting to read from the PLC_ Output area (Both Register and Cyclic Data Channels). Read by PLC.

Value	Description
0x00	Output area of DPR free
0x80	PPC-P11.1 accessing DPR

Table 9-29: Control Output Handshaking Register

In_PLC

Handshaking register written by the PLC when requesting to read from the PLC_Input area (Both Register and Cyclic Data Channels). Read by PPC-P11.1.

Value	Description
0x00	Input/output area of DPR free
0x80	PLC accessing DPR

Table 9-30: PLC Input Handshaking Register

In_MC

Handshaking register written by the PPC-P11.1 when requesting to write to the PLC_ Input area (Both Register and Cyclic Data Channels). Read by PLC.

Value	Description
0x00	Input area of DPR free
0x80	PPC-P11.1 accessing DPR

Table 9-31: Control Input Handshaking Register

INT_REQ_OS

Handshaking register written by the SCP when setting a PCI bus interrupt. (Read by the PLC interface)

Value	Description
0x00	PCI host interrupt free
0x80	SCP accessing DPR

Table 9-32: Operating System Handshaking Request

INT_REQ_RTOS

Handshaking register, based on PLC interface, when requesting a PCI bus interrupt to the PPC-P11.1. (Read by the SCP)

Value	Description
0x00	PCI host interrupt free
0x80	PLC interface register / PCI host interrupt

Table 9-33: Real-time Operating System Handshaking Request

INT_RES_OS

Handshaking register written by the SCP when reading a PCI bus interrupt response from the PPC-P11.1. (Read by PLC interface)

Value	Description
0x00	Response free
0x80	SCP accessing DPR

Table 9-34: Operating System Handshaking Response

INT_RES_RTOS

Handshaking register, based on the PLC interface, when reading a PCI bus interrupt response from the PPC-P11.1. (Read by the SCP)

Value	Description
0x00	PCI response interrupt free
0x80	PLC interface request PPC-P11.1 response interrupt

Table 9-35: Real-time Operating System Handshaking Response

PLC_Ident

This string displays the customer-defined identification for the PLC. This ID is read and displayed in control parameter C-0-2637.

MC_Ident

Optionally used during initialization by the PLC, this string displays the installed firmware version in the PPC-P11.1.

For example: FWC-PFM01*-GP*-09V00

PLC_Phase

Indicates what phase the PLC is in during the Start-up initialization sequence. The following table contains a brief overview of the PLC actions during these phases.

Value	PLC Action During this Phase
0x0000	Clear shared memory area in DPR (0x4000 – 0x5FFF) Set PLC_Stat ready (bit 0) high Set PLC start-up command (PLC_cmd = 0x0002)
0x0001	Check MC_Ident for valid string
0x0002	Set PLC_Stat Run (bit 7) high Start PLC program Start PLC_Count

Table 9-36: PLC Phase

MC_Phase

Indicates what phase the PPC-P11.1 is in during the Start-up initialization sequence. The following table contains a brief overview of the PPC-P11.1 actions during these phases.

Value	PPC-P11.1 Action During this Phase
0x0000	Write MC_Ident
0x0001	Set MC_Stat Ready (bit 0) high
0x0002	Set MC_Stat Run (bit 7) high Start MC_Count

Table 9-37: Control Phase

PLC_Result

(Optional)

PLC writes 0x0001 after comparing MC_Ident to the valid firmware ID strings.

9.8 Operation Registers

The operation registers are used to set interrupts between the Host (PC) and the Local (PPC-P11.1) processor and for initialization of the PPC-P11.1/PLC interface, non-cyclic communication and the SCP interface. All addresses listed in the table below are offsets from the assigned PCI base address.

PCs using both a real-time operating system (RTOS) and a Windows based application should enable local to host mailbox interrupts using only the Windows interface. Sharing of the interrupt vector between two operating systems is not possible. The RTOS will poll the LHDATA mailbox for response to its interrupts set on the PPC-P11.1.

Operation Register	Address Offset		Length
	Start (Hex)	End (Hex)	Bytes
Host Interrupt Control/Status Register (HINT)	0x04E4	0x04E7	4
Host to Local Data Mailbox (HLDATA)	0x04E8	0x04EB	4
Local Processor Interrupt Control/Status Register (LINT)	0x04F4	0x04F7	4
Local to Host Data Mailbox (LHDATA)	0x04F8	0x04FB	4

Table 9-38: DPR Map

HINT - Host Interrupt Control and Status Register

The HINT operation register is used by the SCP and PPC-11.1 for setting a Local to Host Interrupt. This operation register works in conjunction with the LHDATA operation register mailbox. When a request for interrupt is set by the local (LHDATA, bit 24), bit 3 of the HINT interrupt status is enabled. The host then initiates the interrupt by enabling bit 19. After the interrupt is enabled, the contents of the LHDATA mailbox are read.

Note: All control and status bits are initially cleared on power-up.

Function	Address Offset	Bit	Description
Interrupt Event Status	0x04E4	3	0 = no events active (<i>default</i>) 1 = Local to Host mailbox
Interrupt Enable		19	0 = no interrupts are enabled (<i>default</i>) 1 = Local to Host mailbox interrupt enabled

Table 9-39: Host Interrupt Control and Status Register - HINT

LINT - Local Interrupt Control and Status Register

The LINT operation register is used by the PPC-P11.1 for setting a Host to Local Interrupt. This operation register works in conjunction with the HLDATA operation register mailbox. When a request for interrupt is set by the host (HLDATA, bit 24), bit 3 of the LINT interrupt status is enabled. The local then initiates the interrupt by enabling bit 19. After the interrupt is enabled, the contents of the HLDATA mailbox are read.

Note: All control and status bits are initially cleared on power-up.

Function	Address Offset	Bit	Description
Interrupt Event Status	0x04F4	3	0 = no events active (<i>default</i>) 1 = Host to Local mailbox
Interrupt Enable		19	0 = no interrupts are enabled (<i>default</i>) 1 = Host to Local mailbox interrupt enabled

Table 9-40: Host Interrupt Control and Status Register - HINT

Interrupt Mailboxes

The PPC-P11.1 uses the Host-to-Local and Local-to-Host Data Mailboxes in the Operation Registers to facilitate the interrupts sent from the PC (Host) to the PPC-P11.1 (Local) and in the other direction (Local to Host). Byte 0 and byte 1 contain the mailbox message read by the target processor to determine the reason for the interrupt.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	Mailbox High Byte								Mailbox Low Byte							

Table 9-41: Low Word Bit Description

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Description	not used							IRQ	not used							

Table 9-42: High Word Bit Description

LHDATA - Local to Host Data Mailbox

The LHDATA contains the requested local interrupt message. Bit 24 enables the local to send an interrupt to the host. When enabled, a mailbox interrupt is sent to the host processor. The interrupt remains active until cleared by the host in the HINT - Host Control and Status Register.

Function	Address Offset	Bit	Description
Data Word	0x04F8	7-0 (byte 0) 15-8 (byte 1)	Interrupt message written by the local and read by the host processor.
Interrupt to Host		24	0 = inactive 1 = active

Table 9-43: Host Interrupt Control and Status Register - HINT

HLDATA - Host to Local Data Mailbox

The HLDATA contains the requested host interrupt message. Bit 24 enables the host to send an interrupt to the local. When enabled, a mailbox interrupt is sent to the local processor. The interrupt remains active until cleared by the local in the LINT - Host Control and Status Register.

Function	Address Offset	Bit	Description
Data Word	0x04E8	7-0 (byte 0) 15-8 (byte 1)	Interrupt message written by the host and read by the local processor.
Interrupt to Local		24	0 = inactive 1 = active

Table 9-44: Host Interrupt Control and Status Register - HINT

Interrupt “Mailbox” Message Descriptions

Interrupt Requested	Value	Description
PLC initialization	0x0100	PLC has requested to initialize or stop communication with the PPC-P11.1 depending on whether PLC_Cmd = 0x0002 (initialize) or 0x0001(stop)
Non-Cyclic Message	0x0008	PLC requesting information over non-cyclic channel
Program Channel Message	0x0020	Request sent to PPC-P11.1 over Programming channel from PC user interface

Table 9-45: Interrupt Message Description

Clearing Interrupt Bits

Name	Address	Description
HINT	0x04E4	Host Interrupt Control/Status: Host to Local interrupt is cleared with a write to the Local Interrupt Control and Status Registers
LINT	0x04F4	Local Interrupt Control/Status: Local to Host interrupt is cleared with a write to the Host Interrupt Control and Status Registers

Table 9-46: Clearing Interrupt Bits

9.9 Interrupt Handshaking

PCI Interrupt to the PPC-P11.1

The flowchart below shows the procedure for setting a PCI interrupt to the PPC-P11.1.

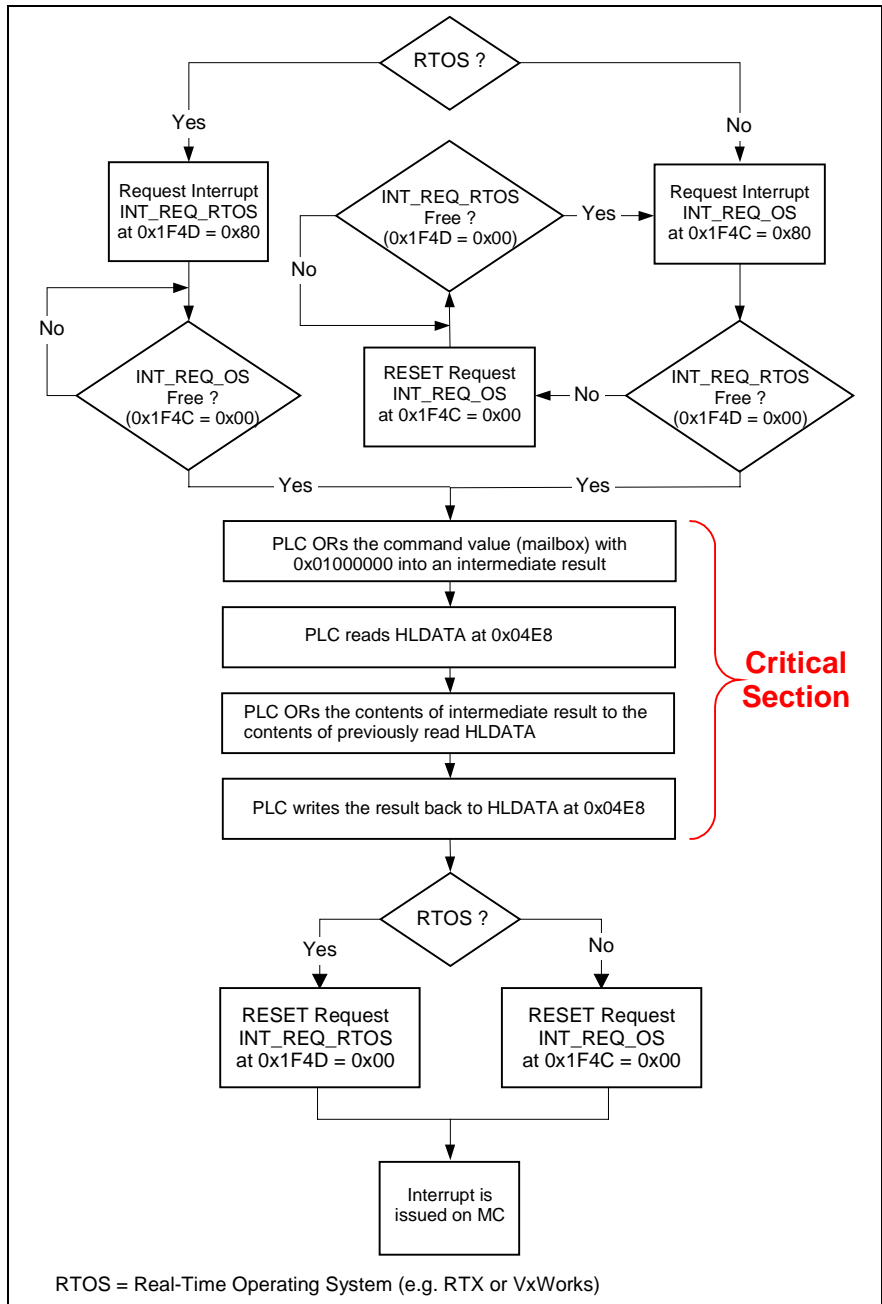


Fig. 9-7: PCI Interrupt to the PPC-P11.1

Receiving Interrupt Response from PPC-P11.1

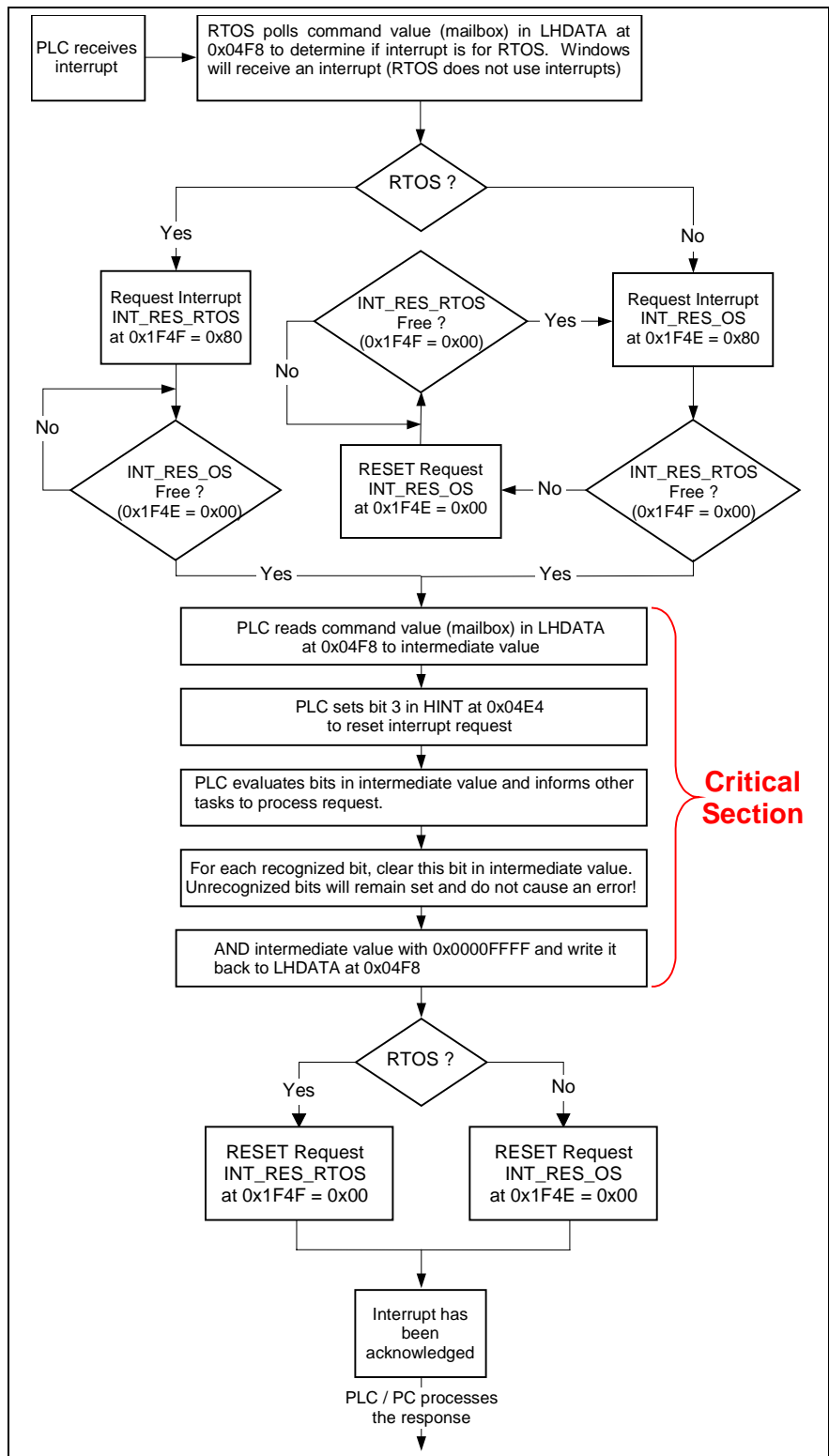


Fig. 9-8: Receiving Interrupt Response from PPC-P11.1

Enabling the PCI Interrupts

The enabling of PCI interrupts is performed by the Windows application (SCP or VM_DDE) and not by the real-time operating system (RTOS) application.

In the RTX application, only clear the handshaking registers and mailbox if the Windows application did not start. The mailbox enable in the HINT register is checked.

In the event that an application is running in a RTOS and the Windows based application has not started, the RTOS application should initialize the mailbox and handshaking registers.

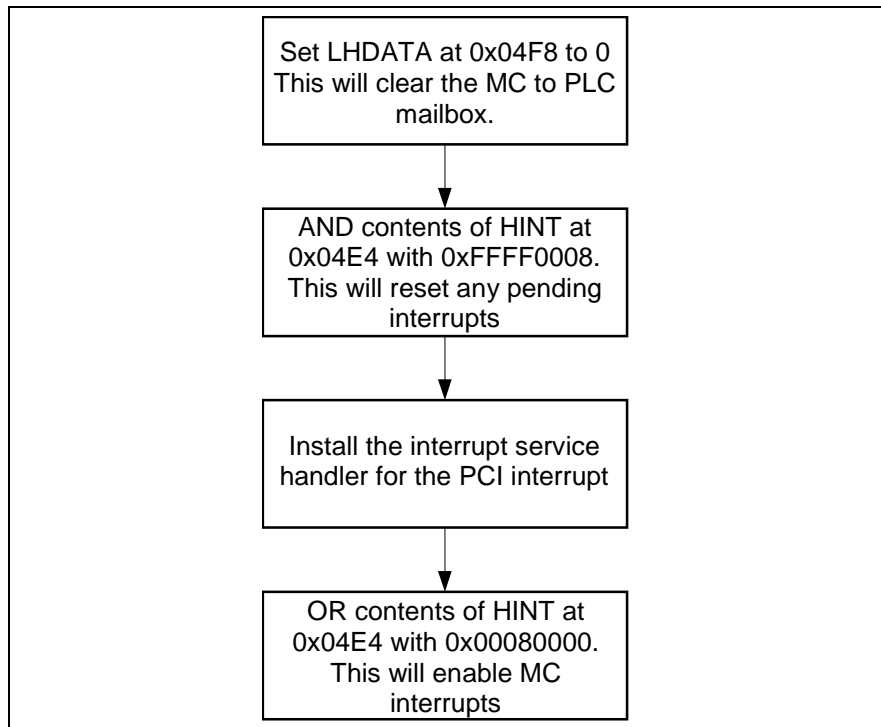


Fig. 9-9: Enabling the PCI Interrupts

9.10 System Initialization

PLC to PPC-P11.1 Communication

The following steps are used to initialize communication between a PLC and the PPC-P11.1:

1. Locate the PPC-P11.1 on the PCI Bus using the Vendor and Device IDs.
2. Map PPC-P11.1 Dual Port RAM areas to Virtual Memory on the PC. Most operating systems do not allow direct access from application programs to the actual hardware memory addresses. The operating system has standard functions in its API, which will handle this mapping. (RTX example: RtMapMemory)
3. Initialize communication between the PLC and PPC-P11.1 using the handshaking method previously described.
4. The PLC should monitor the PPC-P11.1's life counter while running its own life counter to maintain the connection. The easiest way to do this is to mirror back the PPC-P11.1's life counter as the PLC life counter. As long as the lifecounters run, the data in the register, cyclic and non-cyclic channels will be valid. To maintain data integrity on the cyclic channel, use the handshaking registers.

PLC / PPC-P11.1 Initialization Sequence

PLC Phase	Control Phase	Time	PLC	Control (MC)
0	0	0	[PLC starts] PLC checks and clears DPR (0x4000...0x5FFF) PLC sets PLC_Stat, Ready = 1 (Bit 0) PLC sets PLC_Phase = 1 "Start Up Command to MC (PLC_CMD = 0x0002) PLC sets IRQ, Mailbox value: 0x0100	Internal tests MC Resets IRQ; Waits for PLC_Phase = 1 [Indicates Waiting For PLC if PLC communication option is set after initialization]
1	0	1	Waits for MC_Phase = 1 Timeout: 20s	MC copies "FWC-PFM01*-GP*-09VRS-MS" into MC_Ident MC sets MC_Phase = 1 MC sets MC_Stat Ready =1 (Bit 0)
1	1	2	(Optional) PLC checks for MC_id If OK: PLC sets PLC_Phase = 2 Else Incorrect firmware error If do not check ID: PLC sets PLC_Phase = 2	Waits for PLC_Phase = 2
2	1	3	Waits for MC_Stat, Ready = 1 (Bit 0) Timeout 1s	MC sets MC_Stat Ready = 1 (Bit 0) MC sets MC_Phase = 2 MC sets MC_Stat, Run = 1 (Bit 7) Start Life Counter
2	2	4	PLC starts PLC program PLC set PLC_Stat Run =1 (Bit 7) Start Life Counter Normal running	Normal running (Life Counter Active) If SERCOS Phase 2 or Lower: Set MC_Stat, /Download = 0 (Bit 8) Else Set MC_Stat, /Download = 1(Bit 8)

Table 9-47: PLC / PPC-P11.1 Initialization Sequence

PLC / PPC-P11.1 Program Download or Shutdown Sequence

When new programs are activated on the PLC, the life counter may not be updated in the required length of time causing a VisualMotion error. The life counter is ignored when the PLC indicates a program download or shutdown activation.

Note: Following a shutdown, the PPC-P11.1 is not allowed to phase up to phase 4 until a connection is re-initialized or PLC_CMD is not equal to 0x01. MC_Count is not updated.

PLC Phase	Control Phase	Time	PLC	Control (MC)
2	2	0	Normal running	Normal running
2	2	1	If MC_Stat Bit 2 "/Download" = 0 Clears PLC_Stat Run (Bit 7) "Shut Down Command" to MC (PLC_CMD=0x0001) PLC sets IRQ, Mailbox value: 0x0100 Else Error message "Download not possible" Remedy: Switch control in and out of parameter mode and try again.	Normal running: PLC_CMD = 0x0001 indicates program download, ignore PLC life counter if IRQ also received. Note: MC_Stat bit 2 / Download (1 = SERCOS phase 3, 4. 0 = SERCOS phase 2 or lower)
2	2	2		MC leaves run-mode if necessary
0	0	3	re-initialize connection	

Table 9-48: PLC / PPC-P11.1 Program Download or Shutdown Sequence

10 Coordinated Motion

10.1 Standard Coordinated Motion

VisualMotion provides predefined kinematic libraries for controlling industrial robots in coordinated motion applications to achieve accurate high-speed positioning over geometric paths. Refer to Kinematic Library on page 10-13 for detailed descriptions of the supported kinematics.

The assignment of a Kinematic number is done in the *Task Axes Setup* icon within a VisualMotion program, as shown in Fig. 10-1.

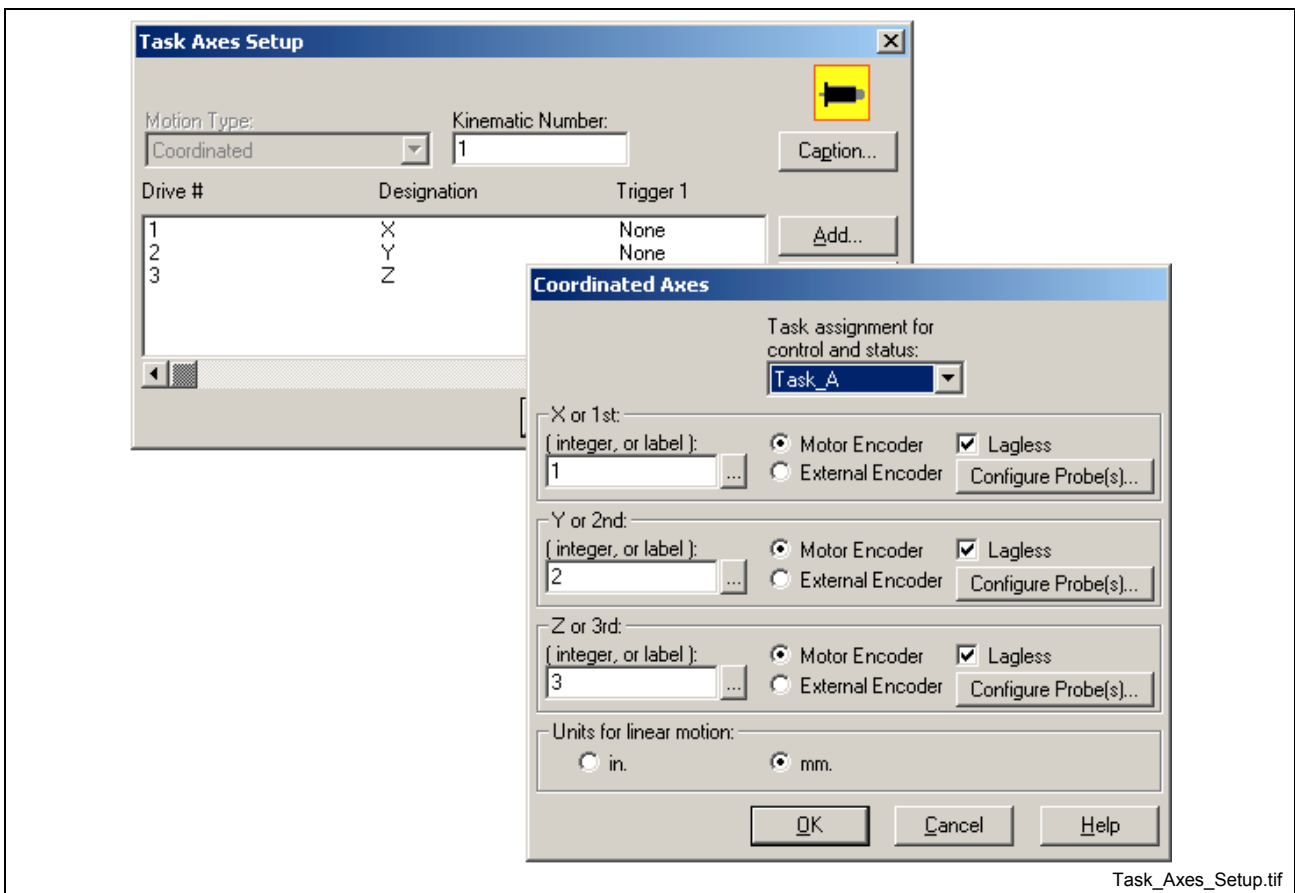


Fig. 10-1: Assigning a Kinematic (Standard Coordinated Motion)

Note: Motion type must be set to "Coordinated" and the kinematic number properly assigned. Coordinated axes numbers are entered using the drive's SERCOS address.

Associated Task Parameters

When a Kinematic (number) is assigned and downloaded to the control, the associated task parameters are given default values. The associated task parameters are:

- T-0-0010 ;Kinematic number
- T-0-0011 through T-0-0013 ;Coordinated axis X, Y and Z
- T-0-0050 through T-0-0059 ;Kinematic value 1 - 10

Task parameter T-0-0010 displays the assigned kinematic number in the axis icon. Task parameters T-0-0011 through T-0-0013 will display the SERCOS drive address of all configured axes in a specific kinematic.

Task parameters T-0-0050 through T-0-0059 represent segment lengths in a robotic arm. Any segment task parameter can be modified to match the exact segment length for a specific robot. Task parameters are modified by selecting **Data** ⇒ **Parameters** from VisualMotion's main menu and selecting the Task tab.

Normal Case Kinematics

When using kinematics, make certain that all the mechanical settings for each axis corresponds to the physical requirements for the robot in use. Each kinematic is assigned default values for each segment in a robot's design (K values). These K values can be modified by the user to meet the requirements of their particular robot. The unit of measurement for each kinematic is defined in parameters A-0-0005 and T-0-0005. Feed constant (k) and gear ratios for each axis are unique to each machine and are defined in the *Drive n Mechanical* window. The exceptions to normal case kinematics are defined under Special Case Kinematics.

Special Case Kinematics

To maximize positioning resolution for kinematics 2, 4, 5 and 9, the user should set the mechanical setting of each axis to the following selections.

Description	Selection	Parameter
Unit of measure for position data	degree	A-0-0005
Type of scaling	Linear	A-0-0004, bit 2
Feed constant k	6.283185 (2 π)	S-0-0123

Table 10-1: Mechanical Settings

10.2 Coordinated Articulation

Coordinated Articulation is an advanced feature in VisualMotion 9 used to move up to six axes in coordinated fashion based on world coordinate inputs from an ELS Group CAM output or manual positions. This feature provides the ability to link cyclic coordinated motion to an ELS master.

Differences from Normal Coordinated Motion

The following table outlines a comparison between standard coordinated motion and Coordinated Articulation. The following outlines the difference in features in Coordinated Articulation from standard Coordinated Motion:

- Points Table - No points table supported
- Motion Type - Coordinated and ELS Group
- Zones - No support for zones
- Number of Axes – up to 6 axes support
- Number of Coordinated Task - 2 total, only one can be assigned to any task A-D
- BTC06 Support - No BTC06 support
- Control Task Register
- Coordinated Limit Parameters
- Coordinated Motion Icons - No additional coordinated motion icon supported. Only Coordinated Articulation icon is used.

Applications using Coordinated Articulation

Coordinated Articulation is used for applications requiring the fast cyclic positioning output of coordinated axes for the moving or picking up of products. Applications examples are high speed transfer tables or cyclic robotic arms.

Block Functionality

This function can be visualized consisting of four types of blocks:

Note: Forward transformations refer to deriving world coordinates from axis positions. The forward transform is only for display purposes.

- A CAM section, taking ELS Group position and transforming it based on the CAM H factor, and Offset.
- A ramp generator, providing a trapezoidal profile for manual moves.
- A switch, selecting CAM output or manual moves for input to kinematic.
- A kinematic, converting between world coordinates and axis positions (Not all kinematics support forward transformations).

World Coordinated View

For each of the six world coordinates, x, y, z, roll, pitch, yaw (x, y, z, Φ , θ , ψ), a CAM ELS Group and manual trapezoidal ramp is provided.

In CA_Sync Mode, at zero position of the ELS Group output, the CAM number, H, and offset are copied. A check is made to insure the output will be within the minimum and maximum of the world coordinate input.

The offset must be equal or greater than the minimum; offset plus H factor must be less than maximum. If output is outside limits, a diagnostic warning is entered, and the output limited.

In CA_Sync Mode, at each SERCOS interrupt, the ELS Group output position is looked up in the selected CAM table and scaled with the H factor and the offset is added. If the CAM number is zero, the output of the CAM section will be the same as the offset.

If in CA_Local Mode and enabled, the ramp generator will use the linear/rotary programmed accel, decel, and velocity to move its world coordinate input to the target position.

In CA_Local Mode, axis CAM switching takes place immediately, no zero crossing of the master is required.

The switch directs the output of the CAM section or ramp generator to the world input of the kinematic section, or switches configured axis into single axis mode.

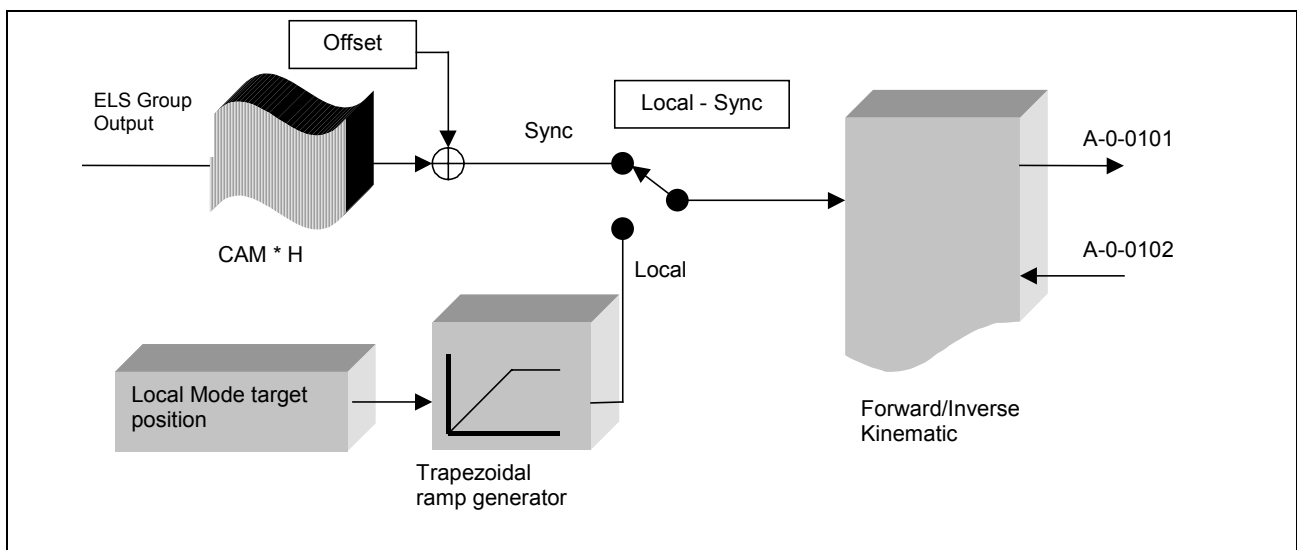


Fig. 10-2: Coordinated Articulated Axis Configuration

Trapezoidal Ramp Generator (Normal Local Mode)

The ramp generator is used for absolute moves when its task is in run.

The target position is checked against minimum and maximum limits at the start of the move (0->1 on Enable Ramp Generator). If the target position is outside the limits, a warning is issued and output limited. The target position, velocity, acceleration, and deceleration are copied at the start of the move; changing these values during the move will not affect the move once started.

An E-STOP, switch to "task manual" or "task stop" during the move will result in ramping down the velocity to zero using the deceleration value that was copied at the start of move.

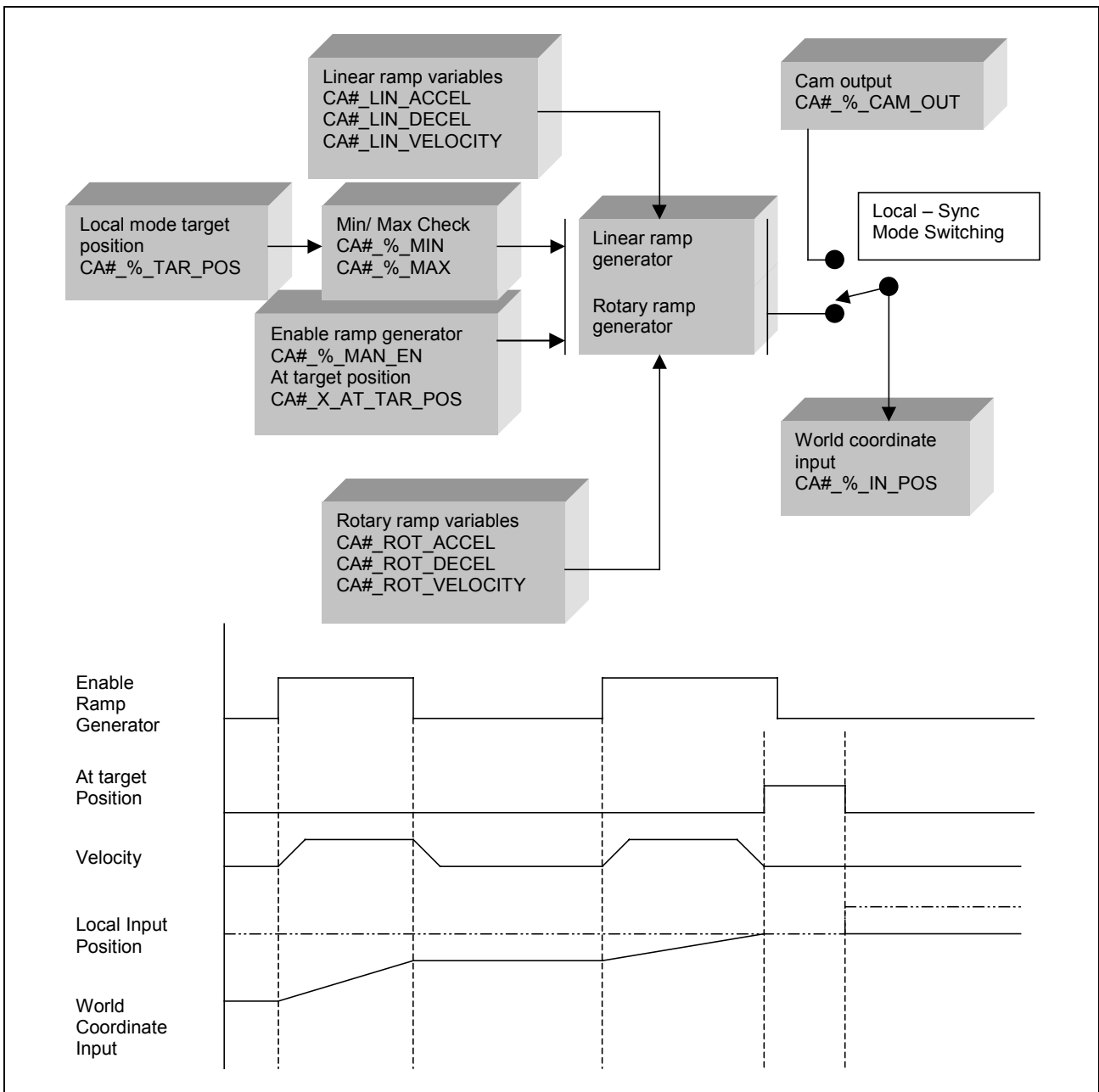


Fig. 10-3: Normal Local Mode

System Considerations

Axis Motion Type

Axis parameter A-0-0003 is set to axis motion type 9 (Coordinated Articulation Axis). This axis motion type has the following configuration (S-0-0258 & S-0-0047 in the MDT):

- Primary Operating Mode: Position Command
- Secondary Operating Mode 1: Single Axis
- Secondary Operating Mode 2: Velocity
- Secondary Operating Mode 3: Torque

Task Control of Coordinated Articulation Axes

All ELS functionality is controlled by task A. All Coordinated Articulation axes are controlled in the task they are declared. Assigned axes primary mode is single axis, secondary mode is position.

Initial State	New State	Comment
2 Parameter	5 Manual	"Single axis" mode
5 Manual	6 Automatic	"Single axis" mode
6 Automatic	7 Task Run	"Single axis" mode. Mode is changed to "position" with "ELS Mode" icon.
7 Task Run	6 Automatic	Axes remain in their current mode, "position" or "single axis". Icon in "CA_Local" mode, axes are ramped to 0 velocity. Icon in "CA_SYNC" mode, axes remain in ELS Group controlled motion.
6 Automatic	5 Manual	Axes remain in their current mode, "Position" or "Single axis". If icon in "CA_SYNC" Mode, ELS Group switches to "Local" mode and stops motion.
5 Manual	2 Parameter	"Single axis" mode

Table 10-2: Task Control

Feedback Support

The coordinated articulation function supports only the primary feedback device of the drive.

E-Stop of Coordinated Articulation Axes

Axes remain in their current mode, "position" or "single axis". The ELS Group ramps to a stop in "Local Mode" using stop deceleration.

Control and Status Registers

The following user defined control and status registers are used to control and monitor Coordinated Articulation. These control and status registers are independent of those assigned to Virtual Masters, System Masters and ELS Groups.

Control Registers

Two registers are used to set the mode of operation of an axis. The user assigns the first starting register and VisualMotion assigns the next consecutive register. Default labels are provided by VisualMotion.

Coordinated Articulation supports the following two modes of operation:

1. Synchronized Mode (Relative or Absolute)
2. Local Mode (Normal or Immediate)

Starting Control Register Number

A Coordinated Articulation configuration uses two control registers. One control register is used for synchronized mode and one for local mode. When a starting control register number is entered, the next consecutive number is used for the local mode control register. The following tables list the functions used in each control register.

Synchronized Mode Control Register The first control register selects the operation mode (sync or local) for each axis.

Register Bit	Label	Description
1	CA#_X_SYNC	<i>Synchronized Mode Bits</i> 0 -> 1 transition enables each axis to Sync mode. This transition is delayed until the ramp generator's velocity is 0. 1 -> 0 transition set each axis to Local Mode. Local mode is used for the manual positioning of each axis.
2	CA#_Y_SYNC	
3	CA#_Z_SYNC	
4	CA#_ROLL_SYNC	
5	CA#_PITCH_SYNC	
6	CA#_YAW_SYNC	
7	CA#_X_REL_MOD	<i>Positioning Mode Bits</i> Enable before setting Synchronized Mode Bit. 1 = Relative Mode 0 = Absolute Mode (default)
8	CA#_Y_REL_MOD	
9	CA#_Z_REL_MOD	
10	CA#_ROLL_REL_MOD	
11	CA#_PITCH_REL_MOD	
12	CA#_YAW_REL_MOD	

Table 10-3: Synchronized Mode Control Register

Local Mode Control Register The second control register enables the ramp generator in local mode for each axis. Local mode allows the user to manually position the world coordinate in-position value to the current Cam output position before synchronizing.

Register Bit	Label	Description
1	CA#_X_MAN_EN	<i>Local Mode (Manual) Bits [used when Sync mode is inactive]</i> Normal Local Mode: 0 -> 1 transition enables the ramp generator, creates a move profile for variable CA#_%_TAR_POS value to variable CA#_%_IN_POS. (where % = X, Y, Z, ROLL, PITCH or YAW) 1 -> 0 transition disable the ramp generator. If move is at velocity, the velocity is ramped down using the value in variable CA#_LIN_DECEL.
2	CA#_Y_MAN_EN	
3	CA#_Z_MAN_EN	
4	CA#_ROLL_MAN_EN	
5	CA#_PITCH_MAN_EN	
6	CA#_YAW_MAN_EN	
7	CA#_X_MAN_IMD	<i>Immediate Mode Bits [used when Sync mode is inactive]</i> Enable before setting Local Mode Bit. 1 = Immediate Mode 0 = Normal Local Mode
8	CA#_Y_MAN_IMD	
9	CA#_Z_MAN_IMD	
10	CA#_ROLL_MAN_IMD	
11	CA#_PITCH_MAN_IMD	
12	CA#_YAW_MAN_IMD	

Table 10-4: Synchronized Mode Control Register

Starting Status Register Number

A Coordinated Articulation configuration uses two status registers. One status register is used for synchronized mode and one for local mode. When a starting status register number is entered, the next consecutive number is used for the local mode status register. The following tables list the functions used in each status registers.

Synchronized Mode Status Register The first register is used to monitor the status of each axis enabled to synchronized mode. It also monitors the equivalence between variables CA#_%_IN_POS and CA#_%_TAR_POS (where % = X, Y, Z, ROLL, PITCH or YAW).

Register Bit	Label	Description
1	CA#_X_SYNCED	1 = In Sync Mode. Using CAM output world coordinated input value. 0 = In Local Mode. Use target position for world coordinated input value.
2	CA#_Y_SYNCED	
3	CA#_Z_SYNCED	
4	CA#_ROLL_SYNCED	
5	CA#_PITCH_SYNCED	
6	CA#_YAW_SYNCED	
7	CA#_X_READY	1 = CAM output value (CA#_%_CAM_OUT) equals world coordinated input value (CA#_%_IN_POS) and ramp generator is at 0 velocity. 0 = Not ready for synchronization
8	CA#_Y_READY	
9	CA#_Z_READY	
10	CA#_ROLL_READY	
11	CA#_PITCH_READY	
12	CA#_YAW_READY	

Table 10-5: Synchronized Mode Status Register

Local Mode Status Register The second register is used to monitor the status of CA#_%_IN_POS equal to CA#_%_TAR_POS.

Register Bit	Bit Label	Description
1	CA#_X_AT_TAR_POS	Only valid in Local Mode. (CA#_%_SYNC = 0) 1 = CA#_X_IN_POS at CA#_X_TAR_POS position and ramp generator at 0 velocity. (where % = X, Y, Z, ROLL, PITCH or YAW) 0 = not, goes to 0 when new target position entered. NOTE: This bit is not set to 1 if the ramp generator move is interrupted (dropped Enable, user task stopped, etc.) before the target position is reached. In such an instance, it is not possible to 'force' the bit to 1 by setting the ramp generator's target value equal to the current input value. Rather, another ramp generator move must be completed before this bit is set to 1.
2	CA#_Y_AT_TAR_POS	
3	CA#_Z_AT_TAR_POS	
4	CA#_ROLL_AT_TAR_POS	
5	CA#_PITCH_AT_TAR_POS	
6	CA#_YAW_AT_TAR_POS	

Table 10-6: Synchronized Mode Status Register

Coordinated Articulation Program Variables

The following program variables (20 integer variables, 93 float variables) are used for Coordination Articulation. In the following tables, “#” denotes the unit number.

Integer Variables

The following 20 integers are available for Coordination Articulation. The integer label column displays the default label issued by VisualMotion.

Number Offset	Default Value	Access	Integer Label	Description
0	0	read-only	CA#_NUMBER	Unit number of coordinated articulation function 1-2 Note: This feature supports 2 robots however they must be defined in different user tasks.
1	0	read/write in phase 2	CA#_OUT_AXIS1	Number of axis used for output 1
2			CA#_OUT_AXIS2	Number of axis used for output 2
3		read-only in phase 4	CA#_OUT_AXIS3	Number of axis used for output 3
4			CA#_OUT_AXIS4	Number of axis used for output 4
5			CA#_OUT_AXIS5	Number of axis used for output 5
6			CA#_OUT_AXIS6	Number of axis used for output 6
7	0	read/write in any phase	CA#_X_CAM_NUM	Input CAM number for % world coordinated position
8			CA#_Y_CAM_NUM	
9			CA#_Z_CAM_NUM	
10			CA#_ROLL_CAM_NUM	
11			CA#_PITCH_CAM_NUM	
12			CA#_YAW_CAM_NUM	
13	0	read/write in phase 2 read-only in phase 4	CA#_TASK_ID	Task function is associated with
14	0	read/write in any phase	CA#_Reserved14	
15	0	Read only	CA#_KINEMATIC	Number of kinematic equation used in transformation
16	1	read/write in phase 2 read-only in phase 4	CA#_ELS_GROUP	ELS Group number used as CAM input
17	0	read/write in any phase	CA#_RESERVED17	
18	0	read/write in any phase	CA#_RESERVED18	
19	0	read/write in any phase	CA#_RESERVED19	

Table 10-7: Program Integers

Float Variables

The following 93 floats are available for Coordination Articulated. The float label column displays the default label issued by VisualMotion.

Number Offset	Default Value	Access	Float Label	Description
0	0.0	read-only	CA#_X_IN_POS	X world coordinate input value in linear units, initialized to 0 at program activation.
1			CA#_Y_IN_POS	Y world coordinate input value in linear units, initialized to 0 at program activation.
2			CA#_Z_IN_POS	Z world coordinate input value in linear units, initialized to 0 at program activation.
3			CA#_ROLL_IN_POS	Roll world coordinate input value in rotary units, initialized to 0 at program activation.
4			CA#_PITCH_IN_POS	Pitch world coordinate input value in rotary units, initialized to 0 at program activation.
5			CA#_YAW_IN_POS	Yaw world coordinate input value in rotary units, initialized to 0 at program activation.
6	0.0	read/write in any phase	CA#_X_TAR_POS	Local mode, X target position for "single axis" move in world coordinates, linear units
7			CA#_Y_TAR_POS	Local mode, Y target position for "single axis" move in world coordinates, linear units
8			CA#_Z_TAR_POS	Local mode, Z target position for "single axis" move in world coordinates, linear units
9			CA#_ROLL_TAR_POS	Local mode, Roll target position for "single axis" move in world coordinates, rotary units
10			CA#_PITCH_TAR_POS	Local mode, Pitch target position for "single axis" move in world coordinates, rotary units
11			CA#_YAW_TAR_POS	Local mode, Yaw target position for "single axis" move in world coordinates, rotary units
12	100.0	read/write in phase 2	CA#_LIN_ACCEL	Local mode, linear acceleration used for "single axis" move(units/sec ²)
13	100.0	read-only in phase 4	CA#_LIN_DECEL	Local mode, linear deceleration used for "single axis" move(units/sec ²)
14	50.0	read/write in phase 4	CA#_LIN_VELOCITY	Local mode, linear velocity used for "single axis" move(units/min)
15	100.0	read/write in phase 4	CA#_ROT_ACCEL	Local mode, rotary acceleration used for "single axis" move(units/Sec ²)
16	100.0	read/write in phase 2	CA#_ROT_DECEL	Local mode, rotary deceleration used for "single axis" move(unitss/Sec ²)
17	50.0	read-only in phase 4	CA#_ROT_VELOCITY	Local mode, rotary velocity used for "single axis" move(RPM)
18	0.0	read-only	CA#_X_CAM_OUT	(CAM[#]*H) + Offset), for % world coordinate
19			CA#_Y_CAM_OUT	
20			CA#_Z_CAM_OUT	
21			CA#_ROLL_CAM_OUT	
22			CA#_PITCH_CAM_OUT	
23			CA#_YAW_CAM_OUT	

Table 10-8: Program Floats (1 of 2)

Number Offset	Default Value	Access	Float Label	Description
24	1.0	read/write in any phase	CA#_X_CAM_H	H CAM factor for % world coordinate
25			CA#_Y_CAM_H	
26			CA#_Z_CAM_H	
27			CA#_ROLL_CAM_H	
28			CA#_PITCH_CAM_H	
29			CA#_YAW_CAM_H	
30	0.0	read/write in any phase	CA#_X_CAM_OFF	Offset for % world coordinate
31			CA#_Y_CAM_OFF	
32			CA#_Z_CAM_OFF	
33			CA#_ROLL_CAM_OFF	
34			CA#_PITCH_CAM_OFF	
35			CA#_YAW_CAM_OFF	
36	-100.0	read/write in phase 2	CA#_X_MIN	Minimum value for % world coordinate
37			CA#_Y_MIN	
38			CA#_Z_MIN	
39	-180.0	read-only in phase 4	CA#_ROLL_MIN	
40			CA#_PITCH_MIN	
41			CA#_YAW_MIN	
42	100.0	read/write in phase 2	CA#_X_MAX	Maximum value for % world coordinate
43			CA#_Y_MAX	
44			CA#_Z_MAX	
45	180.0	read-only in phase 4	CA#_ROLL_MAX	
46			CA#_PITCH_MAX	
47			CA#_YAW_MAX	
48 ... 86	0.0	read/write in phase 2	CA#_KINEMATIC_V1 through CA#_KINEMATIC_V39	Kinematic setup data value 1 through Kinematic setup data value 39
		read-only in phase 4		
87	0.0	read-only	CA#_X_FDBK_POS	X world coordinate forward transform value in linear units, updated every SERCOS cycle in Phase 4.
88			CA#_Y_FDBK_POS	Y world coordinate forward transform value in linear units, updated every SERCOS cycle in Phase 4.
89			CA#_Z_FDBK_POS	Z world coordinate forward transform value in linear units, updated every SERCOS cycle in Phase 4.
90			CA#_ROLL_FDBK_POS	Roll world coordinate forward transform value in rotary units, updated every SERCOS cycle in Phase 4.
91			CA#_PITCH_FDBK_POS	Pitch world coordinate forward transform value in rotary units, updated every SERCOS cycle in Phase 4.
92			CA#_YAW_FDBK_POS	Yaw world coordinate input value in rotary units, initialized to 0 at program activation.

Table 10-9: Program Floats (2 of 2)

Initializing and Synchronizing Coordinated Articulation

Note: The following procedure is provided as a guide in demonstrating the necessary steps to initialize and synchronize a Coordinated Articulated system. It may not contain all the steps required for your specific application.

1. Ensure that all configured axes are in single axis mode by using a Mode Change icon.
2. Set $CA\#_%_SYNC$ to 0 (Local Mode)
3. Set each axis target position variable $CA\#_%_TAR_POS$ to a known world coordinated machine reference position.
4. Set $CA\#_%_MAN_IMD$ to 1 (Immediate Local Mode)
5. Transition the manual enable bit $CA\#_%_MAN_EN$ from $0 \Rightarrow 1$ to immediately store the value of $CA\#_%_TAR_POS$ to $CA\#_%_IN_POS$.

[The value of $CA\#_%_IN_POS$ is passed through the Coordinated Articulation Kinematic and the resulting value is written to axis commanded position parameter A-0-0101]

6. In single axis, use a Move icon to position each configured axis to the calculated axis commanded position A-0-0101 and wait for moves to complete.
7. Using a Change Mode icon, Synch to Master.

[Now that all axes are positioned to a known machine reference point, the Articulated Coordinated axes can now be synchronized to the Coordinated Articulation CAM output]

8. Copy the CAM output position variable, $CA\#_%_CAM_OUT$, to the local mode target position variable $CA\#_%_TAR_POS$ to synchronize each axis position to the CAM output position.
9. Disable immediate local mode by setting $CA\#_%_MAN_IMD$ to 0. (Normal Local Mode)
10. Transition the manual enable bit $CA\#_%_MAN_EN$ from $0 \Rightarrow 1$ to now synchronize the coordinated axes, using the ramp generator, to the Coordinated Articulation CAM output and wait for coordinated move to complete.
11. Now set the Coordinated Articulation function to Sync mode by transiting $CA\#_%_SYNC$ from $0 \Rightarrow 1$ and run program.

10.3 Kinematic Library

Standard Coordinated Motion Kinematics

Kinematic 1

This kinematic represents a standard 3 axes gantry (up to 3 axes, Cartesian coordinates X, Y, and Z) arrangement used with basic coordinated motion applications not requiring complex paths.

Note: Task parameters T-0-0050 through T-0-0059 are not used in kinematic #1.

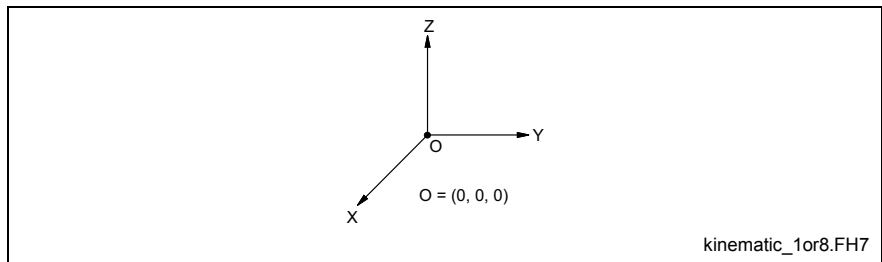


Fig. 10-4: Kinematic 1 Graphic

Kinematic 2

This kinematic represents a 2 axes (C1 and C2) parallelogram robot.

Task Parameters	Kinematic Constants	Default Values	Units
T-0-0050	K1	33.0	Inches
T-0-0051	K2	34.5	Inches
T-0-0052	K3	12.0	Inches
T-0-0053	K4	34.7279	Inches
T-0-0054	K5	6.0	Inches
T-0-0055	K6	8.0	Inches

Fig. 10-5: Kinematic 2 Task Parameters Values

Note: Kinematic 2 requires special mechanical setup. Refer to Special Case Kinematics on page 10-2 for details.

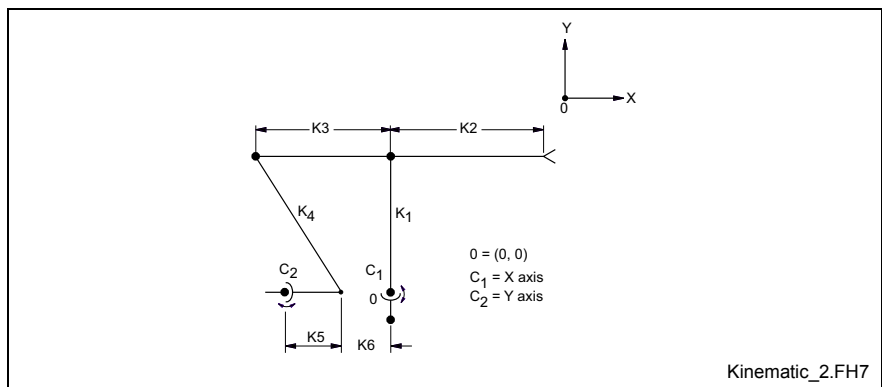


Fig. 10-6: Kinematic 2 Graphic

Kinematic 3

This kinematic represents a 2 axes (C1 and C2) XY tandem motor robot.

Note: Task parameters T-0-0050 through T-0-0059 are not used in kinematic 3.

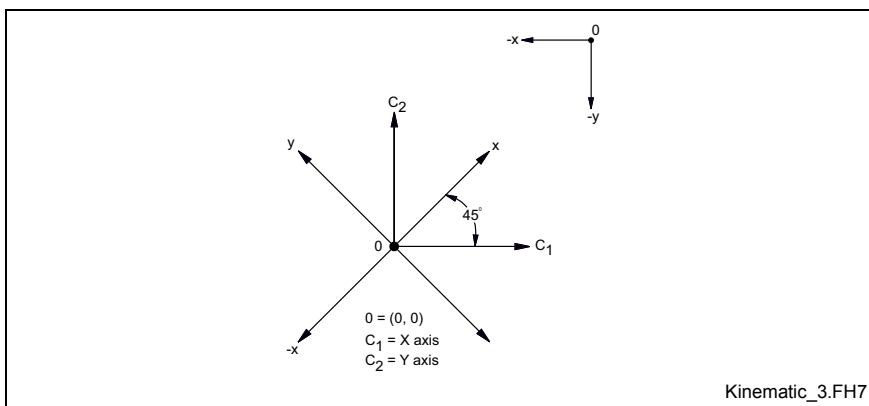


Fig. 10-7: Kinematic 3 Graphic

Kinematic 4

This kinematic represents a 2 axes (C1 and C2) parallelogram robot.

Task Parameters	Constants	Default Values	Units
T-0-0050	K1	850.0	mm
T-0-0051	K2	850.0	mm
T-0-0052	K3	200.0	mm
T-0-0053	K4	850.0	mm
T-0-0054	K5	200.0	mm

Fig. 10-8: Kinematic 4 Task Parameters Values

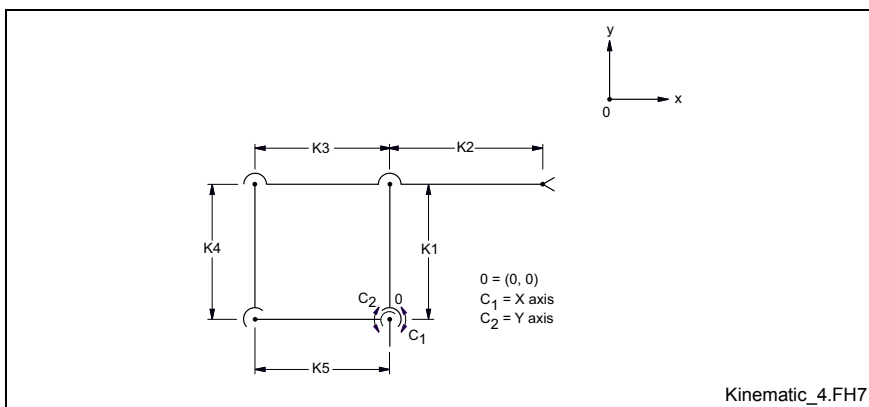


Fig. 10-9: Kinematic 4 Graphic

Note: Kinematic 4 requires special mechanical setup. Refer to Special Case Kinematics on page 10-2 for details.

Kinematic 5

This kinematic represents a 2 axes (C1 and C2) XY tandem motor robot.

Task Parameters	Constants	Default Values	Units
T-0-0050	K1	12.0	Inches
T-0-0051	K2	12.0	Inches
T-0-0052	K3	13.0	Inches
T-0-0053	K4	13.0	Inches
T-0-0054	K5	12.0	Inches
T-0-0055	K6	8.0	Inches
T-0-0056	K7	3.5	Inches
T-0-0057	K8	4.0	Inches

Fig. 10-10: Kinematic 5 Task Parameters Values

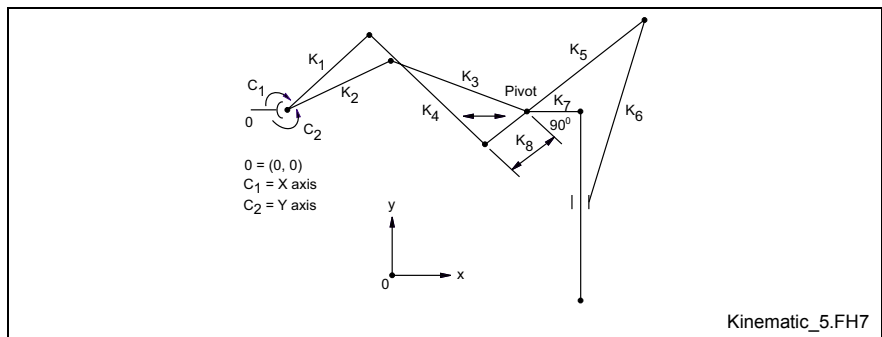


Fig. 10-11: Kinematic 5 Graphic

Note: Kinematic 5 requires special mechanical setup. Refer to Special Case Kinematics on page 10-2 for details.

Kinematic 8 (with Velocity Precision)

This Kinematic represents a 3 axes (X, Y, and Z) Gantry robot.

Note: Task parameters T-0-0050 through T-0-0059 are not used in kinematic 8.

The rate field in kinematic 8 is used to more precisely enter the velocity for each commanded point. The %Sp (percentage of speed) field does not have any affect in speed control.

Note: If a zero value is used for rate, the maximum path velocity (T-0-0020) will be used.

The path velocity is a resultant of all the individual axes' velocities for a kinematic. The control continuously calculates the individual axes' velocities to accomplish the programmed path. If the calculation for a given axis exceeds parameter A-0-0020, then the resultant path velocity for that given path segment will be scaled down.

No.	x:	y:	z:	Blend:	% Sp:	Ac	Dc	Jk	Rate/Roll:
001	0.0	0.0	0.0	0.0	0	0	0	0	0.0
002	0.0	0.0	0.0	0.0	0	0	0	0	0.0
003	0.0	0.0	360.0	0.0	0	0	0	88	0.0
004	90.0	200.0	0.0	0.0	0	0	0	0	0.0
005	360.0	360.0	0.0	0.0	0	0	0	0	0.0

Fig. 10-12: Example Points Table

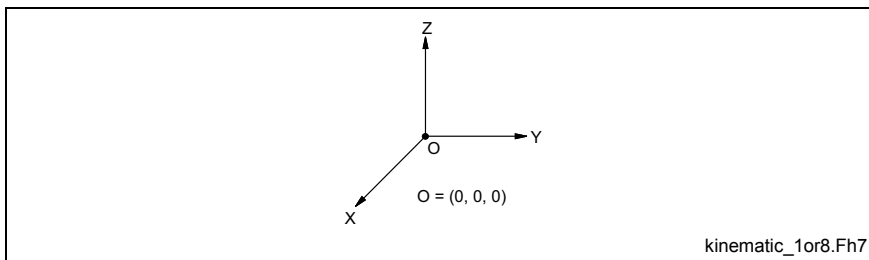


Fig. 10-13: Kinematic 8 Graphic

Kinematic 9

This Kinematic represents a 2 axes (C1 and C2) articulated arm robot.

Task Parameters	Constants	Default Values	Units
T-0-0050	K1	35.669	Inches
T-0-0051	K2	35.669	Inches

Fig. 10-14: Kinematic 9 Task Parameters Values

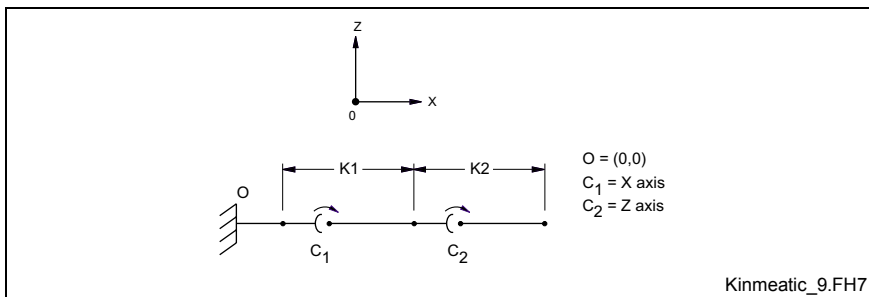


Fig. 10-15: Kinematic 9 Graphic

Note: Kinematic 9 requires special mechanical setup. Refer to Special Case Kinematics on page 10-2 for details.

Kinematic 10

This Kinematic represents a 3 axes (X, Y and Wrist) robot.

Note: Task parameters T-0-0050 through T-0-0059 are not used in kinematic 10.

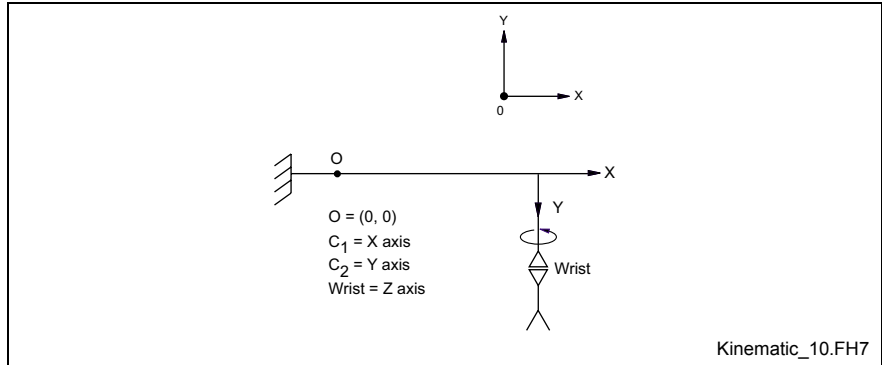


Fig. 10-16: Kinematic 10 Graphic

Note: Jogging of kinematic 10 is controlled by the following bits of Task Jog registers 7-11:

- The X-axis is jogged using bit 9 (Jog_X_Coord)
- The Y-axis is jogged using bit 10 (Jog_Y_Coord)
- The Z-axis (Wrist) is jog using bit 14 (Jog_Joint_6)

Kinematic 12

This Kinematic represents a 2 axes (X and Z) Loader robot.

Task Parameters	Constants	Default Values	Units
T-0-0050	K1	0.0	mm
T-0-0051	K2	711.200	mm
T-0-0052	K3	1642.28	mm
T-0-0053	K4	0.0	mm
T-0-0054	K5	0.0	mm

Fig. 10-17: Kinematic 12 Task Parameters Values

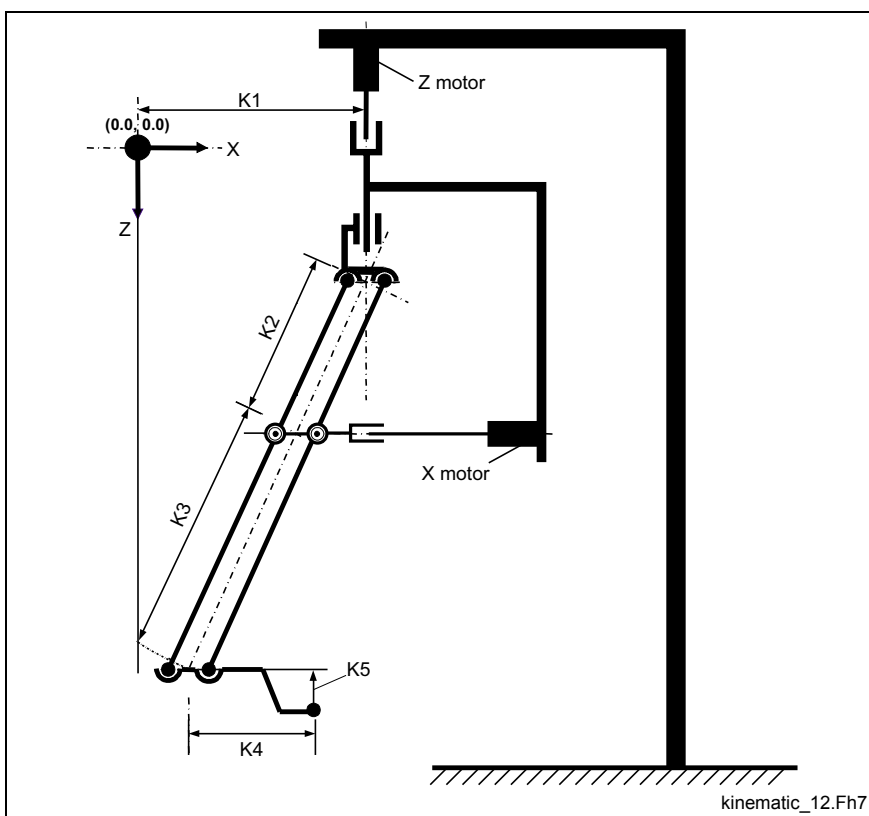


Fig. 10-18: Kinematic 12 Graphic

Coordinated Articulation Kinematics

Kinematic 13

Kinematic 13 is a pass through kinematic. It can be used for up to 6 axes in an Articulated Coordination application. The target position value for each axis must be assigned in the respective float variable CA#_%_TAR_POS (where % is x, y, z, roll, pitch or yaw). There is no output kinematic resolving the position of the axes based on a coordinated move.

This kinematic is not assigned in the axis icon but in the *Articulated Coordination* icon in the *Kinematic Setup* button. Since this kinematic is a pass through, any values assigned as Kinematic Data has no effect on the output. The float variable values assigned is pass through to each axis.

Kinematic 14

Kinematic 14 is a customer specific kinematic used for new Hexapod robot.

Kinematic 15

Kinematic 15 is customer specific and preparatory.

Kinematic 16

This Kinematic represents a 2 axes (X and Z) Loader robot used in a coordinated articulation application.

Task Parameters	Constants	Default Values	Units
T-0-0050	K1	0.0	mm
T-0-0051	K2	711.200	mm
T-0-0052	K3	1642.28	mm
T-0-0053	K4	0.0	mm
T-0-0054	K5	0.0	mm

Fig. 10-19: Kinematic 16 Task Parameters Values

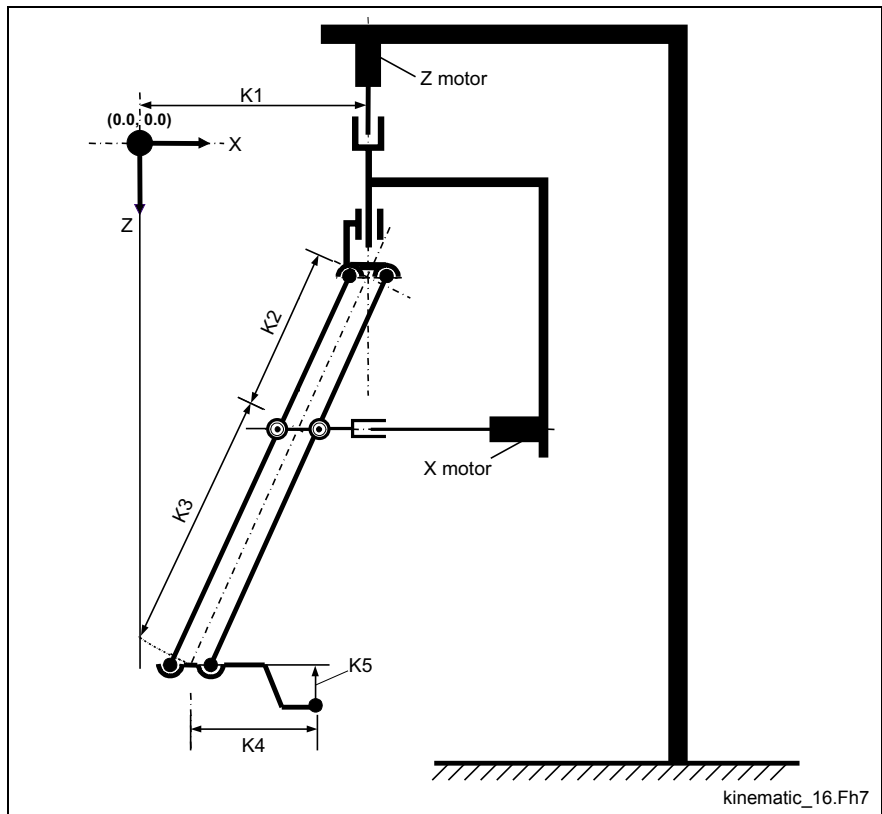


Fig. 10-20: Kinematic 16 Graphic

Kinematic 18

This kinematic represents a 2 axes (C1 and C2) XY tandem motor robot used in a coordinated articulation application.

Note: Task parameters T-0-0050 through T-0-0059 are not used in kinematic 18.

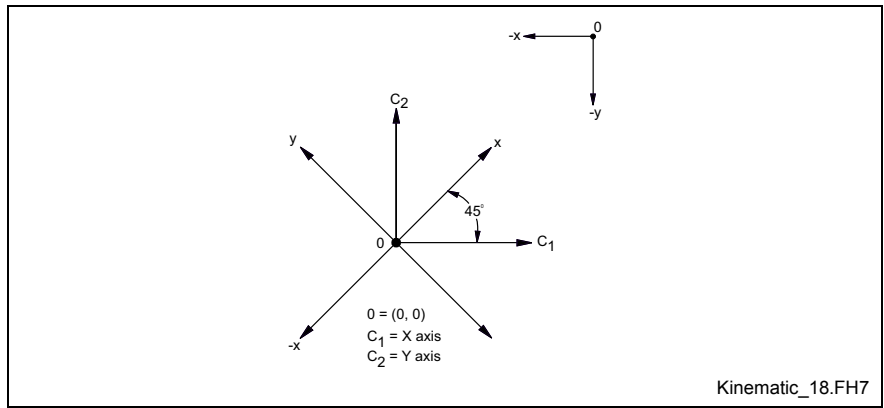


Fig. 10-21: Kinematic 18 Graphic

11 Link Ring Functionality

11.1 Link Ring Interface Overview

Link Ring is a SERCOS loop that can be added to an ELS system to extend the number of drives and controls that can receive position data from a master control. A Link Ring has the capacity to connect up to 31 Link Ring Slaves in series to a Link Ring Master with each control capable of handling 40 drives. All controls in the Link Ring can run the same program or different programs.

Every PPC-R control (also referred to as a node) participating in a Link Ring must have a DAQ03 expansion card. The DAQ03 card is the control's interface for the Link Ring fiber optic cables that transmit the position data to other controls in the Link Ring. Refer to the VisualMotion 9 Project Planning manual for details on the DAQ03 card and hardware setup for Link Ring.

A control participating in a Link Ring is configured as one of the following:

- Link Ring Master
- Link Ring slave
- Passive participant (repeater)

In a Link Ring, only one control can be configured as a Link Ring Master while all remaining controls are either configured as Link Ring Slaves or as passive participants.

Link Ring Master

The Link Ring Master controls Link Ring communication by controlling the SERCOS cycle time, which is fixed at 8 ms, for all Link Ring slaves and SERCOS drives in the Link Ring. The Link Ring Master collects the master axis positions and distributes them cyclically to the other nodes.

Link Ring Slave

A Link Ring slave synchronizes telegram processing with its SERCOS drive ring, collecting positional data from the slave axes and returning it to the Link Ring Master control.

Passive Participant (Repeater)

A passive participant is a slave in the Link Ring that is not active. Passive participants transmit a signal, but do not participate in position data communication.

Link Ring Types

A Link Ring can be set up in either a single or double Link Ring configuration. A single Link Ring connects all Link Ring Slaves in series to the Link Ring Master, transmitting position data in one direction. A double ring is two separate fiber optic rings connected to the primary and secondary, receive and transmit terminals of each DAQ03 card in the Link Ring.

11.2 Hardware Setup

A Link Ring requires the following hardware:

- SERCOS fiber optic cables
- Fiber optic cables for DAQ03 connections

Note the cable lengths during hardware setup. These values will be used later in the software setup. Depending on the length of the fiber optic cable within the Link Ring (from control to control), it may be necessary to adjust the optical output power. If the fiber optic cable (LWL) length is not set within the acceptable range of values in the *Control Setting* window of VisualMotion, the H17 (single ring) or H18 (double ring) LED on the DAQ03 card faceplate (see Fig. 11-1) will turn on.

The H11 and H12 LEDs on the DAQ03 faceplate indicate that the card is configured either as a master (H11 LED on) or slave (H12 LED on). Master and slave designations for a control are assigned in the software.

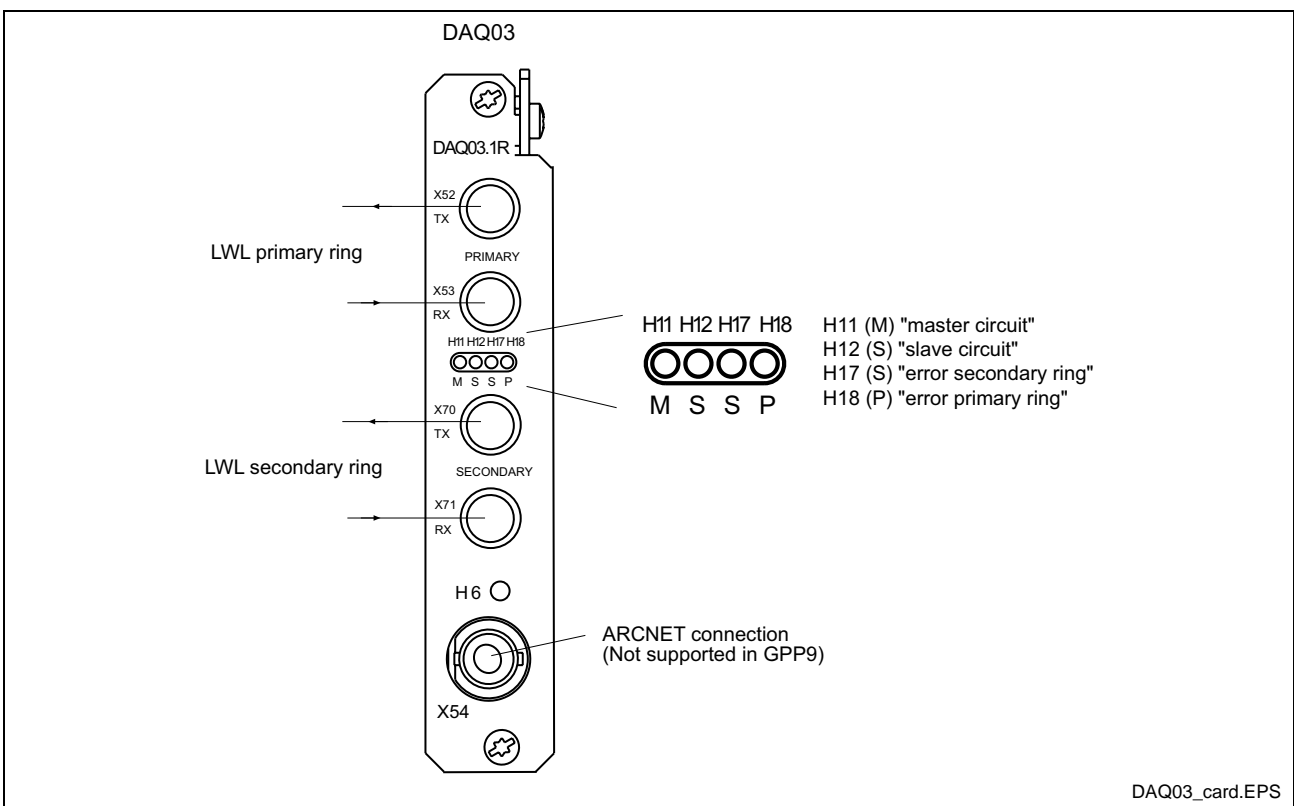


Fig. 11-1: Detail of LEDs on DAQ Card Faceplate

Single Ring

A single ring only uses the primary receive and transmit connections on the DAQ cards, see Fig. 11-3.

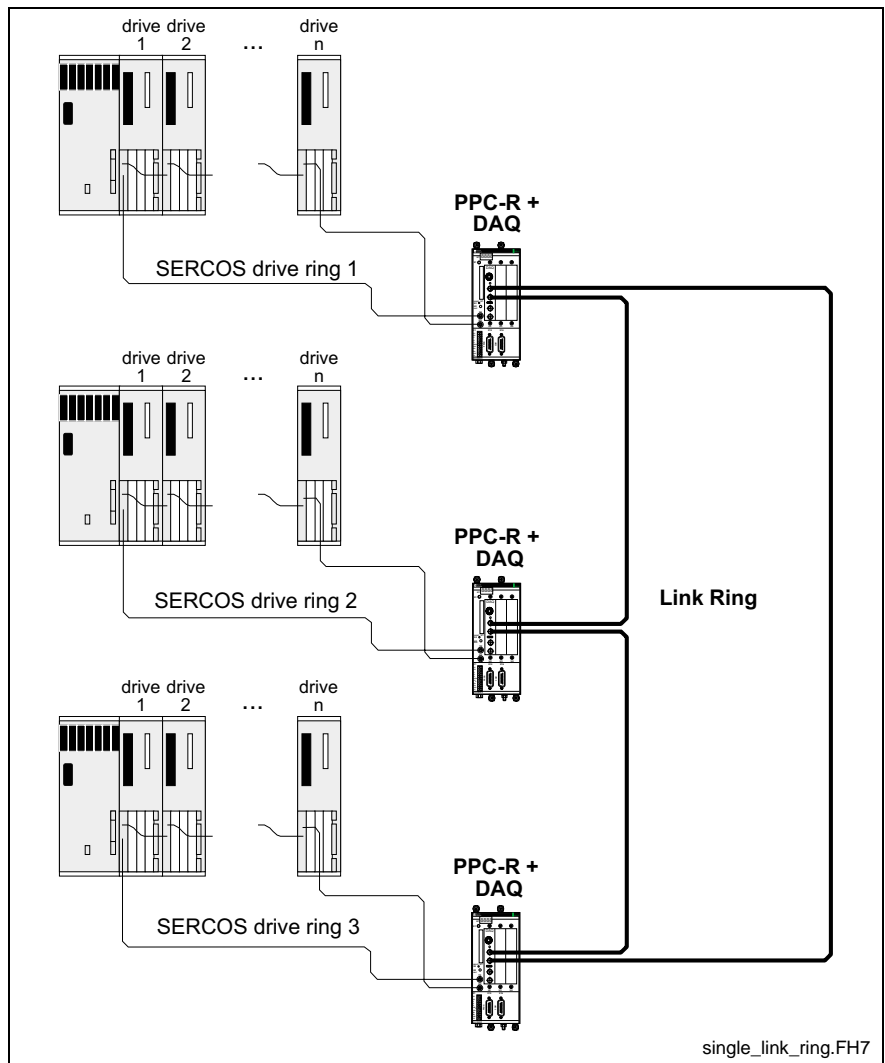


Fig. 11-2: Single Link Ring

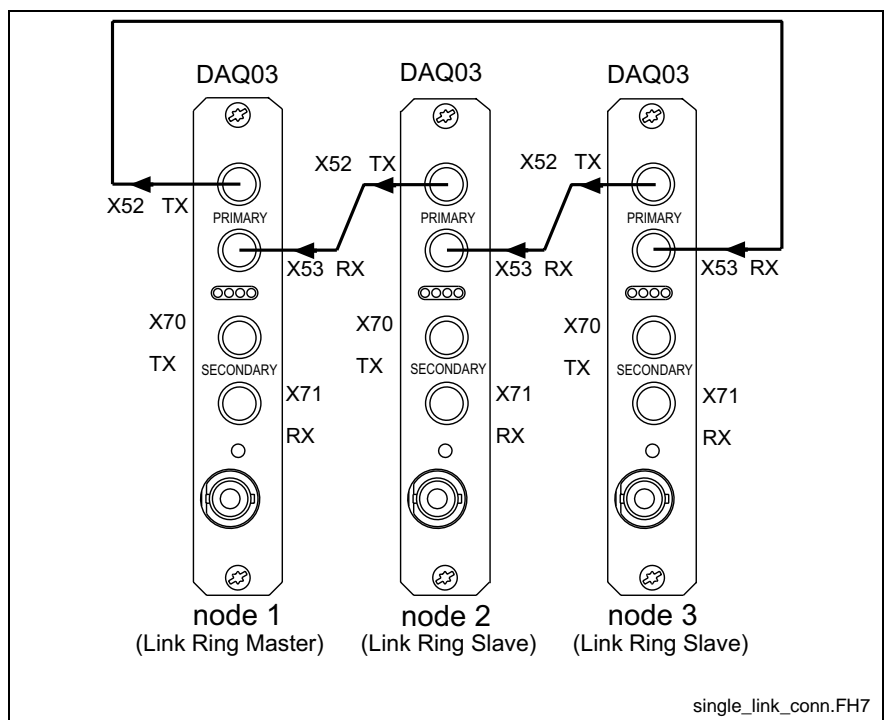


Fig. 11-3: DAQ03 Connection for Single Link Ring

Double Ring

The primary and secondary rings in a double ring are identical except that they transmit signals in opposite directions and the secondary ring transmits only diagnostic signals when the primary ring is active.

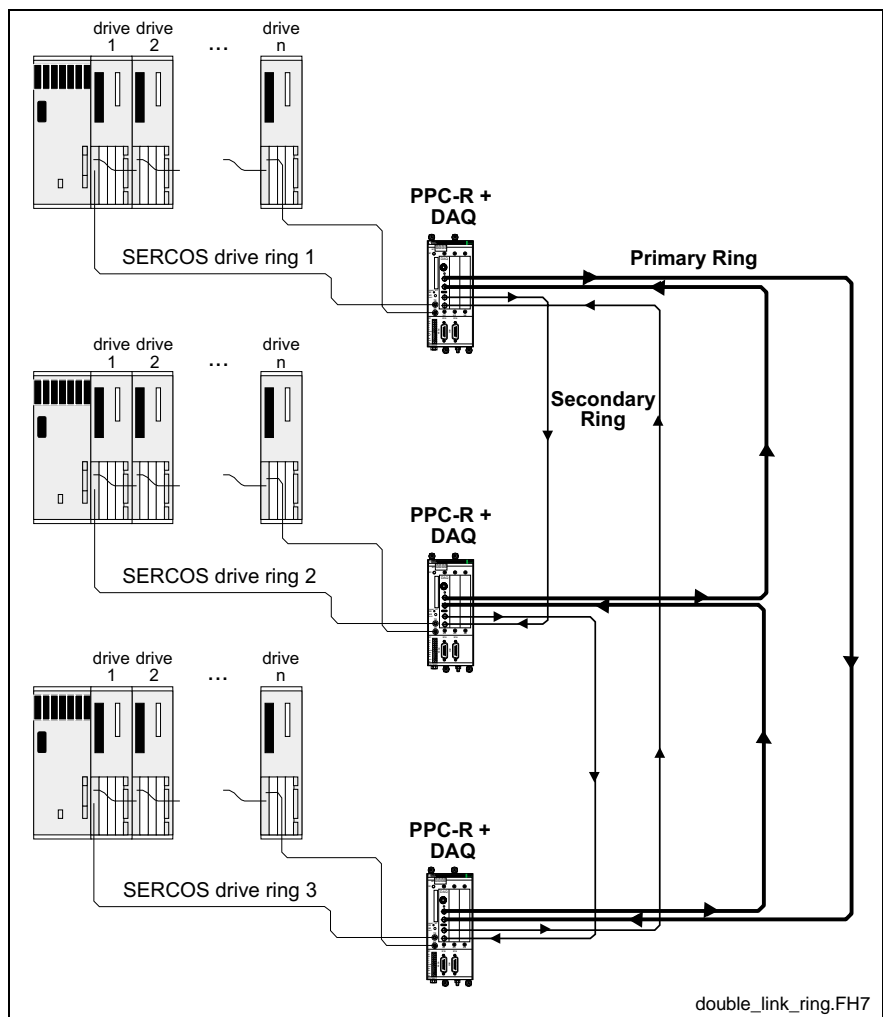


Fig. 11-4: Double Link Ring

The primary and secondary loops are connected in a different order from each other. One transmitting from the Link Ring Master to node 2, the other transmitting from the Link Ring Master to the last node in the series. For example, in Fig. 11-5, the primary Link Ring transmits a signal from the Link Ring Master to node 3 while the secondary ring transmits from the Link Ring Master to node 2.

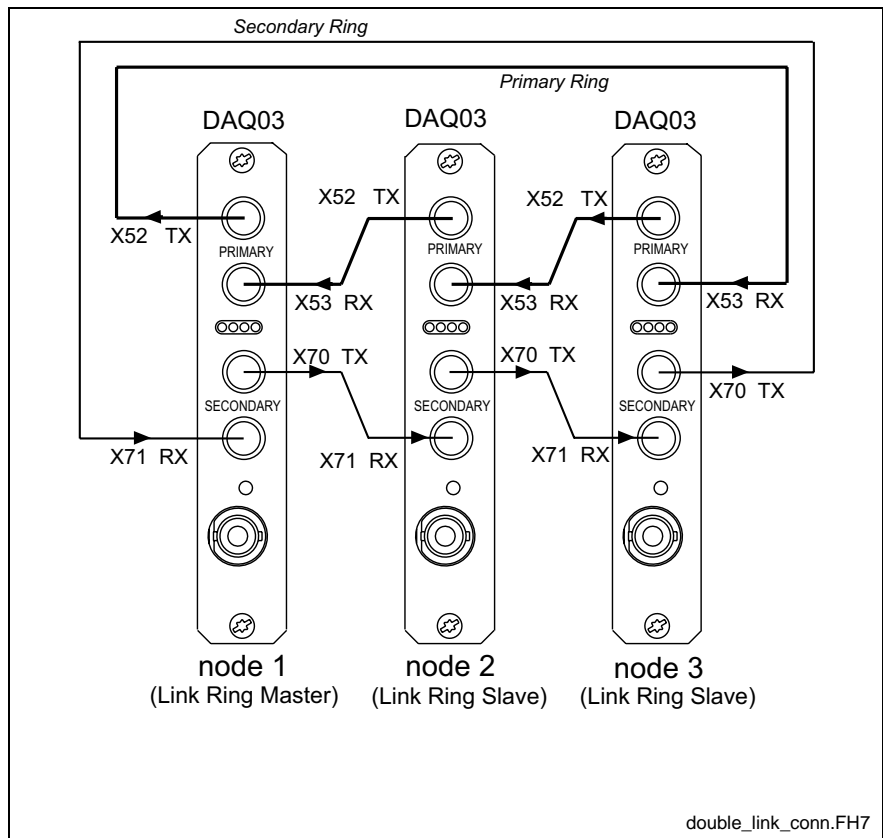


Fig. 11-5: DAQ03 Connection for Double Link Ring

Setting SERCOS Baud Rate in Drive

The SERCOS baud rate in the drive must match the SERCOS baud rate in the control. The baud rate setting can be changed by adjusting the DIP switch S20 on the faceplate of the SERCOS interface card in the drive. Refer to the *Project Planning* manual for additional information on SERCOS transmission baud rate settings.

11.3 Software Setup

The software setup for a Link Ring takes place in the ELG System Master and ELS Group icons of an ELS project. The following Registers, Parameters, and Bits are used to monitor, store and manipulate Link Ring data.

Registers, Parameters, and Bits

Parameters

The following parameters store Link Ring Master and Slave data:

Parameter	Description
C-0-0300	Link Ring Control Word
C-0-0301	Link Ring Primary Fiber Optic Cable Length
C-0-0302	Link Ring Secondary Fiber Optic Cable Length
C-0-0303	Link Ring MDT Error Counter

Table 11-1: Link Ring Parameters

The bits for Parameter C-0-0300 include:

- Bit 1 – Determines the function of the control in the Link Ring.
- Bit 2 – Determines the function of the control in the Link Ring.
- Bit 5 – Sets Link Ring structure type (single or double)

The combination of Bits 1 and 2 can occur in the following:

Bit 1	Bit 2	Behavior	Function
1	1	Active participant	Link master
1	0	Active participant	Link slave
0	N/A	Passive participant	Repeater

Table 11-2: Link Ring Participant Configuration

Only one control in a system can be set as a Link Ring Master and each control in a Link Ring must have the same ring type (single or double), which is set in Bit 5 of the Link Ring control word:

Bit 5	Type
0	Single ring
1	Double ring

Table 11-3: Link Ring Type

Registers

VisualMotion provides one control register and three status registers for configuring and monitoring each control in the Link Ring:

Register	Description	Bits	Description
1	Bit 7 of this register is used to rebuild the double ring configuration if a redundancy loss error is encountered.	7	Rebuild_Double_Ring – This bit is used to clear the Redundancy_Loss bit after a fault has been corrected in the secondary ring of a Double Link Ring.
40	This register monitors errors in the Link Ring and fiber optic rings (primary and secondary) for each configured control.	4 5 6 7	Link_Error – This bit goes high when a Link Ring error has occurred. Error_Prim_Opt_Ring – This bit goes high when the data telegram in the Link Ring has been rerouted in the primary ring. Error_Sec_Opt_Ring – This bit goes high when the data telegram in the Link Ring has been rerouted in the secondary ring. Redundancy_Loss – This bit goes high when redundancy provided by the secondary ring is no longer available.

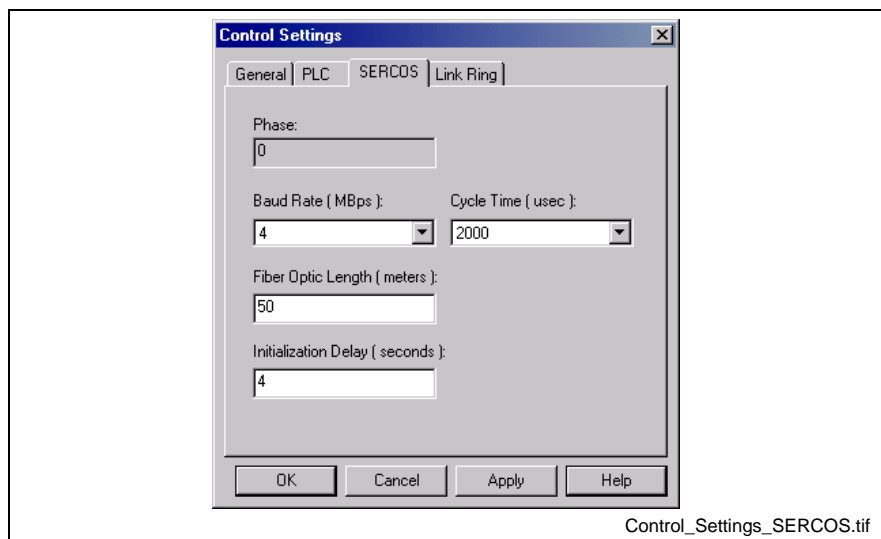


Fig. 11-6: SERCOS Settings for Control

Control Settings for Link Ring

Each Link Ring node in a system is configured individually either as a master, slave or repeater. The following is a list of steps to configure a Link Ring node.

1. Using VisualMotion Toolkit, select **Tools** ⇒ **Control Settings** and set the **Address** (C-0-0002) of each node in the *General* tab. Valid addresses for a Link Ring are 1 to 32.

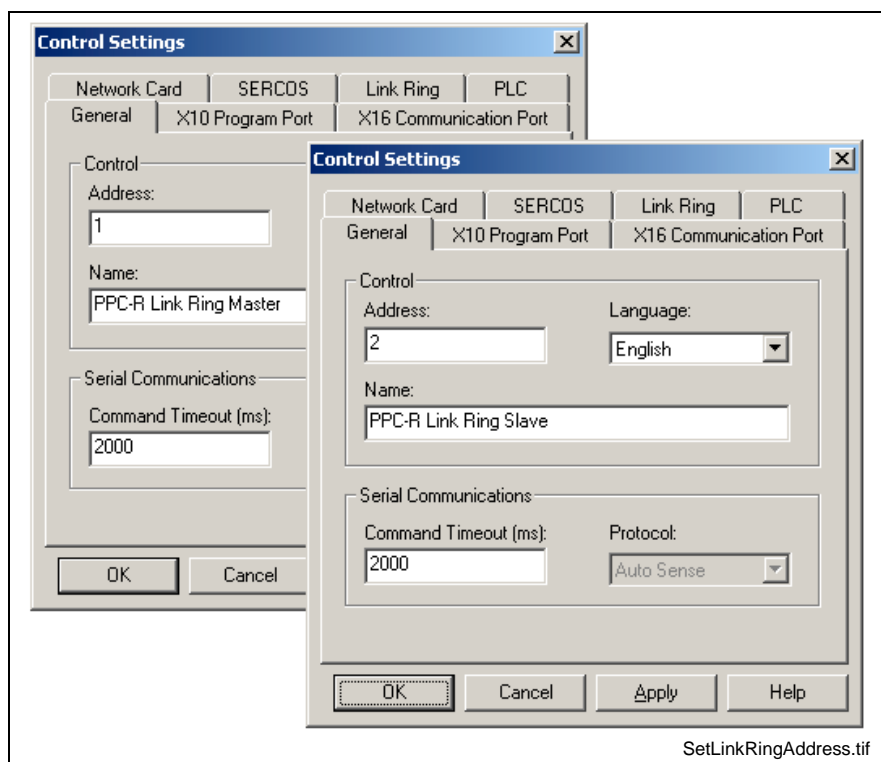


Fig. 11-7: Setting Link Ring Address

2. In the *Link Ring* Tab of the *Control Settings* window, set the Link Ring type (single or double) and the control type (master slave, or repeater), see Fig. 11-8.

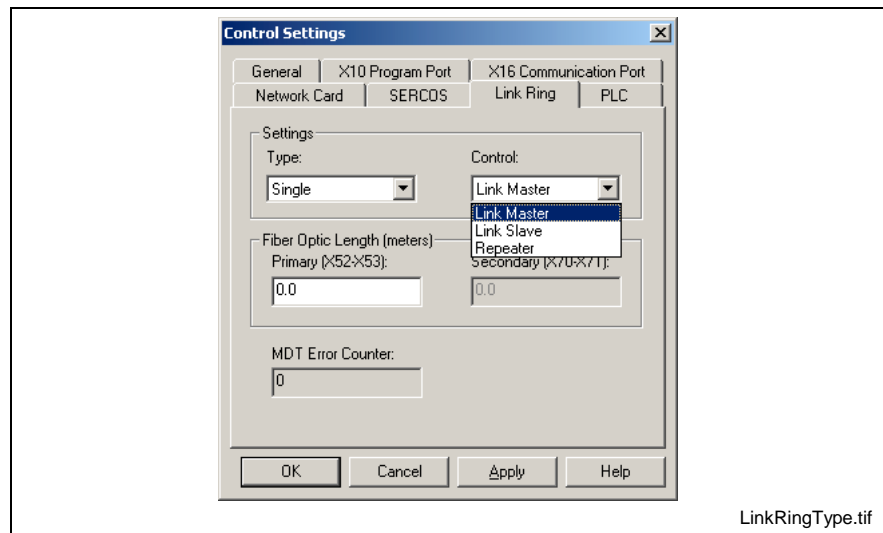


Fig. 11-8: Link Ring Control Type

- Fiber Optic Length** The length of the fiber optic cables will determine the data transmission power of the Link Ring. The length entered for a single ring (between adjacent DAQ03 cards) is stored in card parameter C-0-0301, *Link Ring Primary Fiber Optic Length*. The length entered for the secondary ring is stored in parameter C-0-0302, *Link Ring Secondary Fiber Optic Length*. Parameter C-0-0302 is not used when the system is configured as a single ring. Refer to section “11.2 Hardware Setup” for information on error reactions associated with fiber optic length settings.
- MDT Error Counter** The master data telegram (MDT) transports data from the Link Ring Master to the Link Ring Slaves. The MDT Error Counter field displays the number of MDT errors received by the slave from the master. This value can also be accessed from control parameter, C-0-0303.

Programming Icons

A Link Ring is configured in a VisualMotion project with the ELSMstr1 and ELSSGrp icons.

ELS System Master

Within the *Setup ELS System Master Assignment* window, the Link Ring Master source is selected. The master source is either a Virtual Master, Real Master, or ELS Group that sends position data to the Link Ring slaves. By sending the master source signal through ELS System Master 5, it becomes designated as the Link Ring Master signal, see Fig. 11-1. All nodes in a Link Ring can export one master position and import up to 5 master positions.

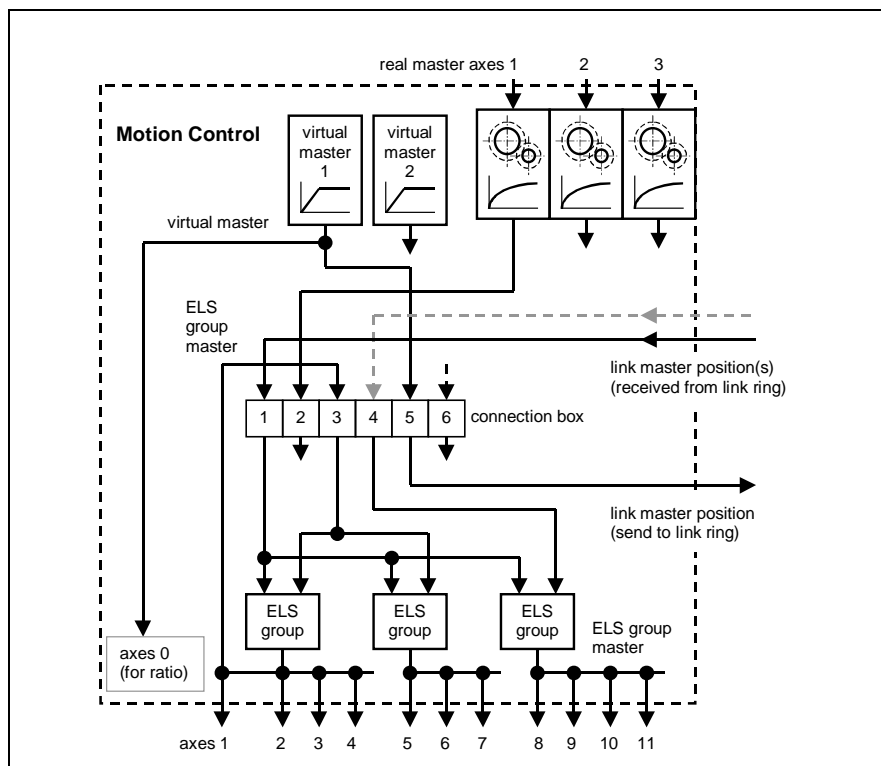


Fig. 11-1: Link Ring Data Routing to Connection Box (ELS System Master)

To establish the Link Ring Master:

1. Select Master 5 in the *ELS System Master Assignment* window and click the **Edit...** button to open the *Setup ELS System Master 5 (Link Ring Output)* window.
2. Select the master type and the settings for it:
 - Real Master – Select Drive Number and filter type
 - Virtual Master – Select the Virtual Number (1 or 2)
 - ELS Group – Select the Group (1 – 8)
3. Select Link Ring as the Type and an External Number from 1 to 32 to indicate which Link Ring Master position (identified by the node's number designation in Link Ring) the ELS System Master will be receiving.

The Link Ring Master output from ELS System Master 5 can be used locally or exported to another system. As shown in Fig. 11-2, ELS System Master 1 of Link Ring Master (node #1) receives its local Link Ring Master output. ELS System Master 1 of Link Ring Slave (node #2) receives its signal from Link Ring Master (node #1) and ELS System Master 2 receives its signal from its local Link Ring Master.

While only one control can be designated as a Link Ring Master, each control (master and slave) can export a Link Ring Master position. This master position can be used locally, exported to another system (each system can import up to six master positions), or both so that there can exist up to 32 master signals being transmitted through the Link Ring.

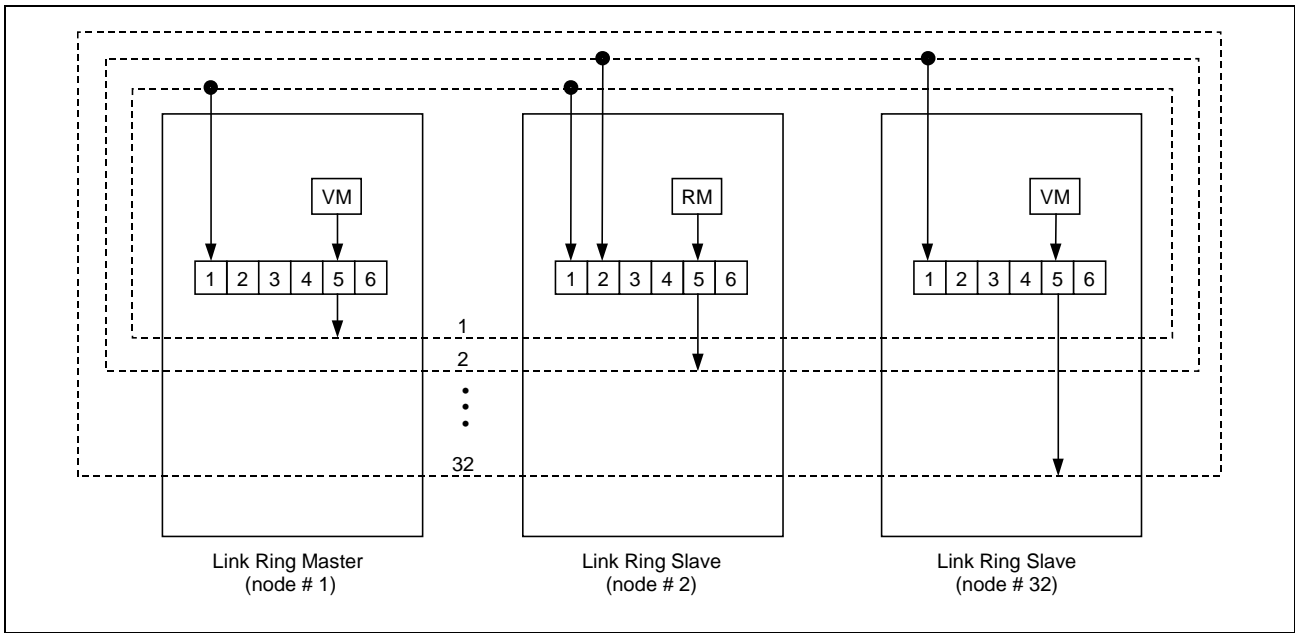


Fig. 11-2: Link Ring Example

ELS Group

Any ELS Group participating in a Link Ring can import a Link Ring Master position from any ELS System Master other than ELS System Master 5 of the same node. The ELS System Master position that the ELS Group imports is set in the *Initialize ELS Group Variables* window, see Fig. 11-3, which is opened by clicking the **ELS Group Variables compile time initialization...** button in the *ELS Group Setup* window. The Link Ring Master set in step 3 of the Link Ring Master setup (Link Ring 1) is listed in the drop-down menus of the Master 1 and Master 2 edit boxes.

Any ELS Group participating in a Link Ring configuration should receive master position data from the Link Ring and not locally from the node's master source. Deriving a position from the master source introduces a phase difference between a local ELS Group and an ELS Group receiving position data through the Link Ring.

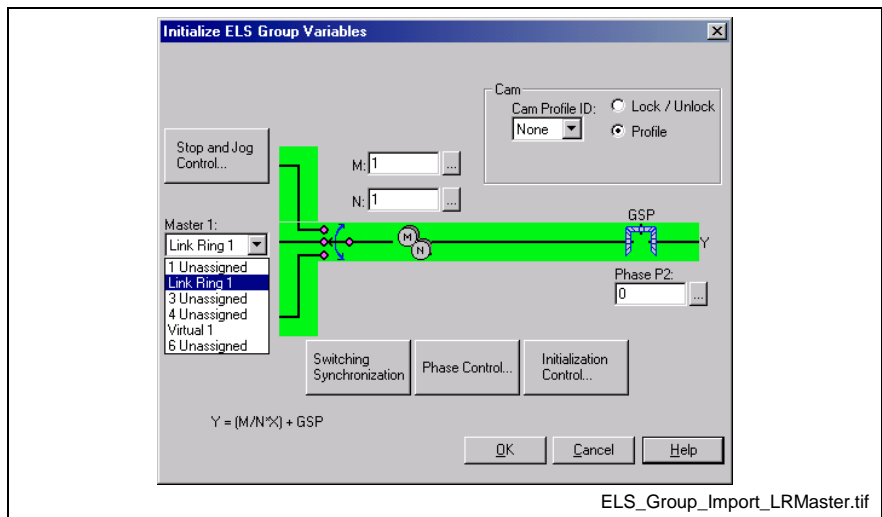


Fig. 11-3: ELS System Master selection for ELS Group

11.4 Fault Tolerance in Double Ring

In a double Link Ring, a single fault (such as a break in the cable) in the primary or secondary ring, or in both the primary and secondary rings between two adjacent controls, will not disrupt signal transmission. The secondary ring allows the signal to be transmitted when a fault occurs in the primary ring by rerouting the signal through the secondary ring to complete the loop.

Under normal operating conditions, communication takes place on the primary ring while the secondary ring transmits only a diagnostic signal for error detection. Because the Primary ring transmits signals in the opposite direction from the Secondary ring, the secondary ring is able to reroute the signal back to the primary ring when a fault in the ring occurs..

Fault in Primary Ring

When an fault, such as a cable break, occurs in the primary ring, the Link Ring data position signal is rerouted to the secondary ring to complete the loop, see Fig. 11-4.

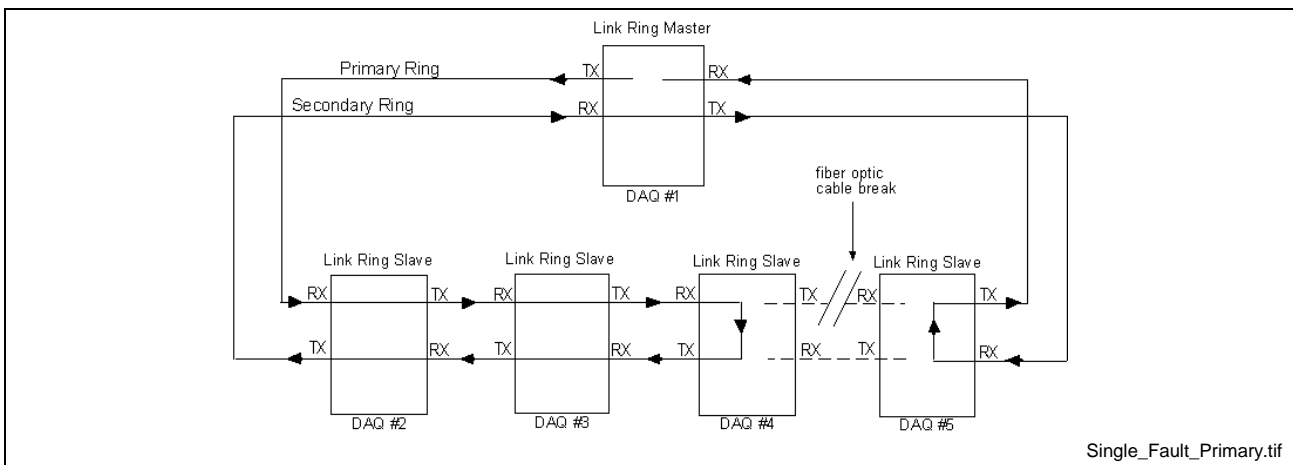


Fig. 11-4: Single Fault in Primary Ring

In this example, a fault occurs in the primary ring between control #4 and control #5. The DAQ03 card in control #5 reacts by turning on the H18 LED, which indicates an error in the primary ring. The Redundancy_Loss, Bit 7, of register 40 (Link_Ring_Status) and Error_Prim_Opt_Ring and/or Error_Sec_Opt_Ring, bits 5 and 6, will go high in at least one of the nodes of the Link Ring. In both DAQ03 #4 and DAQ03 #5, the controls on either side of the cable fault, the signal is routed through the secondary ring, which transmits in the opposite direction from the primary ring. This completes the loop and all controls in the ring continue to receive the signal.

Fault in Secondary Ring

A fault in the secondary ring will not affect the signal transmission in the primary ring. The primary ring will continue to function, but redundancy will be lost when the secondary ring is no longer functional. Bit 7 of Register 40 will go high, indicating that redundancy has been lost, and Bit 6 of Register 40 will go high, indicating that an error has occurred in the secondary ring.

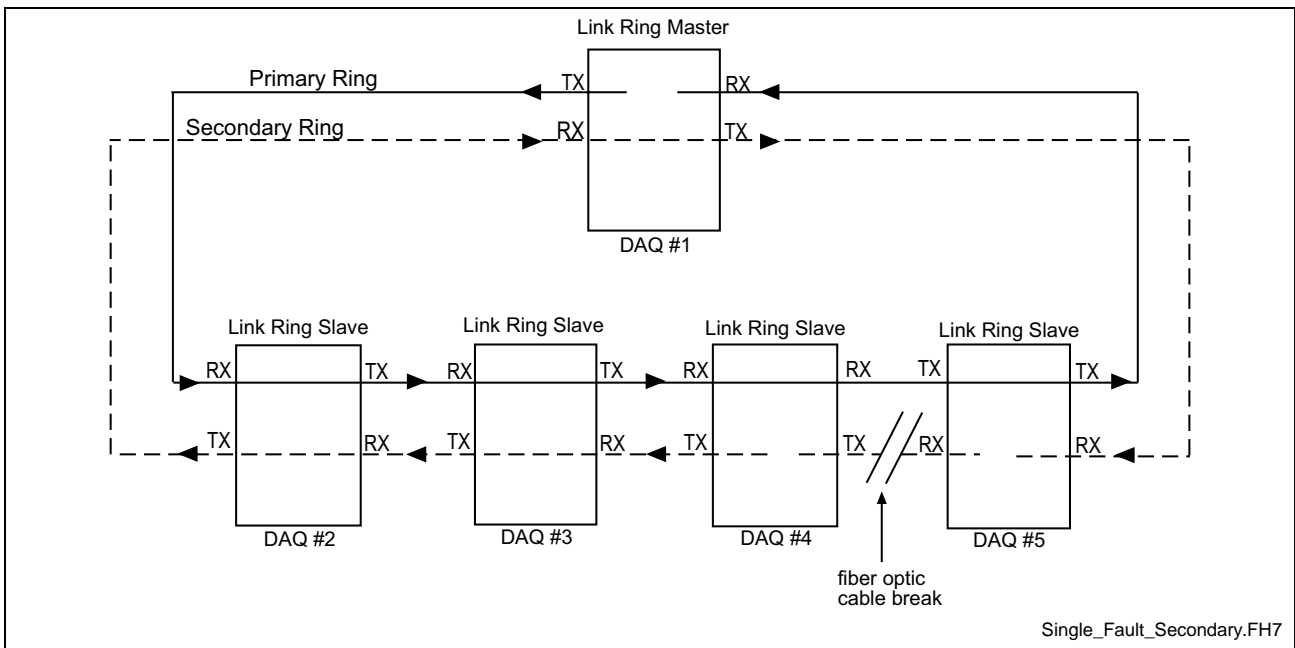


Fig. 11-5: Fault in Secondary Ring

Fault in Primary and Secondary Rings

When a fault occurs in both the primary and secondary rings between the same two nodes, the signal is routed through the secondary ring to complete the loop. Bits 5 of Register 40 will go high to indicate a redundancy loss and either bit 5 or bit 6 will go high to indicate an error in the primary or secondary rings.

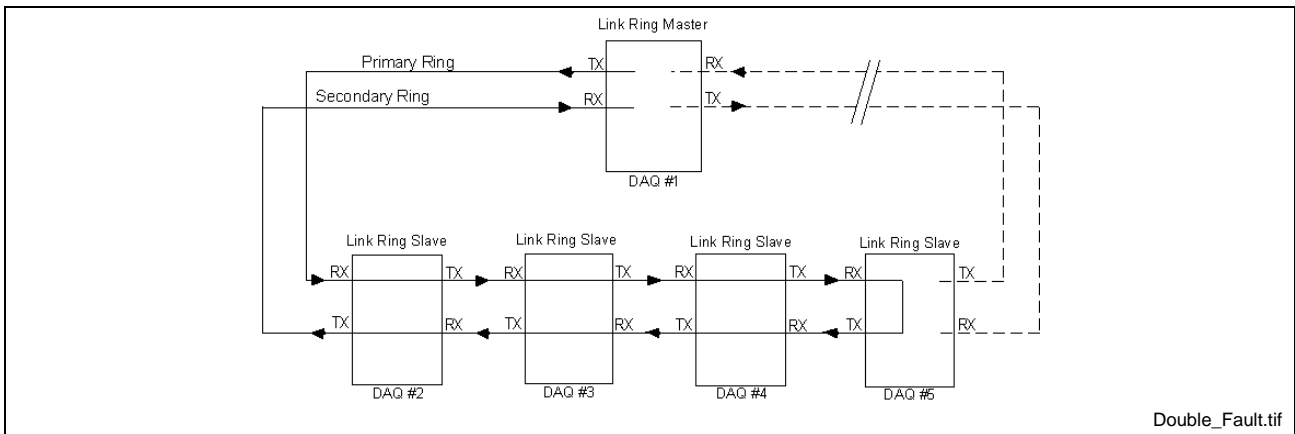


Fig. 11-6: Faults in Primary and Secondary Rings

If a fault occurs in the primary and secondary rings between different controls, the system will shut down. If more than one fault occurs in the Primary ring, the system will shutdown.

Clearing A Redundancy Loss

If a redundancy loss (Register 40, Bit 7) or 541 Link Ring Error (Transmission path defective) is indicated on any node of a given Link Ring, use the following procedure to clear the error.

1. Locate and correct any fiber optic cable problems.
2. Hold the Rebuild Double Ring input (Register 1, Bit 7) on all nodes high until every node's Redundancy Loss bit (Register 40, Bit 7) goes low. You can set a timeout in your program logic in the event that the bit does not go low after 10 seconds. This step must be done with all nodes in phase 4.
3. Set the Clear_All_Errors input (Register 1, Bit 5) high for all nodes that have their Error output (Register 21, Bit 5) set high. The Link Error output (Register 40, Bit 4) can also be used as a flag. This bit will only go high for Link Ring-related errors. You can set the Clear_All_Errors input low when the Error output goes low. This step does not need to be performed for all nodes simultaneously, like the Rebuild Double Ring in step 2.

Note: The Clear_All_Errors input does not force a node to execute a Rebuild Double Ring. If performed on its own, without a Rebuild Double Ring, a Redundancy Loss will remain.

12 Error Reaction

12.1 Overview

Errors in the VisualMotion system can have different root causes like hardware and communication failure, operator input error or a crash in the machine which could obstruct the movement of an axis.

When an error occurs, the error type and associated control and drive parameters determine the system's error reaction. The basic function of the error reaction is for the control to determine which user task(s) A-D should be shutdown. Drives, associated with a user task that is shutdown, are stopped based on the error reaction of the drive. User tasks and drives can be parameterized to react differently to errors.

The following are some examples:

- synchronized stop of several axes in ELS or coordinated motion
- switching axes to torque free mode
- moving axes to a safe position

Errors in the VisualMotion system can originate from the control (system and tasks errors) or from the drives, as illustrated in the Fig. 12-1.

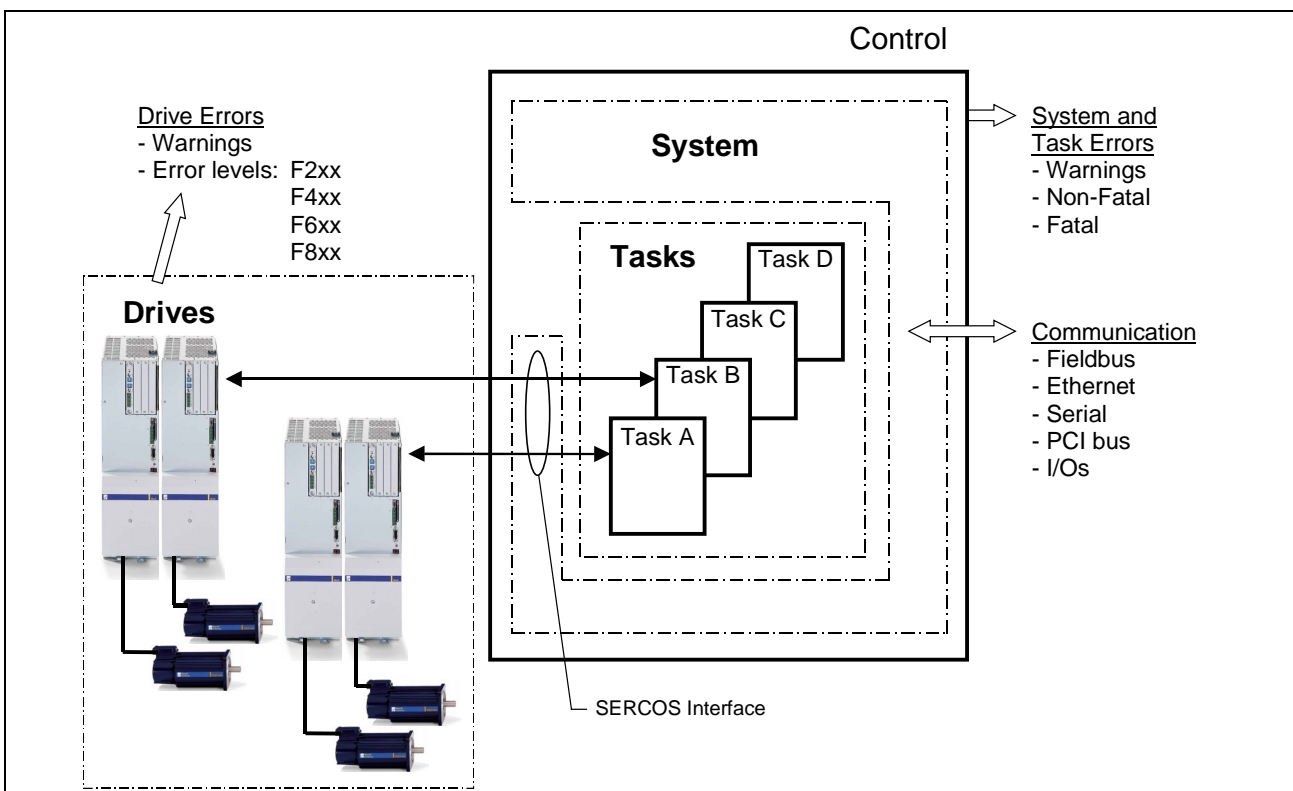


Fig. 12-1: System Overview

Error Types

VisualMotion's error system distinguishes between system, task, and drive errors and identifies them with diagnostic error codes.

System Errors

System errors affect all tasks that are not set to run during errors. These errors include for example, fieldbus errors, PLC interface errors or Option Card PLS errors.

Special Case System Error

When error "409 SERCOS Disconnect Error" occurs, the SERCOS communication stops and the control switches to SERCOS phase 0. Drives react to the loss of SERCOS communication by setting a F4x interface error and stopping based on the error reaction of the drive.

User tasks that are configured to run during errors will continue to run. However, motion-related functionality such as the path planner for coordinated motion, or the ELS system stops when SERCOS communication is not in SERCOS phase 4.

Hardware and Fatal Operating System Errors

If the control stops operating because of a hardware failure or because of a fatal operating system error, the SERCOS communication will also stop. When a fatal operating system error occurs, the PPC-R displays two dots ' .' and the PPC-P11.1 displays a single dot '.' on the H1 display.

Task Errors

Task errors occur as the result of a user program instruction. These errors include for example, parameter transfer error, calculation instruction error, etc.. A task error sets bit 5 (Task_Error) in the relevant task status register.

Drive Errors and Warnings

A drive error sets bit 14 (Shutdown_Error) of the relevant axis status register and the control issues the error "420 Drive D Shutdown Error".

A drive warning sets bit 13 (Class_2_Warning) in the relevant axis status register. The default reaction of the control to a warning is to issue the error "498 Drive Shutdown Warning".

Drive warnings will be ignored by the control when bit 13 of C-0-0010 is set to 1.

Drive errors and warnings are associated with a task. A drive is associated with a task in the Axis or ELS Group icon.

Error Levels

The control distinguishes 3 levels of errors: fatal errors, non-fatal errors and warnings. The following flowchart shows how the control will respond to fatal and non-fatal errors:

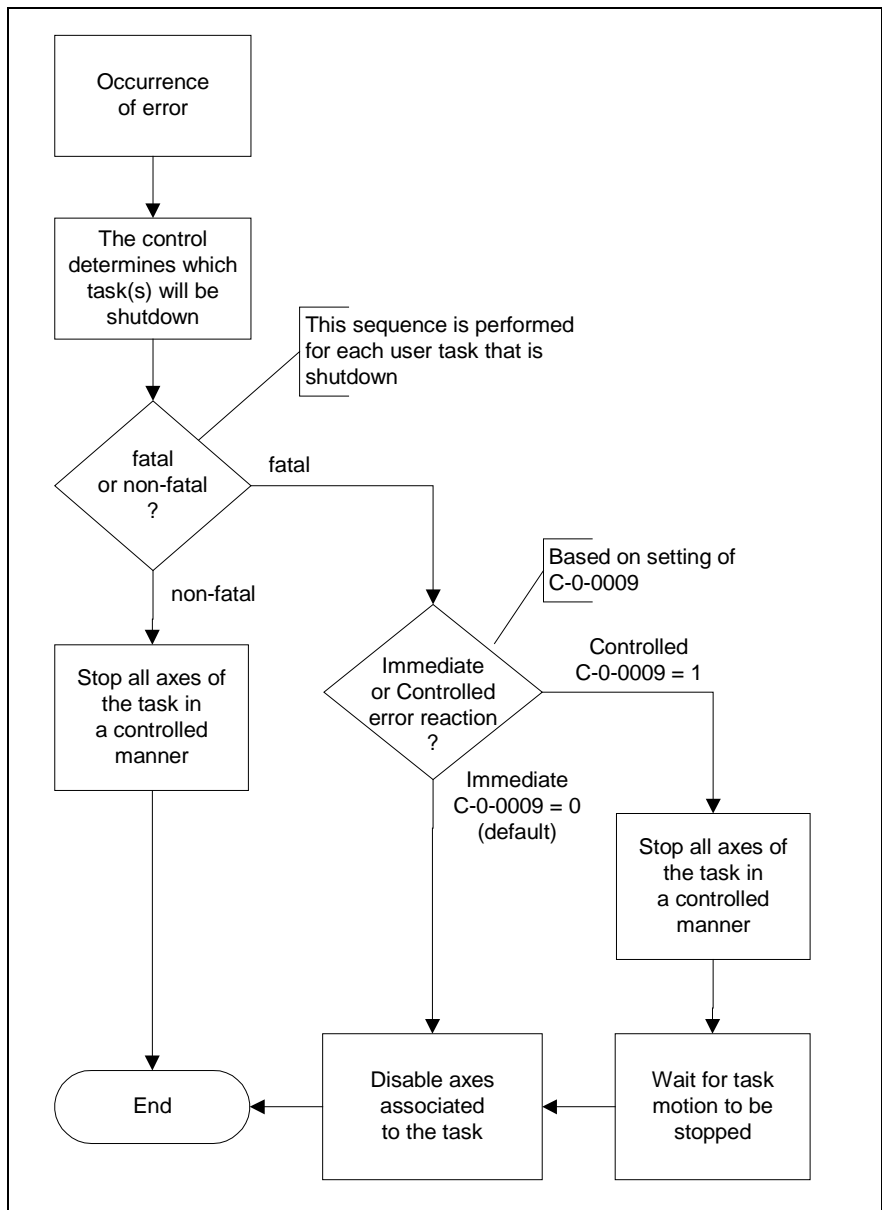


Fig. 12-2: Error Reaction Sequence

In the case of an error, moving axes can be disabled by the control and the error reaction of the drives can determine how the axes are stopped. Refer to *Drive Error Reaction* on page 12-12 for details.

Note: An axis can also be manually disabled by setting bit 1 of the relevant axis control register to 1.

Axes in ELS mode or coordinated motion mode can also be stopped in a controlled manner. This means that the ELS system master(s) or the coordinated motion path planner is ramped down to a standstill with the axes following in a synchronized/coordinated fashion. Refer to *Motion Type Error Reaction* on page 12-10 for details.

Fatal Errors

Fatal errors are considered the most severe. By default, the control reacts to fatal errors by shutting down the tasks and disabling the axes associated with the tasks when bit 1 of T-0-0002 is set to 0.

Some examples for fatal errors are:

- SERCOS disconnect error
- real-time operating system error (417 System Error)
- Emergency Stop (treated as a fatal error)

The error reaction for a fatal error depends on the parameterization of C-0-0009 (Error Reaction Mode).

Immediate Error Reaction for Fatal Errors (C-0-0009 = 0)

By default, all axes, regardless of their operating mode, are immediately disabled and stopped by the drive when C-0-0009 is set to 0. Refer to *Drive Error Reaction* on page 12-12 for details.

Virtual Masters in ELS mode and the path planner in coordinated motion mode are stopped instantaneously. Refer to *Motion Type Error Reaction* on page 12-10 for details.

Controlled Error Reaction for Fatal Errors (C-0-0009 = 1)

All motion comes to a controlled stop before the axes are disabled when C-0-0009 is set to 1. ELS axes stop synchronized and coordinated motion axes stop on path. Axes in single axis mode or velocity mode are switched to drive halt (AH). Refer to Table 12-4 and Table 12-5 for details.

Non-Fatal Errors

By default, the control reacts to non-fatal errors by stopping the ELS system and/or coordinated motion path planner in a controlled manner. Axes in single axis mode or velocity mode are switched to drive halt (AH). The task(s) are stopped, but the axes remain enabled when bit 1 of T-0-0002 is set to 0.

Some examples for non-fatal errors are:

- path planner errors
- program instruction errors (example: divide by zero in Calc icon)
- axis target position out of bounds

Warnings

A warning does not affect tasks and axes. User tasks remain in operation and all associated axes remain enabled. It is up to the user program to respond to warnings. For certain errors, the control can be configured to respond with an error or a warning, for example:

- fieldbus
- slip monitoring

Configurable Error Reaction

The following functions have a configurable error reaction. The following table lists the different settings:

Function	Parameter	Setting	Associated Error Code
RECO I/O Error Reaction	C-0-0010, bits 6-7	<u>bit 6</u> <u>bit 7</u> 0 0 = ignore 1 0 = warning x 1 = fatal error (default)	544 (fatal error) 215 (warning)
Response to a drive warning	C-0-0010, bit 13	0 = drive warning issues an error (default) 1 = ignore drive warnings	498 (non-fatal error)
Axis	C-0-0010, bit 14	0 = error if axis is not ready (default) 1 = no error is issued if axis is not ready	426 (non-fatal error) fatal error for coord. motion
	A-0-0006, bit 1	0 = ignore (default) 1 = issue error when drive is enabled but not referenced	500 (non-fatal error)
Fieldbus Interface	C-0-2635	<u>In case of fieldbus errors</u> 0x0000 = shutdown (default) 0x0001 = warning 0x0002 = ignore	519 (fatal error) 520 (fatal error) 208 (warning) 209 (warning)
Position Monitoring Groups	C-0-32x6, bit 2	0 = warning (default) 1 = fatal error	554 (fatal error) 220 (warning)
Parameter Transfer	T-0-0002, bit 3	0 = non-fatal error (default) 1 = warning	422 (non-fatal error) 205 (warning)
ELS Slip Monitoring	ELS_MSTR_CONFIG, bit 32	0 = fatal error (default) 1 = warning	552 (fatal error) 221 (warning)

Table 12-1: Functions with Configurable Error Reaction

Error Diagnostics and Logging

VisualMotion errors, which are written to C-0-0122, are also stored in the diagnostic log list parameter C-0-2020.

Diagnostic Parameters

Errors are identified by a specific error code number. Some similar errors share the same error code. For these errors, additional information is provided as an extended diagnostic message.

The following parameters are used for diagnostic messages, error codes, and extended diagnostic messages:

Parameter	Description
C-0-0122	Current diagnostic message (displayed on control's H1 display)
C-0-0123	Current diagnostic code
C-0-0124	Extended diagnostic message
T-0-0122	Task diagnostic message
C-0-2020	Diagnostic log

Table 12-2: Diagnostic Parameters

Refer to the *VisualMotion 9 Trouble Shooting Guide* for a complete listing of error codes.

- Accessing Error Codes** Diagnostic parameters and registers can be accessed via the serial, Ethernet or PCI interface using any of the following devices:
- VisualMotion Toolkit
 - HMI
 - PLC

Diagnostic Status Registers

VisualMotion provides the following status register bits to indicate that an error or warning has occurred. These read-only status register bits are set to 1, when an error or warning condition is present.

Register	Bit	Description
021: System Status	5	<i>Error</i> - set when any error or warning is present in the system
	6	<i>Error Active</i> - set when the current diagnostic is a fatal or non-fatal error
	7	<i>Warning Active</i> - set when the current diagnostic is a warning
022-025: Task Status	5	<i>Task Error</i> - set when the current diagnostic is a task error
031-038: Axis 1-8 Status 309-340: Axis 9-40 Status	13	Class 2 Warning
	14	Drive Shutdown Error

Table 12-3: Diagnostic Status Registers

Diagnostic Log

Errors are listed in the diagnostic log in the order in which they occurred. The most recent error is listed first. The diagnostic log can contain up to 100 of the most recent errors.

- Diagnostic Priority** A set of rules governs the priority with which diagnostic messages are displayed and logged. A non-fatal error will overwrite a warning and a fatal error will overwrite a non-fatal error or a warning.

12.2 Task Association and Setup

Axes and path planner are associated with user tasks in the Axis or ELS Group icon.

The error reaction for a given operating mode, such as single axis, or coordinated motion is processed through the associated user task. The ELS system is the only exception. The Virtual Master(s) and ELS Groups are associated with task A. If task A stops running, due to an error, so will the ELS system. The ELS slave axes can be associated with a different task than that of the ELS system. The association of an axis to a user task is performed in the Axis or ELS Group icons.

Task Error Reaction Setup

Each task in a VisualMotion project can be configured to react differently to errors. The programmer must determine which tasks should run and which tasks should be stopped when an error occurs.

Task Options

Task options are set in bits 1, 2, and 5 of task parameter T-0-0002. These bits can be set in VisualMotion Toolkit by right clicking on the desired task letter in the Project Navigator window and selecting **Properties....** The system must in SERCOS phase 2 (parameter mode) for changes to take effect.

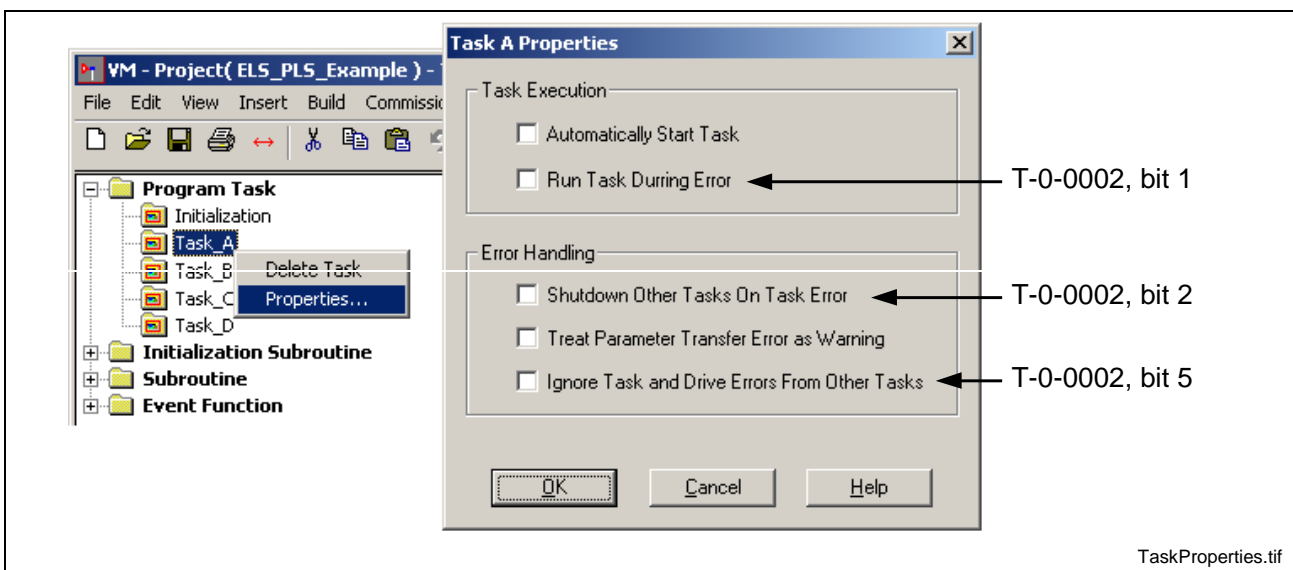


Fig. 12-3: Task Properties

Default Task Error Reaction

When bits 1, 2, and 5 of T-0-0002 are set to 0, any error that occurs will shutdown the task. For Fig. 12-4 to Fig. 12-7, the "Other Tasks" are configured with the default task error reaction.

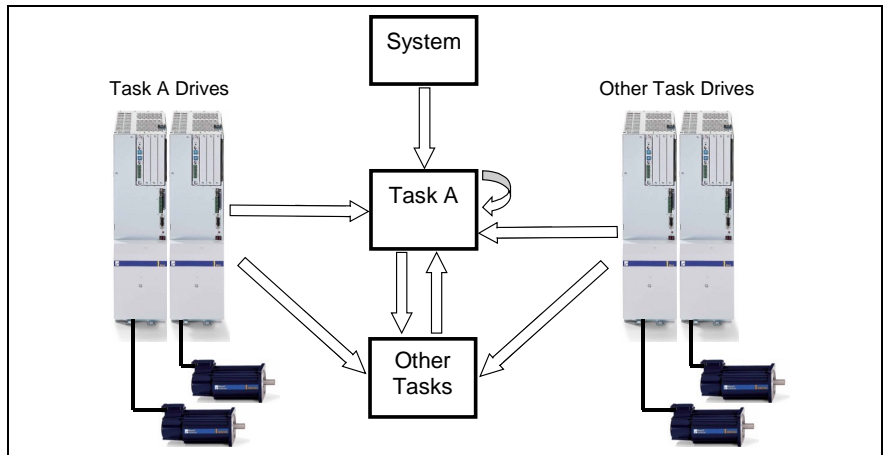


Fig. 12-4: Default Task Error Reaction

Run Task During Error (T-0-0002, Bit 1)

In Fig. 12-5, when bit 1 of T-0-0002 is set to 1, and bits 2 and 5 are set to 0, task A will run during errors unless an error occurs within task A. System errors, drive errors, and task errors of other tasks (B, C, D) will not shutdown task A. However, task A can shutdown other tasks.

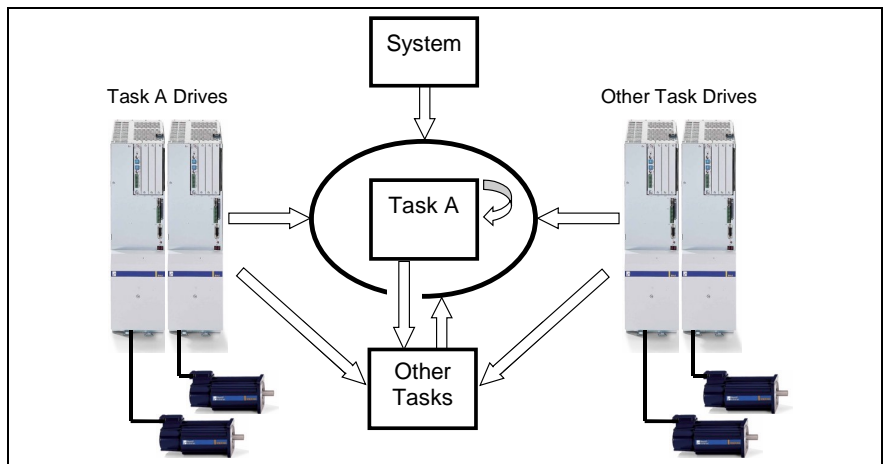


Fig. 12-5: Run Task During Error (T-0-0002, Bit 1)

Shutdown Other Tasks on Task Error (T-0-0002, Bit 2)

In Fig. 12-6, when bit 2 of T-0-0002 is set to 1 and bits 1 and 5 are set to 0, errors that occur within task A do not shutdown other tasks. System errors, drive errors, errors within task A, and task errors from other tasks will shutdown task A.

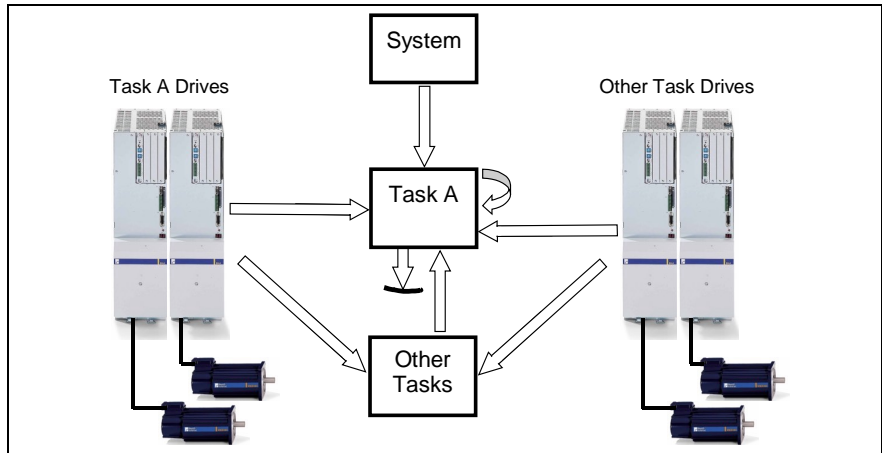


Fig. 12-6: Shutdown Other Tasks on Task Error (T-0-0002, Bit 2)

Ignore Task and Drive Errors from Other Tasks (T-0-0002, Bit 5)

In Fig. 12-7, when bit 5 of T-0-0002 is set to 1 and bits 1 and 2 are set to 0, task A will run during task errors that occur in other tasks, as well as drive errors from drives associated with other tasks. System errors, associated drive errors, and errors within task A will shutdown task A. However, task A can shutdown other tasks.

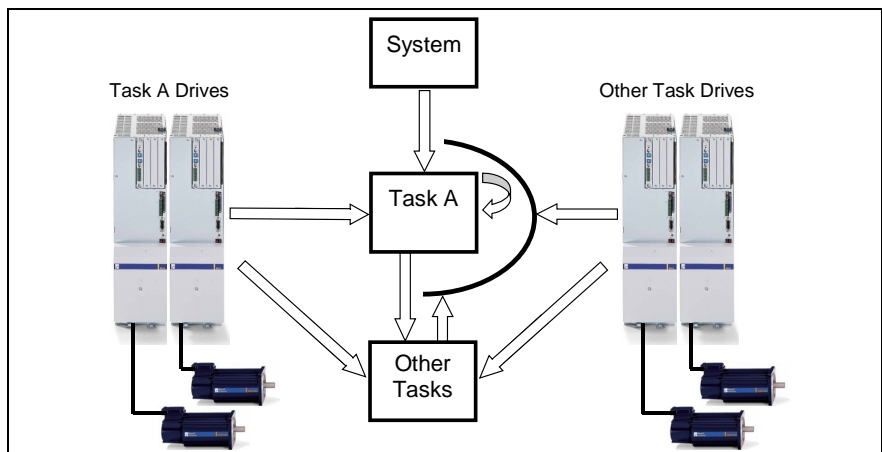


Fig. 12-7: Ignore Task and Drive Errors from Other Tasks

Task Shutdown Sequence

When a fatal or non-fatal error occurs, the control determines if the error is a system, task, or drive error. As the next step, the settings in task parameter T-0-0002 are checked to determine if the task should be shutdown. Axes, associated with the task that is shutdown, are stopped based on whether the error is fatal or non-fatal, their current mode of operation, and the parameterization of the error reaction of the drives. Refer to *Motion Type Error Reaction* on page 12-10 for details.

12.3 Motion Type Error Reaction

The error reaction varies based on the following factors:

- the current operating mode of the axis
- whether the error is non-fatal or fatal
- and the setting of control parameter C-0-0009

The following table lists the error reaction for non-fatal and fatal errors based on operating mode.

Operating Mode	Non-Fatal Error	Fatal Error		User Task Association
		Immediate (C-0-0009 = 0)	Controlled (C-0-0009 = 1)	
Torque Mode	No error reaction	Axis is disabled immediately **	Axis is disabled after all task motion has been stopped	Configured in Axis Icon
Velocity Mode	Axis is switched to drive halt *	Axis is disabled immediately **	Axis is switched to drive halt *	Configured in Axis Icon
Single Axis Mode	Axis is switched to drive halt *	Axis is disabled immediately **	Axis is switched to drive halt *	Configured in Axis Icon
Coordinated Motion	Follows path planner The axis remains enabled	Axis is disabled immediately **	Follows path planner The axis is disabled after all task motion has been stopped	Configured in Axis Icon
Path Planner	Decelerates using maximum deceleration in T-0-0022	Stops instantaneously	Decelerates using maximum deceleration in T-0-0022	Configured in Axis Icon
Virtual Master	Decelerates using float VM#_CMD_DECEL	Stops instantaneously	Decelerates using float VM#_E_STOP_DECEL	Task A
ELS Group	If Virtual Master is the current master, than it continues to follow it to a stop, otherwise it switches to local mode and decelerates using G#_STOP_DECEL	If the Virtual Master is the current master, than it stops instantaneously, otherwise it switches to local mode and decelerates using G#_STOP_DECEL	If Virtual Master is the current master, than it continues to follow it to a stop, otherwise it switches to local mode and decelerates using G#_STOP_DECEL	Task A
Drive ELS	Follows ELS Group The axis remains enabled	Axis is disabled immediately **	Follows ELS Group, then The axis is disabled after all task motion has been stopped	Configured in Axis Icon
Control CAM	Follows ELS Group The axis remains enabled	Axis is disabled immediately **	Follows ELS Group The axis is disabled after all task motion has been stopped	Configured in Axis Icon
Ratio	Follows master or external axis The axis remains enabled	Axis is disabled immediately **	Follows master or external axis, The axis is disabled after all task motion has been stopped	Configured in Axis Icon
Coordinated Articulation	Follows ELS Group The axis remains enabled	Axis is disabled immediately **	Follows ELS Group, then after all task motion has stopped, the axis is disabled	Configured in Axis Icon
* Refer to <i>Drive Halt Setup (AH)</i> on page 12-13 for details.				
** See parameterization of A-0-0004 and P-0-0119				

Table 12-4: Motion Type Error Reaction

Special Case Error Reactions for the ELS system

A special case is the error reaction for when an ELS slave axis is associated with tasks B, C or D. The Virtual Masters and ELS Groups are always associated with task A.

ELS Slave Axis Task Shutdown

The following error reaction occurs when a task, with an associated ELS slave axis, shuts down but task A continues to run:

- In the case of a non-fatal error, the ELS slave axis remains in its operation mode and follows its ELS Group
- In the case of a fatal error, the ELS slave axis is disabled and brought to a stop using the drive's error reaction

ELS Group Task Shutdown

The following error reaction occurs when task A shuts down, but the task with an associated ELS slave axis continues to run:

- The ELS slave axis remains in its operating mode and follows its ELS Group to a stop

Link Ring Error Reaction

The default error reaction for an ELS Group following a Link Ring master is to switch to Local Mode. The ELS Group decelerate to a stop using the rate in program variable G#_STOP_DECEL.

If this is not the desired error reaction, task A can be set to run during errors and the user can program an error reaction. Therefore, the user program should monitor the Node_#_Data_Valid bits in status registers 41 and 42 and run a custom error reaction when the relevant bit goes low.

For example, an ELS Group could perform an "On-the-Fly" switch to a free Virtual Master. The Virtual Master could then move the ELS Group to a safe position.

When a Link Ring master position is used as the input of an ELS Group, the validity of the Link Ring master position is monitored continuously by the control. Error "575 ELS master for ELS Group # is invalid" occurs when the current ELS master is considered to be invalid. The Link Ring master is only updated and valid when the control, providing the Link Ring master position, is in SERCOS phase 4.

12.4 Drive Error Reaction

The error reaction of the drive depends on the error level and the settings in the following drive parameters:

- P-0-0117, NC reaction on error
- P-0-0119, Best possible deceleration

Error Level	Diagnostic Code	Drive Response
Fatal	F8 xx	Drive is switched torque free. The parameterization of P-0-0117 and P-0-0119 is ignored because a drive reaction is impossible.
Travel Range	F6 xx	The velocity command value is immediately set to zero, regardless of the parameterization of P-0-0117 and P-0-0119.
Interface	F4 xx	The drive stops as defined in P-0-0119. A response from the control is impossible, since communication has stopped.
Non-Fatal	F3 xx F2 xx	P-0-0117, Bit 0 = 0, the drive stops as defined in P-0-0119. P-0-0117, Bit 0 = 1, the drive continues to follow the commands of the control for 30 seconds after the error occurred. The control then brings the axis to a controlled stop. Afterwards, the drive stops as parameterized in P-0-0119.

Table 12-5: Drive Error Levels

Power Supply Error Reaction

For applications where a modular drive concept is used (e.g., DIAX04), the user has a choice of how to configure the response of the power supply in the event of a drive error. In the case of an error and default configuration, a signal is sent to the power supply and the power supply removes the bus power from all drives. The user has the option to suppress or delay this in bits 0 and 2 of P-0-0118 (Power off on error). Refer to the relevant *Digital Drive* manual for details.

Disable Axis Command (Ab)

An axis is disabled by the control in the event of a fatal error. The default reaction (A-0-0004, bit 12 = 0) is to stop according to the parameterization of P-0-0119 and then the drive will remove torque from the motor.

In the case of a fatal error, torque is immediately removed from the motor and it will coast to a stop when bit 12 of A-0-0004 is set to 1.

An axis is manually disabled when bit 1 of the relevant axis control register is set to 1.

The following axis and drive parameters determine how the drive will react when the axis is disabled.

Parameter	Description
A-0-0004	bit 12: 0 = Stop axis according to P-0-0119 (Default) 1 = Torque Free
P-0-0119	<u>value</u> 0 Velocity command value set to zero 1 Switch to torque-free state 2 Velocity command to zero with command ramp and filter 3 Return motion

Table 12-6: Drive Disable Method

Drive Halt Setup (AH)

When the drive issues a drive halt, the axis is brought to a stop using a predefined deceleration rate. The deceleration rate that is used varies based on the active operating mode of the axis and the drive firmware being used.

The following table lists the firmware/operating mode combinations:

Drive (Firmware)	Operating Mode	Drive Halt Deceleration Rate
ECODrive (SGP03) ECODrive Cs (MGP01)	Single Axis	S-0-0359 (positioning deceleration) S-0-0260 (positioning acceleration) is used when S-0-0359 = 0
	Velocity	The following conditions apply: <ul style="list-style-type: none"> • P-0-1211 (deceleration ramp 1) is used when the current velocity is less than P-0-1202 (final speed of ramp 1). If P-0-1211 = 0, P-0-1201 (ramp 1 pitch) is used • P-0-1213 (deceleration ramp 2) is used when the current velocity is greater than P-0-1202 (final speed of ramp 1). If P-0-1213 = 0, P-0-1203 (ramp 2 pitch) is used Full torque is used if the active ramp pitch = 0
	All Others	S-0-0138 (bipolar acceleration limit value)
ECODrive (SGP01 or SMT02)	Single Axis	S-0-0260 (positioning Acceleration)
	Velocity	The following conditions apply: <ul style="list-style-type: none"> • P-0-1201 (ramp 1 pitch) is used when the current velocity is less than P-0-1202 • P-0-1203 (ramp 2 pitch) is used when the current velocity is greater than P-0-1202 Full torque is used if the active ramp pitch = 0
	All Others	S-0-0138 (bipolar acceleration limit value)
DIAX04 (ELS05)	All Modes	S-0-0138 (bipolar acceleration limit value)
DIAX04 (SSE03)	Single Axis	S-0-0260 (positioning acceleration)
	Velocity	P-0-1201 (velocity control, deceleration ramp 1)
	All Others	S-0-0138 (bipolar acceleration limit value)

Table 12-7: Drive Halt Deceleration Rate based on Firmware/Operation Mode

13 Communication Protocols

13.1 Protocol Overview

VisualMotion controls can receive and send data by means of a serial interface, network or PCI (PPC11.1) connection. By using a text-based ASCII protocol or binary SIS (standard Bosch Rexroth protocol) protocol described in this chapter, a variety of devices and programs can communicate with the PPC-P11.1 and PPC-R.

VisualMotion controls can be ordered preconfigured with a serial and/or Ethernet communication port. Communication between VisualMotion Toolkit and the control is accomplished across these ports using the VisualMotion DDE Server or the Scalable Communication Platform (SCP). By means of these communication platforms, the user can perform the following functions:

- edit system and drive parameters,
- edit data types, such as variables, point tables, etc.
- download user programs,
- provide diagnostic information,
- etc

Serial Interface Mode The RS-232, RS-422 and RS-485 serial communication interface modes are supported.

Features

- Transmission rate: 9600 through 115200 baud rate
- Maximum transmission path: 15 meters (approx. 50 ft) for RS232 and RS485
- 8 bit ASCII protocol
- no parity
- a stop bit

Network Mode The Ethernet interface is used to transmit data over a local area network or Internet.

Features

- Transmission modes: half and full duplex
- Transfer rate: up to 20 Mbps (**M**egabits **p**er **s**econd)
- Communication Protocol: TCP/IP
- Cable type: RJ-45

PCI Mode The PCI interface is used with the PPC-P11.1 PC card installed in an Industrial PC.

Features

13.2 Communication Protocols

VisualMotion 9 supports the following two protocols:

- ASCII Protocol (standard Bosch Rexroth ASCII format using the VisualMotion DDE server)
- SIS Protocol (standard Bosch Rexroth binary protocol using a Scalable Communication Platform SCP)

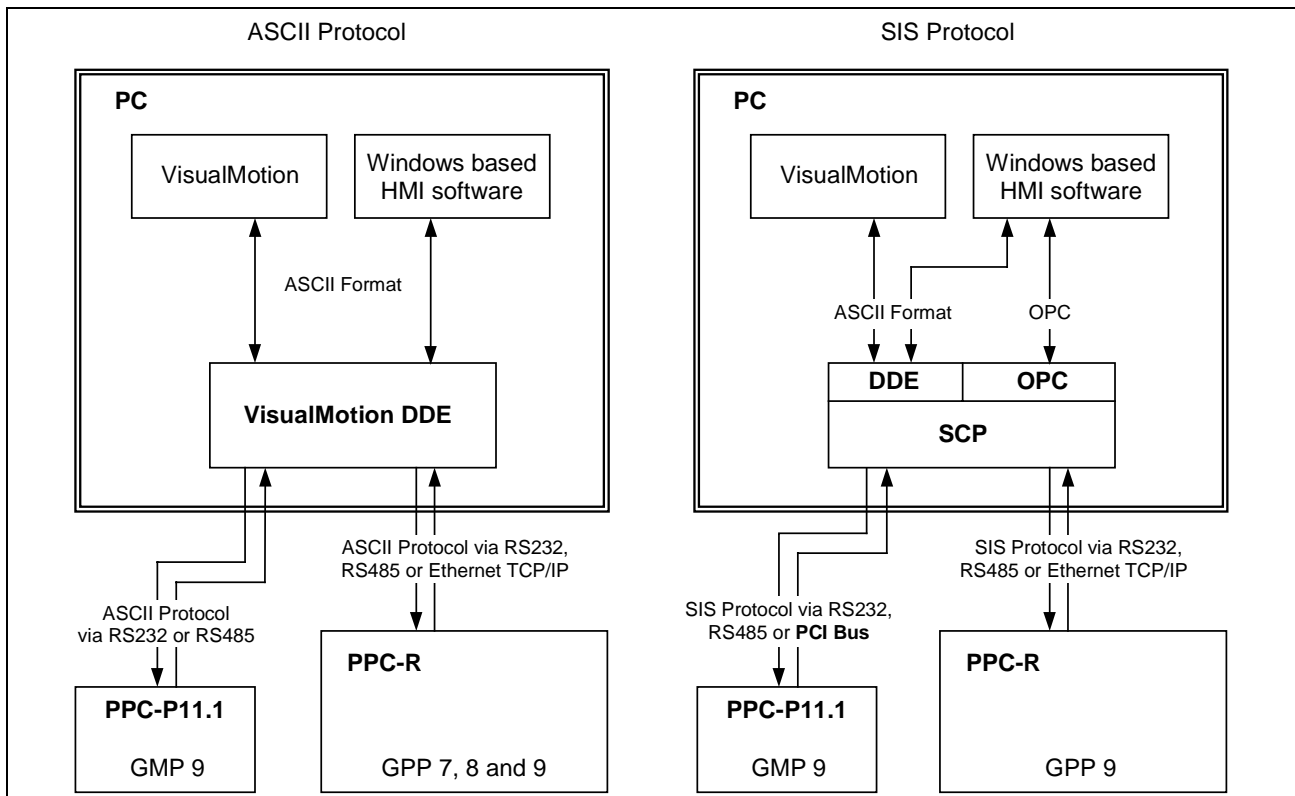


Fig. 13-1: ASCII and SIS Communication Overview

Note: OPC communication specifications can be found in the VisualMotion 9 Application Manual. This chapter will only focus on VisualMotion ASCII and SIS protocols.

ASCII Protocol

VisualMotion control's can send and receive drive parameters, system parameters, user programs, and tables by means of the serial port. By using the ASCII text-based protocol, a variety of devices and programs can communicate with the control. The protocol also supports ASCII PC ISA/ESIA communication.

Structure and Telegram frame

ASCII protocol does not use a telegram frame. Instead the transmitted ASCII string is converted and interpreted. The corrected syntax order must be maintain to successfully send and retrieve data.

SIS Protocol

The SIS protocol defines a peer-to-peer network where any node can be a client or server. Applications can be implemented to perform routing duties for hierarchical networks. SIS uses binary messages transported in telegram frames between a client and server for reading and writing data. Data such as registers, floating point and integer variables, control, axis, or task parameters, as well as drive parameters over various physical media, including EIA-232, EIA-485, and 10Base-T.

Features:

- Binary protocol
- A checksum test is conducted (higher Hamming distance D)
- All telegrams are identified by an unequivocal start symbol
- Defined telegram frame structure

Structure and Telegram frame

The telegrams used for data exchange over SIS is structured as follows:

- Telegram header
- User Data header (dependents on SIS service)
- User data

Communication Parameters

The data exchange is controlled by means of the following parameters:

- **C-0-0003**, baud rate and parity setup for X10
- **C-0-0004**, baud rate and parity setup for X16
- **C-0-0005**, enable communication protocol
- **C-0-0002**, device address

Reading Data from VisualMotion

To read data, the data portion of the string sent from the Host is empty.

Example The Host system requests a drive status message:

```
>1 DP 1.95 \r\n
      | _ No data sent, requesting current data
```

VisualMotion responds with the current status of the specified drive

```
>1 DP 1.95 302 Position Mode Encoder 1 $cs\r\n
      | _ Data (status message)
```

Writing Data to VisualMotion

To write data, send data in the data portion with the same starting protocol. Data received from VisualMotion can be sent in the same format. VisualMotion responds with an acknowledgment or an error message.

Example The Host sends VisualMotion the digital drive's Kv parameter for drive 1:

```
>1 DP 1.104 1.00$cs\r\n
      | _ New data sent
```

VisualMotion has successfully accepted the parameter, since no error message was returned:

```
>1 DP 1.104 $cs\r\n
      | _ No message: data stored successfully
```

Communication Errors

If there is a checksum error, a format error, or an error in the data sent to VisualMotion or the drive, VisualMotion returns an error string in the data field. The string starts with an exclamation "!" character, followed by an error code and a descriptive message. Communication error codes and messages are listed at the end of this section.

Example

```
>1 DP 1.104 !05 Greater than maximum value $cs\r\n
      || | _ Error message
      || | _ Error Code (decimal)
      | _ Error indicator "!"
```

Checksum

A control checksum is sent as two ASCII hexadecimal digits preceded by an ASCII '\$'. The checksum is optional when requesting data from VisualMotion. A checksum is required, unless it is disabled in the serial port setup parameter (C-0-0003 or C-0-0004), when sending data to VisualMotion. The following steps outline how to compute the checksum for **>1 AP 1.1 2:**

add the hexadecimal ASCII values of all of the characters, including the starting ">" character:

>	3E hex
1	31 hex
space	20 hex
A	41 hex
P	50 hex

space	20 hex
1	31 hex
.	2E hex
1	31 hex
space	20 hex
2	<u>32 hex</u>
02	22 hex

add the two least significant digits to the most significant digit:

22 hex two least significant digits

+2 hex most significant digit (calculated: sum shr 8)

24 hex sum

build the two's complement of the sum:

0 hex – 24 hex = FFFFFFFDC hex

take the two least significant digits:

DC

The string that is sent to the control should be: **>1 AP 1.1 2\$DC**

The algorithm for calculating the checksum for any type of direct ASCII communication is the same.

End of Message

An ASCII carriage return (CR = 13 in decimal or 0d hexadecimal) and linefeed (LF, 10 decimal or 0a hex) combination is used for terminal compatibility. This examples in this chapter use the notation '\r' (return) and '\n' (newline), as used with C programming language, interchangeably with the CR LF notation. VisualMotion always sends a CR LF combination, but will accept either a single LF or a CR LF from the Host device.

Backspaces and White Spaces

The ASCII backspace character (8 decimal or hex) erases the previous character from VisualMotion's serial buffer except at the start of a message. This is useful for editing strings entered at a terminal. Also, any white space character (tab or space) can be used as a delimiter in strings. White spaces between fields or at the end of a message are discarded by VisualMotion.

Numeric Data Formats

VisualMotion sends numeric data in ASCII parameter-specified units and scaling format. The format of float data depends on the data's use and how and where it is stored. Float data of fixed precision (e.g., drive data) uses fixed resolution. The resolution of data stored on the control (i.e., local or global variables), depends on the storage precision used (32 or 64-bit).

Float data that is too large or too small to be printed in decimal format is represented in scientific notation. *Hexadecimal data* is sent and received with an '0x' prefix. *Binary data* is represented as a 16 digit string of ASCII "1" or "0" characters.

- Example**
- **Float position data:** 0.0100 123.4567 -12.0000 12.3e+16
(resolution = 0.0001 units)
 - **Integer data:** 0 1000 -10

- **Hexadecimal data:** 0x12AB 0x1234ABCD
- **Binary data:** 0000111100001111

Format of Data Sent to VisualMotion

Any resolution can be used for data sent to VisualMotion. Numbers may be padded with zeros or spaces at either end as a visual formatting aid when entering data from a terminal. Padding applies to data identifiers as well as the data field.

The resolution of the data stored on the control is the resolution of the data it sends to the Host on request. Float numbers may also be sent in scientific notation.

Parameter Class and Subclass

VisualMotion 's System, Axis, Task, as well as Drive parameters follow the same general format. The subclasses are data elements (data, name, units, etc.) as specified in SERCOS. VisualMotion parameters include system configuration data that is entered during the configuration of the system, as well as continuously updated system status values and messages that monitor system operation.

Command Class	Command Subclass
A - VisualMotion Axis Parameters	A - Attributes
C - VisualMotion Control Parameters	B - Block List Parameter
D - SERCOS Drive Parameters	D - Lists/Tables
T - Task Parameters	H - Upper Limit
	L - Lower Limit
	P - Parameter Data
	T - Name Text
	U - Units Text

Table 13-1: Parameter Class and Subclass

Note: Additional information on the parameter sets can be found in chapter 5, Parameters, of this manual, and in Bosch Rexroth's digital drives and SERCOS manuals.

General Format

```
>l CP 1.122 $cs\r\n
  || | |_ number
  || |_ set: Axis, Task, or Drive Address
  |_ subclass: Parameter data, name Text, High or Low
  | limits, Attributes
  |_ class: type of parameter
```

Accessing SERCOS Parameter Sets

The SERCOS specifications allow a digital drive to have both standard and product specific parameter sets. The standard parameters are accessed by the parameter number (e.g., P-0-0095). Product specific parameters can be accessed using a 'P' prefix, which adds an offset of 32768 to the parameter number.

For example, parameter P-0-0005 can also be accessed as "1.32773" or "1.P5".

Example The Host requests the name, data, and units for drive 1 parameter 123:

```
>1 DT 1.123 \r\n ;request drive 1 text name for parameter
                    123
```

```
>1 DP 1.123 \r\n ;request drive 1 parameter data
```

```
>1 DU 1.123 \r\n ;request drive 1 units of measurement
```

The drive responds, through VisualMotion:

```
>1 DT 1.123 Feed Constant $cs\r\n ;the name is "Feed
                    Constant"
```

```
>1 DP 1.123 6.2832 $cs\r\n ;the parameter value is 6.2832
```

```
>1 DU 1.123 mm $cs\r\n ;the measurement units are in mm
```

Attribute Subclass

The "A" subclass requests a hexadecimal long word (16 bits). Bits in this long word are set for data type and scaling according to the SERCOS specification. The attribute data is available for informational purposes, or may be used to detect if a SERCOS parameter is a command or a status value.

Parameter List Block Transfer Subclass

For faster communications, VisualMotion can send and receive parameter lists 16 elements at a time. Drive parameter lists allowing block transfers include cam tables, oscilloscope data, and any other non-text parameter list. The 'B' subclass works similar to the 'D' parameter list subclass, but instead of sending one item at a time, 16 elements are sent.

General Format

```
>u xB a.s.n \r\n
    | | | | _Step: Sequence number (0 to length+1)
    | | | | _ Number: Parameter number
    | | | | _Set: Axis, drive, or task number
    | | | | _Subclass: Command, List Parameter or Table
    | | | | _Class: A=axis, C=control, D=drive, T=task
```

Requesting a Block List Parameter

To request the start of a list, the Host sends sequence number 0 to VisualMotion. VisualMotion responds with the number of elements in the list.

The number of steps in the list is equal to $((\text{elements} + 15) / 16)$. The Host requests this number of steps from the list until the list is finished.

The data in the response strings is space delimited. Float and decimal values are scaled the same as when they are printed individually.

If the number of elements in the list is not evenly divisible by 16, VisualMotion will fill the last response string with space-delimited zeros for each remaining element. If the data cannot be printed in less than 220 characters, the error message "I55 List or string is too long" is issued.

VisualMotion requires that step numbers be incremented by one, but any previous step may be repeated. This allows the host to request any missed data and ensures that the data is sent in the proper order. For example, the sequence (1, 2, 3, 3, 4) is valid, but (1, 2, 3, 5) is not. If an invalid step number is sent, VisualMotion responds with an error.

At the end of the upload, the Host must close the list by sending a sequence number equal to (length of list + 1). The Host must always close a list when it is finished since each new list uses system resources.

Example Host requests a list parameter using block transfer – Step 0:

```
>1 DB 1.32840.0 \r\n ;Parameter P-0-0072 (cam table 1)
```

VisualMotion responds with the number of elements in the list:

```
>1 DB 1.32840.0 1024 $cs\r\n ;1024 points in cam
table = 64 steps
```

Host requests first 16 elements in list – Step 1:

```
>1 DB 1.32840.1 \r\n
```

VisualMotion responds with first 16 elements:

```
>1 DB 1.32840.1 0.0 0.0015 0.002 0.01 0.015 --11
more elements...-- \r\n
```

Host requests elements 17-32 – Step 2:

```
>1 DB 1.32840.2 \r\n
```

VisualMotion responds with next 16 elements:

```
>1 DB 1.32840.1 20.0 20.0015 --14 more elements...--
\r\n
```

Host continues to request items in list as above - Step 3-64

To close the list, host sends sequence number (steps+1):

```
>1 DB 1.32840.65 \r\n
```

VisualMotion acknowledges end of list:

```
>1 DB 1.32840.65 !19 List is finished $cs\r\n
```

Sending a Block List Parameter

To start sending a block list, the Host sends sequence number 0 to VisualMotion, along with the number of elements to be sent. The number of steps in the list is equal to $((\text{elements} + 15) / 16)$. The Host sends this number of steps from the list until the list is finished.

The data in the strings must be space delimited. The host can send the data with any resolution, with or without decimal point.

If the number of elements in the list is not evenly divisible by 16, the host must fill the last string with space-delimited zeros for each remaining element.

If the number of elements in the string is less than 16, VisualMotion responds with the message “!54 List or String is too short”. If the length of the data portion of the string sent to VisualMotion (minus protocol header, checksum, and terminator) is greater than 220 characters, VisualMotion responds with the message “!55 List or string is too long”.

At the end of the download, the Host must close the list by sending a sequence number equal to length of list + 1. The string for this step must

include at least one data element. For simplicity, the host can send 16 space-delimited zeros.

Example Host starts sending a list parameter using block transfer – Step 0:

```
>1 DB 1.32840.0 1024 $cs\r\n ;Parameter P-0-72 (cam
                                table 1)
                                ;1024 points in cam
                                table = 64 steps
```

VisualMotion responds with an acknowledgment:

```
>1 DB 1.32840.0 $cs\r\n
```

Host sends first 16 elements in list – Step 1:

```
>1 DB 1.32840.1 0.0 0.0015 0.002 0.01 0.015 --11
more elements...-- $cs\r\n
```

VisualMotion acknowledges:

```
>1 DB 1.32840.1 $cs\r\n
```

Host continues to send items in list as above – blocks 2-64

To close the list, host sends sequence number (steps+1), with string having at least one zero:

```
>1 DB 1.32840.65 0.0 $cs\r\n
```

VisualMotion acknowledges end of list:

```
>1 DB 1.32840.65 !19 List is finished $cs\r\n
```

Parameter List Transfer of Control Cams

When control-based cams are sent to and received from the control, both X and Y values are transferred. Therefore, when the block data transfer method is used, only 8 rows of the cam file are transferred per string rather than 16. Therefore, the steps in the list is equal to $((\text{elements} + 7) / 8)$. The data in the string is transferred as: "X1 Y1 X2 Y2 X3 Y3", etc. If the control returns a count of 1025, the host must ask for 129 strings, plus the closing sequence.

Example Host requests control cam #1 – Step 0:

```
>1 CB 1.3101.0 \r\n ;Parameter C-0-3101 (cam table1)

>1 CB 1.3101.0 1025 \r\n ;VisualMotion indicates
                                1025 points (129 steps)
```

Host requests data and VisualMotion sends – Steps 1-129:

```
>1 CB 1.3101.1 \r\n

>1 CB 1.3101.1 0.0 0.0015 0.002 0.01 --12 more ...--
$cs\r\n
```

Host closes list (step +1) – Step 130:

```
>1 CB 1.3101.130 \r\n

>1 CB 1.3101.130 !19 List is finished $cs\r\n
```

Parameter Lists Subclass

The D subclass is used to request lists of parameters. Since new versions of VisualMotion and digital drives may expand or change the parameter sets, lists of all parameters and all required parameters can be uploaded by the Host program. Parameters such as the drive's oscilloscope function also use variable-length lists.

Some VisualMotion functions, parameters, and SERCOS parameters are implemented as variable-length data lists. Lists of parameters are used to determine all the parameters present in the drive or VisualMotion control, and to classify or request parameters by function or type. The drive oscilloscope function data tables are accessed as parameter lists. Sequence numbers are used to list each parameter in the list, allowing other transmissions to VisualMotion during the list upload.

General Format

```
>1 xD a.x.n \r\n
  || | | | _ Step: Sequence number (0 to length+1)
  || | | | _ Number: Parameter number
  || | | | _ Set: Axis, drive, or task number
  || | | | _ Subclass: Command, List Parameter or Table
  | _ Class: A=axis, C=control, D=drive, T=task
```

Listing a Parameter

To request a parameter list, the Host sends the list parameter command with the sequence number 0 to VisualMotion. VisualMotion responds with the sequence number replaced with a count of the number of items in the list. The Host then requests each list item sequentially, beginning with sequence number 1. The sequence number is then incremented by one and the request repeated until all needed items or the entire list has been received.

VisualMotion requires parameter list sequence numbers to be incremented sequentially. If an error occurs, the request for the current list item may be immediately repeated, allowing the Host to request missed data and ensuring that the data is sent in the proper order. VisualMotion will respond with an error if sent an invalid sequence number.

At the end of the upload, the Host must signal VisualMotion to close the list by sending a sequence number equal to the length of the list + 1.

If required, more than one parameter list can be active at one time. The Host must always close a list when it is finished since each open list uses system resources.

Example The Host requests a List Parameter – Step 0:

```
>1 DD 1.17.0 \r\n ;Parameter S-0-0017
```

VisualMotion responds with length of the list:

```
>1 DD 1.17.0 180 $cs\r\n ;180 parameters in list
```

The Host requests the first parameter in the list – Step 1:

```
>1 DD 1.17.1 \r\n
```

VisualMotion responds with the first parameter:

```
>1 DD 1.17.1 44 $cs\r\n
```

The Host requests the second parameter in the list – Step 2:

```
>1 DD 1.17.2 \r\n
```

VisualMotion responds with the second parameter:

```
>1 DD 1.17.2 104 $cs\r\n
```

.
. .
.

The Host continues to sequentially request items in list – Steps 3-180

The Host closes the list by sending a sequence number = the list length+1:

```
>1 DD 1.17.181 \r\n
```

VisualMotion acknowledges the end of the list:

```
>1 DD 1.17.181 !19 List is finished $cs\r\n
```

Upper Limit (H) and Lower Limit (L) Subclasses

The H and U subclasses return the range of permissible data entry that is set by VisualMotion or the digital drive for numeric data. Limits are always returned as float type data.

Parameter Data Subclass

The P subclass specifies the actual parameter data, sent and received in ASCII format according to its attributes (decimal, hexadecimal, text, etc.).

Name Text Subclass

The T subclass provides the name of the parameter, provided as a text string in the language selected for VisualMotion. The ability to specify actual text for the parameter name permits the host software to be independent of VisualMotion or drive parameter updates.

Units Text Subclass

The U subclass returns the system units, "in" (inches) or "mm" (millimeters), as an ASCII text string.

User Program Variable Class and Subclass

VisualMotion maintains a unique set of integer and float variables for each user program. An additional set of integer and float global variables is not related to a specific program and may be accessed by any program or device on the bus. User variable data can be exchanged between the Host and VisualMotion using the same format as the float and integer parameters. The current value of a variable is obtained and changed using the P subclass.

Command Class	Command Subclass
F - Float Variables	P - Data
G - Global Integer Variables	T - Text Label
H - Global Float Variables	
I - Integer Variables	

Table 13-2: User Program Variables Class and Subclass

General Format

```
>1 IP h.xx
  | | | _ number: variable table index number
  | | | _ set: Program handle
  | | | _ subclass: P=Send/receive Data, T=print text label
  | | | _ class: I=Integer Variable, F=Float Variable
                G=Global Integer, H=Global Float
```

The user program handle provides access to the variables for any control resident user program. Use the program handle "0" to access the active program's variables.

Data Subclass – “P”

Data in a VisualMotion variable table is accessed by supplying the class (I or F), and the numeric index (e.g. 1 for I[1] or 15 for F[15]) of the desired variable. The variable number "0" is used to request a count of the variables used by the selected program.

Type: Float ("F") or Integer ("I")

Text Label Subclass –“T”

The text label for any variable can be obtained by using the T subclass. If no text label is found, an ASCII space (" ") character is returned. Since the program labels are fixed when the program is compiled, labels cannot be changed with this command.

Type: String

Example The Host requests the number of integer variables used by program 1:

```
>1 IP 1.0 \r\n
```

VisualMotion responds:

```
>1 IP 1.0 20\r\n ;20 variables
```

The Host sends float data, 123.456 to Variable F12 for the program with handle 2:

```
>1 FP 2.12 123.456 $cs\r\n
```

VisualMotion acknowledges:

```
>1 FP 2.12 $cs\r\n
```

The Host requests the label name for Variable I20 for the current program:

```
>1 IP 0.20 \r\n
```

VisualMotion returns the name "count":

```
>1 IP 0.20 count\r\n
```

PID Class and Subclass

Proportional Integral Derivative (PID) information is only available from the active program.

Command Class	Command Subclass
M - PID Loop	B - Set Point Variable, Feedback Variable, Output Variable, First Variable of Control Block
	E - Calculated Set point
	F - Calculated Feedback
	G - Calculated Output
	J - Loop Time
	L - List of Valid PID Loop Numbers
	R - Control Register
	S - Status Register
	T - Type

Table 13-3: PID Class and Subclass

Getting the assigned variables

Example The host requests the control register of PID loop 2 of active program:

```
>0 MR 0.2 $cs\r\n
```

VisualMotion responds with register number, 0 if not assigned:

```
>0 MR 0.2 121 $cs\r\n
```

The host requests the status register of PID loop 2 of active program:

```
>0 MS 0.2 $cs\r\n
```

VisualMotion responds with register number, 0 if not assigned:

```
>0 MS 0.2 122 $cs\r\n
```

The host requests the variables of PID loop 2 of active program:

```
>0 MB 0.2 $cs\r\n
```

VisualMotion responds with set point variable, feedback variable, output variable, and start of control block:

```
>0 MB 0.2 F70 F71 F72 F143 $cs\r\n
```

The host requests the valid PID loop numbers of active program:

```
>0 ML 0.0 $cs\r\n
```

VisualMotion responds with loop numbers 2,3,7,8,9 (Other loops are not used):

```
>0 ML 0.0 2 3 7 8 9 $cs\r\n
```

VisualMotion responds with 0 (No loops are used):

```
>0 ML 0.0 0 $cs\r\n
```

Control Blocks Variables

The control block variables are defined by the PID loop type. The usage for type 1 is given above.

Point Tables Class and Subclass

Each coordinated motion program stored on the control has a unique set of absolute and relative points used to define the endpoints of the standard geometry segments. Each point contains a number of descriptive elements. Either all the elements of a point in the absolute or the relative tables may be accessed individually or as a list. The available elements are shown in the list of Command Classes and Subclasses on page 13-7.

Command Class	Command Subclass
X - Absolute Point Table Y - Relative Point Table	1 - Event 1
	2 - Event 2
	3 - Event 3
	4 - Event 4
	A - % of Max Acceleration
	B - Blend Radius
	D - % of Max Deceleration
	E - Elbow (6-Axis Robot)
	J - Jerk Limiting %
	L - List All of the Above in One Row
	P - Pitch (6-Axis Robot)
	R - Roll (6-Axis Robot)
	R - Rate (Kinematic 8 only)
	S - % of Max Speed
	T - Name Text
	V - List of All User Defined Labels
	W - Yaw (6-Axis Robot)
X - X Coordinate	
Y - Y Coordinate	
Z - Z Coordinate	

Table 13-4: Point Tables Class and Subclass

General Format

```
>1 XS h.nn \r\n
  || | |_ number: Point number
  || | |_ set: Program handle
  || | |_ subclass: Speed request
  | | |_ class: X=ABS, Y=REL
```

The program handle selects which of the user programs is accessed. To request the program points for the active program, use a "0" as the handle.

Point data is accessed by specifying a point number, that is used as a numeric index into the specified point table (E.G., 1 for ABS[1] or 21 for REL[21], the absolute or relative table is specified by the class). To request the total number of points in a table, use a "0" for the point index number.

Point Table Data

The X, Y, Z, R, P, and W coordinate subclasses are in the same format as float variables and parameters. The S, A, D, and J subclasses are in "xxx.x" format. Event subclasses 1, 2, 3, 4, and E are in the same format as integer variables. All subclasses have read/write access. The V subclass is a list of all user defined labels.

The following table lists each subclass, the type of units, the data type, and the data size.

Subclass	Units	Type	Size
1 - Event 1	event number	Integer	16-bit
2 - Event 2	event number	Integer	16-bit
3 - Event 3	event number	Integer	16-bit
4 - Event 4	event number	Integer	16-bit
A - Acceleration	% of max acceleration	Integer	8-bit
B - Blend Radius	distance	Float	32-bit
D - Deceleration	% of max deceleration	Integer	8-bit
E - Elbow State	none	Integer	8-bit
J - Jerk Limit	percent	Integer	8-bit
L - Absolute Points		Float	32-bit
X - X-Coordinate	distance	Float	32-bit
Y - Y-Coordinate	distance	Float	32-bit
Z - Z-Coordinate	distance	Float	32-bit
S - Speed	% of max speed	Integer	8-bit
R - Roll	degrees	Float	32-bit
R - Rate (Kinematic 8)	distance/time	Float	32-bit
P - Pitch	degrees	Float	32-bit
T - Name Text	none	Character	20
W - Yaw	degrees	Float	32-bit

Table 13-5: Point Table Data

Example

```
>1 YJ 0.12 \r\n ;host requests the jerk limit for
point REL[12];in the active program.

>1 YJ 0.12 50.0 $cs\r\n ;VisualMotion responds

>1 X1 3.1 2 $cs\r\n ;host selects event 2 for
;ABS[1]'s first ;event for program 3

>1 X1 3.1 $cs\r\n ;VisualMotion acknowledges
```


Point Table Data, Row Format

The data for a point is obtained in rectangular format, similar to a row in a spreadsheet program. A regular format allows faster downloading and easier conversion to and from spreadsheets and editor programs.

As returned by the control, the 12 elements of a point are separated by an ASCII space character, and the entire point table entry is terminated by a new line. Any number of spaces between elements is allowed when uploading to the control.

Example The Host requests number of absolute points in program 2:

```
>1 XL 2.0 \r\n
```

VisualMotion responds with 10 points:

```
>1 XL 2.0 10 10 10 10 10 10 10 10 10 10 10 10 10 10\r\n
      |_ number of points
```

The Host requests point ABS[1]'s data for the active program:

```
>1 XL 0.1 \r\n
```

VisualMotion Responds:

```
>1 XL 0.1 1.0 2.0 0.0 0.25 50 10 33 50 1 0 0 0 0.0 0.0 0.0 0
ABS[1]
;where   X   Y   Z   B   S   A   D   J   1 2 3 4 R   P   W   E
T
```

The Host requests label for ABS[1]:

```
>1 XT 0.1 \r\n
```

VisualMotion responds:

```
>1 XT 0.1 Robot_Start \r\n
```

The Host sends point REL[2]'s data for program 3:

```
>1 YL 3.2 1.0 2.0 0.0 0.25 100 60 33 50 1 10 0 0 $cs\r\n
```

VisualMotion Acknowledges:

```
>1 YL 3.2 $cs\r\n
```

List of All User-defined Labels Subclass (V)

For all classes that include user-defined labels, this subclass lists the labels. To start the list and get the total number of labels, request data with Step = 0. The labels are listed in the format '*Label_num Label_string*' from the serial port. The type 'Label' is used from the executive interface. Label_num is the index in the corresponding table; for example, '1' for ABS[1]. Label_string is a string of 21 bytes maximum, including null termination.

Event Tables Class and Subclass

Each user program stored on the control has an associated event table. Each element of the Event Table may be accessed individually or as a list. The available elements for the Event Table are shown in the list of Command Classes and Subclasses, and described in the Programming Elements section.

Command Class	Command Subclass
E - Event Table	A - Argument
	D - Direction
	F - Function
	L - List All Elements in One Row
	M - Message
	S - Status
	T - Type

Table 13-6: Event Tables Class and Subclass

General Format:

```
>l ES h.nn\r\n
  || | | _ number: Event number
  || | | _ set: Program handle
  || | | _ subclass: Status request
  | | | | _ class: Event Table
```

The user program handle specifies which user program is accessed. To request events for the active program, use a "0" as the handle.

Event data is accessed by using an event number as an index into the event table (E.G. 1 for EVT[1]). To request the total number of entries in the event table, use a "0" for the event index number.

Event Table Data

Most data in the event table has associated codes corresponding to events in the teach pendant display.

Subclass	Selections	Type	Access
A - Argument	A numeric value: (<u>milliseconds</u> if timed event) (% of the segment distance if coordinated motion) (<u>degrees</u> if repeating axis position event) Contains the probe position read from the drive, if feedback capture event is selected.	Float	Read/write
D - Direction	0 = start of move 1 = end of move	Unsigned Integer	Read/write
F - Function	Valid event function mark	String	Read/write
M - Message	text from 0-80 characters	String	Read/write
S - Status	0 = inactive 1 = queued 2 = pending 3 = executing 4 = done	Unsigned Integer	Read-only
T - Type	0 = undefined 1 = repeating timer 2 = time in coordinated path 3 = percent in coordinated path	Unsigned Integer	Read/write

Subclass	Selections	Type	Access
	4 = single axis distance 5 = repeating axis position (rotary) 6 = task input transition 9 = feedback capture 10 = I/O register event 11 = PPC-R X1 input events		

Table 13-7: Event Table Data

Example The Host requests number of events in program 2:

```
>1 ES 2.0 \r\n
```

VisualMotion responds with 50 events:

```
>1 ES 2.0 50 \r\n
```

The Host requests EVT[20].s for program 1:

```
>1 ES 1.20 \r\n
```

VisualMotion responds with ACTIVE:

```
>1 ES 1.20 1 $cs\r\n
```

The Host selects active EVT[10] as distance:

```
>1 ET 0.10 3 $cs\r\n
```

VisualMotion acknowledges:

```
>1 ET 0.10 $cs\r\n
```

The Host selects event function:

```
>1 EF 0.1 Gripper_On $cs\r\n
```

VisualMotion responds "Label was not found" in program:

```
>1 EF 0.1 !32 Label not found $cs\r\n
```

Event Table Data, Row Format

The data for each event is transferred between VisualMotion and the Host in a regular format that permits easy exchange of data with spreadsheet and text editing programs. Each element of an event is separated by a single space when sent from the control. When sending data to VisualMotion, any number of spaces may be used between the elements of an event. VisualMotion will not return an error if an event's status element is sent; the element will be ignored.

Examples The Host requests EVT[1]'s data for program 5:

```
>1 EL 5.1 \r\n
```

VisualMotion responds:

```
>1 EL 5.1 1 2 0 90.0 In_Zone In the zone! $cs\r\n
      S T D A F M
```

The Host sends data for program 2, EVT[19]:

```
>1 EL 2.19 0 1 1 1000 Calc_Zone Calculating... $cs\r\n
```

Program Communication Class and Subclass

Each user program (text or icon) that is developed and successfully compiled on a PC Host results in a downloadable base-code executable file. These executable files may be exchanged between the Host and VisualMotion using the upload/download method described in this section. Programs resident on the control can be activated and deleted; and lists of variables and subroutines associated with the program may be accessed using the serial communication interface. The number of programs that may be downloaded to a control is limited by the size of the programs and the amount of available non-volatile memory.

Command Class	Command Subclass
P - Program	A - Activate Program / Request Current Program
	C - Clear all Programs
	D - Upload/Download
	E - Delete Program
	F - Event Function List
	H - List of Program Headers
	I - Program Change Indicator
	J - CAM Indexer Block Information
	K - GPP Flash Compression
	L - Maximum Number of Rows in Sequence List
	M - Registration Block Information
	N - Name of Program
	Q - Maximum Number of Rows in Sequence Table
	R - Header Record: Start Upload
	S - Number of Function Slots Available
	T - Selective Table Transfer between Programs
V - List of Program Variable Labels	
W - Header Record: Start Download	
X - Transfer Tables between Programs	

Table 13-8: Program Communication Class and Subclass

User Program Header Record

Each user program contains a header record that identifies the program. This header is used for starting program uploads and downloads. The header contains the program name, size, checksum, and date; and is stored with the program in the control's non-volatile memory.

The size is a decimal count of all bytes in the program file. The checksum is a 4 byte hexadecimal two's complement sum of all the bytes in the file (excluding the checksum). The date is in Month/Day/Year format (E.G., 07/04/93). The program name can include any ASCII characters and can be up to a maximum of 20 characters.

Download Block Size

VisualMotion program file uploads and downloads are performed as a sequence of fixed-length blocks. The block size is selected using parameter C-0-0090 (Download Block Size) and defaults to 16 bytes. The Download Block Size parameter may be changed from 1 to 115 bytes by the Host computer to optimize transmission time or for terminal compatibility.

Activating a Program (PA)

The Host can activate a program by sending its program handle with the "PA" command. If the handle or the program is invalid, VisualMotion returns an error message.

Note: A program can not be activated by the Host if the control is currently running any user tasks.

The Host sends an activate program 2 command:

```
>1 PA 0.0 2$cs\r\n
      |_ Activate program with handle = 2
```

VisualMotion acknowledges:

```
>1 PA 0.0 $cs\r\n
```

Request Currently Active Program (PA)

If the Host sends a "PA" command without specifying a program, VisualMotion returns the handle of the currently active program. Other information about the program is available through task parameter requests.

The Host sends:

```
>1 PA 0.0 \r\n
```

VisualMotion returns:

```
>1 PA 0.0 2 $cs\r\n
      |_ Program 2 is active
```

Clear All Programs (PC)

This command is used to clear all programs residing in the control's flash memory. Once the command is issued, the memory is cleared but remains unusable until a PK (Flash Compression) command is send down to the control. It is recommended that a PK command be send to the control anytime data is deleted from flash memory.

Refer to GPP Flash Compression (PK) on page 13-25 for details.

The Host sends:

```
>1 PC 0.0 1 $cs \r\n
      |_ Clear ALL programs
```

VisualMotion returns:

```
>1 PC 0.0 \r\n
```

All programs were cleared from flash memory

Executing a Download (PD)

After the Host has sent the program header record and obtained a program handle, the Host must send the entire program to the control, before the program may be activated. The Host uses a sequential number to identify each block. Block by block transfer permits other communication with VisualMotion to take place between blocks.

If a communication error is received when sending a program block, the current block may be repeated. VisualMotion's download program requires program block sequence numbers to be incremented by one, or an error message will be returned.

After all program blocks have been downloaded, the Host completes the download by sending a final block with the data area containing at least one non-space character. The final block is identified by a sequence number equal to the last program block sequence number + 1. The checksum is verified and VisualMotion responds with an acknowledgment or an error message.

Note: The PD command must not be executed without a preceding PW or PR command to initialize the transfer.

Example The Host sends a block (using the default block size = 16 bytes):

```
>1 PD 1.1 0123456789ABCDEF0123456789ABCDEF$cs\r\n
| | | |_ 32 hex digits of data (16 bytes)
| | |_ Sequence number (from 1 to (size/block size))
| | |_ Program handle previously obtained by PW or PR
| | |_ command
| | |_ Command: Download
```

VisualMotion responds during download (no error):

```
>1 PD 1.1 $cs\r\n
```

The Host sends the final block after 100 program blocks have been sent:

```
>1 PD 1.101 0 $cs\r\n
| | |_ dummy block not stored by VisualMotion
| | |_ (size of program/block size) +1
```

VisualMotion responds after verifying the checksum:

```
>1 PD 1.101 !19 List is finished $cs\r\n
```

Executing an Upload (PD)

The PD command is also used during uploads, but the Host sends only the sequence number. VisualMotion responds with the data stored in each block. The process repeats sequentially, block by block, in the same manner as the program download (see Executing a Download, above).

The Host requests a block:

```
>1 PD 1.1 $cs\r\n
| | |_ program block sequence number
| | |_ program handle
| | |_ Command: upload
```

VisualMotion responds with data for the requested program and block:

```
>1 PD 1.1 0123456789ABCDEF0123456789ABCDEF$cs\r\n
```

The Host sends the dummy block request, after the last program block has been received:

```
>1 PD 1.101 $cs\r\n
  |_ (size of program/block size) +1
```

VisualMotion responds after checksum verification:

```
>1 PD 1.101 !19 List is finished $cs\r\n
```

Erasing (Deleting) a Program (PE)

This command erases the program identified with the specified program handle. The control's memory previously required by the program becomes available for other programs; and the header and handle are removed from the VisualMotion's list of resident programs.

-
- Notes:**
- 1) A currently active program may not be erased, unless the VisualMotion has been placed in Parameter Mode.
 - 2) The Erase Program command format specifies "0" in the positions normally occupied by the program handle and block sequence number, followed by the numeric program handle of the program to delete.
-

The Host sends an Erase Program Command:

```
>1 PE 0.0 1 $cs\r\n
  |_ Erase Program with handle #1
```

VisualMotion acknowledges:

```
>1 PE 0.0 $cs\r\n
```

List of Event Function Marks (PF)

Text marks for control resident event functions can be listed by using a list sequence. Subroutine marks are listed in alphabetical order, as stored in the user program file. The program handle is used to identify which program's events are required.

Request Format

```
>1 PF h.s
  | |_ sequence number
  |_ program handle
```

Response Format

```
>1 PF h.s Sub_Mark
  |_ subroutine mark (or function name)
```

Example The Host requests the function mark list for the active program:

```
>1 PF 0.0 \r\n
```

VisualMotion responds with the count of marks for the active program:

```
>1 PF 0.0 2 $cs\r\n
    |_ 2 marks are defined
```

The Host requests the first mark:

```
>1 PF 0.1 \r\n
```

VisualMotion returns:

```
>1 PF 0.1 Close_Gripper $cs\r\n
```

The Host requests the next mark (2):

```
>1 PF 0.2 \r\n
```

VisualMotion responds:

```
>1 PF 0.2 Open_Gripper $cs\r\n
```

The Host closes the list:

```
>1 PF 0.3 \r\n
```

```
|_ next sequence number after count of defined marks
```

VisualMotion acknowledges and closes the list:

```
>1 PF 0.3 !19 List is finished $cs\r\n
```

List Control Resident Programs (PH)

The "PH" command provides a list of the programs resident on the control. The list contains the headers of all programs and their corresponding program handles. An upload sequence of commands are used to obtain the list.

The host requests the start of the program list:

```
>1 PH 0.0 $cs\r\n
    |   |_ sequence 0= start of list
    |_ Command: list program headers
```

VisualMotion responds with the number of programs on the control:

```
>1 PH 0.0 4 $cs\r\n
    |_ number of resident programs
```

Host requests a file record from the list:

```
>1 PH 0.1 $cs\r\n
    |_ Sequence number for record 1
```

VisualMotion Responds:

```
>1 PH 0.1 1 1592 ABCD1234 12-25-1992 Program_1 $cs\r\n
    | | |<----- Program header ----->|
    | |_ Program handle for following program header
    |_ Sequence number for program header list
```

CAM Indexer Function Block Variables (PJ)

This communications request allows a user interface to determine the variables assigned to a CAM Indexer function block. A space delimited string in the following format is returned from VisualMotion: "cam_id

Fstart_float Istart_int". The CAM Indexer information is available only from the active program.

Format	Description
Cam_id	Cam number, range 1 to 8
start_float	Starting variable of the float data block
start_int	Starting variable of the integer data block

Table 13-9: CAM Indexer Function Block

In the data request, the Set (before the decimal point) is always equal to 0, indicating the active program. The Number (after the decimal point) is the cam id number block number. When block 0 is requested, the total number of active CAM Indexer blocks is returned.

Example Host: Ask for number of CAM Indexer blocks active in the active program:

```
>0 PJ 0.0
```

VisualMotion (No CAM Indexer blocks are present):

```
>0 PJ 0.0 0
```

VisualMotion (Two CAM Indexer blocks are present):

```
>0 PJ 0.0 2
```

Host: Ask for CAM Indexer block 1 information:

```
>0 PJ 0.1
```

VisualMotion: cam 1, floats at F10, ints at I10:

```
>0 PJ 0.1 1 F10 I10
```

GPP Flash Compression (PK)

This command is used to conditionally compress the flash memory on PPC hardware. Flash memory compression will only take place when less than 512K of available unused memory remains. VisualMotion may request a compression following a program download, deletion or activation. It is also recommended that a flash compression command be sent to the control for any of the above mentioned reasons or if any of the following parameters are written.

Parameter	Description
C-0-2017	I/O User Configuration List
C-0-2600 and C-0-2601	Fieldbus/Data Mapper Cyclic Channel
C-0-2641 and C-0-2642	PLC Interface Register Channel
C-0-2920 through C-0-2922	Option Card PLS Switches
C-0-2930 through C-0-2936	Option Card PLS Outputs
C-0-2940 through C-0-2943	Option Card PLS Masters
C-0-3000	I/O Mapper
C-0-3100	CAM Table Labels
C-0-3101 through C-0-3137	CAM Tables

Table 13-10: Parameters requiring Flash Compression

Example Host request flash compression:

```
>1 PK 0.0 100 \r\n
      |_ request for 100 percent compression
```

VisualMotion Acknowledges:

```
>1 PK 0.0 \r\n
```

To determine when compression is complete, continually request percent remaining until percent is zero.

Host request flash compression:

```
>1 PK 0.0 \r\n
```

VisualMotion responds:

```
>1 PK 0.0 88 \r\n
      |_ 88 percent compression complete
```

....

```
>1 PK 0.0 \r\n
```

VisualMotion responds:

```
>1 PK 0.0 0 \r\n
      |_ Compression complete
```

Registration Block Information (PM)

This communications request allows a user interface to determine the variables and registers that are assigned to a registration function block. A space delimited string in the following format is returned from VisualMotion: "probe_axis Fstart_float lstart_int Rcontrol_reg Rstatus_reg". The registration information is available only from the active program.

Format	Description
probe_axis	Axis (1-40) with the probe measurement value
start_float	Starting variable of the float data block
start_int	Starting variable of the integer data block
control_reg	Registration control register
status_reg	Registration status register

Table 13-11: Register Block Information

In the data request, the Set (before the decimal point) is always equal to 0, indicating the active program. The Number (after the decimal point) is the registration block number. When block 0 is requested, the total number of active registration blocks is returned.

Example Host: Ask for number of registration blocks active in the active program:

```
>0 PM 0.0
```

VisualMotion: No registration blocks are present:

```
>0 PM 0.0 0
```

VisualMotion: Two registration blocks are present:

```
>0 PM 0.0 2
```

Host: Ask for registration block 1 information:

```
>0 PM 0.1
```

VisualMotion: axis 1, floats at F10, ints at I10, control at R101, status at R102:

```
>0 PM 0.1 1 F10 I10 R101 R102
```

Request Name of Program (PN)

The "PN" command may be used to obtain the ASCII text name of a specified program handle from the control. If a handle of 0 is requested, the name of the currently active program is sent. If an invalid handle is requested, VisualMotion returns an error message.

The Host requests the active program:

```
>1 PN 0.0 \r\n
|_ Request active program's name
```

VisualMotion responds:

```
>1 PN 0.0 prog_one\r\n
```

The Host requests the name of program 3:

```
>1 PN 3.0 \r\n
|_ Request program 3's name
```

VisualMotion returns:

```
>1 PN 3.0 prog_3\r\n
```

Initializing an Upload (PR)

To request an upload from VisualMotion, the Host sends a program handle with the "PR" command. If the program handle does not match a program in the control, VisualMotion responds with an error. VisualMotion responds with the corresponding program header. (The "PH" command can be used to obtain a list of all control resident programs and handles.)

Example The Host requests an upload:

```
>1 PR 0.0 1 $cs\r\n
|       |_ Program handle
|_ Command: Start Upload
```

VisualMotion responds with the program's header:

```
>1 PR 0.0 1 1592 ABCD1234 12/25/92 Program_1 $cs\r\n
| |<----- Program Header ----->|
|_ Program handle
```

Selective Table Transfer between Programs (PT)

The control stores a unique set of tables for each user program. When a new program is downloaded, the tables of the new program are set to default or new values. The "PT" command allows the Host to selectively copy one or all of a working program's tables to another program. The transfer command copies selected tables associated with the source program to the destination program.

Tables are copied with the source table overwriting the destination table. If the source and destination tables are of different sizes, the table will be copied so that the destination table size does not increase. If the source table size is less than the destination, the remainder of the destination table will keep the same values.

The event functions in the source program are checked against those that exist in the destination program and new function offsets are stored. If an event function does not exist in the destination program it is set to "NONE".

Example

```
>1 PT s.d 1111111000000000 \r\n
    | | _ program handle of destination program
    | _ program handle of source program
```

Binary bit's value from left to right enables the following tables:

- Float variables table
- Integer variables table
- Point table
- Event table
- Zone table
- Sequencer table
- PLS table

Example The Host requests a copy of program 1's float and integer variable tables to program 3:

```
>1 PT 1.3 1100000000000000 \r\n

VisualMotion responds:

>1 PT 1.3 \r\n
```

List Variable Labels (PV)

Text labels for variables are stored with each program resident in the control, and can be listed using a list sequence. The variable numbers and text names are listed in alphabetical order by text name. The variable number consists of an "I" or "F" ASCII character, followed by a 1-to-3 digit number. The variable number and its label are separated with a space.

Request Format

```
>1 PV h.s
    | | _ sequence number
    | _ program handle
```


VisualMotion responds with a valid program handle:

```
>1 PW 0.0 1 $cs\r\n
      |_ Program handle returned by VisualMotion
```

Transfer All Tables Between Programs (PX)

The control stores a unique set of tables for each user program. When a new program is downloaded, the new programs tables are set to default or new values. The "PX" command allows the Host to copy a working program's tables to another program. The ability to copy a set of working, tested tables to another program under test can be exceedingly useful when debugging a program, or entering/modifying points using the teach pendant.

The transfer command copies all the tables associated with the source program to the destination program. The tables include: the float and integer variables, the absolute and relative point tables, and the event table.

Tables are copied on a one-for-one, table-to-table basis; with the source table overwriting the destination table. If the source and destination tables are of different sizes, the table will be copied so that the destination table size does not increase. If the source table size is less than the destination, the remainder of the destination table will keep the same values.

The event functions in the source program are checked against those that exist in the destination program and new function offsets are stored. If an event function does not exist in the destination program it is set to "NONE".

General Format

```
>1 PX s.0 d \r\n
      |   |_ program handle of destination program
      |_ program handle of source program
```

Example The Host requests that program 3 load a copy of program 1's tables:

```
>1 PX 1.0 3 \r\n
```

VisualMotion responds:

```
>1 PX 1.0 \r\n
```

Example The Host requests that program 1 load a copy of the active program's tables

```
>1 PX 0.0 1 \r\n
```

VisualMotion responds:

```
>1 PX 0.0 \r\n
```

Sequencer/Subroutine Related Subclasses (PI, PL, PQ, PS)

Subclasses:	I Data change identification (integer, read only) L Max number of rows in sequence list (integer, read only) Q Max number of rows in sequence table (integer, read only) S Number of function slots available (integer, read only)
Set:	program handle
Number:	for I subclass, 1=sequencer system for Q, L subclasses, always set to 0 for S subclass: 1=total number, 2=current available
Step:	not used
Exec functions:	get_Program(), put_Program()

Table 13-12: Sequencer/Subroutine Related Subclasses

These classes display the limits on sequence lists and tables that were set up at compile time.

The **I** subclass provides a way for a user interface to detect if data has been changed by another port or the user program. A counter is incremented each time data is stored. To check the sequencer system, set 'number' to 1. Other numbers are reserved for future functions.

The **L** subclass displays the maximum number of rows allowed in a sequence list.

The **Q** subclass displays the maximum number of rows allowed in a sequence table. This number is internal to the control and is set to 100 in the current version.

The **S** subclass displays information about the memory used for sequence tables. The functions entered into a sequence table are stored in memory as an array of function slots that is used by all sequences, which optimizes memory usage and editing. If 'number' is set to 1, the total number of function slots is returned. If 'number' is set to 2, the number of unused slots for functions remaining is returned.

Functions Class and Subclass

K Class - ELS Functions

The K class is used for runtime interaction with ELS Master, Virtual Master and ELS Group instructions. The G, L and M subclasses help the user determine the assigned registers, variables and drives associated with the system. ELS system assignments are only available for the currently active program.

Command Class	Command Subclass
K - ELS Functions	G - Group Registers and Variables
	L - List of Drives Assigned to Group
	M - Master Variables

Table 13-13: K Class - ELS Functions

Example The Host request ELS Group1's registers and variables:

```
>0 KG 0.1 \r\n
```

VisualMotion responds with control register, status register, starting float variable and starting integer variable:

```
>0 KG 0.1 152 241 F170 I140 \r\n
```

The Host request count of drives in ELS Group 1:

```
>0 KL 0.1.0 \r\n
```

VisualMotion responds with 1 drive:

```
>0 KL 0.1.0 1 \r\n
```

The Host request ELS Master's variables for current program:

```
>0 KM 0.0 \r\n
```

VisualMotion responds with control register, status register, starting float variable and starting integer variable:

```
>0 KM 0.0 140 141 F140 I110 \r\n
```


S Class - Function

All subroutines, tasks, and functions in a user program have attributes that can be read using this class. These are used for displaying names and validating data. All attributes are set in the user program and cannot be changed on-line. The function number is used in all references to this function in the sequence table.

Command Class	Command Subclass
S - Function Class	A - Access Type
	H - Maximum Value
	L - Minimum Value
	N - Argument Label
	R - Argument Attributes in a Row Format
	T - Function name
	V - Local Variable Attributes in Row Format
	Y - Type of Data

Table 13-14: S Class - ELS Functions

Subclasses

T	Function name	string, <= 21 byte (read only)
A	Access type	integer (1=not accessible, 2=accessible)
N	Argument label	string, <= 21 byte (read only)
Y	Type of data	1-character string: 'I', 'F', 'X', 'Y', or 'R' (read only)
L	Minimum value	float (read only)
H	Maximum value	float (read only)
R	Argument attributes in row format	string <= 81 byte (read only)
V	Local variable attributes in row format	string <= 81 byte (read only)

Table 13-15: S Class - Functions

- Set:** program handle
- Number:** function number (0 = total number of functions)
- Step:** argument number 1-5 (0= number of args for this function)
- Exec functions:** get_Function(), put_Function()

The **T** subclass returns the function name as a null-terminated string of 20 characters or less.

The **A** subclass displays (2) if the function can be used in a sequence table, (1) if it is hidden from the user.

The **N** subclass displays an argument name as a null-terminated string of 20 characters or less.

The data type **Y** is an ASCII character corresponding to the data class. This class can be used to access label lists for points and registers.

The minimum and maximum values are returned in the **L** and **H** subclasses. The user interface can read these values to determine if data entry is valid.

The **R** subclass provides the serial port interface with subclasses N, Y, L, and H for function arguments in a space delimited string.

Example >0 SR 0.1.1 test_var F 0.0 10000.0 \$cs\r\n

The **V** subclass provides the same information as the **R** subclass, but with local variables. Since there are no argument ranges for local variables, zeros are printed in the last two fields. The number of local variables in a task is obtained with the command ">n SV h.f.0", where n=unit, h=handle, and f=function.

I/O Registers Class and Subclass

The Host system may read VisualMotion's input and output registers at any time; including control, status and programmable registers. VisualMotion's axis, system and task status registers are normally read-only, and are only changed by VisualMotion's I/O Mapper executive task or by the following register forcing commands. Setting I/O registers directly, (using the RB, RX and RD commands) has the lowest priority of all I/O access methods.

Command Class	Command Subclass
R - I/O Registers	B - Current State in Binary
	C - Forcing State Change
	D - Current State in Decimal
	E - Erase all Forcing Masks
	F - Forcing Mask
	M - Current I/O State with Mask
	S - Binary Forcing State
	T - Name Text
X - Current State in Hexadecimal	

Table 13-16: I/O Register Class and Subclass

Directly accessing I/O registers should be done with caution. VisualMotion is a multitasking system, and as such the potential for I/O contention always exists between user tasks, the Host communication, the I/O Mapper, and the I/O subsystem.

Note: It's the programmer's responsibility to anticipate contention problems and synchronize access to data between asynchronous VisualMotion tasks when necessary.

The forcing commands (RE, RF, RC and RS) are provided primarily for debugging purposes. Forcing commands should be used with extreme caution since they can be used to override the state of system control registers, and have higher priority than VisualMotion's I/O Mapper or Host direct access commands.

The requirement for a checksum may be disabled by parameter. This practice is not suggested. It results in no communication error checking of data sent to VisualMotion that may effect safe operation of the system.

General Format

```
>1 Rt 0.nnn $cs\r\n
  | | | _ register number
  | | _ set ID, always 0 for I/O registers
  | _ subclass: type or format (B=binary, D==decimal,
    X=hexadecimal)
```

I/O Register Access (RB), (RX), (RD)

Input registers are accessed using "R(data type)" commands and a register specifying index number within the range of 1 to 200. The current contents of the register may be read as a 16-bit binary number (command "RB"), a 4-digit hex number (command "RX"), or a decimal integer number (command "RD").

Example Host requests the contents of register 1 in binary:

```
>1 RB 0.1 \r\n
```

VisualMotion responds:

```
>1 RB 0.1 0001001000110010 $cs\r\n
          |                               | _ least significant bit
          | _ most significant bit
```

The checksum is optional when reading data from VisualMotion.

Sending a "0" as the register index number returns the number of registers in the current system.

I/O Register Write

The Host may send a value to VisualMotion register in hexadecimal ("RX"), binary ("RB"), or decimal ("RD") using the same format as an I/O read with the addition of a data field and checksum.

Example

```
>1 RX 0.121 0x0040 $cs\r\n
  | | | | _ 16 bit hex word to write
  | | | _ I/O register number 121
  | | | _ always 0 for I/O registers
  | _ read/write to register in hex
```

I/O Forcing State Change (RC)

The most significant 16 bits in this 32-bit word selects which bits in the I/O register may be affected, and the least significant 16 bits change the states of those bits. If read, it returns the state of the all bits.

The data format of the "RC" state change word is always a 32-bit hexadecimal long word.

Example

```
>1 RC 0.2 0x00600040
          | | | | _ 16 bit word of new bit states
          | | | | _ 16 bit mask of bits to change
```

```
>1 RC 0.2 0x00600040
      |  |__ bit 6 on, bit 7 off
      |__ allow changes to bits 6 and 7
```

Erase All Forcing Masks (RE)

This command sets all forcing masks and states to zero and returns the I/O system to normal control. The command only takes effect at the time that it is sent.

Note: Caution should be used when using this command. The I/O registers are directly affected and clearing the mask(s) may cause immediate unwanted motion in the system

The data format of the "RE" command is ASCII integer:

Example

```
>1 RE 0.1 1
      |  |__ set to 1 to erase forcing masks
      |__ always '0.1'
```

I/O Forcing Selection (RF)

The forcing selection (RF) and forcing state (RC and RS) commands allow the Host to selectively force the state of individual bits in the I/O registers. Forcing commands take priority over VisualMotion's I/O Mapper and I/O devices.

The forcing remains in effect until the mask for each forced bit is cleared, or until there is a timeout error on the serial port. When the forcing state changes for bits in VisualMotion's control register, all edge detection is reset.

If a bit in the 16-bit forcing mask is set to 0, the corresponding bit in the I/O register is controlled by the I/O Mapper and physical I/O. If the forcing mask bit is set to 1, the I/O register bit is forced by the Host "RC" or "RS" commands.

The data format of the "RF" 16-bit forcing mask word is always binary.

Example

```
>1 RF 0.2 0000000001001000
      |__|__ bits 4 and 7 are forced bits and
      are controlled by the Host. All other
      bits are controlled by the physical I/O
      and VisualMotion I/O Mapper
```

Set Current I/O State with Mask (RM)

The RB, RD and RX commands affect every bit of the destination I/O register. The new data word replaces the old word. RM allows you to specify a mask in addition to data bits, limiting the I/O register bits that are changed.

The most significant 16 bits in this 32-bit word provide the mask selecting the bits that may be changed. A "1" enables change and a "0" masks any change. The least significant 16 bits changes the state of the I/O register bits. If RM is used to read the register, VisualMotion returns the state of the all bits.

Example >1 RM 0.2 0x00600040
 | | | | | | | | | | | | | | | | 16 bit word of new bit states
 | | | | | | | | | | | | | | | | 16 bit mask of bits to change
 >1 RM 0.2 0x00600040 ;bit 6 is turned on, bit 7 off

RM is a single use, independent equivalent of setting a mask with an RF command, then setting the actual I/O bit states with an RC or RS command. Since RM contains its own mask, it doesn't affect the forcing mask set with RF. Refer to the following RF, RC and RS commands.

I/O Binary Forcing State (RS)

The "RS" command is used to read and write the state of the forcing bits for the selected register. If bits are to be affected, the desired bits in the I/O register must have had forcing enabled by a forcing mask set with the "RF" command.

The data format of the "RS" 16-bit forcing state word is always binary.

Example >1 RS 0.2 0000000100000001
 | | | | | | | | | | | | | | | | bits 1 and 9 are turned on and
 all other bits turned off if the bits have
 forcing enabled by an "RF" command

Register Labels, Bit Labels (RT)

This command is used to request the label (name) that was assigned to a register or bit in the user program.

Example The Host request the label for register 1:

>1 RT 0.1
 | | identifies register 1

VisualMotion responds:

>1 RT 0.1 System_Control

The Host request the label for register 1 bit 5:

>1 RT 0.1.5
 | | identifies bit 5
 | | identifies register 1

VisualMotion responds:

>1 RT 0.1.5 Clear_All_Errors

Sequencer Data Class and Subclass

VisualMotion's serial protocol includes the following classes and subclasses to handle the sequencer and functions with arguments.

Command Class	Command Subclass
L - Sequencer List	T - Name of Sequence List
	P - Print or Store Sequence Tables

Table 13-17: L - Sequencer Data Class and Subclass

Sequence List Class (L)

Subclasses:

T	Name of Sequence List	string, <= 21 byte (read/write)
P	Print/Store Sequence Tables (rows of list)	integer sequence table number

Table 13-18: Sequence List Call (L)

Set: program handle
Number: Sequence List Number (0 returns total number of lists)
Step: sequence table row in this list (0 returns number of rows)
Exec functions: get_SequenceList(), put_SequenceList()

The **T** subclass reads or writes the sequence list name as a null-terminated string of 20 characters or less.

The **P** subclass prints or stores a sequence list as a variable-length list. The tables in the list are identified by the 'number' in the 'Q' data class.

To store a new sequence list, the host sets the 'step' to 0 and sends the size of the list. Setting the size of the list to 0 erases all entries in the list.

To change an existing row in the sequence list, 'step' must be set to the row number.

If a sequence list is edited while it is running, the communication error cannot edit sequence table while running is issued.

Sending a new sequence list Host tells VisualMotion to store list 1 with 5 function tables:

```
>0 LP 0.1.0 5
```

VisualMotion acknowledges:

```
>0 LP 0.1.0
```

Host selects Table number 2 as first table:

```
>0 LP 0.1.1 2
```

VisualMotion acknowledges:

```
>0 LP 0.1.1
```

Host selects Table number 1 as second table:

```
>0 LP 0.1.2 1
```

VisualMotion acknowledges:

```
>0 LP 0.1.2
```

Host closes the list:

```
>0 LP 0.1.6 0
```

VisualMotion acknowledges, table now ready to execute:

```
>0 LP 0.1.6 !19 List finished
```

Changing one row of sequence list Host changes second row in list to Table number 1:

```
>0 LP 0.1.2 2
```

VisualMotion acknowledges:

>0 LP 0.1.2

Sequence Table Class (Q)

Command Class	Command Subclass
Q - Sequence Table	1 - Argument 1
	2 - Argument 2
	3 - Argument 3
	4 - Argument 4
	5 - Argument 5
	F - Function Number
	P - Print/Store Function Numbers and Arguments
	T - Table Name

Table 13-19: Q -Sequencer Data Class and Subclass

Subclasses:

1	Argument 1	float (read/write)
2	Argument 2	float (read/write)
3	Argument 3	float (read/write)
4	Argument 4	float (read/write)
5	Argument 5	float (read/write)
F	Function number	integer (read only)
P	Print/Store function numbers and arguments (rows of table)	string, <= 255 byte or Special structure (read/write)
T	Table Name	string, <= 21 byte (read/write)

Table 13-20: Sequence Tqtable Class (Q)

- Set:** program handle
- Number:** table number (0 returns total number of tables)
- Step:** function row in this table (0 returns number of rows in this table)
- Exec functions:** get_SequenceTable(), put_SequenceTable()

The **T** subclass reads and writes the sequence table name as a null-terminated string of 20 characters or less.

The **P** subclass prints or stores a sequence table as a variable-length list. The functions in the list are identified by the 'number' in the 'S' data class.

To store a new sequence table, the host sets the 'step' to 0 and sends the size of the table. Setting the size of the table to 0 erases all entries in the list.

To change an existing row in the sequence table, 'step' must be set to the row number.

Through the serial protocol the data is a space-delimited string in the format (function number, arg1, .. argn). Arguments are printed as float

values. Arguments that don't exist are printed as '0'. VisualMotion will ignore extraneous arguments when data is sent to it.

From the executive interface, the data is type Special with the structure FUNC_DATA_t. All elements of a function are passed and returned using this structure (see end of this document).

The function name can be requested with the "ST" protocol command. Argument names, types, and limits can be requested with the 'S' data class.

If a sequence table is edited while it is running, the communication error cannot edit sequence table while running is issued.

Subclasses F, 1, 2, 3, 4, and 5 can be used to individually access elements of each row of the table. The individual arguments can be changed while the sequence table is running.

Refer to Sending a new sequence list for an example.

Getting a sequence table from VisualMotion

Host asks for number of functions in table 2:

```
>0 QP 0.2.0
```

VisualMotion responds:

```
>0 QP 0.2.0 3
```

Host asks for first function:

```
>0 QP 0.2.1
```

VisualMotion responds with function number 2:

```
>0 QP 0.2.1 2 12.34 100 25 0 0
```

Host asks for second function:

```
>0 QP 0.2.2
```

VisualMotion responds with function number 5:

```
>0 QP 0.2.2 5 100 0 0 0 0
```

Host asks for second function:

```
>0 QP 0.2.3
```

VisualMotion responds with function number 1:

```
>0 QP 0.2.3 1 12.5 30.2 0 0 0
```

Host closes the list:

```
>0 QP 0.2.4
```

VisualMotion acknowledges:

```
>0 QP 0.2.4 !19 List is finished
```


Control PLS Class and Subclass

Class	Subclass
W: PLS(Programmable Limit Switch)	A - Axis Number
	E - A Switch's On Position
	F - A Switch's Off Position
	G - Switch Lead Time
	H - On & Off Value and Lead Time
	M - Mask Register
	O - Phase Offset
	R - Assigned Register
T - Master Type	

Table 13-21: Control PLS Class and Subclass

General Format

```
>1 Wn h.n\r\n
  || | |_number: Control PLS number (1 or 2)
  || |_set: Program number
  ||_subclass: n = A, E, F,G, H, M, O, R, T
  |_class: PLS
```

The number of switches per program can be requested by sending any request with number equal to 0.

Examples

Host requests PLS count for program 5:

```
>1 WR 5.0 \r\n

VisualMotion responds one PLS:

>1 WR 5.0 1
```

Host requests assigned register to PLS 1 of program 5:

```
>1 WR 5.1 \r\n

VisualMotion responds register 100:

>1 WR 5.1 100
```

Format of Configuration Data

A - Axis	Axis Number(1 - 40)	Integer	8-bit
R - Assigned Register	0=Not Active, 1-1024	Integer	8-bit
O - Offset from Master	distance/degrees	Float	32-bit
T - Master Type	1=ELS, 2=VM, 3=AP, 4=AS	Integer	8-bit

Table 13-22: Control PLS Data Configuration

General Format

```
>1 WH h.n.m\r\n
  || | | |_switch number
  || | |_number: PLS Control number
  || |_set: Program number
  ||_subclass: On/Off Switch Values
  |_class: PLS
```

The number of switches per Control PLS can be requested by sending a request with the switch number equal to 0.

Format of Switch Values

Example Host requests the on position for Control PLS 1, switch 6, of program 4:

```
>0 WE 4.1.6 \r\n
```

VisualMotion responds:

```
>0 WE 4.1.6 98.0
```

Host requests the off position for Control PLS 1, switch 6, of program 4:

```
>0 WF 4.1.6 \r\n
```

VisualMotion responds:

```
>0 WF 4.1.6 167.9
```

Host requests the lead-time for Control PLS 1, switch 6, of program 4:

```
>0 WG 4.1.6 \r\n
```

VisualMotion responds:

```
>0 WG 4.1.6 65 \r\n\
```

Host request on/off positions and lead time of switch 1 of PLS 1 in active program:

```
>0 WH 0.1.1 \r\n
```

VisualMotion responds with on position of 25.1, off position of 35.2 and a lead-time of 55 ms:

```
>0 WH 0.1.1 25.1 35.2 55 \r\n
```

Host request on/off positions and lead time for PLS 1, switch 6, of program 4:

```
>0 WH 4.1.6 \r\n
```

VisualMotion responds:

```
>0 WH 4.1.6 98.0 167.9 65 \r\n
```

Host requests mask register for PLS 1 of program 4:

```
>0 WM 4.1 \r\n
```

VisualMotion responds:

```
>0 WM 4.1 201 \r\n
```

Drive PLS ASCII Protocol

Some SERCOS drives have a PLS associated with them. The following Bosch Rexroth digital drives using the specified firmware have Drive PLS functionality:

- DIAX04 using ELS05VRS or greater allow up to 8 PLS switches.
- ECODRIVE03 using SMT02VRS or SGP01VRS or greater allow up to 16 PLS switches.

Switches of a Drive PLS can be accessed individually or as a list. The available elements are described in the PLS Help in VisualMotion.

List of PLS Switches Host requests the count of On switch's for drive 1:

```
>1 DD 1.32900.0 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.0 8
```

Host requests the first On switch for drive 1:

```
>1 DD 1.32900.1 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.1 25
```

Host requests the last On switch for drive 1:

```
>1 DD 1.32900.8 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.8 350
```

Host requests one more switch to close list for drive 1:

```
>1 DD 1.32900.9 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.9 !19 List is finished
```

Individual Switches The subclass 'E' of the ASCII protocol for drive parameters allows individual switches of the list to be accessed.

Host requests the 5th Off switch for drive 1:

```
>1 DE 1.32901.5 \r\n
```

VisualMotion responds:

```
>1 DE 1.32901.5 270
```

Host writes the 4th Offset swicth for drive 3:

```
>1 DE 3.32902.4 220\r\n
```

VisualMotion responds:

```
>1 DE 3.32902.4
```

Zones Class and Subclass

Each user program stored on the control has a zone table associated with it. All elements of an event in the Zone Table can be accessed individually or as a list. The available elements are shown in the list of Command Classes and Subclasses.

Command Class	Command Subclass
Z - Zones	A - Upper X Coordinate
	B - Upper Y Coordinate
	C - Upper Z Coordinate
	D - Lower X Coordinate
	E - Lower Y Coordinate
	F - Lower Z Coordinate
	L - List All Elements in One Row
	S - Status

Table 13-23: Zones Class and Subclass

Example

```
>1 zS h.nn\r\n
  || | _number: Zone number
  || | _set: Program handle
  | | _subclass: Status request
  | _class: Zone Table
```

The user program handle identifies which zones's events are accessed. To request the active program's zones, send a '0' as the handle. To request the number of zones in the table, send a 0 for the point number with any type of request.

Format of Printed Data

Subclasses	S	Status	0=inactive, 1=active	Integer	8-bit
	A	Upper X-coordinate	distance	Float	32-bit
	B	Upper Y-coordinate	distance	Float	32-bit
	C	Upper Z-coordinate	distance	Float	32-bit
	D	Lower X-coordinate	distance	Float	32-bit
	E	Lower Y-coordinate	distance	Float	32-bit
	F	Lower Z-coordinate	distance	Float	32-bit

Table 13-24: Zones Printed Data Format

Examples Host requests number of zones in program 2:

```
>1 zS 2.0 \r\n
```

VisualMotion responds 50 zones:

```
>1 zS 2.0 50\r\n
```

Host request zone 1 status for prog. 1:

```
>1 ZS 1.20 \r\n
```

VisualMotion responds with ACTIVE:

```
>1 ZS 1.20 1 $cs\r\n
```

Host sends upper X to zone 10:

```
>1 ZA 0.10 3.0 $cs\r\n
```

VisualMotion acknowledges:

```
>1 ZA 0.10 $cs\r\n
```

Data in Row Format

The data for a zone can be printed and stored as it would appear in a row of a text file or a user interface.

The seven elements of a zone are each separated by a space when sent from the control. Any number of spaces between elements can be sent to the control.

Examples Host requests zone one data for program 5:

```
>1 ZL 5.1 \r\n
```

VisualMotion responds:

```
>1 ZL 5.1 1 1.0 2.0 0.0 3.0 4.5 0.0  
      S A B C D E F
```

Host sends zone 19's data for program 2:

```
>1 ZL 2.19 1 1.0 2.0 0.0 3.0 4.5 0.0
```

13.4 SIS Communication

SIS Protocol

The SIS protocol defines a peer-to-peer network where any node can be a client or server. Applications can be implemented to perform routing duties for hierarchical networks. SIS uses binary messages transported in frames between a client and server for reading and writing data. Data such as registers, floating point and integer variables, control, axis, or task parameters, as well as drive parameters over various physical media, including EIA-232, EIA-485, and 10Base-T.

Note: This section only covers basic information on Common and GPP-Specific SIS services. Refer to the information manual, "DOK-GENERL-SIS-DEFINIT-IF02-EN-P", for detailed information, including examples.

Telegrams

The telegrams used for data exchange over SIS are generally structured according to the following scheme:

- Telegram header
- User data header (depends on the SIS service)
- User data

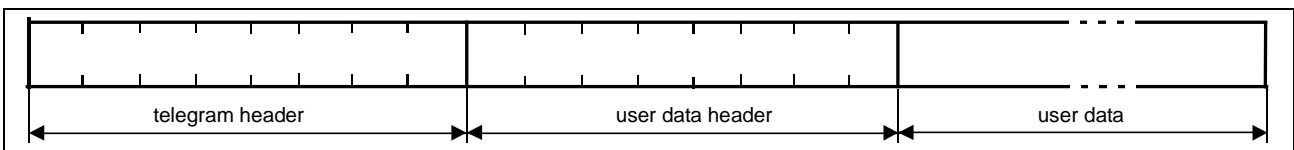


Fig. 13-1: Telegram Structure for Data Exchange via SIS

However, the exact structure of the individual telegrams depends not only on the SIS service but also on the telegram direction and the telegram type.

Telegram Directions and Types

Telegram Directions

The following two telegram directions are supported over SIS.

Telegram Direction	Sender of the Telegram
Command telegram	Master: the 'active' communications partner
Response telegram	Slave: the 'passive' communications partner

Fig. 13-2: Telegram Directions

Theoretically, with a RS485 connection every bus user can be both a master - i.e. can send a command telegram that is then answered by the slave with a response telegram – as well as a slave, simultaneously. However, in practice there is a distinction between master partners (control devices) and slave partners (target devices).

Telegram Types in the Command Telegram

There are four types of command telegrams for the master:

Telegram Type	Data Direction
SEND telegram	Write access: data sent to a slave
RETRIEVE telegram	Read access: data requested from a slave
Group message	Message: data sent to a group of slaves
Broadcast message	Message: data sent to all slaves

Fig. 13-3: Telegram Types in the Command Telegram

Note: Group and broadcast messages are **not** answered by the slaves addressed.

Telegram Types in the Response Telegram

As appropriate, a slave that is addressed at its own address sends a response telegram as follows.

Telegram Content	Telegram Type
Transmission status	with error-free SEND telegram
Transmission status and requested data	with error-free RETRIEVE telegram
Transmission status and error code	with erroneous SEND or RETRIEVE telegram

Fig. 13-4: Telegram Content of the Response Telegram

Sequential Telegrams

Long data sets in the various forms of telegram directions and types may have to be subdivided among several subtelegrams and sent as sequential telegrams.

Structure of the Command Telegram

Write Access

A SEND telegram is made up of:

- Telegram header
- User data header (depends on the SIS service)
- User data

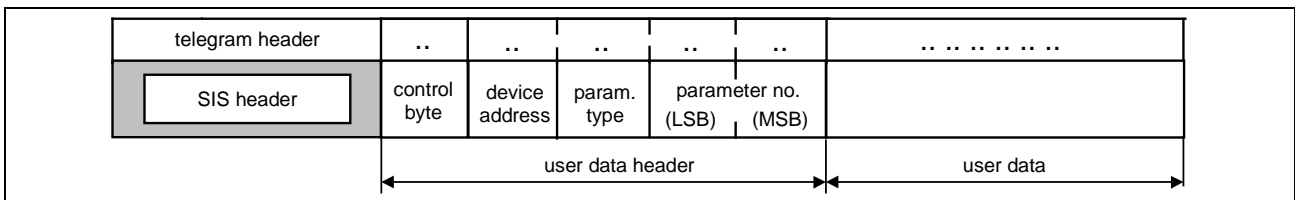


Fig. 13-5: Structure of the Command Telegram: Write Access

Read Access

A RETRIEVE telegram is made up of:

- Telegram header
- User data header (depends on the SIS service)

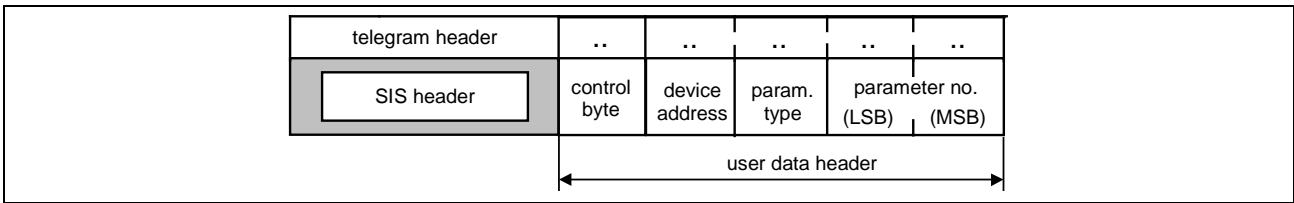


Fig. 13-6: Structure of the Command Telegram: Read Access

The user data header for specifying the read access can also be omitted if it is already set in the telegram header by the SIS service.

Structure of the Response Telegram

Write Access The response telegram is made up:

- Telegram header
- User data header (depends on the SIS service)
- Error code, if applicable

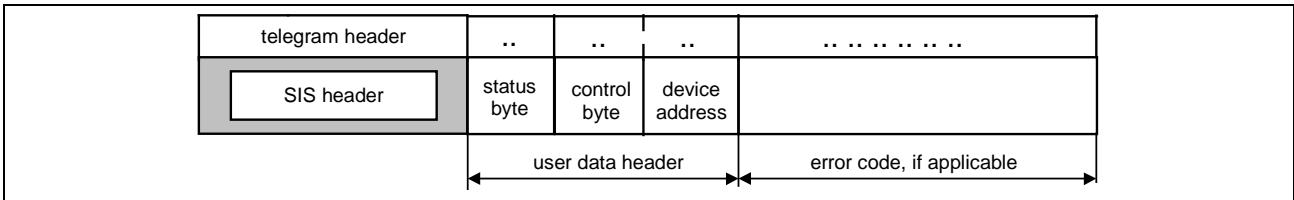


Fig. 13-7: Structure of the Response Telegram: Write Access

Read Access The response telegram is made up of:

- Telegram header
- User data header (depends on the SIS service)
- User data or error code

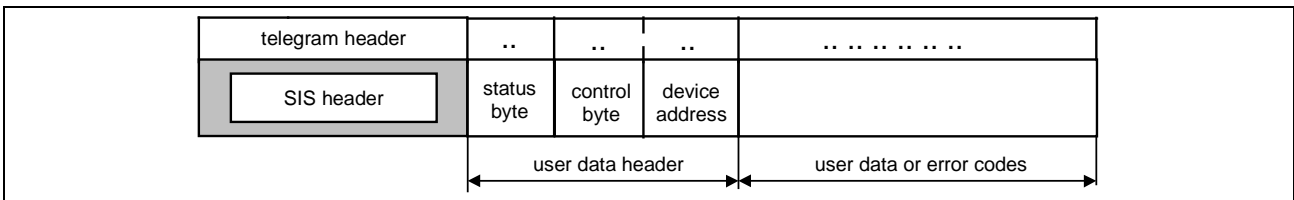


Fig. 13-8: Structure of the Response Telegram: Read Access

The response telegram from the slave has, apart from a few differences, the same telegram and the same user data header as the command telegram. This enables the sender of a command telegram to assign a unique response telegram.

Differences in the telegram header:

In the telegram header, the telegram must be marked as a response telegram. Refer to DOK-GENERL-SIS-DEFINIT-IF02-EN-P for details.

Differences in the user data header:

A status byte for displaying the transmission status is shown in the first byte of the user data header for general SIS services.

Note: The user data depends on the SIS service and the status byte.

Data Formats in the Telegrams

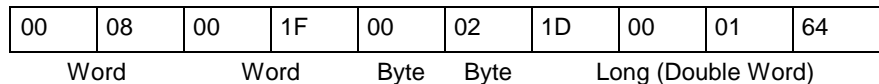
The SIS telegram is a **binary telegram** with standardized, binary telegram header and content for all general SIS services.

Note: The telegram content may also consist of an ASCII data set for the special SIS services in the individual product groups.

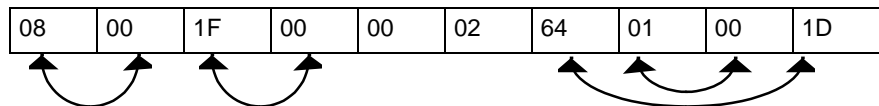
Intel Format The allocation of the individual bytes of data of the 'Word' or 'DWord' type corresponds to the **Intel** convention.

Example: In a defined data structure, the data words 0x0008 and 0x001F as well as the double word 0x1D000164 should be transmitted.

The logical order of the data:



The order of bytes sent in **Intel-Format**:



IEEE Format The **IEEE** format applies to the float display of data.

Telegram Header

The telegram header contains the following elements used for telegram control.

- Payload length
- Subnet routing
- Data integrity
- Service identification
- Source and destination addressing

Bits 0 - 3 in the telegram header control byte

In addition to general technical transmission control, it must also meet the following specific requirements:

- Addressing with up to four sub-addresses
- Forwarding of the telegram to the next recipient in the simplest possible way (the existence of devices that cannot process such telegrams but will have to forward them should be taken into consideration).
- Data in the telegram header that may have to be changed or evaluated when forwarding a telegram should be in a fixed position in the telegram header.

Note: For detail information on the Telegram Header, refer to the SIS Information manual, DOK-GENERL-SIS-DEFINIT-IF02-EN-P.

SIS Header Service Identification

Byte 6 of the SIS header is used for specifying a SIS service. GPP supports the following specific services:

- General services
- Common services
- GPP-Specific services

General SIS Services	Service ID	Function
	0x00	User Identification
	0x03	Initialize SIS communication

Table 13-25: General SIS Services

Common SIS Services	Service ID	Function
	0x10	Read parameter
	0x11	Read list parameter
	0x1E	Write list parameter
	0x1F	Write parameter

Table 13-26: Common SIS Services

GPP-Specific SIS Services	Service ID	Function
	0xC3	Read data access
	0xC4	Write data access

Table 13-27: GPP-Specific SIS Services

User Data Header

The content structure of the user data header depends on the specific common and GPP-Specific SIS services that is initiated in the SIS header. GPP supports the following:

- Common parameter command / response telegrams
- Common list segment command / response telegrams
- GPP-Specific command / response parameter telegrams

Parameter Command / Response Telegrams

For read or write parameters that are listed as common, the command and status telegram formats are described in the figure below. Read operations use the "0x10" SIS service, whereas write operations use the "0x1F" SIS service.

Read / Write Parameter Command Telegram

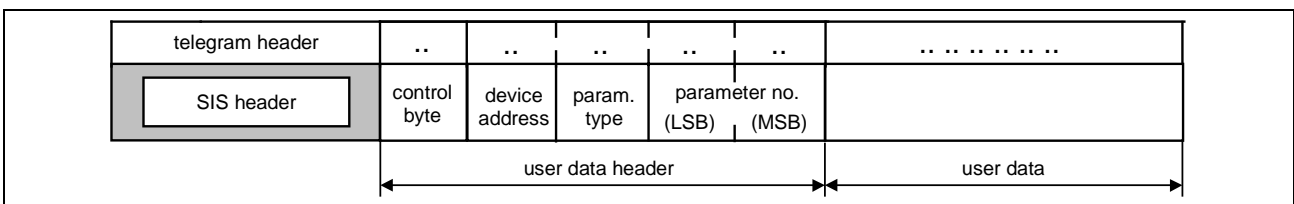


Fig. 13-9: Read / Write Parameter Command Telegram

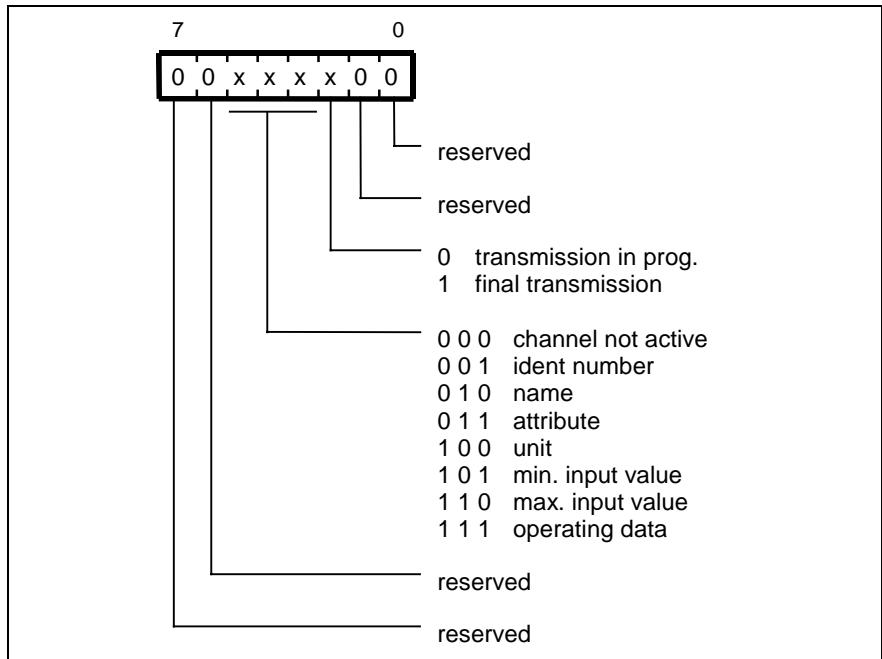


Fig. 13-2: Read / Write Parameter Command / Response Control Byte

Device Address

The unit address of a drive is read in the command telegram and copied into the response telegram.

The SIS protocol permits:

- direct SIS communication with drives supporting SIS interface. In this case the unit address is the same as the SIS address of the receiver.
- accessing drive parameters via a motion control, in case of drives not supporting SIS interface. The SIS address is related to the motion control and the unit address to the drive.

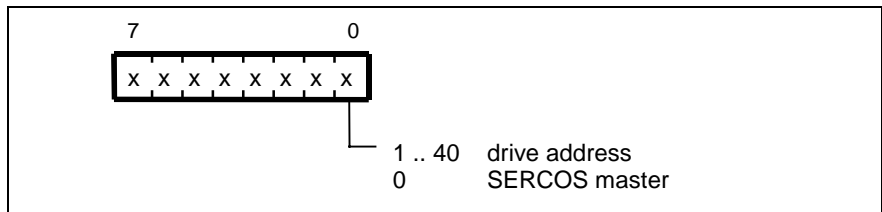


Fig. 13-3: Device Address

Parameter Type and Number

The parameter number has the form as defined in the SERCOS interface specification. To be able to also address control parameters, one byte is set ahead of the address to identify the parameter type.

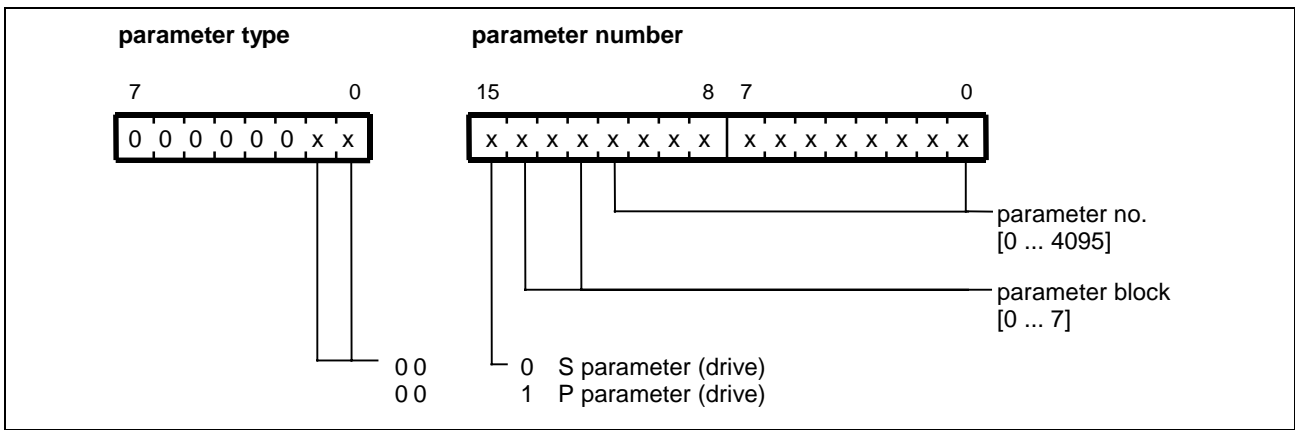


Fig. 13-4: Parameter Identification

Status Byte Error Codes

The status byte of the user data header provides the result of the transmission in the form of an error code. Generally, status codes can be separated into the following three types.

Error type	Error code
Transmission error-free	0x00
Protocol error	0xF0 ... 0xFF
Execution error (see below)	0x01 ... 0xEF

Fig. 13-13: Error types

The protocol error codes are defined as follows:

Code No.	Description	Error Type
0xF0	"Invalid service": The requested service is not specified or is not supported by the addressed user.	Telegram error
0xF1	"Invalid telegram": The telegram cannot be evaluated because, for example, a slave received a response telegram from the master or the start character was not found.	Telegram error
0xF2	"Telegram length error": The two length entries in the telegram do not match.	Telegram error
0xF4	"Checksum error": The transmitted checksum does not match the one calculated internally.	Telegram error
0xF8	"Invalid sequential telegram": Data in the user data header, the sender address or the service has changed in the sequential telegram.	Telegram error

Fig. 13-14: Response Telegram Status Byte Protocol Errors

Execution error codes are defined as follows:

Execution error	Code No.	Description
"Error during parameter transmission"	0x01	An error occurred while reading or writing a parameter (see below "Error codes in SIS services 0x10 – 0x1F")
"Error during phase switching"	0x02	The specified target phase was not achieved (see below " Error codes in SIS

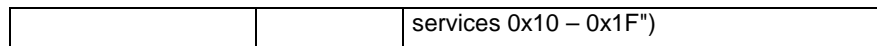


Fig. 13-15: Execution Error Codes

GPP-specific Command / Response Parameter Telegrams

The VisualMotion ACSII protocol is mapped to the SIS protocol and uses the command and response formats described in this section.

For read or write parameters that are listed as GPP-Specific, the command and status telegrams are described in the figure below. Read operations use the "0xC3" SIS service, whereas write operations use the "0xC4" SIS service.

Read / Write Parameter Command Telegram

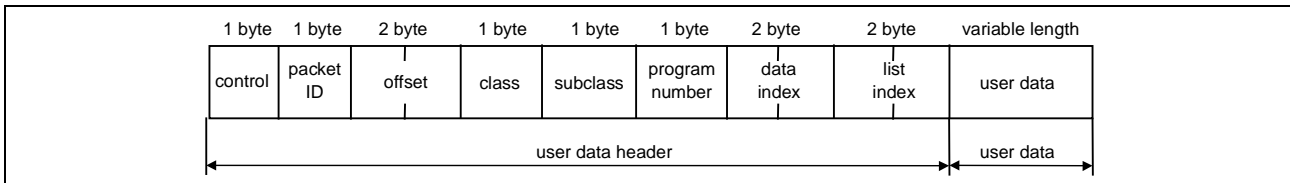


Fig. 13-5: GPP-specific Read / Write Parameter Command Telegram

User Data Header Element	Description
Control	Contains message fragment information and is used in conjunction with the "Packet ID" byte (see below). Bit 2 indicates whether or not the message is fragmented. When fragmented, bit 2 is set to 0 and the "Packet ID" indicates the fragment number in ascending order (e.g. 1—N). This allows GPP to reassemble the message if frames are received out of order, and when there are differing MTU sizes. The last fragment has bit 2 set to 1. When there are no fragments, bit 2 is set to 1 and the "Packet ID" is 1. Refer to section "GPP-Specific Read / Write Parameter Command / Response Control Byte" for bit definitions.
Packet ID	Number indicating the message fragment number (see control byte description)
Offset	Starting location for the return data
Class	Data type such as a register, user program, cam, or variable. Refer to tables 11-42 to 11-27.
Subclass	Requested format, element, or operation. Refer to tables 11-42 to 11-27.
Program Number	Indicates user-program, drive, task, or axis number (binary field). <ol style="list-style-type: none"> 1. A user program is identified by 0--10, where "0" indicates the active program. 2. A drive parameter number is selected by its SERCOS number (1--40). 3. An axis parameter number corresponds to its drive number (1--40). 4. A task parameter is identified as 1 for Task A, 2 for Task B, 3 for Task C, and 4 for Task D.
Data Index	Indicates a variable index, parameter, block, or step number (binary field). <ol style="list-style-type: none"> 1. Indicates variable number from 1 to the total number of variables available. A "0" requests the number of variables. 2. Indicates the parameter number. 3. Indicates the block number during a program transfer. 4. Indicates the step number for the sequencer.
List Index	Requests a specific item number in a list parameter, or the step number. A "0xFFFF" indicates this field is unused (binary field)
User Data	Contains the data read, written, or the operation status (ASCII field)

Table 13-28: GPP-specific Read / Write Parameter Command Telegram Elements

Read / Write Parameter Response Telegram

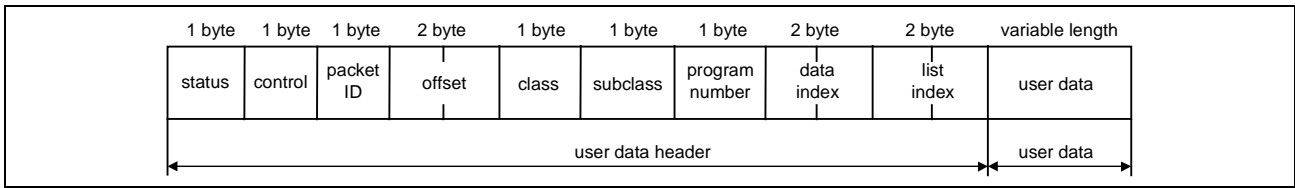


Fig. 13-6: GPP-specific Read / Write Parameter Response Telegram

User Data Header Element	Description
Status	Result of a transmission in the form of a code number.
Control	Same description as command telegram
Packet ID	Same description as command telegram
Offset	Starting location for the return data. Modified by GPP if the return data is larger than the maximum MTU size. The client then resends the command with the modified offset to collect the next fragment.
Class	Same description as command telegram
Subclass	Same description as command telegram
Program Number	Same description as command telegram
Data Index	Same description as command telegram
List Index	Same description as command telegram
User Data	Contains the data read, written, or the operation status (ASCII field).

Table 13-29: GPP-specific Read / Write Parameter Response Telegram Elements

GPP Class and Subclass Types

The following tables contain the SIS protocol class and subclass for accessing VisualMotion GPP data types.

Data Type	Class	Subclass	List	ASCII Format
Registers	0x52	0x42	N	Binary data
		0x44	N	Decimal data
		0x58	N	Hex data
		0x4D (Write Only)	N	Hex set bits
		0x46	N	Binary Force Mask
		0x53	N	Binary Force State
		0x43	N	Hex Force Mask & State
		0x45 (Write Only)	N	Clear all forcing
Program Floats, Program Integers, Global Floats, Global Integers	0x46, 0x49, 0x48, 0x47	0x50	N	Decimal data
		0x58	N	Hex data
		0x4C(Read Only)	N	Label text
Points(ABS, REL)	0x58, 0x59	0x58	N	X coor., float data
		0x59	N	Y coor., float data
		0x5A	N	Z coor., float data
		0x42	N	Blend radius, float data
		0x53	N	% of max. speed, integer data
		0x41	N	% of max. accel., integer data
		0x44	N	% of max. decel., integer data
		0x4A	N	% jerk, integer data
		0x31	N	number of event enabled for this point, integer data
		0x32	N	number of event enabled for this point, integer data
		0x33	N	number of event enabled for this point, integer data
		0x34	N	number of event enabled for this point, integer data
		0x52	N	Roll / Rate, float data
		0x50	N	Pitch, float data
		0x59	N	Yaw, float data
0x45	N	Elbow state, integer data		
0x4C	N	Array format, string data		
0x54	N	Label text		

Table 13-30: GPP Data Types (Registers, Program Variables and Points)

Data Type	Class	Subclass	List	ASCII Format
Zones	0x5A	0x53(Read Only)	N	Status, integer data
		0x41	N	Point 1 X coord., float data
		0x42	N	Point 1 Y coord., float data
		0x43	N	Point 1 Z coord., float data
		0x44	N	Point 2 X coord., float data
		0x45	N	Point 2 Y coord., float data
		0x46	N	Point 2 Z coord., float data
		0x4C	N	Array format, string data
		0x54	N	Label text
Events	0x45	0x53(Read Only)	N	Status, integer data
		0x54	N	Type, integer data
		0x44	N	Direction, integer data
		0x41	N	Argument, float data
		0x46	N	Function, string data
		0x4D	N	Message, string data
		0x4C	N	Array format, string data
		0x4E	N	Label text
PID	0x4D	0x42(Read Only)	N	Variables Used, string data
		0x45(Read Only)	N	Calculated set point, float data
		0x46(Read Only)	N	Calculated feedback, float data
		0x47(Read Only)	N	Calculated output, float data
		0x4A	N	Loop time, integer data
		0x4C(Read Only)	Y	List of valid PID loop numbers, integer data
		0x52(Read Only)	N	Control register, integer data
		0x53(Read Only)	N	Status register, integer data
		0x54(Read Only)	N	Type, integer data
PLS	0x57	0x4F	N	Phase offset, float data
		0x52	N	Assigned output register, integer data
		0x4D	N	Assigned mask register, integer data
		0x54	N	Master type, integer data
		0x41	N	Master axis number, integer data
		0x48	Y	List of On, Off, and Lead time values, string
		0x45	Y	List of On values, float data
		0x46	Y	List of Off values, float data
		0x47	Y	List of Lead-time values, integer data

Table 13-31: GPP Data Types (Zones, Events, PID and PLS)

Data Type	Class	Subclass	List	ASCII Format
Program	0x50	0x57	N	Download program header, string
		0x44	Y	Transfer program data, string
		0x52(Read Only)	N	Upload program header, string
		0x45	N	Delete a program, integer data
		0x48(Read Only)	Y	List of programs, string
		0x41	N	Activate a program, integer data
		0x4E(Read Only)	N	Program name, string
		0x46(Read Only)	Y	List of program event functions, string
		0x56(Read Only)	Y	List of program variables, string
		0x59	N	Copy program data to another program, integer data
		0x49	N	Change Indicator
		0x4A(Read Only)	Y	List of CAM INDEXER control blocks, string
		0x4D	Y	List of REGISTRATION control blocks, string
		0x54	N	Transfer of data between program
		0x4B	N	Compress Flash, integer data
ELS	0x4B	0x47(Read Only)	N	ELS GROUP control blocks, string
		0x4C(Read Only)	Y	List of drives assigned to a ELS GROUP
		0x4D(Read Only)	N	ELS MASTER control blocks, string
		0x56(Read Only)	N	VIRTUAL MASTER control blocks, string
Parameters (A, C, D, T)	0x41, 0x43, 0x44, 0x54	0x41(Read Only)	N	Attributes, hex
		0x42	Y	List of data in block mode, mixed
		0x44	Y	List of data one at a time, mixed
		0x48(Read Only)	N	Upper limit, mixed
		0x4C(Read Only)	N	Lower limit, mixed
		0x50	N	Data, mixed
		0x54(Read Only)	N	Parameter name, string
		0x55(Read Only)	N	Unit text, string

Table 13-32: GPP Data Types (Programs, ELS and Parameters)

Data Type	Class	Subclass	List	ASCII Format
Functions	0x53	0x41	N	Access type, integer data
		0x48	Y	List of Argument upper limits, mixed
		0x4C	Y	List of Argument lower limits, mixed
		0x4E	Y	List of Argument names, string
		0x52	Y	List of argument arrays, name, type, min, max, string
		0x54	N	Function name, string
		0x56	Y	List of Local Variables arrays, string
		0x59	Y	List of Argument types, char
Sequencer	0x4C	0x54	N	Sequencer name, string
		0x50	Y	List of Sequence table numbers, integer data
Sequencer Table	0x51	0x54	N	Table name, string
		0x50	Y	List of table data, string
		0x46	Y	List of functions in table, integer data
		0x31	Y	List of argument 1, mixed
		0x32	Y	List of argument 2, mixed
		0x33	Y	List of argument 3, mixed
		0x34	Y	List of argument 4, mixed
		0x35	Y	List of argument 5, mixed
Jogging	0x4A	0x41	N	Set register bit and start timer1
		0x42	N	Set register bit and start timer2
		0x43	N	Restore register bit and stop timers

Table 13-33: GPP Data Types (Functions, Sequencer and Jogging)

14 VisualMotion BTC06 Teach Pendant Interface

14.1 Overview

Bosch Rexroth's BTC06 Human Machine Interface unit is used to interface with the control, providing the operator with a variety of functionality. The operator can view and modify parameters, jog axes, and interface with machine operations using terminal emulation software. Using ScreenManager software, a machine builder can create customized screens that are specific to an application.

VisualMotion 9 also supports the BTV04, 05 and 06 HMI units using Screen Manager software. For information regarding the setup and use of ScreenManager, refer to the following manuals:

- ScreenManager V01 Functional Description
 - DOK-SUPPL*-SCM*PROG***-FK02-EN-P
- ScreenManager V01 Application Manual
 - DOK-SUPPL*-SCM*BEDIEN*-AW02-EN-P

Note: This chapter will focus on the BTC06 using VT100 Terminal emulation software (SWA-BTC06*-VT-01VRS-MS-C1.44). Refer to the *VisualMotion 9 Project Planning Manual* for details regarding hardware setup.

14.2 BTC06 Teach Pendant Screens

The BTC06 is a hand-held instrument with 16 x 40 character display and a 48-key sealed membrane keypad. The pendant provides a convenient operation and position programming interface for Bosch Rexroth's VisualMotion Control. The BTC06 Teach Pendant gives users a hand held operating interface which allows them to:

- Select operating modes and axis jogging
- Access multi-level menus for functions
- Teach and edit motion control points, events and variables; edit parameters
- Select and activate VisualMotion resident programs

Each category of functions has its own set of menus. The following function categories are available through the pendant BTV06 Main Menu:

```
GPP MAIN MENU
  A
PSM01*-GPP-08V09-M

F1 Program Menu
F2 Table Edit Menu
F3 Jog Menu
F4 Control Menu
F5 Register I/O Menu
F6 Parameter Menu
F7 Security Menu
F8 Diagnostic Menu
```

Menu Map

The following chart maps the submenus and menu links that are found within the main menu. Some menus have direct links to diagnostics, parameters and I/O registers.

Note: Pressing the ESC key will backtrack the Teach Pendant and display the previously viewed screen until it reaches the main menu.

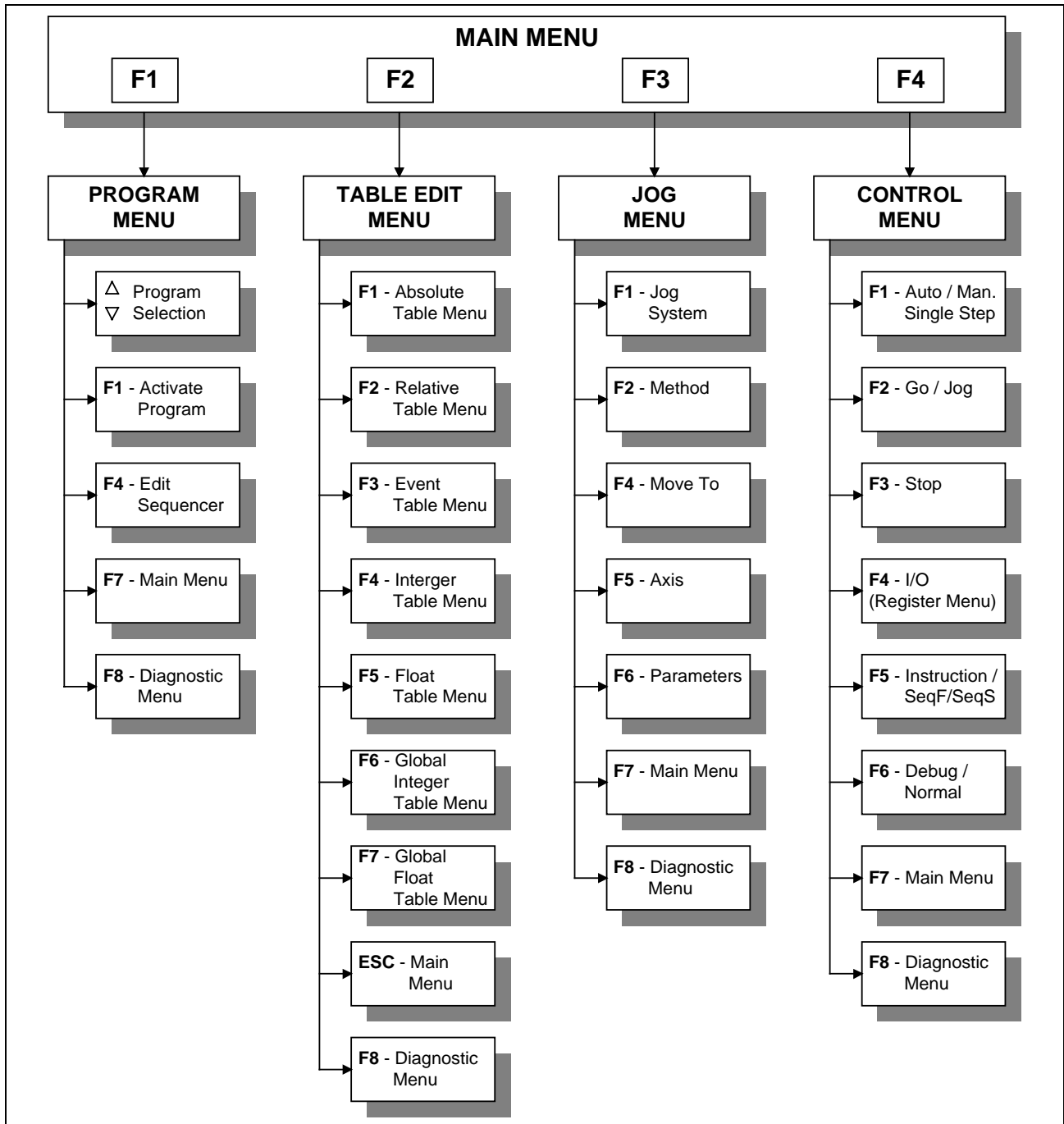


Figure 14-1: Menu Map (F1-F4)

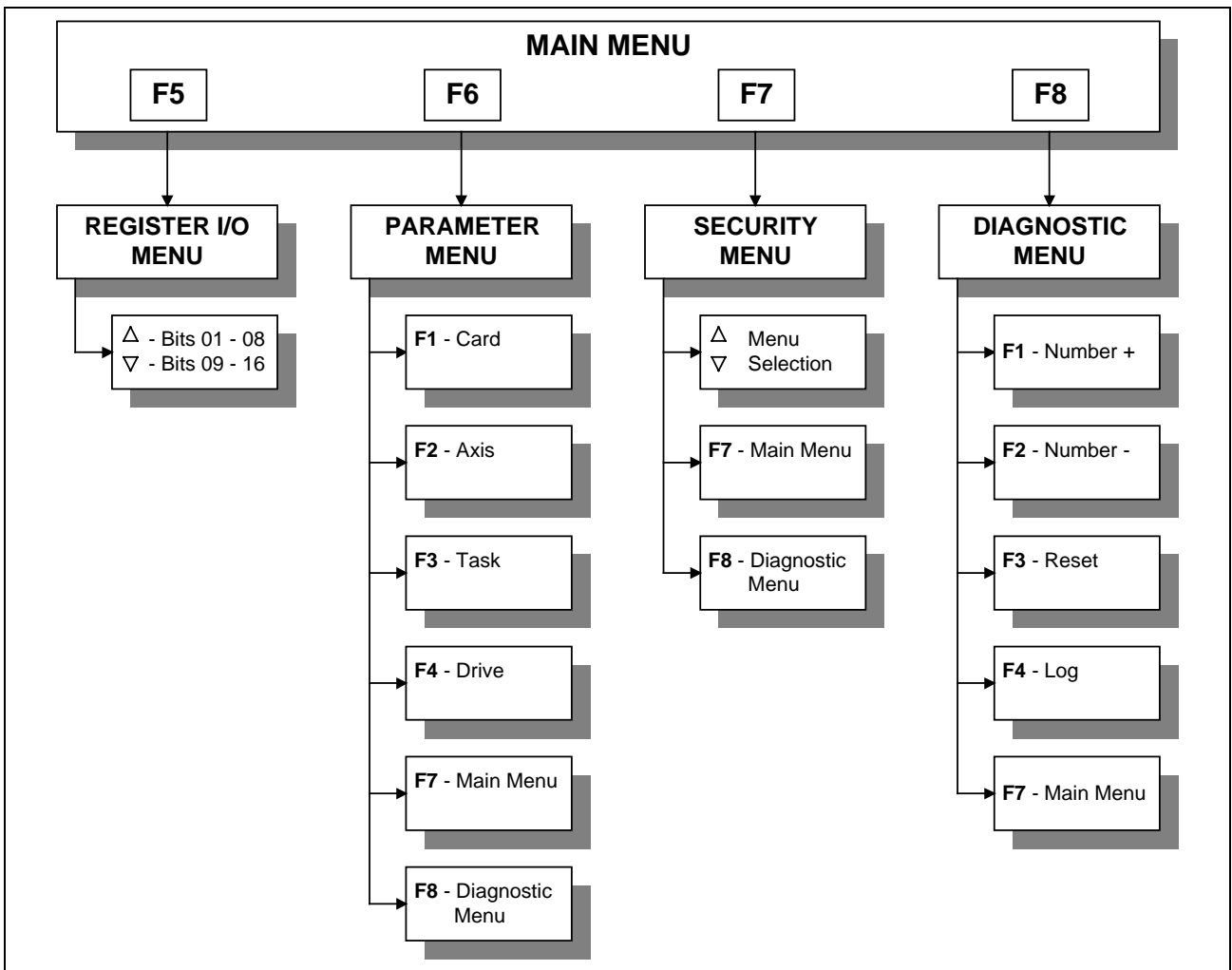


Figure 14-2: Menu Map (F5-F8)

14.3 BTC06 Teach Pendant Setup

When the Teach Pendant is enabled, the following registers and bits are forced at all times by the BTC06. VisualMotion Toolkit provides a register forcing capability that allows a Host system to directly change the state of individual I/O register bits overriding both the physical I/O and the I/O Mapper in VisualMotion.

Note: To use the pendant function keys, as well as the pendant edit features, the System Control Register 1, bit 14 (Pendant Enable) must be set to 1

Task A-D, Control Register 2-5:	bit 1	Mode Auto nManual
	bit 4	Single Step
	bit 6	Cycle Start/Resume
	bit 7	nTask Stop
	bit 12	Step Sequence Step
	bit 13	Step Sequence Function

Registers 98 and 99 define blocking bits for task A, B, C and D. The bits in the register can disable Teach Pendant control of the selected function for the corresponding tasks A-B, C-D. The following functions can be blocked:

Reg. - Bit	Description	Reg. - Bit	Description
98-1	Block Task A Manual	99-1	Block Task C Manual
98-2	Block Task A Auto	99-2	Block Task C Auto
98-3	Block Task A Step	99-3	Block Task C Step
98-4	Block Task A Jog	99-4	Block Task C Jog
98-5	Block Task A Entry	99-5	Block Task C Entry
98-6	Block Task A Teach	99-6	Block Task C Teach
98-9	Block Task B Manual	99-9	Block Task D Manual
98-10	Block Task B Auto	99-10	Block Task D Auto
98-11	Block Task B Step	99-11	Block Task D Step
98-12	Block Task B Jog	99-12	Block Task D Jog
98-13	Block Task B Entry	99-13	Block Task D Entry
98-14	Block Task B Teach	99-14	Block Task D Teach

If a block bit is set, its corresponding function is blocked. If a user selects the function, an error message is issued by the BTC06.
































The BTC06 parameters are automatically preset to the following specifications:
















Menu	Item	Default Setting
Serial Communication	Baud Rate	9600 (Fixed)
	Parity	None
	Data and Stop Bits	8, 1
	Display Serial Errors	Yes
	Audible Serial Errors	Yes
	Support for XON/OFF	Yes
Display	Display CTL Characters	No
	Display ESC Characters	No
	Cursor Visible	Yes
	Auto Line Wrap	No
	New Line on CR	Yes
	Display Self-Test	No
	Backlit Level	7
	Backlit On	Yes
Keyboard	Local Echo	No
	Key Repeat	Off
	Audible Keys	No
	Simplified KB	Yes

Note: When the Teach Pendant is initializing, it automatically sets the baud rate to 9600.

14.4 BTC06 Keyboard Operation

The following defines the keys for the Teach Pendant:

Key	Action
	Soft key defined by active menu
	Soft key defined by active menu
	Soft key defined by active menu
	Soft key defined by active menu
	Soft key defined by active menu
	Soft key defined by active menu
	Soft key defined by active menu
	Soft key defined by active menu
 	left and right refresh of the BTC06 screen
	First press function key help, then press item help (only available for Parameter Menu)
	Display all main menu functions
 	Jog A coordinate plus/minus
 	Jog B coordinate plus/minus Main Menu: respectively turns backlighting on and off
 	Jog C coordinate plus/minus
 	Jog X coordinate plus/minus
 	Jog Y coordinate plus/minus
 	Jog Z coordinate plus/minus
	numeric key
	numeric key and letter combination (use the shift key to access letters)
	numeric key and letter combination
	numeric key and letter combination
	numeric key and letter combination
	numeric key and letter combination
	numeric key and letter combination

Key	Action
	numeric key and letter combination
	numeric key and letter combination
	numeric key and letter combination
	decimal point
	Teach current position to absolute point table
	Select a user task
	Clear field of current item and allow editing
	page up / page down
	up and down arrows
	delete and left arrow (use the shift key to access delete)
	next and right arrow (use the shift key to access next)
	plus and minus (use the shift key to access plus)
	Terminate current operation or return to previous menu
	Confirm entry
	Shift key

Keyboard Map

The BTC06 keyboard is mapped to register 95, 96 and 97. The figure below and to the right outlines the register and bit location in the following format:

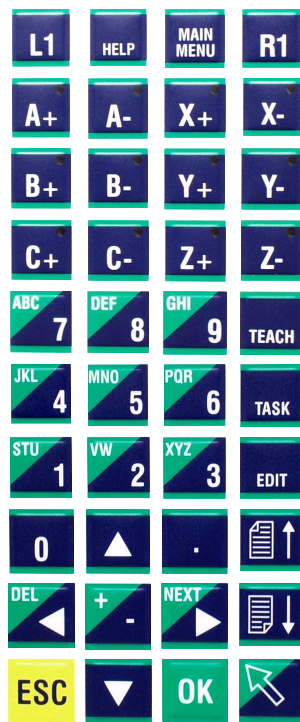
Register - Bit

Example: **95 - 01** ; key is mapped to register 95, bit 01

When a key is pressed its corresponding bit turns on and remains on for as long as the key is pressed.



95-01	95-02	95-03	95-04	95-05	95-06	95-07	95-08
-------	-------	-------	-------	-------	-------	-------	-------



97-13	95-09	97-14	97-15
96-05	96-06	96-07	96-08
96-13	96-14	96-15	96-16
97-05	97-06	97-07	97-08
95-10	95-11	95-12	95-13
96-02	96-03	96-04	95-14
96-10	96-11	96-12	96-09
97-03	95-11	97-02	95-15
96-02	97-04	96-04	95-16
96-01	96-11	97-01	

This shift key is not mapped to a register.

Cursor Control and Editing

The cursor may be moved up or down, left or right by pressing the corresponding arrow key. The left and right arrow keys double as delete and next, respectively. To edit an item, position the cursor over it and press the EDIT key. Doing so clears the field used by the item allowing a new value to be entered. Pressing OK terminates the editor and enters the new value into the system. Sometimes the cursor can be positioned on an item but the EDIT key does nothing when pressed. In this case the item cannot be edited. The cursor may be positioned there for another reason, such as item selection or viewing.

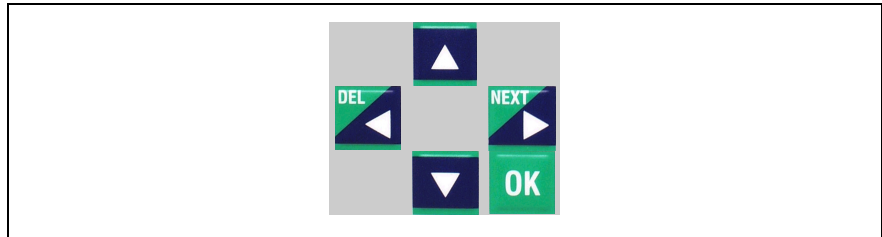



Fig. 14-1: Cursor Control and Editing

Number or Letter Selection

The number keys labeled 1 - 9 also double as letter keys when the shift key  key is pressed. To select the second or third letter contained in the upper left position of the key, the shift key must be pressed and held. If the shift key is pressed and not held, only the first letter will be selected and the key will then default to the number specified.

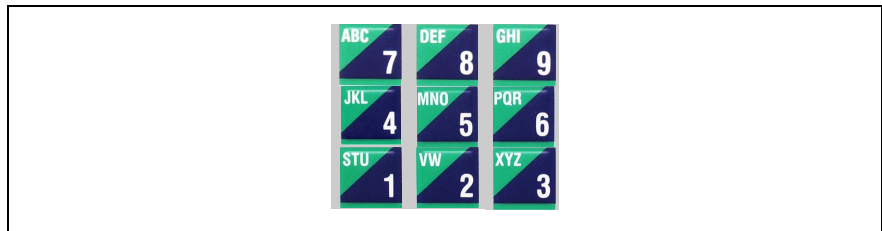


Fig. 14-2: Using the Shift key

Jogging Control

Press the *coordinated jog keys* (X+, X-, Y+, Y-, Z+, Z-) to jog in World, Joint or Tool coordinates. For robotics, the (A+, A-, B+, B-, C+, C-) keys function as Row, Pitch and Yaw in coordinated motion. The jog keys are active only while in the Jog Menu (**F3**). If other coordinated axes are defined in other tasks, then that task must be activated in order to jog from the Teach Pendant. When in single axis mode within the Jog Menu, the (A+, A-) keys light up and are used to jog the axis.

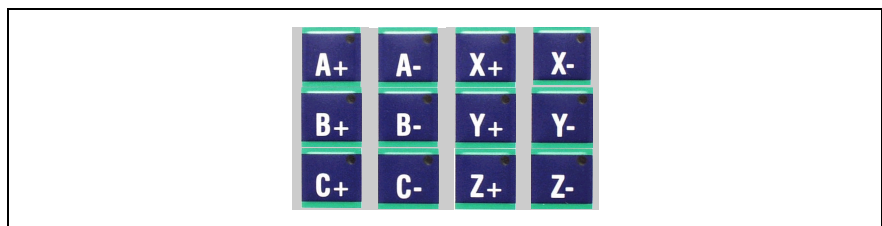



Fig. 14-3: Jogging Control

Task Control

Press the TASK  key to display the task menu. Use the arrow keys to position the cursor in the desired task and press OK, then press ESC to return to the previous menu.

Teach Control


The Teach  key allows the user to store the current position (during a coordinated jog) into the Absolute Point Table. The table point number will flash indicating that point has been recorded in the table.



Fig. 14-4: Teach Control

14.5 F1 Program Menu

The Program menu allows pendant selection and activation of any of the programs that have been downloaded to the control.

Each program consists of one to four user tasks (A, B, C, and D), and the associated Absolute and Relative Point Tables, Event Table, and Variable Tables. Activating a new program replaces the current four motion tasks and tables with the tasks and tables for the new program selection.

The Program menu consists of downloaded user programs with the following information:

- Program number (1-10)
- Program name
- Program date
- Time
- Program size

GPP PROGRAM MENU							
01	SEQ	08/24/00	15:39:27	1572			
02	SEQ1	08/25/00	10:20:15	3452			
03	AB1	08/29/00	16:20:00	1152			
04	AB2	08/29/00	16:20:00	3344			
05	Sequencer	08/30/00	07:15:00	2888			
00							
00							
00							
00							
00							
F1	F2	F3	F4	F5	F6	F7	F8
Actv			Edit			Main	Diag

The up and down arrow keys move the cursor to select a program. Pressing **F1** activates the selected program.

Note: The currently active program must not be running when activating another program.

Sequencer Editing (F4)

The **F4** key (Edit) only applies to programs which contain Sequencers. Pressing **F4** allows the user to edit the Sequencer list, steps and functions of the selected program.

Sequence List Menu

The first screen that appears after pressing **F4** in the *GPP Program Menu* is the *Sequence List Menu*. Use the arrow keys to navigate with the cursor to select the desired *Sequence List*. Press **F4** again to edit the contents of the selected list name within the *Sequence Edit Menu*.

```

SEQUENCE LIST MENU

Program: Sequencer

01 PICK_AND_PLACE
00
00
00
00
00
00
00
00

F1   F2       F4       F7   F8
PgUp PgDn     Edit     Main Diag

```

The name of each list can also be edited. Position the cursor at the end of the list name and press the **Edit** key. The letters of the alphabet are located within the numbered keys. These letters can only be accessed when used in conjunction with the **Shift** key. Use the **F1** key to delete characters to the left of the cursor. Use the **Shift** key to Select **Shift On** and **Off**. This allows you to toggle the keyboard map between numbers and letters. Refer to Number or Letter Selection on page 14-9 for details.

This editing process is functional within all of the following Sequencer menus.

The Sequence Edit Menu

The *Sequence Edit Menu* displays all of the steps within the selected Sequence. Use the arrow keys to navigate the cursor to the desired Sequence Step. Press **F4** again to edit the contents of that Step within the *Step Table Edit Menu*. Press **F3** to cut the selected Sequence Step. Press **F6** to paste a Sequence Step in the current cursor position.

```

SEQUENCE EDIT MENU

Program: Sequencer
Sequence: PICK_AND_PLACE

01 Home_All_Axes
02 Set_Max_Values
03 Pick_Position
00
00
00
00

F1   F2   F3   F4   F5   F6   F7   F8
PgUp PgDn Cut Edit Ins Past Main Diag

```

Table List Menu

Pressing **F5** (Ins) from the *Sequence Edit Menu* will open the *Step List Menu*. This menu contains a list of all the step tables available within the selected Sequence. Use the arrow keys to navigate the cursor to the

desired *Step Table*. Press OK to insert that function into the previous *Sequence*.

```

STEP LIST

Press OK to insert

01 Pick_Position
02 Set_Max_Value
03 Home_All_Axes
00
00
00
00
00

F1    F2
PgUp PgDn

```

The Step Table Edit Menu

The *Step Table Edit Menu* displays all the functions within the selected *Sequence Step*. Use the arrow keys to navigate the cursor to the desired *Sequence Function*. Press **F4** again to edit the contents of that function within the *Function Edit Menu*. Press **F3** to cut the selected function. Press **F6** to paste a function in the current cursor position

```

STEP TABLE EDIT MENU

Program:      Sequencer
Sequence:     Initialize Sequencer
Step:         Home_All_Axes
Empty Slots: 00011

01 Disable_Clamp_Motion
02 Chk_Mold_Open
03 Permit_Eject_Back
04 Chk_Ejectors_Back
05 Home_Axes

F1  F2  F3  F4  F5  F6  F7  F8
PgUp PgDn Cut Edit Ins Past Main Diag

```

Function List

Pressing **F5** (Ins) from the *Step Table Edit Menu* will open the *Function List Menu*. This menu contains a list of all the functions available within the selected *Sequence*. Use the arrow keys to navigate the cursor to the desired *Function*. Press OK to insert that function into the previous *Step Table*.

```

FUNCTION LIST

Press OK to insert

01 HOME_AXIS
02 INIT_POS_VELOCITY
03 CHK_EJECTORS_RETRACT
04 CLEAR_SYSTEM_TIMERS
05 DWELL
06 PERMIT_EJECT_BACK
00
00

F1    F2
PgUp PgDn

```

Function Edit Menu

The *Function Edit Menu* contains a list of all the arguments and their corresponding values. Use the arrow keys to navigate with the cursor to the desired *Function*. Press **F4** again to edit the values assigned to the arguments of that function.

```

FUNCTION EDIT MENU

Program: Sequencer
Sequence: Initialize Sequencer
Step: Home_All_Axes
Function: Home_Axes

01 AXIS_NUMBER                2
02 HOME_OFFSET_POSITION      0.0000
03 SET_HOME_POSITION        0.0000
00
00

F1          F4          F7  F8
Save       List-Edit  Main Diag

```

14.6 F2 Table Edit Menu

The Table Edit menu allows editing of the Absolute and Relative Point Tables, the Event Table, and the Integer and Float variable Tables.

```

GPP TABLE EDIT MENU

F1 Absolute Table Menu
F2 Relative Table Menu
F3 Event Table Menu
F4 Integer Table Menu
F5 Float Table Menu
F6 Global Integer Table Menu
F7 Global Float Table Menu

ESC  F8
Main Diag

```

Absolute Table Menu (F1)

The Absolute Point Table Edit menus permit editing taught or programmed points.

```

ABSOLUTE TABLE

NUM NAME
001 Part_Pickup
002 Regrip
003 Leave_Part
004 ABS[4]
005 ABS[5]
006 ABS[6]
007 ABS[7]
008 ABS[8]
009 ABS[9]
010 ABS[10]

F1  F2  F3  F4  F5  F6  F7  F8
PgUp PgDn Home End Edit Jog Main Diag

```

Select a point by moving the cursor up and down with the arrow keys. Pressing F5 (Edit) will bring up the following menu:

```

ABSOLUTE POINT EDIT
ABS[0001]  ABS[1]
          2.000 X          3.000 Roll
          2.000 Y          1.000 Pitch
          3.000 Z          0.500 Yaw
          0.000 Blend          01 Elbow
          10 Speed          0 Event 1
          5 Accel          0 Event 2
          5 Decel          0 Event 3
          2 Jerk          0 Event 4
F1  F2  F3  F4          F6  F7  F8
Inpt DcPt Home End          Jog  Main Diag
    
```

X X coordinate of the point
 Y Y coordinate of the point
 Z Z coordinate of the point
 Blend Blend Radius

Roll Roll angle
 Pitch Pitch angle
 Yaw Yaw angle
 Elbow Elbow state

Speed Speed Percentage (of task maximum)
 Accel Acceleration Percentage (of task maximum)
 Decel Deceleration Percentage (of task maximum)
 Jerk Jerk Limiting Percentage
 (0 trapezoid, 100 s-shape, 50 between)

Event 1 First event for the point
 Event 2 Second event for the point
 Event 3 Third event for the point
 Event 4 Fourth event for the point

(This value represents an event number from the event table)
 Refer to **Event Table** on page 14-16

Relative Table Menu (F2)

The Relative Point Table Edit menus permit editing of points either taught or programmed.

```

RELATIVE TABLE
NUM NAME
001 REL[1]
002 REL[2]
003 REL[3]
004 REL[4]
005 REL[5]
006 REL[6]
007 REL[7]
008 REL[8]
009 REL[9]
010 REL[10]
F1  F2  F3  F4  F5  F6  F7  F8
PgUp PgDn Home End Edit Jog Main Diag
    
```

Select a point by moving the cursor up and down with the arrow keys. Pressing F5 (Edit) will bring up the following menu:

RELATIVE POINT EDIT						
REL[0001]		REL[1]				
	2.000	Drive_1		3.000	Drive_4	
	2.000	Drive_2		1.000	Drive_5	
	3.000	Drive_3		0.500	Drive_6	
	0.000	Blend		01	Elbow	
	10	Speed		0	Event 1	
	5	Accel		0	Event 2	
	5	Decel		0	Event 3	
	2	Jerk		0	Event 4	
F1	F2	F3	F4	F6	F7	F8
Inpt	DcPt	Home	End	Jog	Main	Diag

X X coordinate of the point
 Y Y coordinate of the point
 Z Z coordinate of the point
 Blend Blend Radius

Roll Roll angle
 Pitch Pitch angle
 Yaw Yaw angle
 Elbow Elbow state

Speed Speed Percentage (of task maximum)
 Accel Acceleration Percentage (of task maximum)
 Decel Deceleration Percentage (of task maximum)
 Jerk Jerk Limiting Percentage
 (0 trapezoid, 100 s-shape, 50 between)

Event 1 First event for the point
 Event 2 Second event for the point
 Event 3 Third event for the point
 Event 4 Fourth event for the point

(This value represents an event number from the event table)

Refer to **Event Table** on page 14-16

Event Table Menu (F3)

The Event Table Edit menu allows pendant editing of the events associated with each task in the Event Table.

The currently selected task determines the portion of the event table allowed to be viewed through the Teach Pendant.

EVENT TABLE						
NUM	ST	TY	RF	ARG	FUNCTION	
001	01	06	00	20.0	Pressure_Switch	
002	01	06	00	40.0	Change_Speed	
003	01	09	00	60.0	Evt_Fn_1	
004	00	00	00	0.0	NONE	
005	00	00	00	0.0	NONE	
006	00	00	00	0.0	NONE	
007	00	00	00	0.0	NONE	
008	00	00	00	0.0	NONE	
009	00	00	00	0.0	NONE	
010	00	00	00	0.0	NONE	
F1	F2	F3		F5	F7	F8
PgUp	PgDn	Home		Repl	Main	Diag

- St** **The Event's status:**
 0 = inactive
 1 = pending
 2 = queued
 3 = executing
 4 = done

- Ty** **Event type:**
 0 = event inactive
 1 = repeating timer
 2 = time in coordinated path
 3 = percent in coordinated path
 4 = single axis distance
 5 = repeating axis position (rotary)
 6 = task input transition
 9 = feedback capture
 10=I/O register event
 11=PPC-R X1 input events

- Rf** **Event Reference:**
 0 = start of segment
 1 = end of segment

- Arg** **Argument for the event**
 (milliseconds if time based, or percent of path and axis distance)

- Function** Task ID and Event number

Integer Table Menu (F4)

This menu allows for viewing and editing integers. Variables can be changed by any task at any time. It is possible, therefore, for editing to be in conflict with a motion task. In this instance, unexpected results may occur. It is at the discretion of the operator to determine the usefulness of such an operation.

INTEGER TABLE				
00001	Pointer_1	20.0		
00002	Pointer_2	40.0		
00003	Timer_1	60.0		
00004	Timer_2	80.0		
00005	Operation Type	00.0		
00006	I[6]	0		
00007	I[7]	0		
00008	I[8]	0		
00009	I[9]	0		
00010	I[10]	0		
F1	F2	F3	F7	F8
PgUp	PgDn	Fmt	Main	Diag

Display Format

Pressing **F3** toggles the display between decimal (20.0) and hexadecimal (0x00000014) notation.

Floating Table Menu (F5)

This menu allows for viewing and editing of float variables. Variables can be changed by any task at any time. Therefore, its possible, for editing to be in conflict with a motion task. In this instance, unexpected results may occur. It is at the discretion of the operator to determine the usefulness of such an operation.

FLOATING TABLE				
00001	F[1]			0.0000
00002	F[2]			0.0000
00003	F[3]			0.0000
00004	F[4]			0.0000
00005	F[5]			0.0000
00006	F[6]			0.0000
00007	F[7]			0.0000
00008	F[8]			0.0000
00009	F[9]			0.0000
00010	F[10]			0.0000
F1	F2	F3	F7	F8
PgUp	PgDn	Fmt	Main	Diag

Display Format

Pressing **F3** toggles the display between floating fixed (100.000) and scientific (1.000e+02) notation.

Global Integer Table Menu (F6)

This menu allows for viewing and editing of global integer variables. Variables can be changed by any task at any time. It is possible, therefore, for editing to be in conflict with a motion task. In this instance, unexpected results may occur. It is at the discretion of the operator to determine the usefulness of such an operation.

GLOBAL INTEGER TABLE				
00001	GI[1]			0
00002	GI[2]			0
00003	GI[3]			0
00004	GI[4]			0
00005	GI[5]			0
00006	GI[6]			0
00007	GI[7]			0
00008	GI[8]			0
00009	GI[9]			0
00010	GI[10]			0
F1	F2	F3	F7	F8
PgUp	PgDn	Fmt	Main	Diag

Display Format

Pressing **F3** toggles the display between decimal (20.0) and hexadecimal (0x00000014) notation.

Global Floating Table Menu (F7)

This menu allows for viewing and editing of global float variables. Variables can be changed by any task at any time. It is possible, therefore, for editing to be in conflict with a motion task. In this instance, unexpected results may occur. It is at the discretion of the operator to determine the usefulness of such an operation.

GLOBAL FLOATING TABLE				
00001	GF[1]			0.0000
00002	GF[2]			0.0000
00003	GF[3]			0.0000
00004	GF[4]			0.0000
00005	GF[5]			0.0000
00006	GF[6]			0.0000
00007	GF[7]			0.0000
00008	GF[8]			0.0000
00009	GF[9]			0.0000
00010	GF[10]			0.0000
F1	F2	F3	F7	F8
PgUp	PgDn	Fmt	Main	Diag

Display Format

Pressing **F3** toggles the display between float fixed (100.000) and scientific (1.000e+02) notation.

14.7 F3 Jog Menu

The Jog menu allows you to jog a stopped system. The following I/O register bits must be on before jogging an axis:

Register 1 - System Control

Bit 6 Pendant Live Man
 Bit 14 Pendant Enable

Register 2, 3, 4, or 5 -Task Control

Bit 1 Mode:! Manual

```

ROBOT JOG MENU
  A
System: World
Method: Continuous/Slow
  0001: ABS[1]
  AXIS      WORLD      TAUGHT
01 Drive_1  12.643     47.500     20.300
02 Drive_2  95.215     18.300     54.200
03 Drive_3  63.609     5.500     16.000
04 Drive_4  0.960      36.800     10.000

00 Single   857.628

F1  F2      F4  F5  F6  F7  F8
Syst Meth  MvTo Axis Para Main Diag
    
```

F1 = System F2 = Method F4 = Move To F5 = Axis
F6 = Parameters F7 = Main Screen F8 = Diagnostics

Press **F1** to select either the Axis, Joint or World jog system. **F2** selects the jog method which can be continuous or incremental.

F4 is a "Move To" function that allows the user to enter a position in the TAUGHT column and move the specified axis to that point by pressing OK. Use the up and down arrow keys to move the cursor to the desired TAUGHT axis. This function is only available for coordinated axes.

F5 selects a single axis to jog.

F6 opens the *Edit Jog Parameters* screen which allows the user to adjust the percent distance and speed parameters, as well as, view the values set for each Task and Axis.

Jog Systems

Axis Jog Menu

The **Single Axis Jog** menu allows jogging a single, non-coordinated axis. Only the selected axis is affected. The BTC06 display is continuously updated with the current position of the axis.

Press **A-** to jog in the negative direction.

Press **A+** to jog in the positive direction.

(The teach pendant beeps at the beginning and end of motion.)

Coordinated Jogging

Press **X-** to jog in the negative X direction.

Press **X+** to jog in the positive X direction.

Press **Y-** to jog in the negative Y direction.

Press **Y+** to jog in the positive Y direction.

Press **Z-** to jog in the negative Z direction.

Press **Z+** to jog in the positive Z direction.

Joint Jog Menu

The **Joint** Jog menu allows jogging of individual axes with a joint number.

Robot World Jog Menu

The Robot **World** Jog menu allows jogging a coordinated or single axis for a task in World Cartesian Space. When jogging in world coordinates, motion will be generated parallel to the selected X, Y, or Z coordinate.

The pendant beeps at the beginning and end of motion. The display is continuously updated to display the current position (X, Y, Z) on each of the axes.

Jog Method

The following Jog Methods are available with the Teach Pendant:

Continuous/Fast	Continues to jog quickly until the button is released
Continuous/Slow	Continues to jog slowly until the button is released
Incremental/Large	Jogs a predetermined large increment and then stops.
Incremental/Small	Jogs a predetermined small increment and then stops.

Teaching Points

To teach the current position (during a coordinated jog) into the Absolute Point Table press TEACH. (Confirm each point by pressing the OK key.)

The table point number will flash indicating that point has been recorded in the table. The point number will automatically advance to the next point.

Jog Fine Adjustments

The jog speed and distance increments are set as a percentage of the Maximum Jog Increment and Maximum Jog Velocity parameters (T-0-0025 and T-0-0026).

Separate percents are used for FAST/SLOW and LARGE/SMALL jog settings in coordinated jog.

While in the Axis Jog or World Jog Menus, pressing F6 (PAR key) displays a screen that permits editing the FAST/SLOW and LARGE/SMALL jog percents.

14.8 F4 Control Menu

The Control menu allows the pendant to control the execution of a task.

When the Teach Pendant powers up, the Control Menu is the first menu displayed.

The Control Menu will provide the following information:

Title	Control Menu Title
Task Status	Current task operating status
Program Name	Name of the currently active program
Sequence	Name of the current sequence executing
Step	Name of the current step executing
Function	Name of the current function executing
Position Title	Axis Position Title (Joint, World, & Target)
Axis 1 Label (X)	Axis defined as axis 1
Axis 2 Label (Y)	Axis defined as axis 2
Axis 3 Label (Z)	Axis defined as axis 3
Axis 4 Label (A)	Axis defined as axis 4
Target Name	Point # and label for the current point executing
Function Keys	Function keys control machine operation
Operation Labels	Specify the machine operations

The control menu can run in one of three different modes. The following pages describe the operation of each mode and illustrate the different menu layouts.

Control Menu: Auto Run/Hold Mode

Text appears when the **F6** key, **DBug** is pressed.

Text disappears when the **F6** key, **Norm** is pressed.

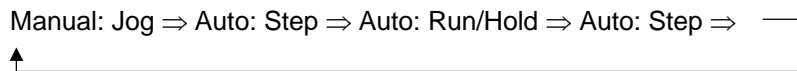
```

                CONTROL MENU
INST: 00FC AXIS_WAIT
STAT: Move to return position
DIAG: Task Running      No Target
AXIS      WORLD      TARGET      STATUS
          1200.00    1200.00    Task A
          00.00     500.00    AUTO
          90.00     90.00     RUN
                               99%

F1  F2  F3  F4  F5  F6  F7  F8
Auto Go  Stop I/O  Inst Norm Main Diag
    
```

F1 - Mode Of Operation

If the *Teach Pendant Enabled Bit (Register 1 bit 14)* is high, pressing the **F1** key will change the mode of operation in the order shown below:



F1 = Auto **F2** = Go **F3** = Stop **F4** = I/O
F6 = Debug/Norm **F7** = Main Menu **F8** = Diagnostics

Note: **F2,F3** and **F5** are dependent on the selected mode of operation.

When *Automatic Mode* is selected by pressing the **F1** key, **F2** will display Go and **F3** will display Stop. By pressing the **F2** key, the active program will start executing instructions. By pressing the **F3** key, program execution will stop. If the **F2** key is pressed again, the program will continue.

To restart at the beginning of the program, the mode of operation must be changed to manual or step and then changed back to auto.

By pressing the **F4** key, the Register Menu will be displayed, allowing the operator to active I/O bits.

In Auto: Step mode **F5** is used to select the step method which can be one instruction, one Sequence Step, or one Sequence Function at a time.

When Debug is selected by pressing the **F6** key, the following text appears in the top half of the screen and the **F6** key text changes to Norm. Pressing **F6** (Normal) again removes this information.

CONTROL MENU	Screen name
INST: 00FC AXIS_WAIT	VisualMotion instruction being processed
STAT: Task Running	Task status
DIAG: Task Running	Diagnostic status

Control Menu: Auto Step Mode

Text appears when the **F6** key, **DBug** is pressed.

Text disappears when the **F6** key, **Norm** is pressed.

CONTROL MENU			
<i>INST: 0194 SET</i>			
<i>STAT: In return position</i>			
<i>DIAG: Instruction Sin</i> No Target			
AXIS	WORLD	TARGET	STATUS
	1200.00	1200.00	Task A
	00.00	500.00	AUTO
	90.00	90.00	RUN
			99%
F1	F2	F3	F4
Step	Go	Stop	I/O
F5	F6	F7	F8
Inst	Norm	Main	Diag

- F1** = Step **F2** = Go **F3** = Stop **F4** = I/O
- F5** = Instruction/Sequence **F6** = DeBug/Norm
- F7** = Main Menu **F8** = Diagnostics

When the Automatic Step Mode is selected by pressing the **F1** key, **F2** will display GO and **F3** will display STOP. Every time the **F2-GO** key is pressed, the program will be sequentially executed one step at a time. The steps can be program instructions, Sequencer steps or Sequencer functions. Pressing the **F5** key selects the step mode that the program will follow:

Instruction	-	INST
Sequence/Steps	-	SEQ/STEP
Sequence/Function	-	SEQ/FUNC

When **F5** (INST) is selected the program will execute only one instruction every time the **F2-GO** key is pressed. When SEQ/STEP is selected the program will execute all the functions within one Sequencer step, one at a time. When SEQ/FUNC is selected the program will execute each Sequencer function, one at a time.

The **F3-STOP** key can be used to immediately halt the execution of the program within a step. If the **F2-GO** key is pressed again, the step will continue to run.

By pressing the **F4** key, the Register Menu will be displayed, allowing the operator to active I/O bits.

Control Menu: Manual Mode

Text appears when the **F6** key, **DBug** is pressed.

Text disappears when the **F6** key, **Norm** is pressed.

```

                                CONTROL MENU
INST: 0194 SET
STAT: In return position
DIAG: Manual mode                No Target
AXIS      WORLD      TARGET      STATUS
          1200.00    1200.00    Task A
          00.00     500.00    AUTO
          90.00     90.00     RUN
                                   99%

F1  F2  F3  F4  F5  F6  F7  F8
Manu Go Stop I/O Inst Norm Main Diag
    
```

F1 = Manual **F2** = Jog **F4** = I/O **F6** = DeBug/Norm
F7 = Main Menu **F8** = Diagnostics

By pressing the **F2** key, the Jog Menu will be displayed, allowing the operator to jog and teach each axis.

By pressing the **F4** key, the Register Menu will be displayed, allowing the operator to active I/O bits.

14.9 F5 Register I/O Menu

The **F4** I/O key is provided on every operator interface control screen. The operator will have the ability to view and edit the register bits that the machine builder selects. When the **F4** key is pressed, the register menu will be displayed. The first register displayed is set in control parameter C-0-0805 *Start of User Accessible Registers on Pendant*. Parameter C-0806 defines the *End of User Accessible Registers on Pendant*. The operator can only edit registers within the range of these two parameters.

For example, parameter C-0805 = register 100, which is labeled End_Of_Arm_Tool_1. When the **F4** key is pressed, the first screen displayed is register 100, along with the register label. Bits 1 through 8 are displayed, along with the bit labels and current state (ON/OFF). This screen is only updated when any of the bits change states.



```

                                REGISTER MENU
REGISTER: 0100 End_Of_Arm_Tool_1
BITS: Forward                01 OFF
      Reverse                 02 OFF
      C_Axis_Vertical         03 ON
      C_Axis_Horizontal       04 OFF
      A_Axis_Forward          05 OFF
      A_Axis_Retracted        06 OFF
      Bit_07                  07 OFF
      Bit_08                  08 OFF

      Down Arrow For Bits 09 - 16
F1  F2  F3  F4  F5  F6  F7  F8
01  02  03  04  05  06  07  08
    
```

The first page of the register menu will display bits 1 through 8. By pressing the down arrow key, bits 9 through 16 will be displayed. Pressing the up arrow key will return to bits 1 through 8.



The page up  and page down  keys move the cursor to the next or previous register available within the limits of card parameters C-0-0805 and C-0-0806.

To jump to a register number:

1. Press the edit key
2. Enter the register number
3. Press the OK key

```

REGISTER MENU
REGISTER: 0100 End_Of_Arm_Tool_1
BITS: Bit_09          01 OFF
      Bit_10          02 OFF
      Bit_11          03 ON
      Bit_12          04 OFF
      Bit_13          05 OFF
      Bit_14          06 OFF
      Bit_15          07 OFF
      Bit_16          08 OFF
      Down Arrow For Bits 01 - 08
F1  F2  F3  F4  F5  F6  F7  F8
01  02  03  04  05  06  07  08

```

The function keys **F1** through **F8** will allow the operator to toggle the state (ON/OFF) of bits 1 through 8 (first page displayed) or bits 9 through 16 (second page displayed).

Note: If an operator needs to change a bit in a register outside the range set by parameters C-0-0805 and C-0-0806, a password will have to be entered or the pendant level protection bits will have to be adjusted. See the Security Menu description.

14.10 F6 Parameter Menu

The Parameter menu allows selection of screens for editing the system, task, axis, and drive parameters.

```

PARAMETER MENU

F1 CARD
F2 AXIS
F3 TASK
F4 DRIVE

F7  F8
Main Diag

```

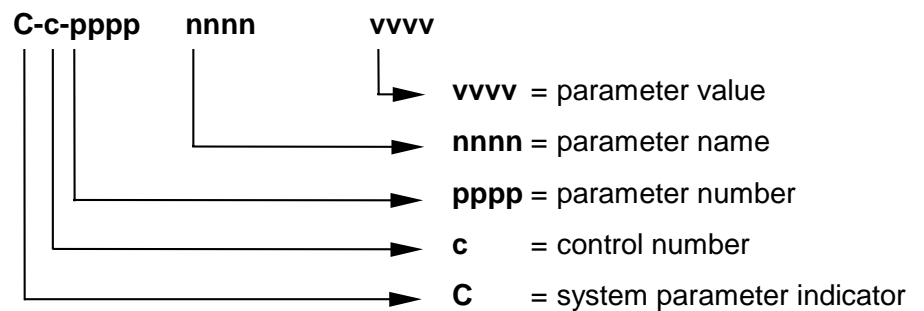
Selecting **F1-F4** will open one of the following Parameter screens.

F1 - Card Parameter Screen

```

CARD PARAMETER MENU
C-c-ppppp nnnnn vvvv
C-0-00001 Language Selection 1
...
...
...
...
...
...
...
C-0-00042 World Large Increment 50

F1 F2 F3 F4 F7 F8
Home End PgUp PgDn Main Diag
    
```



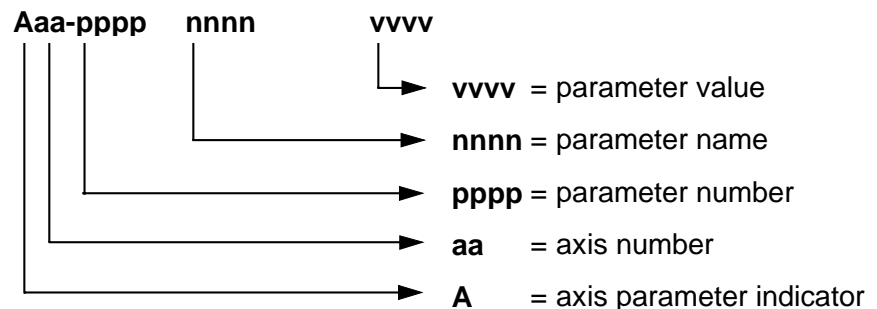
For changing of Card parameter numbers and editing of values, refer to **F2 - Axis Parameter Screen** on page 14-26.

F2 - Axis Parameter Screen

```

AXIS PARAMETER MENU
Aaa-ppppp nnnnn vvvv
A01-00001 Language Selection 1
...
...
...
...
...
...
...
A01-000021 Maximum Acceleration 200

F1 F2 F3 F4 F7 F8
Home End PgUp PgDn Main Diag
    
```



When the operator first enters this screen, the cursor will be flashing on the **aa** axis number. This can also be performed by pressing the **F1 Home** key.

To change the axis number:

1. Press the Edit key
2. Enter the new axis number
3. Press the OK key

When done, all of the axis parameter number will be modified to display the new axis number.

To move the cursor to the parameter number (**pppp**) while the cursor is on the axis number, press the right arrow key. Pressing the **F1 Home** key will return the cursor to the axis number.

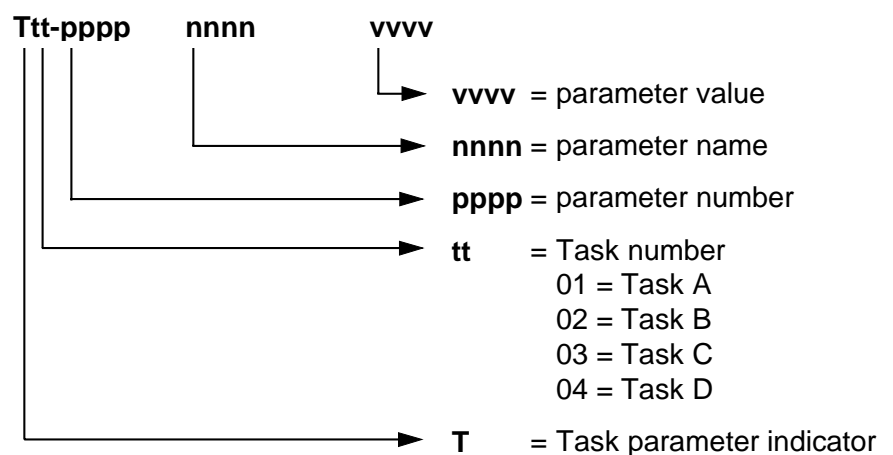
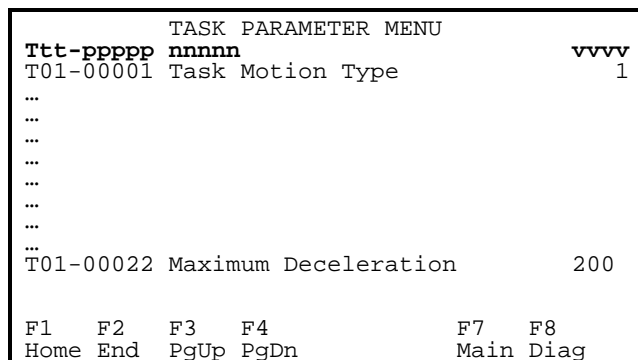
To jump to a given parameter number:

1. Press the Edit key
2. Enter the parameter number
3. Press the OK key

When done, the cursor will jump to the specified parameter number.

To modify the value for a given parameter, press the **F2 End** key to jump to the value field. Press the Edit key, enter the new value and press OK.

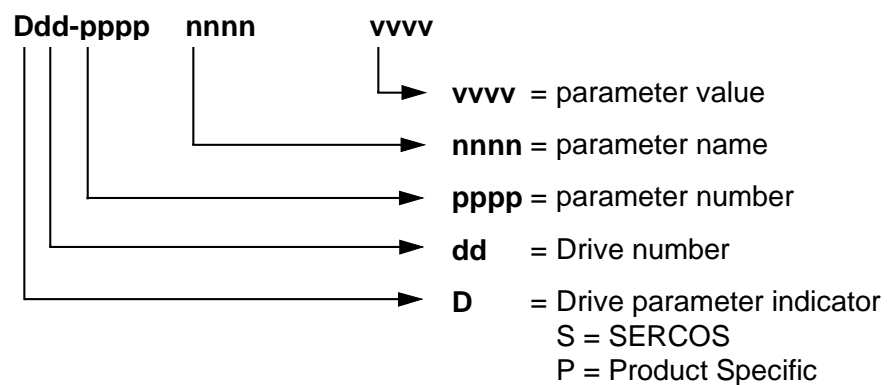
F3 - Task Parameter Screen



For changing of Task parameter numbers and editing of values, refer to **F2 - Axis Parameter Screen** on page 14-26.

F4 - Drive Parameter Screen

DRIVE PARAMETER MENU		
Ddd-ppppp	nnnnn	vvvv
S01-00001	NC Cycle time (TN)	2000
...		
...		
...		
...		
...		
...		
S01-00010	Length of master	20
F1 F2 F3 F4 F7 F8 Home End PgUp PgDn Main Diag		



For changing of Drive parameter numbers and editing of values, refer to **F2 - Axis Parameter Screen** on page 14-26.

This Drive Parameter Menu screen contains S (SERCOS) parameters and P (Product Specific) parameters. Instead of paging down to view P parameters, use the following steps to quickly reach them.

1. Position the cursor over the parameter number
2. Press the Edit key
3. Enter the last SERCOS number (412) and press OK
4. S-0-0412 will appear at the bottom of the list, press **F4** page down to display the first P parameter (P-0-0004).
5. To return back to the start of the SERCOS parameters, page up and position the cursor on a SERCOS parameter.
6. Press the Edit key
7. Enter the first SERCOS number (1) and press OK.

14.11 F6 Security Menu

The Security Menu allows the Teach Pendant manager to assign a protection level code between 0 and 2 for each menu. Different access codes can then be set for various users to provide customized security for system data.

Note: The operation of the BTC06 Security menu is related to the setting in control parameter C-0-0801.

SECURITY MENU	
001 CLC PROGRAM MENU	1
002 CLC TABLE EDIT MENU	1
003 ROBOT JOG MENU	1
004 CONTROL MENU	0
005 REGISTER MENU	2
006 SECURITY MENU	-1
007 CARD PARAMETER MENU	1
008 AXIS PARAMETER MENU	1
009 TASK PARAMETER MENU	1
010 DRIVE PARAMETER MENU	1
011 CLC MAIN MENU	0
	F7 F8
	Main Diag

To alter a menu protection level, place the cursor over the protection level field and press the EDIT button. Key-in the appropriate code (0, 1 or 2) and press OK. The Security Level Menu has a default of -1 to allow initial access to all users.

The user access status for each menu depends on the menu protection level outlined above and the users access code, which is determined by the System Control Register 1, Bits 15 and 16. The access code has to be greater than the menu protection level to allow the user to view and edit a menu. If the levels are the same, the user can only view the menu. A menu with a protection level that is higher than the security level cannot be accessed by a user. The following table lists the level combinations, which determine user access privileges.

Bit 15 Status	Bit 16 Status	Access Code (Bit Resultant)	Protection Level (Preset)	Net Access Status
0	0	0	0 1,2	View Only No Access
1	0	1	0 1 2	View/Edit View Only No Access
0	1	2	0,1 2	View/Edit View Only
1	1	3	0-2	View/Edit

14.12 F8 Diagnostics Menu

When the **F8** key is pressed from any of the Operator Interface Control Menus, the Diagnostics Menu is displayed. The diagnostics menu displays the current Card, Tasks, Axis and Drive status. The diagnostics screen updates continuously. When first entering this menu, by default, Axs=1, Drv=1 and the cursor is positioned on the control number.

```
DIAGNOSTICS MENU
Crd: 03 007 Program Running: AB
      Extended Diagnostic Message

Tsk:AA TASK RUNNING
Tsk:BB TASK RUNNING
Tsk:CC MANUAL MODE
Tsk:DD MANUAL MODE

Axs:01 No Axis Message

Drv: 02 303 Position Mode Encoder 1 / lag

F1   F2   F3   F4           F7
Num+ Num- Rest Log           Main
```

Positioning The Cursor

The up/down arrow keys will position the cursor on the menu item the user may wish to edit.

NOTE: *The Teach Pendant cannot edit the card number or task.*

Cursor Positioned Axis and Drive Number

By pressing the (F1 Num+) or (F2 Num-), the Axis number will increment up or down, respectively.

By pressing the Edit key, the operator can enter the desired Axis number and press the OK key to accept.

By pressing the (F3 Reset) key, the information on the screen is refreshed.

By pressing the (F4 Log) key, the screen will display a list of error that contain the following details:

- Log number
- Date and Time
- Error code

For a complete listing of Diagnostics, refer to the *VisualMotion 9 Trouble Shooting Guide*, Monitoring and Diagnostics.

14.13 Error Screen

If an error is detected during operation, the pendant automatically enters the Error screen and displays a message about the error condition.

If the *BTC06 enable bit (Register 1, bit 14)* is on and an error occurs, the BTC06 will force all tasks into manual mode.

Pressing escape after an error occurs will display the Diagnostic Menu.

```
DIAGNOSTICS MENU
Crd:03 420 Drive 6 Shutdown Error
Tsk:AA MANUAL MODE
Tsk:BB MANUAL MODE
Tsk:CC MANUAL MODE
Tsk:DD MANUAL MODE
Axs:01 No Axis Message
Drv:04 028 Excessive Deviation
F1   F2   F3   F4           F7
Num+ Num- Rest Log           Main
```

F3 - Reset

A basic "Shutdown" error can be cleared by pressing **F3**. If the error is a configuration or hardware error, the source of that error must first be corrected before it can be cleared by the pendant.

15 Textual Language Programming

15.1 Overview

As an alternative to graphical icon programming, VisualMotion also provides a textual language programming environment. Textual language permits writing VisualMotion programs using Windows' Notepad or a similar ASCII text-only editor that does not use extended characters for formatting. The instruction syntax combines features of basic and assembly language, with built-in extensions such as an IF-ELSE-ENDIF construct.

15.2 Directives

Directive instructions are used to control the compiler, the loading of programs between the control and host, and the manner in which VisualMotion manages the different portions of a user program. The following directives are only used to initialize or configure a VisualMotion system. With the exception of the EVENT and TASK directives, the remaining directives are removed from the instruction stream and are not executed during the normal operation of a program.

- EQU (Equate)
- DEFINE (Label a Variable)
- EVENT/START (Start of Event function)
- EVENT/END (Mark End of Event)
- PLS/INIT
- TASK/START (Define the Start of a Task(s))
- TASK/END (Mark the End of a Task)
- VAR/INIT

15.3 VisualMotion Textual Instructions

Instruction Format

The entire control instruction line is limited to a maximum length of 80 characters including the optional label or mark, instruction, arguments, and comment.

White spaces may be used to format lines for easier readability. You may use tabs or spaces for white space. However, do not use white spaces in the middle of a label, mark, instruction, option or argument. If you need to separate a label or mark's characters for clarity, use the underscore character, "_".

VisualMotion Text Language program instructions should be written using the following generalized format:

Example:

```
MARK: INSTRUCTION/OPTION ARG1, ARG2, ...ARGn ;COMMENT
```

where:

MARK:

This instruction is an optional user-assigned symbolic name immediately followed by a colon (:). A mark is typically used as an entry point to a series of program instructions that begin with the line containing the mark.

INSTRUCTION

This is the control instruction mnemonic for the instruction.

OPTION

This is a modifier that may be required by the instruction and is separated from the preceding instruction by a forward slash (/). White space is not allowed between the instruction and its modifier.

ARG1, ARG2, ...ARGn

The number of arguments is specific to the instruction, ranging from none to several. If more than one argument is supplied to an instruction, the arguments must be separated by commas. The argument(s) must be separated from the instruction by one or more white spaces.

Note: An equivalent label may be used in place of a literal specification (such as a constant, variable or table entry), unless otherwise noted.

COMMENT

This is a text string that may be entered to describe the purpose of an instruction. The comment must begin with a semi-colon (;). Comments are used only as an aid to understand the instruction. When generating the executable program code, the compiler ignores everything from the semicolon to the end of the line.

Using Comments

While you're in the process of writing a program you know what you intend the program instructions to accomplish. You may not remember as well a month, or six months later. Using comments liberally always makes it easier to understand what the program is supposed to do. Multiple lines beginning with semicolon may be used to enter several lines

of descriptive information about a subroutine, or as a "header" at the beginning of a program or major program section.

Example:

```
;HOMING SUBROUTINE
;This routine homes the x and y axes
;Expects as Inputs -
; I/O register N = a value, bit n = another value
;Changes outputs -
; I/O register N, bit n
;Changes system variables -
; name1
; name2
;etc.

SUBR01: ;a mark must be used to identify the beginning of
        ;a subroutine's instructions
        .
        . ;subroutine program instructions
        .
RETURN  ;return to the instruction following the calling
        GoSub instruction
```

The available VisualMotion text language instructions are listed in alphabetical order. Each instruction is listed with a description, its syntax and examples.

Braces ("{" and "}") are used to contain optional arguments, where more than a single argument may be supplied.

Example:

```
AXIS/EVENT axis, event1 {, event2, event3, event4}
```

where:

axis and event1 are required, and event2, event3 and event4 are optional.

Labels generated by the DEFINE or EQU directives may be used in place of constant or variable values.

AXIS/EVENT

The *AXIS/EVENT* instruction can enable up to four specified repeating position events for a single-axis, ELS, ratio, or velocity mode axis. The events are armed immediately, and an event function will execute each time the axis reaches the absolute position stored in the "a" field of the event specified in the event table.

Syntax:

```
AXIS/EVENT axis, event1 {, event2, event3, event4}
```

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid axes	axis number
event1 event2 event3 event4	integer - constant - variable Ix, I[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid events	event table index

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
.
Axis_One equ 1
CAM_2 equ 15
CAM_3 equ 16
.
.
axis/event Axis_One,CAM_2{,CAM_3} ;activates repeating
                                events
                                ;CAM_2 and CAM_3 on
                                Axis_One
.
.
```


AXIS/HOME

AXIS/HOME signals the digital drive to perform its drive-internal homing routine. VisualMotion program execution pauses until a homing completed signal (or error message) is received from the drive.

Before *AXIS/HOME* can be used, the appropriate homing parameters must be set in the digital drive. Refer to the respective digital drive manual for a description of the internal homing function and the required drive parameters and settings.

AXIS/HOME can be used for both coordinated and non-coordinated motion.

Syntax:

```
AXIS/HOME axis
```

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

      .
      .
define I05,Axis_AX,
Axis_BX equ 4
speed_BX equ 100.00
accel_BX equ 250.00
      .
      .
axis/home Axis_AX ;homes the DDS-2 drive specified
                  by integer variable 5
axis/home Axis_BX ;homes the DDS-2 drive 4

```

AXIS/INITIALIZE

AXIS/INITIALIZE is used for single axis non-coordinated motion. The instruction associates the specified axis with the user task containing the instruction. Initial speed and acceleration specified by variables may be modified by the user program.

Syntax:

```
AXIS/INITIALIZE axis, speed, accel
```

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number
speed	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label		initial speed
accel	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label		initial acceleration

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
define F10,speed
define F11,accel
define I01,Grip_Event
grip_event equ 5
TaskA_Ax1 equ 3
speed equ 100.0
accel equ 1000.0
.
.
.
axis/initialize TA_Ax1, speed, accel
axis/move TA_Ax1, A, 100.0, Grip_Event
.
.
.

```

AXIS/MOVE (Single Axis, Non-Coordinated)

AXIS/MOVE is used for single axis non-coordinated motion. Digital drives support this feature internally; no control path generation is needed, reducing the computational load on the path planner. Because the motion profile is generated within the drive, time-related events cannot be used with the *AXIS/MOVE* instruction.

Syntax:

AXIS/MOVE axis, mode, dist, {event1, event2, event3, event4}

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable lx, l[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid axes	axis number
mode	ASCII character	A = absolute R = relative	specifies the type of move
dist	float - constant - variable Fx, F[x], - global variable GFx, GF[x] - label		distance to move
event1 event2 event3 event4	integer - constant - variable lx, l[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid events	table index of first event table index of second event table index of third event table index of fourth event

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

speed      equ  100.0
accel      equ  1000.0
grip_event equ  5
TaskC_Ax3  equ  7
F50        equ  100.0
.
.
.
axis/initialize TC_Ax3, speed, accel
axis/move      TC_Ax3, A, F50, Grip_Event
.
.
.
    
```

AXIS/RATIO

AXIS/RATIO establishes a mathematical relationship between the number of revolutions of the slave axis that will result from the rotation of the specified master axis, according to the formula:

$$\text{Slave axis velocity} = \text{Master axis velocity} * (\text{Sfactor}/\text{Mfactor})$$

The *AXIS/RATIO* command automatically updates the master factor parameter A-0-0031 and slave factor parameter A-0-0032. The separate factors are normalized before division so that the calculation maintains maximum precision with repeating decimals such as 2/3.

Specifying a negative number for one of the factors produces reversed rotation of the slaved axis.

Multiple *AXIS/RATIO* instructions may be used to link several slave axes to a single master. A slaved axis must not be assigned to any task other than the task containing the master axis.

Syntax:

AXIS/RATIO axis1, axis2, Mfactor, Sfactor

where:

Argument	Valid Type(s)	Range	Description
axis1	integer - constant - variable Ix, I[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid axes	master axis number
axis2	integer - constant - variable Ix, I[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid axes	slave axis number
Mfactor	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label	32 bits	master axis factor
Sfactor	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label	32 bits	slave axis factor

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Follow ELS Master Axis

Optionally (through parameter A-0-0038), the slave axis may follow the velocity feedback of the master or be maintained in velocity mode instead of position mode. An ELS master may be used as the master axis, by choosing axis 0.

A ratio mode slave can now follow the ELS master (virtual or real). On a system without ELS drives, all slaves can follow the ELS master so that there is no lag between slaves. To follow the ELS master, enter a 0 for

the master axis in the axis_setup icon or axis/setup text language command. If no ELS slave axes are present, it is necessary to set the following ELS parameters using the PARAMETER/INIT instructions: C-0-0150, C-0-0151, C-0-0152, C-0-1000, C-0-1001.

Example:

```
Task_A:
task/start      A                ;start of code for task A
task/axis       1, 2             ;assign axes for task A
axis/ratio      1, 3, 4.0, -5.0 ;axis 3 linked to axis 1 at 5/4
.
.
.
;Operation results in 5 revolutions of axis 3 (in the
reverse direction) for each ;4 revolutions of axis 1.
```

AXIS/SPINDLE (Continuous Velocity Mode)

The *AXIS/SPINDLE* instruction tells the specified axis (and consequently the digital drive) to run at a constant velocity. The acceleration ramp profile is generated within the drive, according to the previously set drive acceleration parameters. The sign of the speed argument determines the direction of rotation.

This instruction also enables the axis to go; the *AXIS/START* instruction is not necessary. The current speed of the axis can be monitored by reading the axis speed parameter from the drive using the *PARAMETER/GET* (Load Parameter to a Variable) instruction.

Syntax:

```
AXIS/SPINDLE axis, rpm
```

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	0 to the maximum number of valid axes	axis number
rpm	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label	32 bits	revolution per minute

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
.
speed equ 100.0
accel equ 1000.0
TA_Ax1 equ 4
.
.
axis/initialize TA_Ax1, speed, accel
axis/spindle TA_Ax1, -22.2
.
.
```

AXIS/START

AXIS/START initiates motion for the specified axis. The instruction signals the digital drive(s) to begin motion toward the target position set by the *AXIS/MOVE* instruction. The axis must be programmed for non-coordinated motion. (E.G., single-axis or velocity mode.)

Once an axis has been started, each new *AXIS/MOVE* instruction initiates a new move, without requiring an *AXIS/START*.

If the specified axis is set to "-1", the instruction initiates motion for all single-axis and velocity mode axes programmed in the task. A program may set target positions for several axes using multiple *AXIS/MOVE* instructions, then initiate motion to all axes at the same time with a single "-1" argument in the *AXIS/START* instruction.

Syntax:

AXIS/START axis

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	0 to the maximum number of valid axes	axis number

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
TaskA_AX1 equ 1 ;use drive 1
TaskA_AX2 equ 3 ;use drive 3
TaskA_AX3 equ 5 ;use drive 5
.
.
axis/move TaskA_AX1, A, F50 ;set position for drive 1
axis/move TaskA_AX2, A, F55 ;set position for drive 3
axis/move TaskA_AX3, R, F60 ;set position for drive 5
.
.
axis/start -1 ;start motion, all axes
.
.
.

```

AXIS/STOP

AXIS/STOP halts motion for the specified axis. The instruction signals the digital drive(s) to decelerate to zero velocity using the rate set in the drive's deceleration parameter. The axis must be programmed for non-coordinated motion. (E.G., single-axis or velocity mode.)

Once an axis has been stopped, an *AXIS/MOVE* instruction does not effect movement of the axis until an *AXIS/START* instruction is executed.

If the specified axis is set to "-1", the instruction disables motion for all single-axis and velocity mode axes programmed in the task. A typical use is to synchronize motion on several axes, or to stop all motion on an I/O or error condition.

Syntax:

AXIS/STOP axis

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable Gx, G[x] - label	0 to the maximum number of valid axes	axis number

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
TaskA_AX1   =      1
TaskA_AX2   =      3
TaskA_AX3   =      5
.
.
.
axis/move   TaskA_AX1, A, F50   ;set position for drive 1
axis/move   TaskA_AX2, A, F55   ;set position for drive 3
axis/move   TaskA_AX3, R, F60   ;set position for drive 5
axis/start  -1                  ;enables all drives
.
.
.
axis/stop   -1                  ;stops motion on all axes
.
.
.
    
```

AXIS/WAIT (Axis Wait For In-Position)

AXIS/WAIT suspends task execution until the digital drive associated with the specified axis signals the control that the axis is in position and zero velocity. The drive's in-position and zero velocity values are set by the drive's Position Window and Zero Velocity parameters.

If the specified axis is set to "-1", the instruction waits until all axes in the task are positioned. A typical use is to synchronize motion on several axes.

If the axis position is never achieved, the task remains waiting (suspended indefinitely).

Syntax:

```
AXIS/WAIT    axis
```

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	0 to the maximum number of valid axes	axis number

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
event1      equ    10
Turn_Gripper_On  equ    12
TA_Ax1      equ    3
define      F10, speed
define      F11, accel
speed       equ    00.0
accel       equ    1000.0
.
.
.
Axis/Initialize  TA_Ax1, speed, accel
Axis/Move        TA_Ax1, R, 100.0, Event1, Turn_Gripper_On
Axis/Wait        TA_Ax1
.
.
.

```


CALL (retval = CALL)

This instruction calls the function 'function_label' with the arguments specified, and optionally returns a value.

Syntax:

```
retval=CALL function_label, arg1, arg2, ...argN
```

where:

Argument	Valid Type(s)	Range	Description
function_label	label	any valid function label	name of function
arg1 ...argN	Fx, lx, GFx, Glx, integer label, float label or point label		values passed to function
retval (optional)	Fx, lx, GFx, Glx, integer label or point label		variable to receive return value of function

Example:

```
; Main task calling subroutine and returning value
Task_A:      TASK/START A
             I1 = CALL    sub, 5, 10, 20
             TASK/END A

; Subroutine to multiply input values
sub:         FUNCTION/START U
             FUNCTION/ARG COUNT1, I, 1, 300
             FUNCTION/ARG COUNT2, I, 1, 300
             FUNCTION/ARG COUNT3, I, 1, 300
             LOCAL/VARIABLE    TAB, I
             TAB = (COUNT1 * COUNT2) * COUNT3
             FUNCTION/END TAB
```

CAM/ACTIVATE

The *CAM/ACTIVATE* instruction is used to associate a CAM to an axis and to supply coefficients for using the CAM. Refer to chapter 7, Icon Programming, for further information on CAM theory of operation.

Syntax:

CAM/ACTIVATE axis, CAM, M, N, H, L

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number
CAM	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid CAM tables	specify CAM table ID
M, N, H, L	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label	32 bits	define coefficients in CAM equation. If L is not specified, it is assumed L=0.

CAM/ADJUST

The *CAM/ADJUST* instruction selects which phase adjust to perform and starts the phase adjust.

There are two phase offset values for a CAM axis: a master phase adjust, and a slave phase adjust. The master phase adjust shifts the position in the CAM table relative to the master position. The slave phase adjust shifts the position of the slave axis. Since it is not related to the shape of the CAM, the slave phase adjust is not multiplied by any of the CAM factors.

The control can perform only one phase adjust at a time. This instruction has no effect until the previous phase adjust is complete if the phase adjust type is different. Bit 4 in the axis status register is set to (0) when a phase offset is in progress, and (1) if the phase offset is complete.

Syntax:

CAM/ADJUST axis, mode, type, degrees

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable Gx, G[x] - label	any valid axis number	axis to adjust
mode	integer - constant	1 - master phase adjust 2 - slave phase adjust	selects master or slave (Mph or Sph in equation)
type	integer - constant	1 - absolute 2 - incremental 3 - continuous + 4 - continuous -	
degrees/ percent	Fx, GFx, Gx, integer or label		phase adjust target value

Example:

```

.
.
CAM/ADJUST 1,2,1,35 ;select and start absolute slave
                    phase adjust of axis 1 at 35 degrees
.
.
    
```

CAM/BUILD

The *CAM/BUILD* instruction builds a CAM internally in the control and stores it in the control for the indicated CAM number. The control's ABS point table is used to build the CAM. The ABS point elements may be changed from within the program. Changes in the point table do not affect the CAM until the *CAM/BUILD* command is executed. For the PCAM option, the X elements of the point table are the master positions, and the Y elements are the corresponding slave positions. The control builds a jerk-limited position profile between these points.

The CAM generation may take a second or more. The 'wait' option can be set to (0) to exit this instruction immediately and keep executing instructions while the CAM is being built.

The instruction can be used to check if a CAM is ready for activation. If the 'wait' option is set to (1), the program flow will be stopped in this instruction until the CAM is ready for activation.

The *CAM/BUILD* instruction can be used to store a CAM to an inactive location in the control. After the CAM has been built, the *CAM/ACTIVATE* instruction is used to activate it for an axis.

Syntax:

```
CAM/BUILD CAM_number, CAM_type, start_point, end_point,
           wait,axis, source
```

where:

Argument	Valid Type(s)	Range	Description
CAM_number	integer - constant - variable Ix, I[x] - global variable Glx, GI[x] - label	1 - 40 = CAM stored on control	integer or constant containing number of CAM to be built
CAM_type	integer - constant	1 = use PCAM utility 2 = use Spline utility 3 = use VCAM utility 4 = use ACAM utility	CAM generation option
start_point	integer - constant		starting index in the ABS point table to use for CAM generation
end_point:	integer - constant		end index in the ABS point table used for CAM generation
wait:	integer - constant	0 or 1	0 = don't wait for completion of build instruction 1 = wait for the CAM to be ready for activation
axis	integer - constant - variable Ix, I[x] - global variable Glx, GI[x] - label	1 to the maximum number of valid axes	axis number
source	integer - constant	0 = points 1 = float array	

Note: Because the CAM is stored in nonvolatile memory in the control, it is not necessary to execute this command each time through the program. A flag variable can be set and checked the next time through the program to avoid long delays when starting the program. For on-line changes, a register bit or variable should be checked each time through the program loop to avoid continually generating the CAM, which consumes control resources and can slow down the program.

It is necessary to size the point table at compile-time to allow enough points for the profiles that will be needed.

Errors will be issued by the control when:

- the selected CAM is currently active for any axis.
- the point range exceeds the bounds of the point table.
- less than two points are defined.
- the CAM number is not valid (out of range or drive is not configured).
- an error occurred while sending the CAM to the drive.
- when using the PCAM option, and the first x position isn't 0 and the last x position isn't 360.
- the x position exceeds the modulo of the master.

CAM/INDEX

Index CAMs are control CAMs that use equations to compute a position, as opposed to a normal CAM, which uses a point table. They operate in real-time which allows their start and end positions to be freely changed within the next index cycle. This makes them ideally suited for high speed film feed applications where the seal time must be held constant over line speed.

Syntax:

```

CAM/INDEX    CAM_number, float_block, int_block
VAR/INIT     (required for floats)
VAR/INIT     (required for integers)
  
```

Note: In order to compile the 10 floats and 4 integers with values, a VAR/INIT instruction must be used for each variable type. Otherwise, the variables are initialized with zeros.

where:

Argument	Valid Type(s)	Range	Description
CAM_number	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 - 40 = CAM stored on control	integer or constant containing number of CAM to be used
float_block	float - variable Fx, F[x]		Starting block for the following 10 floats: - profile_length - profile_start - profile_stop - diameter - event_time - accel - decel - dwell - percent_dwell - reserve_F1
int_block	integer - variable Ix, I[x]		Starting block for the following 4 integers: - Flag1 - profile_type - reserve_I2 - reserve_I1

Number of Allowable Active CAMs

CAM Indexers are identified by a unique number. The number of active CAMs (control table CAMs and CAM Indexers combined) in a VisualMotion control system is limited by the amount of processing power.

- GPP controls (PPC-R) limit the number of active running CAM Indexers to 4. GPP control systems can have a maximum of 40 built CAMs.

CAM/STATUS

This instruction sets a task's status word with a logical result of the CAM status being check. After a CAM is download or build, time is required to calculate and store the new CAM. When a CAM is activated on the control, it does not run until the CAM is at the end of its cycle.

Note: The CAM/STATUS instruction is used as an argument within the IF (If-Else-Endif Conditional Branch) instruction to monitor the status of a CAM.

Syntax:

```
IF      CAM/STATUS   CAM_number   test   condition
```

where:

Argument	Valid Type(s)	Range	Description
CAM_number	integer or constant	1-40	CAM to be checked
test	ASCII characters	!= ==	not equal equivalent to
condition	constant or label	0 - no valid CAM is stored 1 - CAM is being calculated 2 - CAM is being sent to drive (drive CAMs only) 3 - CAM is ready for activation 4 - CAM is active and running	CAM Condition

Note: Errors will be issued when the CAM number is not valid (out of range or drive is not configured).

Example:

```
.
; CAM/Status variations
if CAM/status 1 != 4
    F30 = 5
endif
if CAM/status ( 1) == 4
    F30 = 5
endif
if (CAM/STATUS ( 1) == 4 )
    F30 = 5
endif
```

CAPTURE/ENABLE

The *CAPTURE/ENABLE* instruction enables a feedback capture event for an axis. Each axis can have up to four active capture events (one for each probe edge).

Specifying "0" for the event number disables capture for the specified probe transition.

Probe specifies which of the two probe inputs will be enabled or disabled, and a transition direction (positive or negative going) for the input. An error is issued if the probe and edge were not selected as a trigger using a *CAPTURE/SETUP* instruction.

When the drive detects the selected transition on its probe input, the control transfers the probe position into the event table element "a" (i.e., *EVT[n].a*) specified by the event number, and executes the optional associated event function.

This instruction does not enable a repeating event. A *CAPTURE/ENABLE* instruction must be executed for each new edge transition to be captured.

Syntax:

```
CAPTURE/ENABLE    axis, probe, event
```

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number
probe	integer - constant - label	1 = Probe 1 positive transition 2 = Probe 1 negative transition 3 = Probe 2 positive transition 4 = Probe 2 negative transition	specify digital drive probe and edge transition
event	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid events	enable specified event and trigger on probe transition

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
·
CAPTURE/SETUP Axis_1, In_Place, Not_Place ;setup triggers
·                                     1,2
·
·
```


CAPTURE/SETUP

CAPTURE/SETUP is executed only at program activation. This instruction configures a control axis and associated digital drive to use the internal position feedback capture capability of the drive. One or two triggers can be selected, each may be triggered by either a low-to-high or high-to-low transition of either probe input.

The trigger selects a probe input edge to be enabled in the drive and allocated to SERCOS real-time bit 1. The control also allocates space in the drive's cyclic data for the captured probe position.

Syntax:

CAPTURE/SETUP axis, trigger1, trigger 2

where:

Argument	Valid Type(s)	Range	Description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number
trigger1	integer - constant - label	1 = Probe 1 positive transition 2 = Probe 1 negative transition 3 = Probe 2 positive transition 4 = Probe 2 negative transition	specifies the first drive's probe and trigger edge
trigger2	integer - constant - label	1 = Probe 1 positive transition 2 = Probe 1 negative transition 3 = Probe 2 positive transition 4 = Probe 2 negative transition	specifies the second drive's probe and trigger edge

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
CAPTURE/SETUP Axis_1, In_Place, Not_Place ;setup for trigger
.
CAPTURE/ENABLE Axis_1, In_Place, Mark_edge ;capture
                                           position on
                                           trigger 1
.
CAPTURE/ENABLE Axis_1, Not_Place, Mark_edge ;capture
                                           position on
                                           trigger 2
    
```

COORD_ARTICULATION**Syntax:**

COORD_ARTICULATION ID#, control register, status register,
float block, integers block

where:

Argument	Valid Type(s)	Range	Description
ID number	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 - 2	Up to 2 Coordinated Articulation can be configured in project
control_register	integer - constant	1 - 1024 use default values	
status_register	integer - constant	1 - 1024 use default values	
float_block	float - variable Fx, F[x]	use default values	Starting block for 88 floats
integer_block	integer - variable Ix, I[x]	use default values	Starting block for 20 integers:

DATA/SIZE

The *DATA/SIZE* instruction can be used within each task to specify the amount of memory to be allocated to each data type required by the task. The total control memory allocated for a program is the sum of the allocations for each data type, in each of the four tasks. Once allocated, memory is global and may be accessed by all four tasks.

The *DATA/SIZE* instruction can only be used within the four main tasks. It cannot be used within a subroutine or event function.

Note: Every task containing coordinated motion must allocate the necessary number of absolute and/or relative point tables with each task.

Syntax:

DATA/SIZE Integers, floats, Apoints, Rpoints, Events, Zones, Lists, Steps, Functions

Where:

Argument	Valid Type(s)	Range (Memory Allocation in Bytes)	Description
Integers	integer - constant - label	4 bytes per integer	task memory allocated for Integer variables
floats	" "	4 bytes per integer	task memory allocated for float variables
Apoints	" "	44 bytes per absolute points table	task memory allocated for Absolute Point Table entries
Rpoints	" "	44 bytes per relative points table	task memory allocated for Relative Point Table entries
Events	" "	120 bytes per event	task memory allocated for Event Table entries
Zones	" "	28 bytes per zone	task memory allocated for Zone Table entries
Lists	" "	84 bytes per sequencer list	task memory allocated for sequencer lists
Steps	" "	24 bytes per sequencer step	task memory allocated for sequencer steps
Functions	" "	28 bytes per sequencer function	task memory allocated for sequencer functions

Example:

```

Task_A:
TASK/START   A
DATA/SIZE    20, 20, 30, 30, 2, 3, 1, 2, 3

TASK/END     A

Task_B:
TASK/START   B
DATA/SIZE    0, 0, 15, 15, 0, 0, 0, 0, 0 ; points tables
                                                allocated for
                                                coordinated
                                                motion task

TASK/END     B
    
```

DEFINE (Label a Variable)

The *DEFINE* directive instruction is used to assign a symbolic name to an integer or float variable in the corresponding variable table. It does not initialize the variable to any value. If you need the variable to have an initial value, the value must be set using an expression, after the *DEFINE* instruction and before the variable is used in the program. Using *DEFINE* permits you to use a mnemonic name instead of the "F[n]" or "I[n]" standard format for floats or integers. A defined name may have up to 20 ASCII characters; no white spaces are allowed. A defined name is global to all VisualMotion tasks, subroutines and events.

Unlike equates, defined names are downloaded and stored in separate variable tables in the control. Tables allow access to the variables by a user device through the control's serial communications port; or, if used, through a BTC06 teach pendant.

As with the EQU (Equate) instruction, the compiler generates an error if a symbolic name is used in a user program before it has been defined, or if the name has previously been used to define a variable, axis or identify a subroutine.

Syntax:

```
DEFINE variable, label
```

where:

Argument	Valid Type(s)	Range	Description
variable	integer - variable, Ix or Fx - global variable, GIx or GFIx	a valid variable	
label	ASCII string	20 characters maximum	

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
Define F10, Rate_Of_Change ; Names F10
Define I22, Logical_State ; Names I22
```

DELAY (Suspend Task Execution)

When the program execution flow encounters the DELAY instruction, the task issuing the instruction is suspended for the specified period of time. After the delay, task execution resumes with the next instruction.

Syntax:

`DELAY time`

where:

Argument	Valid Type(s)	Range	Description
time	integer - constant - variable I[x] - global variable GI[x] - label	1 to 360000 (32 bit maximum)	time delay in milliseconds

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

ELS/ADJUST (Adjust ELS Axis)

For ELS drives in velocity synchronous mode, this instruction sets the Ratio Fine Adjust parameter (P-0-0083) on the drive. This parameter is a percent from -100 to 300, allowing an adjustment from stopping the slave to four times the parametric ratio (S-0-0236 and S-0-0237, set in parameter mode).

For ELS drives in phase synchronous mode, this instruction adjusts the phase offset in degrees. The drive will shift in position relative to the master based on the value in this constant or float variable.

If either parameter needs to read, the parameter transfer instruction can be used. This is useful when a variable must be set to an initial phase offset value. The parameter for phase offset is A-0-0151.

Syntax:

`ELS/ADJUST slave_axis, fine_adjust, adjust_type`

or,

`ELS/ADJUST slave_axis, phase_offset, adjust_type`

where:

Argument	Valid Type(s)	Range	Description
slave_axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number of slave
fine_adjust / phase_offset	float - constant - variable Fx, F[x] - global variable GFx, GF[x] - label	Fine: -100% to +300% Phase: 0° to +360°	new value of velocity sync. fine ratio adjust new value of phase sync. phase offset

Argument	Valid Type(s)	Range	Description
adjust_type	integer - constant	1= absolute 2 = incremental 3 = continuous + 4 = continuous -	

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
ELS/INIT      1, 1, 2, 1, 1 ; axis initiate to velocity
                synchronization
```

```
ELS/ADJUST   1, -100, 1 ; Fine adjust for velocity
                sync.
```

```
:
```

Note: The fifth argument in the ELS/INIT instruction determines the following synchronization types:

- 1 = Velocity Synchronization
 - 2 = Phase Synchronization
 - 3 = CAM Synchronization
-

ELS/GROUPM

The *ELS/GROUPM* instruction is used to initialize an ELS Group. The arguments in this instruction identify the group number, control register, status register, start of the float block and the start of the integer block.

Syntax:

```
ELS/GROUPM group_number, control_register, status_register,
           float_block, integer_block
VAR/INIT   (required for floats)
VAR/INIT   (required for integers)
```

Note: In order to compile the 20 floats and 9 integers with values, a VAR/INIT instruction must be used for each variable type. Otherwise, the variables are initialized with zeros.

where:

Argument	Valid Type(s)	Range	Description		
group_number	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 - 8	Up to 8 ELS Groups can be initialized		
control_register	integer - constant	1 - 512 use default values			
status_register	integer - constant	1 - 512 use default values			
float_block	float - variable Fx, F[x]	use default values	Starting block for the following 20 floats: <table border="0" style="width: 100%;"> <tr> <td style="width: 50%; vertical-align: top;"> <ul style="list-style-type: none"> - G#_SYNC_ACCEL - G#_SYNC_VEL - G#_M1 - G#_N1 - G#_REL_M_PH - G#_REL_S_PH - G#_ABS_M_PH - G#_ABS_S_PH - G#_H_LOCKON - G#_H_RUN </td> <td style="width: 50%; vertical-align: top;"> <ul style="list-style-type: none"> - G#_H_LOCKOFF - G#_H_USER - G#_LOCK_WIN - G#_STOP_DECEL - G#_JOG_ACCEL - G#_JOG_VEL - G#_JOG_INC - G#_JOG_ABS - G#_JOG_WIN - G#_LOCK_OFFSET </td> </tr> </table>	<ul style="list-style-type: none"> - G#_SYNC_ACCEL - G#_SYNC_VEL - G#_M1 - G#_N1 - G#_REL_M_PH - G#_REL_S_PH - G#_ABS_M_PH - G#_ABS_S_PH - G#_H_LOCKON - G#_H_RUN 	<ul style="list-style-type: none"> - G#_H_LOCKOFF - G#_H_USER - G#_LOCK_WIN - G#_STOP_DECEL - G#_JOG_ACCEL - G#_JOG_VEL - G#_JOG_INC - G#_JOG_ABS - G#_JOG_WIN - G#_LOCK_OFFSET
<ul style="list-style-type: none"> - G#_SYNC_ACCEL - G#_SYNC_VEL - G#_M1 - G#_N1 - G#_REL_M_PH - G#_REL_S_PH - G#_ABS_M_PH - G#_ABS_S_PH - G#_H_LOCKON - G#_H_RUN 	<ul style="list-style-type: none"> - G#_H_LOCKOFF - G#_H_USER - G#_LOCK_WIN - G#_STOP_DECEL - G#_JOG_ACCEL - G#_JOG_VEL - G#_JOG_INC - G#_JOG_ABS - G#_JOG_WIN - G#_LOCK_OFFSET 				
integer_block	integer - variable Ix, I[x]	use default values	Starting block for the following 9 integers: <table border="0" style="width: 100%;"> <tr> <td style="width: 50%; vertical-align: top;"> <ul style="list-style-type: none"> - G#_CONFIG - G#_MSTR1_AXIS - G#_MSTR2_AXIS - G#_ACTIVE_STATE - G#_ACTIVE_CAM </td> <td style="width: 50%; vertical-align: top;"> <ul style="list-style-type: none"> - G#_LOCKON_CAM - G#_RUN_CAM_ID - G#_LOCKOFF_CAM - G#_USER_CAM </td> </tr> </table>	<ul style="list-style-type: none"> - G#_CONFIG - G#_MSTR1_AXIS - G#_MSTR2_AXIS - G#_ACTIVE_STATE - G#_ACTIVE_CAM 	<ul style="list-style-type: none"> - G#_LOCKON_CAM - G#_RUN_CAM_ID - G#_LOCKOFF_CAM - G#_USER_CAM
<ul style="list-style-type: none"> - G#_CONFIG - G#_MSTR1_AXIS - G#_MSTR2_AXIS - G#_ACTIVE_STATE - G#_ACTIVE_CAM 	<ul style="list-style-type: none"> - G#_LOCKON_CAM - G#_RUN_CAM_ID - G#_LOCKOFF_CAM - G#_USER_CAM 				

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

It is strongly recommended that the programmer use default register and variable assignments. This makes documentation and modifications to user programs an easier task over the scope of a project. Refer to the following table for default values.

Group Number	Group Control Registers	Group Status Registers	Group float Starting Block	Group Integer Starting Block
1	152	243	F170	I140
2	153	244	F200	I150
3	154	245	F230	I160
4	155	246	F260	I170
5	156	247	F290	I180
6	157	248	F320	I190
7	158	249	F350	I200
8	159	250	F380	I210

Example:

```
ELS_GROUPM 1, 152, 243, F170, I140
VAR_INIT I140, 0, 1, 2, 0, 0, 38, 39, 40,0
VAR_INIT F170, 20, 20, 1, 1, 0, 0, 0, 0, 0.5, 1.0, 0.5,
1, 1, 100 ,20 ,20 , 1, 0, 1, 180
```

ELS/GROUPS

GPP supports a maximum of eight ELS Groups. The *ELS/GROUPS* instruction is used to assign a slave axis to an ELS group. Each slave axis that is part of the same ELS Group requires an *ELS/GROUPS* instruction.

Syntax:

```
ELS/GROUPS group_number, axis_number, motion_type
```

where:

Argument	Valid Type(s)	Range	Description
group_number	integer - constant	1 to 8	identifies to which group the slave axis belongs
axis_number	integer - constant - variable Ix, I[x] - global variable Gx, G[x] - label	1 to maximum number of valid axes	axis number of slave
motion_type	integer - constant	1 = velocity synchronization 2 = phase synchronization 3 = card CAM 4 = drive CAM	

Along with each ELS/GROUPS instruction, a PARAMETER/INIT (Initialize a Parameter) instruction may be required, depending on the set motion_type. The following SERCOS parameters can be used:

- P-0-0108 (Master drive polarity)
 - a "0" indicates same direction as master
 - a "1" indicates reverse direction
- S-0-0236 (Master drive 1 revs.)
- S-0-0237 (Slave drive 1 revs.)

ELS/GROUPS motion_type	Required parameters using PARAMETER/INIT
Velocity Synchronization	P-0-0108, S-0-0236, S-0-0237
Phase synchronization	P-0-0108, S-0-0236, S-0-0237
Card CAM	no PARAMETER/INIT required
Drive CAM	P-0-0108

Note: When specifying P class SERCOS parameters, make sure to add an offset of 32768 to the desired parameter.

Example:

```
ELS/GROUPS 1, 1, 2 ;assign axis 1 to group 1 in phase synch
.
PARAMETER/INIT D, 1, 32876, 0 ;set P-0-0108 to 0
PARAMETER/INIT D, 1, 236, 1 ;set S-0-0236 to 1
PARAMETER/INIT D, 1, 237, 1 ;set S-0-0237 to 1
```

ELS/MODE (Set ELS Axis Mode)

The *ELS/MODE* instruction sets the operating mode for an axis. It switches the drive between single-axis mode and synchronous mode. If slave_axis is set to (-1), the mode is switched for all slave axes in the task.

To enable synchronization, set "mode" to 2. The drive is synchronized to the master in phase synchronous or velocity synchronous mode. When switching into phase synchronous mode, the phase offset parameter is automatically set to (slave position - master position).

To disable synchronization at any time during the program, set "mode" to 1. The slave switches to single-axis mode and is halted. To run the slave in single-axis mode, the single-axis instructions may be used.

Set "mode" to 3 to switch an axis that is configured for single-axis mode into velocity mode.

Syntax:

```
ELS/MODE slave_axis, mode
```

where:

Argument	Valid Type(s)	Range	Description
slave_axis	integer - constant lx, l[x], Glx, Gl[x] - label	1 to the maximum number of valid axes	axis number of slave
mode	integer - constant lx, l[x], Glx, Gl[x] - label	1 = single-axis mode 2 = synch. to master 3 = velocity	

the value of the constant, only the single instance of the value in the EQU instruction at the beginning of your program has to be modified.

Every time a user program is compiled, the symbolic names in the program are replaced with their equated literal value.

Example:

```
RATIO1 equ 2.7348 ;a ratio between two coupled shafts
DONE equ 0x1000 ;a bit mask for an I/O register
```

The compiler generates an error if a symbolic name is used in a user program before it has been defined, or if the name has previously been used to define a variable, axis or identify a subroutine.

EVENT/DONE (Signal Event Completed)

The *EVENT/DONE* instruction changes the status of an active event. Time based events are taken out of the event timer queue and distance based events are made inactive if in the event queue or pending. *EVENT/DONE* is primarily used to disable repeating timer events; however, it immediately disables any event.

Executing an *EVENT/DONE* has no effect on an inactive event.

Syntax:

```
EVENT/DONE event
```

where:

Argument	Valid Type(s)	Range	Description
event	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes events	disable the event

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

EVENT/DONE Example:

```
time equ 1 ;use 0 for time
dist equ 1 ;use 1 for dist

Task_A:
Task/Start A
EVT[3].t = time ;Set event 3 type to time
EVT[3].a = 10 ;Set event 3 time to 10ms
EVT[3].m = "Motion Event Message" ; Set the message for
event 3
EVT[3].f = Gripper ;Set the subroutine mark for event 3
. ;Program code
.
EVENT/TRIGGER 3 ;Trigger event 3
EVENT/WAIT 3 ;Wait until done
. ;More program code
.
EVENT/DONE 3 ;event done
TASK/END A
Gripper: ;Define event 0 function subroutine
EVENT/START ;Mark the beginning of the event
PLC/SET 1, 2 ; Turn on gripper
EVENT/END ; Mark the end of the event
```

EVENT/END (Mark End of Event)

The *EVENT/END* instruction is a compiler directive used within a program to indicate the end of an event function. Executing this directive instruction returns program execution to the invoking program, enabling normal cycling. Every *EVENT/END* instruction must begin with an *EVENT/START*. Unlike a called subroutine, an event function should not have a RETURN (Return From Subroutine) instruction at the end of the event function program instructions. Proper termination of an event is handled by the compiler and control multitasking executive.

Syntax:

```
EVENT / END
```

Refer to the **EVENT/DONE Example** for details.

EVENT/START (Start of Event function)

The *EVENT/START* instruction is a compiler directive used within a program to indicate the beginning of an event function. Every *EVENT/START* instruction must end with an *EVENT/END*. Unlike a called subroutine, an event function should not have a RETURN (Return From Subroutine) instruction at the end of the event function program instructions. Proper termination of an event is handled by the compiler and control multitasking executive.

Syntax:

```
EVENT / START
```

Refer to the **EVENT/DONE Example** for details.

EVENT/TRIGGER (Trigger a Task Event)

The *EVENT/TRIGGER* instruction starts a repeating timer event. The subroutine specified by the event is repeatedly called at the rate specified in the event table entry. Task A's events have the highest priority.

An *EVENT/TRIGGER* has no effect on repeating timer events that are already active.

An *EVENT/TRIGGER* instruction in a subroutine does not execute and enters a wait state if the task's event queue is full.



CAUTION

The execution time of the event function must not exceed the specified event repeat time. If the event function does not return before the event is re-triggered, the multitasking executive will execute the event function again, as soon as the event function returns. Program task execution will be blocked. Remember that ALL pending events, for a task, will be completed before any task resumes execution.

Syntax:**EVENT/TRIGGER** event*where:*

Argument	Valid Type(s)	Range	Description
event	integer - constant - variable lx, l[x] - global variable Glx, Gl[x] - label	1 to the maximum number of valid axes events	start event timer

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Refer to the **EVENT/DONE Example** for details.

EVENT/WAIT (Pause Task for Event Done Signal)

The *EVENT/WAIT* instruction suspends the issuing task until an instruction is executed.

A control run-time error results if this instruction is directed towards an inactive event.

Syntax:**EVENT/WAIT** event*where:*

Argument	Valid Type(s)	Range	Description
event	integer - constant - variable lx, l[x] - global variable Glx, Gl[x] - label	1 to the maximum number of valid axes event	pause task for event

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Refer to the **EVENT/DONE Example** for details.

FUNCTION/ARG

The *FUNCTION/ARG* instruction is used to declare arguments in a function. Each argument in a function is declared on a separate line immediately following the *FUNCTION/START* instruction. The arguments are then pass (used) in the function when encountered in the program flow. The arguments must be declared in the order in which they are to appear in the *CALL* (*retval = CALL*) instruction or in the sequence table. A minimum and maximum range is used to define the limits for each argument.

A maximum of 5 arguments can be used in a function. Local variables can also be defined for a function using the *LOCAL/VARIABLE* instruction. A maximum of 16 local variables can be used per function. A function can contain arguments and/or local variables. If a combination of arguments and local variables are used in a function, the total number between them can not exceed 16. Function argument labels are accessible from the teach pendant in a sequencer.

Syntax:

```
FUNCTION/ARG label, type, min_value, max_value
```

where:

Argument	Valid Type(s)	Range	Description
label	ASCII string	1 to 20 characters	name of argument
type	ASCII character	'I' = integer 'F' = float 'ABS' = absolute point index 'REL' = relative point index	specifies type of access
min_value	integer - constant - label		optional minimum value of argument
max_value	integer - constant - label		optional maximum value of argument

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
.
;Function: plc_float
;Reads value from a 16-bit PLC register and return a scaled
value
plc_float: FUNCTION/START U           ;access from teach pendant
      FUNCTION/ARG regnum, I, 150, 160 ;register number of PLC
                                      register
      FUNCTION/ARG scaler, F, 1, 100000 ;scaling value from PLC
                                      to float
      LOCAL/VARIABLE itemp, I         ;local temporary variable
      LOCAL/VARIABLE retval, I       ;local temporary variable
      PLC/READ regnum, 1, itemp      ;read register from PLC
      retval = itemp * scaler        ;scale to float
      FUNCTION/END retval            ;return with retval
```

FUNCTION/END

The FUNCTION/END instruction defines the end of a function. If no value is included, the function returns 0. The function may return only one value, which may be a float or integer. The value is returned in the variable specified in the CALL instruction.

Syntax:

FUNCTION/END value (optional)

where:

Argument	Valid Type(s)	Range	Description
value	integer - constant - label		return value from function
(optional)	label		

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
FUNCTION/END retval ;return with retval
```

FUNCTION/START

The FUNCTION/START instruction indicates the start of a function. The arguments and return value are declared in the FUNCTION/ARG, LOCAL/VARIABLE and FUNCTION/END instructions. If access_type is not included, the function is not accessible in the user function list.

Syntax:

function_label: **FUNCTION/START** access_type

where:

Argument	Valid Type(s)	Range	Description
function_label	label	any valid function label	name of function
access_type	ASCII character	'U' = Accessible in user function sequencer list 'N' = Not accessible	specifies type of access

Example:

```
;Main task calling subroutine and returning value
Task_A: TASK/START A
        I1 = CALL sub, 5, 10, 20
        TASK/END A

;Subroutine to multiply input values
sub: FUNCTION/START U
      FUNCTION/ARG COUNT1, I, 1, 300
      FUNCTION/ARG COUNT2, I, 1, 300
      FUNCTION/ARG COUNT3, I, 1, 300
```

```

LOCAL/VARIABLE      TAB, I
TAB = (COUNT1 * COUNT2) * COUNT3
FUNCTION/END TAB

```

GOSUB (Go To Subroutine)

The *GOSUB* instruction saves the current position in the program execution of the user task, then begins executing subroutine instructions beginning with the instruction identified by the *GOSUB* mark.

When an executing subroutine encounters a RETURN (Return From Subroutine) instruction, program execution resumes at the next executable program instruction following the *GOSUB* instruction that called the subroutine.

Syntax:

```
GOSUB target
```

where:

Argument	Valid Type(s)	Range	Description
target	label	any valid label at the beginning of a subroutine	go to marked function with return

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

L00:   If I01 >= I02 GOSUB Sub2
      Else
          I10 = 255                ;Set I10 to 255
          GOSUB Sub1              ;Call subroutine one
      Endif
      .
      .
      .
Sub1:   I01 = I01*I02              ;Multiply
      RETURN
Sub2:   I01 = I01/I03             ;Divide
      RETURN

```


GOTO (Go To Mark)

The *GOTO* instruction is used as an unconditional branch to jump to other instruction in the program identified by the mark in the this instruction. Since a *GOTO* does not save the current position in the program execution flow, branching to a subroutine or event function results in an error when the execution flow encounters the RETURN instruction.

Syntax:

```
GOTO    target
```

where:

Argument	Valid Type(s)	Range	Description
target	label	a valid target identifying a unique position in a program.	

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
L00:  If I01 >= I02
      GoTo L10
      Else
        I10 = 255
        GoTo L15
      Endif
L10:  I01 = I01*I02
      .
      .           ;program instructions
      .
      GoTo Start1
L15:  I01 = I02/I03
      .
      .           ;alternative program instructions
      .
      GoTo Start2
```

IF (If-Else-Endif Conditional Branch)

The IF-ELSE-ENDIF structure provides conditional execution of the program instructions between the IF and ENDIF keywords, depending upon the evaluation of a test relationship. The control structure also provides an optional ELSE keyword for conditional program branching between two alternative series of program instructions.

If the expression is true (has a non-zero value), and there is no ELSE keyword, the program instruction between the IF and ENDIF keywords are executed. If the optional ELSE keyword is used, the instructions between the IF and ELSE keywords are executed. Program execution then continues with the first program instruction after the ENDIF keyword.

If the expression is false (has a zero value) and there is no ELSE keyword, the program continues with the first instruction after the ENDIF keyword. If the optional ELSE keyword is used, the program instructions between ELSE and ENDIF keyword are executed. The program execution then continues with the first program instruction after the ENDIF keyword. "IF" structures may be nested up to eleven deep. All IF keyword instructions must be balanced with a matching ENDIF.

Syntax:

```
IF(value1 test value2)
    or
    PLC/TEST ( reg, bit )
    CAM/STATUS ( CAM_number, test, condition )
```

ELSE

ENDIF

where:

Argument	Valid Type(s)	Range	Description
value1	integer or float - constant - variable Ix, I[x] or Fx, F[x] - global variable Glx, Gl[x] or GFx, GF[x] - label	any valid constant, variable, or table entry	
value2	same as above		
test	ASCII characters	> >= < <= != == (CAM/STATUS limited to !=,==)	greater than greater than, or equal to less than less than, or equal to not equal equivalent to
reg	integer, constant or label, Ix, Glx	1-512 PPC-R	Source register to test
bit	integer, constant or label, Ix, Glx	1-16	Bit with register to test
condition	constant or label	0 - no valid CAM is stored 1 - CAM is being calculated 2 - CAM is being sent to drive (drive CAMs only) 3 - CAM is ready for activation 4 - CAM is active and running	CAM Condition

Argument	Valid Type(s)	Range	Description
CAM_number	integer or constant	1-40	CAM to be checked

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

:
L00:      If I01 >= I02
           Goto L10
           Else
             I10 = 255
             gosub Sub2
           Endif
L10:      I01 = I01*I02
           .
           .           ;program instructions
           .
Sub2:     I01 = I02*I03
           RETURN

```

PLC/TEST Variations:

Associates the PLC/TEST instruction to an IF-Else-Endif conditional branch. Tests a bit in the specified I/O register table entered. If PLC/TEST checks for an "on" ? condition. If !PLC/TEST checks for an "off" ? condition.

Example:

```

; PLC/TEST variations
if( PLC/TEST 100, stewart )
  F30 = 5
endif
if( !PLC/TEST 100, 2 )
  F30 = 5
endif
if PLC/Test 100, 1
  F30 = 5
endif
if !PLC/Test 100, 2
  F30 = 5
endif
if( PLC/Test( 100, 1 ) )
  F30 = 5
endif
if plc/test( I5, GI6 )
  else
    F31 = 6
  endif
endif
F31= 3.1415927

```

CAM/Status Variations:

Associates the CAM/STATUS instruction to an IF-Else-Endif conditional branch.

After a CAM download or after the CAM/BUILD command, time is required to calculate and store the new CAM. When a CAM is activated on the control, it does not run until the CAM is at the end of its cycle.

IF CAM/STATUS checks the status of a specified CAM. The test is limited to == (equivalent to) or != (not equal) arguments. The status integer returns the specified CAM condition.

Errors will be issued when the CAM number is not valid (out of range or drive is not configured).

Example:

```

.
.
.
; CAM/Status variations
  if CAM/status 1 != 4
    F30 = 5
  endif
  if CAM/status ( 1) == 4
    F30 = 5
  endif
  if (CAM/STATUS ( 1) == 4 )
    F30 = 5
  endif

```

KINEMATIC (Use a Kinematic Definition for a Task)

Each task can have its own kinematic to allow for motion in Cartesian space. This instruction tells the path planner which set of equations in the optional kinematic library to use for motion.

This instruction is active at the start of a Task, and is removed before normal system cycling. The KINEMATIC instruction is allowed only in the main Tasks: A, B, C, and/or D.

Syntax:

KINEMATIC kinematic

where:

Argument	Valid Type(s)	Range	Description
kinematic	integer - constant - label	a valid kinematic library number	

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
kinematic    10        ; Use kinematic library number 10
.
.
.

```

LOCAL/VARIABLE

The LOCAL/VARIABLE instruction declares a local variable including its label and type. A maximum of 16 local variables can be used per function. A function can contain arguments and/or local variables. If a combination of arguments and local variables are used in a function, the total number between them can not exceed 16.

Syntax:

```
LOCAL/VARIABLE label, type
```

where:

Argument	Valid Type(s)	Range	Description
label	ASCII string	1 to 20 characters	name of variable used as local variable
type	ASCII character	'I' = integer 'F' = float 'ABS' = absolute point index 'REL' = relative point index	type of variable used as local variable

Example:

```
;Main task calling subroutine and returning value

Task_A:      TASK/START A
             I1 = CALL          sub, 5, 10, 20
             TASK/END A

;Subroutine to multiply input values

sub:  FUNCTION/START U
      FUNCTION/ARG COUNT1, I, 1, 300
      FUNCTION/ARG COUNT2, I, 1, 300
      FUNCTION/ARG COUNT3, I, 1, 300
      LOCAL/VARIABLE   TAB, I
      TAB = (COUNT1 * COUNT2) * COUNT3
      FUNCTION/END TAB
```

MESSAGE/DIAG (Task Diagnostic Message Definition)

The *MESSAGE/DIAG* instruction provides a user program with the ability to send a diagnostic message to the user that is associated with a specific instruction in the user program. *MESSAGE/DIAG* embeds a tag or index into the instruction following the *MESSAGE/DIAG* instruction.

A diagnostic message provided by the control can be used by the user interface to select and display the message for system maintenance or troubleshooting.

The diagnostic message may also be used for program debugging during program development in a manner similar to embedding print instructions at critical points in the program.

Syntax:

MESSAGE/DIAG message

where:

Argument	Valid Type(s)	Range	Description
message	ASCII	up to 79 characters	diagnostic text message

Example:

```

.
Move/line ABS[1]
MESSAGE/DIAG    Waiting for safe zone bit in PLC to be true
;
;The diagnostic message lets the user know that the program
;has advanced to the program instructions that check the
safe
;zone I/O bit.
;
L10:    Plc/read          I01, 0
        If I01 & 0x100
            Goto L20
            Delay          1000
            Goto          L10
        EndIf
L20:
.

```

MESSAGE/STATUS (Task Status Message Definition)

The *MESSAGE/STATUS* instruction is similar to the *MESSAGE/DIAG* instruction. This instruction also embeds a tag or index into the message following the *MESSAGE/STATUS* instruction. However, this instruction's character strings are stored in a different table than *MESSAGE/DIAG* messages and provide a different series of message index numbers.

By providing two tables, status messages may be used for prompting the system operator, diagnostic messages for program debugging and system maintenance.

Syntax:

```
MESSAGE/STATUS      message
```

where:

Argument	Valid Type(s)	Range	Description
message	ASCII	up to 79 characters	status text message

Example:

```

.
.
Move/line ABS[1]
MESSAGE/STATUS Waiting for safe zone bit in PLC to be true
;
;The status message lets the user know that the program
;has advanced to the program instructions that check the
safe
;zone I/O bit.
;
L10:   Plc/read          I01, 0
       If I01 & 0x100
           Goto L20
           Delay          1000
           Goto          L10
       EndIf
L20:
.

```

MOVE/CIRCLE (Coordinated Move with Circular Interpolation)

The *MOVE/CIRCLE* instruction provides circular motion along a path in Cartesian space defined by three sets of coordinates. Program modifiable integer variables may be used to specify the starting and ending coordinates in the point tables.

The first form provides circular movement using absolute coordinates. Motion occurs from the end point of the last move or current position, through the absolute coordinate specified by the table reference of the first argument, and ends at the absolute coordinate specified by the table reference of the second argument.

The second form allows motion to begin at a relative offset from the end of the last move or current position, then moves through a relative offset specified by the table reference of the second argument, and ends at an absolute coordinate specified by the table reference of the third argument.

Syntax:

```
MOVE/CIRCLE ABS[index1], ABS[index2]
```

OR

```
MOVE/CIRCLE REL[index1], REL[index2]
```

where:

Argument	valid type(s)	range	description
index1	integer - constant - variable lx - global variable Glx - label	a valid absolute or relative point table entry	defines the mid-point of the circular arc
index2	integer - constant - variable lx - global variable Glx - label	a valid absolute or relative point table entry	defines the end-point of the circular arc

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
.
;enter points into absolute point table
ABS[0]x      =      0.0
ABS[1]y      =      1.0
ABS[1]z      =      0.0
.
.      ;assume the current position is -1.0, 0.0,
0.0
MOVE/CIRCLE ABS[1], ABS[2]
;the move results in a semi-circular move
;from x=-1, y=0, through x=0, y=1; to x=1, y=0
```


MOVE/JOINT (Coordinated Move Joint Point to Point)

The *MOVE/JOINT* instruction is an absolute point-to-point move, with only the endpoint of the move specified. The actual path taken to the specified point is undeterminable (i.e., not linear or circular) and may assume whatever form the path planner requires; however, once programmed and planned, the path is repeatable. The path is optimized to minimize time and uses accel and constant velocity rates for the coordinated axes (as opposed to the line and circle coordinated motion commands which use the world rates).

Most commonly used with robotic applications, the *MOVE/JOINT* instruction is also the only method of elbow repositioning.

Syntax:

`MOVE/JOINT ABS[index]`

where:

Argument	Valid Type(s)	Range	Description
index	integer - constant - variable lx - global variable Glx - label	a valid absolute point table entry	specifies the endpoint of move

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
ABS[1].X   =   10.0       ;\
ABS[1].Y   =   20.0       ; -> Sets Absolute point 1
ABS[1].Z   =   15.0       ;/
.
.
.
MOVE/JOINT ABS[1]       ; An absolute point-to-point
                        ; Coordinated motion
.
    
```

MOVE/LINE (Coordinated Move with Straight Line Interpolation)

The *MOVE/LINE* instruction provides motion along a straight line path in Cartesian space, defined by two sets of coordinates.

The first form provides linear movement to absolute coordinates. Motion occurs from the end point of the last move or current position, and ends at the absolute coordinate specified by the table reference of the last argument.

The second form begins motion at a relative offset from the end of the last move or current position, and ends at an absolute coordinate specified by the table reference of the last argument.

Note: Program modifiable integer variables may be used to specify the starting and ending coordinates in the point tables.

Syntax:

MOVE/LINE ABS[index]

or,

MOVE/LINE REL[index]

where:

Argument	Valid Type(s)	Range	Description
index	integer - constant - variable Ix - global variable Glx - label	a valid absolute or relative point table entry	specifies the endpoint of move

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
;assume that the current position is -1.0, 0.0, 0.0

MOVE/LINE    ABS[1]
;This results in a linear move from x=-1, y=0; to x=0, y=1
.
.
.

```

PARAMETER/BIT (Initialize Parameter Bit)

The PARAMETER/BIT instruction is used to set one or more bits in a parameter when the program is activated. The instruction tests for a zero/non-zero logical constant or variable to set or clear the bit(s) enabled by a specified bit mask.

Syntax:

```
PARAMETER/BIT type, set, param, source, bit
```

where:

Argument	Valid Type(s)	Range	Description
type	ASCII character	A C D T	Parameter Type - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	1 Ix, I [x], Glx, GI[x] A, B, C or D	Parameter Set 1 for System parameters a valid drive or axis number for Drive or Axis parameters A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number Add an offset of 32768 for P class drive parameters
source	integer - constant - variable Ix, I[x], Glx, GI[x] - global variable Ix, I[x], Glx, GI[x] - label	zero or, non-zero	logical value for bit - zero value clears the bit - non-zero sets the bit
bit	HEX format or, label	decimal 1 - 16, or hexadecimal, (0X0001 to 0xFFFF)	decimal selects a single bit hexadecimal can select multiple bits e.g., 0x0300 = bits 9 and 10 0x2003 = bits 1,2, and 14

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
TASK_A: TASK/START A
PLC/CLEAR 120, 8 ;Clears bit 8 of register 120
.
PARAMETER/BIT A, 1, 4, I1, 0x0002 ;Sets Rotary mode for
;axis 1
.
AXIS/RATIO 1, 2, 2, 1 ;Axis 2 is linked to axis 1 at 1/2
.
TASK/END A
```

PARAMETER/GET (Load Parameter to a Variable)

The *PARAMETER/GET* instruction retrieves a parameter from the control's system, task or drive parameter tables and stores the value in the specified variable at run-time.

Incorrectly specifying the parameter type will result in a compiler error.

Syntax:

```
PARAMETER/GET      type, set, param, target
```

where:

Argument	Valid Type(s)	Range	Description
type	ASCII character	A C D T	Type of parameter - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	1 Ix, I[x], Glx, GI[x] A, B, C or D	Parameter set - 1 for System parameters - a valid drive or axis number for Drive or Axis parameters - A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number Add an offset of 32768 for P class drive parameters
target	integer or float - variable Fx, F[x], Ix, I[x] - global variable GFx, GF[x] - label	a valid variable	target variable to load with parameter value target variable data type must match parameter data type

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
.
.
PARAMETER/GET D, 1, STATUS, I01
  If I01 == 0
    Goto L10
  Else
    Gosub new_sub
  Endif
.
.
.
L10: MOVE/LINE ABS[2]
      I01 = 100
      PARAMETER/SET D, 1, POSITION, I01
.
.
.
```

PARAMETER/INIT (Initialize a Parameter)

The *PARAMETER/INIT* instruction is used to load the source value into the specified parameter when the program is activated.

Syntax:

```
PARAMETER/INIT    type, set, param, source
```

where:

Argument	Valid Type(s)	Range	Description
type	ASCII character	A C D T	Type of parameter - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	1 Ix, I[x], Gx, G[x] A, B, C or D	Parameter set - 1 for System parameters - a valid drive or axis number for Drive or Axis parameters - A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number Add an offset of 32768 for P class drive parameters
source	integer or float - constant - variable Fx, F[x] - global variable GFx, GF[x], Gx, G[x] - label	a valid variable	source variable to load to parameter source variable data type must match parameter data type

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
PARAMETER/INIT  A, 1, 120, 32.0  ;Sets axis 1 parameter 120
                                   ;to initialize acceleration

```

PARAMETER/SET (Set a Parameter)

The *PARAMETER/SET* instruction updates the control's system, task or drive parameter table with a new value for the specified parameter at run-time.

Incorrectly specifying the parameter type will result in a compiler error.

Syntax:

```
PARAMETER/SET      type, set, param, source
```

where:

Argument	Valid Type(s)	Range	Description
type	ASCII character	A C D T	Type of parameter - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	1 integer, or label A, B, C or D	Parameter set - 1 for System parameters - a valid drive or axis number for Drive or Axis parameters - A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number Add an offset of 32768 for P class drive parameters
source	integer or float - constant - variable Fx, F[x] - global variable GFx, GF[x], Glx, GI[x] - label	a valid variable	source variable to load to parameter source variable data type must match parameter data type

Example:

```

.
.
.
PARAMETER/GET  D, 1, STATUS, I01
  If I01 == 0 Goto L10
  Else
    Gosub New_Subroutine
    .
    .
    .
L10: Move/Line ABS[2]
      I01 = 100
PARAMETER/SET  D, 1, POSITION, I01
.
.
.

```

PATH/ABORT (Aborts Coordinated Motion)

The PATH/ABORT instruction decelerates motion within the path then aborts all segments placed in the task's queue by the path planner. Motion must be restarted with a cycle start, it cannot be continued. All events are lost.

Syntax:

```
PATH/ABORT task
```

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID character

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
TASK_D:TASK/START D
.
.
REL[1].X      =      1.0          ; Sets relative point 1
.
.
IF (I1 == 3)
    PATH/ABORT D          ; Aborts task D operation
```

PATH/POSITION (Get Current Path Absolute Position)

The current position of the path planner is returned to the specified point table entry. The current contents of the point table are overwritten. The other current specifications in the point table (rate, accel/decel, events, etc.) are not affected. One task may obtain the position of another task by specifying the desired task ID.

Syntax:

```
PATH/POSITION      task, ABS[index]
```

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID
index	integer - constant - variable lx - global variable Glx - label	a valid absolute point table entry	index to a target absolute position table entry

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
L10: path/position 1,ABS[6]          ; Get current position
      F[01] = ABS[6].x*ABS[6].x
      F[01] = F[01] + ABS[6].y*ABS[6].y
      F[01] = F[01] + ABS[6].z*ABS[6].z
      F[01] = SQRT( F[01] );          ; Compute vector length
      if F[01] >= 100.0 the
      goto L10 ; Loop if larger than 100
```

NOTE: Add one more point to this instruction.

Example:

```
ABS[x] defined in Path/Position instruction =
# of absolute points defined in Data/Size +1
```


PATH/RESUME (Resume Coordinated Motion)

The *PATH/RESUME* instruction continues coordinated motion halted by a *PATH/STOP* instruction. Any queued time or distance-based events saved by the *PATH/STOP* instruction will also resume when motion continues.

Syntax:

```
PATH/RESUME task
```

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID character

Example:

```
TASK_C:TASK/START C
.
.
PATH/STOP A ;Stops Coordinated motion of task A
DELAY 3000 ;Waits for 3 seconds
PATH/RESUME A ;Resumes Coordinated motion of task A
.
.
```

PATH/STOP (Halt Coordinated Motion)

The *PATH/STOP* instruction commands the path planner to decelerate motion and halt motion on the path. *PATH/STOP* saves the look-ahead path planned by the path planner and any associated time or distance-based events. Motion and events on the path may be resumed by a *PATH/RESUME* instruction.

Syntax:

```
PATH/STOP task
```

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID character

Example:

```
SAMPLE_RATE EQU 0.0 ; Sets equate
.
.
TASK_A:TASK/START A
.
.
IF (SAMPLE_RATE >= 1000.0)
    PATH/STOP B ; Halts Coordinated motion of task B
.
.
TASK/END A
```

PATH/WAIT (Pause Program for Motion)

The *PATH/WAIT* instruction tests the current state of the path planner for a specified point. Since the path planner is typically one or more segments ahead of physical motion, *PATH/WAIT* can be used to temporarily halt program execution until the path planner begins a specific type of processing for a specified point.

Syntax:

```
PATH/WAIT          task, ABS[index], state
```

OR

```
PATH/WAIT          task, REL[index], state
```

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID
index	integer - constant - variable Ix - global variable Gix - label	a valid absolute or relative point table entry	point table entry specifying position to pause program
state	integer -label	0 to 8	requested state of path planner 0 = segment ready 1 = acceleration 2 = constant velocity 3 = blending 4 = target deceleration 5 = controlled stop 6 = stopped 7 = at target 8 = done

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
TASK_D:TASK/START D
      .
      .
MOVE/LINE  ABS[1]          ; Straight Line Coordinated motion
PATH/WAIT  D, ABS[1], 8 ; Waits until point is done
      .
      .
TASK/END   D
```

PID/CONFIG

The PID/CONFIG instruction is used to setup and initialize a Proportional-Integral-Differential loop on the control. Up to 10 PID/CONFIG instructions can be used in one program. The location of the instruction in the program is not important, it is only executed at program activation.

The optional axis parameter data is added to the cyclic SERCOS data for update every SERCOS cycle time. If using the optional axis parameters, care must be taken to avoid using them for other purposes. Internally, the value is converted to float. It's offset is then subtracted and the resultant multiplied by it's scalar. A list of valid axis parameters can be viewed in drive parameter S-0-0188 "List of configurable Data in the MDT".

Set point can be a program or global variable(Fx, GFx, Glx, lx, or label), or signed or unsigned register. Internally, the value is converted to a float. It's offset is then subtracted and the resultant multiplied by it's scalar.

Feedback and output may be a program or global variable(Fx, GFx, Glx, lx, or label), a signed or unsigned register, or an axis parameter. Three axis parameters(position, velocity, and torque) are selectable in a list box. Other axis parameters can be added by selecting optional1 or optional2 and adding the axis parameter number.

Syntax:

```
PID/CONFIG  loop, type, ctrl_reg, st_reg, loop_time,
set_point_type, set_point_number, set_point_axis,
fdbk_type, fdbk_number, fdbk_axis, output_type,
output_number, output_axis, float_ctrl_block
VAR/INIT    (required for floats)
```

Note: In order to compile the 15 floats with values, a VAR/INIT instruction must be used for each variable. Otherwise, the variables are initialized with zeros.

where:

Argument	Valid Type(s)	Range	Description
loop	integer - constant	1 - 10	selects the PID loop number
type	integer - constant	always 1	reserved for future development
ctrl_reg	integer - constant - label		sets the PID loop control register
st_reg	integer - constant - label		sets the PID loop status register
loop_time	integer - constant	8 - 152 increments of 8	sets how often the PID loop is ran
set_point_type	integer - constant	1 = float 3 = register 4 = +/- register	sets the value for the source (ex: temperature)
set_point_number	- constant - variable Fx, F[x], lx, l[x] - global variable GFx, GF[x], Glx, Gl[x] - label		sets value for set point type

Argument	Valid Type(s)	Range	Description
set_point_axis	integer - constant	always 0	
fdbk_type	integer - constant	1 = float 2 = axis parameter 3 = register 4 = +/- register	source of feedback
fdbk_number	- constant - variable Fx, F[x], lx, l[x] - global variable GFX, GF[x], Glx, Gl[x] - label		sets value for feedback type
fdbk_axis	integer - constant - label	1 to maximum number in system	select an axis number when feedback type is 2
output_type	integer - constant	1 = float 2 = axis parameter 3 = register 4 = +/- register	where value is written
output_number	- constant - variable Fx, F[x], lx, l[x] - global variable GFX, GF[x], Glx, Gl[x] - label		sets value for output type
output_axis	integer - constant - label	1 to maximum number in system	select an axis number when output type is 2
float_ctrl_block	float - variable Fx, F[x] - label		starting block for the 15 PID floats: - PID1_CMD_SCALER - PID1_CMD_BIAS - PID1_FDBK_SCALER - PID1_FDBK_BIAS - PID1_KP_VALUE - PID1_KI_VALUE - PID1_Kd_VALUE - PID1_KI_MAX_VALUE - PID1_MIN_OUTPUT - PID1_MAX_OUTPUT - PID1_KI_PRESET - PID1_OUT_SCALER - PID1_OUT_BIAS - PID1_FDBK_CUTOFF - PID1_FDBK_FILTER

Example:

```
PID/CONFIG 1, 1, 164, 194, 8, 1, F530, 0, 1, F531, 0, 1,
F532, 0, F530
```

```
VAR/INIT F530, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0,
-10.0, 10.0, 0.0, 1.0, 0.0, 0.0
```

PLC/CLEAR (Clear I/O Register Bit)

The *PLC/CLEAR* instruction clears the specified bit in the specified I/O register table entry to 0.

Syntax:

```
PLC/CLEAR    register, bit
```

where:

Argument	Valid Type(s)	Range	Description
register	integer - constant - label	a valid I/O register	specifies an I/O register
bit	integer - constant - label	1 to 16	specifies a bit in the register

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
Cycle_On    equ    10    ; Cycle on register
Go          equ    6     ; Bit for go
.
.
Plc/Clear   Cycle_On, Go ; Clear bit 6 of cycle on
.
.

```

PLC/READ (Read I/O Register(s))

The *PLC/READ* instruction copies the contents of one or more sequential 16-bit registers in the control I/O registers table to sequential entries in the integer variable table.

Syntax:

```
PLC/READ    register, count, target
```

where:

Argument	Valid Type(s)	Range	Description
register	integer - constant - label	a valid I/O register	first register to read
count	integer - constant - label	1 to maximum number of I/O registers	number of registers to read
target	integer - variable Ix, GIx, I[x], GI[x] - label	a valid integer variable	first target variable

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

.
.
.
iobase equ    100
count      equ    20
define I5,ioregs
;
PLC/READ    iobase, count, ioregs ;Read the 20 control I/O
;registers, from
;100 through 120 to
;integer variables
;I05 through I25
.
.
.

```

This instruction copies the contents of twenty 16 bit PLC I/O table entries (I/O registers 101 through and including 120) to the integer variables I01 through and including I20.

PLC/SET (Set I/O Register Bit)

The *PLC/SET* instruction sets a specified bit in a specified control I/O register table to 1.

Syntax:

```
PLC/SET      register, bit
```

where:

Argument	Valid Type(s)	Range	Description
register	integer - constant - label	a valid I/O register	specifies an I/O register
bit	integer - constant - label	1 to 16	specifies a bit in the register

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```

Light_Reg equ 20
Light_On  equ 5
.
.
.
PLC/SET    Light_Reg, Light_On      ; Turn on bit 5 of
register 20
.
.
.

```

PLC/TEST (Test I/O Register Bit)

The *PLC/TEST* instruction tests a bit in the specified I/O register table entry and returns a logical value to the specified integer variable according to the state of the I/O register bit. (The value of the integer variable may then be used to control program flow.)

Syntax:

```
PLC/TEST    register, bit, target
```

where:

Argument	Valid Type(s)	Range	Description
register	integer - constant - label - variable Ix, GIx	a valid I/O register	source register to test
bit	integer - constant - label - variable Ix, GIx	1 to 16	bit within register to test
target	integer - variable Ix, I[x], GIx, GI[x] - label	a valid integer variable	variable to receive binary (1 or 0) test result

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
.
.
PLC/TEST    20, 5, I03    ;set I03 flag to I/O register,
                        ;bit 5 state
If I03 > 0 GOSUB calculate ;do calculate if flag is set
Else
  GOSUB subtract
EndIf
.
.
.
```

PLC/WAIT (Pause Program for I/O)

The *PLC/WAIT* instruction may be used to pause program execution with a task until the state of the specified I/O bit equals the specified logical value.

Syntax:

```
PLC/WAIT    register, bit, state
```

where:

Argument	Valid Type(s)	Range	Description
register	integer - constant - label	a valid I/O register	source register to test
bit	integer - constant - label	1 to 16	bit within register to test
state	integer - variable Ix, I[x], GIx, GI[x]	zero or one	variable containing the logical value required to resume program execution

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

PLC/WRITE (Write to I/O Register(s))

The *PLC/WRITE* instruction copies the contents of one or more sequential integer variables from the integer variable table to the I/O register table.

Syntax:

```
PLC/WRITE    register, count, source
```

where:

Argument	Valid Type(s)	Range	Description
register	integer - constant - label	a valid I/O register	starting target register to write
count	integer - constant - label	1 to maximum number of registers	number of registers to write
source	integer - constant - variable Ix, I[x], GIx, GI[x] - label	a valid integer variable	starting source variable

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
I[01] = 0x0001
I[02] = 0x6666
.
.
.
PLC/WRITE 2, 3, I[01]
.
.
.
```

This instruction writes integer variable I[01] to I/O register 3, and I[02] to I/O register 4.

PLS/INIT

The *PLS/INIT* is a compiler directive. Both the primary and secondary PLS setup instructions are executed by VisualMotion's compiler to initialize the PLS table. The location of the *PLS/INIT* instruction is not important because it is only used by VisualMotion's compiler. Use the runtime instruction (PLS[1].t=1) to change the table in the program flow. The PLS is disabled while the output register is zero or when all bits have zero in both the on and off positions.

A programmable limit switch supports 16 outputs and a phase advance. The position input for the PLS is the ELS Master, which can be an external encoder, an axis feedback or a Virtual Master. The outputs are updated every SERCOS cycle.

Syntax:

```
(PRIMARY)
PLS/INIT switch, 0, register, type, axis, offset
or
(SECONDARY)
switch, element, on position, off position
```

where:

Argument	Valid Type(s)	Range	Description
switch	integer - constant - label	1	
register	integer - constant - label		output register
type *	integer - constant - variable lx	1 - ELS 2 - virtual axis 3 - primary axis 4 - secondary axis	type of axis that the limit switch is tracking * Currently ELS is the only valid 'type'
axis		0 if type 1 or 2 else 1-32	axis number
offset		0-360	PLS phase advance

or

Argument	Valid Type(s)	Range	Description
switch	integer - constant - label	1	
element	integer - constant - label	1-16	switch element outputs (output register bits)
on position	integer - constant - variable lx	0-360	position in which switch element turns on
off position	integer - constant - variable lx	0-360	position in which switch element turns off

Example:

```

Task_A:TASK/START A
    local/variable    var1, I
    data/size 50, 50, 20, 20, 10, 0
;Primary PLS setup command
    PLS/INIT 1,0,120,1,0,0.0
;Secondary PLS setup command
    PLS/INIT 1,1,20,40
    PLS/INIT 1,2,20,40
    PLS/INIT 1,3,20,40
    PLS/INIT 1,4,20,40
    PLS/INIT 1,5,20,40
    PLS/INIT 1,6,20,40
    PLS/INIT 1,7,20,40
    PLS/INIT 1,8,20,40
    PLS/INIT 1,9,20,40
    PLS/INIT 1,10,20,40
    PLS/INIT 1,11,20,40
    PLS/INIT 1,12,20,40
    PLS/INIT 1,13,20,40
    PLS/INIT 1,14,20,40
    PLS/INIT 1,15,20,40
    PLS/INIT 1,16,20,40
;Runtime commands
    PLS[1].r=120                ;register
    PLS[1].o=30.0              ;offset
    PLS[1].on1=30
    PLS[1].on16=120
    PLS[1].off1=120
    PLS[1].off16=300
        var1 =call                loop25
TASK/END A
.
.

```

REGISTRATION

The *REGISTRATION* instruction is used to initialize an auto registration procedure. Registration is the process of referencing a product edge or register mark. This allows positioning errors to be detected and corrected before an upstream (web, product) or downstream (die-cutter, print cylinder, etc.) process takes place, depending on the machine design.

This function tightly couples the control system with the Probe Event on the drive. Registration is accomplished by comparing the captured position to a target value and correcting for the difference. The registration error is automatically assimilated into the axis motion profile, providing a smooth, seamless correction. The registration error can be corrected using S-curve, Triangular, and Trapezoidal correction profiles.

Syntax:

```
REGISTRATION axis, controlRegister, statusRegister,
               floatBlock, IntBlock
VAR/INIT      (required for floats)
VAR/INIT      (required for integers)
```

Note: In order to compile the 15 floats and 4 integers with values, a VAR/INT instruction must be used for each variable. Otherwise, the variables are initialized with zeros.

Default label naming scheme and comment for Registration

15 float variables

```
RGIS_01_ PROPOSITION      ;Registration, Axis 01, expected mark position
RGIS_01_ WIN_START        ;Registration, Axis 01, distance before expected to look
RGIS_01_ WIN_STOP         ;Registration, Axis 01, distance after expected to look
RGIS_01_ COR_START        ;Registration, Axis 01, correction to start at this point
RGIS_01_ COR_STOP         ;Registration, Axis 01, correction to be completed before
RGIS_01_ MAX_CORRECT      ;Registration, Axis 01, maximum correction per period
RGIS_01_ SETPOINT         ;Registration, Axis 01, calculated expected position
RGIS_01_ COR_ERROR        ;Registration, Axis 01, calculated correction distance
RGIS_01_ PROBE_VALUE      ;Registration, Axis 01, feedback position at probe 1 trigger
RGIS_01_ RESERVE_F6       ;Registration, Axis 01, reserve float 6
RGIS_01_ RESERVE_F5       ;Registration, Axis 01, reserve float 5
RGIS_01_ RESERVE_F4       ;Registration, Axis 01, reserve float 4
RGIS_01_ RESERVE_F3       ;Registration, Axis 01, reserve float 3
RGIS_01_ RESERVE_F2       ;Registration, Axis 01, reserve float 2
RGIS_01_ RESERVE_F1       ;Registration, Axis 01, reserve float 1
```

4 integer variables

```
RGIS_01_ FLAG1           ;Registration, Axis 01, flag word 1
RGIS_01_ MISS_MARKS      ;Registration, Axis 01, number of missing marks allowed
RGIS_01_ COR_TYPE        ;Registration, Axis 01, correction type
RGIS_01_ RESERVE_I1      ;Registration, Axis 01, reserve integer 1
```

RETURN (Return From Subroutine)

The *RETURN* instruction is used at the end of a Subroutine. When the subroutine execution encounters a *RETURN* instruction, program flow exits the subroutine and resumes with the program instruction following the *GOSUB* instruction that called the subroutine.

Syntax:

RETURN

Example:

```

      .
      .
      .
Sub1: MOVE/LINE    ABS[2]
          PLC/SET          1,1
          Return
      .
      .
      .

```

ROBOT/ORIGIN

The *ROBOT/ORIGIN* instruction is used in coordinated motion programs to construct a zero frame of reference from the x, y, z, roll, pitch and yaw coordinates of a relative point. This moves the effective origin of the robot from the default to the location specified by the programmed relative point.

For example, if $REL[3] = \{1, 2, 3, 0, 0, 0, \dots\}$ and this point is specified as the robot origin, then the robot origin would be offset by one unit along the x axis, two units along the y axis and three units along the z axis. Once the instruction is executed, all jogging, teaching and path locations are affected by the new origin.

Syntax:

ROBOT/ORIGIN REL[index]

where:

Argument	Valid Type(s)	Range	Description
index	integer - constant - variable Ix - global variable Glx - label	a valid relative point table entry	defines the zero frame of reference for a robot

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

ROBOT/TOOL

The *ROBOT/TOOL* instruction is used in coordinated motion programs to construct a tool frame of reference from the x, y, z, roll, pitch and yaw coordinates of a relative point. This moves the effective end-of-arm tool to the location specified by the programmed relative point.

For example, if $REL[4] = \{0, 0, 10, 0, 0, 0, \dots\}$ and this point is specified as the robot tool location, then the robot tool location would be offset by ten units along the z axis from the faceplate of the robot. Once the instruction is executed, all jogging, teaching and path locations are affected by the end-of-arm tool location.

Syntax:

```
ROBOT/TOOL          REL[index]
```

where:

Argument	Valid Type(s)	Range	Description
index	integer - constant - variable Ix - global variable GIx - label	a valid relative point table entry	defines the tool frame of reference for a robot

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

ROTARY/EVENT

The *ROTARY/EVENT* instruction initializes and arms a repeating rotary event (up to 4) for an axis, ELS Master or ELS Group.

Axis

The feedback position from the virtual or real axis is compared to a trigger position from the event table. When this trigger position is crossed either in the clockwise or counterclockwise direction the event's status is set to pending and the event is queued to the event system for processing.

ELS Master

For an ELS master the process is the same except that the output position of the master serves as the reference signal for determining the trigger position for the event.

ELS Group

Likewise, for a ELS Group the output position of the group serves as the reference signal for determining the trigger position for the event.

Syntax:

```
ROTARY/EVENT  type, id, evt1, evt2, evt3, evt4
```

where:

Argument	Valid Type(s)	Range	Description
type	integer - constant	0 = Axis position feedback 1 (motor) 1 = ELS system master 2 = ELS Group output 3 = Axis position feedback 2 (secondary)	determines the signal source that will be used for triggering the event(s)
id	integer - constant - variable Ix - global variable Glx - label	1-32 when type is axis position 1-6 when type is ELS Master 1-8 when type is ELS Group	identifies the type by number
evt1 - evt4	integer - constant - variable Ix - global variable Glx - label		first through fourth index number into event

Example:

```
ROTARY/EVENT 0, 1, 3, 4, 5, 6
```

SEQUENCER

The Sequencer instruction is used to initialize and/or run a Sequencer list. The sequencer name is a number or label equating to a number. The number has a range of 1 to n, where n is the number of sequencers defined in the Size instruction.

Syntax:

```
SEQUENCER    seq_name
```

where:

Argument	Valid Type(s)	Range	Description
seq_Name	integer - constant - variable Ix - global variable Glx - label	determined by program limits	sequencer name

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
;      .
;      .
; Run sequencer IVORY
      SEQUENCER    IVORY
      TASK/END    A
```

SEQ/LIST

The *SEQ/LIST* instruction assigns a sequencer step list to an existing sequencer along with a number which determines the order the sequencer will follow when executed. The name of the step list is also included in this command.

A step list can be used by more than one sequencer. It can also be used repeatedly within a given sequencer.

Syntax:

```
SEQ/LIST          seq_name, list_number, list_name
```

or

```
seq_name, 0, count
```

where:

Argument	Valid Type(s)	Range	Description
seq_Name	integer - constant - variable Ix - global variable Glx - label	determined by program limits	sequencer name
list_Number	integer	valid list number or 0 for count	number of step list
list_Name	label or count (integer)		name of step list or if range=0 this is count (total number of lists)

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
IVORY equ 1
Task_A: TASK/START A

;Declare 10 sequencers,10 steps, 50 function steps
      DATA/SIZE 50,50,0,0,4,0,10,10,50

; Build sequencer IVORY

; Sequencer IVORY will have 5 steps
      SEQ/LIST IVORY,0,5
      SEQ/LIST IVORY,1,Mold_Open
      SEQ/LIST IVORY,2,Move_In
      SEQ/LIST IVORY,3,Grab_Part
      SEQ/LIST IVORY,4,Move_to_Drop
      SEQ/LIST IVORY,5,Drop_Part
      ....
```

SEQ/STEP

The SEQ/STEP instruction defines a sequencer step within a step list. It identifies an existing function or subroutine and passes on up to five function arguments when executed.

Syntax:

```
SEQ/STEP list_name, step_number, function_name, arg1,
         arg2, ...arg5
```

OR

```
list_name, 0, count
```

where:

Argument	Valid Type(s)	Range	Description
list_Name	label		name of step list
step_Number	integer or 0		
function_Name or count	label or count (integer)	any valid function	name of function or if step_number=0 this is count (the total number of steps)
arg1, arg2, ...arg5	integer, float - constant	determined by program limits	function arguments

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
.
; Step Mold_Open will have 4 functions in it

      SEQ/STEP      Mold_Open,0,4
      SEQ/STEP      Mold_Open,1,vac_on,1, , , ,
      SEQ/STEP      Mold_Open,2,waitcool,1000, , , ,
      SEQ/STEP      Mold_Open,3,chk_part, , , , ,
      SEQ/STEP      Mold_Open,4,move_rdy, , , , ,

; Step Move_In will have 4 functions in it

      SEQ/STEP      Move_In,0,4
      SEQ/STEP      Move_In,1,chk_mold, , , , ,
      SEQ/STEP      Move_In,2,movechk,1,1000, , , ,
      SEQ/STEP      Move_In,3,inmoldms,100,555, , , ,
      SEQ/STEP      Move_In,4,vac_on,4, , , ,

; Step Drop_Part will have 1 function in it

      SEQ/STEP      Drop_Part,0,1
      SEQ/STEP      Drop_Part,1,f1234567890123456789, , , ,
,
;
;
```


TASK/AXES (Task Axes Definition)

The *TASK/AXES* instruction defines the axes and their use within a task. A maximum of six axes may be assigned using one *TASK/AXES* instruction. A task may contain several *TASK/AXES* instruction when the task requires additional axes or uses axes in ratio mode.

When type 4 (ratio mode) is specified, only two axis arguments may be supplied. The second argument is the master axis and the third argument is the slave axis. Axes used in ratio mode must have been previously defined (in any task) by another *TASK/AXES* instruction. The mathematical ratio between the two axes must be set by an *AXIS/RATIO* instruction.

Tasks that do not use motion control do not require a *TASK/AXES* instruction. However, all drives connected to the SERCOS ring must be registered to the control by declaring each drive to it's axis in a *TASK/AXES* instruction, even if the axes are not used. This insures that the control will correctly respond to drives automatically identified by the SERCOS initialization procedures.

The *TASK/AXES* instruction may be used only within the main Tasks: A, B, C, or D. This instruction is only active during download and is removed from the instruction stream before normal program cycling.

Syntax:

```
TASK/AXES    type, axis1, {axis2, axis3}
```

where:

Argument	Valid Type(s)	Range	Description
type	integer -label	1 = single axis motion 2 = coordinated motion 3 = velocity mode (no positioning) 4 = for ratioed axes (master/slave) 5 = ELS Slave mode 6 = Torque Mode1	motion types:
axis1 axis2 axis3	integer - constant - label	1 to the maximum number of valid axes	

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Example:

```
TaskB_Ax1    equ 3                                ; Define Task
B axis one
TaskB_Ax2    equ 4                                ; Define Task
B axis two
TaskB_Ax3    equ 6                                ; Define Task
B axis three
.
.
Task_B:
task/start   B                                    ; Start B
task/axes    1, TaskB_Ax1, TaskB_Ax2, TaskB_Ax3 ; Assign axes
to task B
.
.
```

TASK/END (Mark the End of a Task)

The *TASK/END* instruction is a compiler directive used to identify the ending of program instruction associated with a task. The task instructions do not perform any initialization, but are necessary indicators for the control's compiler. Task instructions should not be nested. All control instructions must appear between *TASK/START* and *TASK/END* instructions that specify one of the four tasks.

Syntax:

```
TASK/END    <task>
```

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID character

Example:

```
Task_A:
TASK/START  A
            .
            .
            .
TASK/END    A
            ;
Task_B:
TASK/START  B
            .
            .
            .
```

TASK/START (Define the Start of a Task(s))

The *TASK/START* instruction is a compiler directive used to identify the start of program instruction associated with a task. The task instructions do not perform any initialization, but are necessary indicators for the control's compiler. Task instructions should not be nested. All control instructions must appear between *TASK/START* and *TASK/END* instructions that specify one of the four tasks. Each task must be identified by a "Task_(task letter):" mark at the beginning of the task's program.

Syntax:

TASK/START <task>

where:

Argument	Valid Type(s)	Range	Description
task	ASCII character	A, B, C or D	task ID character

Example:

```

Task_A:
TASK/START  A
            .           ;
            .           ;program instructions
            .           ;
TASK/END    A
;
Task_B:
TASK/START  B
            .
            .
            .

```

VAR/INIT

The *VAR/INIT* instruction is a compiler directive used to initialize program integer (Ix) and float (Fx) variable values at compile time. This instruction can be used to initialize variable values for other instructions such as...

- CAM/INDEX
- ELS/GROUPM
- ELS/MASTERS
- PID/CONFIG
- REGISTRATION

Note: Refer to the above instructions for details requiring the arguments.

Syntax:

```
VAR/INIT var_start, arg1, arg2, arg3,....., arg20
```

where:

Argument	Valid Type(s)	Range	Description
var_start	- variable Ix, Fx		identifies first variable in a block of program variables
arg1, - arg20	integer - constant	depends on main instruction	initializing value.

Example:

```
VAR/INIT F10, 1, 0, 1, 0, 1, 0, 0, 0, 0, 32000, 0, 1, 0
```

VIRTUAL/MASTER

The *VIRTUAL/MASTER* instruction is used to configure and initialize the two Virtual Masters supported in GPP. The arguments in this instruction identify the master number, control register, status register, start of the float block and the start of the integer block.

Syntax:

```
VIRTUAL/MASTER number, control_register, status_register,
                float_block, integer_block
VAR/INIT       (required for floats)
VAR/INIT       (required for integers)
```

Note: In order to compile the 15 floats and 2 integers with values, a VAR/INIT instruction must be used for each variable type. Otherwise, the variables are initialized with zeros.

where:

Argument	Valid Type(s)	Range	Description																
number	integer - constant - variable Ix, I[x] - global variable Glx, G[x] - label	1 and 2	Up to 2 Virtual Masters can be configured Each Virtual Master requires it's own instruction																
control_register	integer - constant	1 - 512 use default values																	
status_register	integer - constant	1 - 512 use default values																	
float_block	float - variable Fx, F[x]	use default values	Starting block for the following 15 floats: <table border="0" style="width: 100%;"> <tr> <td>- VM#_HOME_POS</td> <td>- VM#_MAX_VEL</td> </tr> <tr> <td>- VM#_REL_MOVE_DIST</td> <td>- VM#_MAX_ACCEL</td> </tr> <tr> <td>- VM#_STOP_POS</td> <td>- VM#_MAX_DECEL</td> </tr> <tr> <td>- VM#_CMD_ABS_POS</td> <td>- VM#_JERK_ENABLE</td> </tr> <tr> <td>- VM#_CMD_VEL</td> <td>- VM#_CUR_POS</td> </tr> <tr> <td>- VM#_CMD_ACCEL</td> <td>- VM#_CUR_VEL</td> </tr> <tr> <td>- VM#_CMD_DECEL</td> <td>- VM#_POS_WIN</td> </tr> <tr> <td>- VM#_E_STOP_DECEL</td> <td></td> </tr> </table>	- VM#_HOME_POS	- VM#_MAX_VEL	- VM#_REL_MOVE_DIST	- VM#_MAX_ACCEL	- VM#_STOP_POS	- VM#_MAX_DECEL	- VM#_CMD_ABS_POS	- VM#_JERK_ENABLE	- VM#_CMD_VEL	- VM#_CUR_POS	- VM#_CMD_ACCEL	- VM#_CUR_VEL	- VM#_CMD_DECEL	- VM#_POS_WIN	- VM#_E_STOP_DECEL	
- VM#_HOME_POS	- VM#_MAX_VEL																		
- VM#_REL_MOVE_DIST	- VM#_MAX_ACCEL																		
- VM#_STOP_POS	- VM#_MAX_DECEL																		
- VM#_CMD_ABS_POS	- VM#_JERK_ENABLE																		
- VM#_CMD_VEL	- VM#_CUR_POS																		
- VM#_CMD_ACCEL	- VM#_CUR_VEL																		
- VM#_CMD_DECEL	- VM#_POS_WIN																		
- VM#_E_STOP_DECEL																			
integer_block	integer - variable Ix, I[x]	use default values	Starting block for the following 9 integers: <table border="0" style="width: 100%;"> <tr> <td>- VM#_POS_MODE</td> <td>- VM#_RESERVE_I1</td> </tr> </table>	- VM#_POS_MODE	- VM#_RESERVE_I1														
- VM#_POS_MODE	- VM#_RESERVE_I1																		

Note: Variables or labels used for arguments must equate to valid values at run-time or an error will result.

It is strongly recommended that the programmer use default register and variable assignments. This makes documentation and modifications to user programs an easier task over the scope of a project. Refer to the following table for default values.

Virtual Master Number	Control Registers	Status Registers	float Starting Block	Integer Starting Block
1	150	241	F100	I100
2	151	242	F120	I105

Example:

```
V_MASTER 1,150,180,F100,I100
VAR_INIT F100,0,1,0,0,20,100,100,500,3200,1000,1000,1,0,0,1
VAR_INIT I100,0
.
V_MASTER 2,151,181,F120,I105
VAR_INIT F120,0,1,0,0,20,100,100,500,3200,1000,1000,1,0,0,1
VAR_INIT I105,0
.
```

16 Index

A

- A-0-0001 Task Assignment 5-147
- A-0-0002 Type of Positioning 5-147
- A-0-0003 Axis Motion Type 5-148
- A-0-0004 Axis Options 5-149
- A-0-0005 Linear Position Units 5-154
- A-0-0006 Reference Options 5-154
- A-0-0007 Configuration Mode 5-155
- A-0-0009 Drive PLS Register 5-156
- A-0-0020 Maximum Velocity 5-156
- A-0-0021 Maximum Acceleration... 5-157
- A-0-0022 Maximum Deceleration .. 5-157
- A-0-0023 Jog Acceleration 5-157
- A-0-0031 Control Cam/Ratio Master
Factor (N) 5-159
- A-0-0032 Control Cam/Ratio Slave
Factor (M) 5-159
- A-0-0033 Control Cam Stretch Factor
(H) 5-159
- A-0-0034 Control Cam Currently Active
..... 5-160
- A-0-0035 Control Cam Position
Constant (L) 5-160
- A-0-0036 Ratio Mode Encoder Type5-
161
- A-0-0037 Ratio Mode Step Rate... 5-161
- A-0-0038 Ratio Mode Options 5-162
- A-0-0100 Target Position..... 5-164
- A-0-0101 Commanded Position 5-164
- A-0-0102 Feedback Position 5-165
- A-0-0110 Programmed Velocity 5-165
- A-0-0111 Commanded Velocity..... 5-165
- A-0-0112 Feedback Velocity 5-166
- A-0-0120 Programmed Acceleration5-
166
- A-0-0131 SERCOS Control Word.. 5-167
- A-0-0132 SERCOS Status Word... 5-167
- A-0-0133 AT Error Count..... 5-167
- A-0-0140 Mfg. Class 3 Status Word5-
168
- A-0-0141 Torque Mode Commanded
Torque 5-169
- A-0-0142 Torque Feedback (cyclic)5-
169
- A-0-0145 Current Motion Type 5-169
- A-0-0150 Programmed Ratio Adjust5-
170
- A-0-0153 Control Phase Adjust Average
Velocity 5-171
- A-0-0155 Control Phase Adjust Time
Constant 5-172
- A-0-0156 Phase Offset Velocity
Feedback 5-172
- A-0-0157 Current Phase/ Control Cam
Master Offset 5-173
- A-0-0158 Relative Phase Offset
Distance Remaining..... 5-173
- A-0-0159 Ratio Adjust Step Rate... 5-173
- A-0-0160 Commanded Ratio Adjust5-
174
- A-0-0161 Control Cam Programmed
Slave Adjust..... 5-174
- A-0-0162 Control Cam Current Slave
Adjust (Sph) 5-175
- A-0-0163 Control Cam Output Position5-
175
- A-0-0164 ELS Options 5-175
- A-0-0170 Probe Configuration Status5-
177
- A-0-0171 Probe 1 Positive Captured
Value 5-178
- A-0-0172 Probe 1 Negative Captured
Value 5-178
- A-0-0173 Probe 2 Positive Captured
Value 5-178
- A-0-0174 Probe 2 Negative Captured
Value 5-178
- A-0-0180 Optional Command ID #15-
178
- A-0-0181 Optional Command ID #25-
179
- A-0-0182 Optional Command ID #35-
179
- A-0-0185 Optional Feedback ID #15-179
- A-0-0186 Optional Feedback ID #25-180
- A-0-0190 Command Data #1 5-180
- A-0-0191 Command Data #2 5-181
- A-0-0192 Command Data #3 5-181
- A-0-0195 Feedback Data #1 5-181
- A-0-0196 Feedback Data #2 5-181
- A-0-0200 MDT Multiplex Selection List
(DKC 2.3 only) 5-182
- A-0-0201 AT Multiplex Selection List
(DKC 2.3 only) 5-182
- A-0-0202 MDT Multiplex Ident List (DKC
2.3 only) 5-183
- A-0-0203 AT Multiplex Ident List (DKC
2.3 only) 5-183
- A-0-2000 List of All Parameters 5-184
- A-0-2001 List of Required Parameters5-
184
- About VisualMotion 2-170
- Absolute Point Edit 14-15
- absolute points table
data structure 3-26
- Absolute Table Menu 14-14
- Accel..... 3-13
- Activating a Program (PA) 13-21
- Allowable Drive Address Range 5-47
- Archive..... 2-70
- ASCII Protocol for VisualMotion 13-4
- AT Error Count..... 5-167
- AT Multiplex Ident List (DKC 2.3 only)5-
183
- AT Multiplex Selection List (DKC 2.3
only) 5-182
- AT Telegram
Configuring 5-7
- Attribute Subclass 13-8
- Available Program Memory 5-44
- Axis Fast Jog Velocity 5-40
- Axis Large Increment 5-39
- Axis Motion Type 5-148
- Axis Options..... 5-149
- Axis Parameter Menu 14-26
- Axis Parameters 5-22
- Axis Feedback Capture (Registration)
..... 5-23
- Axis Setup 5-22
- A-0-0025 Maximum Jog
Increment 5-158
- A-0-0026 Maximum Jog Velocity
..... 5-158
- A-0-0030 Master Axis for Ratio
Function..... 5-159
- Axis Status 5-22
- Electronic Line Shaft
A-0-0151 ELS Programmed
Phase Offset..... 5-171
- Electronic Line Shafting 5-23
- Multiplexing Parameters (DKC 2.3
only) 5-24
- Optional SERCOS Data..... 5-24
- Parameter Lists 5-24
- Axis Parameters: 5-147

- Axis Slow Jog Velocity 5-41
 - Axis Small Increment 5-39
 - Axis2 3-14
- B**
- Backspaces and White spaces 13-6
 - X10 port settings 2-160
 - X16 port settings 2-161
 - Block Size 13-21
 - Bootloader Firmware Version 5-48
 - Branch 3-20
 - Breakpoint 2-144
 - BTC06 Error Screen 14-31
 - BTC06 jog method 14-20
 - BTC06 jog system
 - axis jog menu 14-20
 - joint jog menu 14-20
 - world jog menu 14-20
 - BTC06 keyboard I/O map 4-25, 14-8
 - BTC06 keyboard operation
 - cursor control and editing 14-9
 - jogging control 14-9
 - number or letter selection 14-9
 - task control 14-10
 - teach control 14-10
 - BTC06 menus
 - F1 Program Menu 14-11
 - F2 Table Edit Menu 14-14
 - F3 Jog Menu 14-19
 - F4 Control Menu 14-21
 - F5 Register I/O Menu 14-24
 - F6 Parameter Menu 14-25
 - axis parameter menu 14-26
 - card parameter menu 14-26
 - drive parameter menu 14-28
 - task parameter menu 14-27
 - F7 Security Menu 14-29
 - F8 Diagnostic Menu 14-30
 - BTC06 screen map 14-2
 - BTC06 Teach Pendant
 - Keyboard Operation 14-6
 - Setup 14-4
 - Build menu
 - Compile 2-51
 - Program management 2-52
 - Save, compile, download 2-51
- C**
- C-0-0001 Language Selection 5-25
 - C-0-0002 Device Address 5-25
 - C-0-0003 Serial Port A Setup 5-27
 - C-0-0004 Serial Port B Setup 5-27
 - C-0-0005 Communication Protocol
 - Selection 5-28
 - C-0-0009 Error Reaction Mode 5-28
 - C-0-0010 System Options 5-29
 - C-0-0012 Serial Port B Device Type 5-32
 - C-0-0013 Serial Port A Mode 5-32
 - C-0-0014 Serial Port B Mode 5-33
 - C-0-0016 Communication Time-out
 - Period 5-33
 - C-0-0017 Serial Port A Password 5-34
 - C-0-0018 Serial Port B Password 5-35
 - C-0-0020 Transmitter Fiber Optic Length
 - 5-35
 - C-0-0021 User Watchdog Timer 5-35
 - C-0-0022 User Watchdog Task ID 5-36
 - C-0-0035 PLC Communication Option
 - (GPP only) 5-36
 - C-0-0042 World Large Increment 5-37
 - C-0-0043 World Small Increment 5-37
 - C-0-0045 World Fast Jog Speed 5-38
 - C-0-0046 World Slow Jog Speed 5-38
 - C-0-0052 Axis Large Increment 5-39
 - C-0-0053 Axis Small Increment 5-39
 - C-0-0055 Axis Fast Jog Velocity 5-40
 - C-0-0056 Axis Slow Jog Velocity 5-41
 - C-0-0080 Maximum Number of Global
 - Integers 5-41
 - C-0-0081 Maximum Number of Global
 - Floats 5-42
 - C-0-0082 Save Global Variables
 - Command 5-42
 - C-0-0083 Save Global Variables Status
 - 5-43
 - C-0-0090 Download Block Size 5-43
 - C-0-0091 Total Program Memory 5-44
 - C-0-0092 Available Program Memory
 - 5-44
 - C-0-0093 Contiguous Program Memory
 - 5-44
 - C-0-0094 Maximum Executable
 - Program Size 5-45
 - C-0-0098 Initialization Delay 5-45
 - C-0-0099 Minimum SERCOS Cycle
 - Time 5-45
 - C-0-0100 Control Firmware Version 5-46
 - C-0-0101 Control Hardware Version 5-46
 - C-0-0102 Control Version Date 5-47
 - C-0-0103 Allowable Drive Address
 - Range 5-47
 - C-0-0104 Bootloader Firmware Version
 - 5-48
 - C-0-0120 Operating Mode 5-48
 - C-0-0121 SERCOS Communication
 - Phase 5-49
 - C-0-0122 Diagnostic Message 5-49
 - C-0-0123 Diagnostic Code 5-50
 - C-0-0124 Extended Diagnostic 5-50
 - C-0-0125 System Timer Value 5-51
 - C-0-0126 Date and Time 5-51
 - C-0-0127 Current Control Temperature
 - 5-52
 - C-0-0128 Elapsed Time Operational
 - Counter 5-52
 - C-0-0142 Card Label String 5-53
 - C-0-0166 Save Built CAM to Flash ID5-
 - 53
 - C-0-0167 Save Built CAM to Flash
 - Command 5-53
 - C-0-0168 Save Built CAM to Flash
 - Status 5-54
 - C-0-0200 Current Load due to Motion
 - 5-55
 - C-0-0201 Peak Load due to Motion 5-55
 - C-0-0202 Current Load due to I/O 5-56
 - C-0-0203 Peak Load due to I/O 5-56
 - C-0-0300 Link Ring Control Word 5-56
 - C-0-0301 Link Ring Primary Fiber Optic
 - Length 5-57
 - C-0-0302 Link Ring Secondary Fiber
 - Optic Length 5-58
 - C-0-0303 Link Ring MDT Error Counter
 - 5-58
 - C-0-0400 Ethernet Card IP Address
 - (GPP only) 5-59
 - C-0-0401 Ethernet Card Subnet Mask
 - (GPP only) 5-59
 - C-0-0402 Ethernet Card Gateway IP
 - Address (GPP only) 5-60
 - C-0-0403 Ethernet Card CIF Network
 - Control (GPP only) 5-60
 - C-0-0404 Ethernet Card Network
 - Access Control (GPP only) 5-61
 - C-0-0405 Ethernet Card Network
 - Password (GPP only) 5-62
 - C-0-0406 CIF Ethernet Card Hardware
 - ID (GPP only) 5-64
 - C-0-0407 CIF Ethernet Card Firmware
 - Version (GPP only) 5-64
 - C-0-0408 CIF Ethernet Driver Version
 - (GPP only) 5-64

- C-0-0522 Init. Task Diagnostic Message 5-65
- C-0-0523 Init. Task Status Message 5-65
- C-0-0530 Init. Task Current Instr.
Pointer 5-65
- C-0-0531 Init Task Current Instruction 5-66
- C-0-0532 Init. Task Pointer at Error. 5-66
- C-0-0533 Init. Task Composite Instr.
Pointer 5-66
- C-0-0535 Init. Task Current Subroutine 5-67
- C-0-0536 Init. Task Stack Variable Data 5-67
- C-0-0801 Pendant Protection Level 1
Password 5-68
- C-0-0802 Pendant Protection Level 2
Password 5-68
- C-0-0803 Pendant User Accessible
Floats Section 5-68
- C-0-0804 Pendant User Accessible
Integers Section 5-69
- C-0-0805 Pendant Start of User
Accessible Registers 5-69
- C-0-0806 Pendant End of User
Accessible Registers 5-70
- C-0-0807 Pendant Password Timeout 5-70
- C-0-0810 TPT Message and Prompt
Control Word 5-70
- C-0-0811 User Task Controlled Menu ID
for TPT 5-73
- C-0-0812 User Task Controlled Task ID
for TPT 5-74
- C-0-0813 User Task Controlled Axis
Number for TPT 5-75
- C-0-0814 TPT Data Transaction Word 5-75
- C-0-0990 Exit to Monitor Prompt 5-77
- C-0-0993 Software Reset for Control 5-78
- C-0-0994 Shutdown Command for Flash
Programming 5-79
- C-0-0996 Clear Program and Data
Memory 5-79
- C-0-0997 Clear Diagnostic Log 5-80
- C-0-2000 List of All Parameters 5-80
- C-0-2001 List of Required Parameters 5-81
- C-0-2002 List of Invalid A-, C- and T-
Parameters 5-81
- C-0-2010 List of SERCOS Devices 5-82
- C-0-2011 List of SERCOS Drives 5-82
- C-0-2012 List of SERCOS I/O Stations 5-82
- C-0-2013 I/O Configuration List 5-83
- C-0-2016 List of Virtual Axes 5-83
- C-0-2017 I/O User Configuration List 5-84
- C-0-2020 Diagnostic Log List 5-84
- C-0-2021 Diagnostic Log Options 5-85
- C-0-2501 Oscilloscope Signal 1 Type 5-86
- C-0-2502 Oscilloscope Signal 2 Type 5-87
- C-0-2503 Oscilloscope Signal 3 Type 5-87
- C-0-2504 Oscilloscope Signal 1 ID
Number 5-87
- C-0-2505 Oscilloscope Signal 2 ID
Number 5-88
- C-0-2506 Oscilloscope Signal 3 ID
Number 5-88
- C-0-2507 Oscilloscope Signal 1 Axis
Number 5-88
- C-0-2508 Oscilloscope Signal 2 Axis
Number 5-89
- C-0-2509 Oscilloscope Signal 3 Axis
Number 5-89
- C-0-2510 Oscilloscope Sampling Rate 5-89
- C-0-2511 Oscilloscope Signal 1 List 5-90
- C-0-2512 Oscilloscope Signal 2 List 5-90, 5-96
- C-0-2513 Oscilloscope Signal 3 List 5-90
- C-0-2514 Oscilloscope Sample Count 5-90
- C-0-2515 Oscilloscope Trigger Post-
count 5-91
- C-0-2516 Oscilloscope Trigger Type 5-91
- C-0-2517 Oscilloscope Trigger ID
Number 5-92
- C-0-2518 Oscilloscope Trigger Axis or
Mask 5-92
- C-0-2519 Oscilloscope Trigger Level or
Mask 5-93
- C-0-2520 Oscilloscope Trigger Mode 5-93
- C-0-2521 Oscilloscope Trigger Source 5-94
- C-0-2522 Oscilloscope Trigger Control
Word 5-94
- C-0-2523 Oscilloscope Trigger Status
Word 5-95
- C-0-2524 Oscilloscope Signal 4 Type 5-96
- C-0-2525 Oscilloscope Signal 4 ID
Number 5-96
- C-0-2526 Oscilloscope Signal 4 Axis
Number 5-96
- C-0-2527 Oscilloscope Signal 4 List 5-96
- C-0-2600 Fieldbus/PLC Mapper (cyclic
channel) To PLC 5-97
- C-0-2601 Fieldbus/PLC Mapper (cyclic
channel) From PLC 5-97
- C-0-2607 Multiplex Control Word 5-98
- C-0-2608 Multiplex Status Word 5-98
- C-0-2611 Fieldbus/PLC Cyclic Channel
Current Number of Misses 5-99
- C-0-2612 Fieldbus/PLC Cyclic Channel
Peak Number of Misses 5-99
- C-0-2613 Fieldbus/PLC Cyclic Channel
Timeout Counter 5-100
- C-0-2630 Fieldbus Slave Device
Address (GPP only) 5-100
- C-0-2631 Fieldbus Parameter/PCP
Channel Length (GPP only) 5-100
- C-0-2632 Fieldbus/PLC Multiplex
Method (GPP only) 5-101
- C-0-2633 Fieldbus Baud Rate
(DeviceNet only) (GPP only) 5-101
- C-0-2635 Fieldbus/PLC Error Reaction 5-102
- C-0-2636 Fieldbus/PLC Word Swap 5-102
- C-0-2637 Fieldbus/PLC Slave Firmware
Version 5-104
- C-0-2638 Fieldbus/PLC Available Cyclic
IN Parameters 5-104
- C-0-2639 Fieldbus/PLC Available Cyclic
OUT Parameters 5-104
- C-0-2640 PLC Connection Options
(GMP only) 5-105
- C-0-2641 PLC Input Register List 5-105
- C-0-2642 PLC Output Register List 5-106
- C-0-2643 PLC Lifecounter Check
Number of Retries 5-106
- C-0-2644 PLC Lifecounter Check
Current Number of Misses 5-107
- C-0-2645 PLC Lifecounter Check
Peak Number of Misses 5-107
- C-0-2646 PLC Lifecounter Check
Number of Timeouts 5-108
- C-0-2647 ISP Function Block Timeout 5-108

- C-0-2651 PLC Register Channel
 - Current Number of Misses 5-108
- C-0-2652 PLC Register Channel
 - Peak Number of Misses 5-109
- C-0-2653 PLC Register Channel
 - Timeout Counter 5-109
- C-0-2901 PLS 1 Start Output Register5-110
- C-0-2902 PLS 1 Start Mask Register5-110
- C-0-2903 PLS 1 Build Command .. 5-111
- C-0-2904 PLS1 Build Status..... 5-111
- C-0-2905 PLS1 Activate Command5-112
- C-0-2906 PLS1 Activate Status..... 5-112
- C-0-2907 PLS1 Error Code 5-113
- C-0-2908 PLS1 Extended Error Code5-113
- C-0-2909 PLS1 Hardware ID..... 5-113
- C-0-2910 PLS1 Software ID 5-114
- C-0-2920 PLS1 Switch On List..... 5-114
- C-0-2921 PLS1 Switch Off List..... 5-114
- C-0-2922 PLS1 Switch Output List 5-115
- C-0-2930 PLS1 Output Master List5-115
- C-0-2931 PLS1 Output Lead Time List5-115
- C-0-2932 PLS1 Output Lag Time List5-116
- C-0-2933 PLS1 Output One Shot List5-116
- C-0-2934 PLS1 Output Mode List.. 5-117
- C-0-2935 PLS1 Output Direction List5-117
- C-0-2936 PLS1 Output Hysteresis List5-118
- C-0-2940 PLS1 Master Type List .. 5-118
- C-0-2941 PLS1 Master Number List5-119
- C-0-2942 PLS1 Master Encoder List5-119
- C-0-2943 PLS1 Master Phase Offset List..... 5-120
- C-0-3000 I/O Mapper Program 5-120
- C-0-3001 I/O Mapper Options 5-121
- C-0-3003 I/O Mapper Total Operations5-122
- C-0-3004 I/O Mapper File Size 5-122
- C-0-3005 I-O Mapper Executable Size5-122
- C-0-3100 Cam Tags..... 5-123
- C-0-3101 CAM Table 1 through C-0-3137 CAM Table 37 5-123
- C-0-3138 CAM Table 38 through C-0-3140 CAM Table 40 5-124
- C-0-3141 CAM Type..... 5-124
- C-0-32x1 PMG # Maximum Allowed Deviation Window 5-125
- C-0-32x2 PMG # List of Axis 5-126
- C-0-32x3 PMG # List of Position Offsets 5-126
- C-0-32x4 PMG # Current Peak Group Deviation..... 5-127
- C-0-32x5 PMG # Maximum Deviation5-128
- C-0-32x6 PMG # Configuration..... 5-129
- Calc icon
 - Control PLS 8-41
 - Drive PLS 8-41
 - Option Card PLS..... 8-42
- Calc2..... 3-22
 - card PLS..... 3-29
 - control PLS 3-30
 - direct addressing 3-24
 - drive PLS 3-30
 - events table 3-27
 - indirect addressing..... 3-24
 - operators 3-23
 - points table 3-26
 - zone table..... 3-28
- Cam 3-31
 - Number 3-33
- CAM Builder..... 2-146
- CAM Indexer..... 2-112
- CAM Indexer Function Block Variables (PJ) 13-24
- CAM Table 1 through C-0-3137 CAM Table 37 5-123
- Cam Tags..... 5-123
- CAM Type..... 5-124
- Card Label String..... 5-53
- Card Parameter Menu..... 14-26
- CIF Ethernet Card Firmware Version (GPP only)..... 5-64
- CIF Ethernet Card Hardware ID (GPP only) 5-64
- CIF Ethernet Driver Version (GPP only) 5-64
- Circle 3-47
- CLC Table Edit Menu..... 14-14
- Clear All Programs (PC) 13-21
- Clear Diagnostics Log..... 5-80
- Clear Program and Data Memory ... 5-79
- Command..... 3-50
- Command Classes/Subclasses
 - Event Tables 13-18
 - Functions..... 13-34
 - I/O Registers 13-34
 - Parameters..... 13-7
 - PID 13-14
 - Program Communication..... 13-20
 - Sequencer Data..... 13-37
 - Variables 13-13
 - Zones 13-46
- Command Data #1..... 5-180
- Command Data #2..... 5-181
- Command Data #3..... 5-181
- Commanded Position..... 5-164
- Commanded Ratio Adjust..... 5-174
- Commanded Velocity..... 5-165
- Commission menu
 - Coordinated Motion 2-66
 - Drive Overview 2-55
 - Fieldbus Mapper..... 2-58
 - I/O Mapper 2-58
 - I/O Setup 2-57
 - PLS 2-59
 - Transfer 2-75
- Communication Errors 13-5
- Communication Protocol Selection . 5-28
- Communication Time-out Period..... 5-33
- Composite Instruction Pointer 5-143
- Configuration Mode..... 5-155
- Connecting Icons 3-4
- Contiguous Program Memory 5-44
- Control Cam Current Slave Adjust (Sph) 5-175
- Control Cam Currently Active..... 5-160
- Control Cam Output Position 5-175
- Control Cam Position Constant (L) 5-160
- Control Cam Programmed Slave Adjust 5-174
- Control Cam Stretch Factor (H) 5-159
- Control Cam/Ratio Master Factor (N)5-159
- Control Cam/Ratio Slave Factor (M)5-159
- Control Firmware Version 5-46
- Control Hardware Version..... 5-46
- Control Menu
 - auto run/hold mode..... 14-22
 - auto step mode..... 14-23
 - manual mode..... 14-24
- Control Parameters..... 5-11
 - BTC06 Teach Pendant 5-14
 - Cam Table..... 5-18, 5-19

- Control Processor Usage Status (GPP only)..... 5-13
 - Fieldbus Interface 5-16, 5-17
 - I/O Mapper 5-18
 - Initialization Task Parameters 5-14
 - Internal System Monitoring 5-14
 - Jogging and Display 5-11
 - Oscilloscope 5-15
 - PLC Interface 5-17
 - Program Management 5-12
 - System Memory (GPP only)..... 5-15
 - System Parameter Lists 5-15
 - System Setup 5-11
 - System Status 5-12
 - Control Phase Adjust Average Velocity 5-171
 - Control Phase Adjust Time Constant 5-172
 - Control PLS
 - ELS Group 8-11
 - Control PLS 8-1
 - configure 8-9
 - direct ASCII instructions 8-2
 - ELS Master 8-11
 - graph limits 8-13
 - mask register 8-12
 - output register 8-12
 - PLS Master configuration for a Control PLS 8-10
 - PLS register assignment for a Control PLS 8-12
 - switch configuration 8-9
 - Control Selection 2-157
 - Control Version Date 5-47
 - CoordArt 3-51
 - Coordinated Jogging 4-8
 - Coordinated Motion 2-66
 - Archive 2-70
 - Jogging 2-67
 - Task limits 2-66
 - Coordinated X Axis 5-132
 - Coordinated Y Axis 5-133
 - Coordinated Z Axis 5-133
 - Create a New Project 2-7
 - Create a New Project from Program and Data on the Control 2-9
 - Current Control Temperature 5-52
 - Current Instruction 5-142
 - Current Instruction Pointer 5-142
 - Current Load due to I/O 5-56
 - Current Load due to Motion 5-55
 - Current Motion Type 5-169
 - Current Phase/ Control Cam Master Offset 5-173
 - Current Subroutine 5-144
 - Current X Position 5-139
 - Current Y Position 5-140
 - Current Z Position 5-140
 - custom list 2-95
 - modify 2-98
 - custom list group
 - create 2-99
 - Cutoff Frequency
 - ELS Real Master... 2-119, 3-77, 3-102
 - Cycle Control Considerations 4-7
- D**
- Data menu
 - CAM indexer 2-112
 - ELS 2-113
 - Events 2-108
 - Parameter Overview 2-87
 - PID 2-116
 - Points 2-110
 - Registers 2-102
 - Registration 2-123
 - Variables 2-107
 - Zones 2-125
 - Date and Time 5-51
 - defragment memory 2-130
 - Device Address 5-25
 - Diagnostic Code 5-50
 - diagnostic log 2-129
 - Diagnostic Log List 5-84
 - Diagnostic Log Options 5-85
 - Diagnostic Menu 14-30
 - error screen 14-31
 - Diagnostic Message 5-49
 - Diagnostics
 - Show program flow 2-143
 - Diagnostics menu
 - Oscilloscope 2-134
 - System 2-127
 - Tasks 2-132
 - Toggle Breakpoint 2-144
 - direction
 - negative 8-28
 - positive 8-28
 - positive/negative 8-28
 - Directive
 - EVENT/END 15-32
 - EVENT/START 15-32
 - PLS/INIT 15-61
 - TASK/END 15-70, 15-71
 - VAR/INIT 15-72
 - Directives 15-1
 - DEFINE 15-24
 - EQU (Equate) 15-30
 - Double Link Ring 11-4
 - Double Ring 11-4
 - Download Block Size 5-43, 13-21
 - Drive Overview 2-55
 - Drive Parameter Menu 14-28
 - Drive PLS 8-1, 8-3
 - configure 8-14
 - DIAX specifications 8-3
 - drive parameters 8-4
 - ECODRIVE specifications 8-3
 - graph limits 8-17
 - output register 8-17
 - PLS Master configuration for a Drive PLS 8-16
 - PLS register assignment for a Drive PLS 8-16
 - switch configuration 8-15
 - Drive PLS Register 5-156
 - Drive Status Word 5-5
 - Drive Telegram (AT) 2-88, 5-4
- E**
- Edit labels 2-26
 - Bit labels 2-41
 - I/O bit function labels 2-42
 - Register labels 2-40
 - Variable labels 2-39
 - Edit menu
 - Clear Icon Flow 2-20
 - Copy (Ctrl+C) 2-19
 - Cut 2-19
 - Delete (Del) 2-19
 - Edit labels 2-38
 - Find 2-24
 - Find next 2-25
 - Paste (Ctrl+V) 2-19
 - Replace 2-25
 - Select all 2-19
 - Undo (Ctrl+Z) 2-19
 - VM Data 2-26
 - Edit Menu
 - Labels
 - User Labels 2-39
 - Elapsed Time Operational Counter . 5-52

Electronic Line Shaft	5-170
ELS	2-113
ELS Groups	2-114
ELS System Masters	2-114
Virtual Masters	2-115
ELS Functions	13-34
ELS Group Master	3-76
ELS Groups	2-114, 3-62
ELS Options	5-175
ELS Programmed Phase Offset	5-171
ELS System Master	
Velocity Rounding	3-73
ELS System Masters	2-114
ELSAAdj1	3-61
ELSGrp1	3-62
ELSMoDe	3-79
ELSMstr1	3-73
End of Message	13-6
EQU (Equate)	15-30
Erase All Forcing Masks (RE)	13-36
Erasing (Deleting) a Program (PE)	13-23
error levels	12-3
fatal error	12-4
non-fatal error	12-4
error reaction	
configurable	12-5
drive	12-12
motion type	12-10
Error Reaction Mode	5-28
error type	
drive error	12-2
task error	12-2
error types	12-2
system error	12-2
Ethernet Card CIF Network Control	
(GPP only)	5-60
Ethernet Card Gateway IP Address	
(GPP only)	5-60
Ethernet Card IP Address (GPP only)	5-59
Ethernet Card Network Access Control	
(GPP only)	5-61
Ethernet Card Network Password (GPP	
only)	5-62
Ethernet Card Subnet Mask (GPP only)	
.....	5-59
Ethernet settings	2-162
Event Table	14-16
Event Table Data	13-18
Event Table Data, Row Format	13-19
Event Tables	13-18
Event2	3-80
Events	2-108
events table	
data structure	3-27
Executing a Download (PD)	13-22
Executing an Upload (PD)	13-22
Exit to Monitor Prompt	5-77
Extended Diagnostic	5-50
F	
F1 program menu	
F4 editing sequencer	14-12
F2 Table Edit Menu	
Absolute Point Table	14-14
Event Table Menu	14-16
Floating Table	14-17
Global Floating Table	14-18
Global Integer Table	14-18
Integer Table Menu	14-17
Relative Point Table	14-15
F3 Jog Menu	14-19
F4 Control Menu	
control menu	
auto run/hold mode	14-22
auto step mode	14-23
manual mode	14-24
F5 Register I/O Menu	14-24
F6 Parameter Menu	14-25
F6 Security Menu	14-29
F8 Diagnostic Menu	14-30
Feedback Data #1	5-181
Feedback Data #2	5-181
Feedback Position	5-165
Feedback Velocity	5-166
Fieldbus Baud Rate (DeviceNet only)	
(GPP only)	5-101
Fieldbus Mapper	2-58
Fieldbus Parameter/PCP Channel	
Length (GPP only)	5-100
Fieldbus Slave Device Address (GPP	
only)	5-100
Fieldbus/PLC Available Cyclic IN	
Parameters	5-104
Fieldbus/PLC Available Cyclic OUT	
Parameters	5-104
Fieldbus/PLC Cyclic Channel	
Current Number of Misses	5-99
Peak Number of Misses	5-99
Timeout Counter	5-100
Fieldbus/PLC Error Reaction	5-102
Fieldbus/PLC Mapper (cyclic channel)	
From PLC	5-97
Fieldbus/PLC Mapper (cyclic channel)	
To PLC	5-97
Fieldbus/PLC Multiplex Method (GPP	
only)	5-101
Fieldbus/PLC Slave Firmware Version	5-104
Fieldbus/PLC Word Swap	5-102
File menu	
Close	2-10
Exit	2-18
New	2-7
Offline	2-11
Online	2-11
Open	2-10
Print project data	2-17
Recent programs	2-18
Recent projects	2-18
Sample programs	2-18
Save All	2-10
Save As	2-10
Save Program	2-10
Save Project As	2-10
Find, Find Next, Replace	2-24
Finish1	3-87
Floating Table Menu	14-18
Format of Data Sent to VisualMotion	13-7
Function Classes	
K - ELS Functions	13-32
S Class	13-33
Function Edit Menu	14-13
Function List	14-13
G	
Getting started	2-169
Global Floating Table	14-18
Global Integer Table	14-18
Go1	3-88
GPP Flash Compression (PK)	13-25
H	
Help menu	
About VisualMotion	2-170
Getting started	2-169
Registered Help	2-170
Search	2-169
Home	3-88

F

F1 program menu	
F4 editing sequencer	14-12
F2 Table Edit Menu	
Absolute Point Table	14-14
Event Table Menu	14-16
Floating Table	14-17
Global Floating Table	14-18
Global Integer Table	14-18
Integer Table Menu	14-17
Relative Point Table	14-15
F3 Jog Menu	14-19
F4 Control Menu	
control menu	
auto run/hold mode	14-22
auto step mode	14-23

G

Getting started	2-169
Global Floating Table	14-18
Global Integer Table	14-18
Go1	3-88
GPP Flash Compression (PK)	13-25

H

Help menu	
About VisualMotion	2-170
Getting started	2-169
Registered Help	2-170
Search	2-169
Home	3-88

- hysteresis.....8-27
- I**
- I/O Binary Forcing State (RS)..... 13-37
- I/O configuration
 - edit menu.....6-3
 - add I/O module.....6-4
 - add SERCOS device.....6-4
 - auto-assign registers.....6-6
 - modify I/O module.....6-6
 - modify I/O station (drive).....6-6
 - RECO error reaction.....6-7
 - remove I/O module.....6-6
 - remove I/O station (drive).....6-6
 - help menu.....6-9, 7-7
 - importing.....6-11, 7-20
 - offline mode.....6-10
 - service mode.....6-12, 7-21
 - settings menu
 - control selection.....6-8
 - view menu
 - status bar.....6-7
 - toolbar.....6-7
- I/O Configuration
 - view menu
 - display register usage.....6-8
- I/O Configuration List.....5-83
- I/O configuration tool.....6-1
- I/O Forcing Selection (RF)..... 13-36
- I/O Forcing State Change (RC)..... 13-35
- I/O Mapper.....2-58
 - Binary Shift Register.....7-17
 - Card Selection Setup.....7-6
 - Check rungs and convert to Boolean strings.....7-9
 - coil.....7-14
 - Latch.....7-16
 - normal coil.....7-14
 - One Shot.....7-15
 - contacts.....7-13
 - Counter.....7-19
 - cross reference.....7-8
 - Delete row.....7-8
 - Forcing icons.....7-10
 - forcing options.....7-6
 - input logic.....7-13
 - Insert row.....7-8
 - Ladder logic icons.....7-9
 - menu selection.....7-4
 - Edit menu.....7-5
 - File menu.....7-4
 - Setting menu.....7-6
 - View menu.....7-5
 - Window menu.....7-7
 - open default *.iom file.....7-21
 - Forcing.....7-10
 - output logic functions.....7-14
 - Properties.....7-13
 - contact selection.....7-14
 - contact setup.....7-13
 - register and bit.....7-13
 - scan time.....7-6
 - specifications
 - Boolean strings.....7-3
 - total operations.....7-3
 - Timer.....7-18
 - Undo.....7-8
 - Forcing.....7-12
- I/O Mapper Executable Size.....5-122
- I/O Mapper File Size.....5-122
- I/O Mapper Options.....5-121
- I/O Mapper Program.....5-120
- I/O Mapper Total Operations.....5-122
- I/O Register Access (RB), (RX), (RD)13-35
- I/O RegisterAccess
 - I/O Register Read..... 13-35
 - I/O Register Write..... 13-35
- I/O Setup.....2-57
- I/O User Configuration List.....5-84
- I_O.....3-89
- icon palette.....3-1
- Icon Programming.....3-1
- Icons
 - Accel.....3-13
 - Axis2.....3-14
 - Branch.....3-20
 - Calc2.....3-22
 - Cam.....3-31
 - CamBld2.....3-33
 - Circle.....3-47
 - Command.....3-50
 - Connecting Icons.....3-4
 - CoordArt.....3-51
 - ELSAAdj1.....3-61
 - ELSGrp1.....3-62
 - ELSMde.....3-79
 - ELSMstr1.....3-73
 - Event2.....3-80
 - Finish.....3-7
 - Finish1.....3-87
 - Go1.....3-88
 - Home.....3-88
 - I_O.....3-89
 - Join.....3-90
 - Joint.....3-90
 - Line.....3-91
 - Move2.....3-91
 - Msg1.....3-93
 - Param.....3-94
 - ParamBit.....3-96
 - Path.....3-98
 - PID1.....3-100
 - Position.....3-104
 - PrmBit.....3-104
 - PrmInt.....3-105
 - Ratio.....3-107
 - Reg.....3-108
 - Scissor.....3-108
 - Start.....3-7
 - Start1.....3-109
 - Stop1.....3-113
 - Sub1.....3-114
 - Veloc.....3-117
 - VM.....3-117
 - Wait
 - Axis at Position.....3-122
 - Axis in Position.....3-122
 - Coordinated State.....3-122
 - Wait1.....3-122
- Init. Task Composite Instr. Pointer..5-66
- Init. Task Current Instr. Pointer.....5-65
- Init. Task Current Instruction.....5-66
- Init. Task Current Subroutine.....5-67
- Init. Task Diagnostic Message.....5-65
- Init. Task Pointer at Error.....5-66
- Init. Task Stack Variable Data.....5-67
- Init. Task Status Message.....5-65
- Initialization Delay.....5-45
- Initializing a Download (PW).....13-29
- Initializing an Upload (PR).....13-27
- Input/Output Registers.....13-34
- Insert menu
 - Event function.....2-50
 - Subroutine.....2-49
- Instruction Format.....15-2
- Instruction Pointer at Error.....5-143
- ISP Function Block Timeout.....5-108
- J**
- Jog Acceleration.....5-157
- Jog Fine Adjustments.....14-21

Jogging2-153
 Join3-90
 Joint3-90

K

keywords2-29
 Kinematic Number5-132
 Kinematic Value 1 through T-0-0059
 Kinematic Value 105-137
 Kinematics10-1
 Associated Task Parameters10-1
 Kinematic 110-13
 Kinematic 1010-17
 Kinematic 1210-18
 Kinematic 1610-19
 Kinematic 1810-19
 Kinematic 210-13
 Kinematic 310-14
 Kinematic 410-14
 Kinematic 510-15
 Kinematic 8 with Velocity Precision
 10-15
 Kinematic 910-16
 Normal Case10-2
 Special Case10-2

L

labels2-26
 lag time8-26
 Language2-159, 2-167
 Language Selection5-25
 Last Active Event Number5-145
 lead time8-26
 Line3-91
 Linear Position Units5-154
 Link Ring11-1
 Link Ring Control Word5-56
 Link Ring Master11-1
 Link Ring MDT Error Counter5-58
 Link Ring Primary Fiber Optic Length5-
 57
 Link Ring Secondary Fiber Optic Length
 5-58
 Link ring settings2-164
 Link Ring Slave11-1
 List Control Resident Programs (PH)13-
 24
 List of All Parameters 5-80, 5-146, 5-184
 List of Event Function Marks (PF) .13-23
 List of Invalid A-, C- and T- Parameters
 5-81
 List of Required Parameters5-81, 5-146,
 5-184
 List of SERCOS Devices5-82
 List of SERCOS Drives5-82
 List of SERCOS I/O Stations5-82
 List of Virtual Axes5-83
 List Variable Labels (PV)13-28
 Look Ahead Distance5-135

M

mask register
 Control PLS8-13
 Master Axis for Ratio Function5-159
 Master Control Word5-6
 Master Data Telegram (MDT)2-88, 5-4,
 5-5
 Maximum Acceleration5-134, 5-157
 Maximum Deceleration5-135, 5-157
 Maximum Executable Program Size5-45
 Maximum Jog Increment5-136, 5-158
 Maximum Jog Velocity5-137, 5-158

Maximum Number of Global Floats .5-42
 Maximum Number of Global Integers5-
 41
 Maximum Path Speed5-134
 Maximum Velocity5-156
 MDT Multiplex Ident List (DKC 2.3 only)
 5-183
 MDT Multiplex Selection List (DKC 2.3
 only)5-182
 Menu Map (F1-F4)14-2
 Menu Map (F5-F8)14-3
 Mfg. Class 3 Status Word5-168
 Minimum SERCOS Cycle Time5-45
 mode
 lag time8-27
 PT8-27
 Move23-91
 Msg13-93
 Multiplex5-9
 Multiplex Control Word5-98
 Multiplex Status Word5-98
 Multiplexing Parameters5-182

N

Name Text Subclass13-12
 Network settings2-162
 password2-162
 Numeric Data Formats13-6

O

Open Existing Icon Program2-9
 Open Existing project2-9
 Operating Mode5-48
 Operators3-23
 Option Card PLS8-1, 8-5
 configure8-18
 ELS Group8-21
 ELS Master8-21
 graph limits8-23
 mask register8-23
 output configuration8-25
 output register8-23
 parameters8-5
 PLS Master configuration8-20
 PLS outputs8-24
 PLS register assignment8-22
 specifications8-5
 switch configuration8-19
 Optional Command ID #15-178
 Optional Command ID #25-179
 Optional Command ID #35-179
 Optional Feedback ID #15-179
 Optional Feedback ID #25-180
 Options2-167
 oscilloscope2-134
 file menu2-134
 measuring trace signals2-142
 options menu2-138
 signal selection2-136
 source menu2-135
 time controls2-142
 timing2-135
 Oscilloscope2-134
 Oscilloscope Sample Count5-90
 Oscilloscope Sampling Rate5-89
 Oscilloscope Signal 1 Axis Number 5-88
 Oscilloscope Signal 1 ID Number ...5-87
 Oscilloscope Signal 1 List5-90
 Oscilloscope Signal 1 Type5-86
 Oscilloscope Signal 2 Axis Number 5-89
 Oscilloscope Signal 2 ID Number ...5-88
 Oscilloscope Signal 2 List5-90, 5-96
 Oscilloscope Signal 2 Type5-87
 Oscilloscope Signal 3 Axis Number 5-89

- Oscilloscope Signal 3 ID Number 5-88
 - Oscilloscope Signal 3 List 5-90
 - Oscilloscope Signal 3 Type 5-87
 - Oscilloscope Signal 4 Axis Number. 5-96
 - Oscilloscope Signal 4 ID Number 5-96
 - Oscilloscope Signal 4 List 5-96
 - Oscilloscope Signal 4 Type 5-96
 - Oscilloscope Trigger Axis or Mask .. 5-92
 - Oscilloscope Trigger Control Word.. 5-94
 - Oscilloscope Trigger ID Number 5-92
 - Oscilloscope Trigger Level or Mask. 5-93
 - Oscilloscope Trigger Mode 5-93
 - Oscilloscope Trigger Post-count..... 5-91
 - Oscilloscope Trigger Source 5-94
 - Oscilloscope Trigger Status Word.. 5-95
 - Oscilloscope Trigger Type 5-91
 - output configuration
 - direction 8-27
 - mode 8-27
 - output data 8-26
 - output index 8-28
 - output PLS Master 8-29
 - output switches 8-29
 - output data
 - hysteresis 8-27
 - lag time 8-26
 - lead time 8-26
 - PT (Time Duration) 8-26
 - output switches
 - edit a switch's on/off position 8-30
- P**
- P Data 13-13
 - palette 3-1
 - Param 3-94
 - ParamBit 3-96
 - parameter access..... 2-88
 - Parameter Data Subclass 13-12
 - Parameter Identification 5-2
 - parameter list 2-92
 - Parameter List Block Transfer 13-8
 - Parameter Lists 13-11
 - Parameter Lists Subclasses 13-11
 - Parameter Menu 14-25
 - Parameter Overview 2-87
 - system configuration 2-101
 - Parameter Transfer Commands 5-3
 - Parameters 5-1
 - Parameters List 5-11
 - X10 port settings 2-160
 - X16 port settings 2-161
 - Password 2-160, 2-161, 2-166
 - Path 3-98
 - Path Smoothing Filter Constant..... 5-137
 - Peak Load due to I/O 5-56
 - Peak Load due to Motion..... 5-55
 - Pendant End of User Accessible
 - Registers 5-70
 - Pendant Password Timeout 5-70
 - Pendant Protection Level 1 Password5-68
 - Pendant Protection Level 2 Password5-68
 - Pendant Security 2-69
 - Pendant Start of User Accessible
 - Registers 5-69
 - Pendant User Accessible Floats Section 5-68
 - Pendant User Accessible Integers
 - Section..... 5-69
 - Phase Offset Velocity Feedback .. 5-172
 - PID 2-116, 13-14
 - PID1 3-100
 - PLC Communication Option (GPP only)
 - 5-36
 - PLC Connection Options (GMP only)5-105
 - PLC Input Register List 5-105
 - PLC Lifecounter Check
 - Current Number of Misses 5-107
 - Number of Retries 5-106
 - Number of Timeouts 5-108
 - Peak Number of Misses 5-107
 - PLC Output Register List 5-106
 - PLC Register Channel
 - Current Number of Misses 5-108
 - Peak Number of Misses 5-109
 - Timeout Counter 5-109
 - PLC settings 2-165
 - PLS 2-59
 - data structure 3-29
 - offline editing 8-31
 - online editing 8-31
 - project mode editing 8-31
 - save..... 8-36
 - PLS 1 Build Command..... 5-111
 - PLS 1 Start Mask Register..... 5-110
 - PLS 1 Start Output Register..... 5-110
 - PLS ASCII Protocol 13-42
 - PLS configuration
 - Calc icon usage..... 8-40
 - download 8-37
 - edit
 - service mode..... 8-34
 - import in project mode 8-33
 - monitor status..... 8-38
 - save and download..... 8-36
 - upload in project mode 8-38
 - upload in service mode..... 8-38
 - PLS Message in service mode..... 8-40
 - PLS object 8-1
 - PLS tool
 - communication modes 8-7
 - PLS1 Activate Command 5-112
 - PLS1 Activate Status 5-112
 - PLS1 Build Status..... 5-111
 - PLS1 Error Code..... 5-113
 - PLS1 Extended Error Code..... 5-113
 - PLS1 Hardware ID 5-113
 - PLS1 Master Encoder List 5-119
 - PLS1 Master Number List 5-119
 - PLS1 Master Phase Offset List 5-120
 - PLS1 Master Type List..... 5-118
 - PLS1 Output Direction List..... 5-117
 - PLS1 Output Hysteresis List 5-118
 - PLS1 Output Lag Time List 5-116
 - PLS1 Output Lead Time List 5-115
 - PLS1 Output Master List..... 5-115
 - PLS1 Output Mode List..... 5-117
 - PLS1 Output One Shot List..... 5-116
 - PLS1 Software ID 5-114
 - PLS1 Switch Off List 5-114
 - PLS1 Switch On List 5-114
 - PLS1 Switch Output List 5-115
 - PMG # Configuration 5-129
 - PMG # Current Peak Group Deviation5-127
 - PMG # List of Axis 5-126
 - PMG # List of Position Offsets 5-126
 - PMG # Maximum Allowed Deviation
 - Window 5-125
 - PMG # Maximum Deviation 5-128
 - Point Table Data, Row Format 13-17
 - Point Tables..... 13-15
 - Point Table Data..... 13-16
 - Point Table Data, Row Format.. 13-17
 - Points 2-110
 - Position 3-104
 - Position Monitoring Group Parameters5-125
 - Primary Ring 11-2, 11-4
 - PrmBit 3-104
 - PrmInt..... 3-105

- Probe 1 Negative Captured Value 5-178
 Probe 1 Positive Captured Value...5-178
 Probe 2 Negative Captured Value...5-178
 Probe 2 Positive Captured Value...5-178
 Probe Configuration Status5-177
 Program management
 Activate2-53
 Clear all2-53
 Data transfer2-53
 Delete2-53
 Download.....2-53
 Upload2-53
 Program Menu14-11
 Programmed Acceleration.....5-166
 Programmed Ratio Adjust5-170
 Programmed Velocity.....5-165
 Project File (vmj)2-8
 project navigator.....2-3
 Protocol.....2-159
 PT (Time Duration).....8-26
- ## R
- Ratio3-107
 Ratio Adjust Step Rate.....5-173
 Ratio Mode Encoder Type.....5-161
 Ratio Mode Options.....5-162
 Ratio Mode Step Rate5-161
 Reading Data from VisualMotion13-5
 Reference Options5-154
 Reg3-108
 Register
 priority.....2-102
 Register 027: Initialization Task Control
 4-16
 Register 028: Initialization Task Status4-
 16
 Register Labels, Bit Labels (RT)....13-37
 Register Menu14-24
 Registered Tools2-157
 Registers.....2-102, 4-1
 Reg. 001 - System Control.....4-2
 Reg. 002-005 - Task Control.....4-4
 Reg. 006 - System Diagnostic Code4-
 7
 Reg. 007-010 - Task Jog Control ..4-8
 Reg. 011-018 - Axis Jog Control .4-10
 Reg. 019 – Fieldbus/PLC Status .4-11
 Reg. 020 – Fieldbus/PLC Diagnostics
 4-13
 Reg. 021 - System Status4-14
 Reg. 022-025 - Task Status4-15
 Reg. 031-038 - Axis Status4-17
 Reg. 040 – Link Ring Status4-19
 Reg. 041 – Link Ring Data 14-20
 Reg. 042 – Link Ring Data 24-21
 Reg. 050 – Ethernet Status.....4-21
 Reg. 051 – Standard Message Count
 4-22
 Reg. 052 – Cyphered Message Count
 4-22
 Reg. 053 – Invalid Protocol Count4-
 22
 Reg. 054 – SIS Message Count..4-23
 Reg. 086 – PMG Control.....4-23
 Reg. 087 – PMG Status4-24
 Reg. 088 and 089 - Task A Extend
 Event Control.....4-24
 Reg. 090 and 091 - Latch and
 Unlatch.....4-25
 Reg. 092-094 - Mask Pendant Key
 Functionality4-25
 Reg. 095-097 - BTC06 Teach
 Pendant Status.....4-25
 Reg. 098 - Pendant Control Task A,
 B.....4-26
 Reg. 099 - Pendant Control Task C,
 D.....4-27
 Reg. 140 – ELS Master Control...4-27
 Reg. 141 – ELS Master Status ...4-29
 Reg. 150 and 151 - Virtual Master 1
 and 2 Control4-30
 Reg. 152-159 - ELS Group Control4-
 33
 Reg. 197 – Coordinated Articulation
 Synch Mode Control.....4-35
 Reg. 198 – Coordinated Articulation
 Local Mode Control4-37
 Reg. 209-240 - Axis Jog Control .4-10
 Reg. 241 and 242 Virtual Master 1
 and 2 Status.....4-38
 Reg. 243-250 - ELS Groups 1- 8
 Status4-39
 Reg. 288 – Coordinated Articulation
 Synch Mode Status4-41
 Reg. 289 – Coordinated Articulation
 Local Mode Status4-42
 Reg. 309-340 - Axis Status.....4-17
 Registration.....2-123, 3-41
 Registration Block Information (PM)13-
 26
 registration error.....3-42
 Relative Phase Offset Distance
 Remaining5-173
 relative points table
 data structure3-26
 Relative Table.....14-15
 Request Currently Active Program (PA)
 13-21
 Request Name of Program (PN) ...13-27
 Robot Jog Menu.....14-19
- ## S
- Save Built CAM to Flash Command 5-53
 Save Built CAM to Flash ID.....5-53
 Save Built CAM to Flash Status5-54
 Save Global Variables Command ...5-42
 Save Global Variables Status5-43
 Scissor.....3-108
 scissor icon.....3-5, 3-91
 Search2-169
 Secondary Ring11-4
 Security Menu14-29
 Segment Status5-139
 Selective Table Transfer Between
 Programs (PT).....13-28
 Sequence Edit Menu.....14-12
 Sequence List Menu14-12
 Sequencer Data13-37
 L - Sequence List Class.....13-38
 Q - Sequence Table Class.....13-39
 Sequencer Information.....5-145
 Sequencer/Subroutine Related
 Subclasses.....13-31
 Sequencer/Subroutine Related
 Subclasses (PI, PL, PQ, PS)....13-31
 SERCOS Communication Phase....5-49
 SERCOS Control Word.....5-167
 SERCOS Drive Telegram Utility.....5-4
 SERCOS Parameter Sets13-7
 SERCOS settings2-163
 baud rate2-163
 cycle time2-163
 SERCOS Status Word5-167
 SERCOS Telegram Tool.....2-88
 Serial Port A Mode.....5-32
 Serial Port A Password5-34
 Serial Port A Setup5-27
 Serial Port B Device Type5-32
 Serial Port B Mode.....5-33
 Serial Port B Password5-35
 Serial Port B Setup5-27

- Set Current I/O State with Mask (RM)13-36
 - Setting password.....2-166
 - Show Program Flow2-143
 - Shutdown Command for Flash Programming5-79
 - Single Ring.....11-2
 - Slip monitoring setup.....2-116
 - Software Reset for Control5-78
 - Stack Variable Data5-144
 - Start
 - Password Protection3-111
 - Start13-109
 - Step List14-12
 - Step Table Edit Menu.....14-13
 - Stop13-113
 - Sub13-114
 - Subroutine Breakpoint.....5-145
 - System2-127
 - System Control Parameters5-25
 - System Options.....5-29
 - System Timer Value.....5-51
- T**
- T Label Text13-13
 - T-0-0001 Task Motion Type5-130
 - T-0-0002 Task Options.....5-130
 - T-0-0005 World Position Units.....5-132
 - T-0-0010 Kinematic Number5-132
 - T-0-0011 Coordinated X Axis5-132
 - T-0-0012 Coordinated Y Axis5-133
 - T-0-0013 Coordinated Z Axis.....5-133
 - T-0-0020 Maximum Path Speed...5-134
 - T-0-0021 Maximum Acceleration...5-134
 - T-0-0022 Maximum Deceleration ..5-135
 - T-0-0023 Look Ahead Distance.....5-135
 - T-0-0024 Velocity Override.....5-136
 - T-0-0025 Maximum Jog Increment5-136
 - T-0-0026 Maximum Jog Velocity ..5-137
 - T-0-0027 Path Smoothing Filter
 - Constant5-137
 - T-0-0050 Kinematic Value 1 through T-0-0059 Kinematic Value 10.....5-137
 - T-0-0100 Target Point Number.....5-138
 - T-0-0101 Segment Status5-139
 - T-0-0111 Current X Position.....5-139
 - T-0-0112 Current Y Position.....5-140
 - T-0-0113 Current Z Position5-140
 - T-0-0120 Task Operating Mode.....5-141
 - T-0-0122 Task Diagnostic Message5-141
 - T-0-0123 Task Status Message5-141
 - T-0-0130 Current Instruction Pointer5-142
 - T-0-0131 Current Instruction5-142
 - T-0-0132 Instruction Pointer at Error5-143
 - T-0-0133 Composite Instruction Pointer5-143
 - T-0-0135 Current Subroutine5-144
 - T-0-0136 Stack Variable Data5-144
 - T-0-0137 Subroutine Breakpoint...5-145
 - T-0-0138 Sequencer Information...5-145
 - T-0-2000 List of All Parameters.....5-146
 - T-0-2001 List of Required Parameters5-146
 - Target Point Number5-138
 - Target Position5-164
 - Task Assignment5-147
 - Task Diagnostic Message5-141
 - Task Jog Control Registers
 - Coordinated Jogging:4-8
 - Task Motion Type.....5-130
 - Task Operating Mode5-141
 - task options
 - ignore other task and drive errors12-9
 - run task during error12-8
 - shutdown other tasks on task error12-9
 - Task Options5-130
 - Task Parameter Lists5-146
 - Task Parameter Menu14-27
 - Task Parameters5-20
 - Coordinated Motion5-20
 - Coordinated Motion Status5-21
 - Task Parameter Lists5-21
 - Task Setup5-20
 - Task Status5-21
 - T-0-0200 Last Active Event Number.....5-145
 - Task Status Message5-141
 - Tasks.....2-132
 - Teach Pendant screens14-1
 - Teaching Points.....14-20
 - Text Language Programming
 - Directives15-1
 - VisualMotion Textual Instructions Instruction Format.....15-2
 - Textual Language Programming
 - AXIS/EVENT15-4
 - AXIS/HOME15-5
 - AXIS/INITIALIZE15-6
 - AXIS/MOVE.....15-7
 - AXIS/RATIO15-8
 - AXIS/SPINDLE15-9
 - AXIS/START15-10
 - AXIS/STOP15-11
 - AXIS/WAIT15-12
 - CALL15-13
 - CAM/ACTIVATE.....15-14
 - CAM/ADJUST15-15
 - CAM/BUILD15-16
 - CAM/INDEX15-18
 - CAM/STATUS15-19
 - CAPTURE/ENABLE15-20
 - CAPTURE/SETUP15-21
 - DATA/SIZE15-23
 - DEFINE15-24
 - DELAY15-25
 - ELS/ADJUST.....15-25
 - ELS/GROUPM15-27
 - ELS/GROUPS15-28
 - ELS/MASTERS15-30
 - ELS/MODE15-29
 - EQU (Equate)15-30
 - EVENT/DONE15-31
 - EVENT/END.....15-32
 - EVENT/START.....15-32
 - EVENT/TRIGGER15-32
 - EVENT/WAIT15-33
 - FUNCTION/ARG15-34
 - FUNCTION/END15-35
 - FUNCTION/START15-35
 - GOSUB15-36
 - GOTO.....15-37
 - IF (If-Else-Endif)15-38
 - KINEMATIC.....15-40
 - LOCAL/VARIABLE15-41
 - MESSAGE/DIAG15-42
 - MESSAGE/STATUS.....15-43
 - MOVE/CIRCLE.....15-44
 - MOVE/JOINT15-45
 - MOVE/LINE.....15-46
 - PARAMETER/BIT15-47
 - PARAMETER/GET15-48
 - PARAMETER/INIT15-49
 - PARAMETER/SET15-50
 - PATH/ABORT15-51
 - PATH/POSITION.....15-52
 - PATH/RESUME15-53
 - PATH/STOP15-53
 - PATH/WAIT.....15-54
 - PID/CONFIG15-55
 - PLC/CLEAR15-57

- PLC/READ 15-57
 - PLC/SET 15-58
 - PLC/TEST 15-59
 - PLC/WAIT 15-60
 - PLC/WRITE 15-60
 - PLS/INIT 15-61
 - REGISTRATION 15-63
 - RETURN 15-64
 - ROBOT/ORIGIN 15-64
 - ROBOT/TOOL 15-65
 - ROTARY/EVENT 15-65
 - SEQ/LIST 15-67
 - SEQ/STEP 15-68
 - SEQUENCER 15-66
 - TASK/AXES 15-69
 - TASK/END 15-70, 15-71
 - VAR/INIT 15-72
 - VIRTUAL/MASTER 15-73
 - Tools menu
 - CAM Builder 2-146
 - Control Selection 2-157
 - Control Settings 2-159
 - Jogging 2-153
 - Options 2-167
 - Registered Tools 2-157
 - Torque Feedback (cyclic) 5-169
 - Torque Mode Commanded Torque 5-169
 - Total Program Memory 5-44
 - TPT Data Transaction Word 5-75
 - TPT Message and Prompt Control Word 5-70
 - Transfer
 - CAM 2-76
 - Events 2-77
 - Fieldbus Mapper 2-78
 - Floats 2-79
 - Global Floats 2-79
 - Global Integers 2-80
 - I/O Mapper 2-82
 - I/O Setup 2-82
 - Integers 2-81
 - Parameters 2-83
 - PLS 2-84
 - Points 2-85
 - Zones 2-86
 - Transfer Tables Between Programs (PX) 13-30
 - Transmitter Fiber Optic Length 5-35
 - Type of Positioning 5-147
- ## U
- Units Text Subclass 13-12
 - Upper Limit, L: Lower Limit Subclasses 13-12
 - User Labels 2-39
 - User Program Header Record 13-20
 - User Program Variables 13-13
 - User Task Controlled Axis Number for TPT 5-75
 - User Task Controlled Menu ID for TPT5-73
 - User Task Controlled Task ID for TPT5-74
 - User Watchdog Task ID 5-36
 - User Watchdog Timer 5-35
- ## V
- Variables 2-107
 - Veloc 3-117
 - Velocity Override 5-136
 - View and Edit Control Data in "Service" Mode 2-9
 - View menu
 - Event functions 2-46
 - Function comment 2-49
 - Icon captions 2-49
 - Icon comments 2-49
 - Icon palette 2-48
 - Subroutines 2-44
 - Task 2-44
 - Zoom out F6 2-47
 - Virtual Master 4-30
 - Virtual Masters 2-115
 - VisualMotion and Drive Parameters and Subclasses 13-7
 - VisualMotion menus
 - Build menu 2-51
 - Commission menu 2-55
 - Data menu 2-87
 - Diagnostics menu 2-126
 - Edit menu 2-19
 - Help menu 2-169
 - Insert menu 2-49
 - Tools menu 2-146
 - View menu 2-43
 - Window menu 2-168
 - VisualMotion System Overview 1-1
 - VisualMotion User's Program
 - Communication
 - I/O Binary Forcing State (RS) ... 13-37
 - VisualMotion's User Program
 - Communication 13-20
 - Activating a Program (PA) 13-21
 - CAM Indexer Function Block
 - Variables (PJ) 13-24
 - Clear All Programs (PC) 13-21
 - Download Block Size 13-21
 - Erase All Forcing Masks (RE) ... 13-36
 - Erasing (Deleting) a Program (PE) 13-23
 - Executing a Download (PD) 13-22
 - Executing an Upload (PD) 13-22
 - GPP Flash Compression (PK) .. 13-25
 - I/O Forcing Selection (RF) 13-36
 - I/O Forcing State Change (RC) 13-35
 - I/O Register Access (RB), (RX), (RD) 13-35
 - Initializing a Download (PW) 13-29
 - Initializing an Upload (PR) 13-27
 - Input/Output Registers 13-34
 - List Control Resident Programs (PH) 13-24
 - List of Event Function Marks (PF) 13-23
 - List Variable Labels (PV) 13-28
 - Register Labels, Bit Labels (RT) 13-37
 - Registration Block Information (PM) 13-26
 - Request Currently Active Program (PA) 13-21
 - Request Name of Program (PN) 13-27
 - Selective Table Transfer Between Programs (PT) 13-28
 - Sequencer/Subroutine Related Subclasses (PI, PL, PQ, PS) 13-31
 - Set Current I/O State with Mask (RM) 13-36
 - Transfer Tables Between Programs (PX) 13-30
 - User Program Header Record .. 13-20
 - VM 3-117
 - VM Data 2-26
- ## W
- Wait1 3-122
 - World Fast Jog Speed 5-38
 - World Large Increment 5-37
 - World Position Units 5-132
 - World Slow Jog Speed 5-38
 - World Small Increment 5-37

Writing Data to VisualMotion 13-5

X

X10 port settings 2-160
X16 port settings 2-161

Z

zone tables
 data structure 3-28
Zone Tables 13-44
Zones 2-125

17 Service & Support

17.1 Helpdesk

Unser Kundendienst-Helpdesk im Hauptwerk Lohr am Main steht Ihnen mit Rat und Tat zur Seite. Sie erreichen uns

- telefonisch: **+49 (0) 9352 40 50 60**
über Service Call Entry Center Mo-Fr 07:00-18:00
- per Fax: **+49 (0) 9352 40 49 41**
- per e-Mail: service@boschrexroth.de

Our service helpdesk at our headquarters in Lohr am Main, Germany can assist you in all kinds of inquiries. Contact us

- by phone: **+49 (0) 9352 40 50 60**
via Service Call Entry Center Mo-Fr 7:00 am - 6:00 pm
- by fax: **+49 (0) 9352 40 49 41**
- by e-mail: service@boschrexroth.de

17.2 Service-Hotline

Außerhalb der Helpdesk-Zeiten ist der Service direkt ansprechbar unter

oder **+49 (0) 171 333 88 26**
+49 (0) 172 660 04 06

After helpdesk hours, contact our service department directly at

or **+49 (0) 171 333 88 26**
+49 (0) 172 660 04 06

17.3 Internet

Ergänzende Hinweise zu Service, Reparatur und Training sowie die **aktuellen** Adressen unserer Service- und Vertriebsbüros finden Sie unter www.boschrexroth.com – einige Angaben in dieser Dokumentation können inzwischen überholt sein.

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem lokalen Ansprechpartner auf.

- Verkaufsniederlassungen
- Niederlassungen mit Kundendienst

Additional notes about service, repairs and training as well as the **actual** addresses of our sales- and service facilities are available on the Internet at www.boschrexroth.com – some information in this documentation may meanwhile be obsolete.

Please contact the sales & service offices in your area first.

- sales agencies
- offices providing service

17.4 Vor der Kontaktaufnahme... - Before contacting us...

Wir können Ihnen schnell und effizient helfen wenn Sie folgende Informationen bereithalten:

detaillierte Beschreibung der Störung und der Umstände.

Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern.

Tel./Faxnummern und e-Mail-Adresse, unter denen Sie für Rückfragen zu erreichen sind.

For quick and efficient help, please have the following information ready:

1. Detailed description of the failure and circumstances.
2. Information on the nameplate of the affected products, especially typecodes and serial numbers.
3. Your phone/fax numbers and e-mail address, so we can contact you in case of questions.

17.5 Kundenbetreuungsstellen - Sales & Service Facilities

Deutschland – Germany

vom Ausland:

(0) nach Landeskennziffer weglassen!

from abroad:

don't dial (0) after country code!

Vertriebsgebiet Mitte Germany Centre	SERVICE	SERVICE	SERVICE
Bosch Rexroth AG Bgm.-Dr.-Nebel-Str. 2 97816 Lohr am Main Kompetenz-Zentrum Europa Tel.: +49 (0)9352 40-0 Fax: +49 (0)9352 40-4885	CALL ENTRY CENTER MO – FR von 07:00 - 18:00 Uhr from 7 am – 6 pm Tel. +49 (0) 9352 40 50 60 service@boschrexroth.de	HOTLINE MO – FR von 17:00 - 07:00 Uhr from 5 pm - 7 am + SA / SO Tel.: +49 (0)172 660 04 06 oder / or Tel.: +49 (0)171 333 88 26	ERSATZTEILE / SPARES verlängerte Ansprechzeit - extended office time - ♦ nur an Werktagen - only on working days - ♦ von 07:00 - 18:00 Uhr - from 7 am - 6 pm - Tel. +49 (0) 9352 40 42 22
Vertriebsgebiet Süd Germany South	Gebiet Südwest Germany South-West	Vertriebsgebiet Ost Germany East	Vertriebsgebiet Ost Germany East
Bosch Rexroth AG Landshuter Allee 8-10 80637 München Tel.: +49 (0)89 127 14-0 Fax: +49 (0)89 127 14-490	Bosch Rexroth AG Vertrieb Deutschland – VD-BI Geschäftsbereich Regionalzentrum Südwest Ringstrasse 70 / Postfach 1144 70736 Fellbach / 70701 Fellbach Tel.: +49 (0)711 57 61–100 Fax: +49 (0)711 57 61–125	Bosch Rexroth AG Beckerstraße 31 09120 Chemnitz Tel.: +49 (0)371 35 55-0 Fax: +49 (0)371 35 55-333	Bosch Rexroth AG Regionalzentrum Ost Walter-Köhn-Str. 4d 04356 Leipzig Tel.: +49 (0)341 25 61-0 Fax: +49 (0)341 25 61-111
Vertriebsgebiet West Germany West	Vertriebsgebiet Mitte Germany Centre	Vertriebsgebiet Nord Germany North	Vertriebsgebiet Nord Germany North
Bosch Rexroth AG Vertrieb Deutschland Regionalzentrum West Borsigstrasse 15 40880 Ratingen Tel.: +49 (0)2102 409-0 Fax: +49 (0)2102 409-406	Bosch Rexroth AG Regionalzentrum Mitte Waldecker Straße 13 64546 Mörfelden-Walldorf Tel.: +49 (0) 61 05 702-3 Fax: +49 (0) 61 05 702-444	Bosch Rexroth AG Walsroder Str. 93 30853 Langenhagen Tel.: +49 (0) 511 72 66 57-0 Fax: +49 (0) 511 72 66 57-95	Bosch Rexroth AG Kieler Straße 212 22525 Hamburg Tel.: +49 (0) 40 81 955 966 Fax: +49 (0) 40 85 418 978

Europa (West) - Europe (West)

vom Ausland: (0) nach Landeskennziffer weglassen, **Italien:** 0 nach Landeskennziffer mitwählen
from abroad: don't dial (0) after country code, **Italy:** dial 0 after country code

Austria - Österreich Bosch Rexroth AG Stachegasse 13 1120 Wien Tel.: +43 (0)1 985 25 40 Fax: +43 (0)1 985 25 40-93	Austria – Österreich Bosch Rexroth AG Industriepark 18 4061 Pasching Tel.: +43 (0)7221 605-0 Fax: +43 (0)7221 605-21	Belgium - Belgien Bosch Rexroth AG Electric Drives & Controls Industrielaan 8 1740 Ternat Tel.: +32 (0)2 5830719 Service: +32 (0)2 5830717 Fax: +32 (0)2 5830731 indramat@boschrexroth.be	Denmark - Dänemark Bosch Rexroth A/S Zinkvej 6 8900 Randers Tel.: +45 (0)87 11 90 60 Fax: +45 (0)87 11 90 61
Great Britain – Großbritannien Bosch Rexroth Ltd. Broadway Lane, South Cerney Cirencester, Glos GL7 5UH Tel.: +44 (0)1285 863000 Fax: +44 (0)1285 863030 sales@boschrexroth.co.uk service@boschrexroth.co.uk	Finland - Finnland Rexroth Mecman Oy Ansatie 6 017 40 Vantaa Tel.: +358 (0)9 84 91-11 Fax: +358 (0)9 84 91-13 60	France - Frankreich Bosch Rexroth S.A. Avenue de la Trentaine BP. 74 77503 CHELLES CEDEX Tel.: +33 (0)164 72-70 00 Fax: +33 (0)164 72-63 00 Hotline: +33 (0)608 33 43 28	France - Frankreich Bosch Rexroth S.A. 1270, Avenue de Lardenne 31100 Toulouse Tel.: +33 (0)5 61 49 95 19 Fax: +33 (0)5 61 31 00 41
France - Frankreich Bosch Rexroth S.A. 91, Bd. Irène Joliot-Curie 69634 Vénissieux – Cedex Tel.: +33 (0)4 78 78 53 65 Fax: +33 (0)4 78 78 53 62	Italy - Italien Bosch Rexroth S.p.A. Via G. Di Vittoria, 1 20063 Cernusco S/N.MI Tel.: +39 02 2 365 270 Fax: +39 02 700 408 252378	Italy - Italien Bosch Rexroth S.p.A. Via Paolo Veronesi, 250 10148 Torino Tel.: +39 011 224 88 11 Fax: +39 011 220 48 04	Italy - Italien Bosch Rexroth S.p.A. Via del Progresso, 16 (Zona Ind.) 35020 Padova Tel.: +39 049 8 70 13 70 Fax: +39 049 8 70 13 77
Italy - Italien Bosch Rexroth S.p.A. Via Mascia, 1 80053 Castellammare di Stabia NA Tel.: +39 081 8 71 57 00 Fax: +39 081 8 71 68 85	Italy - Italien Bosch Rexroth S.p.A. Viale Oriani, 38/A 40137 Bologna Tel.: +39 051 34 14 14 Fax: +39 051 34 14 22	Netherlands - Niederlande/Holland Bosch Rexroth B.V. Kruisbroeksestraat 1 (P.O. Box 32) 5281 RV Boxtel Tel.: +31 (0)411 65 19 51 Fax: +31 (0)411 65 14 83 indramat@hydraudyne.nl	Netherlands - Niederlande/Holland Bosch Rexroth Services B.V. Kruisbroeksestraat 1 (P.O. Box 32) 5281 RV Boxtel Tel.: +31 (0)411 65 19 51 Fax: +31 (0)411 67 78 14
Norway - Norwegen Bosch Rexroth AS Berghagan 1 or: Box 3007 1405 Ski-Langhus 1402 Ski Tel.: +47 (0)64 86 41 00 Fax: +47 (0)64 86 90 62 jul.ruud@rexroth.no	Spain - Spanien Bosch Rexroth S.A. Centro Industrial Santiga Obradors s/n 08130 Santa Perpetua de Mogoda Barcelona Tel.: +34 9 37 47 94 00 Fax: +34 9 37 47 94 01	Spain - Spanien Bosch Rexroth S.A. Goimendi S.A. Parque Empresarial Zuatzu C/ Francisco Grandmontagne no.2 20018 San Sebastian Tel.: +34 9 43 31 84 21 - service: +34 9 43 31 84 56 Fax: +34 9 43 31 84 27 - service: +34 9 43 31 84 60 satindramat-goimendi@adeqi.es	Sweden - Schweden Rexroth Mecman Svenska AB Varuvägen 7 125 81 Stockholm Tel.: +46 (0)8 727 92 00 Fax: +46 (0)8 647 32 77
Sweden - Schweden Rexroth Mecman Svenska AB Ekvåndan 7 254 67 Helsingborg Tel.: +46 (0) 42 38 88 -50 Fax: +46 (0) 42 38 88 -74	Switzerland West - Schweiz West Bosch Rexroth Suisse SA Rue du village 1 1020 Renens Tel.: +41 (0)21 632 84 20 Fax: +41 (0)21 632 84 21	Switzerland East - Schweiz Ost Bosch Rexroth Schweiz AG Hemrietstrasse 2 8863 Buttikon Tel. +41 (0) 55 46 46 205 Fax +41 (0) 55 46 46 222	

Europa (Ost) - Europe (East)

vom Ausland: (0) nach Landeskennziffer weglassen
from abroad: don't dial (0) after country code

Czech Republic - Tschechien	Czech Republic - Tschechien	Hungary - Ungarn	Poland – Polen
Bosch -Rexroth, spol.s.r.o. Hviezdoslavova 5 627 00 Brno Tel.: +420 (0)5 48 126 358 Fax: +420 (0)5 48 126 112	DEL a.s. Strojirenská 38 Zdar nad Sázavou 591 01 Czech republic Tel.: +420 616 64 3144 Fax: +420 616 216 57	Bosch Rexroth Kft. Angol utca 34 1149 Budapest Tel.: +36 (1) 364 00 02 Fax: +36 (1) 383 19 80	Bosch Rexroth Sp.zo.o. Biuro Poznan ul. Dabrowskiego 81/85 60-529 Poznan Tel.: +48 061 847 64 62 /-63 Fax: +48 061 847 64 02
Rumania - Rumänien	Russia - Russland	Russia - Russland	Turkey - Türkei
Bosch Rexroth Sp.zo.o. Str. Drobety nr. 4-10, app. 14 70258 Bucuresti, Sector 2 Tel.: +40 (0)1 210 48 25 +40 (0)1 210 29 50 Fax: +40 (0)1 210 29 52	Bosch Rexroth Wolokolamskoje Chaussee 73 Zimmer 406, 408 RUS – 123424 Moskau Tel.: +7 095/ 232 08 34 +7 095/ 232 08 35 Fax: +7 095/ 232 08 36 info.rex@rexroth.ru	ELMIS 10, Internationalnaya Str. 246640 Gomel, Belarus Tel.: +375/ 232 53 42 70 Fax: +375/ 232 53 37 69 elmis_ltd@yahoo.com	Bosch Rexroth Otomasyon San & Tic. A. S. Fevzi Cakmak Cad No. 3 34630 Sefaköy Istanbul Tel.: +90 212 541 60 70 Fax: +90 212 599 34 07
Slowenia - Slowenien			
DOMEL Otoki 21 64 228 Zelezniki Tel.: +386 5 5117 152 Fax: +386 5 5117 225 brane.ozebek@domel.si			

Africa, Asia, Australia – incl. Pacific Rim

vom Ausland:
from abroad:

(0) nach Landeskenziffer weglassen!
don't dial (0) after country code!

<p>Australia - Australien</p> <p>AIMS - Australian Industrial Machinery Services Pty. Ltd. Unit 3/45 Horne ST Campbellfield , VIC 3061 Melbourne Tel.: +61 (0) 393 590 228 Fax: +61 (0) 393 590 286 Hotline: +61 (0) 419 369 195 terryobrien@aimservices.com.au</p>	<p>Australia - Australien</p> <p>Bosch Rexroth Pty. Ltd. No. 7, Endeavour Way Braeside Victoria, 31 95 Melbourne Tel.: +61 (0)3 95 80 39 33 Fax: +61 (0)3 95 80 17 33 mel@rexroth.com.au</p>	<p>China</p> <p>Bosch Rexroth Ltd. Wai Gaoqiao Free Trade Zone No.122, Fu Te Dong Yi Road Shanghai 200131 - P.R.China Tel.: +86 21 58 66 30 30 Fax: +86 21 58 66 55 23 roger.shi_sh@boschrexroth.com.cn</p>	<p>China</p> <p>Bosch Rexroth (China) Ltd. 15/F China World Trade Center 1, Jianguomenwai Avenue Beijing 100004, P.R.China Tel.: +86 10 65 05 03 80 Fax: +86 10 65 05 03 79</p>
<p>China</p> <p>Bosch Rexroth (China) Ltd. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023, P.R.China Tel.: +86 411 46 78 930 Fax: +86 411 46 78 932</p>	<p>China</p> <p>Bosch Rexroth (Changzhou) Co.Ltd. Guangzhou Repres. Office Room 1014-1016, Metro Plaza, Tian He District, 183 Tian He Bei Rd Guangzhou 510075, P.R.China Tel.: +86 20 8755-0030 +86 20 8755-0011 Fax: +86 20 8755-2387</p>	<p>Hongkong</p> <p>Bosch Rexroth (China) Ltd. 6th Floor, Yeung Yiu Chung No.6 Ind Bldg. 19 Cheung Shun Street Cheung Sha Wan, Kowloon, Hongkong Tel.: +852 22 62 51 00 Fax: +852 27 41 33 44 alexis.siu@boschrexroth.com.hk</p>	<p>India - Indien</p> <p>Bosch Rexroth (India) Ltd. Plot. A-58, TTC Industrial Area Thane Turbhe Midc Road Mahape Village Navi Mumbai - 400 701 Tel.: +91 (0)22 7 61 46 22 Fax: +91 (0)22 7 68 15 31</p>
<p>India - Indien</p> <p>Bosch Rexroth (India) Ltd. Plot. 96, Phase III Peenya Industrial Area Bangalore - 560058 Tel.: +91 (0)80 8 39 73 74 Fax: +91 (0)80 8 39 43 45</p>	<p>Indonesia - Indonesien</p> <p>PT. Rexroth Wijayakusuma Building # 202, Cilandak Commercial Estate Jl. Cilandak KKO, Jakarta 12560 Tel.: +62 21 7891169 (5 lines) Fax: +62 21 7891170 - 71</p>	<p>Japan</p> <p>Bosch Rexroth Automation Corp. Service Center Japan Yutakagaoka 1810, Meito-ku, NAGOYA 465-0035, Japan Tel.: +81 (0)52 777 88 41 +81 (0)52 777 88 53 +81 (0)52 777 88 79 Fax: +81 (0)52 777 89 01</p>	<p>Japan</p> <p>Bosch Rexroth Automation Corp. 1F, I.R. Building Nakamachidai 4-26-44, Tsuzuki-ku YOKOHAMA 224-0041, Japan Tel.: +81 (0)45 942 72 10 Fax: +81 (0)45 942 03 41</p>
<p>Korea</p> <p>Bosch Rexroth-Korea Ltd. 1515-14 Dadae-Dong, Saha-Ku Pusan Metropolitan City, 604-050 Republic of South Korea Tel.: +82 (0)51 26 00 741 Fax: +82 (0)51 26 00 747 gyhan@rexrothkorea.co.kr</p>	<p>Malaysia</p> <p>Bosch Rexroth Sdn.Bhd. Head Office No. 3, Block B, Jalan SS 13/5 Subang Jaya Industrial Estate 47500 Petaling Jaya - Selangor Tel.: +60 (0) 3 73 44 870 Fax: +60 (0) 3 73 44 864 hockhwa@hotmail.com</p>	<p>Singapore - Singapur</p> <p>Robert Bosch (SEA) Pte Ltd. Dept. RBSI-R/SAT 38-C Jalan Pemimpin Singapore 577180 Tel.: +65 35 05 470 Fax: +65 35 05 313 kenton.peh@sg.bosch.com</p>	<p>South Africa - Südafrika</p> <p>TECTRA Automation (Pty) Ltd. 28 Banfield Road, Industria North RSA - Maraisburg 1700 Tel.: +27 (0)11 673 20 80 Fax: +27 (0)11 673 72 69 Hotline: +27 (0)82 903 29 23 georgv@tectra.co.za</p>
<p>Taiwan</p> <p>Rexroth Uchida Co., Ltd. No.17, Lane 136, Cheng Bei 1 Rd., Yung Kang, Tainan Hsien Taiwan, R.O.C. Tel.: +886 (0)6 25 36 565 Fax: +886 (0)6 25 34 754 indramat@mail.net.tw</p>	<p>Thailand</p> <p>NC Advance Technology Co. Ltd. 59/76 Moo 9 Ramintra road 34 Tharang, Bangkhen, Bangkok 10230 Tel.: +66 2 943 70 62 +66 2 943 71 21 Fax: +66 2 509 23 62 sonkawin@hotmail.com</p>		

Nordamerika – North America

USA Hauptniederlassung - Headquarters Bosch Rexroth Corporation Electric Drives and Controls 5150 Prairie Stone Parkway Hoffman Estates, IL 60192-3707 Tel.: +1 847 6 45 36 00 Fax: +1 847 6 45 62 01 service@boschrexroth.com	USA Central Region - Mitte Bosch Rexroth Corporation Electric Drives and Controls 1701 Harmon Road Central Region Technical Center Auburn Hills, MI 48326 Tel.: +1 248 3 93 33 30 Fax: +1 248 3 93 29 06	USA Southeast Region - Südwest Bosch Rexroth Corporation Electric Drives and Controls Southeastern Technical Center 3625 Swiftwater Park Drive Suwanee, Georgia 30124 Tel.: +1 770 9 32 32 00 Fax: +1 770 9 32 19 03	USA SERVICE-HOTLINE - 7 days x 24hrs - +1-800-860-1055
USA East Region –Ost Bosch Rexroth Corporation Electric Drives and Controls Charlotte Regional Sales Office 14001 South Lakes Drive Charlotte, North Carolina 28273 Tel.: +1 704 5 83 97 62 +1 704 5 83 14 86	USA Northeast Region – Nordost Bosch Rexroth Corporation Electric Drives and Controls Northeastern Technical Center 99 Rainbow Road East Granby, Connecticut 06026 Tel.: +1 860 8 44 83 77 Fax: +1 860 8 44 85 95	USA West Region – West Bosch Rexroth Corporation 7901 Stoneridge Drive, Suite 220 Pleasant Hill, California 94588 Tel.: +1 925 227 10 84 Fax: +1 925 227 10 81	
Canada East - Kanada Ost Bosch Rexroth Canada Corporation Burlington Division 3426 Mainway Drive Burlington, Ontario Canada L7M 1A8 Tel.: +1 905 335 55 11 Fax: +1 905 335-41 84 michael.moro@boschrexroth.ca	Canada West - Kanada West Bosch Rexroth Canada Corporation 5345 Goring St. Burnaby, British Columbia Canada V7J 1R1 Tel.: +1 604 205-5777 Fax: +1 604 205-6944 david.gunby@boschrexroth.ca	Mexico Bosch Rexroth S.A. de C.V. Calle Neptuno 72 Unidad Ind. Vallejo MEX - 07700 Mexico, D.F. Tel.: +52 5 754 17 11 +52 5 754 36 84 +52 5 754 12 60 Fax: +52 5 754 50 73 +52 5 752 59 43	

Südamerika – South America

Argentina - Argentinien Bosch Rexroth S.A.I.C. "The Drive & Control Company" Acassusso 48 41/47 1605 Munro Prov. Buenos Aires Tel.: +54 (0)11 4756 01 40 Fax: +54 (0)11 4756 01 36 mannesmann@mannesmannsaic.com.ar	Argentina - Argentinien NAKASE Servicio Tecnico CNC Calle 49, No. 5764/66 1653 Villa Balester Prov. - Buenos Aires Tel.: +54 (0) 11 4768 36 43 Fax: +54 (0) 11 4768 24 13 nakase@usa.net nakase@nakase.com	Brazil - Brasilien Bosch Rexroth Ltda. Av. Tégula, 888 Ponte Alta, Atibaia SP CEP 12942-440 Tel.: +55 (0)11 4414 56 92 +55 (0)11 4414 56 84 Fax sales: +55 (0)11 4414 57 07 Fax serv.: +55 (0)11 4414 56 86 alexandre.wittwer@rexroth.com.br	Brazil - Brasilien Bosch Rexroth Ltda. R. Dr.Humberto Pinheiro Vieira, 100 Distrito Industrial [Caixa Postal 1273] BR - 89220-390 Joinville - SC Tel./Fax: +55 (0)47 473 58 33 Mobil: +55 (0)47 9974 6645 prochnow@zaz.com.br
Columbia - Kolumbien Refflutec de Colombia Ltda. Calle 37 No. 22-31 Santafé de Bogotá, D.C. Colombia Tel.: +57 1 368 82 67 +57 1 368 02 59 Fax: +57 1 268 97 37 reflutec@inter.net.co			

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Phone +49 93 52-40-0
Fax +49 93 52-40-48 85
www.boschrexroth.de

