



VisualMotion 8 (GPP) Multi-Axis Motion Control

Functional Description

Title	VisualMotion 8 (GPP) Multi-Axis Motion Control														
Type of Documentation	Functional Description														
Document Typecode	DOK-VISMOT-VM*-08VRS**-FK02-AE-P														
Internal File Reference	<ul style="list-style-type: none"> • Document Number, 120-2300-B312-02/AE 														
Purpose of Documentation	<p>This document is a functional description for the PPC-R multiple axes motion control using GPP 8 firmware. The information provided in this document is intended to support the VisualMotion 8 system and covers the following areas:</p> <ul style="list-style-type: none"> • system and I/O configurations • enhanced I/O Mapper and PLS functionality • control registers/parameters • VisualMotion 8 menu and icon descriptions • Kinematics and BTC06 descriptions • textual language programming • direct ASCII communications 														
Record of Revisions	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Description</th><th style="text-align: center; padding: 2px;">Release Date</th><th style="text-align: left; padding: 2px;">Notes</th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">01</td><td style="text-align: center; padding: 2px;">03/01</td><td style="padding: 2px;">initial release</td></tr> <tr> <td style="padding: 2px;">02</td><td style="text-align: center; padding: 2px;">08/01</td><td style="padding: 2px;">updated release</td></tr> <tr> <td style="padding: 2px;"></td><td style="text-align: center; padding: 2px;"></td><td style="padding: 2px;"></td></tr> </tbody> </table>			Description	Release Date	Notes	01	03/01	initial release	02	08/01	updated release			
Description	Release Date	Notes													
01	03/01	initial release													
02	08/01	updated release													

Copyright	© 2001 Rexroth Indramat GmbH 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	All rights are reserved with respect to the content of this documentation and the availability of the product.
Published by	Rexroth Indramat GmbH • Bgm.-Dr.-Nebel-Str. 2 • 97816 Lohr am Main • Germany • Tel.: 09352/40-0 • Telex: 689421 • Fax: 09352/40-4885 Rexroth Indramat Division • 5150 Prairie Stone Parkway • Hoffman Estates, IL 60192 • USA • Tel.: 847-645-3600 • Fax: 847-645-6201 http://www.rexroth.com/indramat Dept. ESV (DPJ/EN)

Table of Contents

1 VisualMotion 8.0 Overview	1-1
1.1 System Overview	1-1
GPP 8.0 System Overview	1-1
System Components.....	1-2
PLC Support.....	1-2
Interface Support	1-2
2 VisualMotion's I/O System	2-1
2.1 I/O Devices Supported in GPP	2-1
2.2 I/O Configuration Tool	2-1
I/O Initialization	2-2
Configuring I/O Devices Offline.....	2-2
Configuring I/O Devices Online.....	2-3
2.3 I/O Configuration Menus	2-3
File Menu	2-3
Edit Menu	2-4
View Menu	2-6
Settings Menu	2-8
Help Menu.....	2-8
2.4 Local RECO02 I/O Modules.....	2-8
Local RECO02 I/O Configuration (Offline).....	2-10
Add an I/O Module	2-10
Local RECO02 I/O Configuration (Online).....	2-11
Configure RECO02 Modules	2-12
2.5 Remote I/O Stations.....	2-13
2.6 Remote SERCOS Drive I/O Cards	2-14
Drive I/O Configuration	2-14
Remote Analog Input Drive Cards	2-15
2.7 Assigning Register Numbers	2-15
Register Usage	2-16
Required Registers Based on I/O Device	2-17
Configuring RECO02 I/O Modules for the PPC-R	2-18
VisualMotion n Register Definition	2-19
DEA Digital Drive I/O Cards.....	2-21
3 Enhanced I/O Mapper	3-1
3.1 I/O Mapper Specifications	3-2
Rung Specifications	3-2
Maximum Rows Allowed in Ladder Window.....	3-4
Total Operations	3-4
3.2 Creating and Downloading a New I/O Mapper	3-5

3.3	Downloading the Default I/O Mapper	3-5
3.4	Menu Selection	3-6
	The File Menu	3-6
	The Edit Menu.....	3-7
	The View Menu	3-7
	The Setting Menu.....	3-7
	The Window Menu	3-9
	The Help Menu	3-9
3.5	Ladder Logic Toolbar	3-10
3.6	Input Logic Functions	3-10
	Properties.....	3-10
3.7	Output Logic Functions	3-12
	Coil Relay.....	3-12
	Binary Shift Register (BSR) 	3-15
	Timer (TON) 	3-16
	Counter (UDC) 	3-17
3.8	Forcing Icon Toolbar	3-18
	Forcing Options.....	3-18
	Using the Forcing Icons	3-19
3.9	Additional Tools.....	3-21
	Delete row	3-21
	Undo.....	3-21
	Insert row	3-21
	Cross Reference	3-21
	Cut, Copy, Paste and Delete	3-22
	Check rungs and convert to Boolean strings	3-22

4 Programmable Limit Switch Functionality 4-1

4.1	PLS Description	4-1
	PLS Object.....	4-1
	Control PLS.....	4-2
	Drive PLS.....	4-3
	Option Card PLS.....	4-5
4.2	Configure a Control PLS	4-7
	Switch Configuration for a Control PLS	4-7
	PLS Master Configuration for a Control PLS	4-9
	PLS Register Assignment for a Control PLS	4-10
4.3	Configure a Drive PLS	4-12
	Switch Configuration for a Drive PLS	4-13
	PLS Master Configuration for a Drive PLS	4-14
	PLS Register Assignment for a Drive PLS	4-14
4.4	Configure an Option Card PLS	4-17
	Switch Configuration for an Option Card PLS	4-18
	PLS Master(s) Configuration for an Option Card PLS	4-19
	PLS Register Assignment for an Option Card PLS	4-21

PLS Outputs.....	4-23
Output Configuration.....	4-23
4.5 Saving and Downloading PLS Configurations	4-30
Saving a PLS to a File	4-30
Downloading a PLS to the Control.....	4-30
4.6 Uploading PLS Configurations	4-31
Uploading a PLS Configuration from a File	4-31
Uploading a PLS Configuration from the Control.....	4-31
4.7 Editing PLS Configurations	4-31
Editing with a Right Mouse Click	4-32
4.8 Monitoring a PLS Status	4-34
PLS Message.....	4-34
4.9 Access PLS Data via the Calc Icon	4-35
Control PLS.....	4-35
Drive PLS	4-36
Option Card PLS.....	4-36

5 Registers 5-1

Register 001: System Control.....	5-3
Registers 002-005: Task Control	5-5
Cycle Control Considerations	5-8
Register 006: System Diagnostic Code.....	5-8
Registers 007-010: Task Jog Control	5-9
Registers 011-018, 209-232: Axis Control.....	5-11
Register 019: Fieldbus Status.....	5-12
Register 020: Fieldbus Diagnostics	5-14
Register 021: System Status	5-15
Registers 022-025: Task Status	5-16
Register 026: Fieldbus Resource Monitor	5-17
Registers 031-038, 309-332: Axis Status	5-19
Registers 040: Link Ring Status	5-21
Registers 041: Link Ring Data 1	5-22
Registers 042: Link Ring Data 2	5-22
Registers 050: Ethernet Status	5-23
Register 051: Standard Message Count.....	5-23
Register 052: Cyphered Message Count.....	5-23
Register 053: Invalid Protocol Count	5-23
Registers 088 and 089: Task A Extend Event Control	5-24
Registers 090 and 091: Latch and Unlatch.....	5-24
Registers 092-094: Mask Pendant Key Functionality.....	5-24
Registers 095-097: BTC06 Teach Pendant Status	5-25
Registers 098: Pendant Control - Task A, B	5-26
Registers 099: Pendant Control - Task C-D	5-26
Registers 150 and 151: Virtual Master 1 & 2 Control	5-27
Registers 152-159: ELS Groups 1-8 Control.....	5-29
Registers 241 and 242: Virtual Master 1 & 2 Status.....	5-31

Registers 243 - 250: ELS Groups 1-8 Status	5-32
6 Parameters	6-1
6.1 Overview	6-1
6.2 Parameter Identification	6-2
6.3 Parameter Transfer Commands	6-4
6.4 List of Parameters	6-5
System (Control) Parameters - Class C	6-5
Task Parameters - Class T	6-12
Axis Parameters - Class A	6-14
6.5 System Control Parameters – Class C	6-17
System Setup (C-0-0001 through C-0-0035).....	6-17
Jogging and Display Parameters (C-0-0042 through C-0-0056).....	6-26
Program Management (C-0-0090 through C-0-0099)	6-29
System Status (C-0-0100 through C-0-0127)	6-33
Control Processor Usage Status (C-0-0200 through C-0-0203).....	6-39
Link Ring Parameters (C-0-0300 through C-0-0303)	6-41
Ethernet Parameters (C-0-0400 through C-0-0408).....	6-43
BTC06 Teach Pendant (C-0-0801 through C-0-0814)	6-48
Internal System Monitoring (C-0-0990).....	6-58
System Memory Parameters (C-0-0994 and C-0-0996).....	6-59
System Parameter Lists (C-0-2000 through C-0-2021).....	6-60
Oscilloscope Parameters (C-0-2501 through C-0-2523)	6-66
Fieldbus Interface Parameters (C-0-2600 through C-0-2701).....	6-75
Card PLS Interface Parameters (C-0-2901 through C-0-2943).....	6-83
I/O Mapper Parameters (C-0-3000 through C-0-3005)	6-94
CAM Table Parameters (C-0-3100 through C-0-3109)	6-96
6.6 Task Parameters - Class T	6-97
Task Setup (T-0-0001 and T-0-0002)	6-97
Coordinated Motion (T-0-0005 through T-0-0026)	6-100
Robotics (T-0-0035 through T-0-0059)	6-106
Coordinated Motion Status (T-0-0100 through T-0-0113)	6-107
Task Status (T-0-0120 through T-0-0200)	6-110
Task Parameter Lists (T-0-2000 and T-0-2001)	6-115
6.7 Axis Parameters – Class A	6-116
Axis Setup (A-0-0001through A-0-0038)	6-116
Axis Status (A-0-0100 through A-0-0145).....	6-134
Electronic Line Shafting (A-0-0150 through A-0-0164).....	6-141
Axis Feedback Capture (Registration) (A-0-0170 through A-0-0174)	6-147
Optional SERCOS Data (A-0-0180 through A-0-0196)	6-148
Multiplexing Parameters (A-0-0200 through A-0-0203) (DKC 2.3 only)	6-151
Axis Parameter Lists (A-0-2000 and A-0-2001)	6-153
7 VisualMotion Toolkit Menu Commands	7-1
7.1 Introduction.....	7-1
7.2 The <u>File</u> Menu	7-2

Standard Windows Functions	7-3
Importing and Exporting of Program Labels	7-3
Printing Program Data	7-5
Accessing Sample Programs.....	7-6
7.3 The <u>Edit</u> Menu.....	7-6
Windows Editing Features	7-6
Clearing the Icon Workspace (Clear Current Task).....	7-7
Find, Find Next, Replace	7-7
Add Subroutine	7-8
Add Event Function.....	7-8
Edit Labels	7-9
7.4 The <u>View</u> Menu	7-15
Tasks A, B, C, and D	7-16
Subroutines.....	7-16
Event Functions	7-17
Zoom Out F6.....	7-18
Icon Palette	7-19
Icon Comments.....	7-19
Icon Captions	7-20
7.5 The <u>Build</u> Menu.....	7-20
 Save, Compile, Download	7-20
Compile	7-20
Display Code.....	7-20
Program Management	7-21
7.6 The <u>Commission</u> Menu	7-23
Drives	7-24
I/O Setup.....	7-24
I/O Mapper	7-25
Fieldbus Mapper	7-25
PLS (Programmable Limit Switch).....	7-26
Coordinated Motion.....	7-27
Archive	7-30
Transfer.....	7-32
7.7 The <u>On-Line Data</u> Menu.....	7-38
Parameters	7-38
Registers	7-51
Variables	7-54
Events	7-54
Points	7-56
CAM Indexer	7-58
ELS	7-59
PID	7-62
Registration	7-67
Conveyor Tracker	7-71
Sequencer.....	7-72
Zones	7-75

7.8 The <u>Diagnostics</u> Menu	7-76
System	7-76
Drives	7-77
Drives on Ring	7-77
Diagnostic Log	7-78
Tasks.....	7-79
Oscilloscope.....	7-80
Break Point.....	7-80
Show Program Flow.....	7-81
7.9 The VM <u>Tools</u> Menu.....	7-82
CAM Builder.....	7-82
Jogging.....	7-88
CLC_DDE Release7 / CLC_DDE Release8.....	7-91
IoBox Release7 / IoBox Release8	7-91
VisualMotion32 Release 7 / VisualMotion32 Release 8	7-91
7.10 The <u>Settings</u> Menu	7-92
Card Selection	7-92
Configuration.....	7-93
Control Serial Ports	7-94
7.11 The <u>Help</u> Menu.....	7-95
Getting Started.....	7-95
Search.....	7-95
Drives Help Directories	7-95
Version Change Log	7-96
About VisualMotion	7-96

8 Icon Programming 8-1

8.1 Introduction.....	8-1
Working with VisualMotion Toolkit's Icon Palettes.....	8-1
Connecting Icons	8-3
Icon Labels and Comments	8-5
8.2 VisualMotion Toolkit Icons	8-6
Single Axis Icon Palette	8-6
Coordinated Motion Icon Palette	8-7
ELS Icon Palette	8-8
Utility Icon Palette	8-9
Accel	8-10
Axis	8-11
AxisEvt	8-16
Branch.....	8-17
Calc	8-19
CAM	8-28
CAMAdj	8-29
CAMBuild	8-30
CAM Indexer	8-34
Circle	8-45

Command	8-46
CvyrInit (Conveyor Tracker Setup)	8-47
Decel	8-50
ELSAdj	8-51
ELSGrp	8-52
ELSMstr	8-62
ELSMode	8-66
Event	8-67
Finish	8-68
Go	8-68
Home	8-69
I/O Setup	8-70
Join	8-71
Joint	8-71
Line	8-72
Move	8-73
Msg	8-74
Param	8-75
Path	8-77
PathCalc	8-78
PID	8-79
PLS	8-81
Position	8-82
PrmBit	8-83
PrmInt	8-86
Probe	8-88
ProbeEvt	8-89
Ratio	8-90
Register Transfer	8-91
RobotOrg	8-92
RobotTool	8-92
Scissors	8-93
Sequencer	8-93
Size	8-95
Start	8-96
Stop	8-97
Sub	8-98
Velocity	8-99
VM (Virtual Master)	8-100
Wait	8-104

9 Kinematics**9-1**

9.1 Description	9-1
Associated Task Parameters	9-2
Normal Case Kinematics	9-2
Special Case Kinematics	9-2

9.2 Kinematic Libraries.....	9-3
Kinematic #1	9-3
Kinematic #2	9-4
Kinematic #3	9-4
Kinematic #4	9-5
Kinematic #5	9-5
Kinematic #8 with Velocity Precision	9-6
Kinematic #9	9-7
Kinematic #10	9-7
Kinematic #12	9-8
10 VisualMotion Human Machine Interface	10-1
10.1 Overview	10-1
10.2 BTC06 Teach Pendant Screens	10-1
Menu Map	10-2
10.3 BTC06 Teach Pendant Setup	10-4
10.4 BTC06 Keyboard Operation.....	10-6
Keyboard Map.....	10-8
Cursor Control and Editing.....	10-9
Number or Letter Selection.....	10-9
Jogging Control.....	10-9
Task Control.....	10-10
Teach Control.....	10-10
10.5 F1 Program Menu	10-11
Sequencer Editing (F4).....	10-12
10.6 F2 Table Edit Menu.....	10-14
Absolute Table Menu (F1)	10-14
Relative Table Menu (F2)	10-15
Event Table Menu (F3)	10-16
Integer Table Menu (F4)	10-17
Floating Table Menu (F5).....	10-17
Global Integer Table Menu (F6).....	10-18
Global Floating Table Menu (F7)	10-18
10.7 F3 Jog Menu	10-19
Jog Systems	10-20
Jog Method	10-20
Teaching Points	10-21
Jog Fine Adjustments	10-21
10.8 F4 Control Menu	10-21
Control Menu: Auto Run/Hold Mode	10-22
Control Menu: Auto Step Mode.....	10-23
Control Menu: Manual Mode.....	10-24
10.9 F5 Register I/O Menu.....	10-24
10.10 F6 Parameter Menu	10-25
F1 - Card Parameter Screen.....	10-26
F2 - Axis Parameter Screen.....	10-26

F3 - Task Parameter Screen.....	10-27
F4 - Drive Parameter Screen.....	10-28
10.11F6 Security Menu	10-29
10.12F8 Diagnostics Menu	10-30
10.13Error Screen	10-31

11 Textual Language Programming 11-1

11.1 Overview	11-1
11.2 Directives.....	11-1
11.3 VisualMotion Textual Instructions	11-2
Instruction Format.....	11-2
AXIS/EVENT	11-4
AXIS/HOME	11-5
AXIS/INITIALIZE	11-6
AXIS/MOVE (Single Axis, Non-Coordinated.....	11-7
AXIS/RATIO.....	11-8
AXIS/SPINDLE (Continuous Velocity Mode	11-9
AXIS/START	11-10
AXIS/STOP	11-11
AXIS/WAIT (Axis Wait For In-Position).....	11-12
CALL (retval = CALL).....	11-13
CAM/ACTIVATE	11-14
CAM/ADJUST	11-15
CAM/BUILD.....	11-16
CAM/INDEX	11-18
CAM/STATUS.....	11-19
CAPTURE/ENABLE.....	11-20
CAPTURE/SETUP.....	11-21
DATA/SIZE.....	11-22
DEFINE (Label a Variable)	11-23
DELAY (Suspend Task Execution).....	11-24
ELS/ADJUST (Adjust ELS Axis)	11-24
ELS/GROUPM	11-26
ELS/GROUPS.....	11-27
ELS/MODE (Set ELS Axis Mode)	11-28
ELS/MASTER	11-29
EQU (Equate)	11-29
EVENT/DONE (Signal Event Completed)	11-30
EVENT/END (Mark End of Event)	11-31
EVENT/START (Start of Event function)	11-31
EVENT/TRIGGER (Trigger a Task Event).....	11-31
EVENT/WAIT (Pause Task for Event Done Signal)	11-32
FUNCTION/ARG.....	11-33
FUNCTION-END	11-34
FUNCTION/START.....	11-34
GOSUB (Go To Subroutine)	11-35

GOTO (Go To Mark)	11-36
IF (If-Else-Endif Conditional Branch)	11-37
KINEMATIC (Use a Kinematic Definition for a Task)	11-39
LOCAL/VARIABLE.....	11-40
MESSAGE/DIAG (Task Diagnostic Message Definition)	11-41
MESSAGE/STATUS (Task Status Message Definition).....	11-42
MOVE/CIRCLE (Coordinated Move with Circular Interpolation)	11-43
MOVE/JOINT (Coordinated Move Joint Point to Point).....	11-44
MOVE/LINE (Coordinated Move with Straight Line Interpolation).....	11-45
PARAMETER/BIT (Initialize Parameter Bit)	11-46
PARAMETER/GET (Load Parameter to a Variable)	11-47
PARAMETER/INIT (Initialize a Parameter)	11-48
PARAMETER/SET (Set a Parameter).....	11-49
PATH/ABORT (Aborts Coordinated Motion).....	11-50
PATH/POSITION (Get Current Path Absolute Position)	11-51
PATH/RESUME (Resume Coordinated Motion).....	11-52
PATH/STOP (Halt Coordinated Motion)	11-52
PATH/WAIT (Pause Program for Motion)	11-53
PID/CONFIG	11-54
PLC/CLEAR (Clear I/O Register Bit)	11-56
PLC/READ (Read I/O Register(s))	11-56
PLC/SET (Set I/O Register Bit).....	11-57
PLC/TEST (Test I/O Register Bit)	11-58
PLC/WAIT (Pause Program for I/O)	11-59
PLC/WRITE (Write to I/O Register(s)	11-59
PLS/INIT.....	11-60
REGISTRATION	11-62
RETURN (Return From Subroutine).....	11-63
ROBOT/ORIGIN	11-63
ROBOT/TOOL	11-64
ROTARY/EVENT	11-64
SEQUENCER	11-65
SEQ/LIST	11-66
SEQ/STEP	11-67
TASK/AXES (Task Axes Definition).....	11-68
TASK/END (Mark the End of a Task)	11-69
TASK/START (Define the Start of a Task(s))	11-70
VAR/INIT	11-71
VIRTUAL/MASTER	11-72

12 Direct ASCII Communication

12-1

12.1 Overview	12-1
12.2 VisualMotion Communication Protocol	12-1
Reading Data from VisualMotion	12-2
Writing Data to VisualMotion.....	12-2
Communication Errors	12-2

Checksum	12-3
End of Message	12-3
Backspaces and White Spaces	12-4
Numeric Data Formats.....	12-4
Format of Data Sent to VisualMotion.....	12-4
12.3 Command Classes/Subclasses	12-5
Parameters	12-5
Variables	12-5
PID	12-6
Point Tables	12-6
Event Tables	12-7
Program Communication	12-7
Functions.....	12-8
I/O Registers	12-8
Sequencer Data	12-8
PLS	12-9
Zones	12-9
12.4 VisualMotion and Drive Parameters and Subclasses	12-10
Parameter Data Subclass	12-10
Name Text Subclass.....	12-10
Units Text Subclass	12-10
Upper Limit, L: Lower Limit Subclasses.....	12-10
Attribute Subclass	12-10
Parameter Lists Subclasses	12-11
SERCOS Parameter Sets.....	12-11
12.5 Parameter Lists	12-12
Listing a Parameter.....	12-12
Parameter List Block Transfer	12-13
12.6 User Program Variables.....	12-16
'P': Data.....	12-16
'T': Label Text.....	12-16
12.7 PID	12-17
Getting the assigned variables.....	12-17
12.8 Point Tables	12-18
Point Table Data	12-19
Point Table Data, Row Format	12-20
List of All User-defined Labels (V)	12-20
12.9 Event Tables	12-21
Event Table Data	12-21
Event Table Data, Row Format	12-22
12.10 VisualMotion's User Program Communication	12-23
User Program Header Record	12-23
Download Block Size	12-23
Activating a Program (PA)	12-23
Request Currently Active Program (PA)	12-24
Executing a Download (PD).....	12-24

Executing an Upload (PD)	12-25
Erasing (Deleting) a Program (PE)	12-25
List of Event Function Marks (PF)	12-26
List Control Resident Programs (PH)	12-27
CAM Indexer Function Block Variables (PJ).....	12-27
GPP Flash Compression (PK)	12-28
Registration Block Information (PM)	12-29
Request Name of Program (PN).....	12-29
Initializing an Upload (PR)	12-30
Selective Table Transfer Between Programs (PT)	12-30
List Variable Labels (PV)	12-31
Initializing a Download (PW).....	12-32
Transfer All Tables Between Programs (PX).....	12-32
Sequencer/Subroutine Related Subclasses (PI, PL, PQ, PS).....	12-33
12.11 Function Classes.....	12-34
K Class - ELS Functions	12-34
S Class.....	12-34
12.12 Input/Output Registers	12-35
I/O Register Access (RB), (RX), (RD).....	12-36
Erase All Forcing Masks (RE).....	12-37
I/O Forcing Selection (RF)	12-37
Set Current I/O State with Mask (RM)	12-38
I/O Binary Forcing State (RS)	12-38
Register Labels, Bit Labels (RT).....	12-39
12.13 Sequencer Data	12-39
Sequence List Class (L).....	12-39
Sequence Table Class (Q).....	12-40
12.14 PLS.....	12-42
Format of Configuration Data.....	12-43
Format of Switch Values	12-43
Drive PLS ASCII Protocol	12-44
12.15 Zone Tables	12-45
Format of Printed Data.....	12-45
Data in Row Format.....	12-46

13 Index

13-1

14 Service & Support

14-1

1 VisualMotion 8.0 Overview

1.1 System Overview

VisualMotion is a programmable multi-axis motion control system capable of controlling up to 32 digital intelligent drives from Rexroth Indramat. The PC software used for motion control management is named VisualMotion Toolkit. The hardware used with VisualMotion 8 GPP firmware is the PPC-R control.

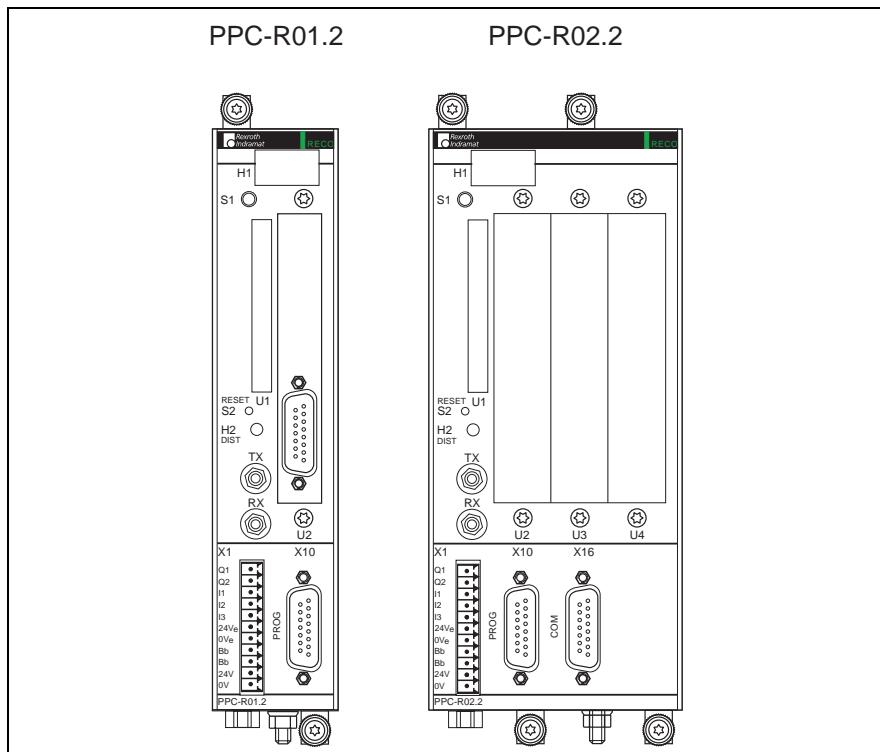


Fig. 1-1: PPC-R Motion Control

GPP 8.0 System Overview

The PPC-R is a stand-alone multi-axis motion control. It has the RECO02 form factor, a form factor used by Rexroth Indramat 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 Rexroth Indramat's DIAX04 and/or ECODRIVE03 digital servo drives. The communication between control and digital servo drives is performed using the SERCOS fiber optic interface, the international standard for real-time communication.

VisualMotion can provide multi-axis coordinated or non-coordinated motion control with tightly integrated RECO02 I/O logic control functions. The flexibility of GPP firmware supports a variety of applications, from general motion control to sophisticated multiple master electronic line shafting (ELS) and robotics.

System Components

The VisualMotion 8 system is composed of the following components:

- PPC-R control using GPP 8 firmware
- RECO02 I/O modules
- VisualMotion Toolkit (VMT) Windows program for motion control programming, parametrization, system diagnostics and motion control management. VMT also includes a DDE Server (communication protocol between Windows programs and the control).
- DIAX04 (with SSE03 or ELS05 firmware) or ECODRIVE03 (SMT01, SMT02, SGP01 or SGP03 firmware) drives and motors. Up to 32 intelligent digital drives can be connected to one control over the SERCOS fiber optic ring.
- HMI interfaces (BTC06, BTV04, BTV05, BTV06)

PLC Support

The Rexroth Indramat MTS-R is a PLC that interfaces with the VisualMotion system and is available preconfigured in two sizes.

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

The expansions slot(s) can be configured with, e.g., fieldbus or serial interface cards.

Interface Support

VisualMotion GPP 8 supports the following interfaces:

- Fieldbus Interfaces:
 - Profibus-DP slave interface(32 words)
 - DeviceNet slave interface (32 words)
 - Interbus slave interface (14 words)
 - ControlNet slave interface
- Additional Interfaces:
 - Ethernet Interface
 - Optional Programmable Limit Switch Card (16 or 32 outputs)
 - Link Ring for Master/Slave interfacing of PPC-R controls

2 VisualMotion's I/O System

VisualMotion's I/O system is used for operating functions such as System, Task and Axis control. I/O is also used to track system status and diagnostics. The VisualMotion user program instructions and remote I/O systems can all read and write I/O data. VisualMotion uses an internal block of memory to manage I/O systems. The memory is arranged in a linear array of "Registers" which are 16 bits wide. Each bit can have a value of 1 or 0 which corresponds with its "On" or "Off" state. VisualMotion's control memory allocation allows for a maximum of 512 I/O registers for use in the PPC-R.

2.1 I/O Devices Supported in GPP

The GPP firmware installed in the PPC-R hardware supports the following I/O devices:

- Local RECO02 I/O modules
- Remote I/O Stations (SERCOS RECO02 I/O modules)
- Remote SERCOS drive I/O cards
 - DEA04, DEA05 and DEA06 (15 inputs and 16 outputs)
 - DEA08, DEA09 and DEA10 (32 inputs and 24 outputs)
- Remote Analog input drive I/O cards
 - DAE02.1 14-bit analog inputs

2.2 I/O Configuration Tool

VisualMotion allows the user to commission or setup I/O devices supported in GPP using the *I/O Configuration* tool. The configuration of I/O devices can be performed either offline or online.

Note: All drive-based digital DEA and analog I/O cards cannot be detected and must be configured manually.

The I/O Configuration tool is launched by selecting ***Commission⇒ I/O Setup***.

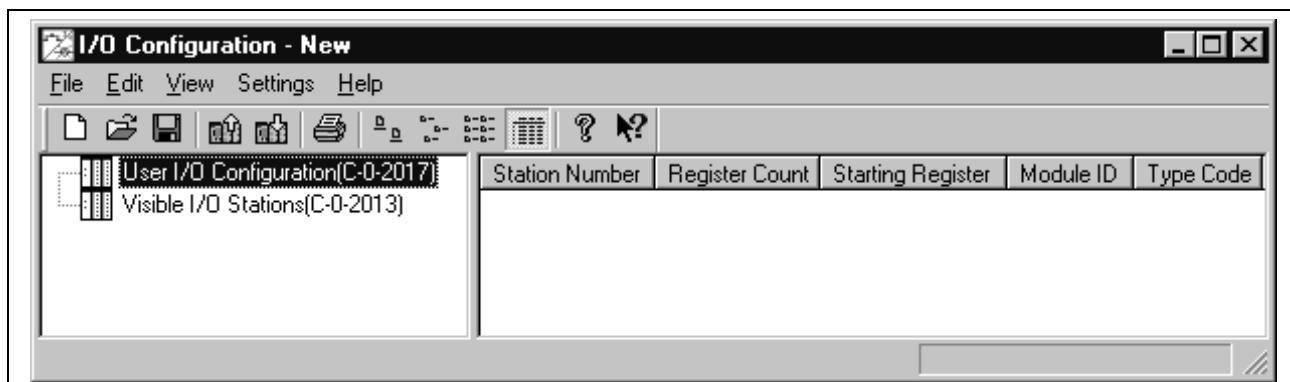


Fig. 2-1: I/O Configuration Tool

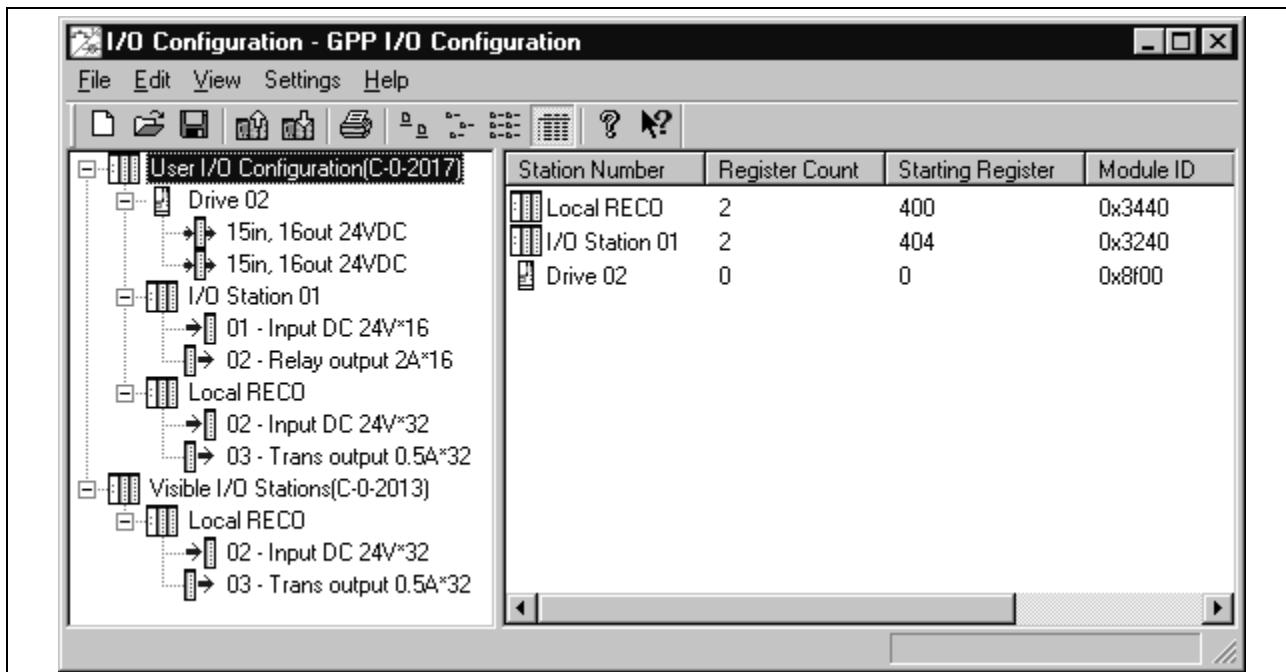


Fig. 2-2: I/O Configuration Tool

The I/O configuration window uses a tree-structure similar to Windows™ Explorer. The example I/O Configuration in Fig. 2-2 is divided up in to the following two main branches...

- User Configuration (C-0-2017)
- Visible I/O Stations (C-0-2013)

Each branch is associated with a corresponding GPP control parameter. The *User Configuration* branch can contain all RECO I/O read from parameter C-0-2013 as well as any configured drive-based I/O. The *Visible I/O Station* branch will only display any local RECO as well as any SERCOS RECO I/O from parameter C-1-2013.

I/O Initialization

During SERCOS phase initialization of GPP firmware, RECO I/O modules are scanned and processed in the following manner.

- Phase 2 – all RECO I/O modules are scanned and written to control parameter **C-0-2013, I/O Configuration List**.
- Phase 2 ⇒ Phase 3 – all RECO I/O modules are re-scanned and compared to card parameter C-0-2013.

Configuring I/O Devices Offline

VisualMotion's I/O Configuration tool allows the user the ability of configuring the I/O devices that will be used without actually being connected to the system.

Note: When configuring offline, I/O devices can be configured in any order. You can configure your drive's I/O first, then your Local and Remote RECO02 modules, or your RECO02 modules first. The order doesn't matter.

To configure I/O devices offline, use the following steps:

1. Start VisualMotion and select **Commission ⇒ I/O Setup** to open the I/O Configuration tool.

Note: I/O devices are first configured under the User I/O Configuration icon and then downloaded to the control by clicking on the download icon (). The control must be in parameter mode (Phase 2) to download.

2. Configure a Local RECO station by right-clicking on "User I/O Configuration" and selecting "Add an I/O Station". Refer to Local RECO02 I/O Configuration (Offline on page 2-10).
3. Configure an I/O Station (Remote) as in step 2. Refer to Remote I/O Station Configuration on page 2-13.
4. Configure a DIAX03/04 digital drive with I/O cards by right-clicking on "User I/O Configuration" and selecting "Add SERCOS Drive with I/O". Refer to Drive I/O Configuration on page 2-14.
5. Save your user I/O configuration and download it to the control.

Configuring I/O Devices Online

Before I/O devices can be configured, the control should be directed to visibly detect installed RECO02 modules. At power-up, VisualMotion automatically detects any installed Local RECO02 modules and identifies them in control parameter C-0-2013.

To configure I/O devices online, use the following steps:

1. Start VisualMotion and select ***Commission*** \Rightarrow ***I/O Setup*** to open the I/O Configuration tool.
2. Upload the Visible I/O Stations by selecting ***File*** \Rightarrow ***Get I/O Configuration from GPP*** or click on the upload icon (). Refer to Local RECO02 I/O Configuration (Online) on page 2-11.

Note: If a previously configured I/O file exists on the control, it will be displayed under the "User I/O Configuration" icon. The Visible I/O Stations will only display installed RECO02 I/O modules (Local and Remote).

3. Configure an I/O Station (Remote) as in step 2. Refer to Remote I/O Station Configuration on page 2-13.
4. Configure a DIAX03/04 digital drive with I/O cards by right-clicking on "User I/O Configuration" and selecting "Add SERCOS Drive with I/O". Refer to Drive I/O Configuration on page 2-14.
5. Save your user I/O configuration and download it to the control.

2.3 I/O Configuration Menus

File Menu

VisualMotion 8 uses the following standard Windows™ file commands:

<u>New</u>	Creates a new document.
<u>Open...</u>	Opens an existing I/O configuration (*.prm) file.
<u>Save</u>	Saves the currently open I/O configuration file.
<u>Save As...</u>	Saves the current configuration file after a name is entered using the *.prm file format.

The bottom half of the file menu displays the four previously opened pages as a shortcut to them.

<u>Exit</u>	Closes the GPP I/O Configuration window.
--------------------	--

Recent File	The last 4 files that were saved will be available for quick opening.
--------------------	---

Get I/O Configuration from GPP

When selected, control parameter C-0-2013 is read and the scanned I/O structure visible to the control is listed under the Visible I/O Stations.

Send I/O Configuration to GPP

When selected, the User Configuration CP-0-2017 is downloaded to the PPC-R. This file contains all visible I/O and any additional SERCOS and drive I/O configured by the user.

Printing

The following are standard Windows™ printing commands:

Print... Ctrl+P	Prints out the complete Visible I/O Stations and User I/O Configuration in a tree-structure.
Print Preview	Displays a copy of the printed page.
Print Setup...	Allows the user to modify printer settings.

Edit Menu

The Edit menu is used to make additions and modifications to the User I/O Configuration before downloading to the control.

Add a Drive

Adds a drive to the User I/O Configuration structure when “**User I/O Configuration (C-0-2017)**” is highlighted in the I/O Configuration window. Otherwise, this menu selection will appear grayed-out.

Add an I/O Station

Adds a RECO I/O Station to the User I/O Configuration structure when “**User I/O Configuration (C-0-2017)**” is highlighted in the I/O Configuration window. Otherwise, this menu selection will appear grayed-out.

Add a Drive I/O Card

Adds a drive I/O card to the selected drive under the User I/O Configuration (C-0-2017) structure. After a drive is added using the **Add a Drive** menu selection, highlight the drive and select this menu selection. Otherwise, this menu selection will appear grayed-out.

Add a RECO Module

Adds a RECO Module to the selected I/O Station under the User I/O Configuration (C-0-2017) structure. After an I/O Station is added using the **Add an I/O Station** menu selection, highlight the I/O Station and select this menu selection. Otherwise, this menu selection will appear grayed-out.

Note: The previously mentioned **Add** features are also available by right-clicking on the User I/O Configuration and any subsequent I/O Station and SERCOS Drive.



Copy “Visible Configuration” to “User Configuration”

This selection copies the visible configuration (C-0-2013) to the user configuration (C-0-2017). After the “**Get I/O Configuration from the PPC**” is selected from the file menu, it must be copied to the User I/O Configuration before additions can be made.

Assign registers to “User Configuration” I/O

This menu selection allows the user to add default starting registers to all configured I/O devices. When selected, a VisualMotion Message will be displayed warning the user that existing register numbers will be overwritten. This warning protects any configured I/O devices that have been downloaded to the control. If “Yes” is selected, the window in Fig. 2-3 will open allowing the user to accept the predefined register numbers or select different registers.

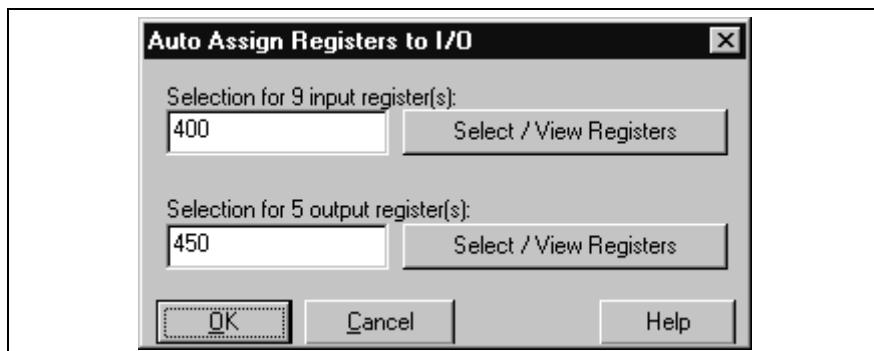


Fig. 2-3: Auto Assign Registers to I/O

View Menu

Toolbar

The Toolbar menu selection turns the icon toolbars on or off. A check to the right of the name indicates that the function is active. The icon toolbar contains standard Windows functions as illustrated in Fig. 2-4.

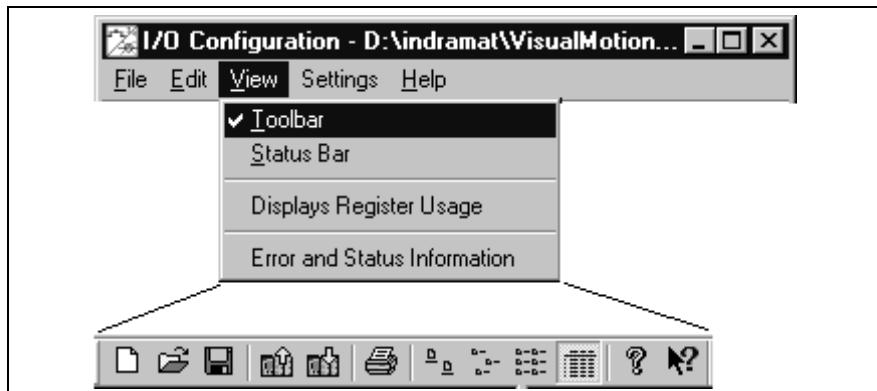


Fig. 2-4: I/O Configuration Toolbar

Status Bar

The Status Bar menu selection turns the status indicator at the bottom of the window on or off. A check to the right of the name indicates that the function is active. The status bar contains icon smart text to the left and process information, and downloading and uploading percentage to the right.

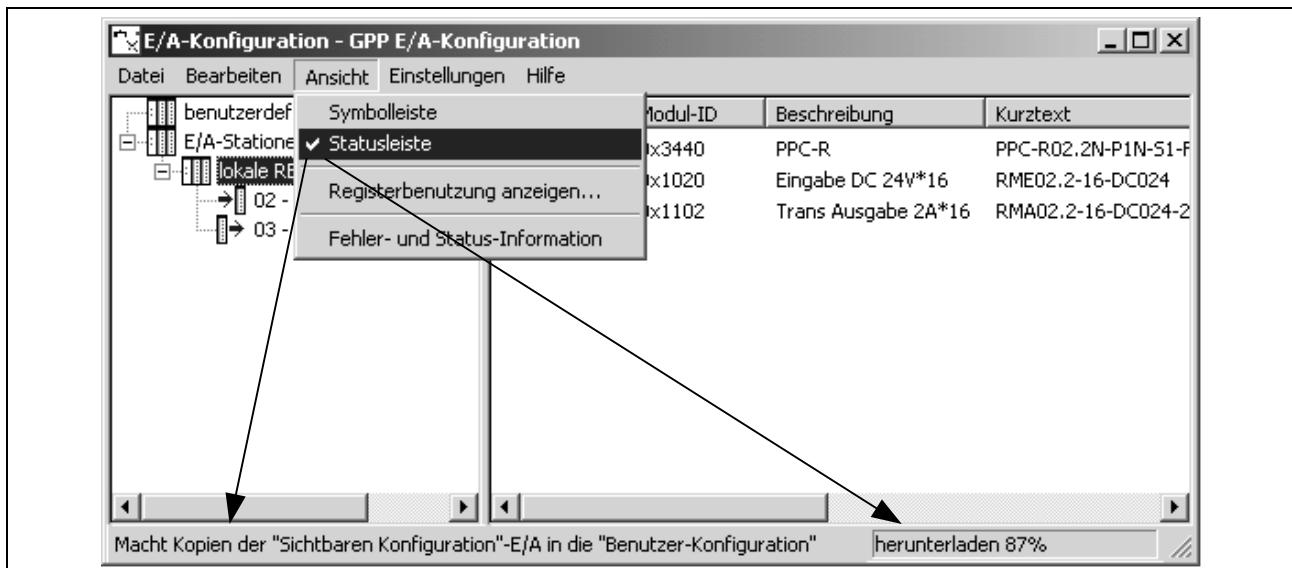


Fig. 2-5: I/O Configuration Status Bar

Display Register Usage

The Displays Register Usage selection 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 either an “In” for inputs or an “Out” for outputs. An asterisk is an indication that a register contains labels. Label names appear in the Register Label field at the bottom of the window as shown in Fig. 2-6.

Register Usage										x
	0	1	2	3	4	5	6	7	8	9
0		x		x	x	x	x	x	x	x
10	x	x	x	x	x	x	x	x	x	x
20	x	x	x	x	x	x	x			x
30	x	x	x	x	x	x	x	x	x	x
40	x	x	x	x	x	x	x	x	x	x
50	x	x	x	x	x	x	x	x	x	x
60	x	x	x	x	x	x	x	x	x	x
70	x	x	x	x	x	x	x	x	x	x
80	x	x	x	x	x	x	x	x	x	x
90	x	x				x	x	x	x	x
100	x									
110										
120	x	x								
130	In	In								
140	In	In								
150	Out									

Register Label: System_Control

Register_usage.tif

Fig. 2-6: I/O Configuration Register Usage

Error and Status Information

This window monitors all Local RECO and I/O Stations configured under User I/O Configuration that have been downloaded to the control. If all RECO I/O (Local or Stations) modules in the system are functioning properly, an OK status will be displayed. If a RECO I/O related error is encountered, the Status column will display the problem while the Slot Number column will indicate which slot contains the module in question, as shown in Fig. 2-7.

Error and Status Information			
Station Number	Status	Slot Number	Error
Local RECO	OK		

Error and Status Information			
Station Number	Status	Slot Number	Error
Local RECO	Missing 24V	3	

Fig. 2-7: Error and Status Information

Settings Menu

When settings is selected, the Card Selection Setup window in Fig. 2-8 will open. This setup window determines the method that will be used for communicating to the control.

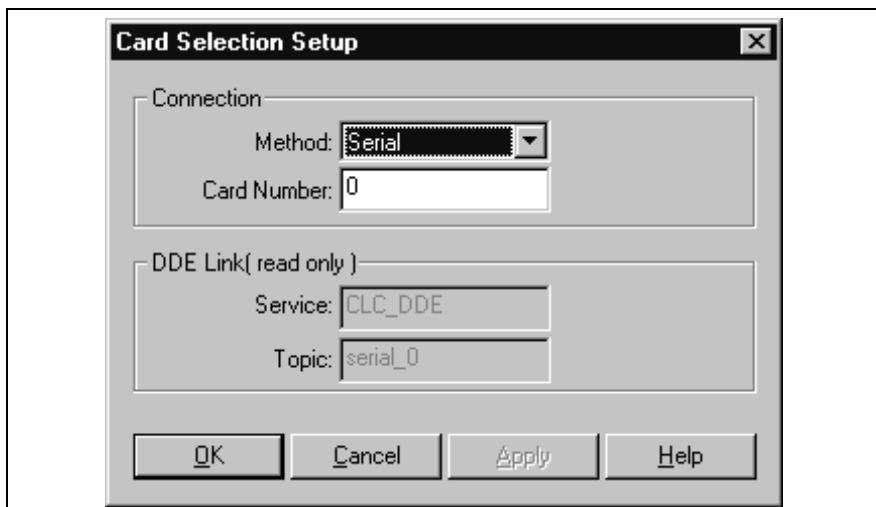


Fig. 2-8: I/O Configuration Card Selection Setup

Help Menu

The help menu contains a link to the help system as well as an About reco1... window where the current version and build date are displayed.

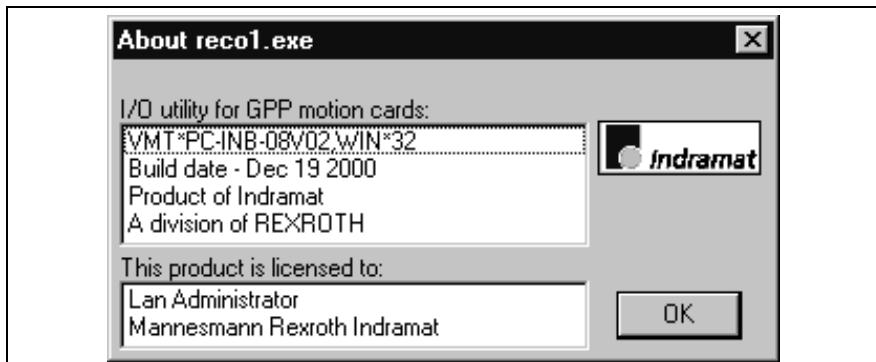


Fig. 2-9: About reco1.exe window

2.4 Local RECO02 I/O Modules

The PPC-R control is physically installed into a RMB02.2 rack designed to hold the control along with the RECO02 I/O modules. RMB02.2 racks are a 4 slotted mounting platform that provide power and backplane communication with installed modules. The PPC-R control can use up to 2 slots (PPC-R02.2) leaving the remaining slots available for I/O modules. The I/O modules that plug in to the same rack as that of the control are identified as Local RECO02 I/O modules. Any additional RMB02.2 rack (maximum of 4) containing I/O modules connected directly to the control's rack is also identified as Local RECO02 I/O modules. Refer to the *VisualMotion 8 (GPP) Project Planning Manual* for hardware details. Local RECO02 I/O modules are automatically detected by the control and identified in control parameter C-0-2013 (I/O Configuration List).

Registers can then be assigned to each I/O module by using the I/O Configuration tool.

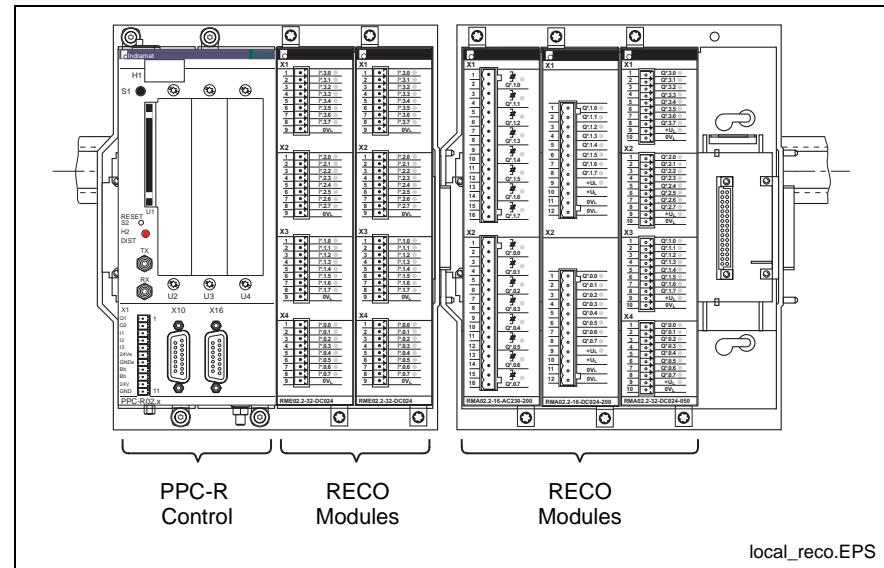


Fig. 2-10: Local RECO02 I/O

Note: Local RECO02 I/O and Remote I/O Stations are the only configurations that varies whether offline or online. When working online, both Local and Remote I/O can automatically be detected and uploaded to the I/O Configuration tool.

Local RECO02 I/O Configuration (Offline)

Use the following steps to configure Local RECO02 I/O modules offline:

1. Select **File** ⇒ **New** from the I/O Configuration tool.
2. Right-click on "User I/O Configuration" and select "Add an I/O Station".
3. Select the I/O Station Number as **Local RECO**.
4. Select the appropriate PPC-R type installed in the RMB rack.
5. Assign a register number to the Local RECO by clicking on the "Select/View Registers" button. Refer to **Assigning Register Numbers** on page 2-15.

Note: The first two registers that are assigned to the Local RECO are reserved for PPC-R error and status information. The next two registers are assigned to any installed RECO02 I/O modules. Refer to Fig. 2-7 for details.

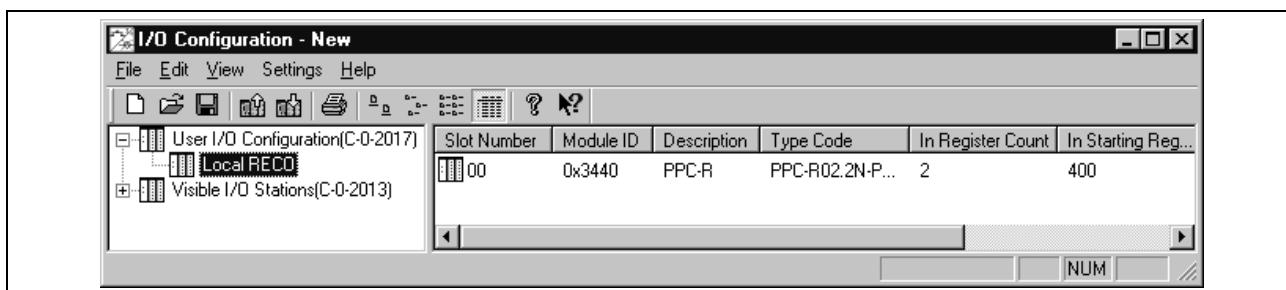


Fig. 2-11: Local RECO02 Configuration

Add an I/O Module

RECO02 I/O Modules can be added to Local RECO02 or Remote I/O Stations by using the following steps.

1. Right-click on Local RECO and select "Add an I/O Module" to display the *Add a RECO Module* window.
2. Add a RECO02 module by selecting the appropriate *Type* (radio button) and selecting the I/O module from the drop-down list. Refer to the following table for a listing of supported RECO02 I/O modules.

Type (Radio Button)	Selection (Drop-Down List)
Analog	RMC02.2-2E-1A (2 Inputs / 1 output)
Digital Input	RME02.2-16-DC024 (16 inputs)
	RME02.2-16-AC115 (16 inputs)
	RME02.2-32-DC024 (32 inputs)
Digital Output	RMA02.2-16-RE230-200 (16 outputs)
	RMA02.2-16-DC024-200 (16 outputs)
	RMA02.2-16-AC230-200 (16 outputs)
	RMA02.2-32-DC024-050 (32 outputs)

3. From the RECO Module Setup window, select the slot number where the module will reside. Refer to the *VisualMotion 8 Project Planning Manual* for proper slot addressing.
4. Assign a starting block register number to the RECO02 module by clicking on the "Select/View Registers" button. Refer to **Assigning Register Numbers** on page 2-15.

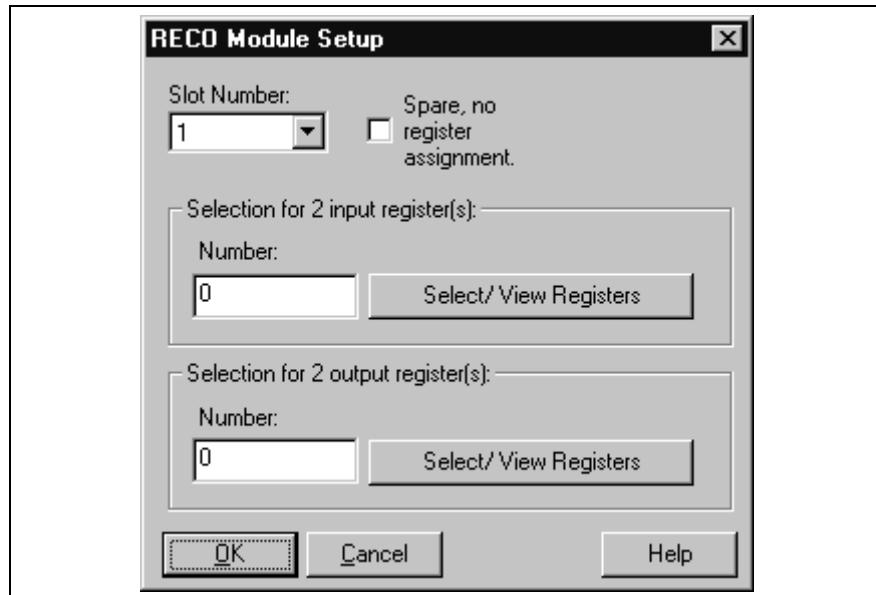


Fig. 2-12: RECO Module Setup Window

5. Repeat these steps until all Local RECO02 modules are configured.

Local RECO02 I/O Configuration (Online)

Local RECO modules must first be uploaded to the I/O Configuration tool before register numbers can be assigned to each module. Upload Visible I/O Stations (Local RECO) from the control using the following steps.

1. Select ***File*** \Rightarrow ***Get I/O Configuration from PPC*** or click on the upload icon ().
2. Select ***Edit*** \Rightarrow ***Copy "Visible Configuration" to "User Configuration"*** for assignment of register numbers.

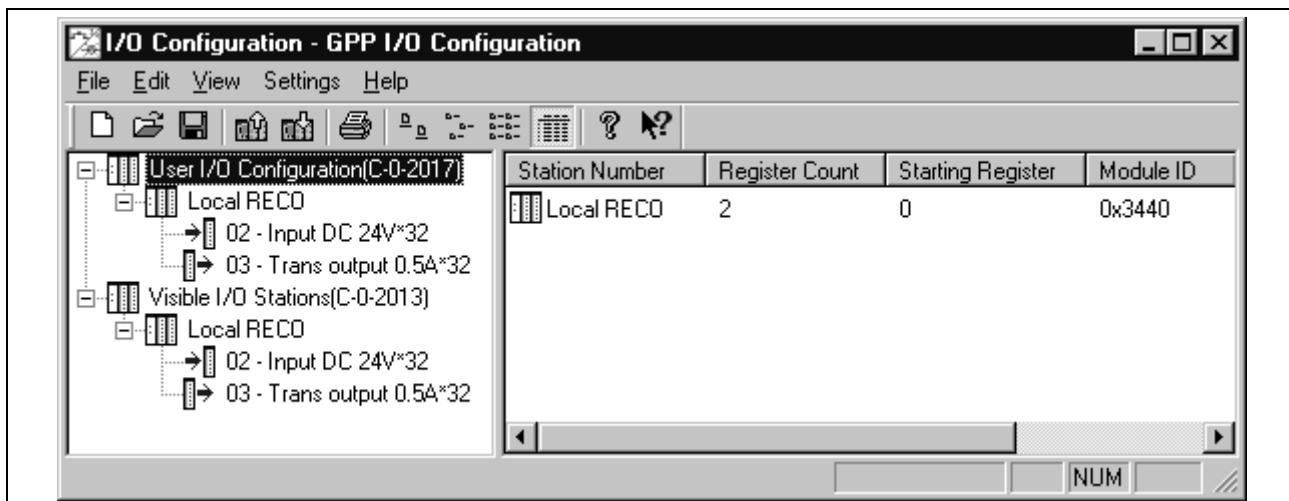


Fig. 2-13: Upload Visible I/O

Note: In this example a PPC-R02.2 control is installed in the RMB rack's first two slots (00 and 01) with a RME02.2 input module in slot 02 and a RMA02.2 output module in slot 03.

3. Double-click "Local RECO" under "User I/O Configuration" to open the I/O Station Setup window.
4. Select the appropriate PPC-R type installed in the RMB rack.

5. Assign the starting block register and click OK. Refer to Assigning Register Numbers on page 2-15.

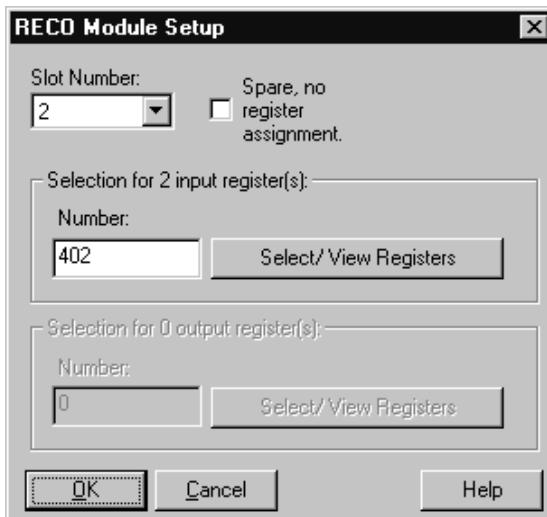


Note: The first two registers that are assigned to the Local RECO are reserved for PPC-R error and status information. The next two registers are assigned to any installed RECO02 I/O modules. Refer to Fig. 2-7 for details.

Configure RECO02 Modules

1. Double-click on the first RECO02 module and assign a starting block register number for the RME02.2. Refer to Assigning Register Numbers on page 2-15. This example is for an RECO02 input module.

Note: For RECO02 output modules, the "Selection for # output register(s)" will be displayed.



2. Repeat this process for any remaining RECO02 I/O modules.

2.5 Remote I/O Stations

An I/O station is a RMB rack configured with a remote SERCOS coupling unit (RMK02.2) and up to 3 RECO02 I/O modules connected to the control via the SERCOS fiber optic ring.

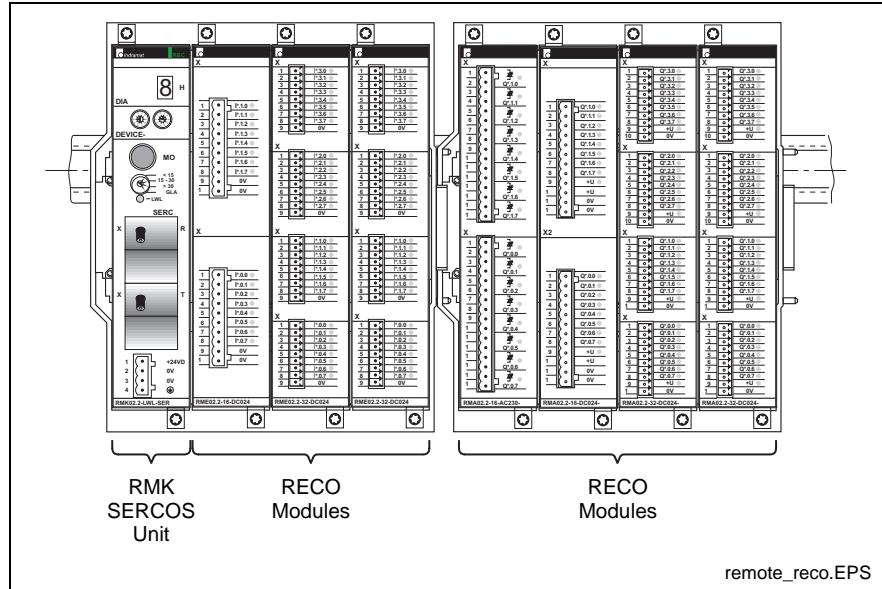


Fig. 2-14: Remote I/O Stations

Remote I/O Station Configuration

Add an I/O station to the User Configuration as follows:

1. Right-click on "User I/O Configuration" and select "Add an I/O Station".



2. Select an I/O Station number (SERCOS address).

Note: Refer to the VisualMotion 8 Project Planning Manual for proper "SERCOS Drive Address Settings" of the RMK02.2.

3. Assign a register number to the I/O Station by clicking on the "Select/View Registers" button. Refer to Assigning Register Numbers on page 2-15.

4. Add appropriate RECO02 I/O modules. Refer to "Add an I/O Module" on page 2-10.
5. Save your user I/O configuration and download it to the control.

2.6 Remote SERCOS Drive I/O Cards

Rexroth Indramat digital drives of the DIAX03/04 families can be configured to hold up to 3 digital I/O cards of the following types.

- DEA04, DEA05 and DEA06 (16 inputs and 15 outputs)
- DEA08, DEA09 and DEA10 (32 inputs and 24 outputs)

DEA digital I/O cards are configured as a "*SERCOS drive with I/O*" and assigned registers using the I/O Configuration tool.

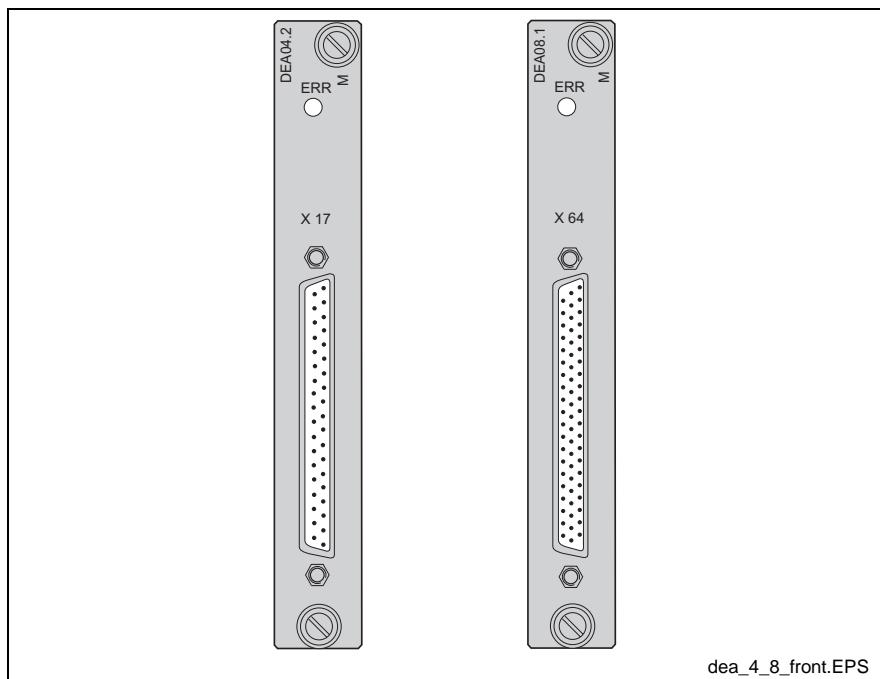


Fig. 2-15: Remote SERCOS Drive I/O Card (DEA)

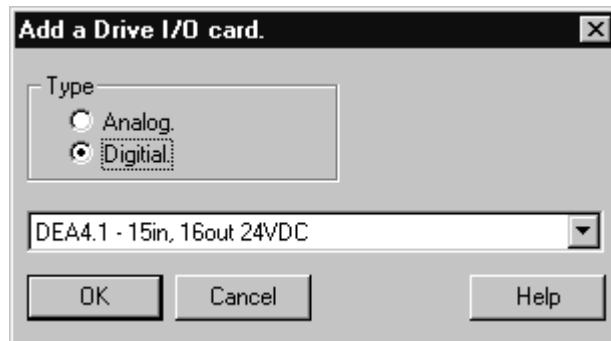
Drive I/O Configuration

Add a digital drive with I/O as follows:

1. Right-click on User configuration and "Add SERCOS Drive with I/O".
2. Select the Drive SERCOS Number (SERCOS address).

Note: Refer to the VisualMotion 8 Project Planning Manual for proper "SERCOS Drive Address Settings" of the DSS02.1.

3. Right-click on the drive added in step 2 and select "Add a Drive I/O Card".
4. Select an Analog or Digital I/O card.



5. Assign a register number to the I/O Station by clicking on the "Select/View Registers" button. Refer to Assigning Register Numbers on page 2-15.
6. Save your user I/O configuration and download it to the control.

Remote Analog Input Drive Cards

Rexroth Indramat digital drives of the DIAx03/04 families can be configured to hold an analog input card to receive an input from a feedback device for use in a VisualMotion user program. Allowable analog input cards are...

- DAE02.1 14-bit analog inputs

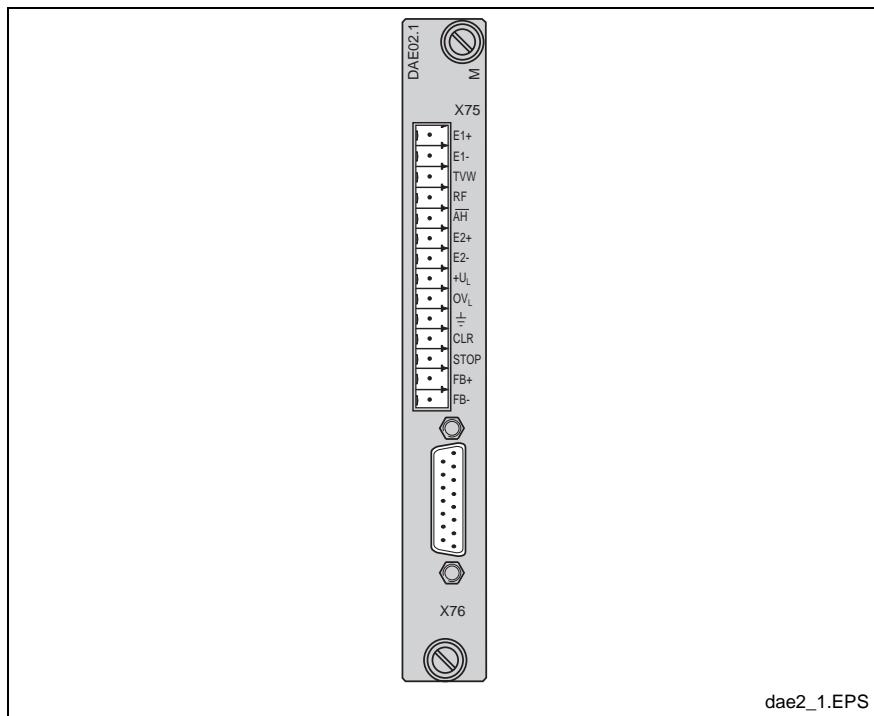


Fig. 2-16: DAE02.1 Analog Input Card

2.7 Assigning Register Numbers

Register numbers are assigned to external I/O devices in order to create a link to VisualMotion program instructions. 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. In order to make the selection and

assignments of registers in a system easier, the I/O Configuration tool has a "Select/View Registers" button as illustrated in Fig. 2-17.

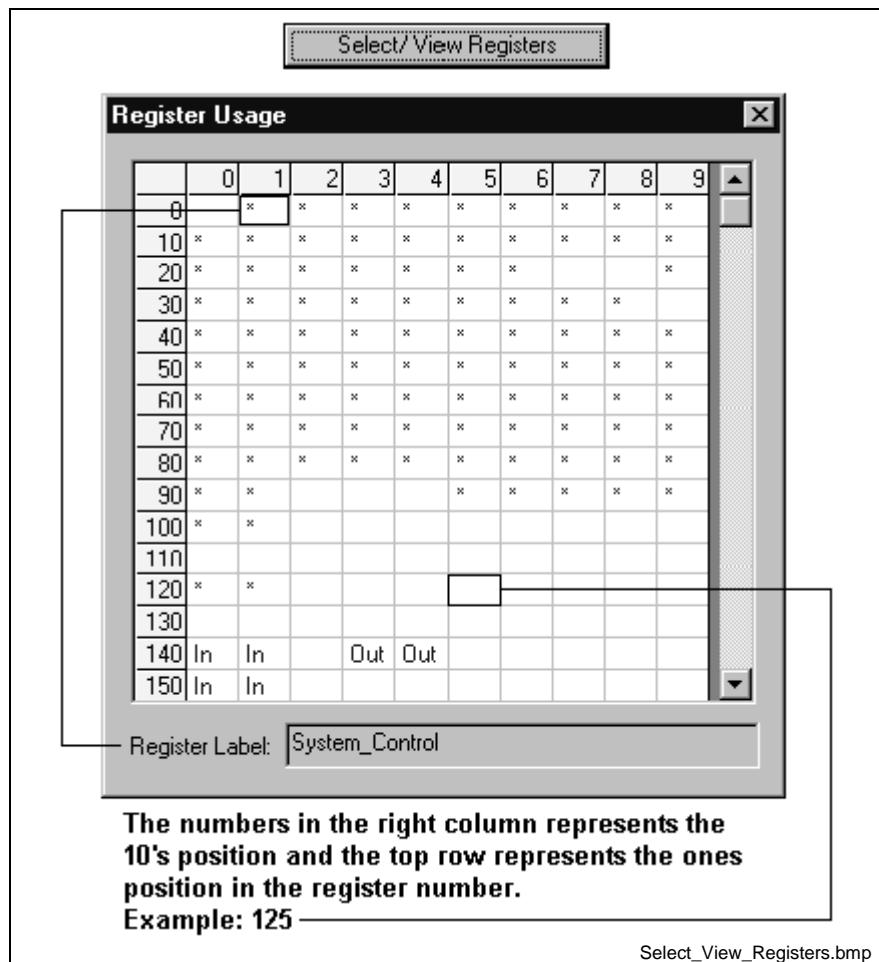


Fig. 2-17: Select/View Registers

Register Usage

The Register Usage window is designed to visually display all 512 registers in VisualMotion. The asterisk (*) represents an assigned label for the selected register. A register containing an asterisk may or may not have an assigned I/O function in the VisualMotion system. The label for the selected register is displayed in the grayed-out field at the bottom of the screen. Register labels can be assigned by selecting **Edit** ⇒ **Edit Labels** ⇒ **Edit Register Labels** from the VisualMotion Toolkit's main menu.

Note: When opened, the Register Usage window will display an asterisk for all registers in the currently open program containing a label.

Assigning a Register Starting Block

Registers are assigned to I/O devices, such as RECO modules or drive I/O as starting blocks. Some devices require more than one register for accessing I/O points. The starting block number simply represents the number at which the assignment begins. If a device require multiple registers, the Register Usage window will automatically assign the necessary consecutive registers to that device.

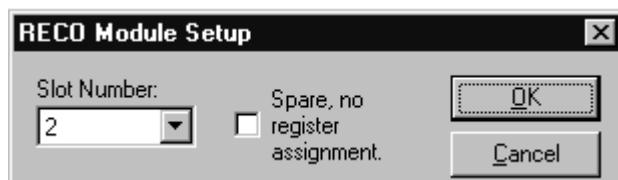
To select a register starting block for an I/O device, follow these steps

1. Click on the "Select/View Registers" button.
2. Locate an unused register with an unused consecutive register(s).
3. Double-click on the register number.

The register usage window will close and the starting block number will appear in the number field.

The next time the register usage window is opened, an "In" will appear for registers assigned to input modules and an "Out" will appear for registers assigned to output modules.

Note: A RECO02 I/O module can be installed as a spare for future use and not assigned to any register by checking the "Spare, no register assignment".



Required Registers Based on I/O Device

The number of used registers in a VisualMotion I/O system is based on the configured I/O device. Refer to the following table for an overview of required register numbers.

Description	Type	Number of Input Registers	Number of Output Registers
RME02.2-16	RECO02 Digital Input	1	0
RME02.2-32	RECO02 Digital Input	2	0
RMA02.2-16	RECO02 Digital Output	0	1
RMA02.2-32	RECO02 Digital Output	0	2
RMC02.2	RECO02 Analog Module	2	2
DEA04,05,06	Digital Drive I/O Card	1	1
DEA08,09,10	Digital Drive I/O Card	2	2
DAE02.1	Analog Input Card	2	0

Table 2-1: Required Registers Based on I/O Device

Configuring RECO02 I/O Modules for the PPC-R

VisualMotion communicates with RECO02 I/O modules by assigning registers to specific modules. The assignment of registers to I/O devices is performed using VisualMotion's I/O Configuration tool. Refer to the VisualMotion Functional Description manual for details.

This section describes how RECO02 I/O modules are assigned to VisualMotion registers. The labeling of RECO02 modules is illustrated in Fig. 2-18.

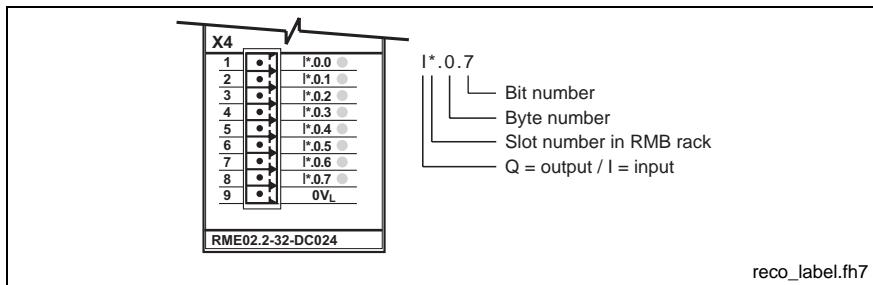


Fig. 2-18: RECO02 Bit Label

RECO02 16-Bit I/O Modules

The RME02.2-16 (Input) and RMA02.2-16 (Output) modules each use 1 VisualMotion register for accessing their respective 16 bits. The lower 8 VisualMotion register bits are assigned to the lower connector X2 (Byte 0) and the upper 8 bits are assigned to the upper connector X1 (Byte 1).

RECO02 32-Bit I/O Modules

The RME02.2-32 (Input) and RMA02.2-32 (Output) modules each use 2 VisualMotion registers for accessing their respective 32 bits. The first VisualMotion register (n) is assigned to the lower two connectors (X3 and X4). The second adjacent VisualMotion register (n+1) is assigned to the upper two connectors (X1 and X2). Bit distribution of each register is similar to that of the 16-bit input module.

Note: A VisualMotion register is equivalent to 2 Bytes or 1 Word. Each RECO02 module connector is identified as a Byte. RECO02 module connectors begin at Byte 0 (lowest connector) up to Byte 3 (in the case of a 32-bit module with 4 connectors).

The bits on a RECO02 module are labeled from 0-7 for the lower Byte and 0-7 for the upper Byte. VisualMotion assigns bit numbers 1-8 for the lower Byte and 9-16 for the upper Byte.

Fig. 2-19 illustrates the labeling structure of a RECO02 Input or Output module and the relationship to VisualMotion assigned register bits.



Accessing RECO I/O bits in VisualMotion 8 has been changed from VisualMotion 7.

⇒ Refer to Chapter 2 of the VisualMotion 7 Functional Description manual for details when accessing RECO I/O using VisualMotion 7.

VisualMotion n Register Definition

This document makes mention of a (*n*) register and a (*n + 1*) register. During the configuration of RECO02 I/O modules in VisualMotion's GPP I/O Configuration interface, a RECO Module Setup window is used to identify the Slot Number for the module as well as the register number that will be assigned to the inputs or outputs, if applicable.

The *Number* assigned to the inputs or outputs is considered the (*n*) register. The (*n + 1*) register is the next adjacent register number assigned to 32-Bit RECO02 modules. 16-Bit modules only use one register (*n*).

For example,

If register 400 is assigned to the inputs of a RECO02 I/O module, then register 401 is the (*n + 1*) register.

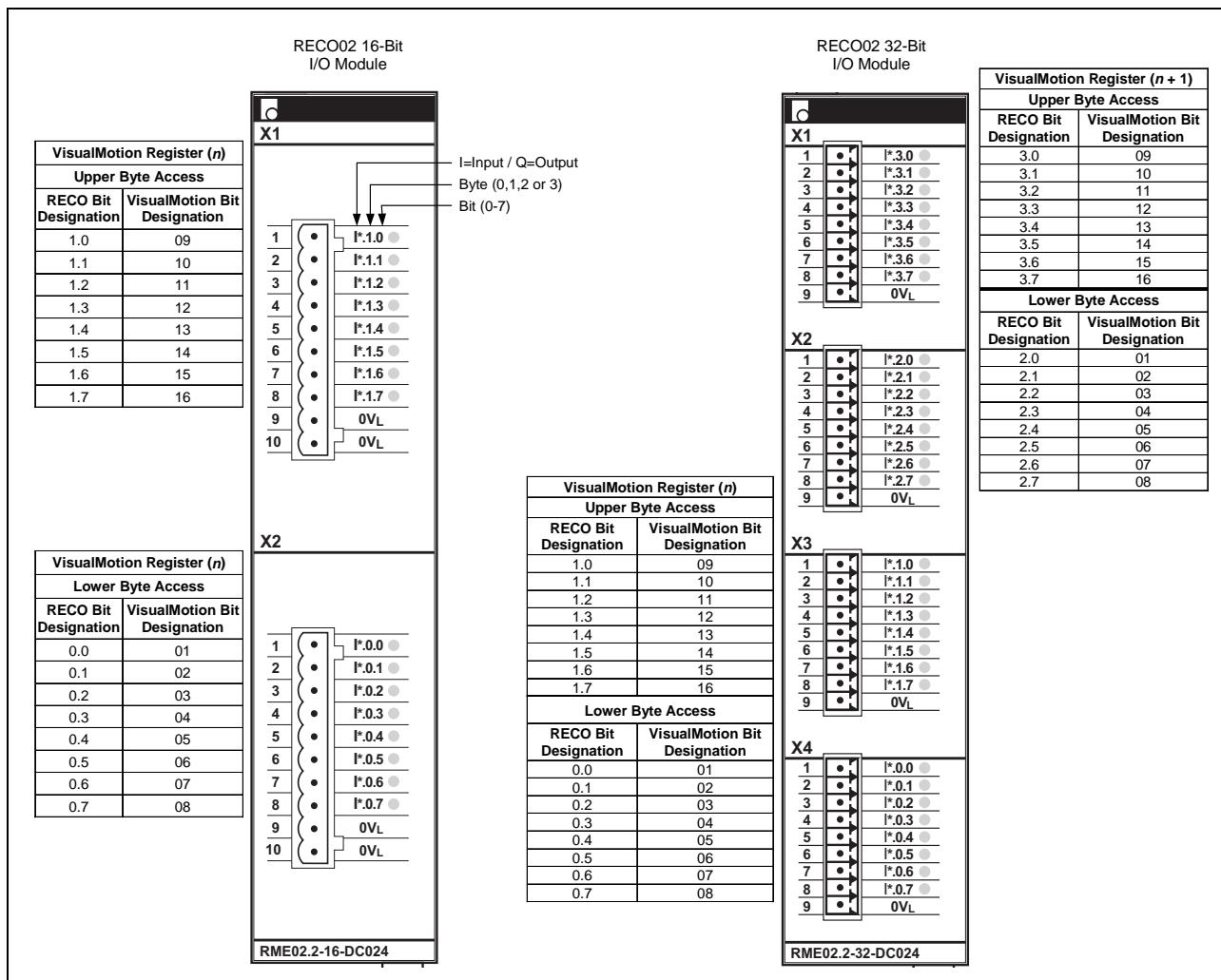


Fig. 2-19: RECO02 Digital Input and Output Module Bit Access

RMC02.2-2E Analog Input and Output Module

The RMC02.2 analog RECO02 module uses 2 VisualMotion registers for accessing the X1 and X2 input connectors and 1 VisualMotion register for accessing the X3 output connector. A second VisualMotion register is used for scaling the output value of X3. The first (n) VisualMotion input register is assigned to the upper X1 connector. The second ($n + 1$) adjacent VisualMotion input register is assigned to the center X2 connector. For output, the lower X3 connector is assigned to the first (n) VisualMotion output register. The second ($n + 1$) adjacent VisualMotion output register is reserved for the output scaling of connector X3. Refer to the *SERCOS I/O Unit RECO02.2* configuration manual for details.

Note: The scaling for the output is done in VisualMotion Toolkit's GPP I/O Configuration interface when configuration an analog RECO02 module.

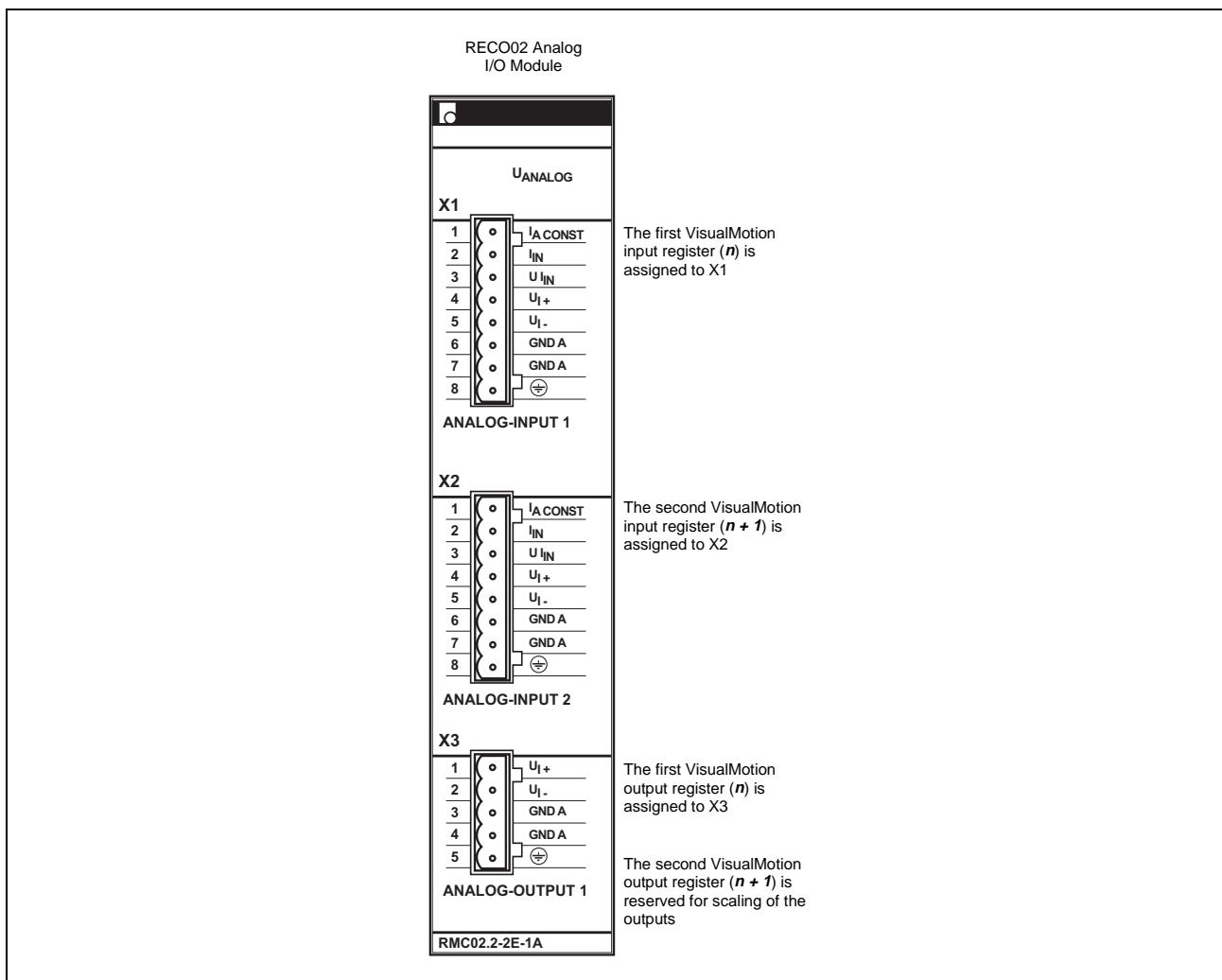


Fig. 2-20: RECO02 Analog Input and Output Module Bit Access

DEA Digital Drive I/O Cards

DIAX03/04 digital drives can hold up to 3 DEA I/O cards. The DEA04, 05 and 06 each require 1 register for 16 inputs and the 1 register for 15 outputs. The DEA08, 09 and 10 each require 2 registers for 32 inputs and 2 registers for 24 outputs. The pin-out for each card is illustrated in Fig. 2-21.

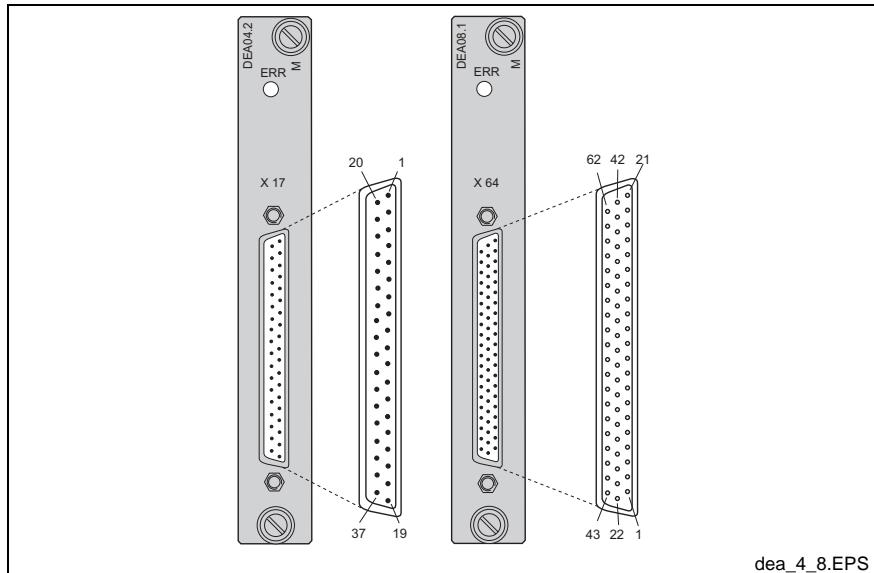


Fig. 2-21: DEA Digital Drive I/O Cards

Register	Bits	DEA04 Pin-out
405 (Input)	01 - 15	01 - 15
453 (Output)	16 - 31	16 - 31

Fig. 2-22: DEA04 Digital I/O Pin-out

Register	Bits	DEA08 Pin-out
406 (Input)	01 - 11	01 - 11
406 (Input)	12 - 16	22 - 26
407 (Input)	01 - 06	27 - 32
407 (Input)	07 - 16	43 - 52
454 (Output)	01 - 08	12 - 19
454 (Output)	09 - 16	33 - 40
455 (Output)	01 - 08	53 - 60

Fig. 2-23: DEA08 Digital I/O Pin-out

3 Enhanced I/O Mapper

VisualMotion's I/O Mapper is used to map physical I/O lines to control and status registers using a ladder logic and Boolean equations. A simple PLC functionality is included, with AND, OR, and NOT functions. The I/O Mapper program is stored as Boolean equations on the control. The I/O Mapper program interpreter executes all Boolean equations every I/O Mapper scan time (2ms or 4ms) in all modes, independent of the User Tasks. This allows a deterministic response to critical I/O.

The scan time is set by selecting **Settings** ⇒ **Scan Time** from the I/O Mapper's main menu.

The I/O Mapper provides the following 3 windows for the creation and manipulation of an I/O file.

- Rung Select - navigate between rungs
- Ladder - create graphic Ladder logic
- Boolean Equations - view and modify Boolean equations

The windows can be Cascaded or Tiled from the **Window** menu selection. By default, the Rung Select and Ladder windows are initially displayed when a new file is created or opened. To display the Boolean Equations window select **Window** ⇒ **Boolean Equations**.

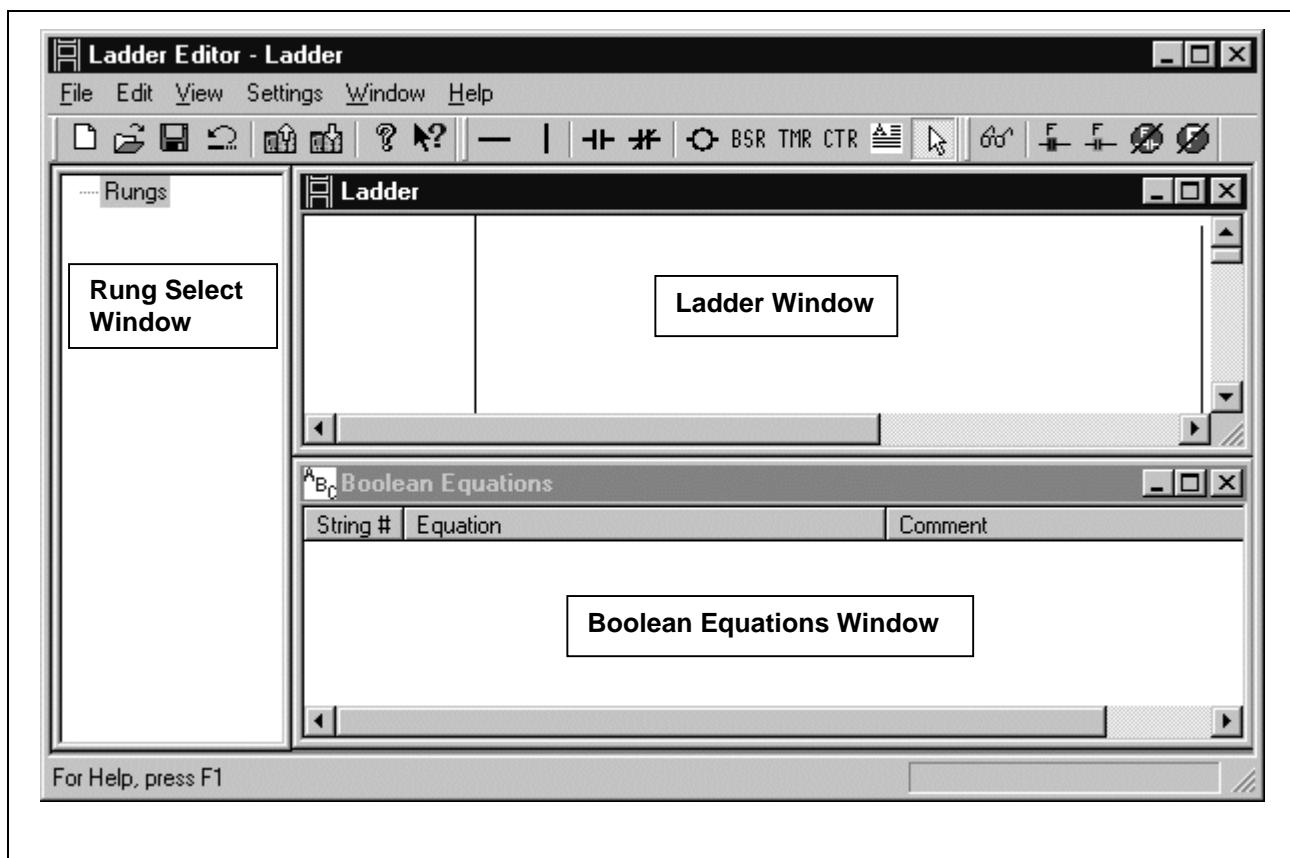


Fig. 3-1: I/O Mapper Windows

3.1 I/O Mapper Specifications

The I/O Mapper specifications mentioned in this section are provided as a reference to the user and are as follows:

- Rung Specifications
- Maximum Rows Allowed in Ladder window
- Total Operations

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.

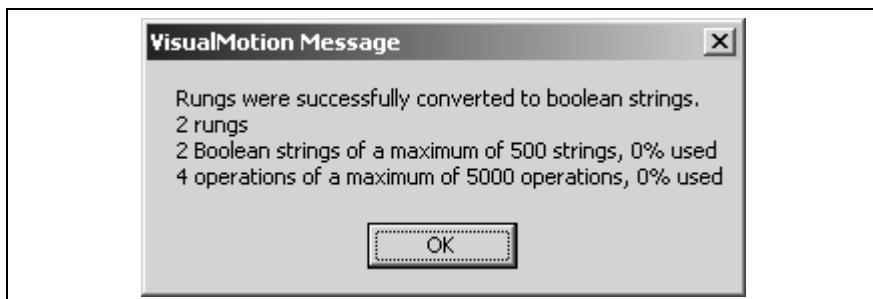


Fig. 3-2: Memory Usage

Rung Specifications

The size and number of rungs within the Ladder window is determined by the following:

- Ladder Matrix Size
- Boolean String Length

Ladder Matrix Size

A rung in the Ladder window can have a "7x7" matrix for input contact logic. A "7x7" rung matrix is limited by the following:

- 7 contacts per row
- 7 rows per rung
- 160 characters per rung

The Ladder window can display a rung with 7 rows containing 7 contacts per row, for a total of 49 contacts. However, this would exceed the 160 characters allowed per rung. Fig. 3-3 displays a small matrix example.

Note: Each contact displays an ID in blue and a label in orange (if the label is defined). The number of characters in the ID is part of the string, but not the label.

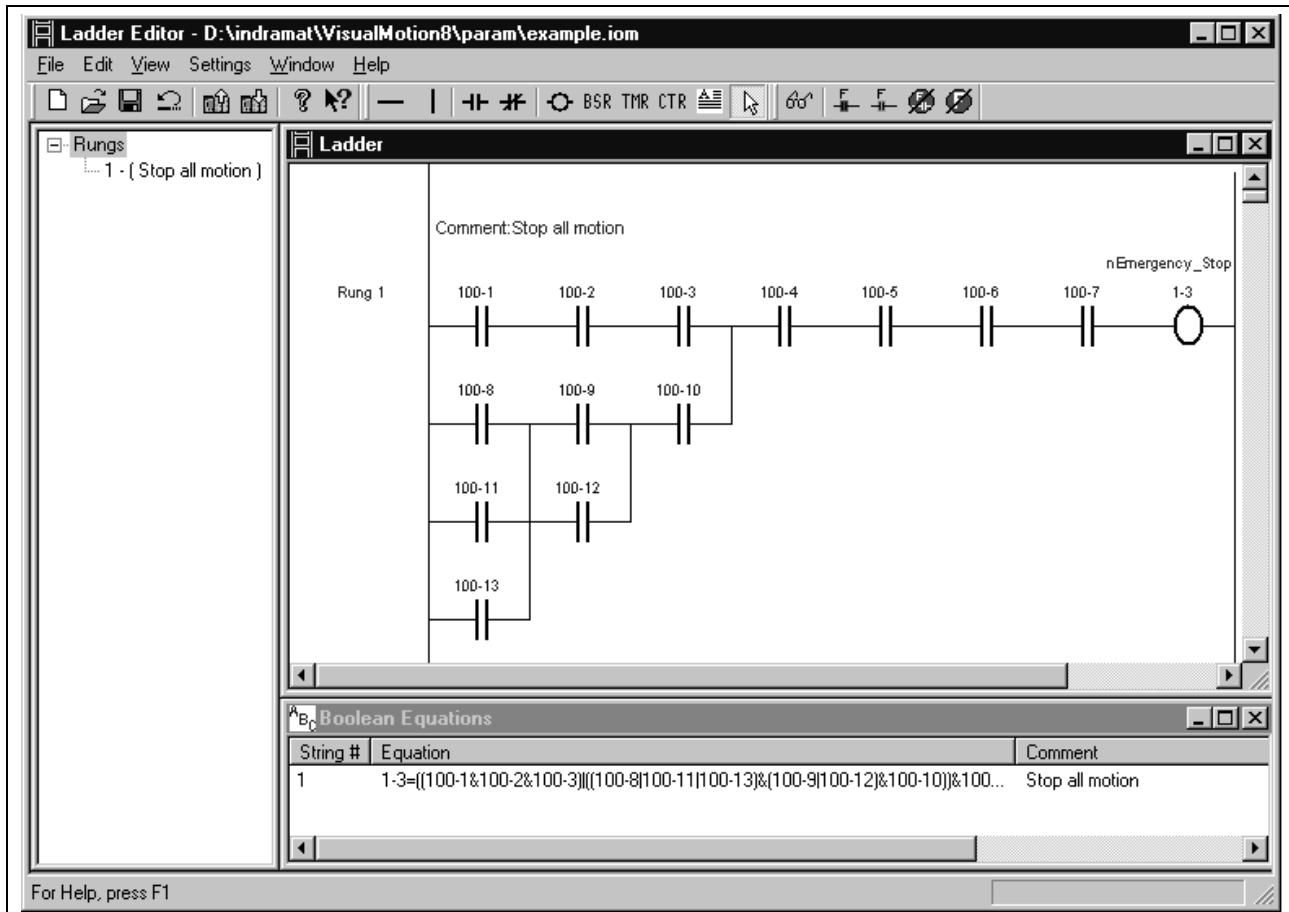
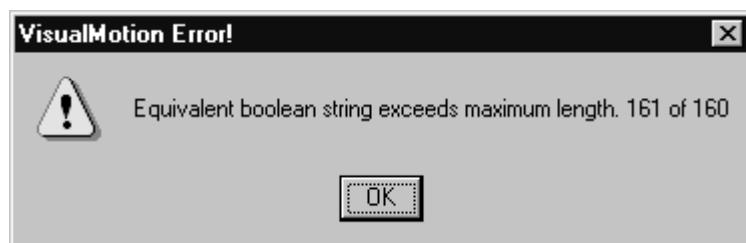


Fig. 3-3: Matrix Example

Boolean String Length

Boolean strings are displayed by selecting **Windows** \Rightarrow **2 Boolean Equations**. The maximum character length for a rung's Boolean string is 160 characters. The Boolean string representing the rung displays the ID along with the operations [-,), (, &, |, !, =] for the string. If the maximum characters for a rung's Boolean string is exceeded, the following VisualMotion Error will be displayed when the rungs are converted to Boolean strings.



The maximum length for comments is 80 characters.

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 operations
- 14 characters in the comment, including the semicolon and null termination

Note: A maximum of 500 Boolean strings can exist in an I/O Mapper file.

Maximum Rows Allowed in Ladder Window

The rows in the Ladder window have the following specifications:

- Maximum number of rows
- Spaces between rows

Maximum Number of Rows

The maximum number of rows allowed in the Ladder window is 1000. A row in the Ladder window can consist of any of the following:

- A comment for a rung
- A single line rung
- Each line in a multiple line rung matrix
- Each line in an output function, such as a counter

Spaces between Rows

Spaces between rows are automatically removed and the next rung or comment is moved up when the I/O Mapper is saved or downloaded to the control.

Total Operations

Operations in the Boolean string, processed by the control, are used to create a mathematical equivalent to the Ladder logic. The total number of operations is limited to a maximum of 5000. This is limited to ensure that all the operations are processed during one scan. These operators are listed in the following table.

Character Symbol	Description
(open parenthesis
)	closed parenthesis
=	equal to
&	AND statement
!	NOT statement
	OR statement

Table 3-1: Boolean Operators

3.2 Creating and Downloading a New I/O Mapper

To create a new I/O Mapper, follow these basic steps:

1. From VisualMotion's main menu select, **Commission → I/O Mapper** to open the Ladder Editor window.
2. Select **File ⇒ New** to open a new Ladder window or click on the  New icon.

Note: The Ladder logic icon toolbar (above the Ladder window) is active when the Ladder window is selected. When the Rung Select window is selected, the Ladder logic icon toolbar is grayed-out.

3. From the Ladder icon toolbar, click once on the desired icon. Next, move the cursor to the Ladder window. The cursor will change to the icon type selected.
4. Place contacts to the left of the view area and coils to the right. The cursor will change to this symbol () when attempting to place an icon in the wrong location.

Note: Each rung in the Ladder Window can contain up to 7 (input logic) contacts. However, only one coil or output function is allowed per rung. Do not leave blank rows between rungs. Refer to Spaces between Rows for details.

5. When an icon is placed, a Properties window will open. Choose the register and bit or modify the contact type under the Contact Setup tab. Refer to Input Logic Functions for details.

Note: Select the Comment icon () to enter a description for the rung. The maximum number of characters for a comment is 80.

6. When all the rungs have been created, select **File ⇒ Send Ladder to control** or click on the download icon () to download the new I/O Mapper to the control.

Note: The control must be in parameter mode to download a file. The file can also be saved to a hard drive for downloading at a later date.

Note: A serial connection, using cable IKB0005, is necessary to download an I/O Mapper file to the control.

3.3 Downloading the Default I/O Mapper

During the initial installation of VisualMotion, a default I/O Mapper file (*Def100.iom*) is installed under the `\indramat\VisualMotion8\param` folder. Follow these steps to open and download the default I/O Mapper to the control.

1. Start the I/O Mapper and select, **File** \Rightarrow **Open**, locate the Def100.iom file and click on Open. The Ladder representation of the file will open in the Ladder window.
2. To download the I/O Mapper to the control, select **File** \Rightarrow **Send Ladder to control** or click on the download icon (). The control must be in parameter mode to download a file.

Note: If no I/O Mapper exists in the control, switch the control to parameter mode by setting Register 001, bit 01 to 1. Select **On-Line Data** \Rightarrow **Registers** from VisualMotion's main menu to modify registers.

3.4 Menu Selection

Many of the menu selections used in the I/O Mapper have standard Windows™ functionality. This section will only describe those menu selections that are unique to the I/O Mapper.



Fig. 3-4: I/O Mapper Menus and Icon Toolbars

The File Menu

The **File** menu contains commands for:

- standard Windows functions, such as Open, Save, Print, etc.
- downloading and upload I/O files

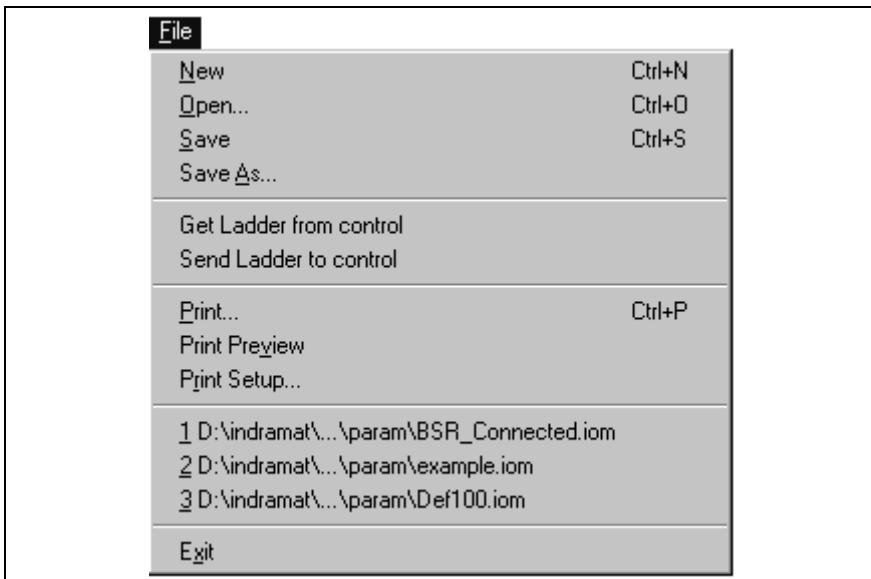


Fig. 3-5: The File Menu

Uploading and Downloading I/O Mapper Files

Selecting the following commands is equivalent to clicking on the upload () or download () icons.

Get Ladder from control

This menu selection copies the ladder program on the control to the Ladder Editor.

Send Ladder to control

This menu selection downloads the ladder program in the I/O Mapper to the control. If a change has been made to the ladder, the user will be prompted to check the rungs before the ladder is downloaded.

Printing an I/O Mapper File

Selecting the **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.

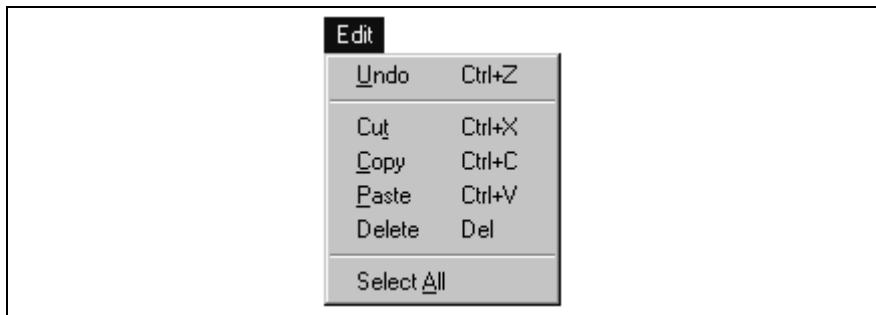


Fig. 3-6: The Edit Menu

The View Menu

The View menu is used to activate the toolbar and status bar within the I/O Mapper. If checked, the view is visible. To increase the size of the other windows, make these views invisible.

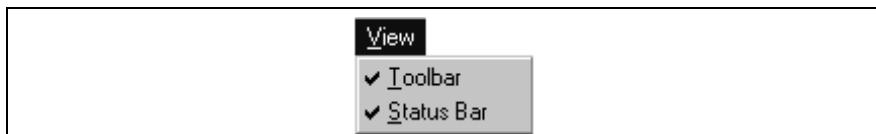


Fig. 3-7: The View Menu

The Setting Menu

The Settings 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.

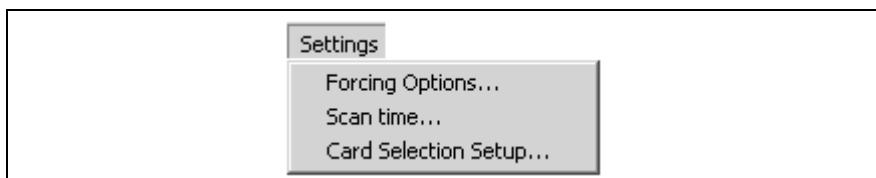


Fig. 3-8: The Settings Menu

Forcing Options

Selecting **Settings** ⇒ **Forcing Options** enables the user to set the 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.

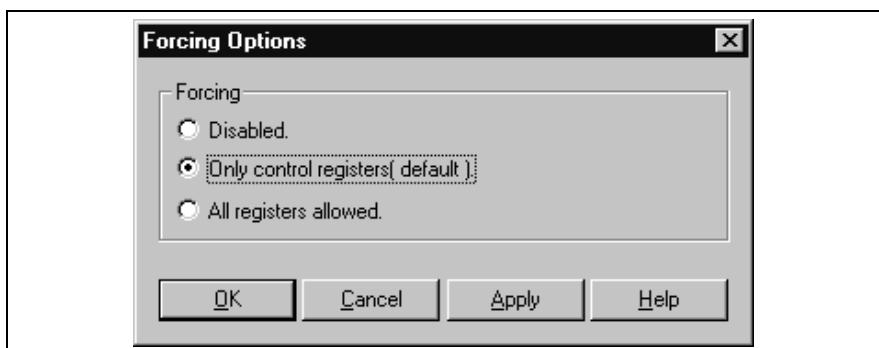


Fig. 3-9: Forcing Options

Scan Time

Selecting **Settings** ⇒ **Scan Time** enables the user to set the I/O Mapper scan time to either 2ms or 4 ms in parameter mode.

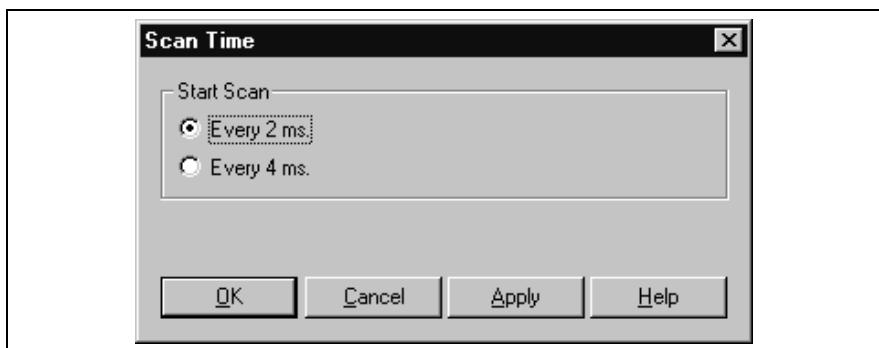


Fig. 3-10: Scan Time

Card Selection Setup

The *Card Selection Setup* window is used to setup communications with the control. The user selects the *Method* and *Card Number* of the control and clicks on OK or Apply to define the link between the PC and the control.

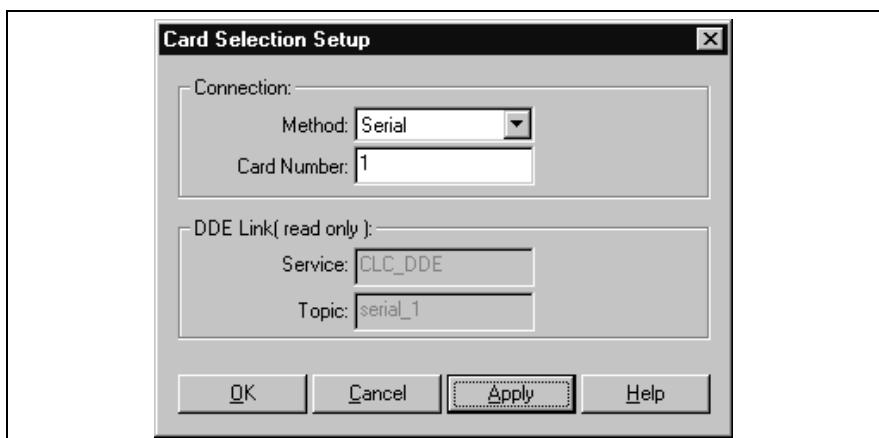


Fig. 3-11: DDE

The choices that are typically used within the I/O Mapper are as follows:

- Serial - RS-232 or RS-422 communication
- Network - Ethernet connect using TCP/ IP

The Window Menu

The Window menu is used to arrange the Ladder Editor's windows. The windows can be Cascaded or Tiled to the user's preference. The currently active window has a check mark to the left of the window name.

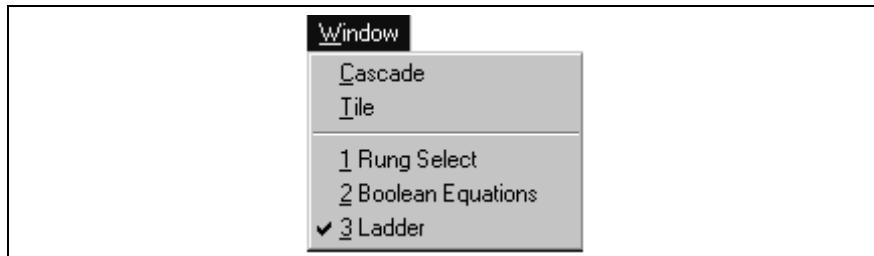


Fig. 3-12: The Window

The Help Menu

The I/O Mapper help is launched when *Help Topics* is selected.

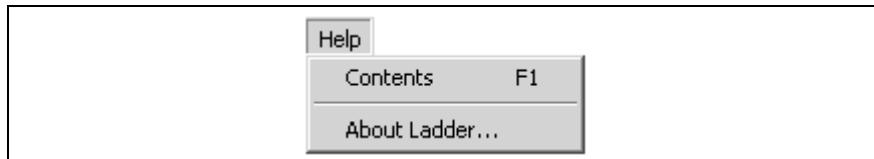


Fig. 3-13: The Help Menu

Selecting **Help** ⇒ **About Ladder** displays the following window containing information about the Ladder Editor.

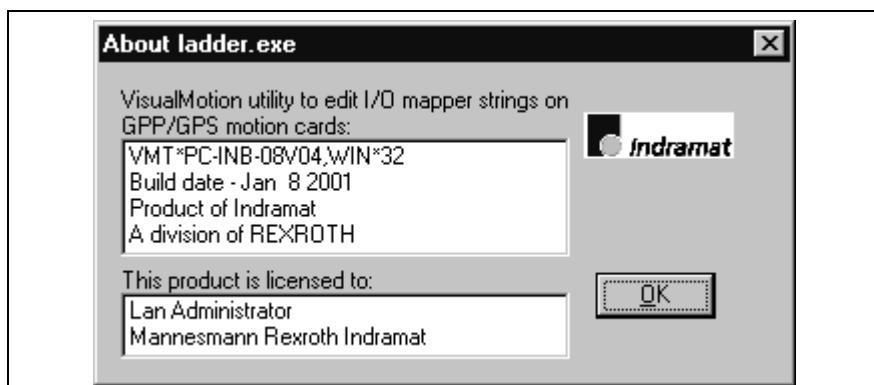


Fig. 3-14: About Ladder

3.5 Ladder Logic Toolbar

The Ladder logic icons are only accessible when the Ladder window is active. The Ladder logic icons are described in Fig. 3-15.

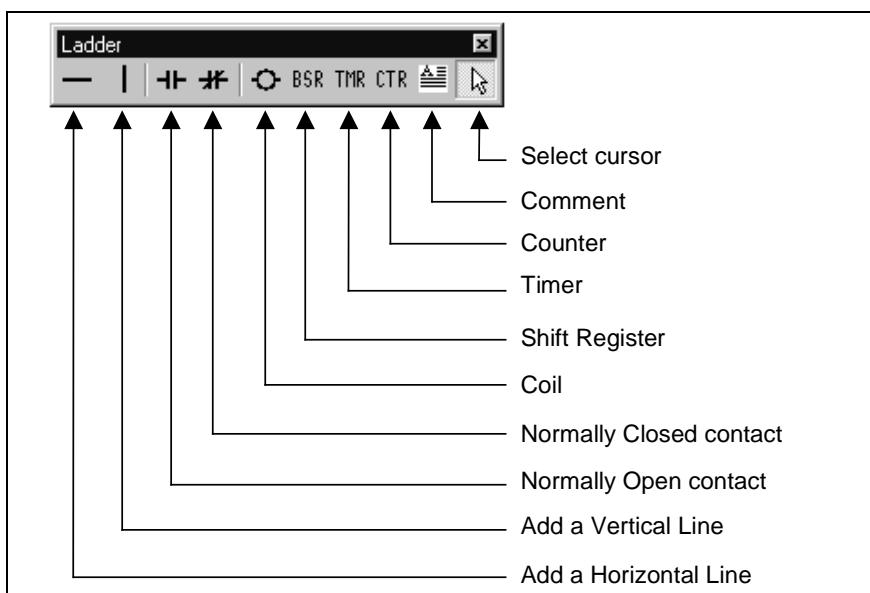


Fig. 3-15: Ladder Logic Icons

3.6 Input Logic Functions

The I/O Mapper provides a Normally Open contact (**¶**) and a Normally Closed (**¶**) contact as input logic. When either contact is placed in the Ladder window, a Properties window opens.

Properties

The Properties window is used for assigning an association 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.

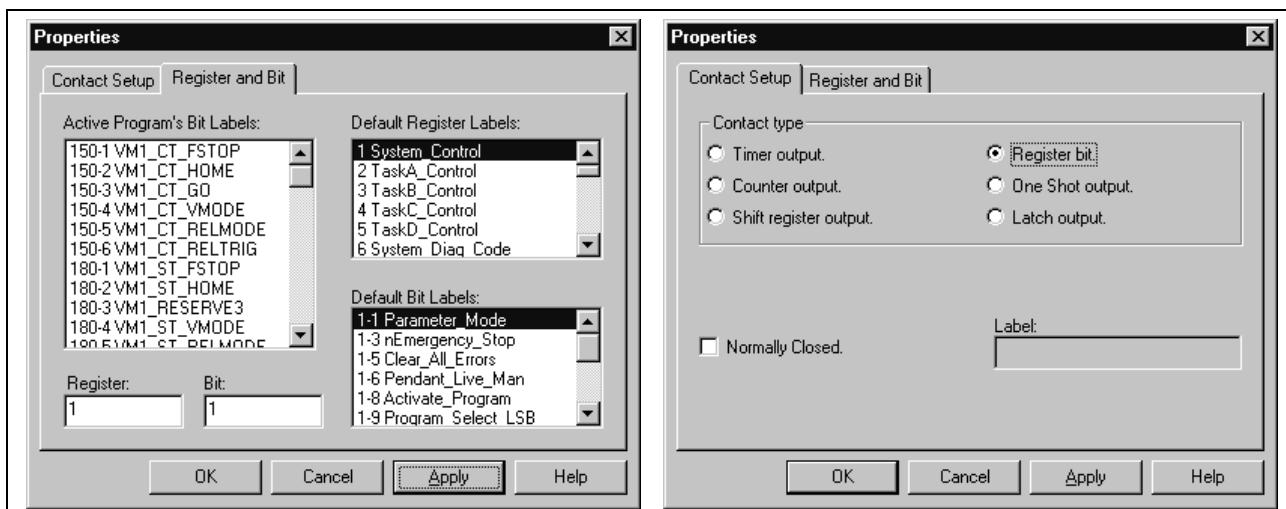


Fig. 3-16: Properties Window for Contacts

Register and Bit

The Register and Bit tab is used to select a register and bit. They can be entered directly, selected from a Default Register Bit Label (provided by VisualMotion) or selected from a label from 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 type:

 T	Timer output	 R	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 used as the output signal for the output functions provided in the I/O Mapper. Refer to Output Logic Functions for details.

Note: When a contact type other than Register Bit is selected, the *Register and Bit* tab changes to the *Contact Selection* tab.

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 3-2: 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

	Shift Register Bit within output function (01-32)
	Binary Shift Register output number (01-32)

Refer to Binary Shift Register (BSR)  for details.

3.7 Output Logic Functions

The I/O Mapper provides the following logic types for use as output logic:

- Coil Relay
- Binary Shift Register
- Timer
- Counter

Coil Relay

A coil is used to represent an output state that when TRUE, the corresponding assigned register bit is set, causing a response in the user program.

Four types of coils are available:

- Normal
- One Shot
- Latched
- Unlatched

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 and Counter.

Fig. 3-17 shows an example of a coil being used as a simple output function. Refer to Fig. 3-20, Fig. 3-21 and Fig. 3-22 for examples of a normal coil function complementing a more complex function.

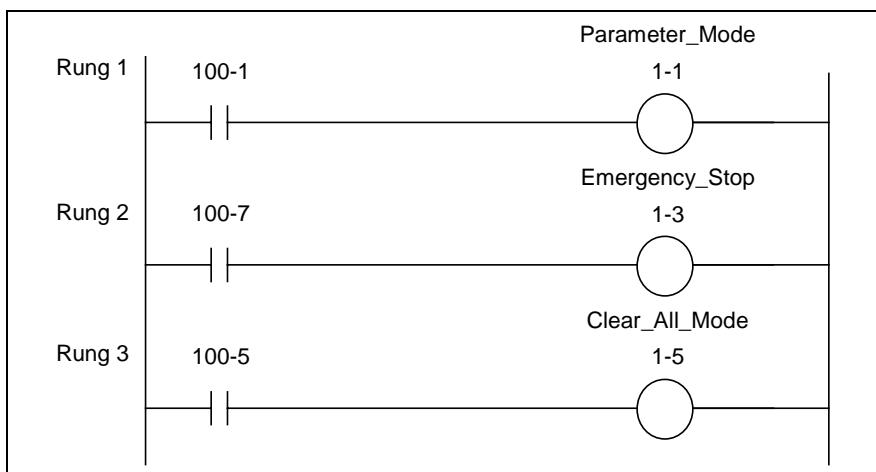


Fig. 3-17: Coil used as a Simple Output Function

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. The scan time can be set to 2 or 4 ms. A One Shot will reinitialize when the next raising edge is detected.

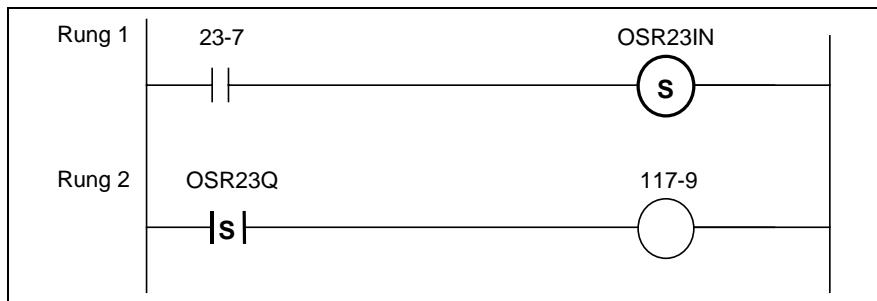


Fig. 3-18: One Shot Example

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
OSRx _x IN	a rising edge set Q TRUE for one scan
OSRx _x Q	output is set to 1 for one scan when rung is TRUE

Table 3-3: One Shot Functions

The One Shot input and output functions use the predefined Global Integers (GIx) in Table 3-4: One Shot Global Integers (GIx) for storing their values. All functions are initialized to 0 on power-up.

OSR ID	Global Integer	Description
OSRx _x where, xx represents counters 01-32	GI425	One Shots Q state, 32 bits
	GI426	One Shots IN , 32 bits
	GI427-GI428	Reserved

Table 3-4: 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. A maximum of 32 Latch functions is available.

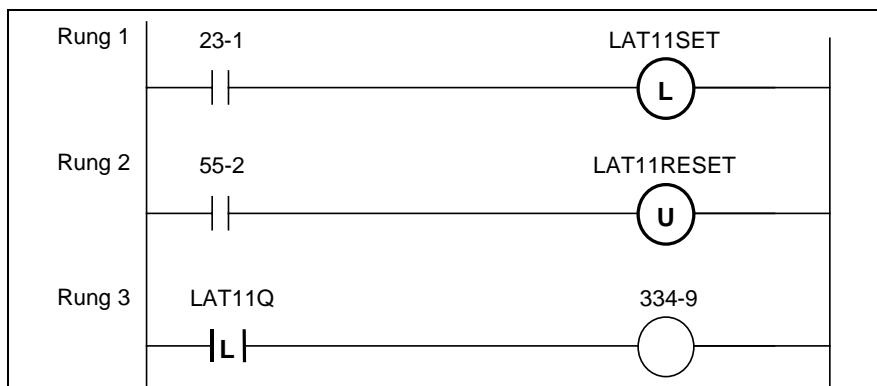


Fig. 3-19: Latched and Unlatched Example

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
LATxxSET	a 1 sets the latch Q output to 1
LATxxRESET	a 1 resets the latch Q output to 0
LATxxQ	Latch output

Table 3-5: Latch and Unlatch Functions

The Latch and Unlatch input and output functions use the predefined Global Integers (GIx) in Table 3-6: Latch and Unlatch Global Integers (GIx) for storing their values. All functions are initialized to 0 on power-up.

LAT ID	Global Integer	Description
LATxx where, xx represents counters 01-32	GI465	Latches Q state, 32 bits
	GI466	Latches history state, 32 bits
	GI467-GI468	Reserved

Table 3-6: Latch and Unlatch Global Integers (GIx)

Binary Shift Register (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. A maximum of 32 Binary Shift Registers is available.

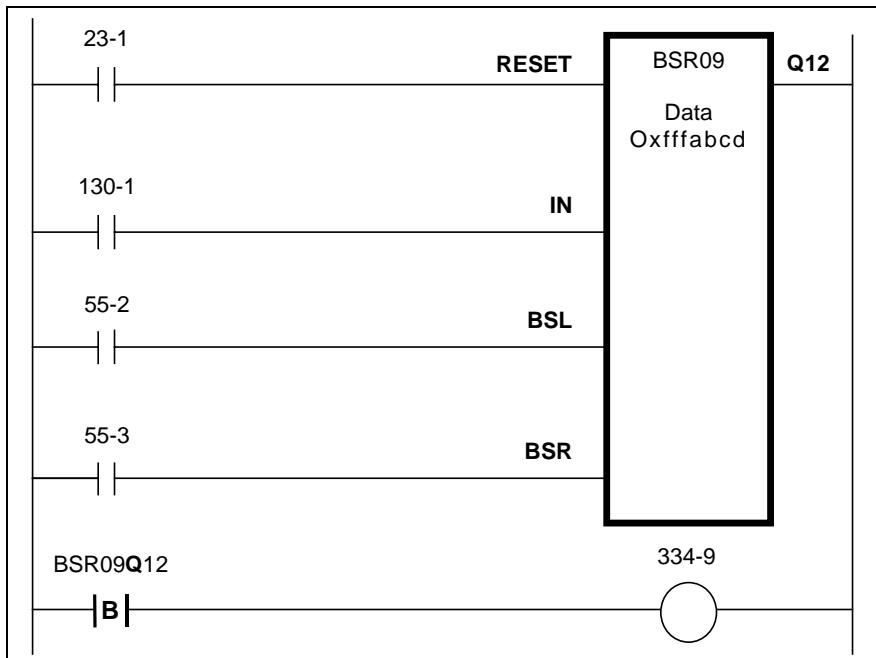


Fig. 3-20: Binary Shift Register Example

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
BSRx _x RESET	a 0 to 1 transition sets all shift register bits to 0
BSRx _x IN	input data for bit 1 when BSL is set, bit 32 when BSR is set
BSRx _x BSL	0->1 transition moves shift register data 1 bit to left, sets BSRx_x/IN value into bit 1
BSRx _x BSR	0->1 transition moves shift register data 1 bit to right, sets BSRx_x/IN value into bit 32
BSRx _x Qyy	the number "yy" in the Q output contact refers to the bit number within the shift register (1 to 32).

Table 3-7: Binary Shift Register Functions

The Binary Shift Register's input and output functions use the predefined Global Integers (Glx) in Table 3-8: Binary Shift Register Global Integers (Glx) for storing their values. All functions are initialized to 0 on power-up.

BSR ID	Global Integer	Description
BSRx _{xx} 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 3-8: Binary Shift Register Global Integers (GIx)

Timer (TON)

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. The scan time can be set to 2 or 4 ms. 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. A maximum of 32 Timers is available.

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.

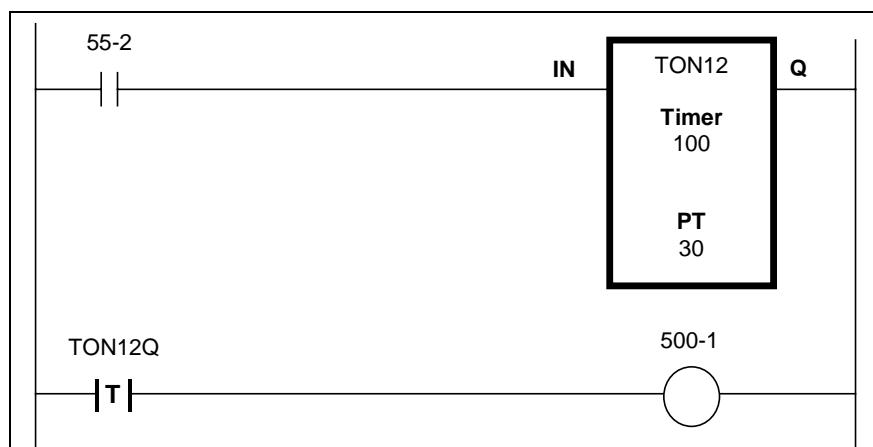


Fig. 3-21: Timer Example

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 3-9: Timer Functions

The Timer's input and output functions use the predefined Global Integers (GIx) in Table 3-12 for storing their values. All functions are initialized to 0 on power-up.

Timer ID	Global Integer	Description
TON xx where, xx represents timers 01-32	GI357	Timers Q state, 32 bits
	GI358	Timers IN history state, 32 bits
	GI359-GI360	Reserved
	GI361-GI392	Timers TIME, integer
	GI393-GI424	Timers PT, integer

Table 3-10: Timer Global Integers (GIx)

Counter (UDC)

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. A maximum of 32 counters is available.

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.

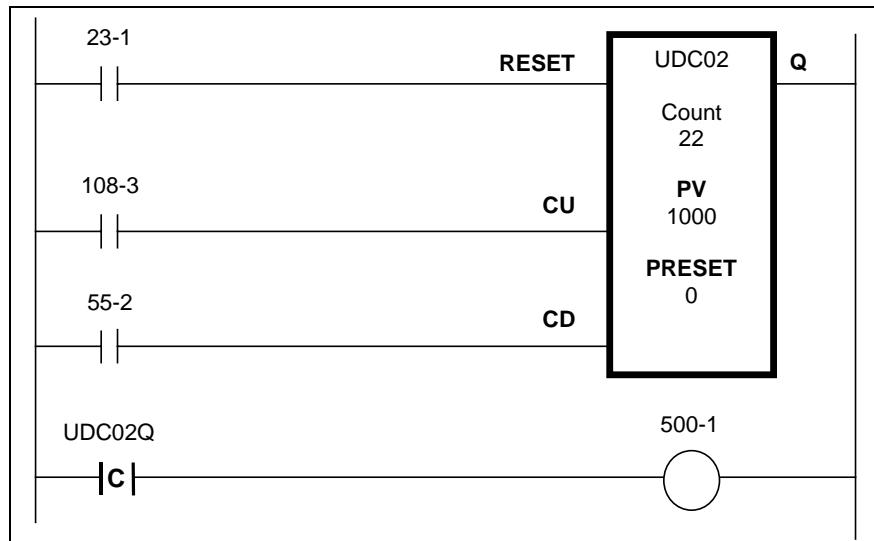


Fig. 3-22: Counter Example

In the counter example in Fig. 3-22, 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
UDCx x RESET	a 0 to 1 transition sets the <i>Count</i> value to the <i>PRESET</i> value
UDCx x CU	a 0 to 1 transition increments the <i>Count</i> by one
UDCx x CD	a 0 to 1 transition decrements the <i>Count</i> by one
UDCx x PV	the target value used to set the counter's corresponding contact (UDCx x Q)

Signal Name	Description
UDCx _x PRESET	the initial value that is used by the counter when activated or after a <i>RESET</i>
UDCx _x Q	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 3-11: Counter Functions

The counter's inputs and output functions use the predefined Global Integers (GIx) in Table 3-12 for storing their values. All functions are initialized to 0 on power-up.

Counter ID	Global Integer	Description
UDCx _x where, xx represents counters 01-32	GI257	Counters Q state, 32 bits
	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 3-12: Counter Global Integers (GIx)

3.8 Forcing Icon Toolbar

The forcing icons are only accessible when the Ladder window is active. The Ladder logic icons consist of the icons in Fig. 3-23.

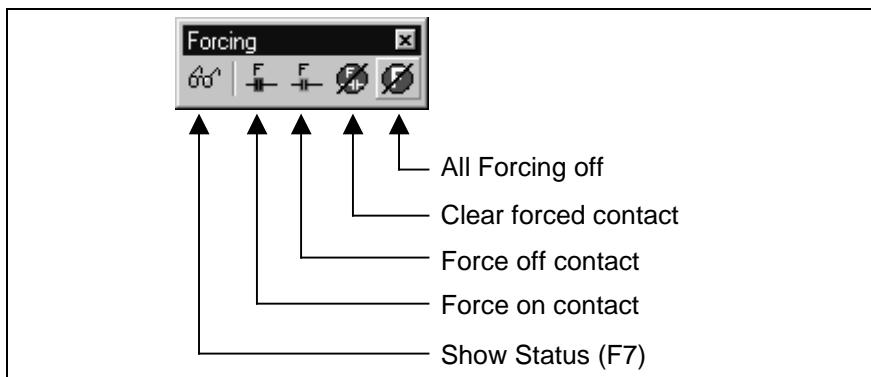


Fig. 3-23: Forcing Icons

Forcing Options

Before the forcing icons are functional, the forcing options must be set by selecting **Settings** \Rightarrow **Forcing Options**. The control must be in Parameter mode before changing the forcing option.



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 cannot be forced.



Fig. 3-24: Forcing Options

Disabled

When this option is selected, the state of a contact or coil cannot be forced using the forcing icons.

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 using the forcing icons.

All Registers Allowed

When this option is selected, the state of all contacts and coils can be forced using the forcing icons.

Using the Forcing Icons

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 option under **Settings** ⇒ **Forcing**.

Note: Contacts that are used as a function's output (Q) cannot be forced. They are dependent solely on the functions output.

Forcing Example

To force the state of a contact or a coil, follow the steps below:

1. Select the desired contact or coil using the left mouse button. A green cross hatch area will cover the object.

Note: The Select cursor () icon must be used when performing this function.

2. From the Forcing icon palette, select one of the following choices:



Force on



Force off



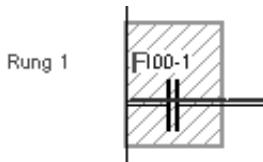
Clear force



Forcing off

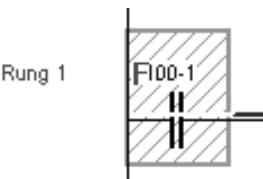
Note: In order to view contact state changes and logic flow, the Status (F7)  icon must be depressed.

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:



The red **F** next to the ID indicates that the contact has been forced.
The contact's state is TRUE and a second line above the rung line shows the flow of the logic.

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:



The red **F** next to the ID indicates that the contact's state is still forced.
However, the contact's state is FALSE and the logic flow line above the contact disappears.

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 entry Ladder, select ( Forcing off).

VisualMotion Messages for Forcing

The following messages appear when the forcing icons are used incorrectly:

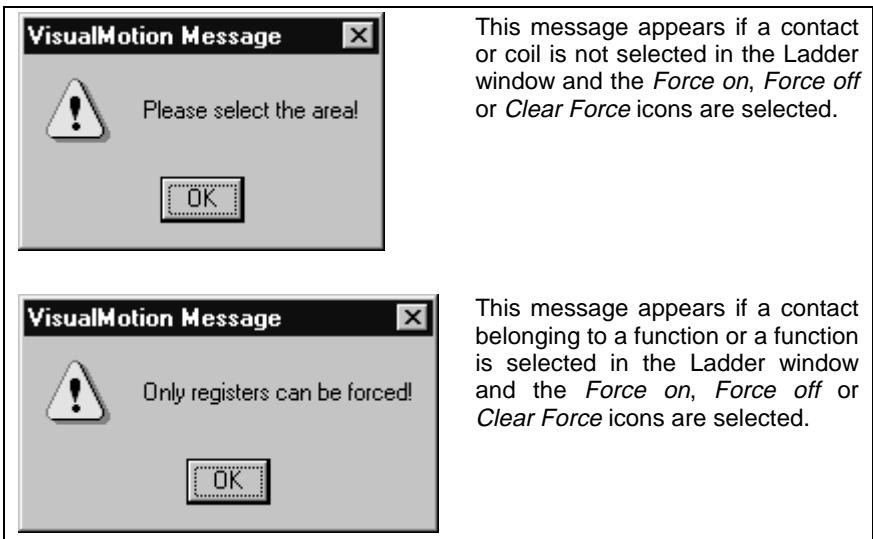
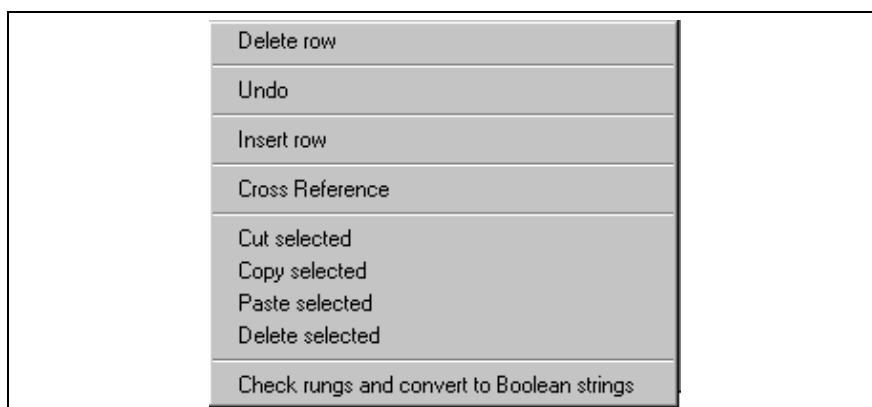


Fig. 3-25: VisualMotion Messages for Forcing

3.9 Additional Tools

The following additional tools are available in the Ladder view area by right-clicking anywhere in the view area.



Delete row

To delete a row (rung), click the select cursor over the desired row, right-click and select *Delete row*. VisualMotion warns before deleting the row. The information in all the rows below the one deleted, moves 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.

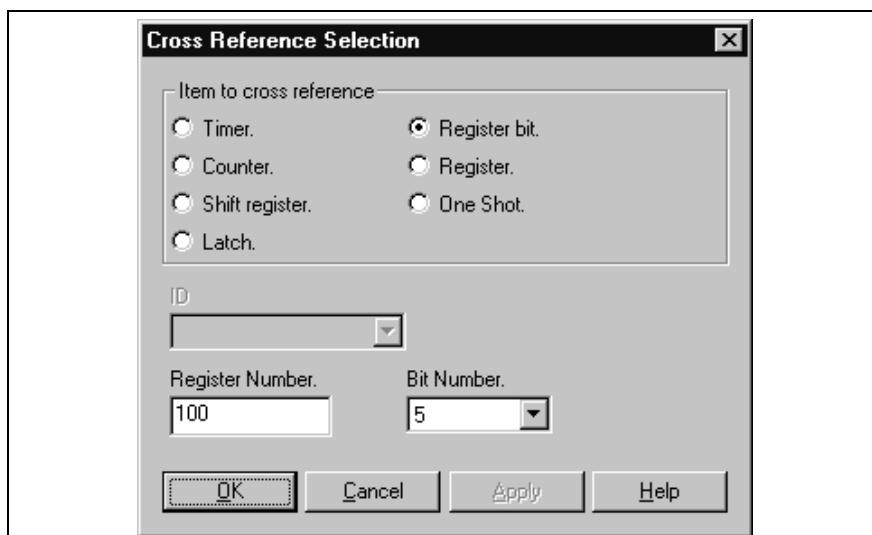


Fig. 3-26: Cross Reference Selection

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.

Rung	Use
6	Normally open contact
9	Normally open contact
12	Normally open contact
15	Normally open contact

Fig. 3-27: Cross Reference Results

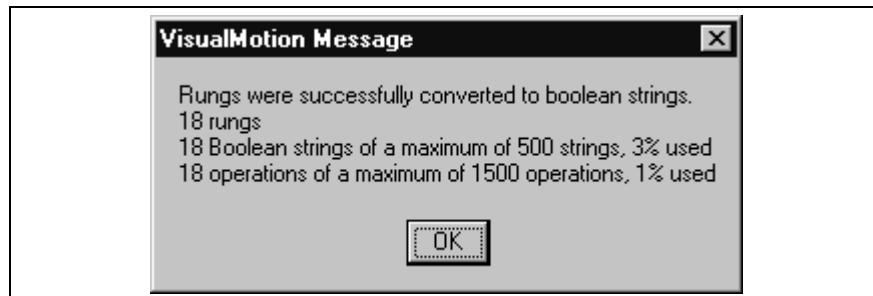
Double-clicking on any location will display the rung at the top of the ladder viewing area.

Cut, Copy, Paste and Delete

To perform these basic Window's functions (with the arrow cursor selected), left-click over an area then right-click and select the desired function. Click and hold the mouse button to select a large area.

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.



4 Programmable Limit Switch Functionality

4.1 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.

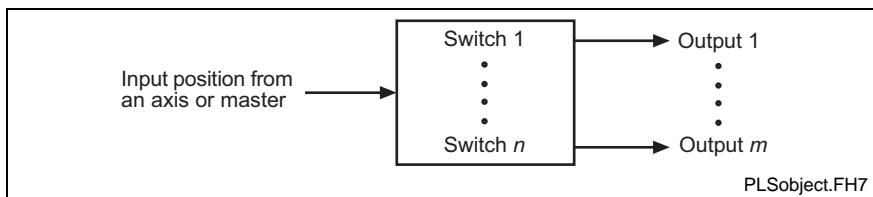


Fig. 4-1: Basic PLS Object

VisualMotion 8 supports three PLS types (Control, Drive and Option Card PLS). The number of supported PLS objects by type are:

- 1 for Drive PLS
- 2 for Control 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 8.

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 4-1: PLS Types

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.

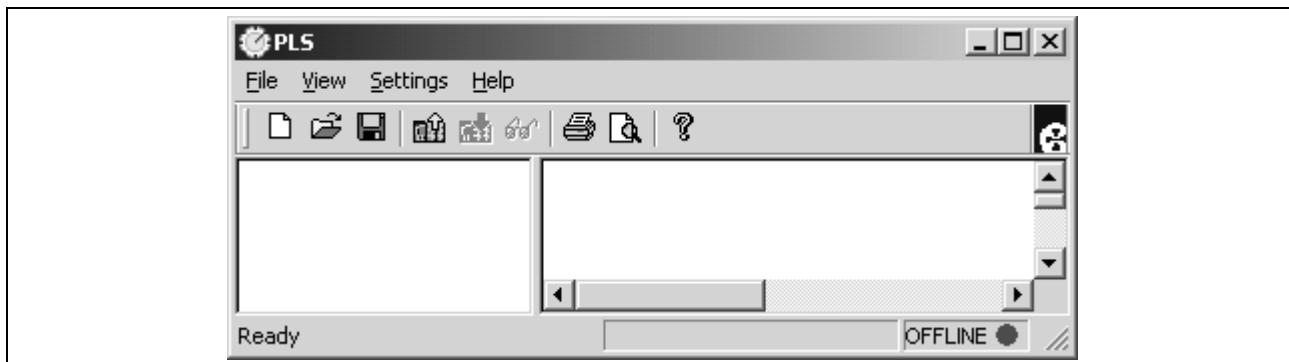


Fig. 4-2: PLS Configuration Tool

Control PLS

A *Control PLS* data structure exists in every VisualMotion user program. The settings can be made using a Control PLS icon or the PLS tool then downloaded to the control. Before a Control PLS can be uploaded to the PLS tool, there must be an activate program in the control. Otherwise, only Drive and Option Card PLSs, if configured, will be displayed. A VisualMotion user program can have a maximum of 2 Control PLS objects.

Note: A Control PLS **only** resides with the compiled user program and cannot be saved separately.

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.

A Control PLS has the following specifications:

Control PLS 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 (1-32)

ASCII Communication for Accessing Control PLS Data

Unlike Drive and Option Card PLSs that use system parameters for data storage, a Control PLS is stored as part of the compiled VisualMotion user program. Standard ASCII communication protocol can be used to access Control PLS data. The following table lists each instruction with a brief description. Refer to [Direct ASCII Communication](#) in Chapter 12 for details.

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 4-2: ASCII Protocol Instructions for a Control PLS

Drive PLS

The parameters of a Drive PLSs can be configured offline using the PLS tool and downloaded to the control. With a parameter editor, the associated Drive PLS drive parameters can also be parameterized during runtime.

To access a Drive PLS, open the PLS Configuration tool and click on the *Get from GPP Control* icon (). If configured, all Control, Drive and Option Card PLSs in the VisualMotion system will be uploaded into the PLS tool. Make the necessary modifications to the Drive PLS and download it to the control. Refer to **Configure a Drive PLS** on page 4-11 for details.

Note: Refer to the respective Digital Drive Functional Description manual for details about the drive PLS feature.

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 4-16 for details.

ECODRIVE03 Drive PLS Specifications

Note: ECODRIVE03 supported firmware versions are as follows:

- FWA-ECODR3-SGP-01VRS-MS
- 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 4-3: Drive PLS Parameters

Option Card PLS

The *Option Card PLS* is an option card for the PPC-R 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 preconfigured with an Option Card PLS and either 16 or 32 digital outputs for programmable limit switches.

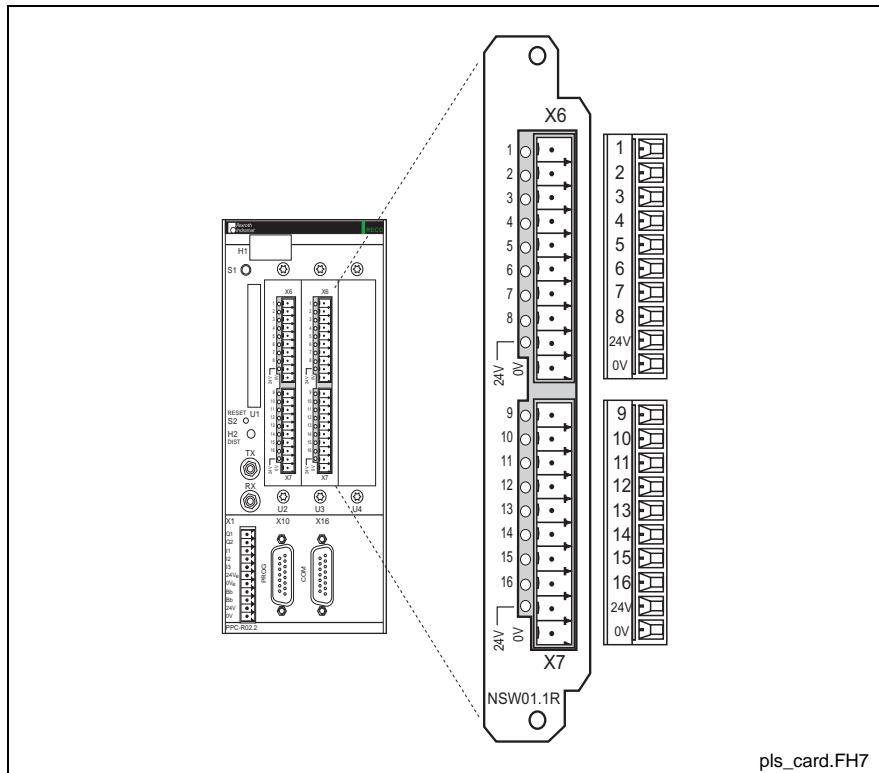


Fig. 4-3: NSW01.1R Option Card PLS

Option Card PLS Specifications

- Up to 32 digital outputs with a high-speed update rate of 250 µs
- 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 type from the following:
 - ELS Masters (1-6 System Masters),
 - ELS Groups (1-8) or
 - Drives (1-32)

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. Refer to **System (Control) Parameters - Class C** in Chapter 6 for details.

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)
C-0-2908	PLS1 Extended Error Code	Phase 2 (read)
C-0-2909	PLS1 Hardware ID	Phase 2 (read)
C-0-2910	PLS1 Software ID	Phase 2 (read)

Table 4-4: 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 4-5: 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 4-6: 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 4-7: PLS Master Parameters for an Option Card PLS

4.2 Configure a Control PLS

Open the PLS Configuration tool by selecting ***Commission*** ⇒ ***PLS...*** from VisualMotion's main menu. Select ***File*** ⇒ ***New*** or click on the New icon () to launch the *New PLS Configuration* wizard. Select **Control** and **Control #.**.

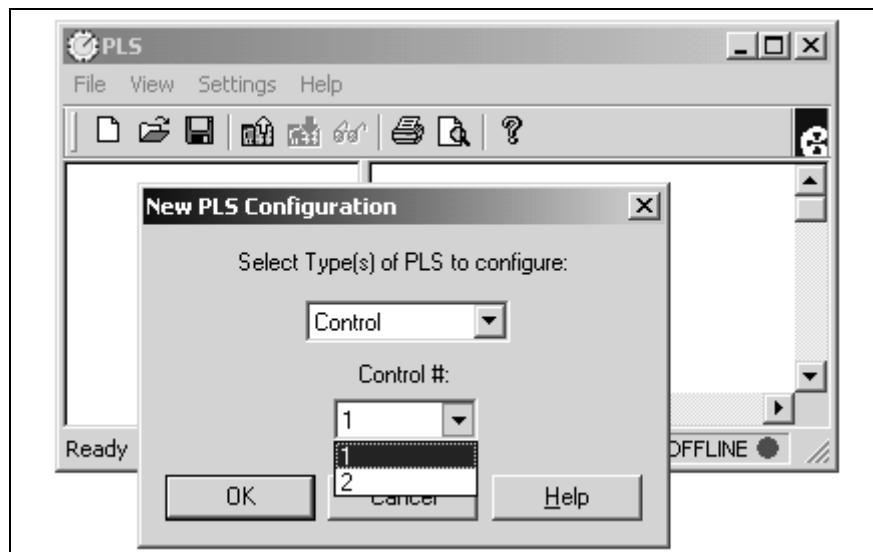


Fig. 4-4: New PLS Configuration

From the *PLS Switches* window in Fig. 4-5. 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.

Switch#	On Pos	Off Pos	Lead Time
1	0.000000	0.000000	0
2	0.000000	0.000000	0
3	0.000000	0.000000	0
4	0.000000	0.000000	0
5	0.000000	0.000000	0
6	0.000000	0.000000	0
7	0.000000	0.000000	0
8	0.000000	0.000000	0
9	0.000000	0.000000	0
...

Fig. 4-5: PLS Switches Window

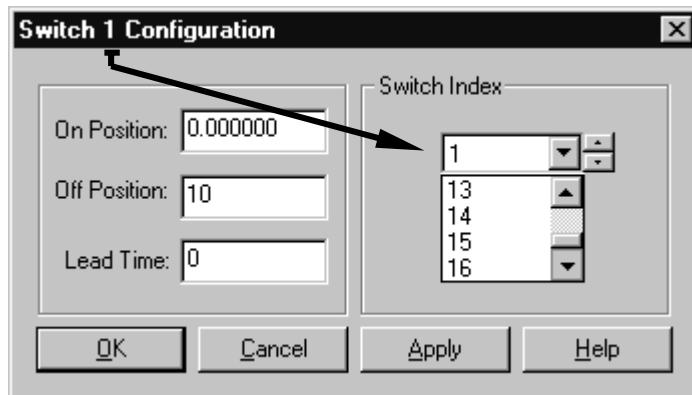
Switch Configuration for a Control PLS

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.

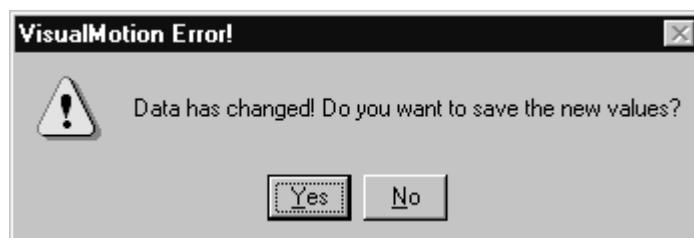
Example: If these parameters are set to 1 (mm), then the unit of measurement for the switch's On and Off positions is mm.

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.



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)

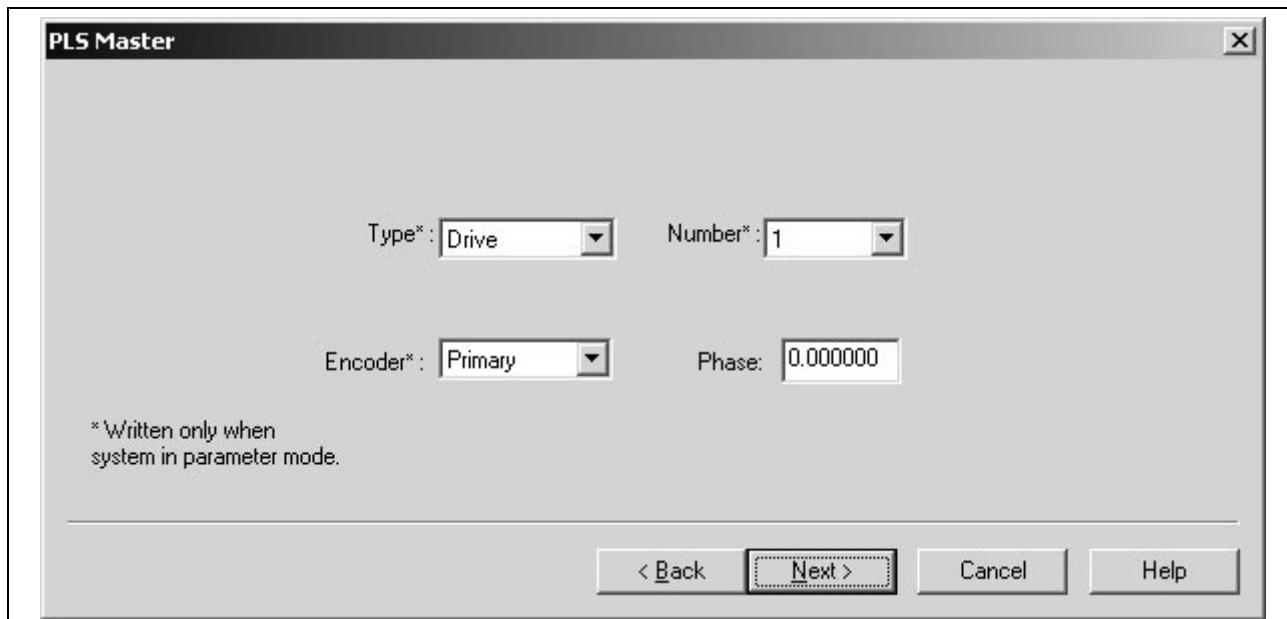


Fig. 4-6: PLS Master Configuration for a Control PLS

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	1-32 (SERCOS address)	Primary or Secondary

Note: When a Drive is selected, the allowable number range is 1-32 for up to 32 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 input *Type*.

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 8, 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. 4-6, 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. 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.

1. Accept the default register or set a different *Output Register*.
2. 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 **On-Line Data** \Rightarrow **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.

3. Set the *Graphs Limit's* minimum and maximum value.
4. Press the **Finish** button to end the configuration wizard.

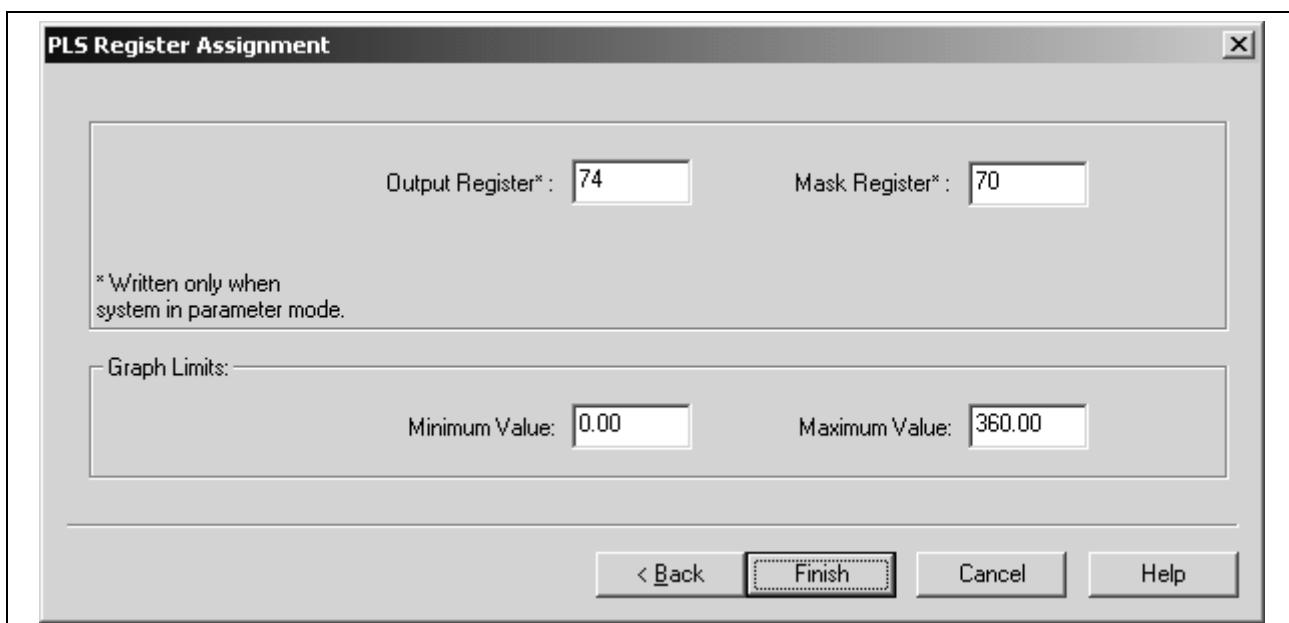


Fig. 4-7: PLS Register Assignment for a Control PLS

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.

Graph Limits

Set the minimum and maximum values for the graphical representation of all Control PLS switches.

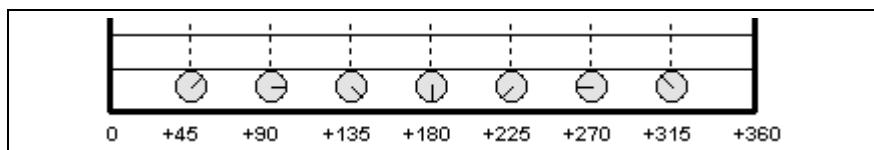


Fig. 4-8: Graph Limits

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

4.3 Configure a Drive PLS

Open the PLS Configuration tool by selecting **Commission** \Rightarrow **PLS...** from VisualMotion's main menu. Select **File** \Rightarrow **New** or click on the New icon () to launch the *New PLS Configuration* wizard. Select **DIAX Drive** or **ECO Drive** and enter the drive address for **Drive #** from 1-32.

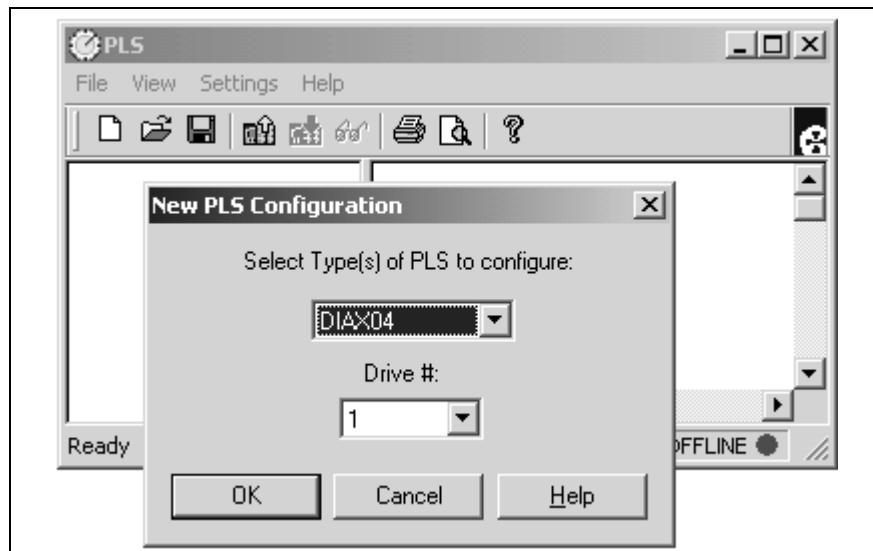


Fig. 4-9: New PLS Configuration

From the *PLS Switches* window in Fig. 4-10. 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.

Switch#	On Pos	Off Pos	Lead Time
1	0.000000	0.000000	0
2	0.000000	0.000000	0
3	0.000000	0.000000	0
4	0.000000	0.000000	0
5	0.000000	0.000000	0
6	0.000000	0.000000	0
7	0.000000	0.000000	0
8	0.000000	0.000000	0

Fig. 4-10: PLS Switches Window

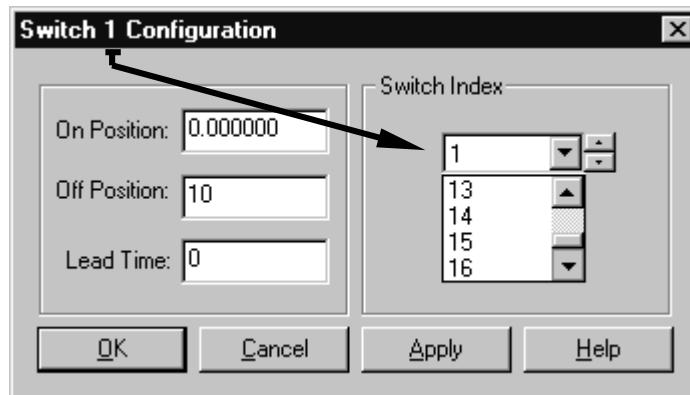
Switch Configuration for a Drive PLS

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.

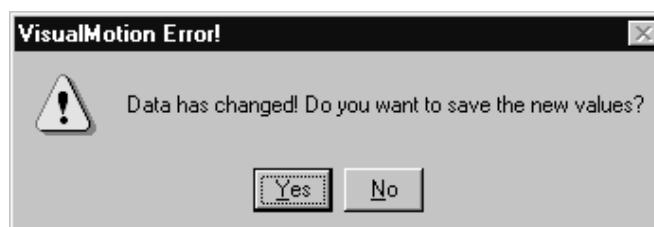
Example: If these parameters are set to 1 (mm), then the unit of measurement for the switch's On and Off positions is mm.

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.



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:

- Motor Encoder
- External Encoder

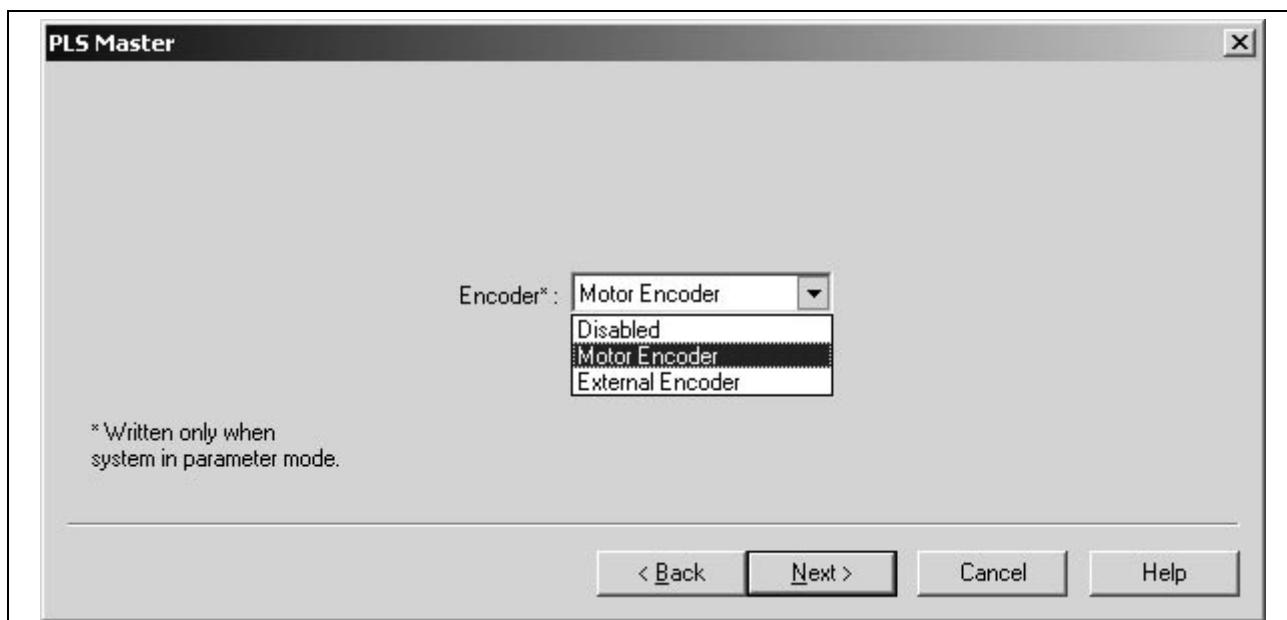


Fig. 4-11: PLS Master Configuration for a Control PLS

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.

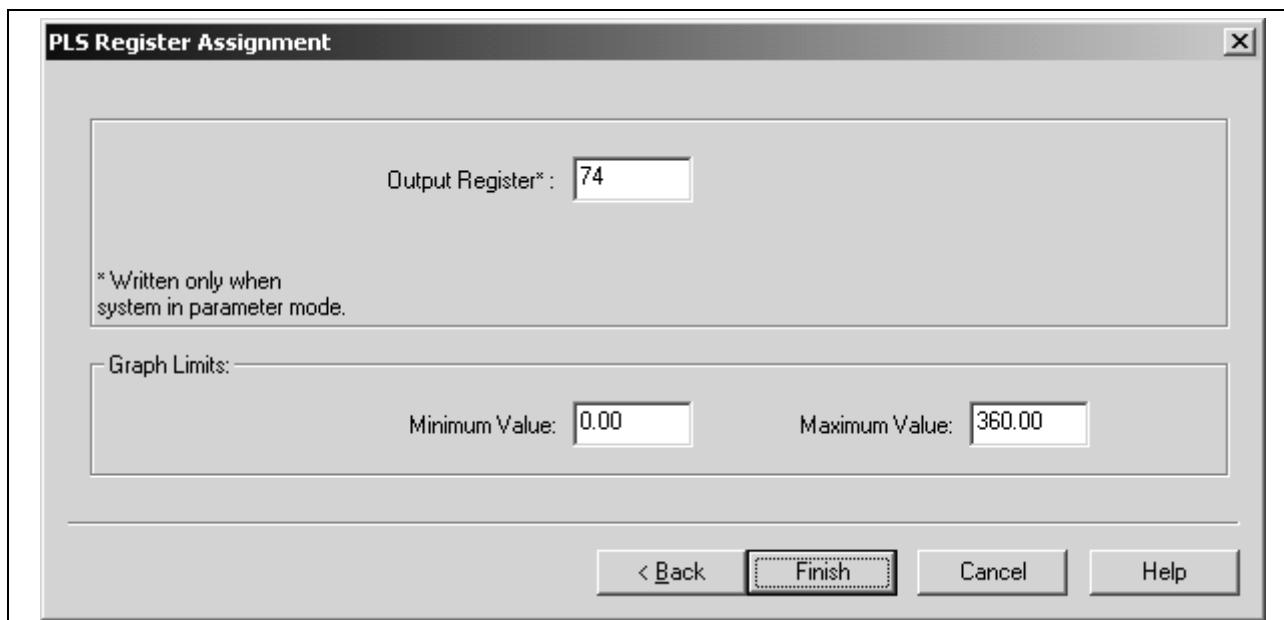


Fig. 4-12: PLS Register Assignment for a Control PLS

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. 4-13: 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 4-14.
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. Refer to on page for details.
4. Download the I/O configuration to the control in parameter mode.

During phase initialization (P2 to P3), VisualMotion reads and compares the register numbers assigned to the DIAX04 Drive PLS and the DEA04.2M, 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 DEA04.2M, 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.

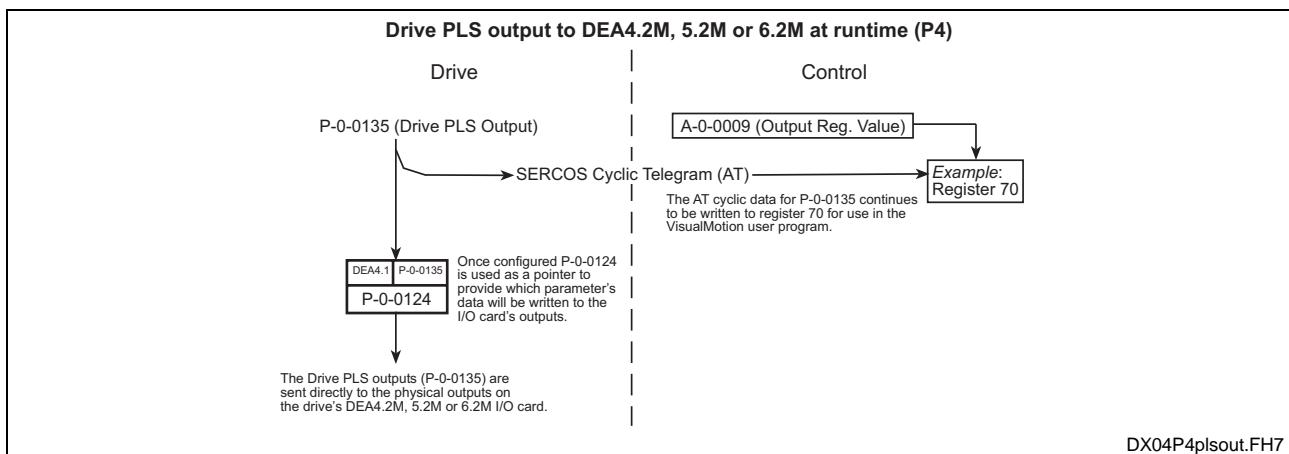


Fig. 4-14: DEA4 I/O Configured for Drive PLS Output

4.4 Configure an Option Card PLS

Open the PLS Configuration tool by selecting **Commission** ⇒ **PLS...** from VisualMotion's main menu. Select **File** ⇒ **New** or click on the New icon (□) to launch the *New PLS Configuration* wizard, then select **Card**.

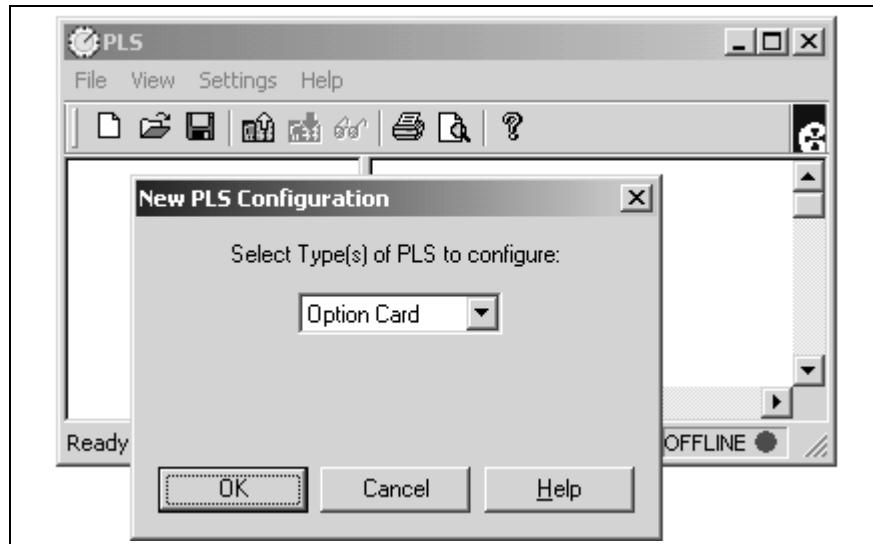


Fig. 4-15: New PLS Configuration

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. 4-16. 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.

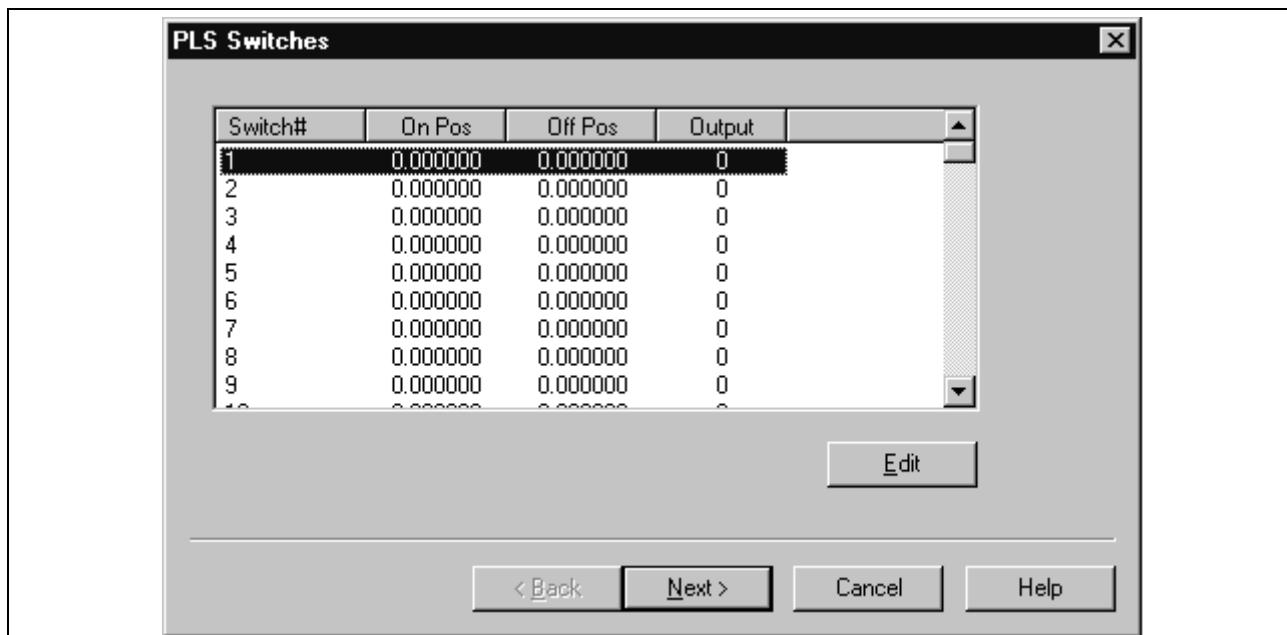


Fig. 4-16: PLS Switches Window

Switch Configuration for an Option Card PLS

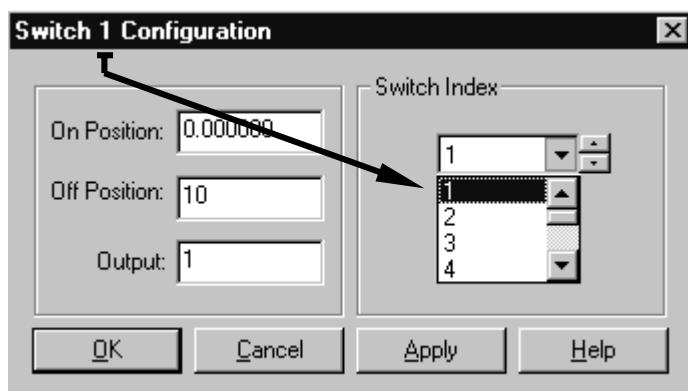
The following steps configure a switch's On / Off position and assigns it to an output.

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.

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.

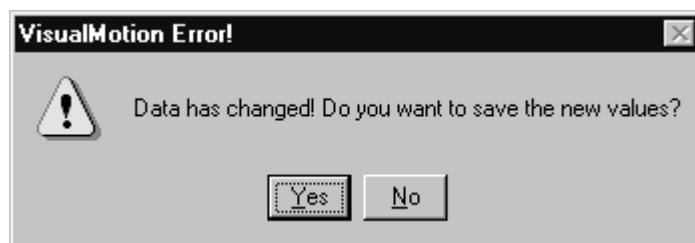
3. Enter an Output number for the switch.

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.

4. When all the switches are configured, press the **OK** button to close the *Switch # Configuration* window and return to the *PLS Switches* window.
5. 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.



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)

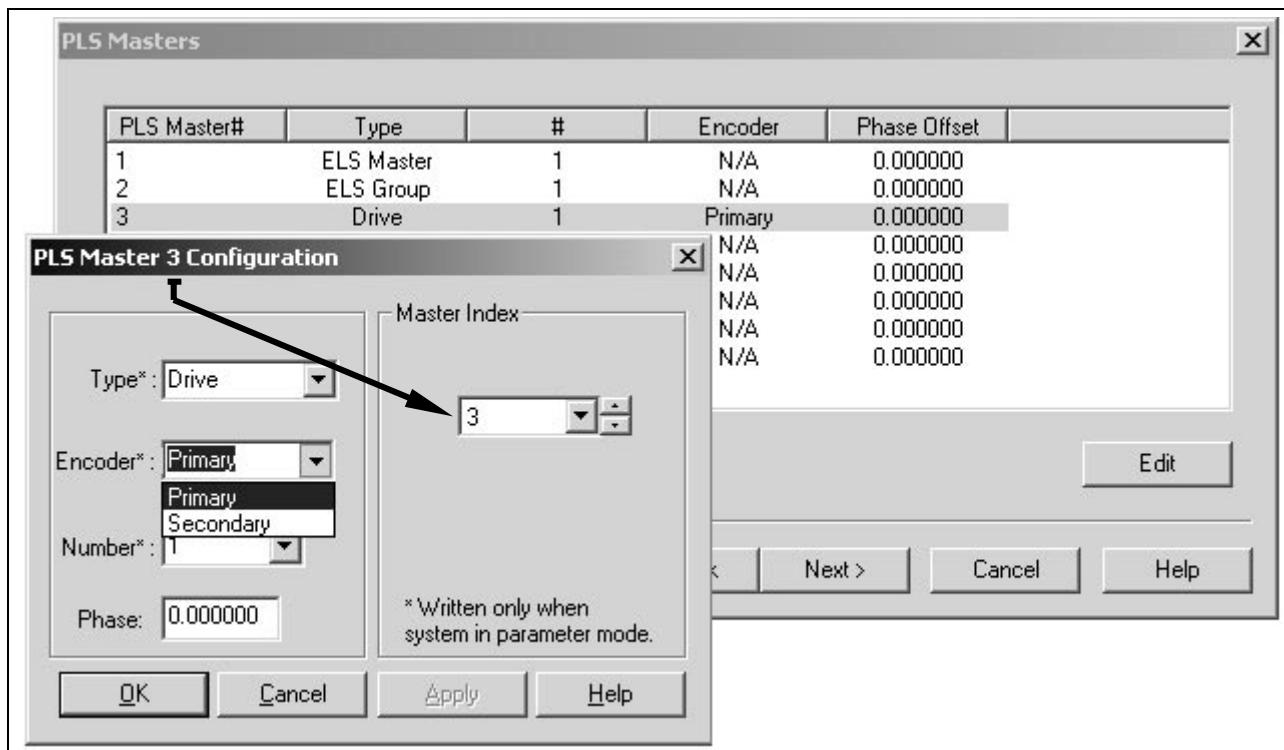


Fig. 4-17: PLS Master Configuration

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

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	1-32 (SERCOS address)	Primary or Secondary

Note: When a Drive is selected, the allowable number range is 1-32 for up to 32 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 input *Type*.

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 8, 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.

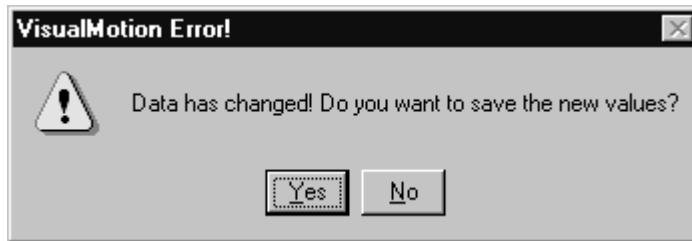
ELS Group

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. 4-17, 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.

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.



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 PLS cards 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 PLS card 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 *On-Line Data* ⇒ *Registers* from VisualMotion main menu. Only the Output Register whose Mask Register is set to (1) will output a signal on the PLS card.

3. Set the *Graphs Limit's* minimum and maximum value.
4. Press the **Next >** button to continue to the PLS Outputs window.

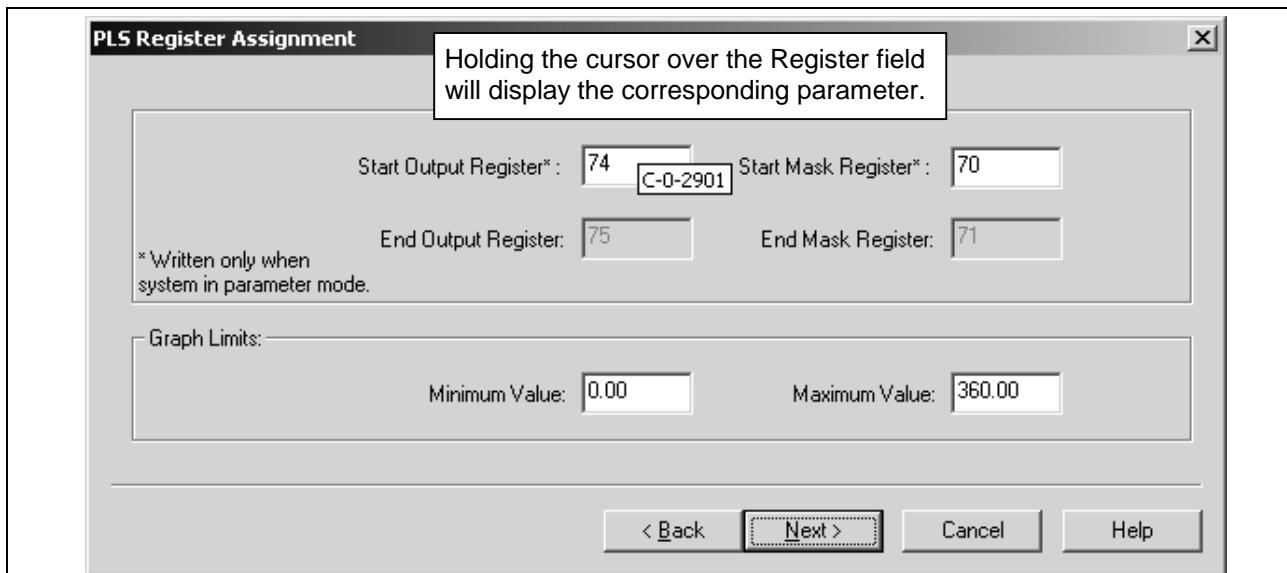


Fig. 4-18: PLS Register Assignment

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.

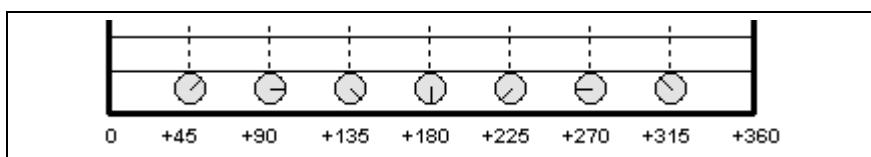


Fig. 4-19: Graph Limits

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. 4-20, 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.

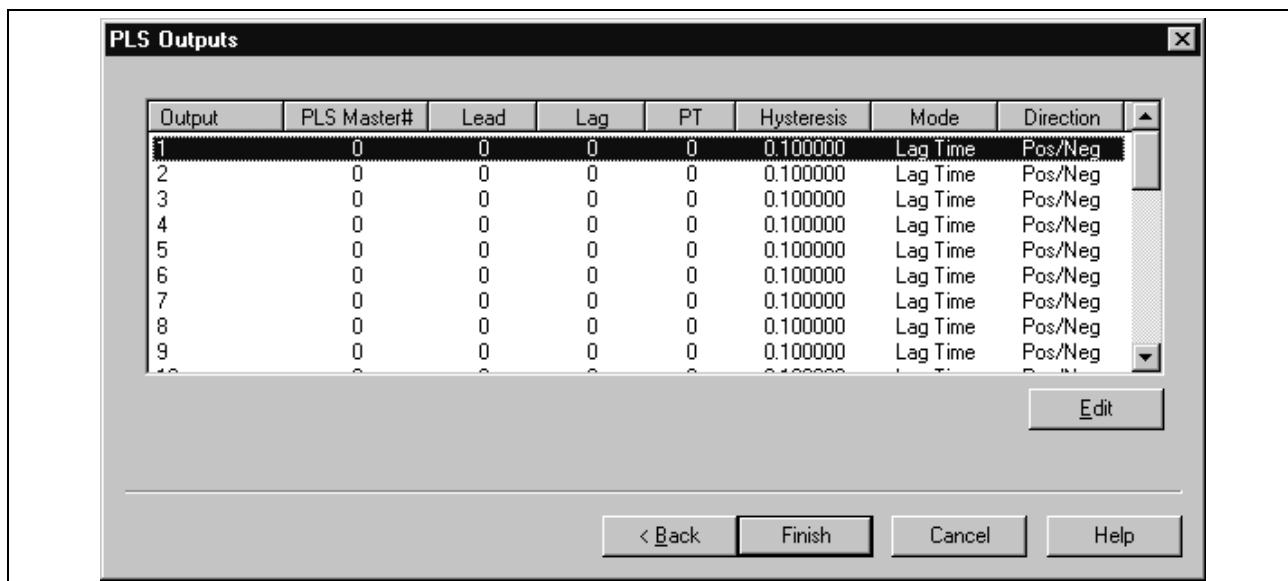


Fig. 4-20: Output Assignment

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.

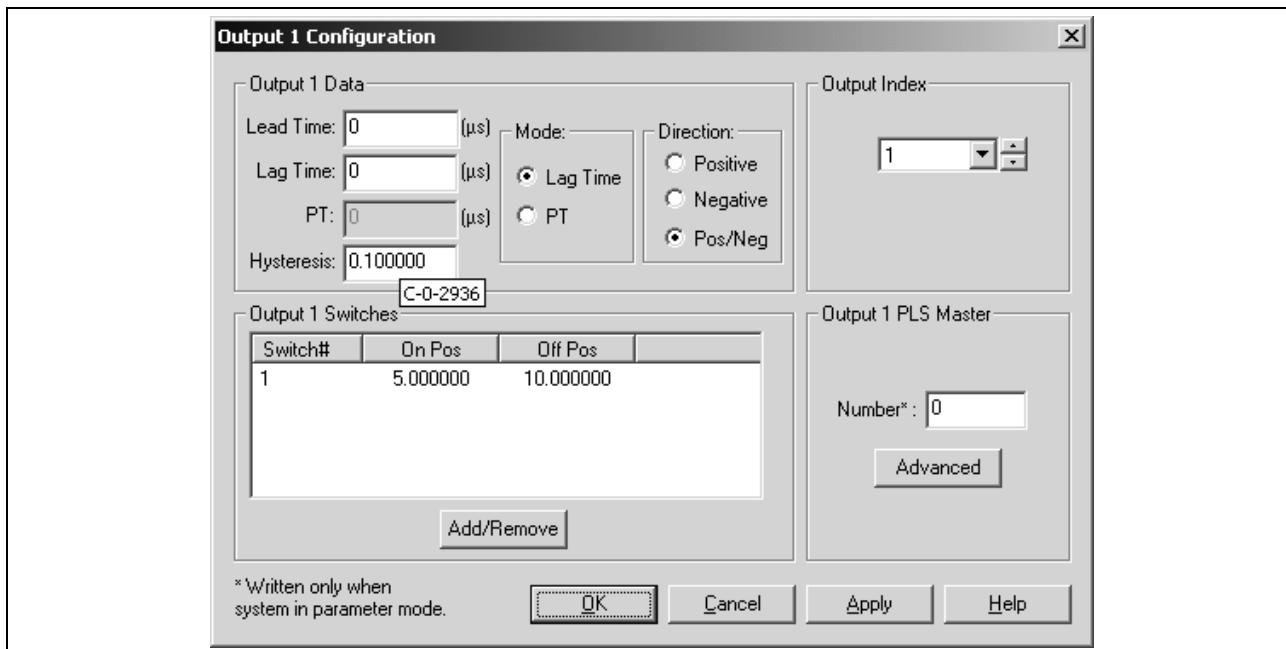


Fig. 4-21: Output Configuration Window

Holding the cursor over the following fields will display the corresponding parameter to which the configured value is stored.

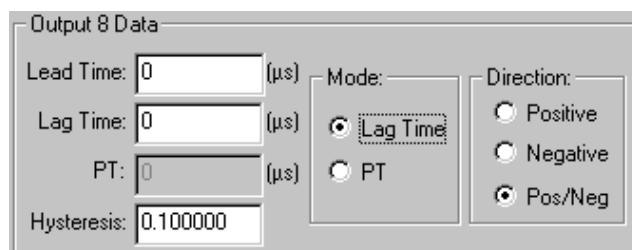
Field Name	Parameter	Condition
Lead Time	C-0-2931	Always active
Lag Time	C-0-2932	Active when Mode = Lag Time
PT (Time Duration)	C-0-2933	Active when Mode = PT Mode
Hysteresis	C-0-2936	Always active
Mode	C-0-2934	Always active
Direction	C-0-2935	Always active
Number (Output # PLS Master)	C-0-2941	Always active

Table 4-8: 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.



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 250 μ s, up to a maximum of 500,000 μ s.

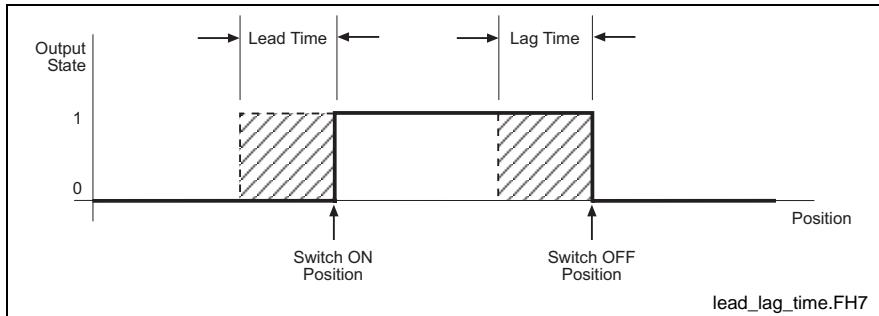


Fig. 4-22: Lead and Lag Time

- **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 250 μ s, up to a maximum of 1,000,000 μ s.

Note: When the Output Mode is set to PT and a value of \emptyset 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 is resettable 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 250 μ s 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. 4-23 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.

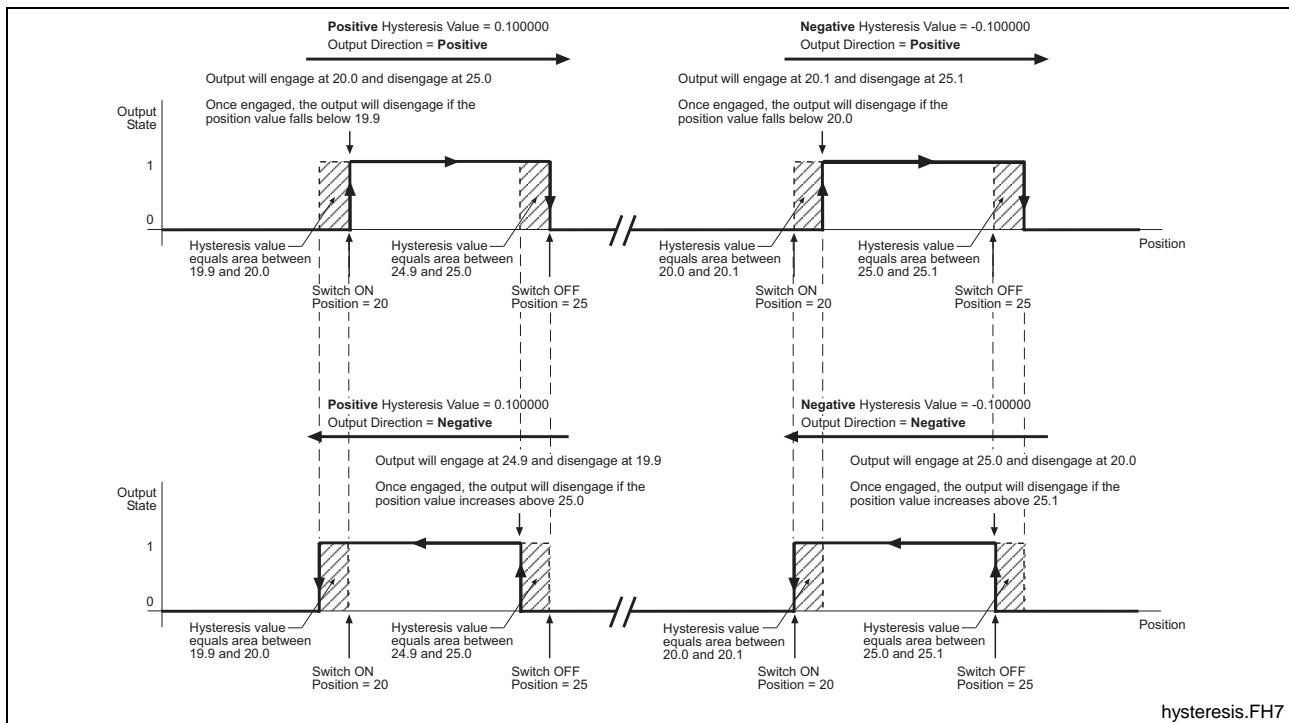


Fig. 4-23: Positive and Negative Hysteresis Value Reaction

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.

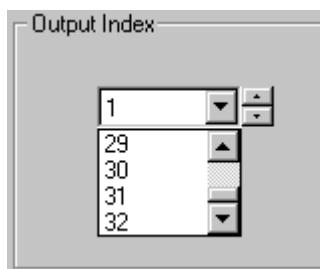
- Positive** When set to positive direction, all switches associated with the output will enable the output **only** in the positive direction with the On switch position and disable the output with the Off switch position.
- Negative** When set to negative direction, all switches associated with the 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.

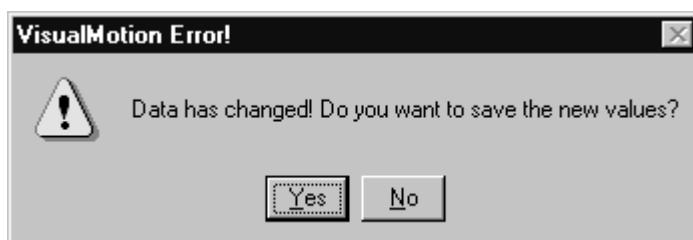
- Positive/Negative** This setting is a combination of the positive and negative direction settings. The outputs will react as described for both the positive and negative direction settings.

Output Index

The *Output Index* section is used to navigate between all 32 outputs.



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.

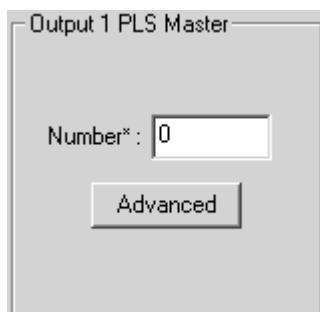


Example:

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.



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 4-19 for details. Every configured output must have a PLS Master assigned to function.

Note: The PLS Output Master Number field can only be updated offline or in parameter mode.

Output 1 Switches

The *Output # Switches* section displays all the configured PLS switches that were assigned to the output.

Output 1 Switches			
Switch#	On Pos	Off Pos	
1	10.000000	20.000000	
Add/Remove			

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.

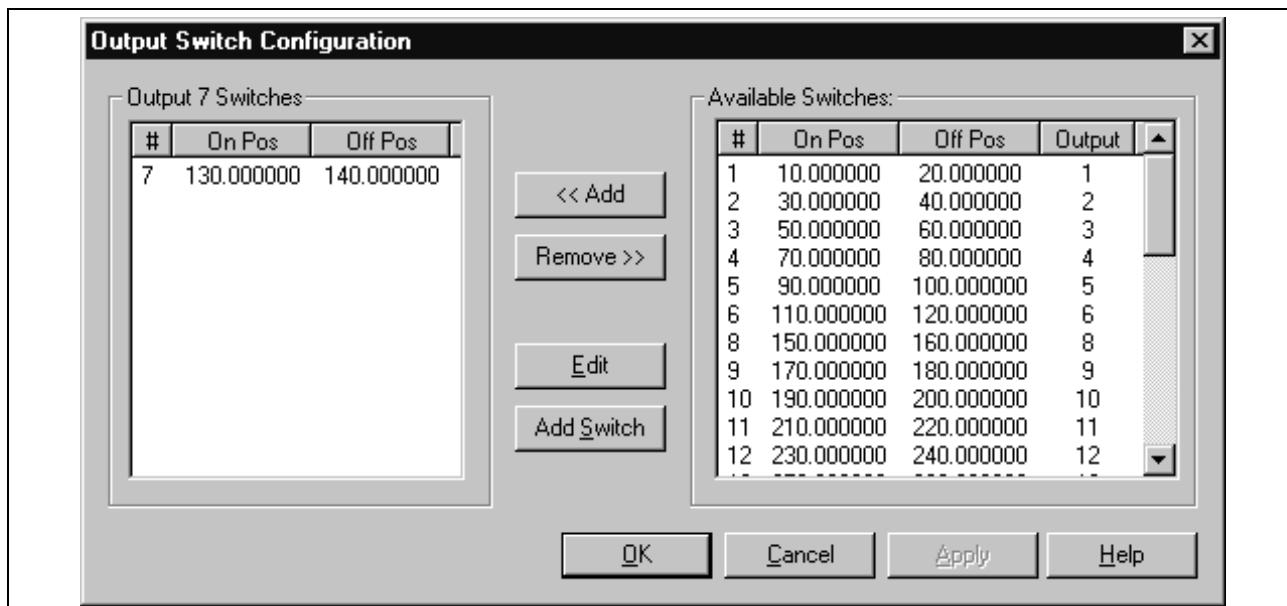


Fig. 4-24: Output Switch Configuration

From the window in Fig. 4-24, 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 4-18 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.

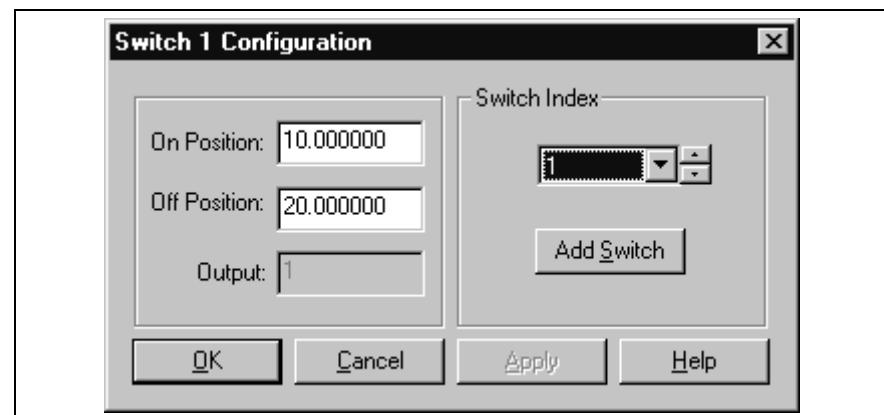


Fig. 4-25: Edit a Switch

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.

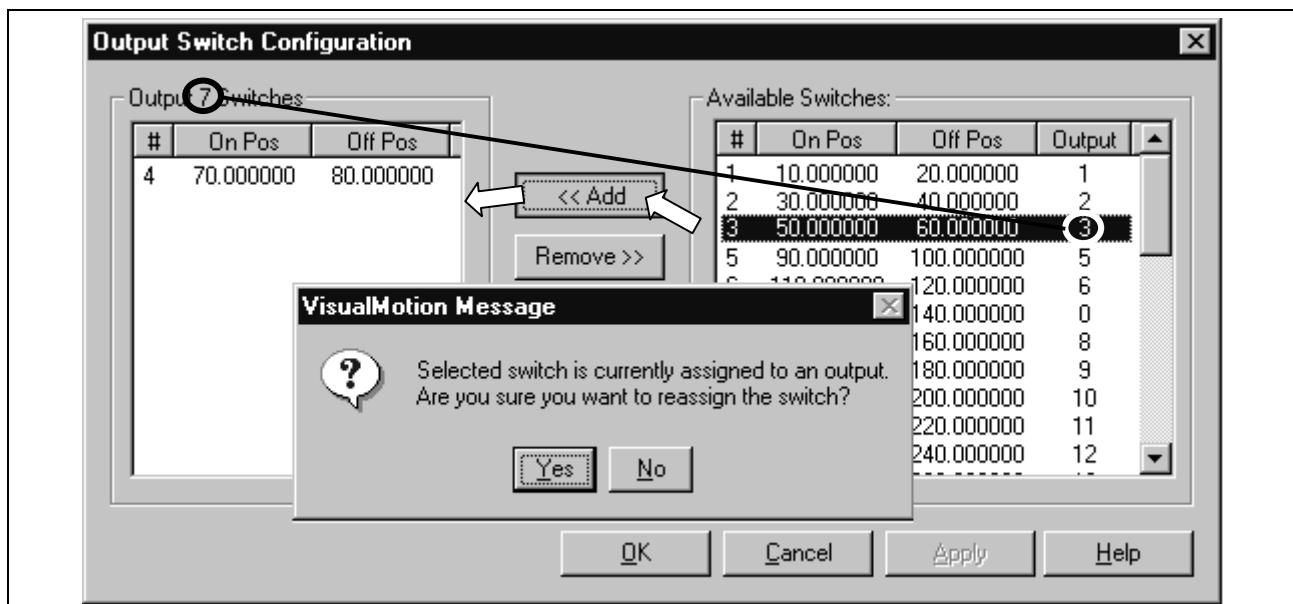


Fig. 4-26: VisualMotion Message for Output Switch Configuration

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.

4.5 Saving and Downloading PLS Configurations

Saving a PLS to a File

PLS configurations are saved to the hard drive by selecting **File** ⇒ **Save As...** or by pressing the Save icon (). The file is saved to the VisualMotion 8 project folder with a *.prm extension.

Note: All PLS configurations (except for a Control PLS) are saved to one file. A Control PLS is only saved with the user program if compiled. Individual PLS configurations can be saved only when configured using **File** ⇒ **New** and saving a single configuration type.

Downloading a PLS to the Control

PLS configurations are downloaded to the control and drives by selecting **File** ⇒ **Send PLS Configuration to GPP Control** or by pressing the download icon ().

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.

After a successful download, the user has the option to build and activate the new High Speed Option Card PLS if present in the system.



Otherwise, all Control and Drive PLS are activated automatically in the active user program and drive respectively.

Note: Remember that an output will not fire unless the complementary mask register bit is set to 1.

4.6 Uploading PLS Configurations

Uploading a PLS Configuration from a File

PLS configurations are 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?"

Uploading a PLS Configuration from the Control

PLS configurations are uploaded to the PLS Configuration tool by selecting ***File*** ⇒ ***Get PLS Configuration from GPP Control*** or by pressing the upload icon ().

Note: When selected, all PLS configuration types (Control, Drive and Card) are uploaded. Control PLS(s) are stored with their respective user programs and are only read and uploaded if contained in the active user program. Drive PLS(s) are scanned over the SERCOS fiber optic ring and uploaded. A Option Card PLS is uploaded by reading the pertinent parameters.

4.7 Editing PLS Configurations

To edit a PLS configuration, first upload all the PLS configurations from the control by selecting ***File*** ⇒ ***Get PLS Configuration from GPP Control*** or by pressing the upload icon (). The PLS Configuration tool will go online to retrieve all active PLS configurations and then return to an offline mode. Fig. 4-27 shows the active PLS configurations uploaded from the control in a tree structure layout.

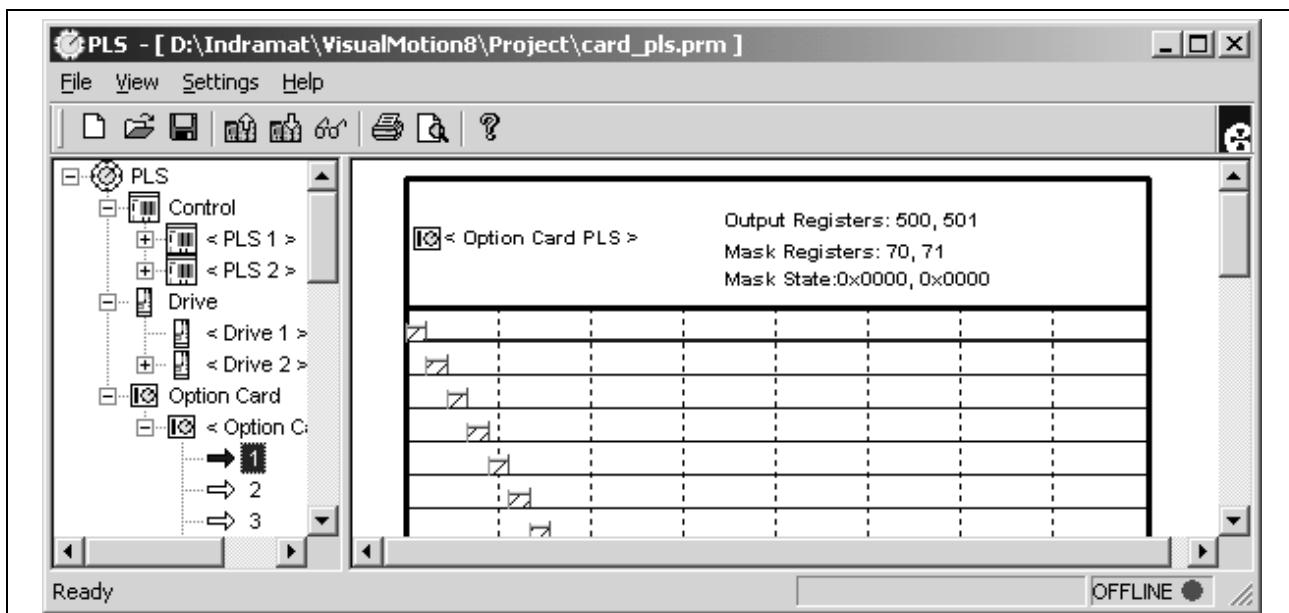


Fig. 4-27: Uploaded PLS Configurations

Select the type of PLS to edit from the PLS tree structure in the left most 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. 4-28 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 4-7
- Configure a Drive PLS on page 4-12
- Configure an Option Card PLS on page 4-17

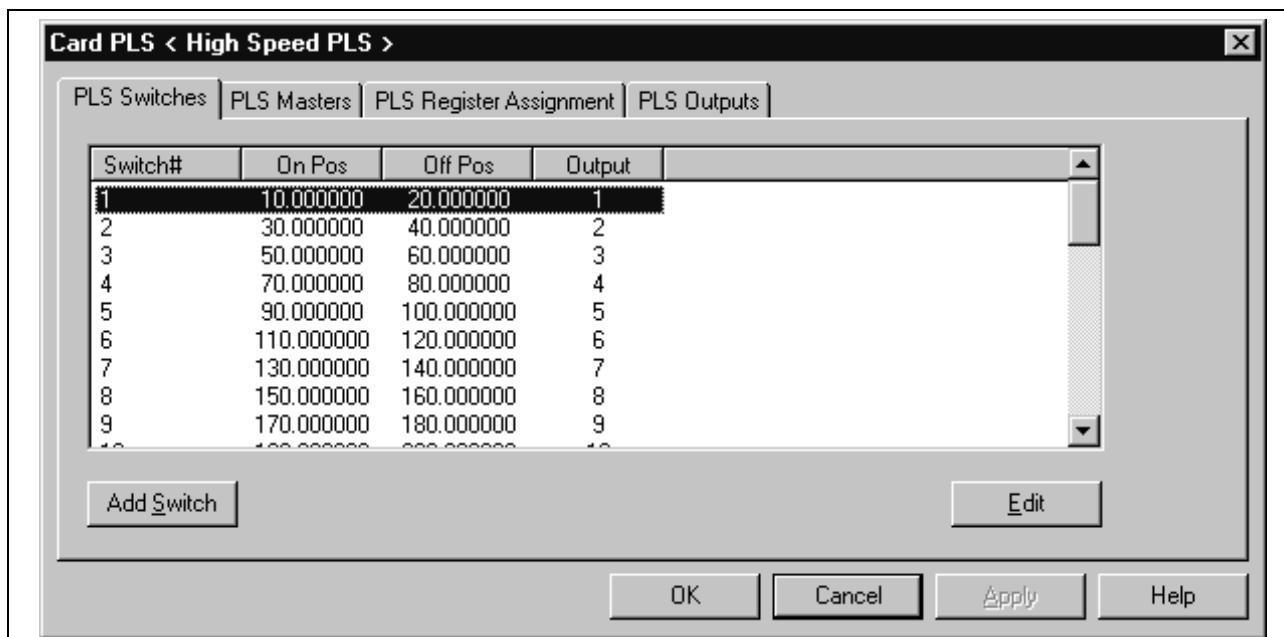


Fig. 4-28: Option Card PLS Edit Window

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:

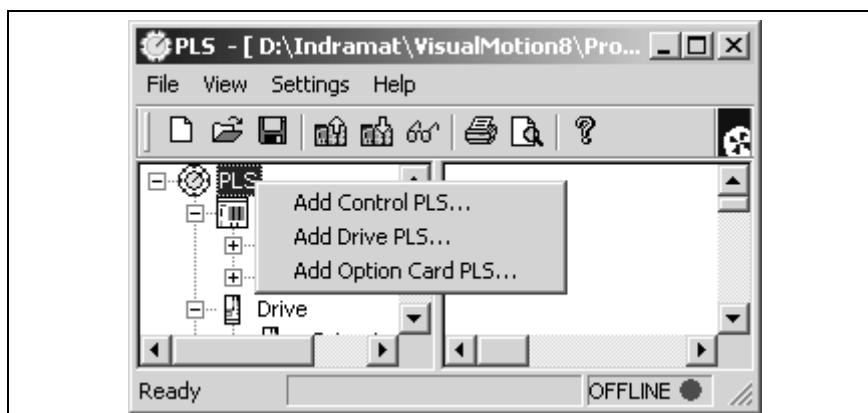


Fig. 4-29: Adding Additional PLSs

Right clicking over an existing PLS type tree icon (Control, Drive or Card) allows the user to add a second or third (if allowed) configuration of that type.

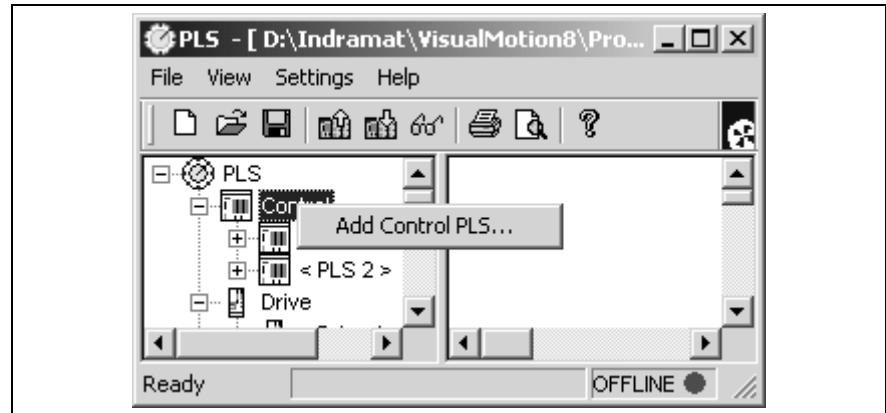


Fig. 4-30: Add a Second Control PLS

Right clicking over an existing PLS configuration allows the user to edit or remove the selected configuration.

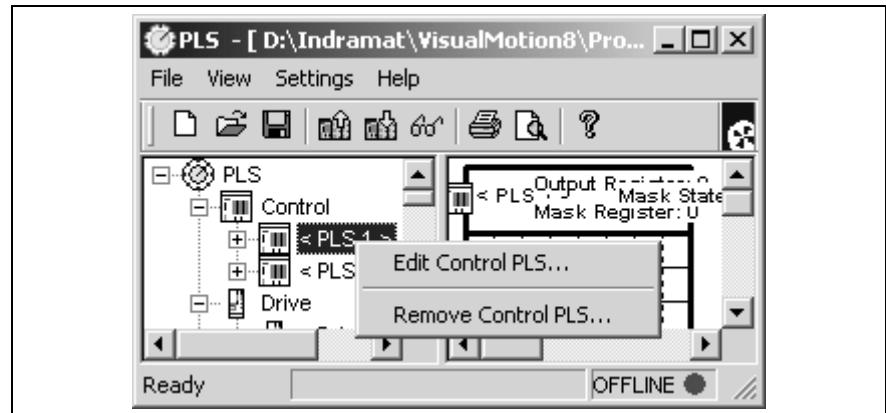


Fig. 4-31: Edit or Remove a PLS Configuration

Note: The above mentioned mouse shortcuts launch the PLS configuration wizard for the corresponding PLS type.

4.8 Monitoring a PLS Status

The user can monitor the status of the output registers by pressing the Monitor Status icon (or the F7 key. When pressed, the PLS Configuration tool switches to online mode, grays out the switch selection window and adds a red LED to each switch. Each output's red LED will turn green when the output is enabled.

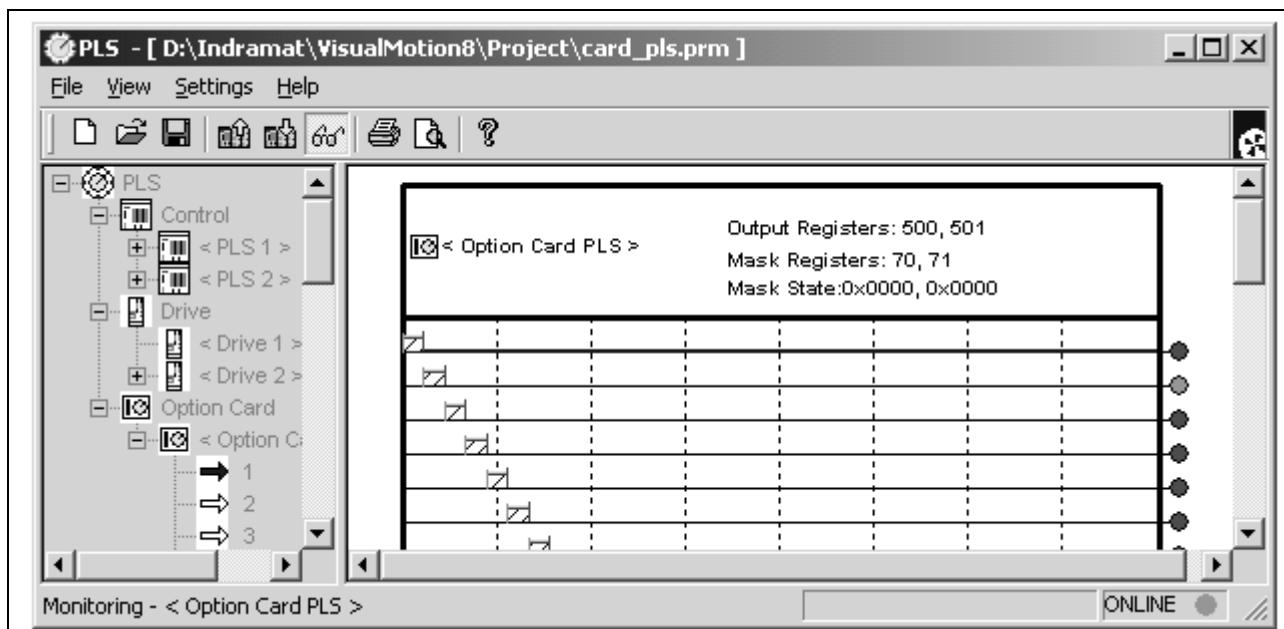
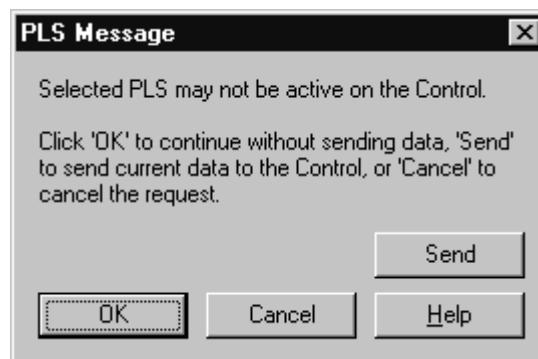


Fig. 4-32: Monitoring a PLS Status

PLS Message

The PLS Configuration tool will issue the following *PLS Message* window if any modifications are made to an active PLS and the Monitor Status icon (or the F7 key is pressed.



Note: If the **Send** button is pressed, the control should be in parameter mode to send all data relevant to the PLS configuration.

4.9 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.

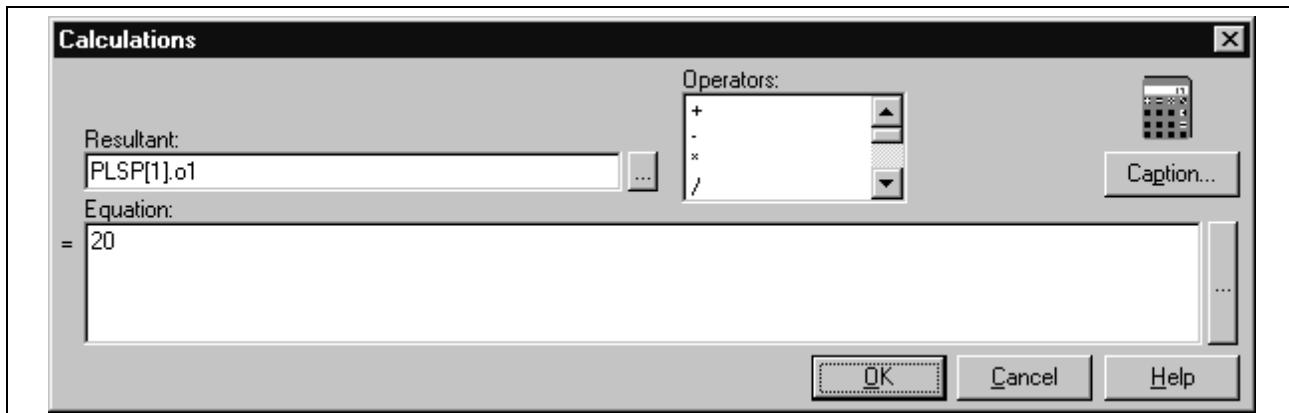


Fig. 4-33: Calc Icon Accessing a PLS

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 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	1-32 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 4-9: Elements of a Control PLS

Drive PLS

VisualMotion 8 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 1-32				

Table 4-10: 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 4-11: 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 4-12: 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].lt x	Lead Time	PLSP[1].lt2=500	add a lead time of 500µs to output 2	C-0-2931
PLSP[1].lg x	Lag Time	PLSP[1].lg2=500	add a lag time of 500µs to output 2	C-0-2932
PLSP[1].ss x	PT (Time Duration)	PLSP[1].ss2=1000	maintain output 2 On for 1000µs	C-0-2933
PLSP[1].hx	Hysteresis	PLSP[1].h2=0.1	add a hysteresis of 0.1 to the output 2	C-0-2936
PLSP[1].md x	Mode	PLSP[1].md2=0	set mode of output 2 to Lag Time (0=Lag Time, 1=PT)	C-0-2934
PLSP[1].dr x	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 4-13: Output Elements of an Option Card PLS

5 Registers

VisualMotion uses registers as a method for communication and system monitoring between the control (using I/O) and an external system. VisualMotion provides 512 registers. Some registers are reserved for control and system functions, others are recommended as defaults, making the remainder available for use by the programmer. Registers can be viewed by selecting **On-Line Data** ⇒ **Registers** from VisualMotion Toolkit's main menu. A breakdown of VisualMotion registers can be found in Table 5-1.

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
026	Fieldbus Resource Monitor	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 Protocol Count	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
150 and 151	Virtual Master 1 & 2 Control (GPP only)	System Default
152-159	ELS Groups 1 – 8 Control (GPP only)	System Default
241 and 242	Virtual Master 1 & 2 Status (GPP only)	System Default
243-250	ELS Groups 1 – 8 Status (GPP only)	System Default
209-232	Axis Control 9-32	Reserved for System
309-332	Axis Status 9-32	Reserved for System

Table 5-1: Control Register

Note: Registers 150 - 159 and 240 - 249 are reserved as system default for VisualMotion's multiple master functionality when configured and assigned as default registers labels with the ELS icons.

Compiler Error under VM08V09, Label too long"

Earlier versions of VisualMotion 8 had 3 register labels exceeding the 20-character label limit (registers 051-053). A compile time check was added to release VM08V09 for this condition. Programs developed with earlier versions of VisualMotion 8 and compiled with release VM08V09 will produce a compiler error, "Label too long".

Invalid register labels:

051 Standard_Message_Count
052 Cyphered_Message_Count
053 Invalid_Protocol_Count

Solution:

To resolve this error in an Icon program, edit register labels for register 051, 052 and 053.

1. Start VisualMotion 8
2. Open the Icon program producing the error.
3. Edit the register labels by selecting **Edit** ⇒ **Edit Labels** ⇒ **Register Labels...** from the main menu.
4. Sort labels by "ID number". Sorting by name will produce confusing results.
5. Edit the labels as follows:
 - 051 Standard_Msg_Count
 - 052 Cyphered_Msg_Count
 - 053 Invalid_Count
6. Save the program and compile.

Note: In addition, VisualMotion keeps a copy of the register labels in each project folder (REGISTER.LST). All Icon programs created in a project use this file as the base default labels. To prevent a new Icon program created in an existing project folder from using these 3 faulty labels, the customer must edit the register label file (REGISTER.LST) with Notepad or an equivalent text editor.

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 5-2: 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 stopped. 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 errors. This also includes system, task and drive errors. If drives were previously enabled, they are restored to an enabled state. To run the tasks, another cycle start transition is required unless the automatic start option is enabled. To clear errors, a low-to-high transition (0-1) is required.

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.

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. The state of the control register bits determine how VisualMotion executes the user task programs. User programs may read, set or clear bits in these registers to monitor and control each task's status. The task control bits permit independent control of the operations within each of the four tasks in a user program.

Register	Register Label Name	Bit	Function
002-005	Task A-D Control	1	Mode:Auto /nManual
		2	Override Automatic Start
		3	Not Used
		4	Single Step
		5	Clear Task Error
		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 5-3: 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.

Switch from Manual to Automatic (0 to 1):

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.

Switch from Automatic to Manual (1 to 0)

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

Switch from Manual to Automatic (0 to 1):

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.

Switch from Automatic to Manual (1 to 0)

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

The ‘Automatically Start Task’ parameter option (T-0-0002, bit 4) allows a task to start immediately upon exit from parameter mode or clearing of an error. The “Override Automatic Start” bit can be used to temporarily disable this function. The bits in the task control register are enabled as long as bit 2 is high (1), and are ignored when the bit is low (0). This allows the automatic start option to be disabled in case the task needs to be stopped or the debugging bits need to be used.

Bit 4: Single Step Select

When Bit 4 = 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 Bit 4 = 0, the task is in Normal Run Mode. 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 5: Clear Task Error

A low-to-high transition (0-1) will clear any errors in the current task.

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.



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.

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.

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 **On-Line Data ⇒ Registers**.

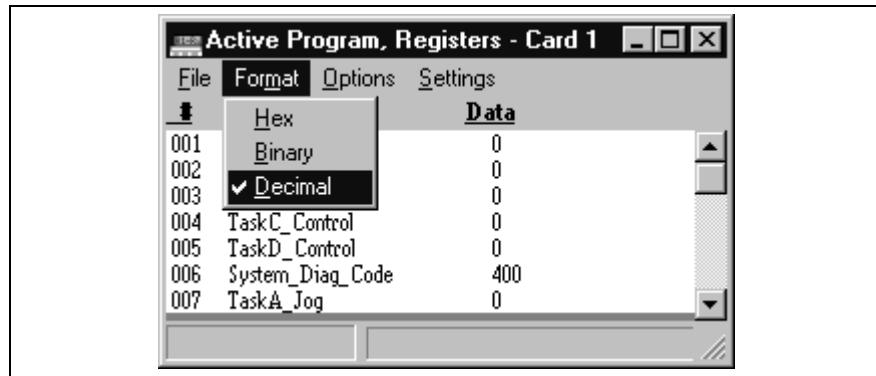


Fig. 5-1: Displaying Register 6

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.

Axis Jogging

Any drive-controlled axis (single-axis or velocity) may be jogged independently when its associated task is in manual mode.

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. 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.

Register	Register Label Name	Bit	Function
007-010	Task A-D Jog	1	Continuous nStep
		2	Coordinate Jog Forward
		3	Coordinate Jog Reverse
		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
		12	Jog Joint 4
		13	Jog Joint 5
		14	Jog Joint 6
		15 and 16	Not Used

Table 5-4: 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.

Bit 1=0, nStep jogging. Motion stops after the large or small distance is reached, or when the jog bit is set to 0.

Bit 1=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 (bit 2) and Reverse (bit 3)

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.

Joint jog: jogs individual axes, with the joint number determined in bits 9 through 14.

Tool jog: jog axes in the tool coordinates selected with bits 9, 10, and 11. It is available only in future 6-axis robot versions.

Bit 5	Bit 4	Jog Type Selected
0	0	World Jog
0	1	Joint Jog
1	0	Tool Jog (6-axis robot version)
1	1	not used

Fig. 5-2: 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

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

Bit number	World and Tool Jog	Joint Jog
12	Roll Axis (SAR only)	Joint 4 (SAR only)
13	Pitch Axis (SAR only)	Joint 5 (SAR only)
14	Yaw Axis (SAR only)	Joint 6 (SAR only)

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.

Registers 011-018, 209-232: Axis Control

Registers 011-018 are used for Axes 1-8. Registers 209-232 are used for Axes 9-32. These Registers are reserved for axis control. These axes correspond to the SERCOS drive address (DSS02.1 for DIAX04 and programming module for ECODRIVE03) 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 209-232	Axis 1-8 and 9-32 Control	1	Disable Axis
		2	Jog Forward
		3	Jog Reverse
		4	Synchronized Jog
		5	Bits 5 through 16 are not reserved by VisualMotion for system control and can be used by the programmer for any other function.
		:	
		:	
		16	

Table 5-5: Registers 011-018: Axis Control

Bit 1: Disable Axis

When this bit is set to (1), all motion is disabled for this axis. The drive is immediately set to zero velocity and the position loop is disabled. Motion commands in the user program and from the control have no effect. Note that the drive can produce torque to hold position. It is still in the SERCOS ring. I/O and some diagnostics are still enabled.

When set to (0), motion is enabled, provided other conditions allow it and there are no errors on the drive or 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

ELS slave axes may be jogged independently or synchronously when their associated task is in manual mode.

When the Synchronized Jog bit is set to (0), the axis can be jogged independently. It does not follow the master, and it may be jogged continuously or incrementally in single-axis mode. Axes can be jogged individually to set up phase offsets or to prepare the machine for operation.

When the Synchronized Jog bit is set to (1), the slave axis follows the real or virtual master. This jog mode can be used to initially thread the material into sections of the machine, maintaining the position or velocity relation to the master. The master is jogged using the coordinated jog bits in the Task Jog register.

Register 019: Fieldbus 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 5-6 below contains the bit assignment for the diagnostic object 5ff2.

Register	Register Label Name	Bit	Function
019	Fieldbus 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	Not Used
		15	Cyclic Data Valid
		16	Not Used

Table 5-6: Register 019: Fieldbus Status

Bit 1 and 2: Fieldbus Initialization OK; LSB (bit 1) and MSB (bit 2)

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)

The combination for bit 1 and 2 are listed in Table 5-7:

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 5-7: 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 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.

Register 020: Fieldbus Diagnostics

Register 020 holds the information for "Fieldbus Diagnostics." Table 5-8 below contains the bit assignment for the diagnostic object 5ff0.

Register	Register Label Name	Bit	Function
020	Fieldbus 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 5-8: Register 020: Fieldbus Diagnostics

Bits 13- 16: Fieldbus 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	Ethernet
0	1	1	1	<Not Defined>
1	1	1	1	Indramat PLC Interface

Table 5-9: Identification of the Fieldbus Interface

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 – 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 5-10: 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).

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.

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
022-025	Task A-D Status	1	Mode Automatic nManual
		2	Coordinated Running
		3	Not Used
		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 5-11: 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

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.

Register 026: Fieldbus Resource Monitor

The "Fieldbus Resource Monitor" can be used as a method for monitoring the attempts made to process the Cyclic Mapping Lists in parameters C-0-2600 and C-0-2601 across the Dual-port RAM. If after 8 ms, the Cyclic Mapping Lists are not successfully transmitted, a "miss" is noted. Register 026 is divided into the following three counter types:

- Current Miss Counter.
- Peak Miss Counter.
- Fieldbus Timeout Counter.

Register	Register Label Name	Bit	Function
026	Fieldbus Resource Monitor	1	FB Map Timeout, LSB
		2	FB Map Timeout Bit 2
		3	FB Map Timeout Bit 3
		4	FB Map Timeout Bit 4
		5	FB Map Timeout Bit 5
		6	FB Map Timeout Bit 6
		7	FB Map Timeout Bit 7
		8	FB Map Timeout, MSB
		9	FB Peak Miss Counter, LSB
		10	FB Peak Miss Counter Bit 2
		11	FB Peak Miss Counter Bit 3
		12	FB Peak Miss Counter, MSB
		13	FB Current Miss Counter, LSB
		14	FB Current Miss Counter Bit 2
		15	FB Current Miss Counter Bit 3
		16	FB Current Miss Counter, MSB

Table 5-12: Register 026: Fieldbus Resource Monitor

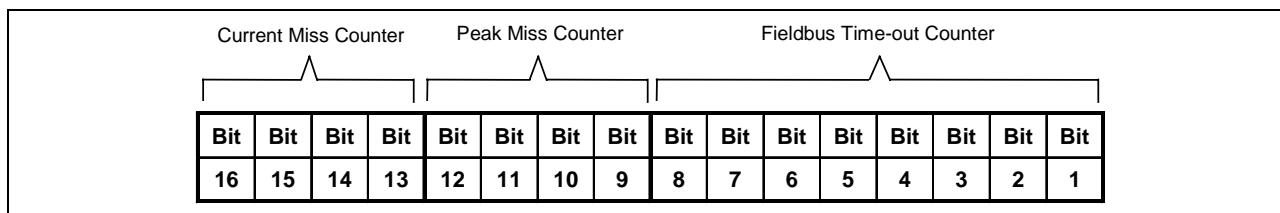


Fig. 5-3: Bit Assignment for Fieldbus Resource Monitor

Note: View Register 026 in Hexadecimal format to monitor the fieldbus counters more easily.

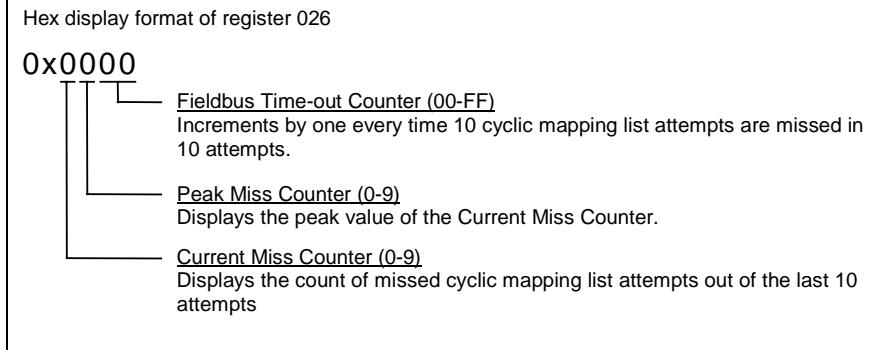


Fig. 5-4: Hexadecimal Display of Register 026

Note: To view registers in hex, select **On-Line Data** ⇒ **Registers** within VisualMotion Toolkit. If the registers are not currently viewed in hex format, select **Format** ⇒ **Hex** in the **Active Program, Register** window.

Bits 1- 8: Fieldbus Timeout Counter

Status bits for the "Fieldbus Timeout Counter."

The count of these bits increments by one every time the "Current Miss Counter" encounters 10 out of 10 missed attempts of the cyclic mapping list update. A count incremented by one represents a Fieldbus Mapping Timeout Error and is processed by GPP according to the selected "Fieldbus Error Reaction" in parameter C-0-2635.

Bits 9- 12: Peak Miss Counter

Status bits for the "Peak Miss Counter."

These bits monitor the "Current Miss Counter's" peak count between a value from 0-9 and hold that value until a larger count is encountered.

Bits 13- 16: Current Miss Counter

Status bits for the "Current Miss Counter."

An attempt is made to transmit the Cyclic Mapping Lists, C-0-2600, across the Dual-port RAM every 8 ms. For every 10 mapping list update attempts (80 ms), the failed attempts are counted and displayed in these bits (values can range from 0-9). If 10 out of 10 mapping list update attempts are missed, the "Fieldbus Timeout Counter" is incremented by one. This is an indication of a Fieldbus Mapping Timeout Error.

Registers 031-038, 309-332: Axis Status

Registers 031-038 are for axes 1-8. Registers 309-332 (read only) for Axes 9-32. 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 309-332	Axis 1-8 and 9-32 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
		11	Axis Halted
		12	Class 3 Status
		13	Class 2 Warning
		14	Shutdown Error
		15	Drive Ready LSB
		16	Drive Ready MSB

Table 5-13: Registers 031-038: Axis Status

Bit 1: Multiplex Channel Enabled

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 (CLC 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

This bit corresponds to Drive Parameter S-0-0182 bit 1. It is set to (1) when the feedback velocity is less than the drive's zero velocity window.

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 5-14: Description of Bits 15 and 16 for Axis Status

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 5-15: 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 in Primary Optic Ring status bit is set when the node has detected a fiber optic cable break in the Link Ring's primary SERCOS ring. 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 in Secondary Optic Ring status bit is set when the node has detected a fiber optic cable break in the Link Ring's secondary SERCOS ring. 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.

Registers 041: Link Ring Data 1

This register will provide a bit to indicate that each node (1-16 respectively) is reliably communicating its Master position data across the Link Ring.

Register	Register Label Name	Bit	Function
041	Link Ring Data 1	1-16	Nodes 1-16 (respectively) Data Valid

Table 5-16: Registers 041: Link Ring Data 1

Bit 1-16: Node 1-16 Data Valid

When a bit is set high (1), the Master position data can be communicated reliably across the Link Ring.

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.

Registers 042: Link Ring Data 2

This register will provide a bit to indicate that each node (17-32 respectively) is reliably communicating its Master position data across the Link Ring.

Register	Register Label Name	Bit	Function
042	Link Ring Data 2	17-32	Nodes 17-32 (respectively) Data Valid

Table 5-17: Registers 042: Link Ring Data 2

Bit 17-32: Node 17-32 Data Valid

When a bit is set high (1), the Master position data can be communicated reliably across the Link Ring.

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.

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 5-18: 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 12: Invalid Protocol

A (1) indicates that an invalid protocol has been received (standard or encrypted ASCII).

Register 051: Standard Message Count

This register indicates the number of messages that have been received in standard ASCII protocol.

Register 052: Cyphered Message Count

This register indicates the number of messages that have been received in encrypted ASCII protocol.

Register 053: Invalid Protocol Count

This register indicates the number of messages that have been received with a protocol that can not be determined.

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.

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.

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.

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.

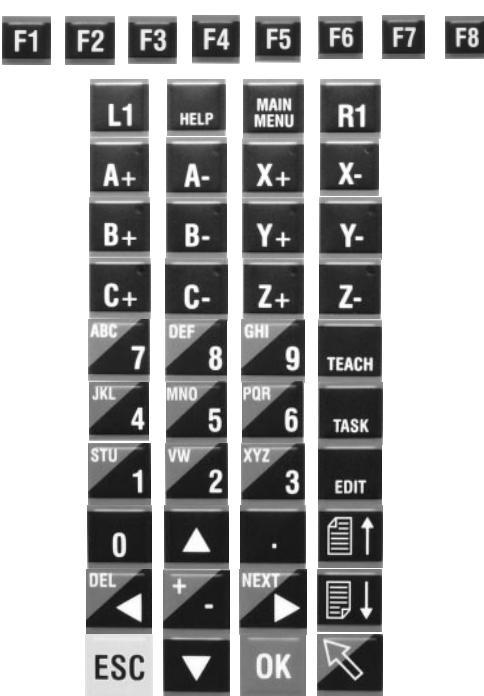
Note: These bits are all read-only.

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.



F1	F2	F3	F4	F5	F6	F7	F8	95-01	95-02	95-03	95-04	95-05	95-06	95-07	95-08
L1	HELP	MAIN MENU	R1					97-13	95-09	97-14	97-15				
A+	A-	X+	X-					96-05	96-06	96-07	96-08				
B+	B-	Y+	Y-					96-13	96-14	96-15	96-16				
C+	C-	Z+	Z-					97-05	97-06	97-07	97-08				
ABC	DEF	GHI	TEACH					95-10	95-11	95-12	95-13				
7	8	9						96-02	96-03	96-04	95-14				
JKL	MNO	PQR	TASK					96-10	96-11	96-12	96-09				
4	5	6						97-03	95-11	97-02	95-15				
STU	VW	XYZ	EDIT					96-02	97-04	96-04	95-16				
1	2	3						96-01	96-11	97-01					
0	▲	.													
DEL	+	NEXT													
◀	▼	OK													
ESC															

This shift key is not mapped to a register.

Fig. 5-5: Teach Pendant: Function Keys to Register Map

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 5-19: Register 098: Pendant Control – Task A-B

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 5-20: Register 099: Pendant Control – Task C-D

Registers 150 and 151: Virtual Master 1 & 2 Control

GPP 8 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.

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 5-21: 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. 5-6: 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 5-22: 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.

Registers 152-159: ELS Groups 1-8 Control

Register	Register Label Name	Bit	Function
152-159	ELS Group 1-8 Control	1	G#_CT_LOCK_OFF
		2	G#_CT_MSTR_PH
		3	G#_CT_SLV_PH
		4	G#_CT_MSTR_SEL
		5	G#_CT_VAR_CLK
		6	G#_CT_LOCAL
		7	G#_CT_JOG_CONT
		8	G#_CT_JOG_ABS
		9	G#_CT_JOG_PLUS
		10	G#_CT_JOG_MINS
		11-16	Not Used

Table 5-23: Registers 152-159: ELS Group Control

Bit 1: Group Lock Off/ Lock on Cams

A low-to-high (0-1) transition actives 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 actives 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 Phase Adjust

A low-to high (0-1) transition starts the master phase adjust. Bit 2 (G#_ST_MSTR_PH) of the status register goes to a 1 at the start of the phase adjust, and to a 0 when the phase adjust is completed.

Bit 3: Group Slave Phase Adjust

A low-to high (0-1) transition starts the slave phase adjust. Bit 3 (G#_ST_SLV_PH) of the status register goes to a 1 at the start of the phase adjust, and to a 0 when the phase adjust is completed.

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 5-24: Bit 9: Group Jog in Positive Direction.

Bit 10: Group Jog in Negative Direction

This bit functions the same as bit 9 but in the negative direction.

Registers 241 and 242: Virtual Master 1 & 2 Status

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 5-25: Registers 241-242: Virtual Master 1 & 2 Status

Bit 1: Virtual Master 1 or 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.

Registers 243 - 250: ELS Groups 1-8 Status

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
		11-16	Not Used

Table 5-26: 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#_ST_MSTR_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_MSTR_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_SLV_PH) 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 jogged.

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 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 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.

6 Parameters

6.1 Overview

Rexroth Indramat's VisualMotion control provides user-configurable 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 may be specific to an application and/or a single machine installation. These 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:

- System (Control)
- Task
- Axis
- Drive (SERCOS and Product)

System parameters contain setup and status parameters for the control. Task and Axis parameters are associated with user programs. System, Task and Axis parameters are downloaded and stored in the control. Digital drive parameters are stored in each SERCOS-compatible digital drive and are necessary for configuring a motion system.

Each parameter within a class also has a set and number identifier. All parameter elements (data, name, units) can be accessed using VisualMotion Toolkit or from a user specified interface such as an ASCII terminal.

6.2 Parameter Identification

All parameters in VisualMotion have a unique **IDentification Number (IDN)** and use the following formats:

SERCOS Format	ASCII Format																		
<pre>## X-s-nnnn _ 4 digit parameter number __ Set: 0 or 7 __ Class: Parameter type _____ Class Number</pre> <p>Example: 01 C-0-0001 Language selection control parameter for drive 1.</p> <p>Legend:</p> <ul style="list-style-type: none"> ##: Class Number (01-32) SERCOS drive address for C,S,P and A 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 between (0001 - 4095). 	<pre>XP ##.nnnn _ 1 to 5 digit parameter number __ Class Number __ Subclass: P = Parameters _____ Class: Parameter type</pre> <p>Example: CP 1.1 Language selection control parameter for drive 1.</p> <p>n : 1 to 4 digit parameter number for C,T,A and S parameters. 5 digit parameter number for P parameters. An offset of 32768 (0x8000) must be added to the number.</p> <p>Example:</p> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; width: 33.33%;"></th> <th style="text-align: center; width: 33.33%;"><i>SERCOS</i></th> <th style="text-align: center; width: 33.33%;"><i>ASCII</i></th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1 digit:</td> <td style="text-align: center;">01 A-0-0002</td> <td style="text-align: center;">SP 1.2</td> </tr> <tr> <td style="text-align: center;">5 digit:</td> <td style="text-align: center;">01 A-0-4014</td> <td style="text-align: center;">PP 1.36768</td> </tr> </tbody> </table> <p>To access 7-set S drive parameters, add an offset of 28672 (0x7000) to the number. For P drive parameters, add an offset of 61440 (0xF000)</p> <p>Example:</p> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; width: 33.33%;"></th> <th style="text-align: center; width: 33.33%;"><i>SERCOS</i></th> <th style="text-align: center; width: 33.33%;"><i>ASCII</i></th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">01 S-7-0100</td> <td style="text-align: center;">DP 1.28772</td> <td></td> </tr> <tr> <td style="text-align: center;">01 P-7-0004</td> <td style="text-align: center;">DP 1.61444</td> <td></td> </tr> </tbody> </table>		<i>SERCOS</i>	<i>ASCII</i>	1 digit:	01 A-0-0002	SP 1.2	5 digit:	01 A-0-4014	PP 1.36768		<i>SERCOS</i>	<i>ASCII</i>	01 S-7-0100	DP 1.28772		01 P-7-0004	DP 1.61444	
	<i>SERCOS</i>	<i>ASCII</i>																	
1 digit:	01 A-0-0002	SP 1.2																	
5 digit:	01 A-0-4014	PP 1.36768																	
	<i>SERCOS</i>	<i>ASCII</i>																	
01 S-7-0100	DP 1.28772																		
01 P-7-0004	DP 1.61444																		

Fig. 6-1: Parameter format

Note: Parameters in this chapter will be described using the SERCOS format.

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 "C" parameters have only a 0-set.

(Example: *C-0-0nnn*, etc.)

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.

(Example: Task A *01 T-0-nnnn*, Task C *03 T-0-nnnn*, etc.)

Class "A" axis parameter types include the parameters used to configure and display information about the each axis. A maximum of 32 different sets of parameters is possible. A set corresponding to one of the eight axes is identified by a single digit in the range 01-32.

(Examples: Axis 01 A-0-nnnn, Axis 03 A-0-nnnn, etc.)

Class "S" SERCOS drive parameters are accessed through the SERCOS Service Channel. The drive parameter sets 1-8 directly correspond to the SERCOS drive address 1-8 and the control axes 1-8.

Class "P" Product specific drive parameters include the parameters required to configure and operate Rexroth Indramat's digital drives. These parameters are unique to Indramat drives and are necessary for proper communication between the drives and external devices.

(Examples: Drive 01 A-0-nnnn, Drive 03 A-0-nnnn)

Most servo and position loop parameters are contained in SERCOS compatible digital drive. 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. A list of more common drive parameters is included at the end of this chapter. For a complete description of the "S" and "P" parameters, refer to the corresponding SERCOS and drives manual.

Some parameters may be accessed by the user program through the VisualMotion parameter transfer commands or the text programming language parameter commands. VisualMotion also allows access to specific groups of parameters for editing and archiving. See the relevant chapters on Icon Programming or Text Language Programming and the section on Direct ASCII Communication for information on transferring parameters.

Lists of all parameters and all non-volatile parameters are available through the control's serial communication port. Transferring parameters using direct ASCII communication follows the same format, with various subclasses for names, units, limits, and data. See the chapter on Direct ASCII Communication for detailed information on the protocol and syntax for direct communication.

6.3 Parameter Transfer Commands

Parameter transfer capability is built into VisualMotion's programming language. Parameter transfers allow the monitoring of system values and SERCOS command execution. The System, Axis, and Task parameter descriptions in this section specify the type of data (Float, Integer, etc.) set returned by the PARAMETER/BIT, PARAMETER/GET, PARAMETER/INIT, PARAMETER/SET text language commands.

All control and drive parameters of the floating point type may be read into a VisualMotion floating point 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 classified as "Read/write at any time" may be changed by a user program.

To transfer SERCOS parameters not listed in this section, refer to the corresponding SERCOS and drive manuals for a specific parameter's type of data. In general: feedback, tuning, and measurement values are floating point; procedure commands are integer. Use the List Parameter capability to obtain a list of specific parameters, then find the type of variables that can contain specific parameters.

The full capability of a SERCOS drive can be accessed through a VisualMotion user program. Procedural commands such as homing can be performed from a user program by initiating the SERCOS homing sequence, then testing the appropriate parameter to determine when the operation has completed. The SERCOS probing functions may be used to implement registration to a mark, or for accurate positioning measurements. In addition, some of the limits and offsets can be set up at program initialization, or a user program may be written to dynamically tune a drive based on a system's performance feedback.

6.4 List of Parameters

The information provided in this chapter will focus on the following VisualMotion System, Drive, Axis, and Task parameters. For specific drive and SERCOS parameter information, refer to the corresponding SERCOS and drive manuals.

System (Control) Parameters - Class C

VisualMotion system parameters include System Setup, PLC Interface, Jogging and Display, Program Management, System Status, Control Processor Usage, Link Ring, Ethernet, Pendant, System Monitoring, Memory Parameters, Parameter Lists, Oscilloscope, Fieldbus Interface, Card PLS Interface, I/O Mapper and Cam Tables.

System Setup Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-0001	Language Selection	X	X
C-0-0002	Unit Number	X	X
C-0-0003	Serial Port A Setup	X	X
C-0-0004	Serial Port B Setup	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-0-0020	Transmitter Fiber Optic Length	X	X
C-0-0021	User Watchdog Timer	X	X
C-0-0022	User Watchdog Task ID	X	X
C-0-0035	PLC Communication Option	X	X

Table 6-1: System Setup Parameters

PLC Interface Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-0035	PLC Communication Option	X	X
C-0-2641	Indramat/PLC Input Register List		X
C-0-2642	Indramat/PLC Output Register List		X

Table 6-2: PLC Interface Parameters

Jogging and Display Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-3: Jogging and Display Parameters

Program Management Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP8
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-0-0095	Path Planner to SERCOS Time Factor	X	X
C-0-0098	Initialization Delay	X	X
C-0-0099	Minimum SERCOS Cycle Time	X	X

Table 6-4: Program Management Parameters

System Status Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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
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

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-0127	Current PPC-R Temperature	X	X
C-00142	Card Label String		X

Table 6-5: System Status Parameters

Control Processor Usage Status Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-6: Control Processor Usage Status Parameters

Link Ring Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-0300	Link Ring Control Word		X
C-0-0301	Link Ring Primary Fiber Optic Length		X
C-0-0302	Link Ring Secondary Fiber Optic Length		X
C-0-0303	Link Ring MDT Error Counter		X

Table 6-7: Link Ring Parameters

Ethernet Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-0400	Card IP Address		X
C-0-0401	Card Subnet Mask		X
C-0-0402	Card Gateway IP Address		X
C-0-0403	Card CIF Network Control		X
C-0-0404	Card Access Network Control		X
C-0-0405	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 Driver Version		X

Table 6-8: Ethernet Parameters

BTC06 Teach Pendant Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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	BTC06 message and prompt control word	X	X
C-0-0811	User Task Controlled Menu ID for BTC06	X	X
C-0-0812	User Task Controlled Task ID for BTC06	X	X
C-0-0813	User Task Controlled Axis Number for BTC06	X	X
C-0-0814	BTC06 Data Transaction Word	X	X

Table 6-9: Pendant Parameters

Internal System Monitoring

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-0990	Exit to Monitor Prompt	X	X

Table 6-10: Internal System Monitoring

System Memory Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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

Table 6-11: System Memory Parameters

System Parameter Lists

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-2000	List of All Parameters	X	X
C-0-2001	List of Required 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

Table 6-12: System Parameter Lists

Oscilloscope Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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-0-2504	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-0-2507	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-0-2510	Oscilloscope sample rate	X	X
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-0-2514	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 6-13: Oscilloscope Parameters

Fieldbus Interface Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-2600	Fieldbus Mapper (cyclic channel) To PLC	X	X
C-0-2601	Fieldbus Mapper (cyclic channel) From PLC	X	X
C-0-2630	Fieldbus Slave Device Address	X	X
C-0-2631	Fieldbus Parameter Channel Length	X	X
C-0-2632	Fieldbus Multiplex Method	X	X
C-0-2633	Fieldbus Baud Rate (DeviceNet only)	X	X
C-0-2635	Fieldbus Error Reaction	X	X
C-0-2636	Fieldbus Word Swap (DeviceNet only)	X	X
C-0-2637	Fieldbus Slave Firmware Version	X	X
C-0-2638	Fieldbus Available Cyclic IN Parameters	X	X
C-0-2639	Fieldbus Available Cyclic OUT Parameters	X	X
C-0-2700	Fieldbus Mapper (acyclic channel) To PLC	X	X
C-0-2701	Fieldbus Mapper (acyclic channel) From PLC	X	X

Table 6-14: Fieldbus Interface Parameters

Card PLS Interface Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
C-0-2901	PLS1 Start Output Register		X
C-0-2902	PLS1 Start Mask Register		X
C-0-2903	PLS1 Build Table Command		X
C-0-2904	PLS1 Build Table Status		X
C-0-2905	PLS1 Switch Table Command		X
C-0-2906	PLS1 Switch Table Status		X
C-0-2907	PLS1 Error Code		X
C-0-2908	PLS1 Extended Error Code		X
C-0-2909	PLS1 Hardware ID		X
C-0-2910	PLS1 Software ID		X
C-0-2920	PLS1 Switch On List		X
C-0-2921	PLS1 Switch Off List		X
C-0-2922	PLS1 Switch Output List		X
C-0-2930	PLS1 Output Master List		X
C-0-2931	PLS1 Output Lead Time		X
C-0-2932	PLS1 Output Lag Time		X
C-0-2933	PLS1 Output One Shot (PT) List		X

C-0-2934	PLS1 Output Mode List		X
C-0-2935	PLS1 Output Direction List		X
C-0-2936	PLS1 Output Hysteresis		X
C-0-2940	PLS1 Master Type List		X
C-0-2941	PLS1 Master Number List		X
C-0-2942	PLS1 Master encoder List		X
C-0-2943	PLS1 Master Phase Offset List		X

Table 6-15: Card PLS Interface Parameters

I/O Mapper Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-16: I/O Mapper Parameters

Cam Table Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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

Table 6-17: Cam Table Parameters

Task Parameters - Class T

VisualMotion user task parameters include Task Setup, Coordinated Motion, Robotics, Coordinated Motion Status, Task Status and Parameter Lists.

Task Setup Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
T-0-0001	Task Motion Type	X	X
T-0-0002	Task Options	X	X

Table 6-18: Task Setup Parameters

Coordinated Motion

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
T-0-0005	World Position Units	X	X
T-0-0010	Kinematic 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	Maximum Path Jerk		X

Table 6-19: Coordinated Motion Parameters

Robotics

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
T-0-0035	Relative Point Used for Origin	X	X
T-0-0036	Relative Point Used for Tool Frame	X	X
T-0-0050	Kinematic Value 1	X	X
T-0-0051	Kinematic Value 2	X	X
T-0-0052	Kinematic Value 3	X	X
T-0-0053	Kinematic Value 4	X	X
T-0-0054	Kinematic Value 5	X	X
T-0-0055	Kinematic Value 6	X	X
T-0-0056	Kinematic Value 7	X	X

T-0-0057	Kinematic Value 8	X	X
T-0-0058	Kinematic Value 9	X	X
T-0-0059	Kinematic Value 10	X	X

Table 6-20: Robotics Parameters

Coordinated Motion Status

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-21: Coordinated Motion Status Parameters

Task Status

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-22: Task Status Parameters

Task Parameter Lists

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
T-0-2000	List of All Parameters	X	X
T-0-2001	List of Required Parameters	X	X

Table 6-23: Task Parameters Lists

Axis Parameters - Class A

VisualMotion axis parameters include Axis Setup, Axis Status, Feedback Capture, Optional SERCOS Data, Multiplexing and Parameter lists.

Axis Setup Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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
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

Table 6-24: Axis Setup Parameters

Axis Status Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-25: Axis Status Parameters

Axis Electronic Line Shafting Parameters

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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-0-0157	Current Phase/ Control Cam Master Offset	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 (Sph)	X	X
A-0-0163	Control Cam Output Position	X	X
A-0-0164	ELS Options	X	X

Table 6-26: Axis Electronic Line Shafting Parameters

Axis Feedback Capture (Registration)

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-27: Axis Feedback Capture (Registration)

Optional SERCOS Data

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-28: Optional SERCOS Data

Multiplexing Parameters (DKC 2.3 only)

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
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 6-29: Multiplexing Parameters

Axis Parameter Lists

Parameter Number	Description	Valid for	
		GPP 7	GPP 8
A-0-2000	List of All Parameters	X	X
A-0-2001	List of Required Parameters	X	X

Table 6-30: Axis Parameters Lists

6.5 System Control Parameters – Class C

System 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 7 and GPP 8 firmware. For this reason, any system parameter that is used specifically for GPP 7 or GPP 8 will be indicated at the end of the description.

For example:

"C-0-0104 Bootloader Firmware Version (**GPP 8 only**)"

No specification to a firmware version indicates that the message is valid for both GPP 7 and GPP 8. Refer to the tables on page 6-5 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 several languages. This parameter determines the output language for the text. The language type is identified as follows:

- 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 Unit Number

VisualMotion allows up to 32 controls to be accessed across a Multi-Drop RS-485 connection. This parameter sets the network address for each control. For VisualMotion Toolkit to communicate, the serial port that is connected to the control must be set to RS-485. Also, change the unit number in VisualMotion Toolkit by selecting ***Settings*** ⇒ ***Card Selection*** and enter the number of the control.

C-0-0002 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	31
Default value:	0
Access:	read / write in any Phase

C-0-0003 Serial Port A Setup

This parameter sets the baud rate and options for Serial Port A (X10 on PPC-R), which communicates with a PC, a terminal, or any device that follows the VisualMotion ASCII Host Protocol. The port always operates with 8 data bits, 1 stop bit, and no parity.

If the checksum option is enabled, the control will send a checksum as described in the Host Protocol Description, and will check the checksum in data received from the host. The options are added together with the baud rate. For example, to set 9600 baud, checksum on, the parameter would be set to $(9600 + 1) = 9601$. The allowable baud rates are 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200.

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 and options for Serial Port B (X16 on PPC-R). The device connected to this port is selected in parameter C-0-0012 Serial Port B Device Type. The port always operates with 8 data bits, 1 stop bit, and no parity. Refer to the description of parameter C-0-0003 Serial Port A Setup for Host Protocol options. Additional options are added to the baud rate to enable checksum (1) or status message enable (2).

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-0-0009 Error Reaction Mode

This parameter selects VisualMotion's reaction to a fatal shutdown error on the control or the drive. Fatal errors include Emergency Stop, Zone Violation, or Drive Class 1 Shutdowns.

When this parameter is set to 0 (default), the drives are immediately disabled. Velocity is immediately set to zero and the brakes are engaged.

When this parameter is set to 1, all motion will come to a controlled stop before the drives are disabled. In the case of coordinated or single-axis motion, the maximum task or axis deceleration will be used. For an ELS Virtual Master, the E-Stop deceleration will be used. This Parameter works with drive parameters A-0-0117, A-0-0118 and A-0-0119.

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 any 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 6-9: Reserved for future development
	Bit 10: System timing monitoring
	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. 6-2: 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 any Phase 2

Bit 1: Simulation Mode

When set to 1, the 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.

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 (Parameter Mode), all drive-resident I/O is scanned every 500ms by default (bits 2 and 3 =0). This can slow communications when downloading parameter lists from the drives, depending upon how much I/O is visible. If it is not critical to have fast updates of I/O in Phase 2, set these bits 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

When bit 5 is set to 1, the control transmits at 4 Mbits/s. When it is 0, 2 Mbits/s. All Indramat drives except DIAX01 and DIAX02 can be set for 4 Mbits/s, via a switch on the SERCOS interface card. The only applications that may require 4 Mbits/s are those with more than 8 drives. In all other cases, the 2Mbits/s rate should be used. 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 is 2ms.

Bits 6-9: Reserved for future development

Bit 10: System Timing Monitoring

System timing monitoring can be disabled by setting this bit to (1). This allows a user to continue running the system until they get assistance for solving the problem.

Bit 11: Jog in Auto Mode

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.

Any axis that is currently in single-axis or velocity mode can be jogged with the axis jog bits. The ELS Virtual Master can be jogged with the Task Jog Register coordinated Jog Forward and Jog Reverse bits.

Bit 12: Prioritized Service Channel

Only one task or user interface can use a drive's SERCOS service channel at a time. When drive parameter lists, long text strings, or oscilloscope data are transferred from one task. The other task could be suspended from 100ms to 5 seconds.

If the timing for user task service channel access is critical, this bit should be set to (1). The user tasks will suspend any SERCOS transmission of any text strings or lists from the user interface, and the communication error "!78 Service channel in use" will be issued.

If user task service channel access is not critical, parameter lists and oscilloscope are seldom used during normal operation, or nuisance "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 lists from user interface

Note: Even with prioritization, service channel access can vary between 10 and 100 ms. Therefore, any time-critical parameter transfers should be from the cyclic data if possible; or should be put into a non-critical section of the user program.

Bit 13: Ignore Drive Warnings

By default, VisualMotion issues the error "498 Drive D Shutdown Error" when the drives issue a Class 2 Diagnostic Warning. Sometimes it is necessary to disable this error to allow warning checks to be performed in the user program. These warning checks should be performed before the system is shut down, or to prevent nuisance faults from being issued during testing. This is a global option for all axes in the system.

- 0 = Drive warnings cause the control to issue a shutdown error
- 1 = The control ignores drive warnings

Bit 14: Ignore Axis Ready Status in Program Commands

In some applications, it is necessary to add and remove drives from the system by setting the disable bit in the axis control register. If this bit is (0), VisualMotion will issue an error if the drive is ready before any start motion commands. These commands include the start command, the homing command, and the operation mode switch. If this bit is (1), VisualMotion does not issue an error if the drive is not ready.

- 0 = Error is issued if axis is not ready
- 1 = No error is issued if axis is not ready

Bit 16: Disable AT Timing Check

An option to disable drive telegram time checking is necessary for older versions of DSS 2.1 SERCOS interface cards (DSS and DSI cards). When set to (0), the AT (Drive Telegram) time check is enabled. When it is (1), the AT time check is disabled. If SERCOS disconnect errors occur with a new firmware version that has not occurred with previous versions, the AT time check should be disabled or the DSS card should be upgraded.

- 0 = Check SERCOS AT timing (RECOMMENDED)
- 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 PPC-R. This selection takes effect at the next reset (from power-up or reset switch) of the control. 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 = ASCII Host Protocol and 2 = BTC06 Teach Pendant.

C-0-0012 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-only

C-0-0013 Serial Port A Mode

This parameter selects the communication mode for X10 on the PPC-R. This selection takes effect immediately. The baud rate and options for these ports are configured in parameter C-0-0003 Serial Port A Setup. The allowable values are 232 = RS232, 422 = RS422 and 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 PPC-R. This selection takes effect immediately. The baud rate and options for these ports are 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. A control shutdown error occurs if the communication error timer changes states from the enabled to the timed-out state. Only a *Fault Clear* can move the communication timer from the timed-out to the disabled state. If the communication timer is enabled, the timer is reset after each valid Reset Timer Command. The communication timer is reset by both a Timer Reset command and a change of state from disabled to enabled (i.e., Start Timer command).

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:	200
Access:	read / write in any Phase

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 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
Maximum value:	2000
Default value:	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 an 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 run mode, 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

Parameter C-0-0022 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

This parameter is used to initialize the handshaking between the control and the PLC. 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 any Phase 2

Jogging and Display Parameters (C-0-0042 through C-0-0056)

C-0-0042 World Large Increment

This parameter is used to set the incremental coordinated jogging distance that is used when Large Increment is selected. It is based on the value entered in task parameter T-0-0025 Maximum Jog Increment.

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 used to set the incremental coordinated jogging distance used when Small Increment is selected. It is based on the value entered in task parameter T-0-0025 Maximum Jog Increment.

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 used to set the coordinated jogging speed used when Fast Jog is selected. It is based on the value entered in task parameter T-0-0026 Maximum Jog Velocity.

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 used to set the coordinated jogging speed used when Slow Jog is selected. It is based on the value entered in task parameter T-0-0026 Maximum Jog Velocity.

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 used to set the incremental jogging distance used when Large Increment is selected. It is based on the value entered in axis parameter A-0-0025 Maximum Jog Increment.

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 used to set the incremental jogging distance used when Small Increment is selected. It is based on the value entered in axis parameter A-0-0025 Maximum Jog Increment.

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 used to set the axis jogging speed used when Fast Jog is selected. It is based on the value entered in axis parameter A-0-0026 Maximum Jog Velocity.

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 used to set the axis jogging speed used when Slow Jog is selected. It is based on the value entered in axis parameter A-0-0026 Maximum Jog Velocity.

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

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 will be used for user program downloads to the control. Refer to the section on [Direct ASCII Communication](#) for more information.

C-0-0090 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	200
Default value:	16
Access:	read / write in any Phase

C-0-0091 Total Program Memory

This is the total file memory on the control that can be used for programs and text messages. Total program memory includes control point, variable, I/O, and event 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:	0
Maximum value:	0
Default value:	total available program memory of installed firmware
Access:	read-only

C-0-0092 Available Program Memory

This is the amount of memory the control currently has available for storage of programs and text messages.

C-0-0092 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	bytes
Minimum value:	0
Maximum value:	0
Default value:	current available memory
Access:	read-only

C-0-0093 Contiguous Program Memory

VisualMotion dynamically allocates the non-volatile memory used for user programs and data. This is the largest unit of storage (i.e. the largest program or cam) that can be stored on the card. If this number is much smaller than C-0-0092 Available Program Memory, the memory is fragmented. If the message "Insufficient Program Space" is sent from the control, upload programs, delete them, then download them again to de-fragment the memory.

C-0-0093 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	bytes
Minimum value:	0
Maximum value:	0
Default value:	current amount available
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 issued.

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:	current amount available
Access:	read-only

C-0-0095 Path Planner to SERCOS Time Factor

This parameter allows the coordinated motion path planner to run less often (a multiple of the SERCOS rate) than the SERCOS cycle time to improve system performance.

Example: If the SERCOS rate is set to 4ms and this parameter is set to 5, the path planner would run every 5th SERCOS cycle or every 20ms.

This factor can reduce overhead due to path planner calculations by 120% or more. This allows user Tasks A-D to appear to run much faster.

If this factor is too large, it can result in a degradation of the coordinated path. Optimal settings for this parameter would be between 3 and 6.

C-0-0095 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

C-0-0098 Initialization Delay

This parameter causes the control to delay for the specified number of seconds before it initializes the SERCOS ring. This prevents the control from issuing a "No drives were found on ring" errors 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 any 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 coordinated motion using multiple tasks or cam motion using multiple cams, 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 layer and modify C-0-0099, all drive's SERCOS cycle time parameter A-0-0002 and NC cycle time parameter A-0-0001 to match. If the value in this parameter is greater than 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:	1000
Maximum value:	16000
Default value:	2000
Access:	read in any Phase / write in any Phase 2

Note: If larger SERCOS cycle times are used and communications errors occur, it might be necessary to 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 serial 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-08V09-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:	0
Maximum value:	0
Default value:	installed firmware version
Access:	read-only

C-0-0101 Control Hardware Version

This parameter displays the hardware version of the control.

Format:

PPC-R02.2N-N-NN-NN-NN-FW Rev: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:	0
Maximum value:	0
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:	0
Maximum value:	0
Default value:	installed firmware release date
Access:	read-only

C-0-0103 Maximum Number of Axes Allowed

This parameter is read-only and displays the maximum SERCOS drive address that can be set in any SERCOS device within the control's fiber optic ring. The allowable range is from 01 to 32. 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.

Note: This parameter's description was changed from "Maximum Number of Axes Allowed" to "Allowable Drive Address Range" for firmware version GPP08V27 or later. The parameter's functionality remains the same.

The maximum number of axes in a VisualMotion system is limited to 32. The value in parameter C-0-0103 indicates the maximum SERCOS drive address that can be assigned to any of the 32 axes in a system.

C-0-0103 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	01
Maximum value:	32
Default value:	32
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:	0
Maximum value:	0
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

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 phase if an error exists at the drive level.

C-0-0121 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed 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.

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:	0
Maximum value:	0
Default value:	Highest priority message is displayed
Access:	read-only

Refer to the *VisualMotion 8 GPP 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.

Example: 4 for 004 Emergency Stop

C-0-0123 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	Highest priority code is displayed
Access:	read-only

Refer to the *VisualMotion 8 GPP 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:	0
Maximum value:	0
Default value:	dependent on C-0-0122
Access:	read-only

Refer to the *VisualMotion 8 GPP 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 is a global system variable and can be read into an integer variable 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 (2³¹), 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-0-0123 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	milliseconds
Minimum value:	0
Maximum value:	0
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 flash identification and the diagnostic log. It also provides a time stamp for parameters when the parameter list is uploaded. 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 *Diagnostics* ⇒ *Diagnostic Log* or by directly modifying this parameter.

When using Direct ASCII Communication, the control requires 'SET' before the new date and time:

Format:

Month-Day-Year	Hour-Minute-Second in 24 hour format
06-22-2000	15:33:00

>SET 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:	0
Maximum value:	0
Default value:	01-01-1980 00:00:00 (at power up)
Access:	read-only

C-0-0127 Current PPC-R Temperature

This parameter displays the current internal temperature for the PPC-R hardware in degrees Celsius.

C-0-0123 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	degrees
Minimum value:	0
Maximum value:	0
Default value:	current operating temperature
Access:	read-only

Note: The maximum allowable internal temperature for the PPC-R when operating at a maximum ambient temperature of 45 °C is 70°C.

C-0-0142 Card Label String (GPP 8 only)

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-0-0124 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

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 A-0-0002(SERCOS Cycle Time).

Example: If C-0-0200 = 10% and A-0-0002 = 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
Maximum value:	100
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 A-0-0002 (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
Maximum value:	100
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 Options).

C-0-0202 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percent
Minimum value:	0
Maximum value:	100
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 Options).

C-0-0203 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	percent
Minimum value:	0
Maximum value:	100
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.

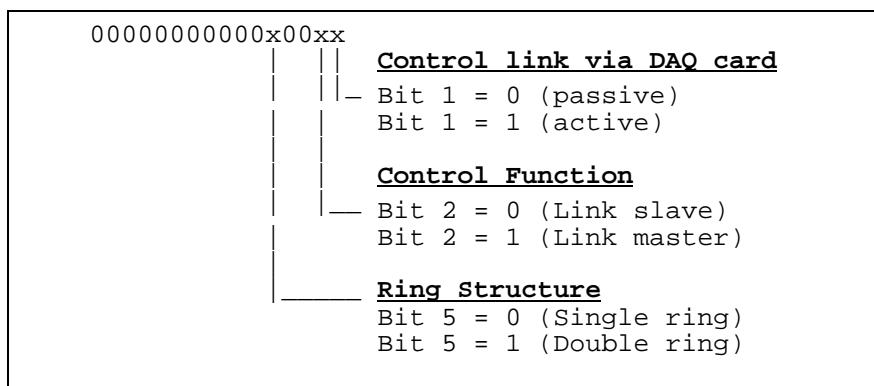


Fig. 6-3: Bit Description C-0-0300

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
Maximum value:	2000
Default value:	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
Maximum value:	2000
Default value:	0
Access:	read / write in any Phase

C-0-0303 Link Ring MDT Error Counter

Every link slave counts all illegal master data telegrams (MDT). If more than one MDT is sequentially skipped, then the control responds as follows:

- All drives are brought to a standstill "deceleration as best as possible" (P-0-0119) best possible standstill.
- All master axis positions remain.
- All outputs "link participant # data valid" (C-0-0002) are set to zero.
- Error message "541 Link Ring Error" is issued.

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. To reset the counter, write a 0 to this parameter.

C-0-0303 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read-only

Ethernet Parameters (C-0-0400 through C-0-0408)

C-0-0400 Card IP Address

This parameter contains a unique IP Address, specified in dot notation i.e., 172.16.11.200, assigned to the control's Ethernet card. An IP address is provided by the IT department. This parameter is initially configured in the DDE server under menu selection **Settings** ⇒ **Network Communications**.

C-0-0400 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read / write in any Phase

C-0-0401 Card Subnet Mask

This parameter contains a subnet mask used to determine what subnet the IP address in parameter C-0-0400 belongs to.

C-0-0401 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read / write in any Phase

C-0-0402 Card Gateway IP Address

This parameter contains the Gateway IP Address, specified in dot notation i.e., 172.16.1.1, assigned to the Ethernet card. A Gateway IP address identifies the router to which the IP Address is assigned. This parameter is initially configured in

C-0-0402 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read / write in any Phase

C-0-0403 Card CIF Network Control

This parameter specifies whether the control is connected to a half or full duplex switch. To change the mode, the user enters the following:

- HALF - Half-duplex mode
- FULL - Full-duplex mode

Note: The text is case sensitive and must be enter as all caps.

C-0-0403 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	last configured duplex mode
Access:	read / write in any Phase

C-0-0404 Card Access Network Control

This parameter displays the control's current network access level (if configured) for communication across an Ethernet connection. Once configured, the access level will revolve between No Access, Read and ReadWrite.

Note: The text is case sensitive and must be entered as follow:
No Access, Read, ReadWrite

Any of the following methods can be used to change the access level of an Ethernet-ready control.

Serial Connection:

1. Enter the desired access level (**No Access, Read or ReadWrite**) in control parameter C-0-0404.
2. Enter the network password (C-0-0405) directly in parameter C-0-0404 to revolve between the three access levels.
3. Enter the desired access level using the "*password.access level*" syntax.

Example: If the password is "gpp" (case sensitive). Then the following is allowed:

gpp.No Access
gpp.Read
gpp.ReadWrite

Ethernet Connection:

In order to change the access level across an Ethernet connection, the user must know the network password in control parameter C-0-0405.

1. Enter the network password (C-0-0405) directly in parameter C-0-0404 to revolve between the three access levels.
2. Enter the desired access level using the "*password.access level*" syntax.

Example: If the password is "gpp" (case sensitive). Then the following is allowed:

gpp.No Access
gpp.Read
gpp.ReadWrite

Access level	Description
not defined (default)	no access has been initialized.
No Access	the control can not be accessed via Ethernet.
Read	only read access has been allowed.
ReadWrite	the user has full access to all control functions.

Note: When communicating over a serial connection, parameter C-0-0404, *Card Access Network Control*, can be directly modified by entering the desired network access level (No Access, Read, ReadWrite).

When communicating over an Ethernet connection, the network access level is changed every time the password in C-0-0405, *Card Network Password*, is entered in C-0-0404.

C-0-0404 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	current network access level
Access:	read / write in any Phase

C-0-0405 Card Network Password

This parameter contains a password assigned by the primary user across a serial connection to the control. This parameter is used to change the access level of parameter C-0-0404. An alpha numeric password, up to a maximum of 80 characters, can be entered by the primary user to allow different levels of access (via the Ethernet card) to the control.

Note: When connected to the control across an Ethernet connection, the password in this parameter is displayed as asterisks. Only the user with access to the control serially can view the text.

C-0-0405 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	Network Password (initial text displayed serially)
Access:	read / write in any Phase

C-0-0406 CIF Ethernet Card Hardware ID

This parameter displays the current 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:	0
Maximum value:	0
Default value:	current Ethernet hardware typecode if installed
Access:	read-only

C-0-0407 CIF Ethernet Card Firmware Version

This parameter displays the current 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:	0
Maximum value:	0
Default value:	current Ethernet firmware version if installed
Access:	read-only

C-0-0408 CIF Driver Version

This parameter displays the current Ethernet card CIF 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:	0
Maximum value:	0
Default value:	current Ethernet CIF driver if installed
Access:	read-only

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.

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 implementation.

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).

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).

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:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	512
Default value:	1
Access:	read / write in any Phase

C-0-0806 Pendant End of User Accessible Registers

This parameter defines the end 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 next register to be displayed is the number stored in the End of User Accessible Registers parameter.

C-0-0806 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	512
Default value:	1
Access:	read / write in any Phase

C-0-0807 Pendant Password Timeout

This parameter sets a timeout on the BTC06 password. 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

Note: If the password is used, the pendant protection level bits (Register 1, bits 15 and 16) do not function.

C-0-0810 BTC06 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.

0000000000000000 _ 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. 6-4: 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. 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. Example: #####.##

Searching left to right, the first '#' sign found determines the field's location. 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 BTC06

This parameter allows the programmer to control or “force” which menus the user can access during a task. 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:	signed 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 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 BTC06

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.

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 BTC06

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.

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 ($\neq 0$) for parameter C-0-0813 to be functional.

C-0-0814 BTC06 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.

0000000000000000 _ Bit 1 Bits 1-9: Data index Bits 10-14: Menu ID number Bits 15-16: Control and Status
--

Fig. 6-5: 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 1-9: 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 10-14: Menu ID Number

The following table lists the available menu selections along with its index type.

Bits 10-14	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

Bits 10-14	Menu No.	Menu Name	Index Type
			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 15-16: Status and Control

These bits set the status and control for the BTC06 display.

Bit 15-16	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 ($\neq 0$) for parameter C-0-0814 to be functional.

Generic Cases The following table is a list of generic cases for this parameter.

Bits 1-9	Bits 10-14	Bits 15-16	Action
0	0	10	clears all locks in all menus
0	$\neq 0$	10	clears locks in a specific menu
0	0	11	locks all data indexes in all menus
$\neq 0$	$\neq 0$	11	locks data index in a specific menu
$\neq 0$	0	11	locks data index in the current menu
$\neq 0$	$\neq 0$	10	sets data index in a specific menu
$\neq 0$	0	10	sets data index in the current menu
0	0	01	requests data index from current menu the user is viewing
0	$\neq 0$	01	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" while the control is in parameter mode. The following screen will appear:



Note: Two points 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, re-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

System Memory Parameters (C-0-0994 and C-0-0996)

C-0-0994 Shutdown Command for Flash Programming

Reserved for future development

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 writing a "YES" to C-0-0996 the GPP application will be aborted and all program- and data memory will be cleared (clear the non-volatile AutoStore™ memory):

- Reset the NV_block
- Reset the diagnostic log
- Reset the IFS Structures (mfsd, fmad)

Note: The contents of the IFS FLASH buffers will be erased upon the next power cycle.

- Reset the active User Program's data.
- Reset all parameters (c_param, a_param, and t_param).
- Reset all Teach Pendant parameters.

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

Note: Depending on the current Phase, some parameters contained within the list may have write access.

C-0-2001 List of Required Parameters

This parameter contains a list of all required control parameters that are part of the current firmware version. These are all the parameters stored in nonvolatile control RAM that must be set for proper control operation. 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

Note: Depending on the current Phase, some parameters contained within the list may have write access.

C-0-2010 List of SERCOS Devices

During Communication 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 I/O stations (RMK modules for PPC-R). 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 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2011 List of SERCOS Drives

During Communication Phase 1 initialization, the control scans the SERCOS ring for all connected SERCOS digital drives. These digital drives include any Rexroth Indramat 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 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

C-0-2012 List of SERCOS I-O Stations

During Communication Phase 1 initialization, the control scans the SERCOS ring for all connected SERCOS I/O devices. These devices include all I/O stations (LUTZE for control or RMK modules for PPC-R) that are connected via the SERCOS ring. 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-0-2012 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

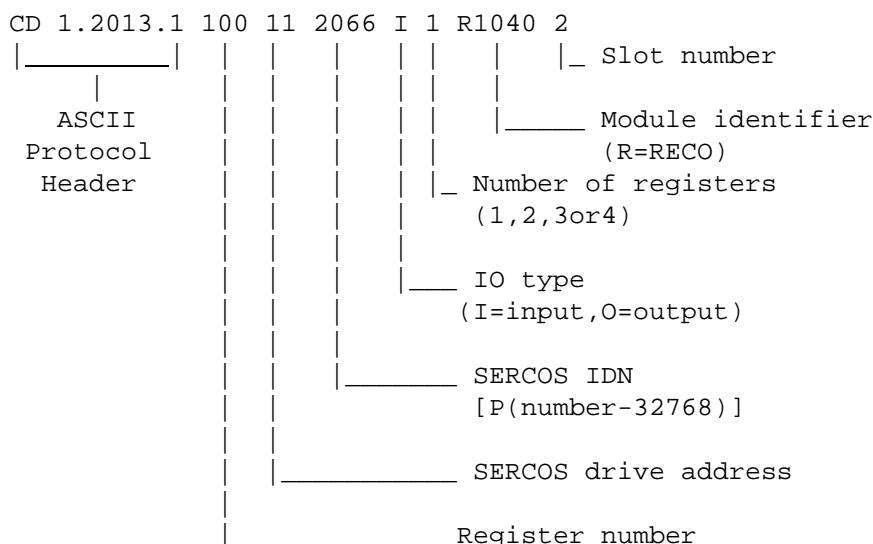
C-0-2013 I/O Configuration List

During Communication Phase 1 initialization, the control scans the SERCOS ring for any RECO I/O modules (Local and SERCOS). These devices include any I/O Stations (RMK modules for PPC-R) and Local RECO modules. 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-0-2013 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

The elements in the SERCOS I/O Configuration List are displayed using the following format:



C-0-2016 List of Virtual Axes

This parameter has no functionality in GPP firmware.

C-0-2016 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

C-0-2017 I/O User Configuration List

This parameter is used to store I/O devices configured using VisualMotion's GPP 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 by selecting ***File*** ⇒ ***Send I/O Configuration to PPC***.

C-0-2017 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 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. The list is retained in battery-backed RAM.

C-0-2020 Attributes

Data length:	variable length 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	current diagnostic log list
Access:	read-only

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
|   |   |   |   |   |   |_ 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-0-2021 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.

0000000000000000	_ Bit 1
Bit 1: Include E-Stop errors (400) Bit 2: Include warnings Bit 3: Include drive undervoltage errors (26) Bit 4: Include user warnings Bit 5: Include user errors	

Fig. 6-6: 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:	00000000000000100 (drive undervoltage error)
Access:	read / write in any Phase

Oscilloscope Parameters (C-0-2501 through C-0-2523)

The oscilloscope function is activated in VisualMotion Toolkit by selecting **Diagnostics** ⇒ **Oscilloscope**. Entering a value in this parameter is equivalent to selecting a signal type under *Signal Selection* in the oscilloscope window.

C-0-2501 Oscilloscope Signal 1 Type

This parameter sets the first signal type that will be displayed when capturing a signal using VisualMotion's oscilloscope function.

C-0-2501 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	8
Default value:	0
Access:	read / write in any Phase

The following grouping of parameters work in conjunction with each other for the signals listed in the table below.

- C-0-2501, C-0-2504 and C-0-2507
- C-0-2502, C-0-2505 and C-0-2508
- C-0-2504, C-0-2506 and C-0-2509

C-0-2501	C-0-2504	C-0-2507	Signal Description	Valid for	
				GPP7	GPP8
0	0	1	None	X	X
1	Program Float Number	1	Program Float Variable (Fx)	X	X
2	Program Integer Number	1	Program Integer Variable (Ix)	X	X
3	Global Float Number	1	Global Float Variable (GFx)	X	X
4	Global Integer Number	1	Global Integer Variable (GIx)	X	X
5	Axis Parameter Number (Refer to Note 1)	Axis Number (1-32)	Axis Parameter	X	X
6	Register Number (1-512)	Register Bit Number	Register Bit	X	X
7	Register Number (1-512)	1	+/- Register (used to monitor a register's value)	X	X
8	Control Parameter Number (Refer to Note 1)	1	Control Parameter	X	X
1	Programmed Float Variable G#_IN_POS	1	ELS Group # Input Position (Refer to Note 2)	X	X
1	Programmed Float Variable G#_IN_VEL	1	ELS Group # Input Velocity	X	X
1	Programmed Float Variable G#_OUT_POS	1	ELS Group # Out Position	X	X
1	Programmed Float Variable G#_OUT_VEL	1	ELS Group # Output Velocity	X	X

C-0-2501	C-0-2504	C-0-2507	Signal Description	Valid for	
				GPP7	GPP8
1	Programmed Float Variable G#_OUT_ACC	1	ELS Group # Acceleration	X	X
1	Programmed Float Variable VM1_CUR_POS	1	VM1 Position (Virtual Master signal)	X	X
1	Programmed Float Variable VM1_CUR_VEL	1	VM1 Velocity	X	X
1	Programmed Float Variable VM2_CUR_POS	1	VM2 Position	X	X
1	Programmed Float Variable VM2_CUR_VEL	1	VM2 Velocity	X	X

Note 1: Only a selected list of axis and control parameters can be accessed as a signal type when using the oscilloscope function. Refer to the tables for parameter C-0-2504 Oscilloscope Signal 1 ID Number for a listing of these parameters.

Note 2: The # symbol represents ELS Groups 1-8. VisualMotion automatically assigns this same signal to each configured ELS Group in the system.

C-0-2502 Oscilloscope Signal 2 Type

Refer to C-0-2501 Oscilloscope Signal 1 Type for a description of this signal.

C-0-2503 Oscilloscope Signal 3 Type

Refer to C-0-2501 Oscilloscope Signal 1 Type for a description of this signal.

C-0-2504 Oscilloscope Signal 1 ID Number

This parameter identifies the numeric value that corresponds to the selected signal type in parameter C-0-2501 Oscilloscope Signal 1 Type.

C-0-2504 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read / write in any Phase

When signal types 5 (Axis Parameter) or 8 (Control Parameter) are selected in C-0-2501, only the values in the following table are accessible as a valid number.

Valid Axis Parameters		Valid for	
C-0-2504	Description	GPP7	GPP8
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
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

Valid Control Parameters		Valid for	
C-0-2504	Description	GPP7	GPP8
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

C-0-2505 Oscilloscope Signal 2 ID Number

Refer to C-0-2504 Oscilloscope Signal 1 ID Number for a description of this signal.

C-0-2506 Oscilloscope Signal 3 ID Number

Refer to C-0-2504 Oscilloscope Signal 1 ID Number for a description of this signal.

C-0-2507 Oscilloscope Signal 1 Axis Number

This parameter is only applicable when C-0-2501, C-0-2502 or C-0-2503 is set with one of the following signal types:

- 5 (Axis number) – value equals SERCOS drive address for selected axis parameter in C-0-2504, C-0-2505 or C-0-2506. Allowable values are between 1-32.
- 6 (Register) – value equals bit number for selected register number in C-0-2504, C-0-2505 or C-0-2506. Allowable values are between 1-16.

C-0-2507 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	1
Maximum value:	32
Default value:	1
Access:	read / write in any Phase

Note: A value of 1 is defaulted in this parameter when a signal type other than 5 or 6 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 signal.

C-0-2509 Oscilloscope Signal 3 Axis Number

Refer to C-0-2507 Oscilloscope Signal 1 Axis Number for a description of this signal.

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.

Example: [(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:	--
Minimum value:	1
Maximum value:	64
Default value:	1
Access:	read / write in any Phase

C-0-2511 Oscilloscope Signal 1 List

This parameter stores all captured data from signal 1. This information is uploaded to the oscilloscope after the capture is complete.

C-0-2511 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

C-0-2512 Oscilloscope Signal 2 List

Refer to C-0-2511 Oscilloscope Signal 1 List for a description of this signal.

C-0-2513 Oscilloscope Signal 3 List

Refer to C-0-2511 Oscilloscope Signal 1 List for a description of this signal.

C-0-2514 Oscilloscope Sample Count

This parameter sets the sample count to determine how often the sample rate in C-0-2510 Oscilloscope Sampling Rate is captured. 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 50, 100, 200, 300, 400 and 500.

Example: [(C-0-2510 = 8ms) * (C-0-2514 = 500)] = 4000 ms

C-0-2514 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	10
Maximum value:	512
Default value:	500
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.

Example: $[(C-0-2510 = 500) * (\text{Pretrigger of } 10\%)] = 50$

$$C-0-2515 = (500-50) = 450$$

This means that the first 50 samples will be captured prior to the actual trigger mark followed by the remaining 450 samples. A trigger mark on the oscilloscope screen appears as a vertical line.

C-0-2515 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	512
Default value:	500
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:	8
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:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	0
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:

- 5 (Axis number) – value equals SERCOS drive address for selected axis parameter in C-0-2517. Allowable values are between 1-32.
- 6 (Register) – value equals bit number for selected register number in C-0-2517. Allowable values are between 1-16.

C-0-2518 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	1
Maximum value:	32
Default value:	1
Access:	read / write in any Phase

Note: A value of 1 is defaulted in this parameter when a signal type other than 5 or 6 is used.

C-0-2519 Oscilloscope Trigger Level or Mask

This parameter sets the level or threshold to which the oscilloscope will 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
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 is captured and stored to parameter C-0-2511 Oscilloscope Signal 1 List. This data is then uploaded to the oscilloscope window for display.

To initiate a capture manually, the state of bits 1-3 must read as follows:

Capture state: 00000000000000111

A low-to-high (0-1) transition of bit 3 allocates memory and configures all oscilloscope signals. A high-to-low (1-0) transition frees memory and disables the system.

0000000000000111 _ Bit 1 Bit 1: Start recording Bit 2: Trigger start Bit 3: Enable oscilloscope
--

Fig. 6-7: Bit Description C-0-2521

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 state of this parameter changes based on the progress during the capture process:

Capture in progress: 0000000000000111

Capture complete: 0000000000001101

0000000000001111 _ Bit 1
Bit 1: Trigger activated Bit 2: Data acquisition in progress Bit 3: System ready (memory allocated) Bit 4: Capture complete

Fig. 6-8: Bit Description C-0-2523

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 / write in any Phase

Fieldbus Interface Parameters (C-0-2600 through C-0-2701)

C-0-2600 Fieldbus Mapper (cyclic channel) To PLC

This parameter stores the fieldbus object-mapping list configured using the Fieldbus Mapper utility in VisualMotion. The data is a series of equations showing the relationship between fieldbus objects and VisualMotion data types. The fieldbus object-mapping list is transmitted cyclically from the fieldbus slave device to the PLC.

C-0-2600 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 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 Mapper (cyclic channel) From PLC

This parameter stores the fieldbus object-mapping list configured using the Fieldbus Mapper utility in GPP. The data is a series of equations showing the relationship between fieldbus objects and VisualMotion data types. The fieldbus object mapping list is transmitted cyclically from the PLC to the fieldbus slave device.

C-0-2601 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 configured objects
Access:	read in any Phase / write in Phase 2

Note: Data from this parameter is transmitted to the fieldbus slave device in the order in which it appears.

C-0-2630 Fieldbus Slave Device Address

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.:

- 1-125 for Profibus
- 1-63 for DeviceNet
- 1-99 for ControlNet

C-0-2630 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x0002
Maximum value:	0x0FF
Default value:	0x0063
Access:	read in any Phase / write in Phase 2

Note: Every Fieldbus slave device configured in a system must have a unique Fieldbus Slave Device Address.

C-0-2631 Fieldbus Parameter Channel Length

For Profibus systems using VisualMotion, 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. This parameter allocates the first 4 or 6 words of the Profibus cyclic DP channel as follows:

- 0x0000 = No parameter channel
- 0x0008 = allocates the first 4 words (8 bytes allocated)
- 0x000C = allocates the first 6 words (12 bytes allocated)

C-0-2631 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x0000
Maximum value:	0x000C
Default value:	0x0000
Access:	read in any Phase / write in Phase 2

C-0-2632 Fieldbus Multiplex Method

This parameter sets the primary or secondary multiplex method for Profibus fieldbus interfaces. The settings in this parameter vary based on the method configured in VisualMotion Toolkit's Fieldbus utility. Refer to the section on Profibus multiplexing methods in the *VisualMotion 8 Application manual*. The setting 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)

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 Error Reaction

You can choose how you would like the system to react in case of a Fieldbus error. This reaction can be set in the "Configure Slave Telegram" screen, using the combo box labeled "Fieldbus Error Reaction." Three options are available for the Error Reaction setting. Depending on the selected setting, the value 0, 1, or 2 is displayed in this parameter. The allowable error reaction are as follows:

- 0x0000 = shutdown
- 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 Word Swap (DeviceNet only)

In the Fieldbus Mapper, it is possible to enable automatic word and byte swapping for DeviceNet fieldbuses (for both input and output), depending on the type of PLC used. This parameter defines the word swapping options for *32-bit Objects* and *Explicit Message*. The allowable selections are:

- 0x0000 = no swapping
- 0x0001 = 32-bit object word swapping
- 0x0002 = Explicit message byte swapping
- 0x0003 = Both 32-bit and Explicit

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

- **32-bit Object Word Swapping** - 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. The default setting, "Do not swap words" ("Swap Words" checkbox unchecked under the Advanced Options) causes the words to be transmitted in their usual order: [Word 1], [Word 2]. The "Swap Words" setting ("Swap Words" checkbox checked under the Advanced Options) causes the words to be transmitted in inverted order: [Word 2], [Word 1]. The setting of this option is stored in Card Parameter C-0-2636, bit 0.
- **Explicit Message Byte Swapping** - The setting of this option determines the order in which the bytes of non-cyclic data >4 bytes long are transmitted. The default setting, "Do not swap bytes" ("Swap Bytes" checkbox unchecked under the Advanced Options) causes the bytes to be transmitted in their usual order: [Byte 1], [Byte 2], [Byte 3], [Byte 4], [Byte 5], [Byte 6].... The "Swap Bytes" setting ("Swap Bytes" checkbox checked under the Advanced Options) causes each pair of bytes to be transmitted in inverted order: [Byte 2], [Byte 1], [Byte 4], [Byte 3], [Byte 6], [Byte 5].... The setting of this option is stored in Card Parameter C-0-2636, bit 1.

C-0-2637 Fieldbus Slave Firmware Version

This parameter displays the current firmware version and release date of the installed and configured fieldbus interface. If no fieldbus interface is detected, this parameter contains no value.

Format:

Firmware version	Date released
V01.034	08.10.99

C-0-2637 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	firmware version of installed fieldbus interface
Access:	read-only

C-0-2638 Fieldbus 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 Mapper (cyclic channel) To PLC.

C-0-2638 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 supported parameters
Access:	read-only

C-0-2639 Fieldbus 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 Mapper (cyclic channel) From PLC.

C-0-2639 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 supported parameters
Access:	read-only

C-0-2641 Indramat/PLC Input Register List

This parameter contains the configured list of system registers that are placed in the Dual Port RAM memory for transfer to the PLC. The user configures this list in the *Parameter Overview* as follows:

1. Select On-Line 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.
5. Click on OK to return to the *Parameter Overview* window.

Note: Do not select any status registers. These register are read-only and no data will be transmitted across DPR memory.

C-0-2641 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 in any Phase / write in Phase 2

C-0-2642 Indramat/PLC Output Register List

This parameter contains the configured list of system registers that are placed in the Dual Port RAM memory for transfer to the PLC. The user configures this list in the *Parameter Overview* as follows:

1. Select On-Line 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.
5. Click on OK to return to the *Parameter Overview* window.

C-0-2642 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 supported parameters
Access:	read in any Phase / write in Phase 2

C-0-2700 Fieldbus Mapper (non-cyclic channel) To PLC

This parameter was originally intended to support non-cyclic mapping lists based on equivalent functionality found in GPS systems. However, due to the creation of pre-defined mapping objects for GPP, this parameter is now considered redundant. The Fieldbus Mapper tool in VisualMotion Toolkit does not support access to this parameter, and it is recommended to use Mapped Data in place of the non-cyclic mapping lists for GPP systems. Refer to "Mapped Data" in the non-cyclic channel section of the Fieldbus Mapper documentation.

C-0-2700 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 supported parameters
Access:	read in any Phase / write in Phase 2

C-0-2701 Fieldbus Mapper (non-cyclic channel) From PLC

This parameter was originally intended to support non-cyclic mapping lists based on equivalent functionality found in GPS systems. However, due to the creation of pre-defined mapping objects for GPP, this parameter is now considered redundant. The Fieldbus Mapper tool in VisualMotion Toolkit does not support access to this parameter, and it is recommended to use Mapped Data in place of the non-cyclic mapping lists for GPP systems. Refer to "Mapped Data" in the non-cyclic channel section of the Fieldbus Mapper documentation.

C-0-2701 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 supported parameters
Access:	read in any Phase / write in Phase 2

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 Card PLS's output modules. Two consecutive registers are used to display the status of the Card PLS outputs. The bits of the displayed register number represent outputs 1-16, respectively. The bits of the next register, represent 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:	512
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:	512
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 Card PLS parameters (for example, change settings for on / off positions). This parameter is used to issue a build command for the 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.

To start the build command, a binary 3 is written to this parameter. The response can be observed in parameter C-0-2904.

Note: This build command is not necessary if modifications to the 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. If the build 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-0-2904 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	0000000000000000
Maximum value:	0000000000000111
Default value:	0000000000000000
Access:	read-only

C-0-2905 PLS1 Activate Command

This parameter is used to activate the 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 is reserved for future development of Card PLS diagnostics.

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 is reserved for future development of Card PLS diagnostics.

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 Card PLS. The hardware ID is displayed as a hexadecimal value. The high byte represents the hardware ID for the Card PLS. The lower byte represents the hardware ID of the FPGA on the 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 firmware displays the firmware version on the Card PLS. Card PLS firmware is contained within GPP firmware. To update the Card PLS firmware to a later version requires that the PSM memory module be updated with current GPP firmware.

C-0-2910 Attributes

Data length:	variable length 1 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	1
Maximum value:	20
Default value:	0
Access:	read-only

C-0-2920 PLS1 Switch On List

This parameter displays a list of the On positions of all 96 Card PLS switches. The units for the value is based on the setting of axis parameter A-0-0005. The output of any switch whose On position matches it's respective Off position in C-0-2930 will not turn on.

C-0-2920 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 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 Card PLS switches. The units for the value is based on the setting of axis parameter A-0-0005.

C-0-2921 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 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 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).

C-0-2922 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 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 Card PLS outputs. The value displayed for each output represents the number of the master assigned to that output. A Card PLS can have up to 8 masters. Masters can be defined in the PLS tool as either an ELS Master, ELS Group or 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 Card PLS outputs. The value displayed for each output represents the lead time (μ s) assigned to that output. 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 μ s. The allowable range is 0 - 500000 μ s (in total 200 ms). When using the PLS tool, the user will be prompt when entering a value that is not in increments of 250 μ s.

Note: The output's lead time affects the On position of the assigned switches. A lead time turns on the output before reaching the actual On position. Example, if a switch's On position is set to 10 and the output has a lead time of 2, then the output will turn on when the switch reaches 8 and not 10.

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:	extended character set
Display format:	text
Units:	--
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 Card PLS outputs. The value displayed for each output represents the lag time (μs) assigned to that output. The assigned lag time is always active, regardless of any additional functions, such as a one shot. Lag times are written in increments of $250 \mu\text{s}$. The allowable range is $0 - 500000 \mu\text{s}$ (in total 200 ms). When using the PLS tool, the user will be prompt when entering a value that is not in increments of $250 \mu\text{s}$.

Note: The output's lag time affects the Off position of the assigned switches. A lag time turns off the output before reaching the actual Off position. Example, if a switch's Off position is set to 20 and the output has a lead time of 2, then the output will turn off when the switch reaches 18 and not 20.

Note: There are 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:	extended character set
Display format:	text
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 Card PLS outputs. The value displayed for each output represents the on time (μ s) assigned to that output, when configured as a one shot output. On times for one shots are written in increments of 250 μ s. The allowable range is 0-1000000 μ s (in total 1 s). When using the PLS tool, the user will be prompt when entering a value that is not in increments of 250 μ s.

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:	extended character set
Display format:	text
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 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. 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 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 / write in any Phase

C-0-2935 PLS1 Output Direction List

This parameter displays a list of 32 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. The allowable direction values within the list are:

- 0 - Positive
- 1 - Negative
- 2 – Positive and Negative

C-0-2935 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 value of 2
Access:	read / write in any Phase

C-0-2936 PLS1 Output Hysteresis List

This parameter displays a list of 32 Card PLS outputs. The value displayed for each output represents the hysteresis assigned to that output. Refer to for details.

C-0-2936 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 value of 2
Access:	read in any Phase / write in Phase 2

C-0-2940 PLS1 Master Type List

This parameter displays a list of 8 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)

C-0-2940 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 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 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)
- 1-32 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 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0 (value of data in list)
Maximum value:	32 (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 Card PLS masters. The value displayed for each master represents the encoder type for any real master(Drive) assigned in C-0-2940. The available encoder types for a master type of 3 assigned in C-0-2940 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 4 byte data
Data type:	extended character set
Display format:	text
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 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:	extended character set
Display format:	text
Units:	--
Minimum value:	0 (value of data in list)
Maximum value:	2 (value of data in list)
Default value:	0
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 configured using the I/O Mapper utility in VisualMotion Toolkit. At the start of the list, the number of Boolean strings is sent to/from the control. This program is handled as a standard parameter list. If a count of zero is sent, the mapping is deleted.

C-0-3000 Attributes

Data length:	variable length 4 byte data
Data type:	extended character set
Display format:	text
Units:	--
Minimum value:	0
Maximum value:	500
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.

0000000000000001	
	_ Bits 1
	Bits 1-2: Forcing options
	Bits 3-4: not used
	Bit 5: Scan time

Fig. 6-9: 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 is determined by the control's firmware version. The following table shows the allowable scan times based on control firmware versions.

Bit 5	Firmware Version	Description
0	GPP 7/8	I/O Mapper scanned every 2 ms.
1	GPP 7/8	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-0-3109)

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 4 bytes data
Data type:	extended character set
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-0-3108 CAM Table 8

Each parameter is a list that contains all the points of a built CAM table for CAM's 1-8. CAMs are built and transferred in VisualMotion Toolkit under menu select **VM Tool** ⇒ **Cam Builder**.

C-0-3101 Attributes

Data length:	variable length 4 bytes data
Data type:	extended character set
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-0-3109 CAM Table 9 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 9-40. CAMs are built and transfer in VisualMotion Toolkit under menu select **VM Tool** \Rightarrow **Cam Builder**.

C-0-3109 Attributes

Data length:	variable length 4 bytes data
Data type:	extended character set
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.

6.6 Task Parameters - Class T

Each task in VisualMotion has a set of parameters that selects options and displays status information. This following sections 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 TASK and PATH commands, and provides the type of task or coordinated motion. 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 = Normal coordinated motion

- 2 = Minimum-time coordinated motion
- 3 = Constant velocity coordinated motion

Note: Downloaded coordinated motion icon programs will always default to type 3 (Constant velocity).

T-0-0001 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	3
Default value:	0
Access:	read-only

T-0-0002 Task Options

Several options can be selected for User Tasks A-D. These options are selected via bits in this parameter for each task.

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. 6-10: Bit Description T-0-0002

Bit 1: Run Task during Errors

Some User Tasks are used for motion and others for communication or I/O. The motion tasks need to be shutdown during E-stop, drive, or axis errors to prevent damage or injury. The communication tasks, however, need to convey diagnostic information to an external device at all times. These tasks cannot be shutdown when an error occurs.

When bit 1 is set to (0), the task will be shutdown by any error, according to the other options in this parameter. When set to (1), the task will run during errors unless a task error is issued in this task. System errors, drive errors, or task errors from other tasks do not stop the task or cause it to execute the error handler.



CAUTION

This option should not be selected for a task that has axes associated with it, since the drives will not be shut down during system errors such as E-stop. Option bit 5 in this parameter is a safer method of ignoring drive errors from other tasks.

Bit 2: Task Errors Do Not Shutdown Other Tasks

This option allows errors that occur during the operation of a task to be ignored by other tasks. This option can be used to make motion tasks independent of a communication task.

When bit 2 is set to (0), task errors issued from this task will shutdown all other user tasks. When set to (1), all other tasks will continue running during errors issued from this task. Parameter transfer errors and other non-fatal errors will not stop the motion in the other tasks. Drive errors from drives associated with this task are not ignored. To ignore drive errors, it is necessary to enable the option "Ignore task and drive errors from other tasks" (bit 5) for each task.

Bit 3: Parameter Transfer Errors are Warnings

This option allows the current task to continue running if a parameter transfer error occurs. The message "*205 Parameter Transfer Warning: see Task A diag.*" is issued. If a parameter value is critical to the operation of the task, the task error bit can be tested after the parameter transfer, or this option should be disabled.

Bit 4: Automatically Start Tasks

This option automatically starts tasks and keeps them running. All task control 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. The bits in the control register are enabled as long as bit 2 is high (1), and are ignored when the bit is low (0). This allows the automatic start option to be disabled in case the task needs to be stopped or the debugging bits need to be used.

Bit 5: Ignore Task and Drive Errors from Other Tasks

When this option is enabled, errors related to this task's program logic or motion will not affect any other tasks. This option can be used to allow each task to run an independent subsystem of a machine.

When bit 5 is set to (0), all task errors and drive errors will shutdown this task. The task stops running, and motion is stopped according to the severity of the error, and the control and drive error reaction modes.

When set to (1), this task will continue running during task errors issued by other tasks and drive errors from drives associated with other tasks. This task shutdown for system errors such as E-Stop or SERCOS disconnect, task errors issued by this task and drive errors from drives associated with this task.

Types of Errors

System Errors include errors that affect the entire VisualMotion system. "400 Emergency Stop" is a system error that shutdown all tasks and all drives. "409 SERCOS Disconnect Error" prevents communication to all drives. Other system errors include internal control operating system errors.

Task errors include any error that was issued as the result of a user program instruction. These errors include parameter transfer errors, errors in the Calc instruction, bound checking, or motion limit errors related to the instruction, etc. Task errors always set the Task Error bit in the task status register.

Drive errors are issued by the drive, and the control displays the error "420 Drive D Shutdown Error." Drive errors set the Class 1 Shutdown bit in the axis status register, but does not set the Task Error bit.

T-0-0002 Attributes

Data length:	2 byte data
Data type:	binary number
Display format:	binary
Units:	--
Minimum value:	00000000000000000000
Maximum value:	--
Default value:	00000000000000000000
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 coordinated motion position, speed, and acceleration data. No unit conversions are performed when changing this parameter. The following values 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:	12
Default value:	0
Access:	read-only

T-0-0011 Coordinated X Axis

This parameter provides the axis number (SERCOS address) corresponding to the X-axis of this task (selected in the user program TASK/AXIS command). If the number is zero, no coordinated axes are assigned to this task.

T-0-0011 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	8
Default value:	0
Access:	read-only

T-0-0012 Coordinated Y Axis

This parameter provides the axis number (SERCOS address) corresponding to the Y-axis of this task (selected in the user program TASK/AXIS command). If the number is zero, no coordinated axes are assigned to this task.

T-0-0012 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	8
Default value:	0
Access:	read-only

T-0-0013 Coordinated Z Axis

This parameter provides the axis number (SERCOS address) corresponding to the Z-axis of this task (selected in the user program TASK/AXIS command). If the number is zero, no coordinated axes are assigned to this task.

T-0-0013 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	8
Default value:	0
Access:	read-only

T-0-0020 Maximum Path Speed

This parameter sets the maximum speed allowed for this task's coordinated motion. The speed entries in the control's Absolute and Relative point tables are percentages of this value.

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
Access:	read / write in any Phase

T-0-0021 Maximum Acceleration

This parameter sets the maximum acceleration allowed for this task's coordinated motion. The acceleration entries in the control's Absolute and Relative point tables are percentages of this value.

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 this task's coordinated motion. The deceleration entries in the control's Absolute and Relative point tables are percentages of this value.

T-0-0022 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-0-0023 Look Ahead Distance

This parameter sets the minimum look ahead distance that the control's path planner uses to calculate a path. The difference between the current target position and the actual position is called the look ahead distance.

T-0-0023 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches or millimeters
Minimum value:	0.0
Maximum value:	10000.0
Default value:	10.0
Access:	read / write in any Phase

The look ahead distance should never be set to zero. 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.

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 intermediate positions the path planner will calculate ahead of the currently commanded position. Of course, if the parameter is set too large, the path planner may process many statements before physical motion takes place. The path planner can't identify potential real-time events that may stop motion or require a program branch. This can result in wasted calculations and control resources.

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 and a lighter load on control resources. However, motion geometry segments using blending may be missed if the look ahead distance value is set too small.

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:	inches or millimeters
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 Fast and Slow 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:	inches/min or mm/min
Minimum value:	0.0001
Maximum value:	100000.0
Default value:	100.0
Access:	read / write in any Phase

T-0-0027 Maximum Path Jerk

This parameter defines the maximum path jerk used for coordinated jogging. The Fast and Slow 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:	inches/sec ³ or mm/sec ³
Minimum value:	0.01
Maximum value:	1e+07
Default value:	20000
Access:	read / write in any Phase

Robotics (T-0-0035 through T-0-0059)

T-0-0035 Relative Point Used for Origin

This is the relative point in which the robot origin is stored. The point index in the relative table is stored here at the execution of the robot/origin instruction so that the origin remains the same when jogging in manual mode. It can also be edited directly in parameter mode so that the origin can be set up before the execution of the instruction.

T-0-0035 Attributes

Data length:	2 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	0
Default value:	0
Access:	read / write in any Phase

Note: This parameter takes effect at execution of the robot/origin instruction and at exit from parameter mode.

T-0-0036 Relative Point Used for Tool Frame

This is the relative point in which the robot tool frame is stored. The point index in the relative table is stored here at the execution of robot/tool instruction so that the origin remains the same when jogging in manual mode. It can also be edited directly in parameter mode so that the tool frame can be set up before the execution of the instruction.

T-0-0036 Attributes

Data length:	
Data type:	2 byte data
Display format:	integer
Units:	signed decimal
Minimum value:	--
Maximum value:	0
Default value:	0
Access:	0
	read / write in any Phase

Note: This parameter takes effect at execution of the robot/origin instruction and at exit from parameter mode.

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.

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:	10.0
Default value:	set by the control
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 shows 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.



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 shows 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, 1 and 7 that are transitional or do not apply to the current segment. Use code 8 to check if the target position has been reached. 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	Segment in queue for path planner
1	Acceleration	Acceleration in progress
2	Slew (constant speed)	Slew 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)
7	At Target	Segment at target position
8	Done	Motion on the segment is complete

T-0-0101 Attributes

Data length:	4 byte data
Data type:	integer
Display format:	signed decimal
Units:	--
Minimum value:	0
Maximum value:	8
Default value:	dynamically updated by control
Access:	read-only

T-0-0111 Current X Position

This parameter displays the current commanded position of the X-axis in world coordinates.

T-0-0111 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches or millimeters
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

T-0-0112 Current Y Position

This parameter displays the current commanded position of the Y-axis in world coordinates.

T-0-0112 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches or millimeters
Minimum value:	--
Maximum value:	--
Default value:	--
Access:	read-only

T-0-0113 Current Z Position

This parameter returns the current commanded position of the Z-axis in world coordinates.

T-0-0013 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches or millimeters
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 values 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 and/or code. During normal operation, a Message/Diag statement in the user program sets this message. If an error occurs during task execution, this diagnostic message is overwritten with an error message. The following strings are available:

- 001-199: Status Messages
- 201-399: Warning Messages
- 400-599: Shutdown Messages

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 Message/Status command 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 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:	--
Default value:	read by control
Access:	read-only

T-0-0131 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.

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

As a future enhancement, the control can possibly execute an error routine if an error in a task occurs. This status parameter displays the instruction pointer where the error occurred.

T-0-0132 Attributes

Data length:	4 byte data
Data type:	unsigned integer
Display format:	hexadecimal
Units:	--
Minimum value:	0x00000000
Maximum value:	--
Default value:	read by control
Access:	read-only

T-0-0133 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.

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

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

The following is a main task example:

1 0020 ; one pointer in main task executing at relative memory address 0020.

The following is a subroutine example:

2 0020 0034 ; two pointer in a subroutine executing at relative memory address 0020 0034.

The following is an event example:

-4 0020 0034 1205 1344 ; 4 pointer in an event executing at relative memory address 0020 0034 1205 1344.

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. 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.

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, the string "NONE" is returned.

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. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

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 current firmware version. These are all the parameters stored in nonvolatile control RAM that must be set for proper control operation. The contents of this parameter can be viewed by double clicking on the parameter number in the *Parameter Overview* tool.

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

6.7 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-0-0001through A-0-0038)

A-0-0001 Task Assignment

This parameter associates an axis with a user task. The TASK/AXES command and the Axis Setup Icon automatically set the parameter when a program is activated. The following assignments 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 displays the selected SERCOS drive control mode. For coordinated, ELS phase, Cam, and single-axis motion, it switches between normal and lagless positioning modes. The following positioning types 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 sets the type of motion for an axis and enables the axis/drive address on the SERCOS ring. The applicable SERCOS parameters are automatically sent to the drive. If a user program is present on the card, this parameter is automatically set when exiting from parameter mode, based on the commands that set up ELS, coordinated or single-axis motion. The following motion types are available:

- 0 = Disabled
- 1 = Single Axis
- 2 = Coordinated Axis
- 3 = Velocity Mode
- 4 = Ratio Slave
- 5 = ELS Slave
- 6 = Torque Mode
- 7 = Control Cam Axis
- 8 = Torque Following Mode
- 9 = Coordinated Articulation

Single Axis runs the axis as a single, non-coordinated axis. The control sends only the position to the digital drive. The digital drive generates and controls the position motion profile.

Coordinated Axis is used for coordinated circular or linear motion profiles generated by the control's path planner. The axes used for each path are determined by user program statements.

Velocity Mode 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.

Ratio Slave designates the axis as a slave to the master axis designated in Parameter A-0-0030 Ratio Mode Master Axis. The axis' drive operates in position or velocity mode, with its commanded signal equal to the product of the master axis signal times a ratio. The drive remains in position or velocity and at the commanded signal at all times, even when the user task is not running.

ELS Slave designates the axis as an Electronic Line Shafting (ELS) slave.

Torque Mode Axis: The axis runs in torque modes, with torque commands sent from the control and no velocity or position loop.

Control Cam Axis: The axis runs from a cam table stored on the control and uses the ELS virtual or Real Master.

Torque Following Mode: The axis runs with added torque following mode.

Coordinated Articulation: runs the axis as a control cam with coordinate transformation.

A-0-0003 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	8
Default value:	0
Access:	read-only

A-0-0004 Axis Options

This parameters sets all the necessary options for a Rexroth Indramat digital drive.

0000000000000001	_ Bit 1
	Bit 1: Position initialization Bit 2: Positioning mode Bit 3&4: ELS synchronization mode Bit 5: Not used 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. 6-11: Bit Description A-0-0004

Bit 1: Position Initialization

0 = Reset feedback in Phase 2

1 = Keep feedback value in Phase 2

Default: (1)

When a single-turn absolute or incremental encoder is used with Rexroth Indramat's digital drives; the drive resets the encoder to a one-turn measurement each time it exits SERCOS Phase 2. The digital drive uses the parameter "Starting Position Value" to override the default and initialize the feedback to the specified value.

Reset Feedback in Phase 2

When set to (1), the control writes the drive's starting position value to the Position Feedback, and the drive's position remains the same after the system is switch out of parameter mode.

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.

Note: This parameter does not affect a multi-turn absolute encoder feedback. To set the multi-turn encoder's position, follow the procedure described in the drive's manual, using the "Set Absolute Position Measurement" and homing commands.

Bit 2: Positioning Mode

0 = Linear Positioning Mode

1 = Rotary Positioning Mode

Default: (0)

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 PARAMETER/INIT.

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 3 and 4: 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 3	Bit 4	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.
1	0	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: Reserved for future use**Bit 6: Optional Cyclic Data**

0 = Use smallest cyclic data configuration

1 = Include optional cyclic data (velocity feedback, programmed acceleration)

Default: (0)

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 8: Drive PLS Fast Write (DKC2.3 only)

0 = Drive PLS Fast Write disabled

1 = Drive PLS Fast Write enabled

Default: (0)

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

1 = Secondary mode is velocity mode

Default: (0)

For an ELS axis, this bit selects the secondary, non-synchronized mode for a digital drive. This is the default mode until the ELS/MODE command 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 ELS/MODE command, 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

1 = Shortest path disabled

Default: (0)

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)

1 = Use secondary feedback (encoder 2)

Default: (0)

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

1 = Coast to a stop

Default: (0)

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 SERCOS control word (D-1.00134).

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

1 = Linear axis uses modulo positioning

Default: (0)

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

1 = Minimum cyclic data

Default: (0)

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.

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 = Modulo positioning

1 = Absolute positioning

Default: (0)

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.

Absolute Positioning

When set to (0) (default), the rotary axis position resets to the modulo value for every revolution.

Bit 16: Disable Automatic Scaling

0 = Control sets SERCOS scaling parameters (RECOMMENDED)

1 = Control does not set SERCOS scaling parameters

Default: (0)

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 Indramat 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
- 2 = degrees

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

The drive's display and scaling parameters are automatically set by the control. All data is transmitted in floating point format with the same resolution as the data transmitted by the drive. Velocity, acceleration, and jerk data are always transmitted with three decimal place accuracy. 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

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 explicitly save each required drive mechanical parameter before changing the Linear Position Units parameter.

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 2-16: Reserved for future development

Fig. 6-12: 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 Encoder procedure is used to set the reference position. This option should only be set when an absolute encoder is used, because it prevents an incremental homing procedure from being used.

Bits 2-16: Reserved for Future Use

Other options will be added in the future.

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.

When set to (0), the axis can be used in the program with its defined axis type and its presence on the ring will be verified.

When set to (1), the drive is excluded from the program, but can be jogged using default values.

When set to (2), the drive is not configured and is put into a torque-free mode after initialization.

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 selects the register to which the drive-based Programmable Limit Switch (PLS) will be associated. 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.

If the register is associated with a DEA card, the drive internally updates the outputs every 2ms. If the register is not assigned to a DEA, this register will be updated with the PLS status bits read from the drive every SERCOS cycle.

Note: *SERCOS configuration:* The PLS Status IDN P-0-0135 will be placed in the Amplifier Telegram in the cyclic data, with 2 bytes allocated. Parameter P-0-0124 (Assign IDN->DEA) is automatically set by the control if this register is a DEA output register.

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 the configured axis. The axis velocity is limited by this value during a coordinated move. For single-axis or velocity mode motion, this is the maximum velocity that can be programmed.

A-0-0020 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches/min, mm/min or RPM
Minimum value:	0.001
Maximum value:	1e+07
Default value:	1000.0
Access:	read / write in any Phase

Note: The Bipolar Velocity limit parameter (S-0-0091) on the drive is the maximum velocity allowed in a system.
Parameter A-0-0020 should always be set to a value less than or equal to S-0-0091.

A-0-0021 Maximum Acceleration

This parameter sets the maximum programmable acceleration for the configured axis. The axis acceleration is limited by this value during a coordinated move. For single-axis or velocity mode motion, this is the maximum acceleration that can be programmed in VisualMotion.

A-0-0021 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches/sec ² , mm/sec ² or rad/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 the configured axis. The axis deceleration is limited by this value during a coordinated move. For single-axis or velocity mode motion, this is the maximum deceleration that can be programmed in VisualMotion.

A-0-0022 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches/sec ² , mm/sec ² or rad/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 manual 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 based on this value.

A-0-0025 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 setting
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 multiplied by this value to calculate jog velocity.

A-0-0026 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	inches/min , mm/min or RPM
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 sets the axis that is used as the master for ratio, ELS, or Cam functions. A slave axis to an ELS Virtual Master is set to 0. If a Real Master is used, it is set to the axis on which the encoder resides. For the ratio function, the master axis is set independently for each slave at program compile time. The velocity feedback of the master is multiplied by the ratio established by parameters A-0-0031 Control Cam/Ratio Master Factor (N) and A-0-0032 Control Cam/Ratio Slave Factor (M).

A-0-0030 Attributes

Data length:	2 byte data
Data type:	unsigned integer
Display format:	unsigned decimal
Units:	--
Minimum value:	0
Maximum value:	32
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 motion 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.

$$\text{Scmd} = \text{Cam} + L * \text{Mcmd}$$

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 option sets the type of master used for Ratio Mode. The primary encoder or secondary encoder can be used. Options (0) and (1) select the master drive's primary feedback, which is the value read from drive parameter S-0-0051. Option (2) selects the secondary feedback, read cyclically from drive parameter S-0-0053 (feedback 2). The following modes are available:

- 0 = Not used or primary feedback is used
- 1 = Primary feedback is used
- 2 = Secondary feedback is used

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.

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. The available master types are...

- Real Master (Primary or Secondary Feedback)
- Virtual Master 1

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 send to the ratioed slave axis every SERCOS cycle.

0000000000000001
_ Bit 1
Bit 1: Control mode (Default = 0)
Bit 2: Feedback mode (Default = 0)

Fig. 6-13: 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:

- **Bit 1 = 0:** Sets the ratioed slave axis in Position Mode (Default)
- **Bit 1 = 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:

- **Bit 2 = 0:** The master's position is evaluated by VisualMotion (Default)
- **Bit 2 = 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* command 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 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* command 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:	inches/min, mm/min or RPM
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 coordinated and velocity modes. 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:	inches/min, mm/min or RPM
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:	inches/min, mm/min or RPM
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 service channel.

A-0-0120 Programmed Acceleration

The value in this parameter is the programmed acceleration used by the drive in single-axis mode. Programmed accelerations are generated every time an ACCEL command 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:	inches/sec ² , mm/sec ² or rad/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 VisualMotion program. The values for this parameter are written to the SERCOS Master Control Word (S-0-0134). It defines important control functions, like...

- Drive enable
- Drive halt
- Selected mode of operation

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. It defines important drive functions, like...

- Class 1 errors, drive lock
- Operation readiness
- Actual active operation mode

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 error counter can be used for troubleshooting of SERCOS connections. If the value of this parameter is increasing while it is being displayed, there may be a noisy SERCOS connection or a faulty communication card on the drive associated with this axis.

If two consecutive ATs are invalid, the control issues a SERCOS disconnect error.

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

This is the cyclic equivalent of parameter S-0-0182 on Rexroth Indramat's digital drives. It is always included in the cyclic data in single-axis mode. In ELS mode, it is included only when phase adjust is executed on the drive. The control sets the axis status register based on some of these bits.

00000000000000000001	_ Bit 1
	Bit 1: Drive lock active (AS)
	Bit 2: Axis in motion
	Bit 3: Drive Ready
	Bit 4: Warning
	Bit 7: In target position (additional)
	Bit 8: Amplifier at 90 % load
	Bit 9: In Synchronization
	Bit 10: Synchronization ended
	Bit 11: In target position
	Bit 12: Drive halt active
	Bit 13: Endposition reached

Fig. 6-14: Bit Description A-0-0140

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.

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.

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

In a VisualMotion program, the mode change icon can be used to change the mode of operation at any time. This parameter displays the current motion type corresponding to the drive's operating mode. 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
- 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:	6
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, corresponding to drive parameter P-0-0083. It is adjusted from -100 to 300 percent using the ELS/ADJUST command. If fine adult 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 ELS_Adjust or Cam_Adjust command.

A-0-0151 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	A-0-0005 settings
Minimum value:	--
Maximum value:	--
Default value:	0.0
Access:	read / write in any Phase

Note: When dynamic synchronization is enabled on the drive, parameters A-0-0151 through A-0-0163 are not used.

A-0-0153 Control Phase Adjust Average Velocity

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. When A-0-0153 is used, it sets the average velocity of the phase offset move. Note that the 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:	inches/min, mm/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 move. 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 that the 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-0-0157 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 settings
Minimum value:	--
Maximum value:	--
Default value:	read from control
Access:	read / write in any Phase

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.

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 CAM/ADJUST command.

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 an 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.

```

0000000000000001
|_ Bit 1

Bit 1: Set phase offset profile
Bit 2: Remove "Phase Offset" 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 Cam control/status
Bit 6: Do not automatically set phase offset when switching
      into synchronization

```

Fig. 6-15: 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:(0)

On Rexroth Indramat drives, the phase offset for ELS and drive-based cams can be performed on the drive. Enabling the phase offset on the drive frees control resources.

Note: The control filter and other related parameters and bits are disabled with this option.
Enabling the phase offset on the drive automatically places parameter S-0-0182 into the cyclic data so that the phase adjusted bit in the axis status register can be updated with bit 8 of this parameter.

When the phase offset is performed on the drive, the 'phase adjusted' status in the axis status register depends on parameter S-0-0228 (position synchronization window).

Bit 2: Remove phase offset from cyclic data

- (0) = Phase offset is in cyclic data
 (1) = Phase offset is through service channel
 Default: (0)

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 user program command will take up to 50ms to execute. If the control is generating the phase offset profile, the value will change instantly.

Bit 5: Use service channel for cam control/status

- (0) = Cam control and status uses drive real time bit
 (1) = Cam control and status is through service channel
 Default: (0)

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 to (1).

Bit 6: Do not automatically set phase offset

- (0) = Sets phase offset automatically at synchronization and stores value in A-0-0157
 (1) = Phase offset can be initialized with programmed offset (A-0-0151 Programmed Phase Offset) before synchronization.
 Default: (0)

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.

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. Rexroth Indramat 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. 6-16: 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 ($0 \Rightarrow 1$) 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 SERCOS telegram, this parameter allows the user to add an additional drive parameter from S-0-0188 to the MDT by entering the parameter IDN number.

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 telegram and A-0-0190 is used to update the value for A-0-0180 in real-time.

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 parameter transfer command 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 SERCOS telegram, this parameter allows the user to add an additional drive parameter from S-0-0187 to the AT by entering the parameter IDN number.

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 telegram and A-0-0195 is used to monitor the value for A-0-0185 in real-time.

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 parameter transfer command 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 parameter A-0-0180 Optional Command ID #1. Changes to this parameter will affect the value of the drive parameter set in A-0-0180.

A-0-0190 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
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 parameter A-0-0185 Optional Feedback ID #1.

A-0-0195 Attributes

Data length:	4 byte data
Data type:	float number
Display format:	signed number with exponent
Units:	--
Minimum value:	--
Maximum value:	--
Default value:	0
Access:	read / write in any Phase

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 SERCOS telegrams 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 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 4 byte data
Data type:	extended character set
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 4 byte data
Data type:	extended character set
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 4 byte data
Data type:	extended character set
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 current firmware version. These are all the parameters stored in nonvolatile control RAM that must be set for proper control operation. 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

7 VisualMotion Toolkit Menu Commands

7.1 Introduction

VisualMotion Toolkit 8 menus are structured to provide a logical grouping of commands. This chapter is a reference for these menu commands. Fig. 7-2 illustrates the new menu structure found in VisualMotion 8.

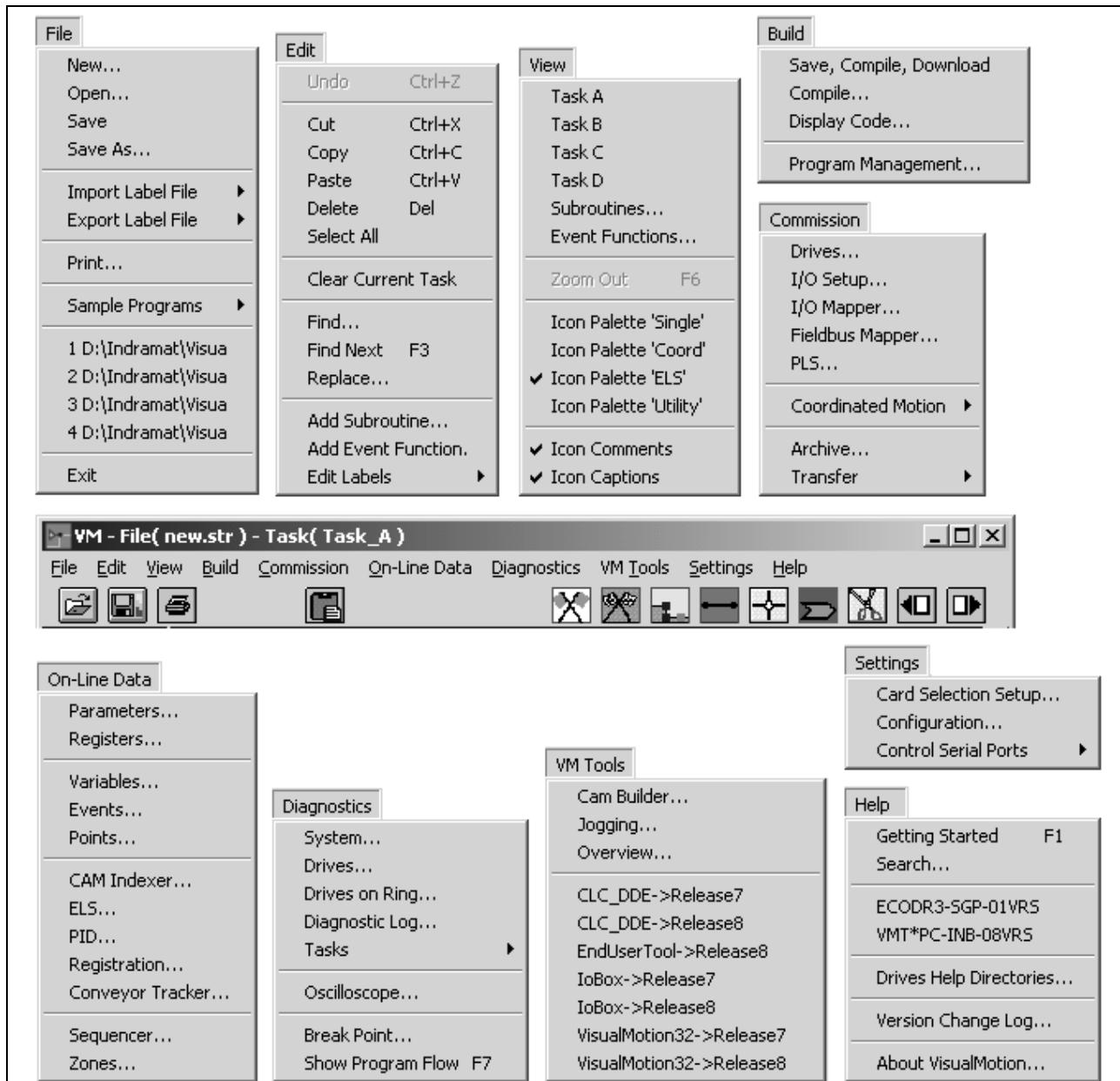


Fig. 7-1: VisualMotion 8 Menu Structure

Some of the frequently used menu commands also have shortcut buttons represented in the Icon Button Bar.

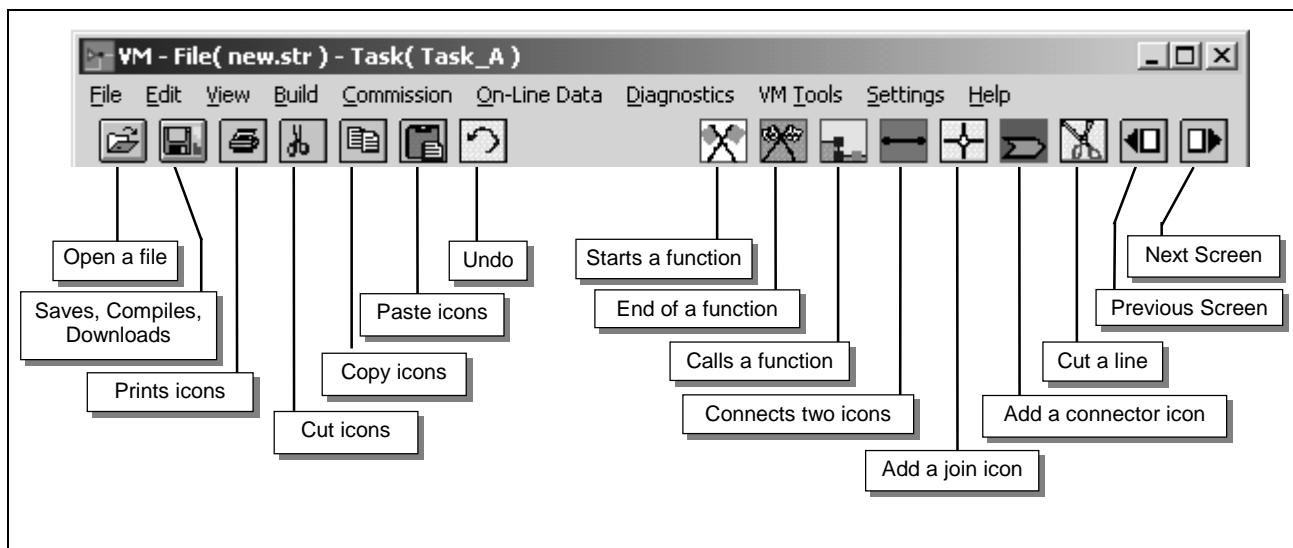


Fig. 7-2: VisualMotion Program Menu and Icon Button Bar

7.2 The File Menu

The **File** Menu contains commands for standard Windows functions, import and exporting of program labels, printing VisualMotion programs/elements (variables, etc.) and accessing sample programs.

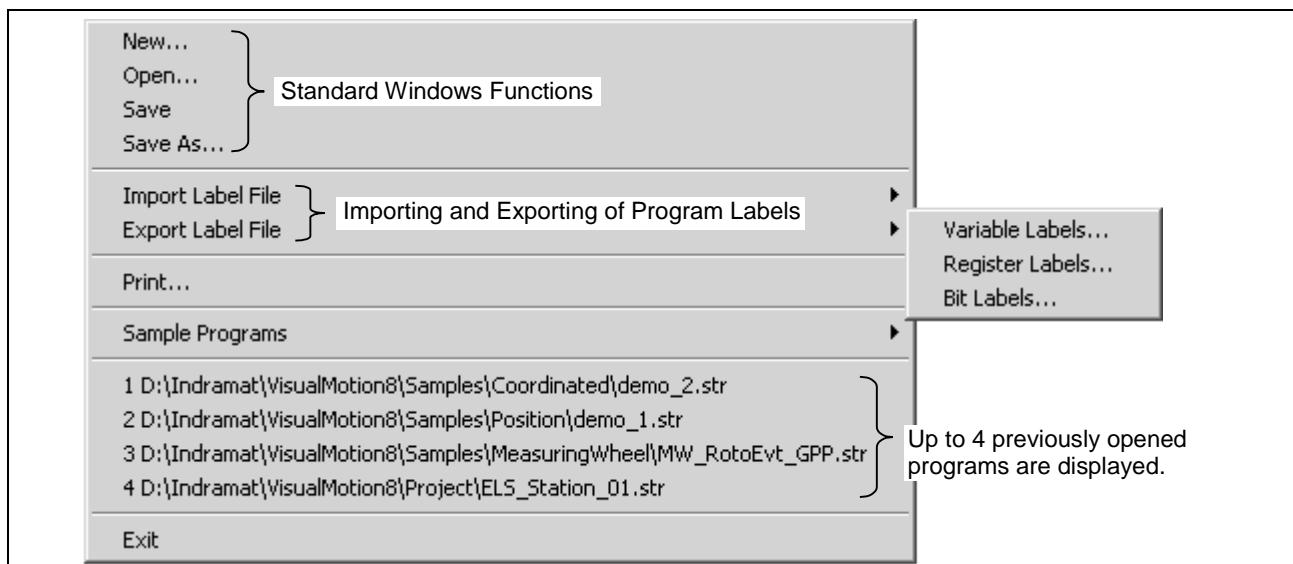


Fig. 7-3: The File Menu

Standard Windows Functions

New...

This menu command opens a new Icons or Textual language program using GPP 7 or GPP8 as the target firmware.



Fig. 7-4: New VM Program Development Window

Open...

This menu command opens an existing icons or textual language program.

Save

This menu command saves a named icon program to a file, prompting for a filename the very first time.

Save As...

This menu command saves an icon program to a file, always prompting for a filename.

Exit

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

Importing and Exporting of Program Labels

Import Label File

Selecting **File** ⇒ **Import Label File** and one of the three options (**Variable Labels**, **Register Labels**, or **Bit Labels**) opens an **Import Label File** Window, which permits you to import a previously saved label file into the current program.

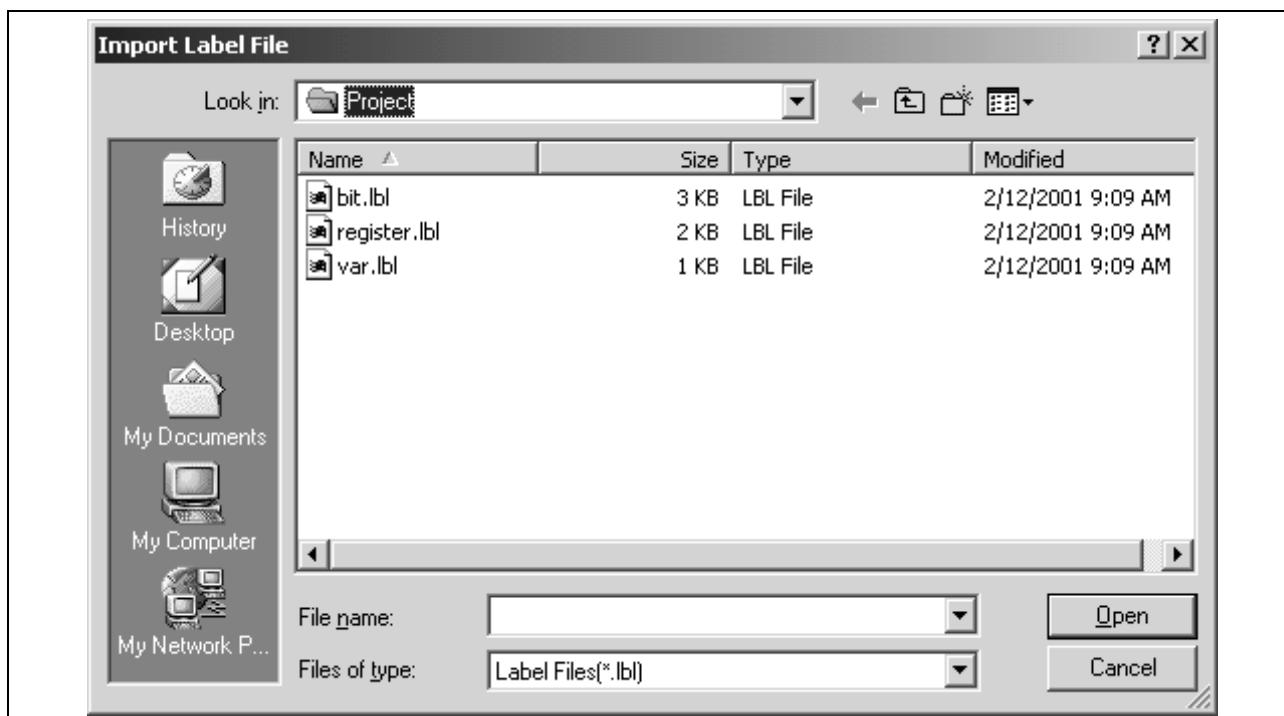


Fig. 7-5: "Import Label File" Window

Importing a label file overwrites all existing labels in the program. User-defined label files are stored by default in the *VisualMotion8\project* folder with a *.lbl file extension.

Export Label File

Selecting *File* ⇒ *Export Label File* and one of the three options (**Variable Labels**, **Register Labels**, or **Bit Labels**) opens a **Save As** window. Depending on the option selected, the corresponding filename is listed by default.

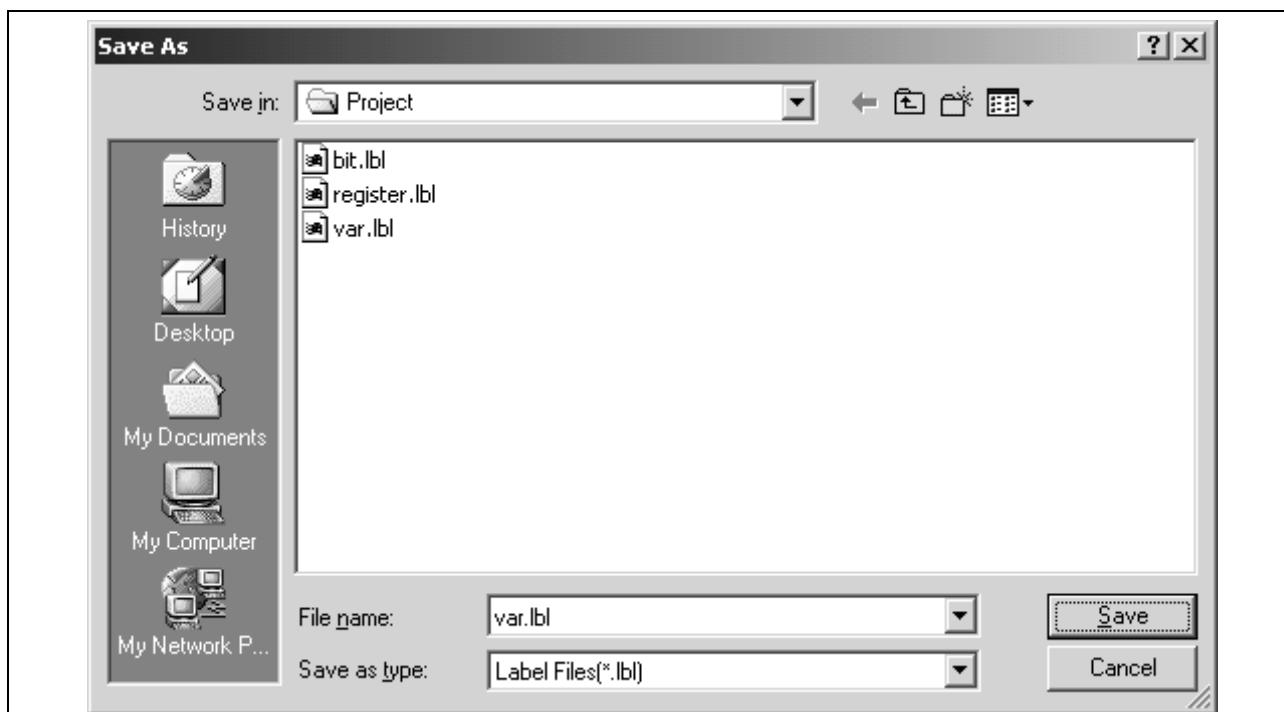


Fig. 7-6: "Save As" Window for Exporting Labels

Saved label files may be subsequently imported into another programs allowing you to maintain standardized user labels for various categories of programs. User-defined label files are saved by default in the VisualMotion8\project directory with a *.lbl file extension.

Printing Program Data



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

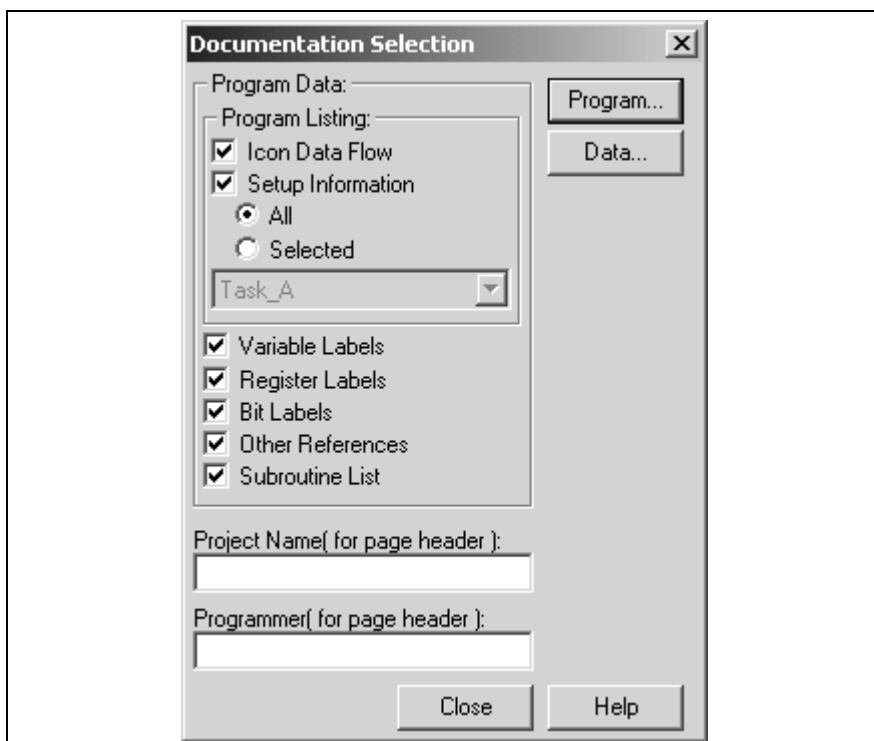


Fig. 7-7: "Documentation Selection" Window

Program Listing

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 **User Variables**, **Register Variables**, and/or **Bit Variables** 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.

Accessing Sample Programs

Sample Programs

As a part of VisualMotion Toolkit, three sample programs (For the applications "Measuring Wheel GPP", "Position Mode", and "Coordinated Mode") have been included.

7.3 The Edit Menu

The **Edit** menu contains Windows editing features for icon-based programs, clearing the icon workspace, find/replace functions, add subroutine and event functions, and edit functions for variable, register, register bits and I/O function bit labels.

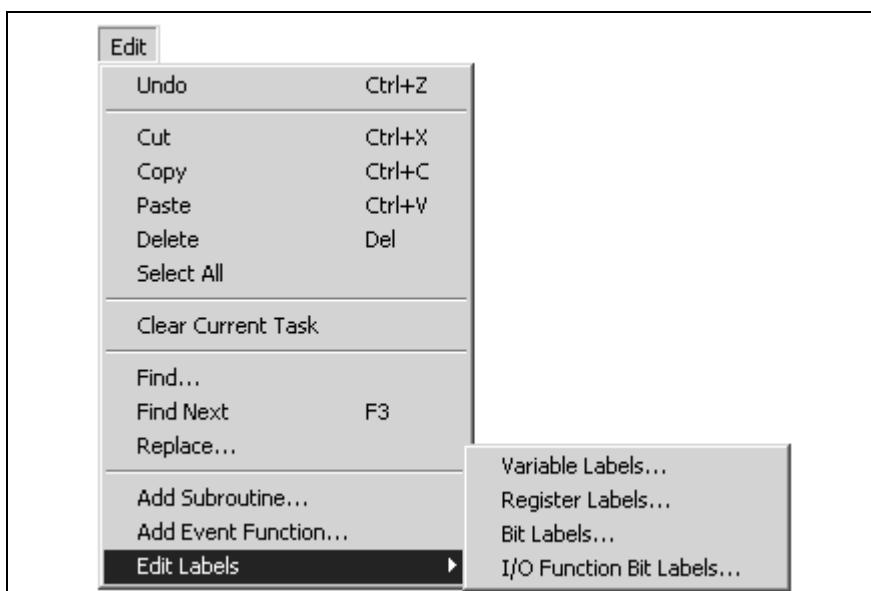


Fig. 7-8: The Edit Menu

Windows Editing Features

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

Undo (Ctrl+Z)

This menu command undoes 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.

Clearing the Icon Workspace (Clear Current Task)

This menu selection deletes all contents of the current VisualMotion task, subroutine, or event workspace of the open program.

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:

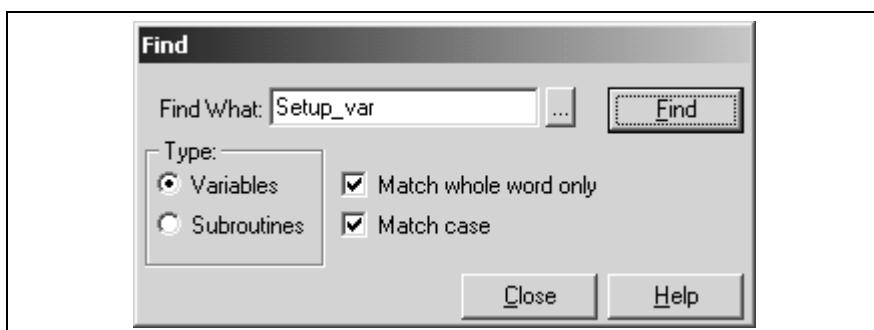


Fig. 7-9: "Find" Window

1. Select **Variables** or **Subroutines** in the **Type**: grouping.
2. Enter the text in the **Find What:** box. 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.

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:

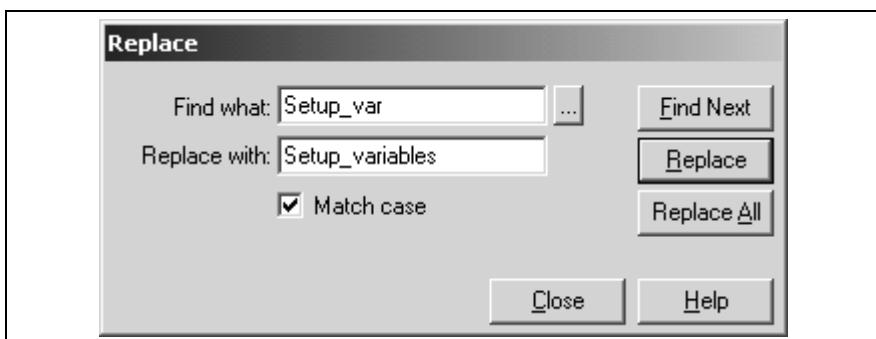


Fig. 7-10: "Replace" Window

Add Subroutine

Selecting **Edit** ⇒ **Add Subroutine** opens a Subroutine Control Block window. This menu item must be used when creating a Sequencer subroutine. For other subroutine types, the Subroutine Icon is also available (refer also to [Subroutine Icon](#) in the chapter, "Icon Programming").

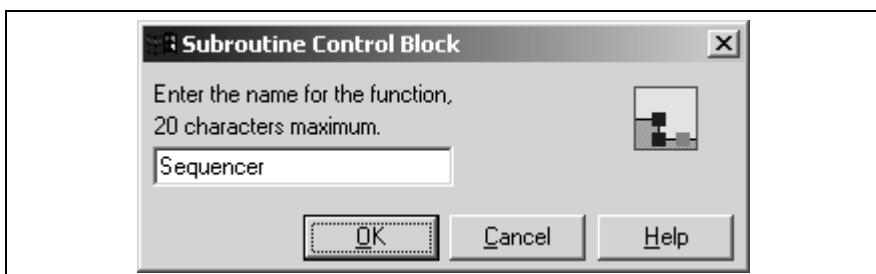


Fig. 7-11: "Subroutine Control Block" Window

Enter the name of the desired subroutine in the field. The name must begin with an alpha character, and no spaces are permitted. Pressing the **OK** button opens a subroutine workspace with the name entered and the **Start** and **Finish** icons 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.

Note: A maximum of 200 screens, consisting of tasks, subroutines, and event functions are allowed.

Add Event Function

Selecting **Edit** ⇒ **Add Event Function** opens an Event Function Control Block window.

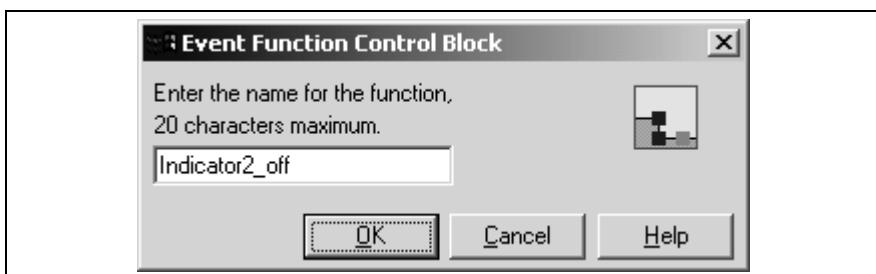


Fig. 7-12: "Event Function Control Block" Window

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 also to [Calc Icon](#) and [Event Icon](#) in the chapter, "Icon Programming").

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

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 200 screens, consisting of tasks, subroutines, and event functions are allowed.

Edit Labels

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 its 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. Following is a list of the VisualMotion keywords:

Current VisualMotion Keywords				
AA_XFER	CAM_STATUS	END	MOVER_CIRCLE	SEQ_STEP
ABORT	CALL	EVENT	MOVER_PATH	SEQUENCER
ABORT_PATH	CALL_FUNC	EVENT_DONE	MSG_DIAG	SET_PARAM
ABS_INIT	CAP_ENABLE	EVENT_ENABLE	MSG_STATUS	SET
ACCEL	CAP_SETUP	EVENT_END	PARAM_BIT	START
AXES	CLEAR	EVENT_START	PARAM_INIT	STOP
AXES_GROUP	CNC_CIRCLE	EVENT_WAIT	PATH_CALC	STOP_PATH
AXIS_ATPOSITION	COMP	EVT_INIT	PATH_HOLD	Task_A
AXIS_EVENT	CONVEYOR_INIT	FUNC_ARG	PID_CONFIG	Task_B
AXIS_WAIT	CONVEYOR_LIST	FUNC_END	PLS_WRITE	Task_C
BNE	CONVEYOR_PICK	FUNC_START	PLS_INIT	Task_D

Current VisualMotion Keywords				
BEQ	DATA_SIZE	GET_PARAM	PLS1_INIT	TEST
BGT	DEC	GO	POSITION	V_MASTER
BLT	DECEL	HOME	RA_XFER	VAR_INIT
BGE	DRVCOM	INC	RATIO	VEL
BLE	DRVCOM_STATUS	INIT	READ	WAIT
BRA	EE_XFER	KINEMATIC	REGISTRATION	WAIT_IO
BROADCAST	ELS_ADJUST	LOCAL_VAR	REL_INIT	WAIT_PATH
CALC	ELS_ADJUST1	MESSAGE	RESUME_PATH	WRITE
CAM_ADJUST	ELS_GROUPM	MODE	RETURN	ZONE_INIT
CAM_ACTIVATE	ELS_GROUPS	MOVE_JOINT	ROBOT_ORIGIN	ZZ_XFER
CAM_BUILD	ELS_INIT	MOVEA_AXIS	ROBOT_TOOL	
CAM_BUILD1	ELS_MODE	MOVEA_CIRCLE	ROTARY_EVENT	
CAM_ENGAGE	ELS_MASTER	MOVEA_PATH	RR_XFER	
CAM_INDEX	ELS_STOP	MOVER_AXIS	SEQ_LIST	

Table 7-1: Current VisualMotion Keywords

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.

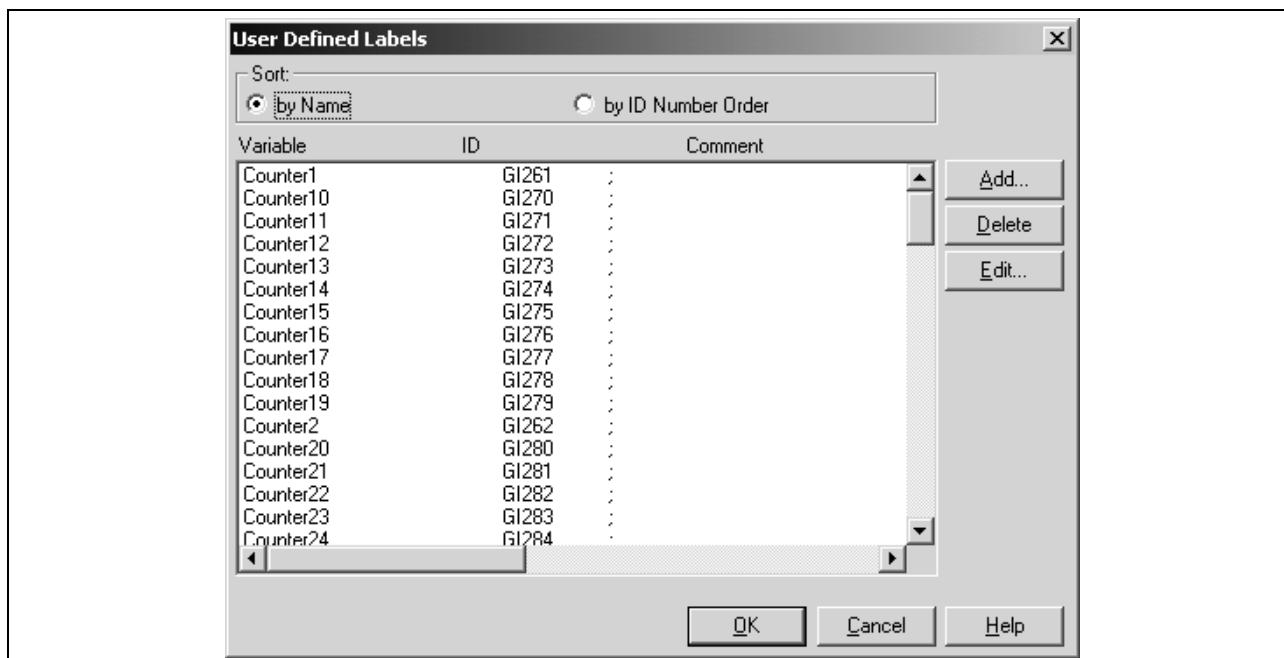


Fig. 7-13: "User Defined Labels" Window

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.

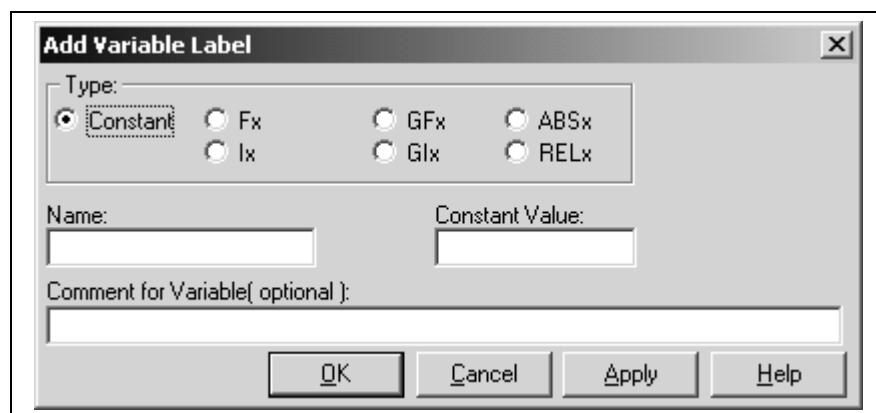


Fig. 7-14: "Add Variable Label" Window

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:

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.

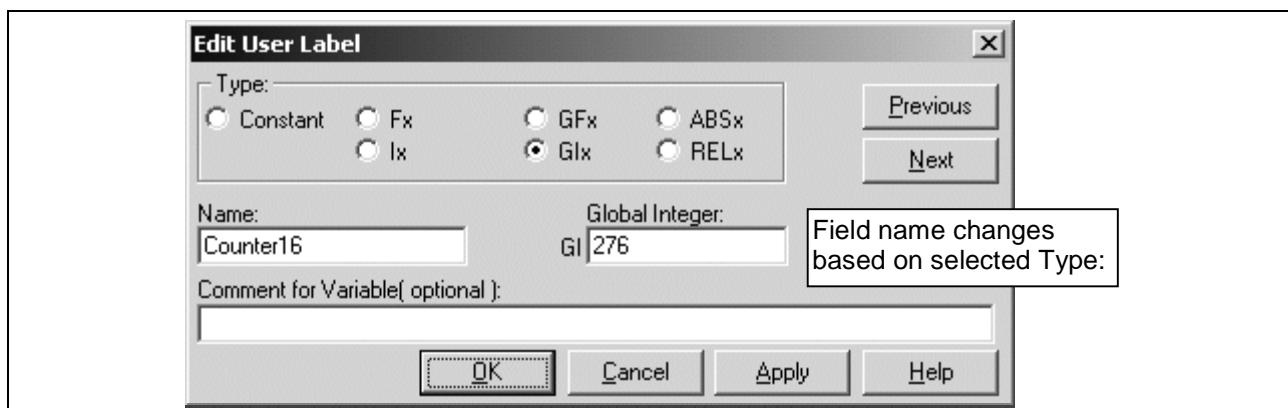


Fig. 7-15: "Edit User Label" Window

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.

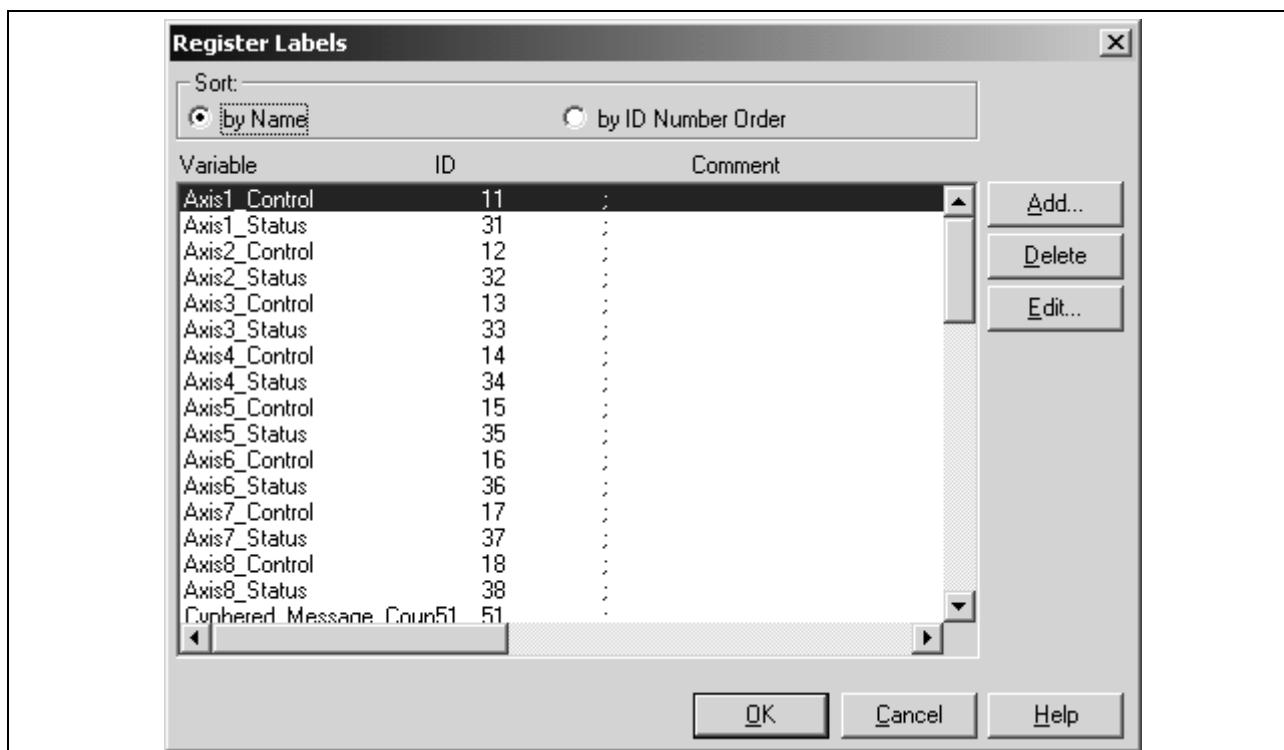


Fig. 7-16: "Register Labels" Window

The first 87 registers are reserved for VisualMotion system use and are assigned names for system, task, drive, and axis use. These registers may not be modified. Registers 88 and above may have user-assigned names. If drivers for a VisualMotion-supported I/O system are installed, the associated I/O register names are listed, typically beginning at register 100. 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 the section entitled **Input/Output Systems** for the 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.)

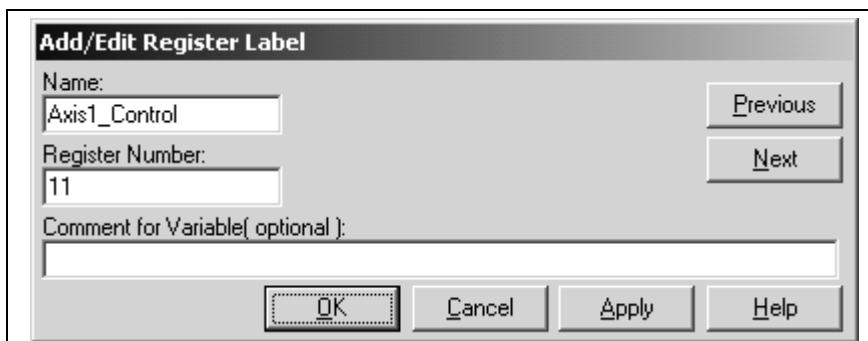


Fig. 7-17: "Add/Edit Register Label" Window

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.

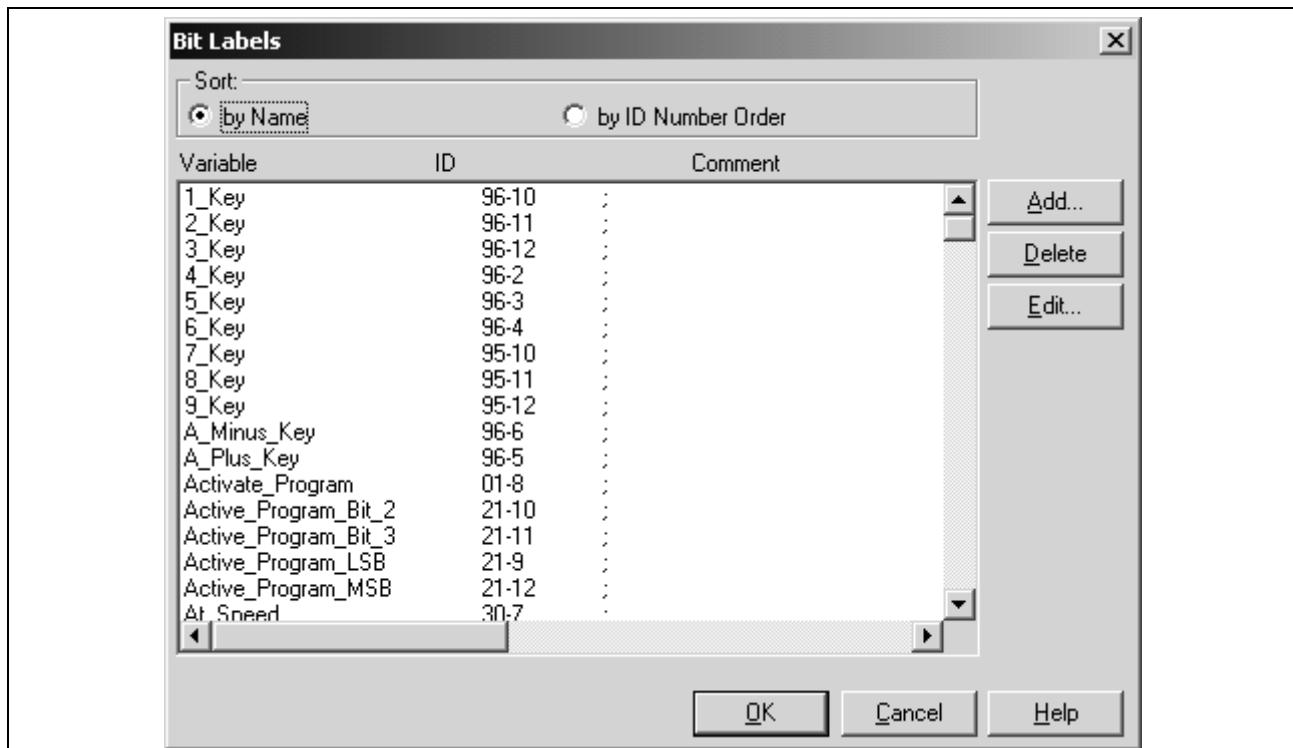


Fig. 7-18: "Bit Labels" Window

The bit labels for the first 39 registers may not be modified. 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. See [Input/Output Systems](#) for 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.)

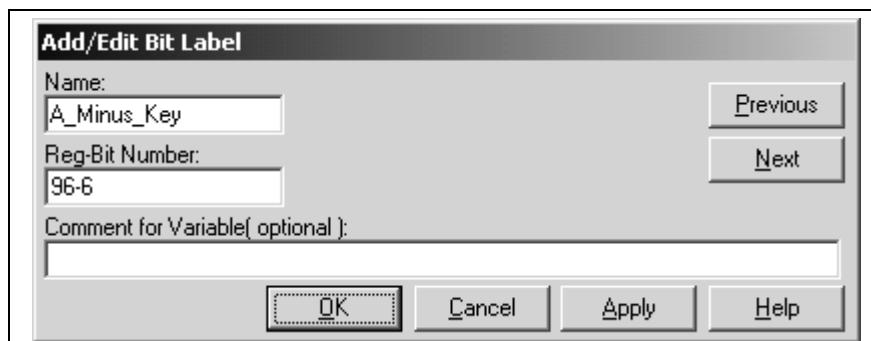


Fig. 7-19: "Add/Edit Bit Label" Window

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.

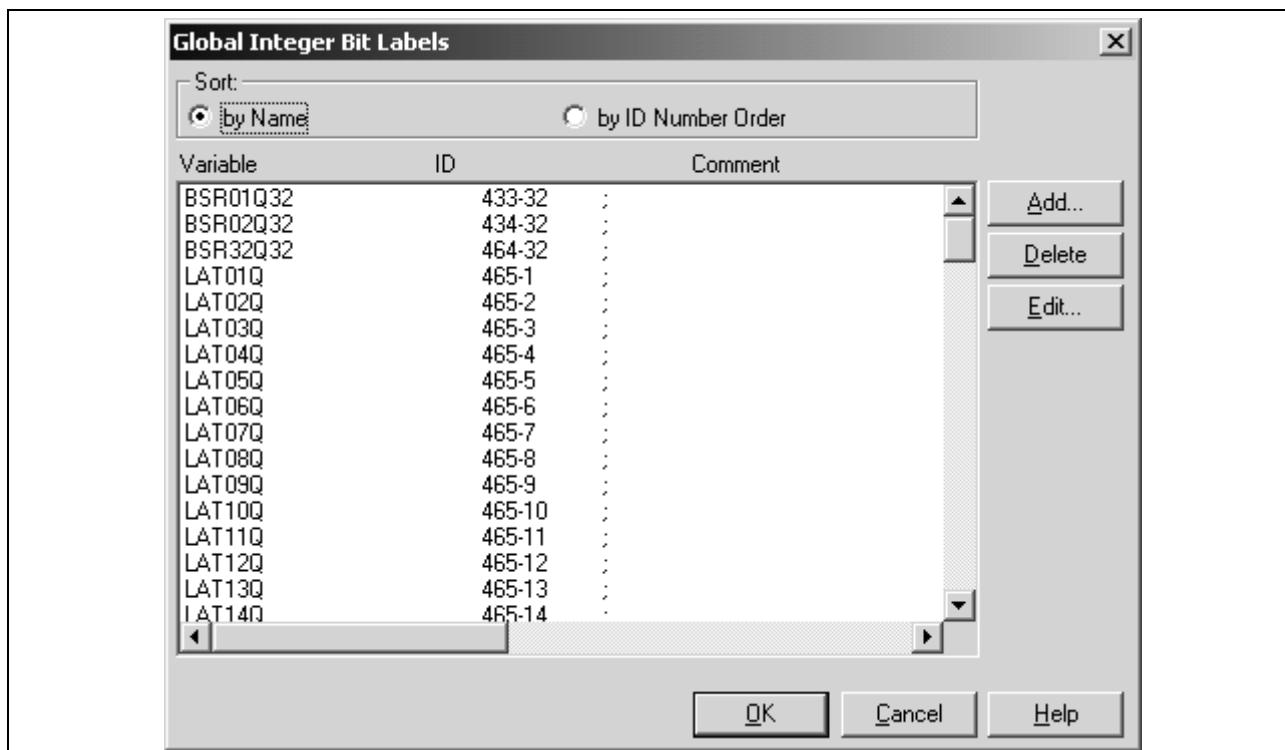


Fig. 7-20: "Global Integer Bit Labels" Window

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. See [Input/Output Systems](#) for 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.)

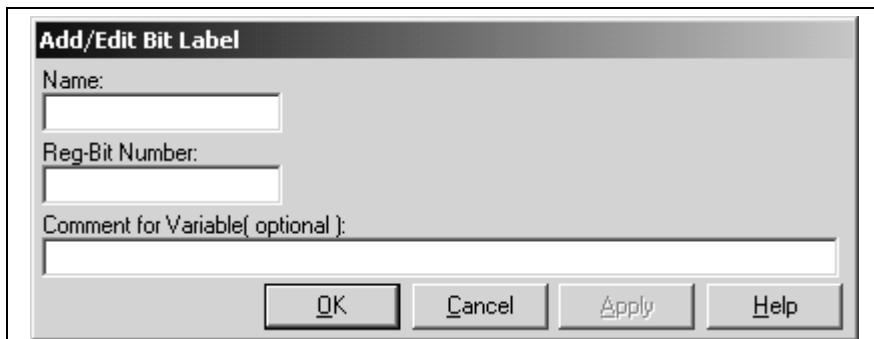


Fig. 7-21: "Add/Edit Bit Label" Window

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

7.4 The View Menu

The **View** menu allows the user to select a task (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 a utility palette.

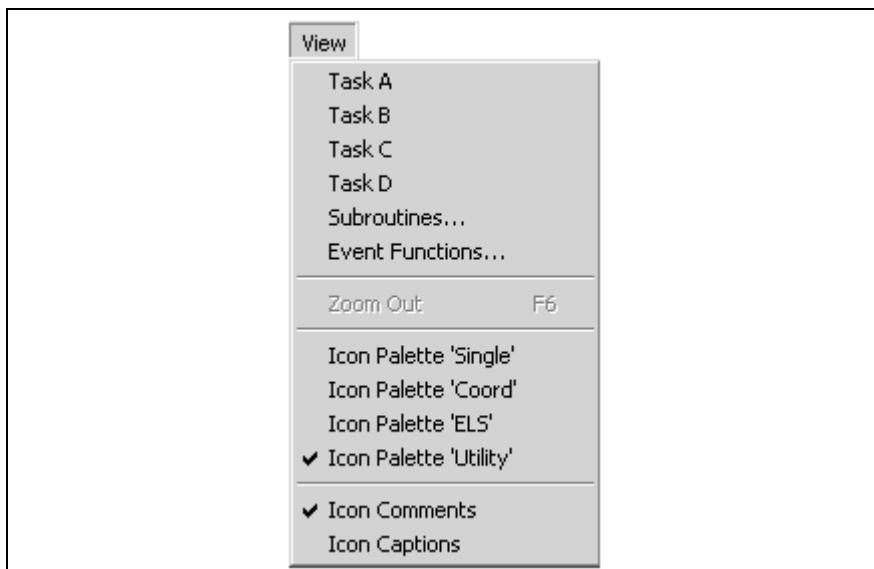


Fig. 7-22: The View Menu

Choosing one of the menu selections displays the selected program portion in the VisualMotion workspace, replacing the current workspace contents.

Tasks A, B, C, and D

Selecting **View** ⇒ **Task [x]** (A - D) displays the selected task in the workspace.

Subroutines

Selecting **View** ⇒ **Subroutines** displays the **Subroutines** window. Viewable subroutines must have been previously created by selecting **Edit** ⇒ **Add Subroutine**.

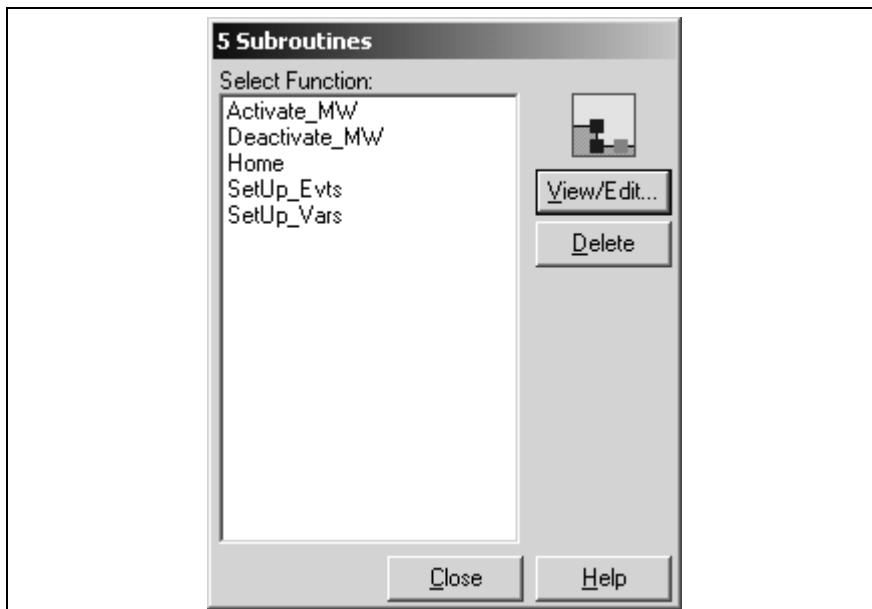


Fig. 7-23: "Subroutines" Window

The programmed subroutines are listed by name in a selection list and the total number is listed in the title bar. Selecting an existing subroutine from the displayed list, permits deleting it or replacing the current VisualMotion workspace with the subroutine for viewing/editing.

A subroutine may also be loaded to the workspace by double-clicking an existing subroutine icon to display the **Subroutines** icon window.

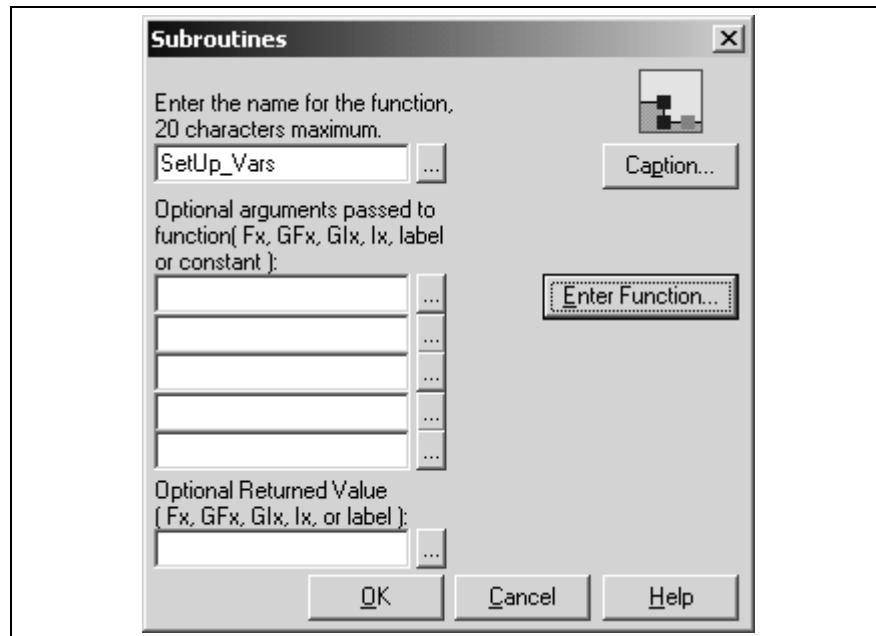


Fig. 7-24: "Subroutines" Icon Window

In this window, optional arguments and an optional returned value could be entered. The subroutine icon's 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** \Rightarrow **Event Functions** displays the **Event Functions** window. Viewable event functions must have been previously created by selecting **Edit** \Rightarrow **Add Event Function**.

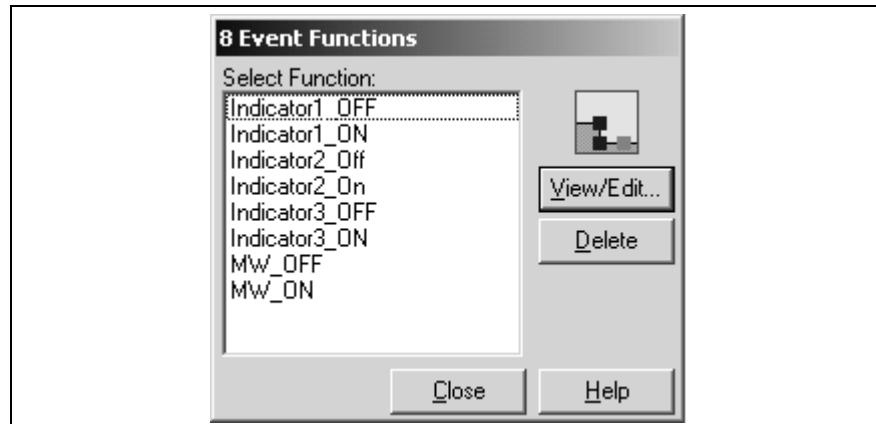


Fig. 7-25: "Event Function" Window

The programmed event functions are listed by name in a selection list and the total number are listed in the title bar. Selecting an existing event function from the displayed list permits deleting it or replacing the current VisualMotion workspace with the event function for viewing/editing.

Note: The only way an event function workspace can be displayed is through the **View** \Rightarrow **Event Functions** menu item.

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), can be used to move from the back to the front of 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:

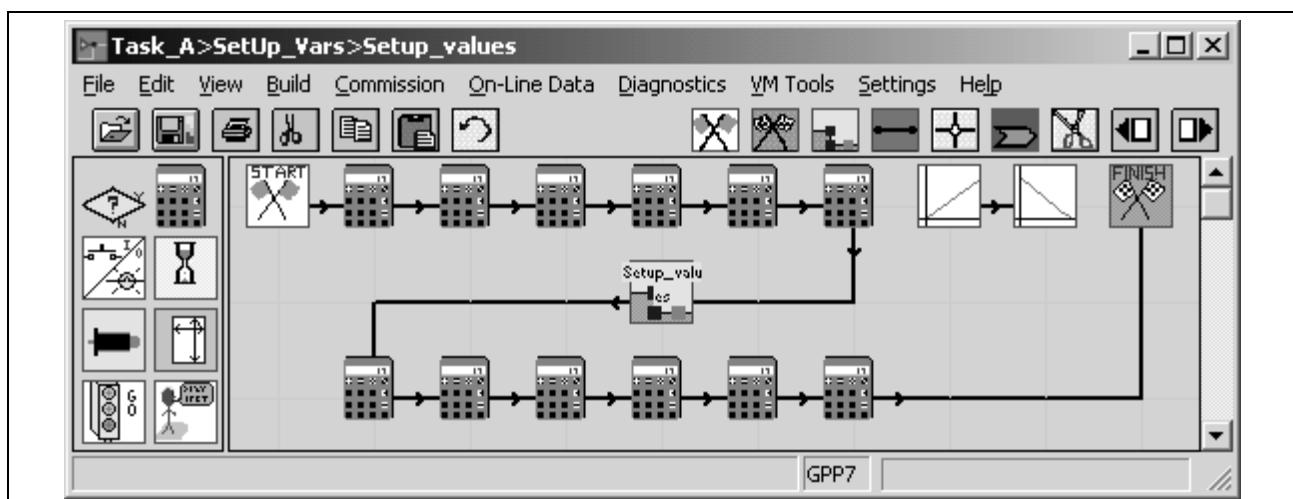
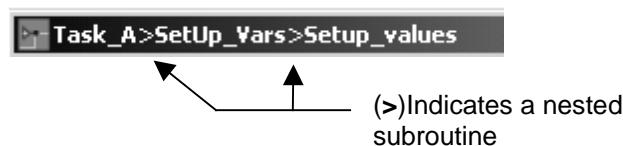


Fig. 7-26: Nested Subroutine Queue in the VisualMotion Menu Bar

The **Zoom Out** command is unavailable when any of the four main tasks (A - D) are loaded in the workspace or when the presently loaded subroutine was opened via the **Subroutines** menu command instead of the **Enter Function...** button of a subroutine icon. 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

The View menu provides pre-configured palettes for single axis, coordinated, Electronic Line Shaft (ELS) and Utility. Choosing one of these menu items loads the selection onto the palette area on the left of the VisualMotion workspace.

Icon Palette 'Single' from the Options menu selects a set of icons used frequently in single axis control.

Icon Palette 'Coord' selects a set of icons used frequently in coordinated control.

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

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

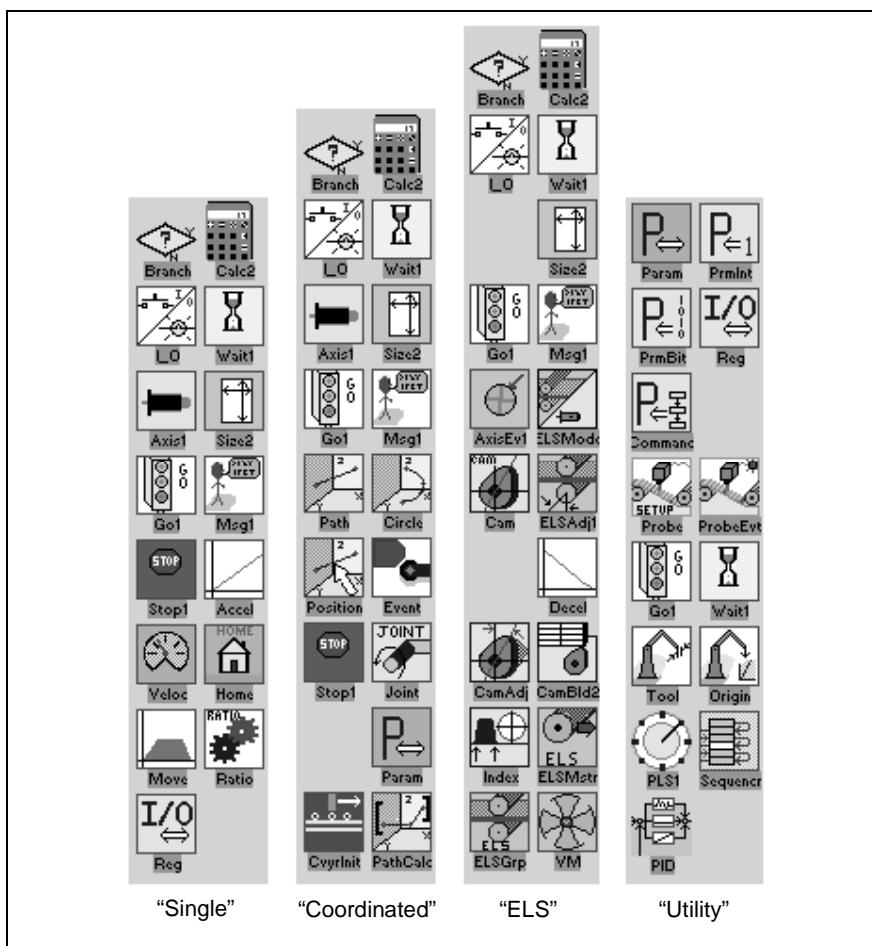


Fig. 7-27: Single, Coordinated, ELS, and Utility Icon Palettes

Icon Comments

Choosing **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 box. The setup box opens when the icon is first placed or by double clicking the left mouse button while the pointer is over the icon. Each box 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

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

7.5 The Build Menu

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

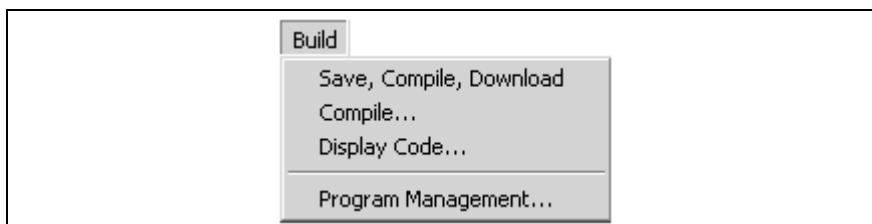


Fig. 7-28: The Build Menu

Save, Compile, Download

Saves the currently open program, compiles it, and downloads it to the control. The Program Management window in Fig. 7-29 opens when all three functions have successfully been carried out.

Compile

Checks and converts icon or textual language program to code the control can interpret, prompting for a program name.

Display Code

Displays the "Base Code," which is intermediate code generated by the first-pass compiler and used by the second pass compiler, using the default text editor (e.g. Notepad, WordPad, etc.). This base code is useful only for reference to errors generated by the second-pass compiler.

Program Management

The Program Management window is used to activate and download VisualMotion user programs to the control. Ten (10) user programs can be downloaded to the control with only one activated at any give time.

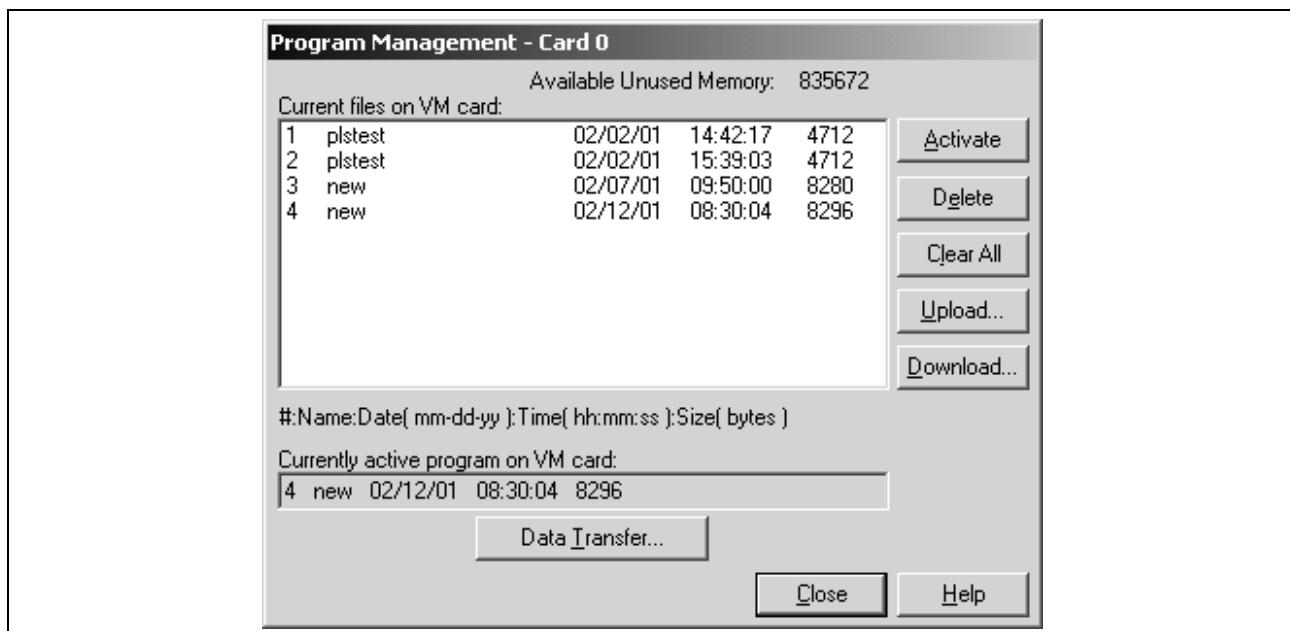


Fig. 7-29: "Program Management" Window

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. Each activation of a different user program reduces the amount of flash memory available in VisualMotion. 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 activates the highlighted program on the control. A program cannot be run until it is activated. The active program can only be changed when the active program is not running.

Delete

This button deletes the highlighted program from the control. A confirmation window opens. If a program is currently active, it cannot be deleted from the control.

Clear All

This button erases all resident programs and data from the control's memory. A confirmation window opens. **See Archive** before clearing.

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. After clicking the **Download** button, select or type the desired filename, and then click the **Open** button. Executable programs are stored with the ".exc" file extension, together with other project files in the Project Directory, which is defined using the **Settings** ⇒ **Configuration** command. Once the file has been opened, another window is displayed for specifying a program handle (number). You may choose any handle (1 - 10) not presently used in the control. The default handle is the next unused handle. After clicking **OK**, the download is executed, and a progress bar shows the download status.

Close

This button closes the current action and closes the Program Management window.

Data Transfer

This button transfers data items between the active program (Source) and the (Destination) program selected for activation. Clicking on the **Data Transfer** button opens the Data Transfer window.

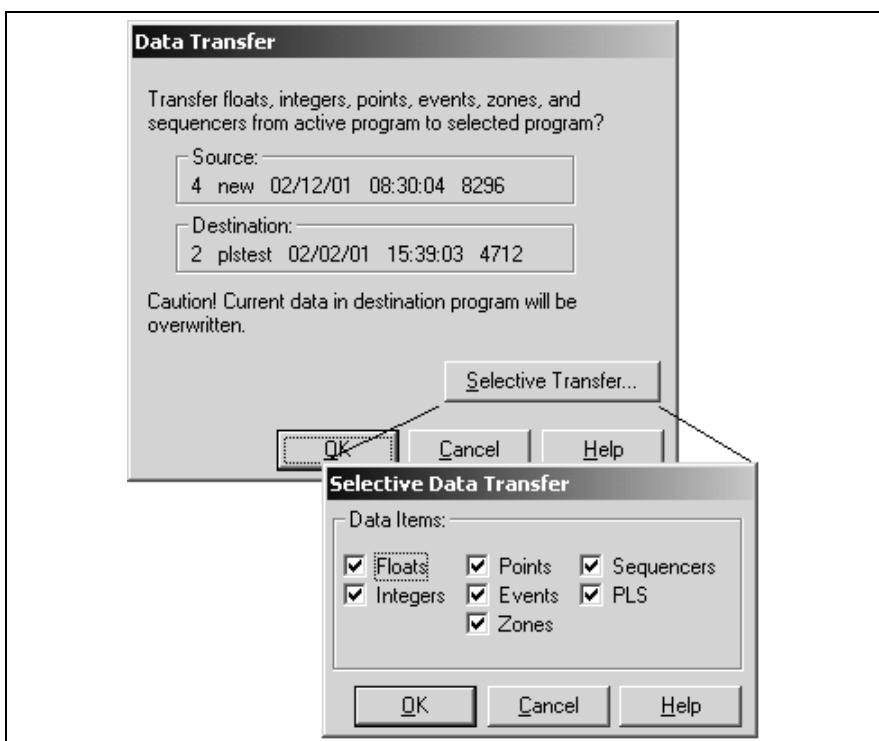


Fig. 7-30: Data Transfer

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 destination program's data sets.**

⇒ If the source program's "Data Size" allocation (i.e., the size of the data sets) is larger than the target, then only the data elements within the "Data Size" allocated by the target are transferred.

For Example:

If the active program has 75 events and the selected target program has a Data Size allocation specifying 50 events, 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.

7.6 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 coordinated motion requirements, archive and transfer program data.

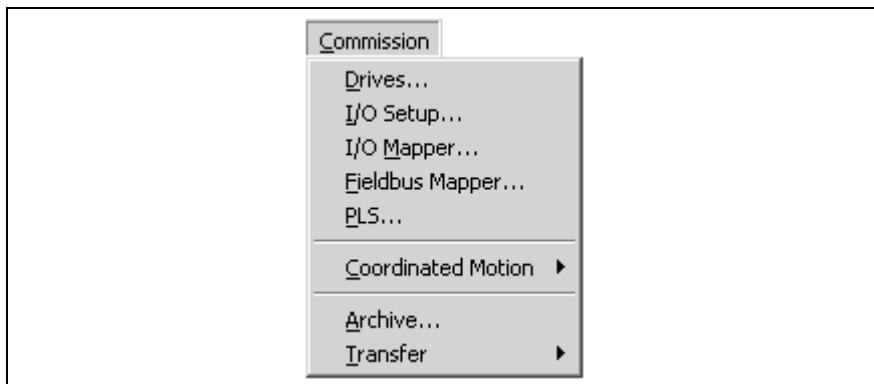


Fig. 7-31: The Compile Menu

Drives

Selecting **Commission** ⇒ **Drives** opens the Drive Parameter Editor Window in Fig. 7-32.

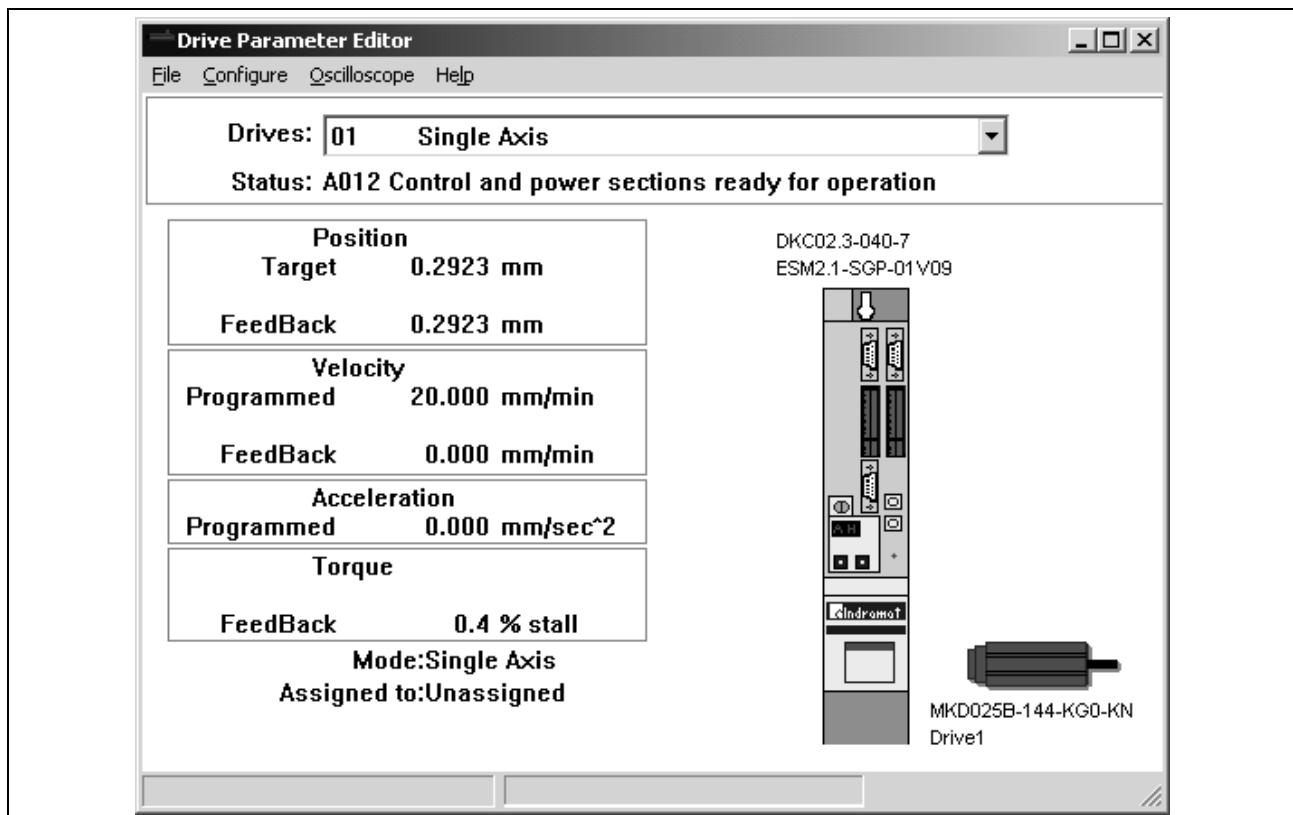


Fig. 7-32: Drive Parameter Editor

For instructions on how to use this program, refer to [VisualMotion 8 \(GPP\) Application Manual, Drive Parameter Editor.](#)

I/O Setup

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

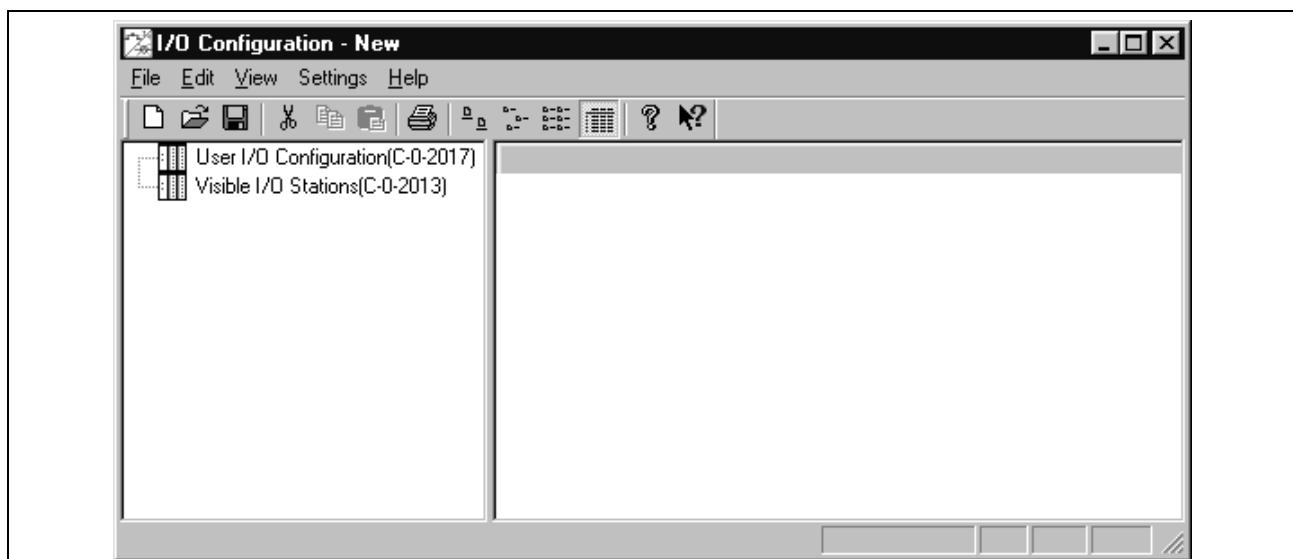


Fig. 7-33: I/O Configuration Window

For information about how to use this program, refer to chapter 2, [I/O Configuration Tool](#)

I/O Mapper

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

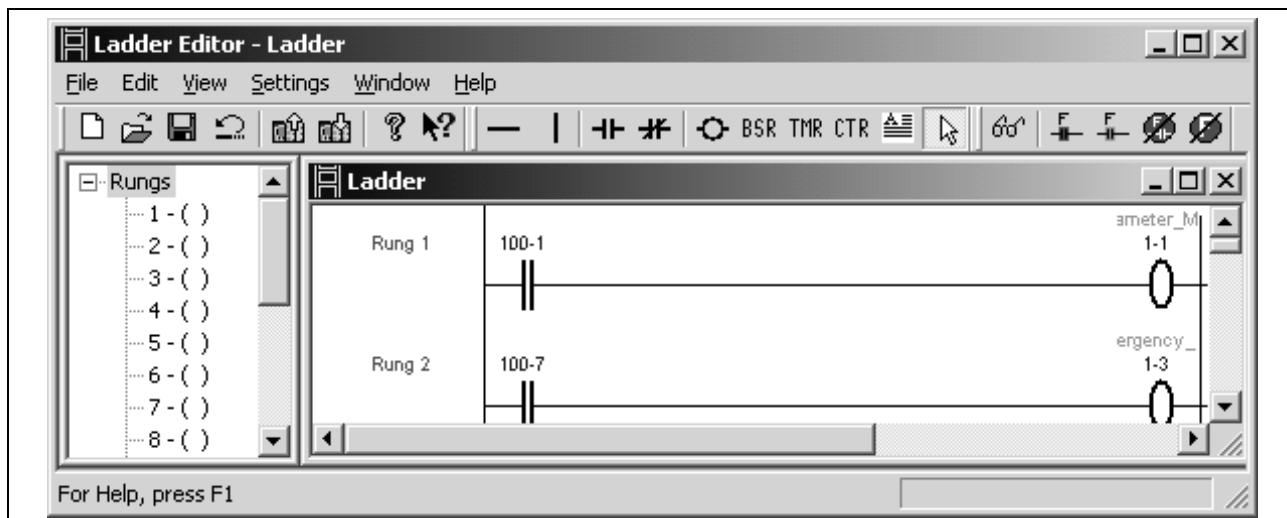


Fig. 7-34: Enhanced I/O Mapper

For information about how to use this program, refer to chapter 3, [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:

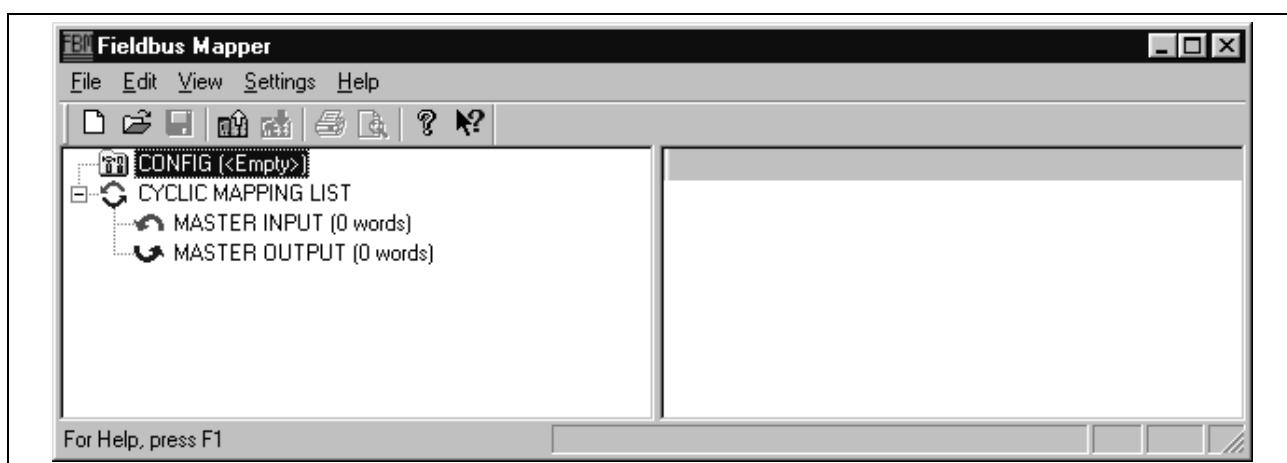


Fig. 7-35: Fieldbus Mapper, Main Window

The available fieldbus types for VisualMotion 8 are:

- Profibus
- DeviceNet
- Interbus
- ControlNet

The data types that can be mapped include:

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

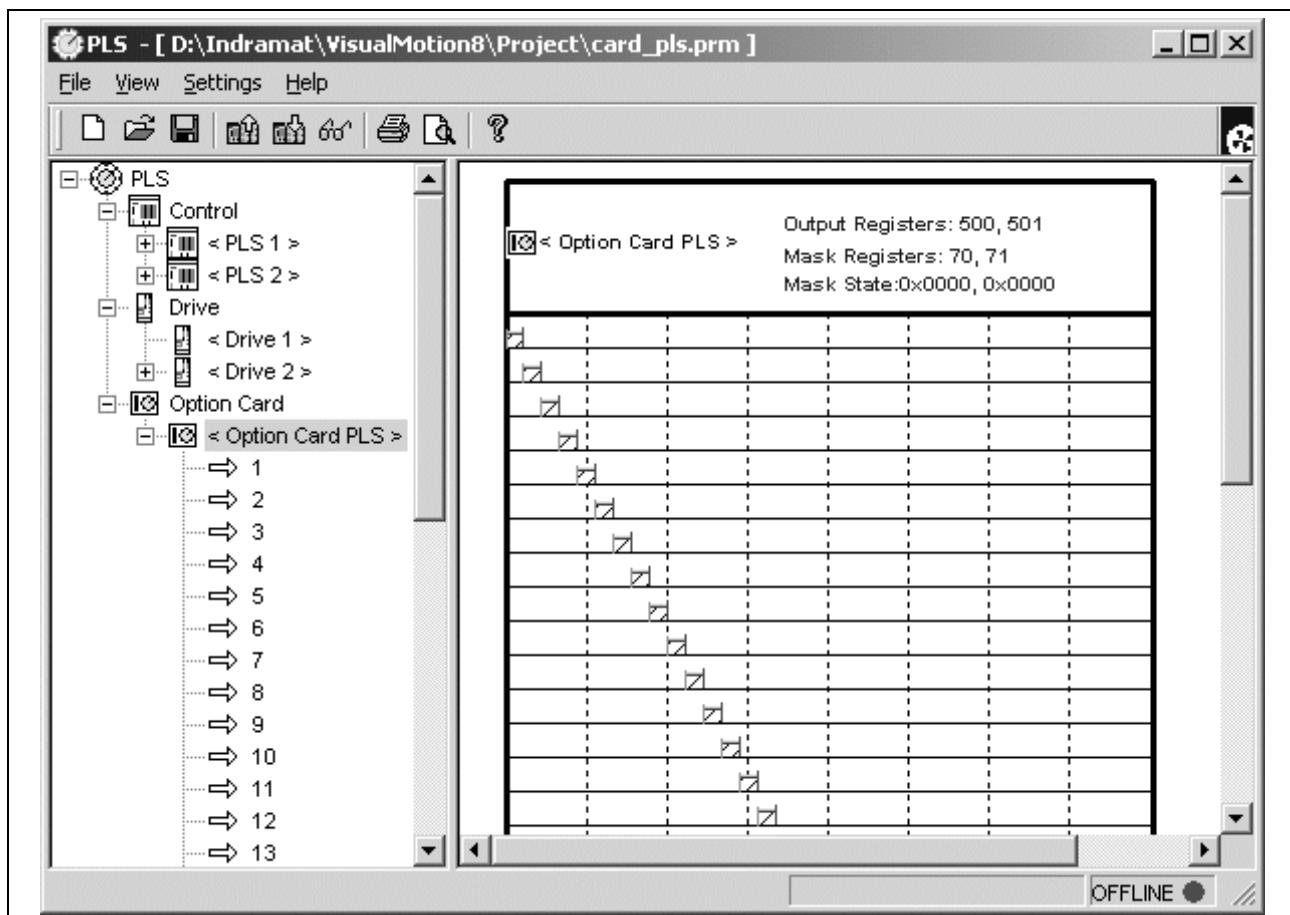
For specific information about each fieldbus and for directions on how to map data, refer to the *VisualMotion 8 Application Manual (DOK-VISMOT-VM*-08VRS**-AW01-AE-P)*: [Profibus Fieldbus Interface](#), [DeviceNet/ControlNet Fieldbus Interface](#) and [Interbus Fieldbus Interface](#).

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.



PLS Configuration Tool

For information about how to use this program, refer to chapter 4, [Programmable Limit Switch Functionality](#).

Coordinated Motion

The Coordinated Motion menu selection is used to configuration coordinated motion functionality.

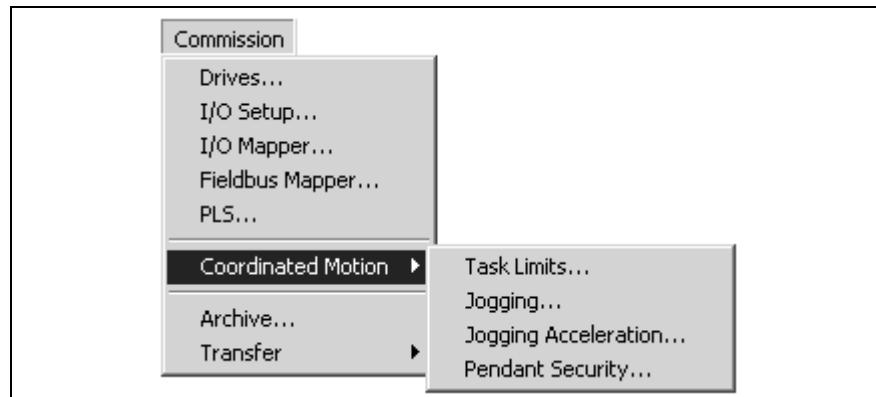


Fig. 7-36: *Commission* ⇒ *Coordinated Motion*

Task Limits

The Task Maximum Path Limits window allows speed, acceleration and deceleration limits for coordinated motion to be individually set for each task. Selecting ***Commission*** ⇒ ***Coordinated Motion*** ⇒ ***Task Limits*** opens the following window:

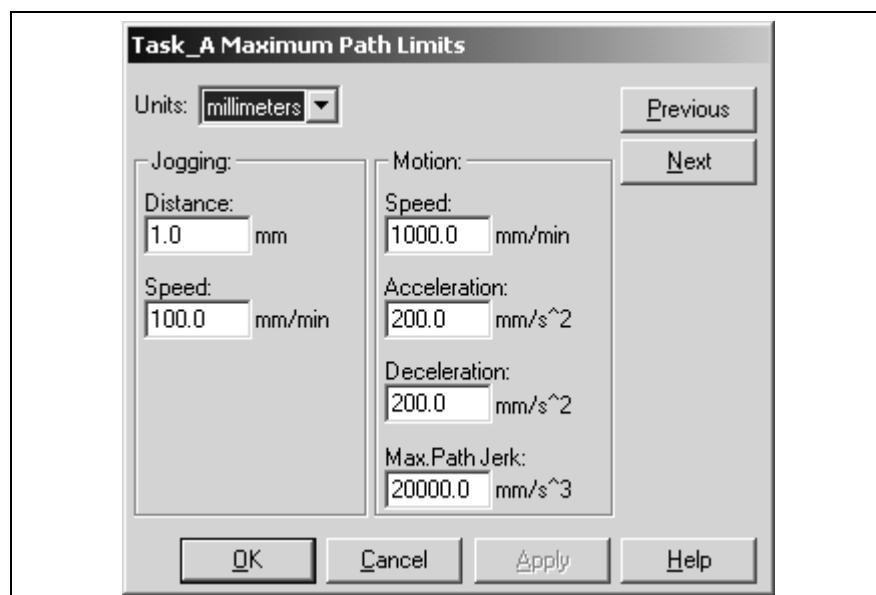


Fig. 7-37: "Task_x Maximum Path Limits" Window

This window allows setting the units (to inches, millimeters or radians) for the Maximum Path Limit values. It also permits setting the jogging distance and speed, 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.

Clicking **OK** downloads the changed values to the control, without requiring the control to be in Parameter Mode. The **Previous** and **Next** buttons switch between tasks. **Cancel** exits the window.

Jogging

This window allows the user to set the increments and velocities used for fast and slow jogging. Selecting ***Commission*** \Rightarrow ***Coordinated Motion*** \Rightarrow ***Jogging*** opens the following window:

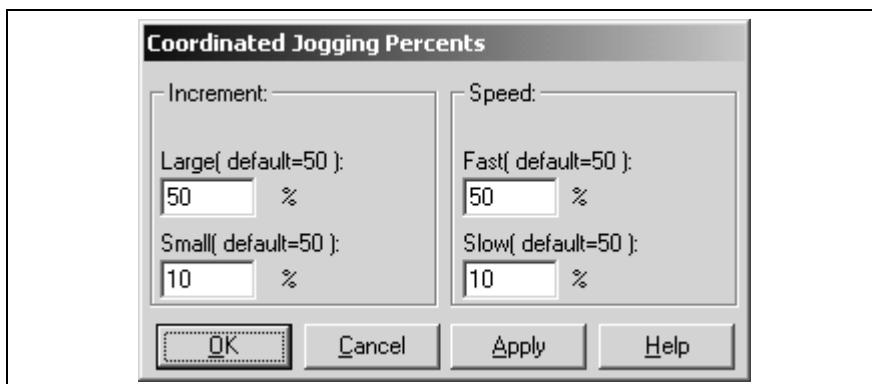


Fig. 7-38: "Coordinated Jogging Percents" Window

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" (A-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" (A-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 users to set the jogging acceleration and deceleration for each axis. ***See Parameter A-0-0021 and A-0-0022 for more information.***

Selecting ***Commission*** \Rightarrow ***Coordinated Motion*** \Rightarrow ***Jogging Acceleration*** opens the following window:

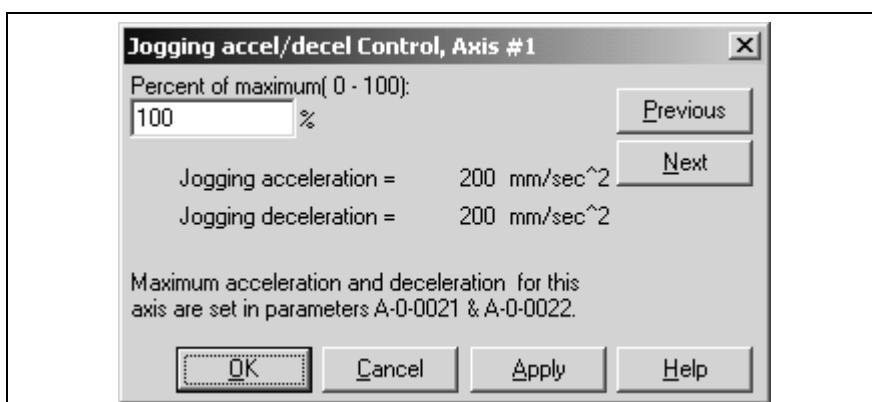


Fig. 7-39: "Jogging accel/decel Control" Window

The **Previous** and **Next** buttons scroll through the different axis numbers. After making the desired changes, click **Apply** or **OK**.

Pendant Security

Selecting ***Commission*** ⇒ ***Coordinated Motion*** ⇒ ***Pendant Security*** opens the Teach Pendant Security window.

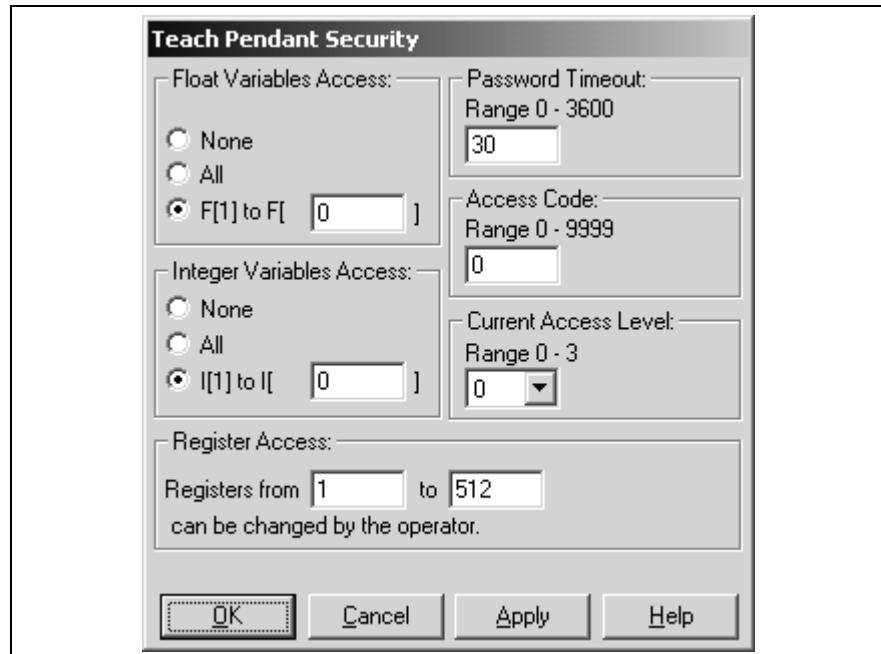


Fig. 7-40: "Teach Pendant Security" Window

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

- **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
- **Password Timeout** – restricts the time the user has to enter the password (in ms).
- **Access Code** – sets the password (access code) that must be entered to gain access to the designated variables and registers.
- **Current Access Level** - 0= , 1= , 2= , 3=
- **Register Access** – sets user access to a range of registers.

Archive

Selecting **Commission** ⇒ **Archive** displays the **Archive** window.

This window provides backup and restore functions to the control. A full or selective backup or restore is possible. Selected backup items such as parameters, programs, drives, etc. are saved by default to the \Indramat\VisualMotion 8\project\saveset directory. The **Browse...** button allows the user to select another directory location.

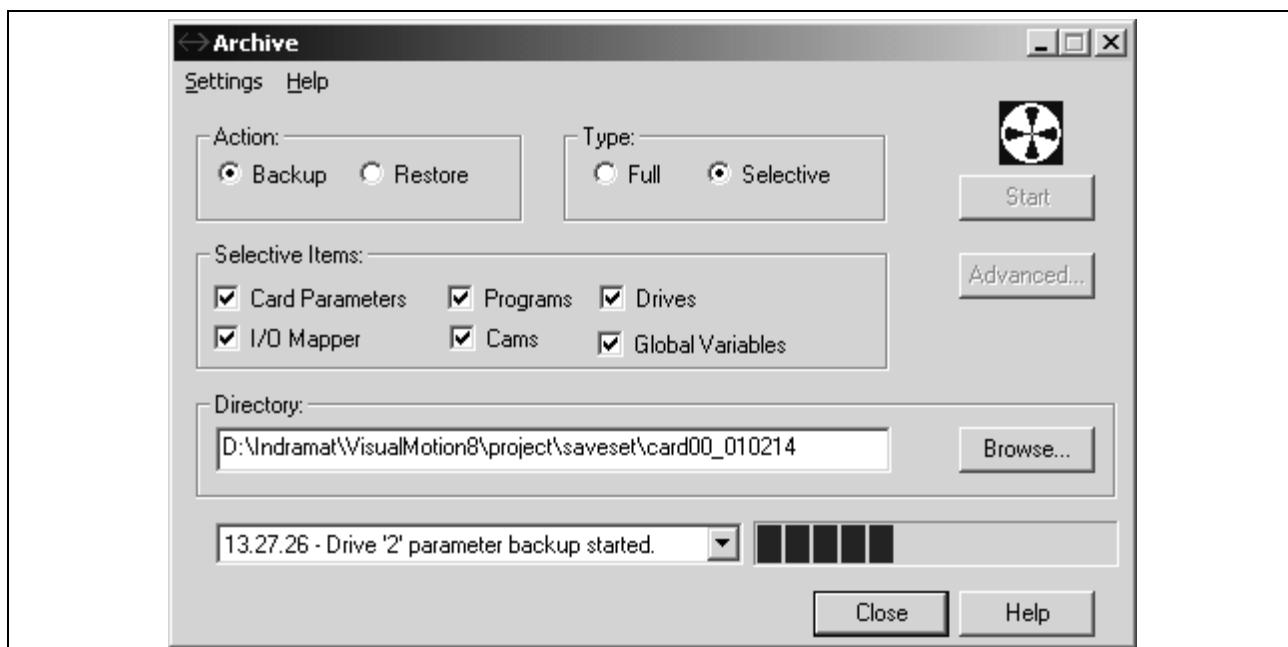


Fig. 7-41: "Archive" Window

Select **Backup** or **Restore**, select **Full** or **Selective**, then press the **Start** button to transfer. If **Full** is chosen, all of the Selected Items below are automatically grayed-out and selected. If **Selective** is chosen, the checkboxes are enabled to allow selection data to be backed up or restored. During data transfer, a progress bar at the bottom right of the screen shows the transfer status.

Note: The directory field automatically creates a folder under the saveset folder for the data being backed up. The folder name is designed to include the control number and date of backup in the following format:

```
card00_010214
    |      |____ computer date (yr/month/day)
    |      |
    |_____|____ control number
```

Only parameter sets of active drives (those defined in the active program) are backed up. VisualMotion cannot detect other drives. Backup files are named as follows:

Cam Tags	"camname.txt"	info contained in C-0-3100
Saveset	"chklist.txt"	system info
Drives	"drivxx.prm"	where xx is drive number
System	"system.prm"	
I/O Mapper	"mapper.iom"	

Programs "mmmm.exn" where *mmmm* is program name, *n* is the program handle (1-9 or "a")
 Cams "cam.csv"

Advanced Restore

When performing a Restore - Selective, the "**Advanced...**" button becomes active and displays the following window when clicked.

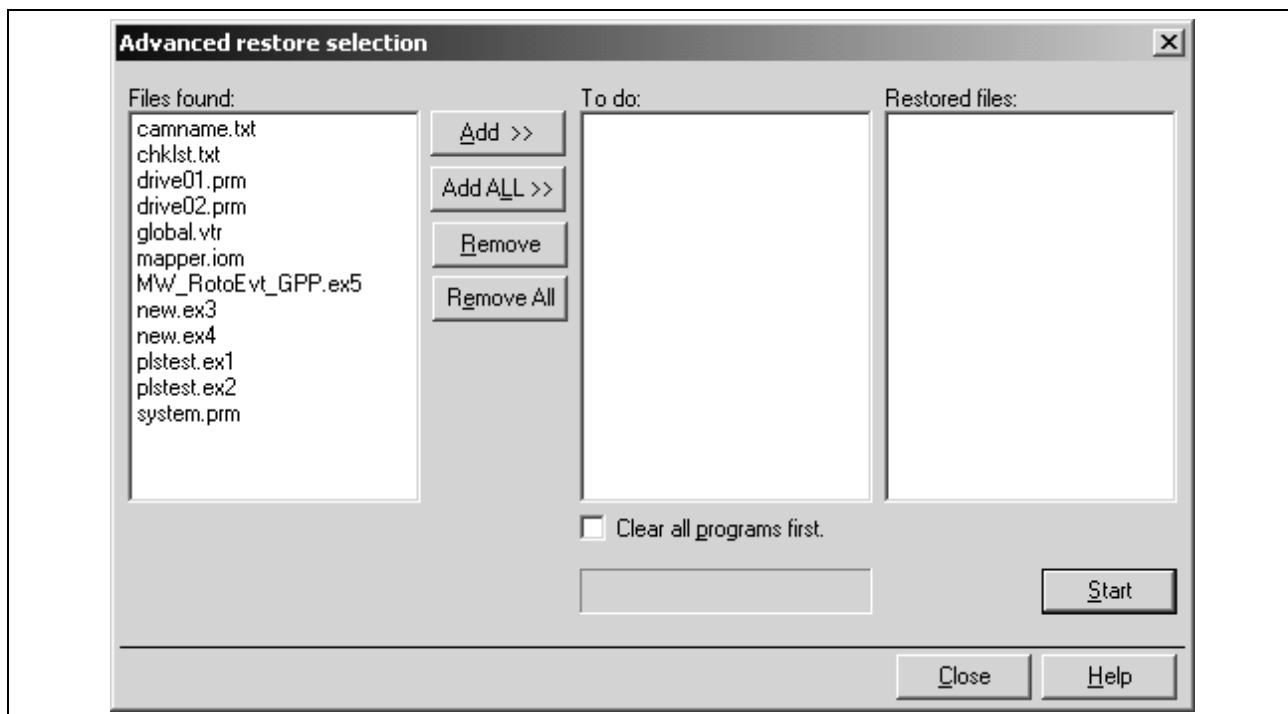


Fig. 7-42: "Advanced Restore" Window

The **Advanced restore selection** window allows the user to restore backup files to the control. The following buttons are available:

ALL>>

Adds all backup files under "Files found:" to the "To do:" list. Click **OK** to restore all the files.

Note: If files that are not backup files reside in the directory designated in the main archive screen, these files are listed under "Files found:" but they are ignored when clicking **ALL>>**. No error or warning is displayed.

Add>>

Adds the highlighted backup file(s) under "Files found:" to the "To do:" list. Click **OK** to restore the selected files. You can also add individual files by double-clicking each filename in the "Files found:" list.

Note: If files are highlighted that are not backup files, they are listed under "Files found:" but they are ignored when clicking **Add>>**. No error or warning is displayed.

Remove

Removes the highlighted file from the "To do:" list.

Remove All

Removes all files from the "To do:" list.

Clear all programs first checkbox - Allows the user to clear all stored programs on the control before restoring any selected programs.

Restoring a Drive Parameter

When a drive parameter (e.g. drive01.prm) is selected to be restored, the following window opens:

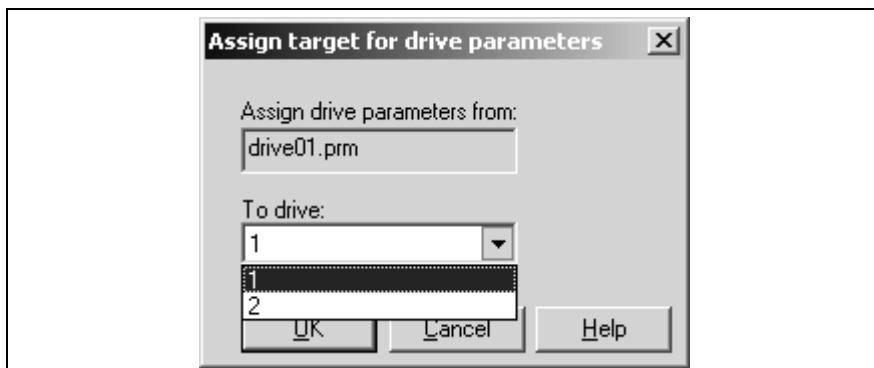


Fig. 7-43: Drive Parameter Assignment

This function allows the user to select a drive other than the original drive for restoring drive parameters, if desired. For example, the parameters from drive 1 can be restored to drive 2 (as shown above), or to any of the other drives in the system. The "To drive:" pull-down list displays all the SERCOS drives from control parameter C-0-2011 (list of SERCOS drives).

Transfer

The Transfer command is used to transfer program data between programs. Selecting **Commission** ⇒ **Transfer** opens a third menu selection where the user selects the data to transfer.

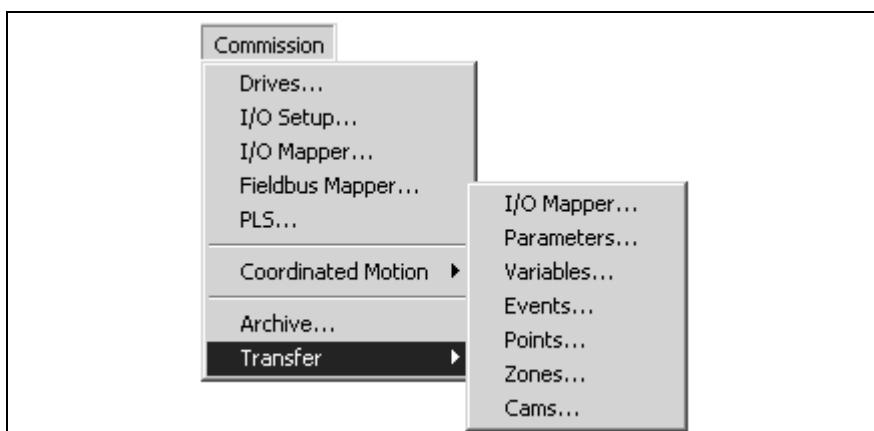


Fig. 7-44: Commission ⇒ Transfer

I/O Mapper

This menu item allows for uploading or downloading I/O Mapper strings to or from the VisualMotion control, and performs the same functions as a selective backup (upload) or selective restore (download) using the **Archive** window.

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



Fig. 7-45: "Transfer I/O Mapper Strings" Window

Uploading prompts for a filename to be saved, and downloading prompts for an existing filename to be opened. I/O Mapper string files are stored with the ".iom" file extension with other project files in the Project Directory defined using the **Settings** ⇒ **Configuration** command.

Parameters

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

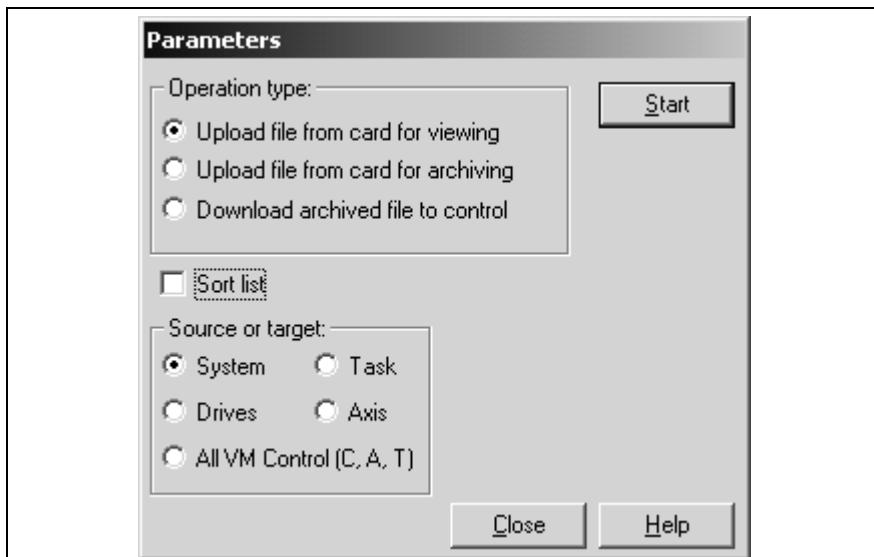


Fig. 7-46: "Parameters" Window

Parameters may be uploaded in one of two formats:

- **Upload file from card for viewing** saves the file to the same sub-directory as a text file with a ".txt" extension and may be viewed using Notepad or another ASCII text editor or file viewer.

Note: A *.txt parameter setup loaded for viewing cannot be downloaded to the control.

- **Upload file from card for archiving** saves the file with the ".prm" file extension in the :\indramat\VisualMotion8\project directory, with the data in the proper format for downloading to the control.

All control parameters may be transferred at one time, or parameter sets for the control, tasks or axes may be individually transferred. In addition, the parameter set for a selected drive connected to the control may be transferred through the SERCOS communication system.

Variables

Selecting **Commission** ⇒ **Transfer** ⇒ **Variables** displays the following window:

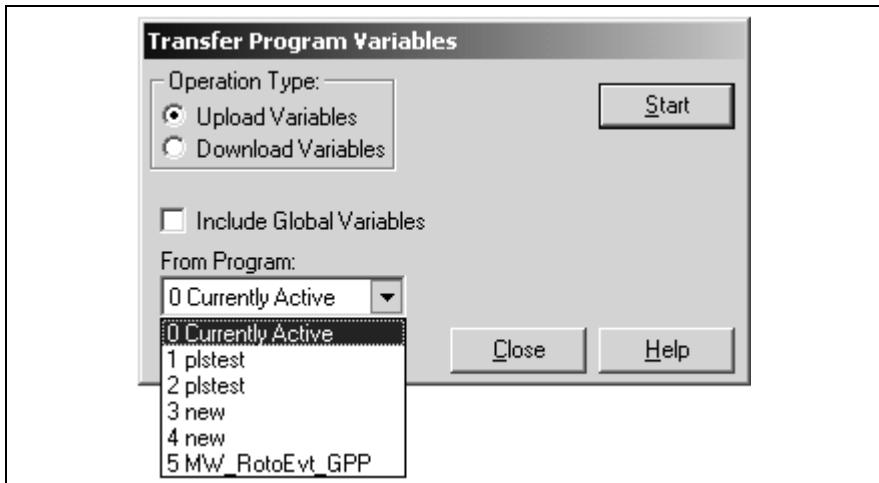


Fig. 7-47: "Transfer Program Variables" Window

This window allows for uploading or downloading program variable data from/to a program on the control to/from a file. All float (Fx) and integer (Ix) program variables are saved together in one file. Global variables can be included if desired by checking next to **Include Global Variables**. By default, the filename for a set of variables has a ".var" extension and is saved to the \VisualMotion 8\project directory. Because variable data is part of the program, it is also saved when the program is saved. This function is not available in the **Archive** window, where only global variables can be backed up or restored. Another window prompts the filename to be saved or opened. During the transfer, a progress bar indicates the transfer status.

Events

This menu item uploads and downloads event data between a program on the control and a file. This function is not available in the **Archive** window.

By default, event filenames have a ".evt" extension and are saved to the :\indramat\VisualMotion8\project directory. Because event data is part of the program, it is also saved when the program is saved. Selecting **Commission** ⇒ **Transfer** ⇒ **Events** displays the following window:

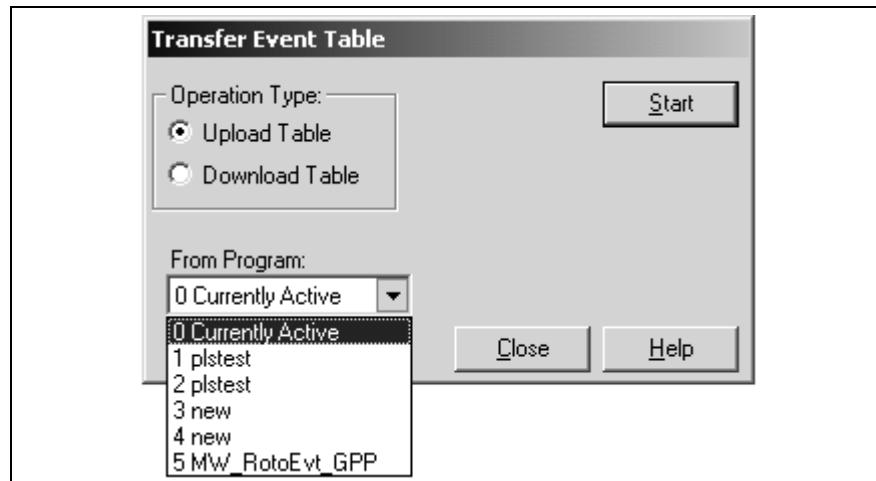


Fig. 7-48: "Transfer Event Table" Window

Select the operation type. The "To Program:" (when "Download Table" is chosen) or "From Program:" (when "Upload Table" is chosen) selection uses a combo box which contains a list of programs on the control. Click the **Start** button and another window prompts for a filename to be saved or opened. During the transfer, a progress bar indicates the transfer status.

Points

Selecting **Commission** ⇒ **Transfer** ⇒ **Points** displays the following window.

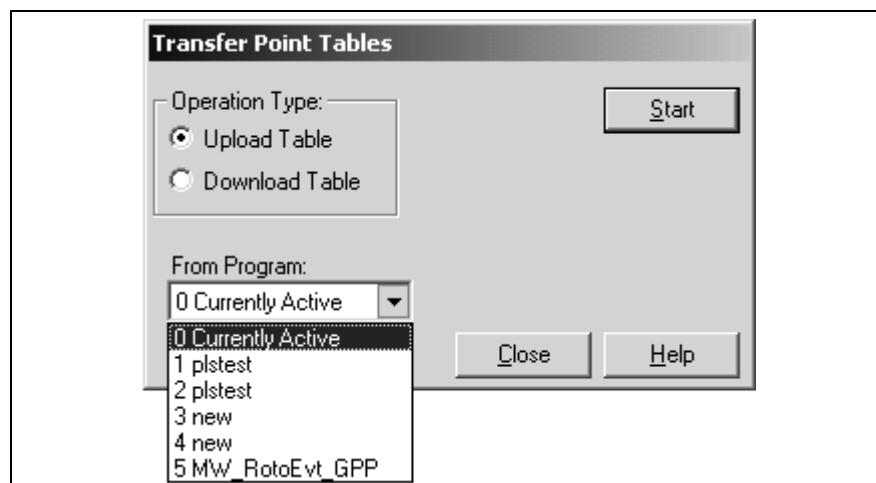


Fig. 7-49: "Transfer Point Tables" Window

The Transfer Point Tables window is used to upload or download the absolute and relative point tables to/from a resident control program from/to a point file on the Host system hard drive. Both the absolute and relative point tables are transferred at the same time. This function is not available in the **Archive** window.

Selecting **Upload Tables** and a control program opens a Save As window prompting for a filename. Point files are stored with the ".pnt" file extension together with other project files in the Project Directory defined using the **Settings** ⇒ **Configuration** command.

Selecting **Download Tables** and a control program opens a File Open window with \\indramat\VisualMotion8*.pnt selected by default. Clicking on **Open** downloads the selected file to the specified control program's absolute and relative point tables. A user may specify a different

subdirectory and file extension as long as the file contents are in a proper format.

The "To Program:" (when "Download Tables is chosen) or "From Program:" (when "Upload Tables" is chosen) selection uses a combo box which contains a list of programs on the control.

Note: Attempting to download a data set larger than that specified by the Size Icon in the control's resident program causes an error. Acknowledging the error terminates the operation with the remaining points not transferred.

Zones

Selecting **Commission** \Rightarrow **Transfer** \Rightarrow **Zones** displays the following window:

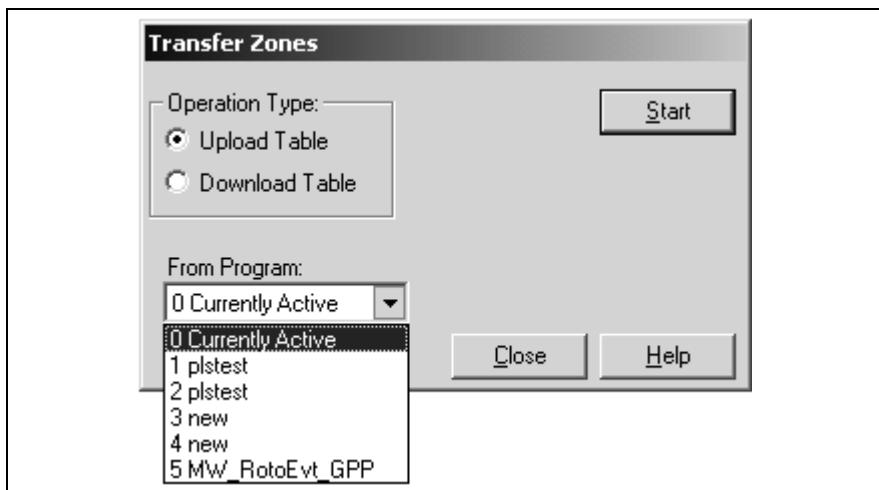


Fig. 7-50: "Transfer Zones" Window

This window allows for uploading or downloading program zone data from/to a program on the control to/from a file. This function is not available in the **Archive** window. System Zone files are stored with the ".zon" file extension with other project files in the Project Directory defined using the **Settings** \Rightarrow **Configuration** command.

Cams

This menu item is used to upload a cam table from the card or drive to a file for archiving, download a cam table file to the VisualMotion control or a drive, or delete a cam table on the card or a drive. This menu item can perform the same function as using the Selective Backup (Upload to Control Card) or Selective Restore (Download from Control Card) functions in the **Archive** window. Additionally, uploading and downloading of drive cams can be performed using this menu item.

Cam table files are stored with the ".csv" file extension with other project files in the Project Directory (set using the **Setting** \Rightarrow **Configuration** command). The ".csv" format is a standard used by EXCEL and other spreadsheets. Each line of the file consists of two ASCII floating numbers separated by a comma, and the line is finished with a <CRLF> (carriage return and line feed). Selecting **Commission** \Rightarrow **Transfer** \Rightarrow **Cams** displays the following window:

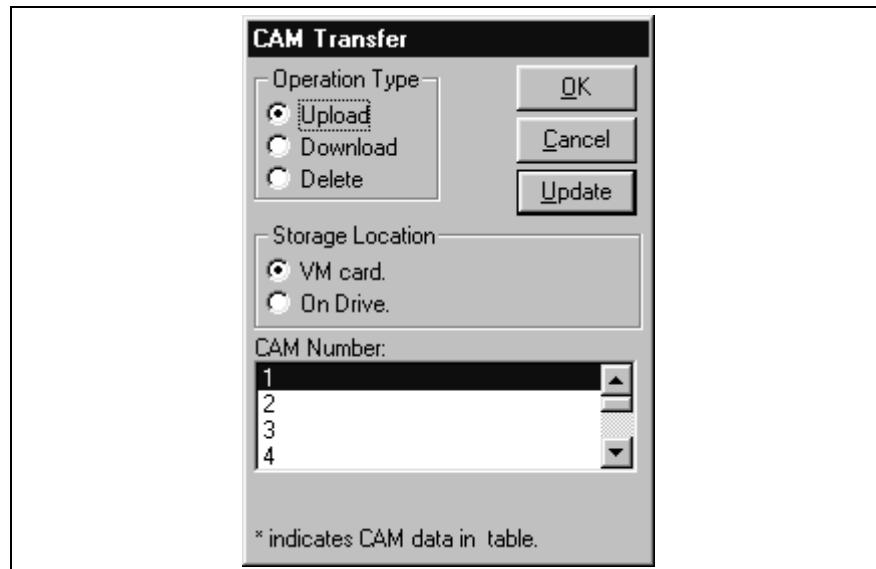


Fig. 7-51: "Cam Transfer" Window

The number of lines in a cam file to be downloaded can be from 10 to 1024. At the end of the download, the control expands the table to 1024 lines using a 5th order polynomial. When a table is uploaded from the control, the expanded table (1024 lines) is saved to a file.

There are 8 cam tables possible on controls. These tables are dynamically allocated to conserve memory when not used. There can be two cam tables on each drive. When a table is present, a "*" follows the number in the list box.

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

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.)

Select the operation type, storage location (when **On drive** is selected, also select the drive number in the list) and cam number, then press the **OK** button. Another window prompts the filename to be saved or opened. When selecting **Delete**, a warning is issued. During cam table transfer, a progress bar indicates the transfer status.

7.7 The On-Line Data Menu

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

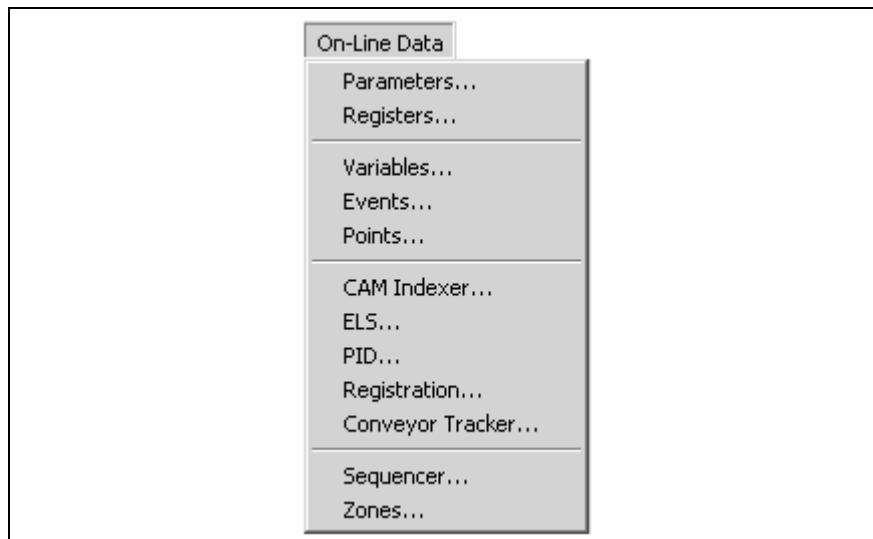


Fig. 7-52: The On-Line Data Menu

Parameters

The *Parameter Overview* tool in the figure below can be launched by selecting **On-Line Data** \Rightarrow **Parameters** from VisualMotion's main menu or by selecting **Configure** \Rightarrow **Parameters** from the Drive Parameter Editor.

This tool is used to view and modify existing Control, Task, Axis and SERCOS Ring 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®:

- 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 32) is selected.
- SERCOS All parameters for the selected SERCOS digital drive, up to a maximum of 32, and SERCOS I/O devices are listed.
- Custom Any existing custom list created by the user is listed.

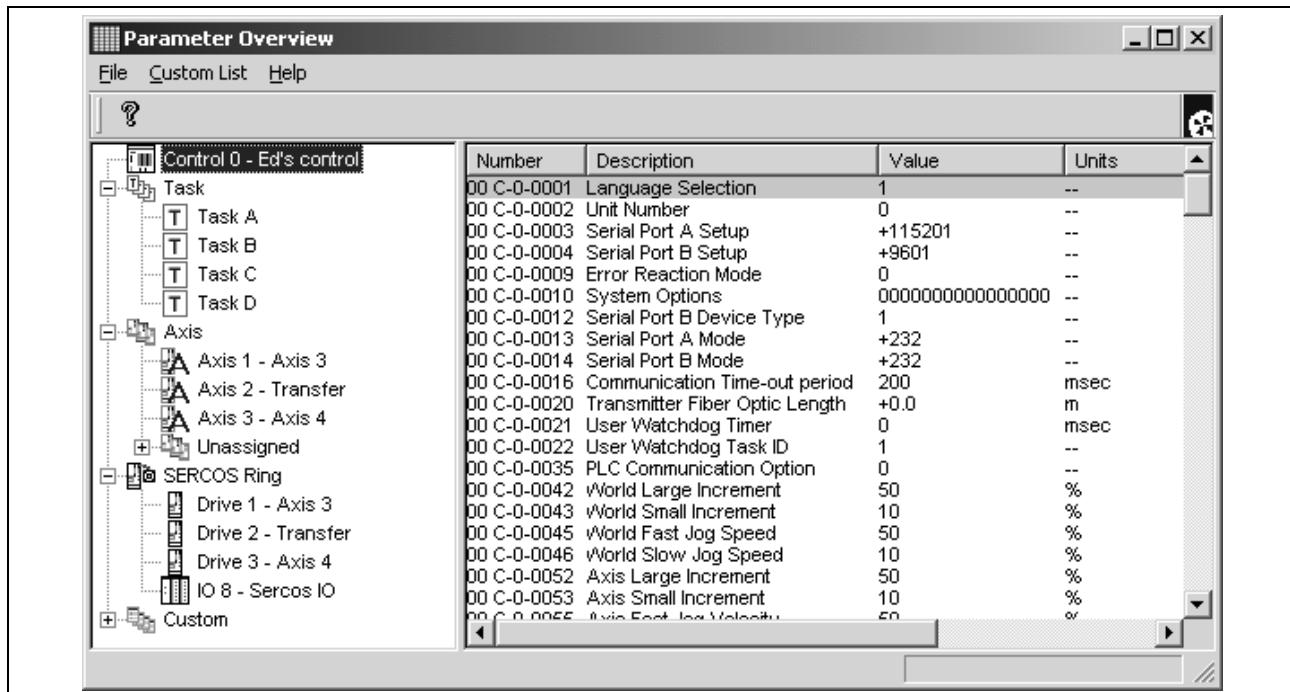


Fig. 7-53: Parameter Overview Window

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	Access Level
gray text	read only
black text	read/write
blue text (parameter list)	read/write
red text (Phase switch error)	read/write

Table 7-2: Parameter Access

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. The figures below show the three basic parameter edit windows that can be displayed. A gray input field is an indication that the parameter is read-only. Pressing the F1 key displays help on the edit window type. Pressing the Help button displays help for the selected parameter.

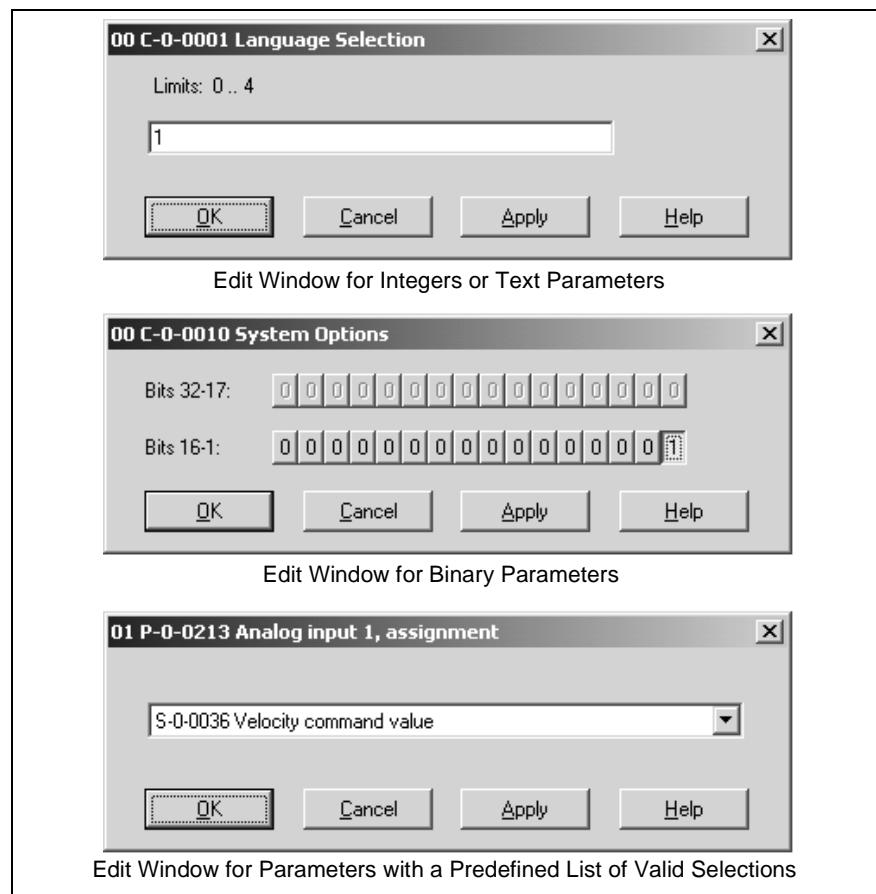


Fig. 7-54: Parameter Edit Windows

Help System not Found

VisualMotion issues the following error message when help is requested for a help system that does not exist. 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.

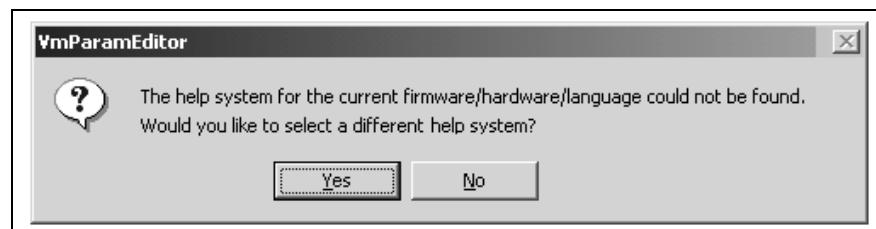


Fig. 7-55: Help System Error Message

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

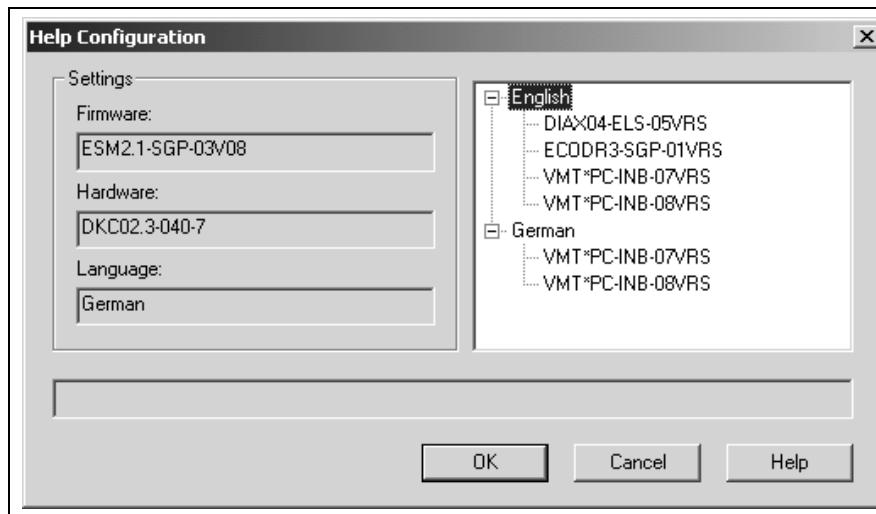


Fig. 7-56: Help System Location Window

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.

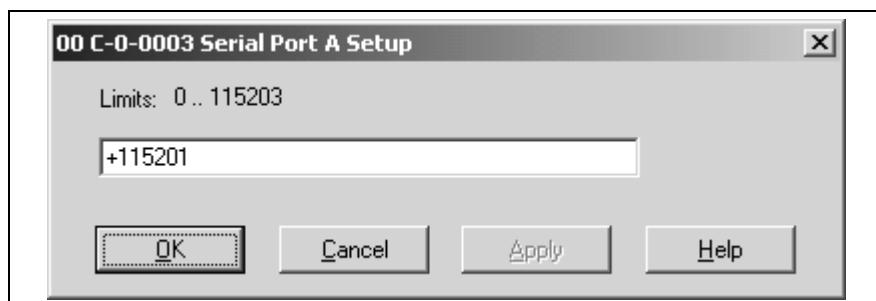


Fig. 7-57: Parameter Edit Window

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.

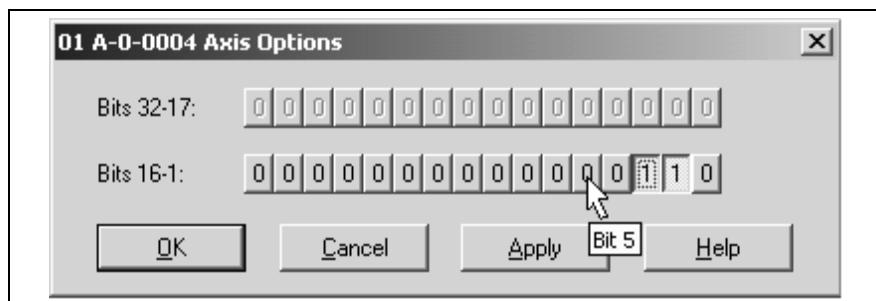


Fig. 7-58: Binary Parameter Edit Window

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.

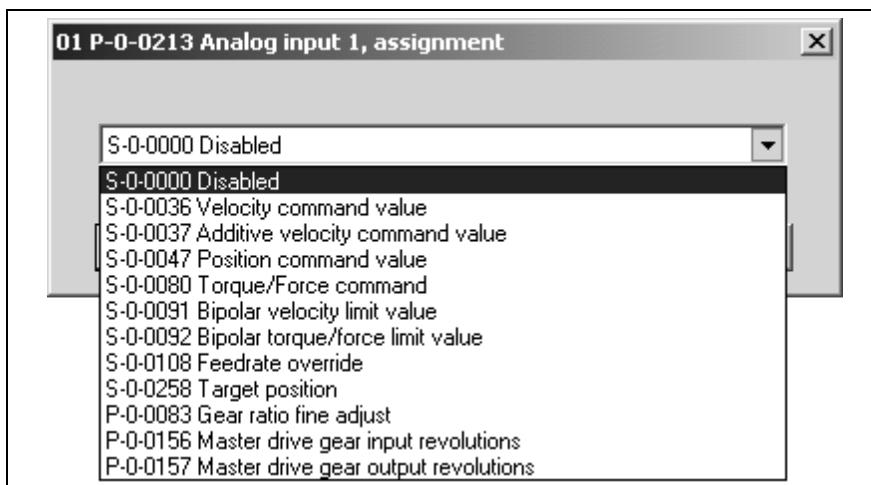


Fig. 7-59: Predefined Parameter List Edit Window

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.

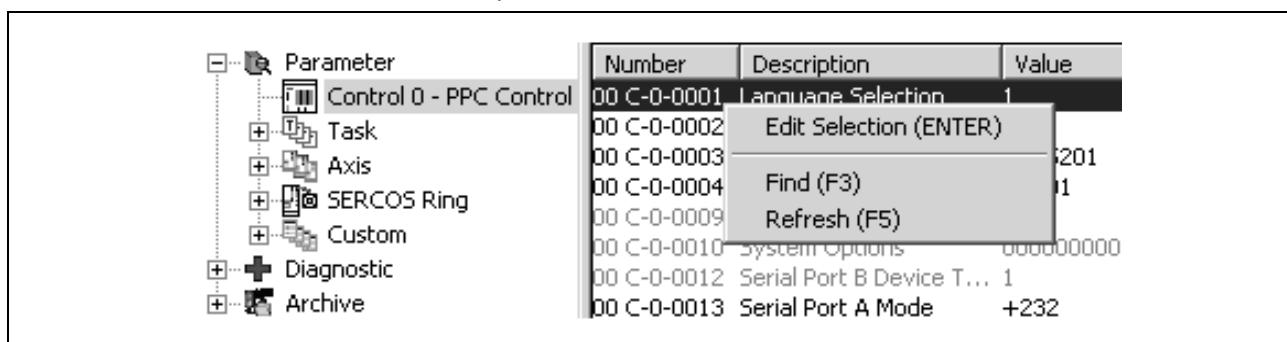


Fig. 7-60: Parameter Options

Note: Parameters and List 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.

Index	Value
1	1
2	2

OK Cancel Apply Help

Fig. 7-61: Parameter List Data Format

IDN Format

Parameter lists displayed in IDN (IDentification Number) format will contain an Index, Parameter Number and Description as shown in the figure below.

Index	Number	Description
1	00 C-0-0001	Language Selection
2	00 C-0-0002	Unit Number
3	00 C-0-0003	Serial Port A Setup
4	00 C-0-0004	Serial Port B Setup
5	00 C-0-0009	Error Reaction Mode

OK Cancel Apply Help

Fig. 7-62: 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.

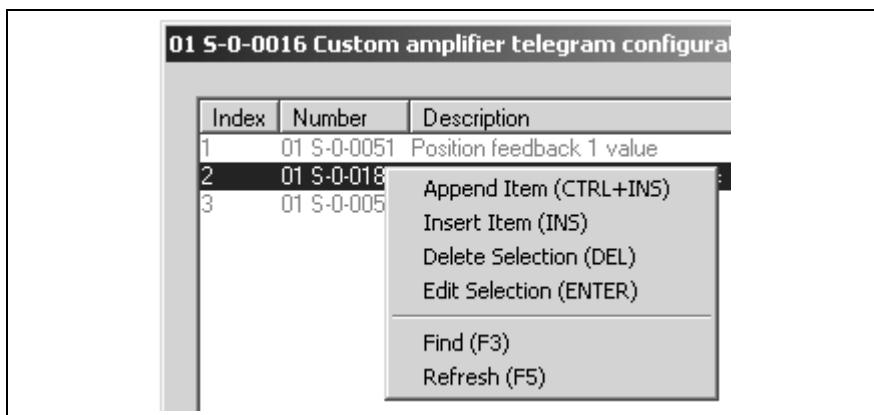


Fig. 7-63: Parameter List Options

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.

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 have this functionality.

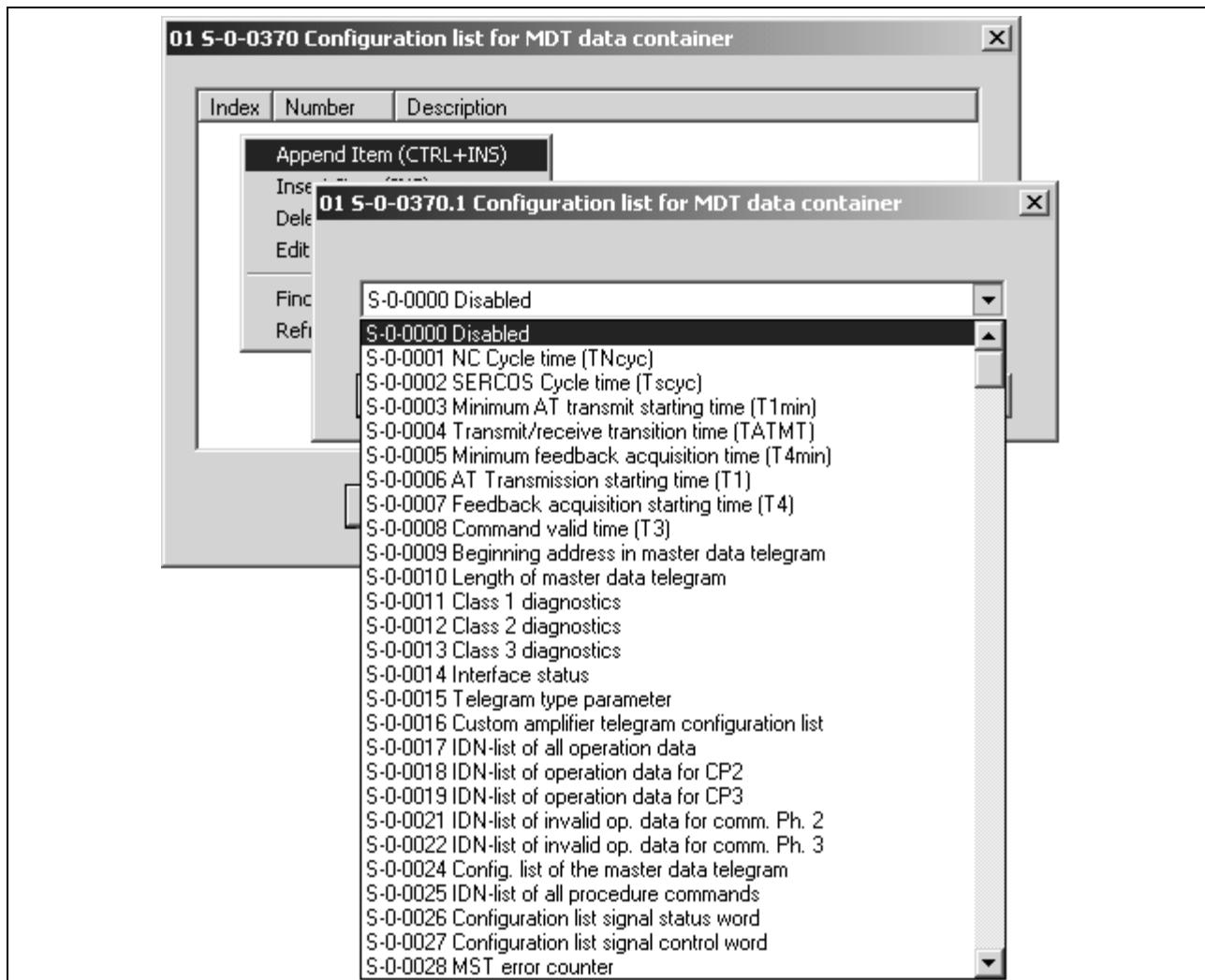


Fig. 7-64: Parameter Selection Edit Window

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 **\Indramat\VisualMotion8\param** directory. Custom list are saved with a ".custom" file extension and custom list groups appear as subfolders under **param**.

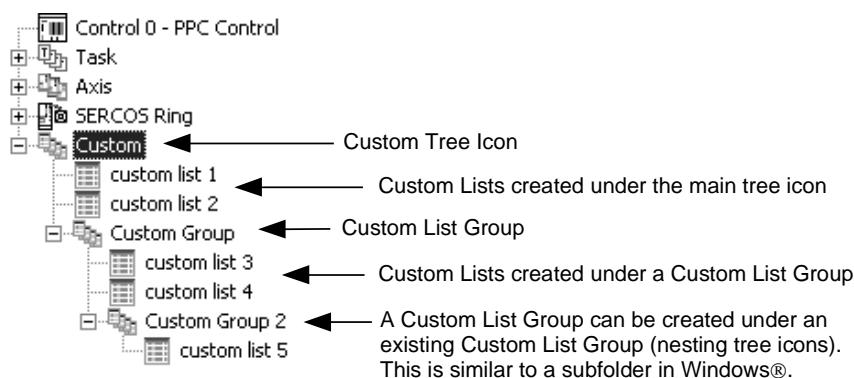


Fig. 7-65: Custom List File Structure

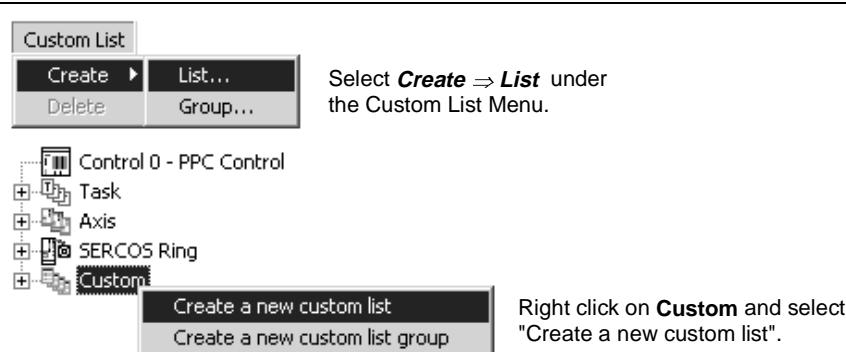
Create a Custom List

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.

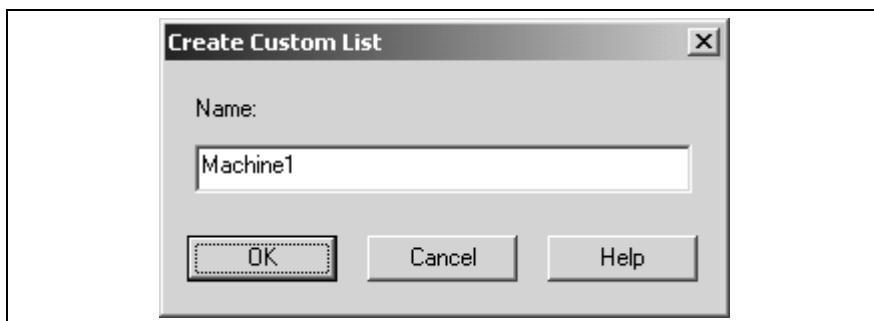
To create a new custom list, use the following steps:

1. Select **Custom List** \Rightarrow **Create** \Rightarrow **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.

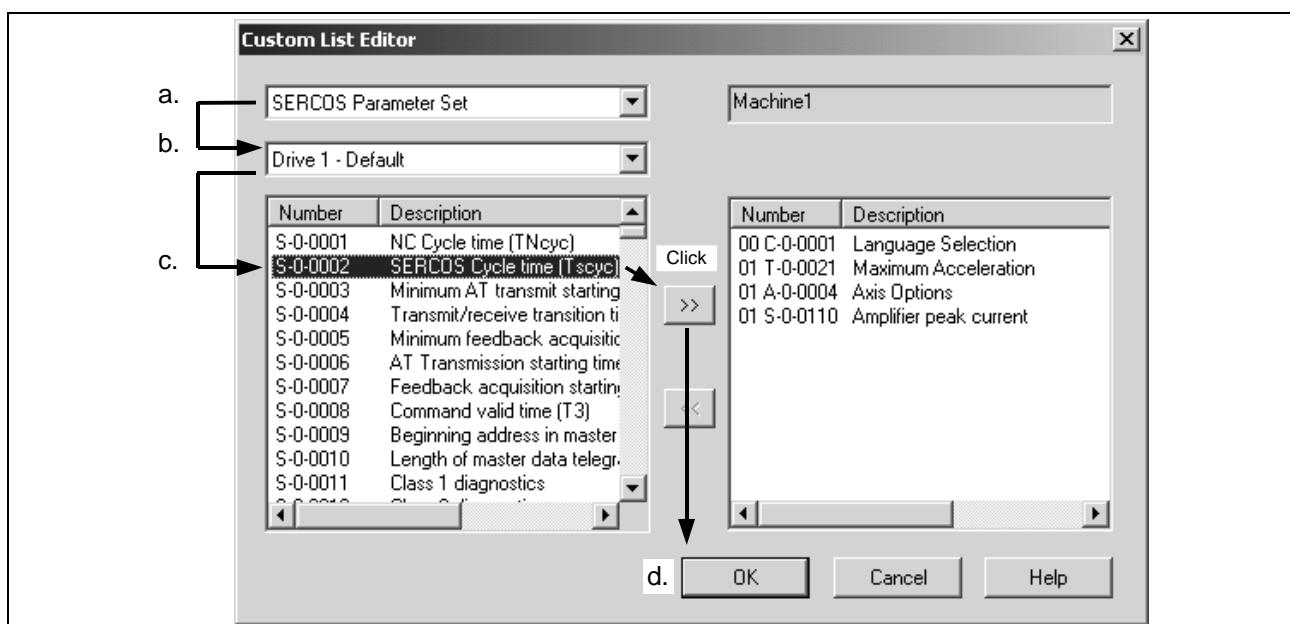


2. Enter a name in the *Create Custom List* window, up to a maximum of 80 characters, that identifies the list and press the **OK** button.



3. The following sequence is how parameters are added to a custom list within the *Custom List Editor* window.
- 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)
 - If applicable, select a task (A, B, C or D) for Task Parameter Set and an address number for Axis or SERCOS Parameter Sets.
 - 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 same or different parameter set until all the desired parameters are added to the list.
 - Press the **OK** button to complete the process.

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.



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

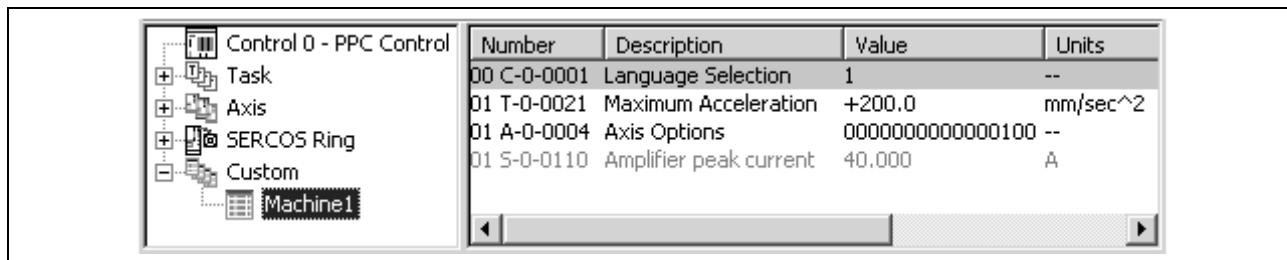


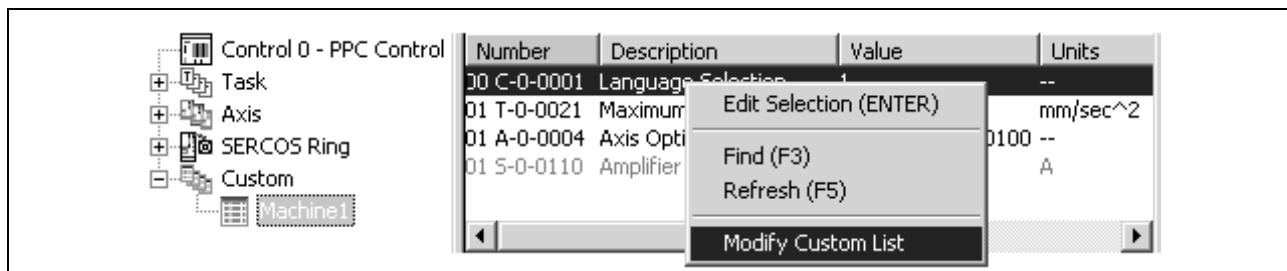
Fig. 7-66: Custom Tree Selection

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.



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".

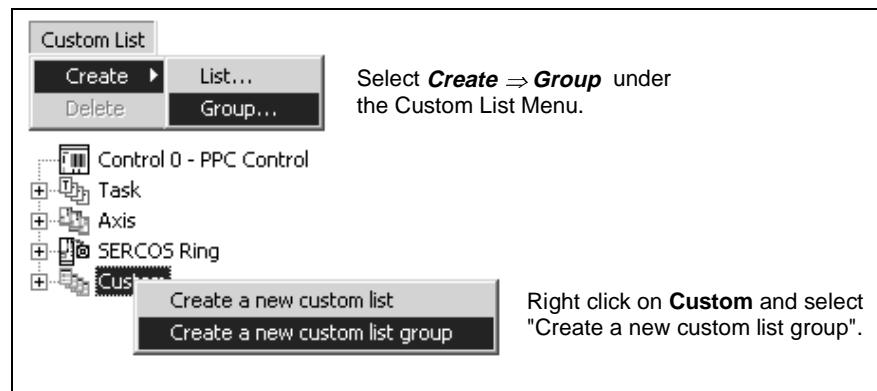
Create a Custom List Group

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).

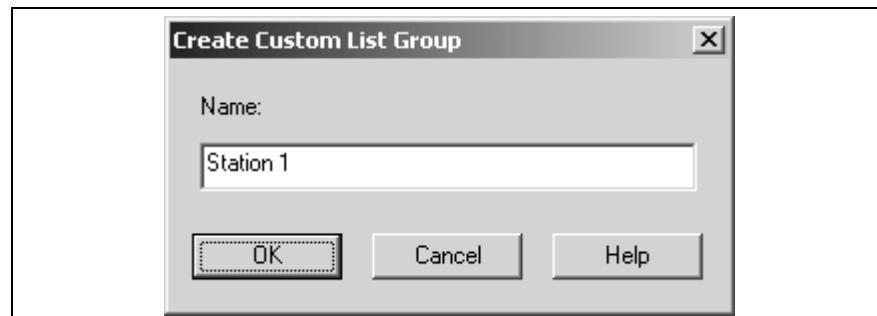
To create a custom list group, use the following steps:

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".

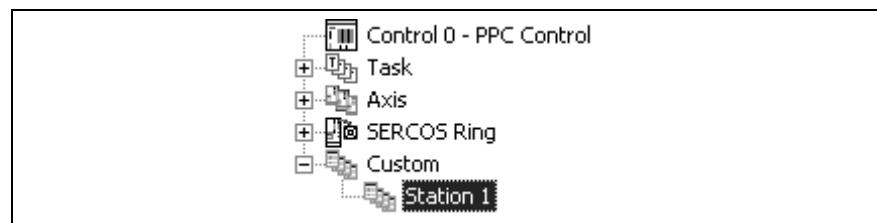
Note: In both cases, the **Custom** tree icon must first be highlighted before the selections are made available.



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.



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.

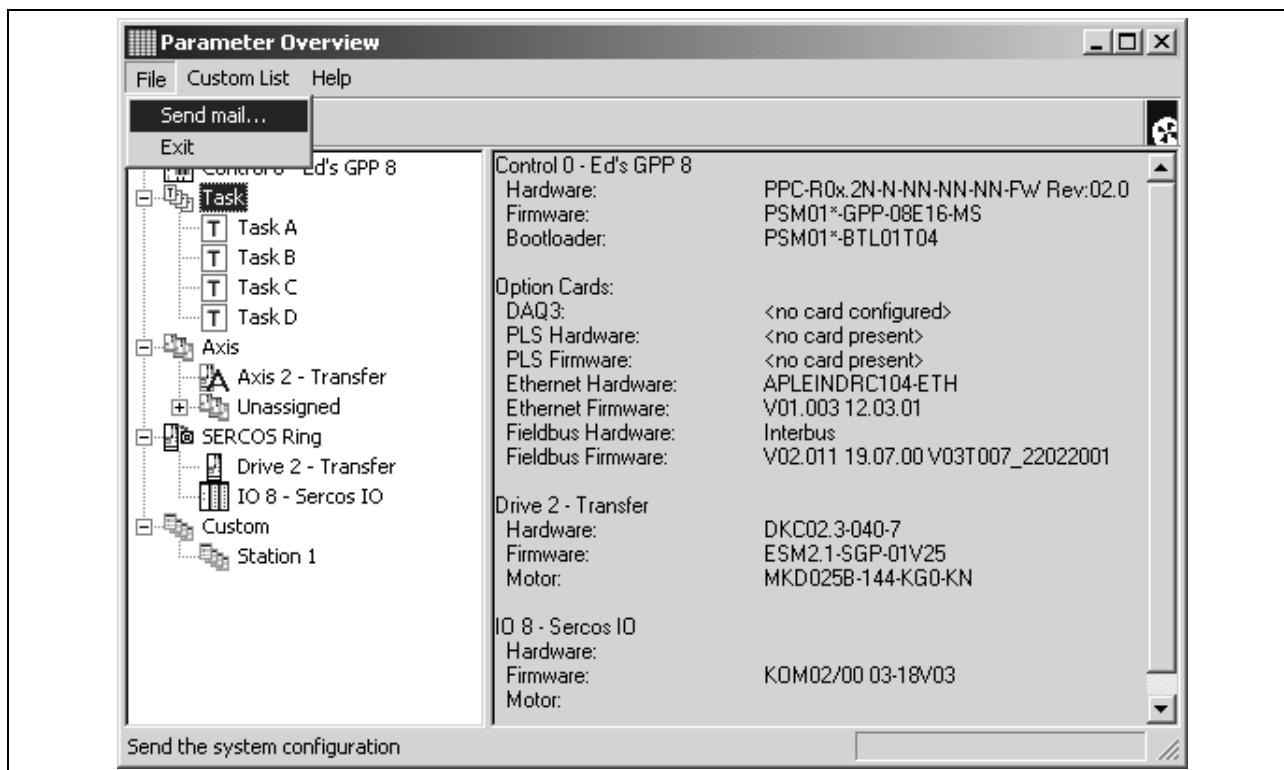


Fig. 7-67: System Configuration

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.

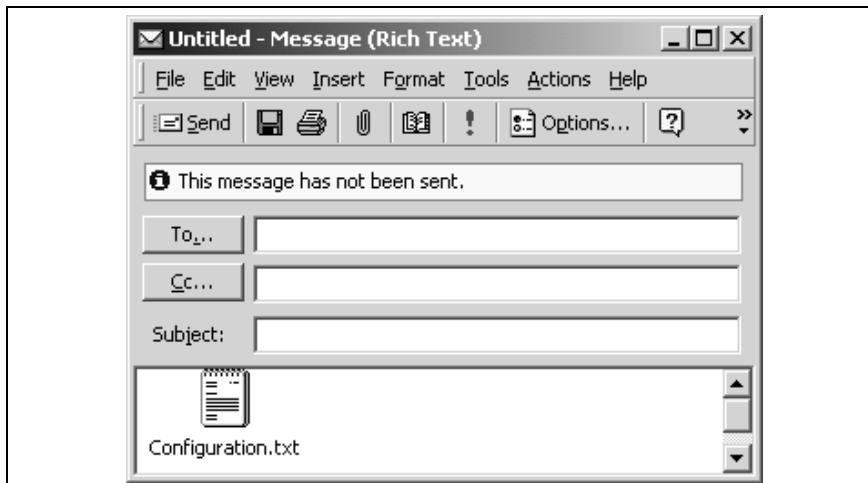


Fig. 7-68: Send Mail

Note: The **Send mail...** option will not be visible if the connected PC does not have a configured e-mail service.

Registers

Selecting **On-Line Data** \Rightarrow **Registers** opens the *Registers* window below. This Windows utility allows viewing and editing of registers on the control. Refer to the [Control and Status Registers](#) for a description of the registers reserved by the control for system control and status.

Registers can be changed by editing the register, editing the register's bits, or forcing the register bits. Register priority is as follows from highest to lowest:

- PC Access - registers written directly by an external device.
- I/O Drivers - registers control by I/O device drivers.
- Register Forcing - via serial port.
- I/O Mapper - via I/O Mapper Boolean equations.
- Direct Register Access - via serial port or program instructions.

The Physical Name of the registers in the list box are taken from register labels, register labels can be assigned to non-system registers.

Choosing the Registers menu item from the Data menu opens a Registers window listing the register ID numbers and the symbolic label names associated with the registers (from the register.lst file). When the window opens, it automatically uploads and displays the current contents of the control registers.

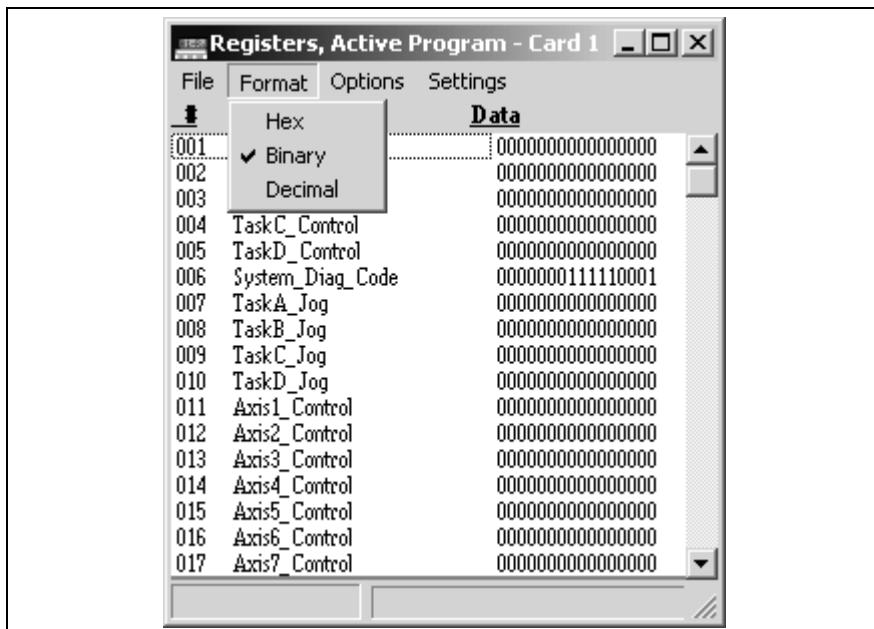


Fig. 7-69: Registers

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

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

Options - Shows the available F-key commands for editing, forcing and clearing.

Options ⇒ Edit Bits... F2

Selecting a list entry and pressing **F2**, or simply double clicking on a list entry, opens a Bit Names window for the selected register and displays the bit names and values associated with the register. Bit names are obtained from the "bit.lst" file in the Host system :\\VisualMotion 8\\project sub-directory and are not uploaded from the control.

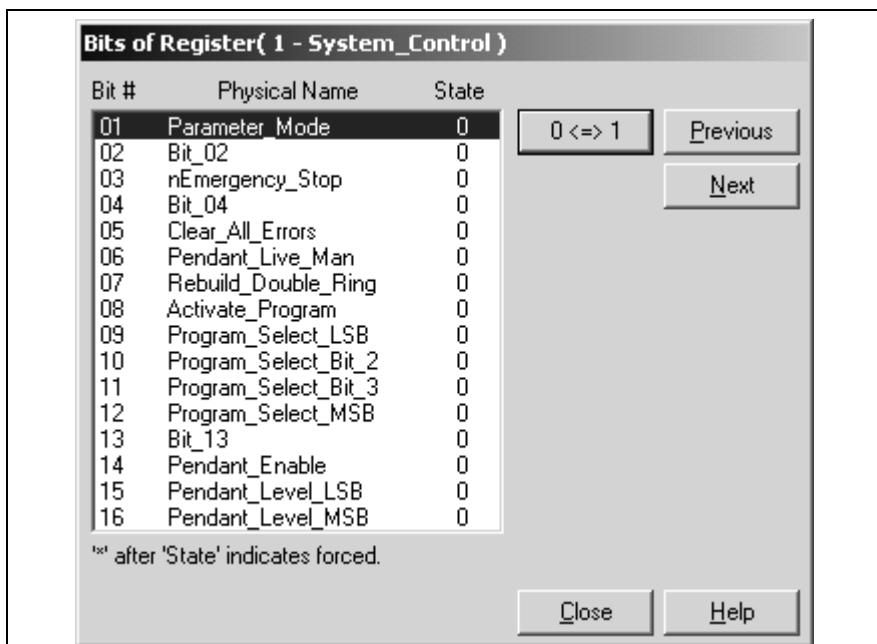


Fig. 7-70: Bits of Registers

Selecting a list item and clicking on the "0 <=> 1" button inverts the current state of the bit and downloads the change of state to the control through the serial communication link. The next periodic upload from the control reflects the change of state in the displayed list.

Bits controlled by the I/O, user program, or I/O Mapper will not be altered using the "0 <=> 1" toggle button. However, their state may be "forced" using the Forcing option (F4). Items denoted with an "x" in the rightmost column have forcing in effect for the associated bit.

Options ⇒ Register... F3

Selecting a list entry and pressing **F3** opens an Edit Reg. window allowing change of the value in the selected I/O register table. Clicking the Save button downloads the changed value to the control.

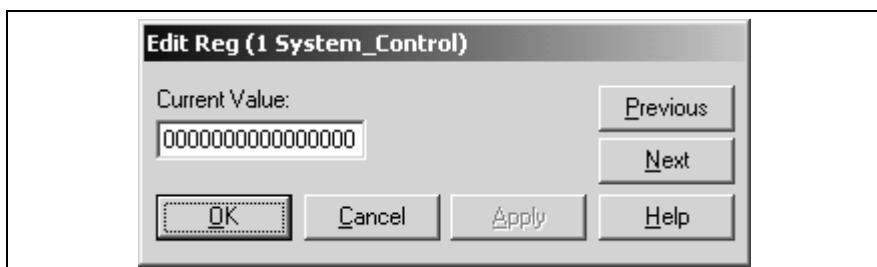


Fig. 7-71: Edit Registers

Options ⇒ Force a Register... F4

System installation and troubleshooting may require directly changing the state of register bits without depending on the I/O sub-system. Selecting a register and pressing **F4** opens a Register Forcing window that allows you to setup a forced bit during system setup and debugging.

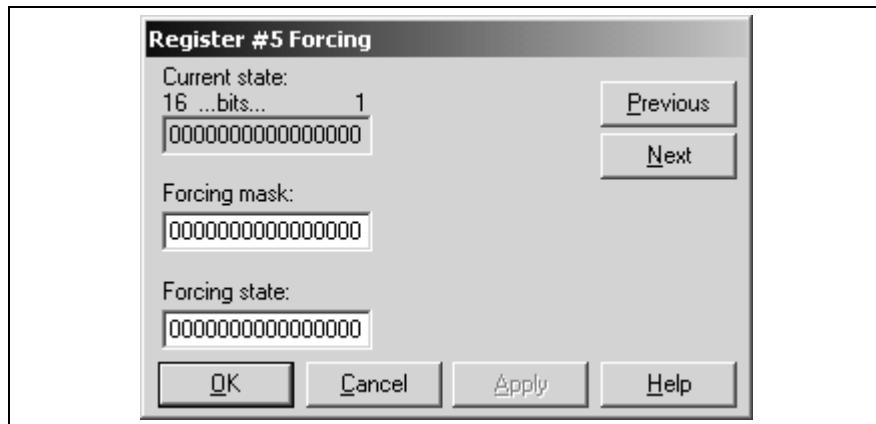


Fig. 7-72: Register Forcing

Forced bits cannot be affected by a VisualMotion 8 program or the I/O sub-system. 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 on-to-off.

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.

The Forcing mask value enables which bits may be affected by forcing. The Forcing state value determines the actual state of the enabled bits. The following algorithm mathematically describes how the final register state is set:

New register state = (old state & inverted forcing mask) | (forcing state & forcing mask)

Example:

forcing mask	0000 0000 0000 1000
forcing state	0000 0000 0000 1000
inverted forcing mask	1111 1111 1111 0111
old register state	0000 1101 1010 0001
new register state	0000 1101 1010 1001



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.

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

Variables

Selecting **On-Line Data** ⇒ **Variables** opens the *Variables* window below. The Variables window permits viewing the integer and floating point variables of a control resident program. The window automatically uploads and displays variables of the active program on the control. The list display and variable selection for editing is similar to the View Point Table window. The program is selected with **File/Program Select** and the variable type is selected under the **Types** menu.

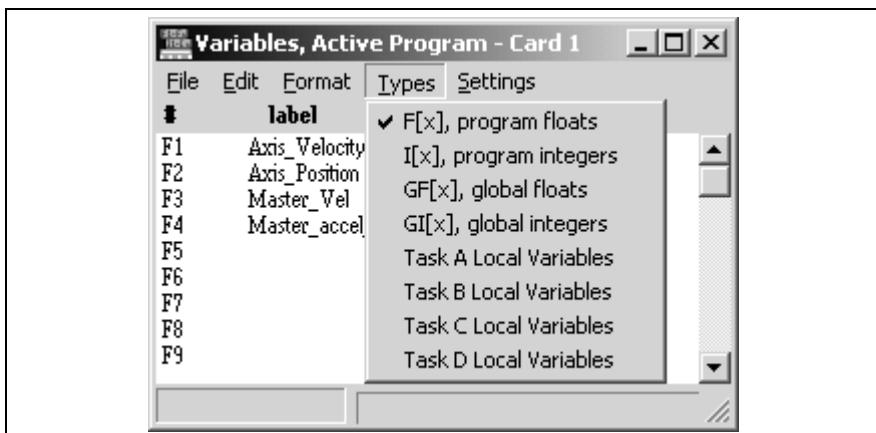


Fig. 7-73: Variable

Selecting a list entry and clicking the **Edit** button, or double clicking the list entry, opens a Variable Edit window allowing change of the value in the selected variable in the VisualMotion 8 resident program. Exiting the window automatically downloads the changed values to the control.

Events

Selecting **On-Line Data** ⇒ **Events** opens the *Events* window below. 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 View Events Table 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. Up to ten events are allowed for each VisualMotion program.

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

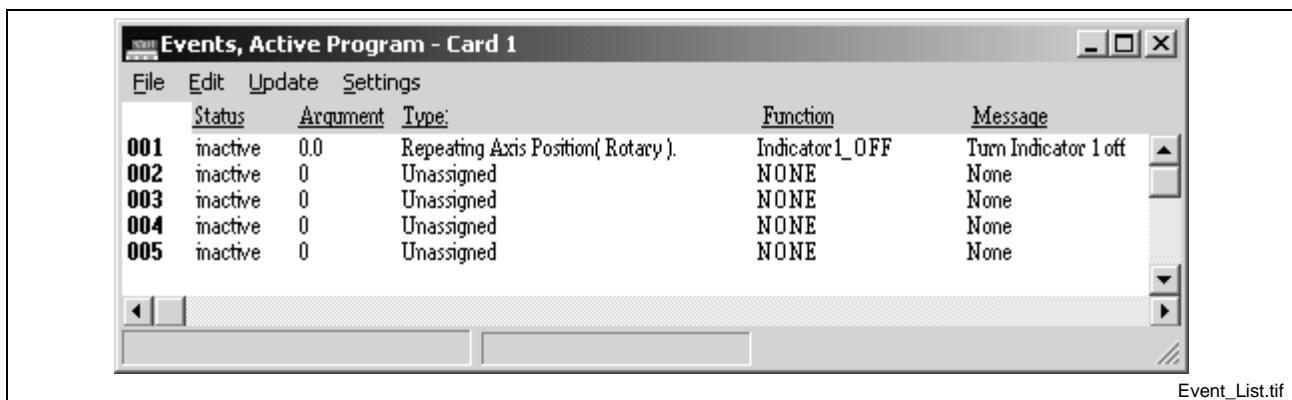


Fig. 7-74: Event Runtime Utility

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.

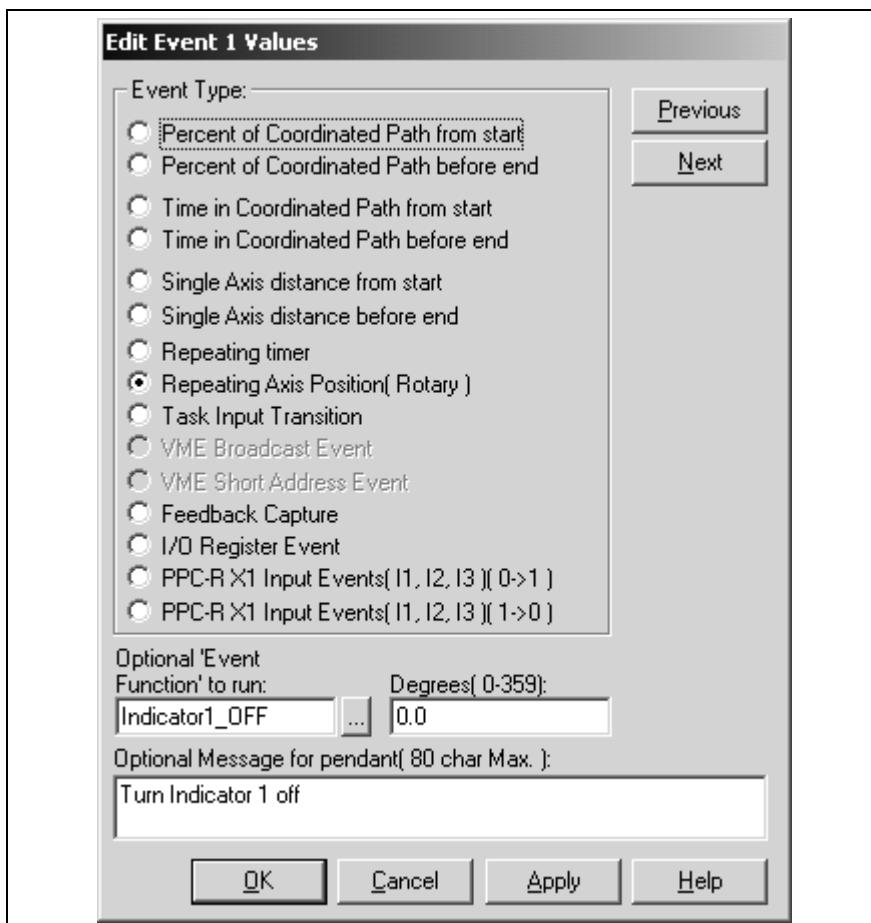


Fig. 7-75: Edit Event Window

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 or time on the path.

- **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 may be setup to be triggered by a continuously Repeating Timer or as an angular position on an axis using rotary motion.

- **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 Rexroth Indramat drives to trigger a VisualMotion 8 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.

Points

Selecting **On-Line Data** ⇒ **Points** opens the *Points* window permitting the user to view and edit the absolute and relative points in a program that has already been downloaded to the control.

Note: To use points, you must allocate memory in the VisualMotion **Size Icon**.

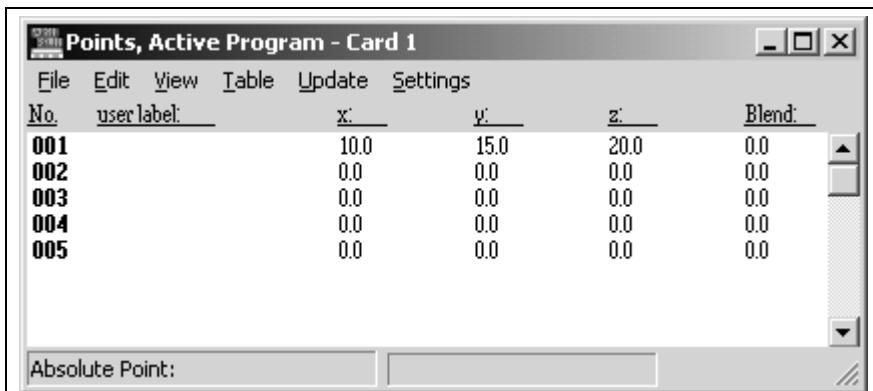


Fig. 7-76: Points Window

The window automatically uploads and displays the contents of the point table for the VisualMotion 8 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 **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.

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.

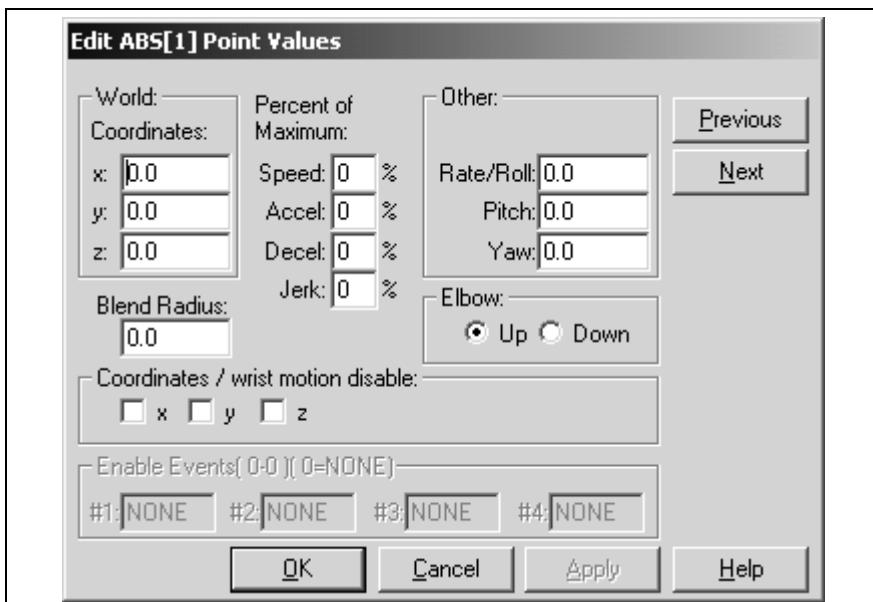


Fig. 7-77: Edit Points

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.

View Menu

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 Menu

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 *Card Selection Setup* window for the DDE-Server settings.

CAM Indexer

Selecting **On-Line Data** \Rightarrow **Cam Indexer** opens the *CAM Indexer Runtime Utility* window below:

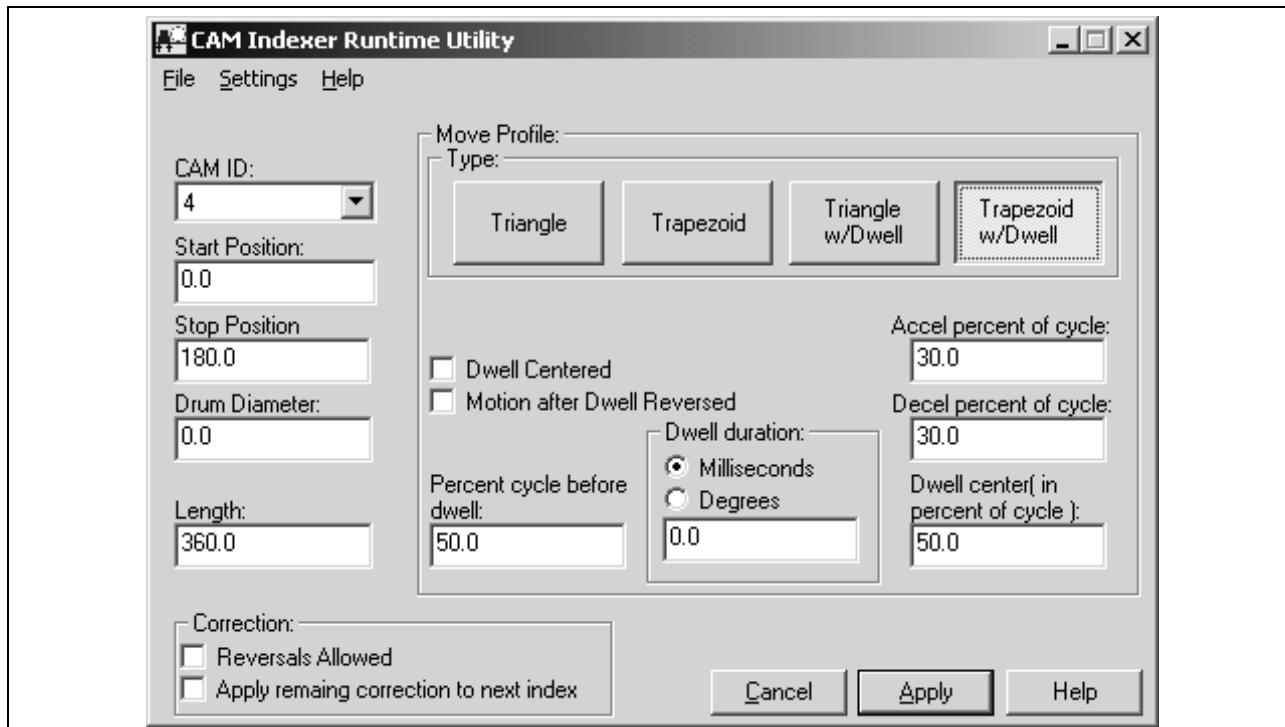


Fig. 7-78: CAM Indexer Runtime Utility

This utility allows for making on-the-fly changes to Cam Indexers in the active program that were 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 **On-Line Data** ⇒ **ELS** opens the **ELS Runtime Utility** window below:

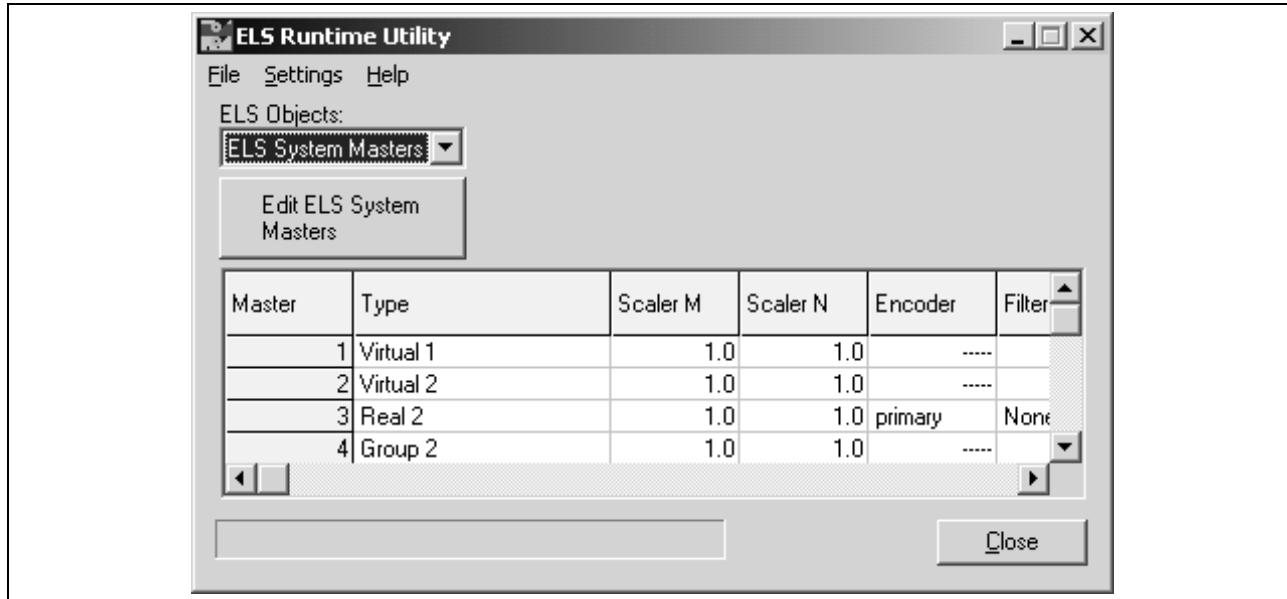


Fig. 7-79: ELS Runtime Utility

This utility is a tool designed for modifying default program variables. 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 initially compiled and downloaded to the GPP overwrites the default program variables on the GPP. These 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** scroll box at the top left allows selection of any ELS Masters, ELS Groups or Virtual Masters that have been setup. The button below the scroll 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.

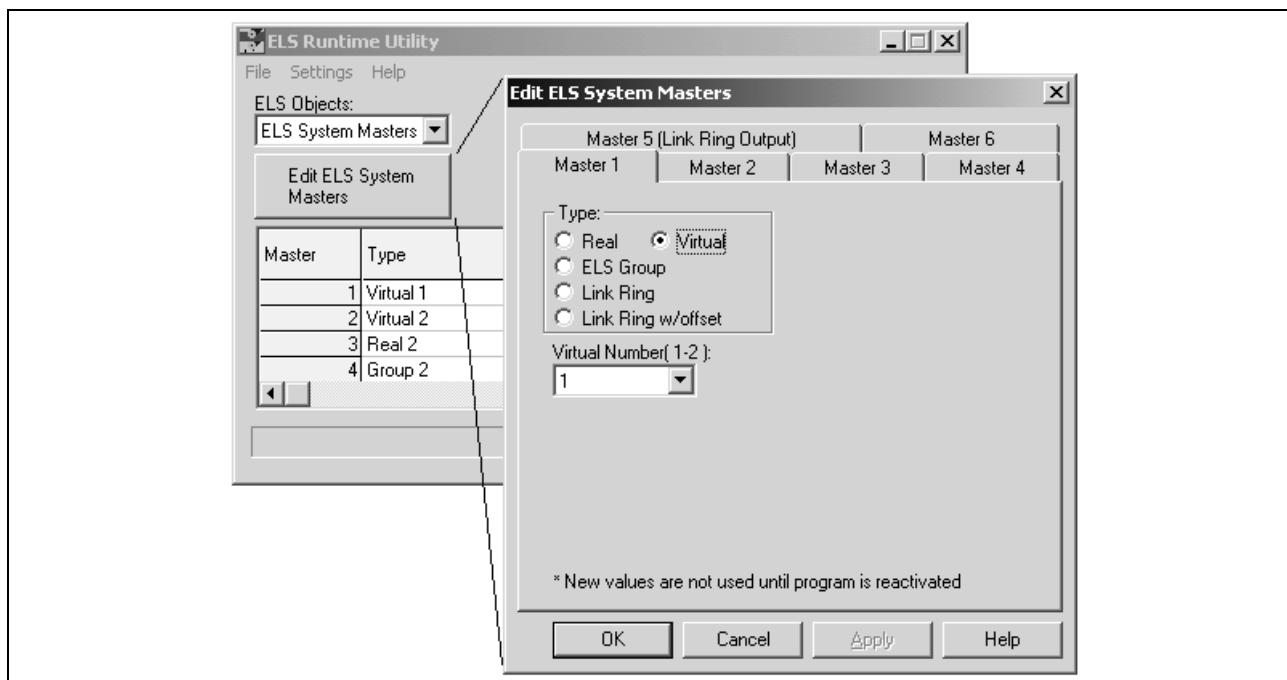


Fig. 7-80: Edit ELS System Masters

For more information about the initial setup of system masters and setting the values in this window, refer to the **ELSMstr** icon in the section titled **Icon Programming**.

ELS Group

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

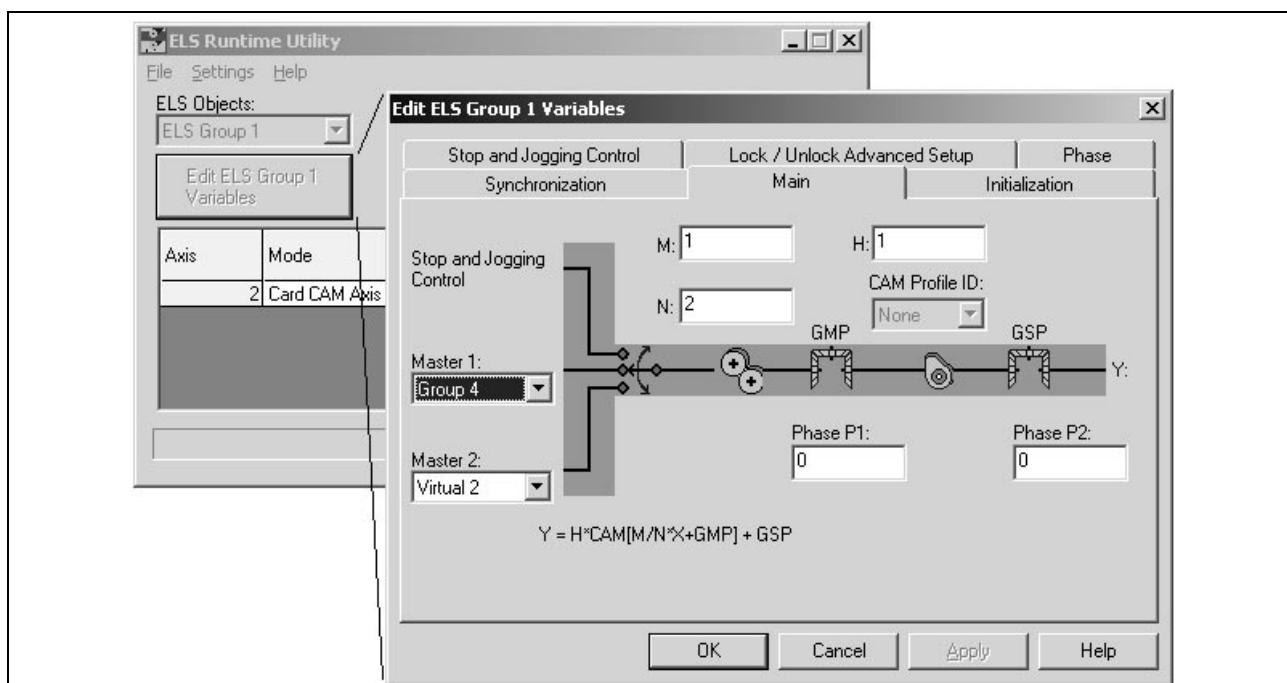


Fig. 7-81: Edit ELS Group x Variables

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.

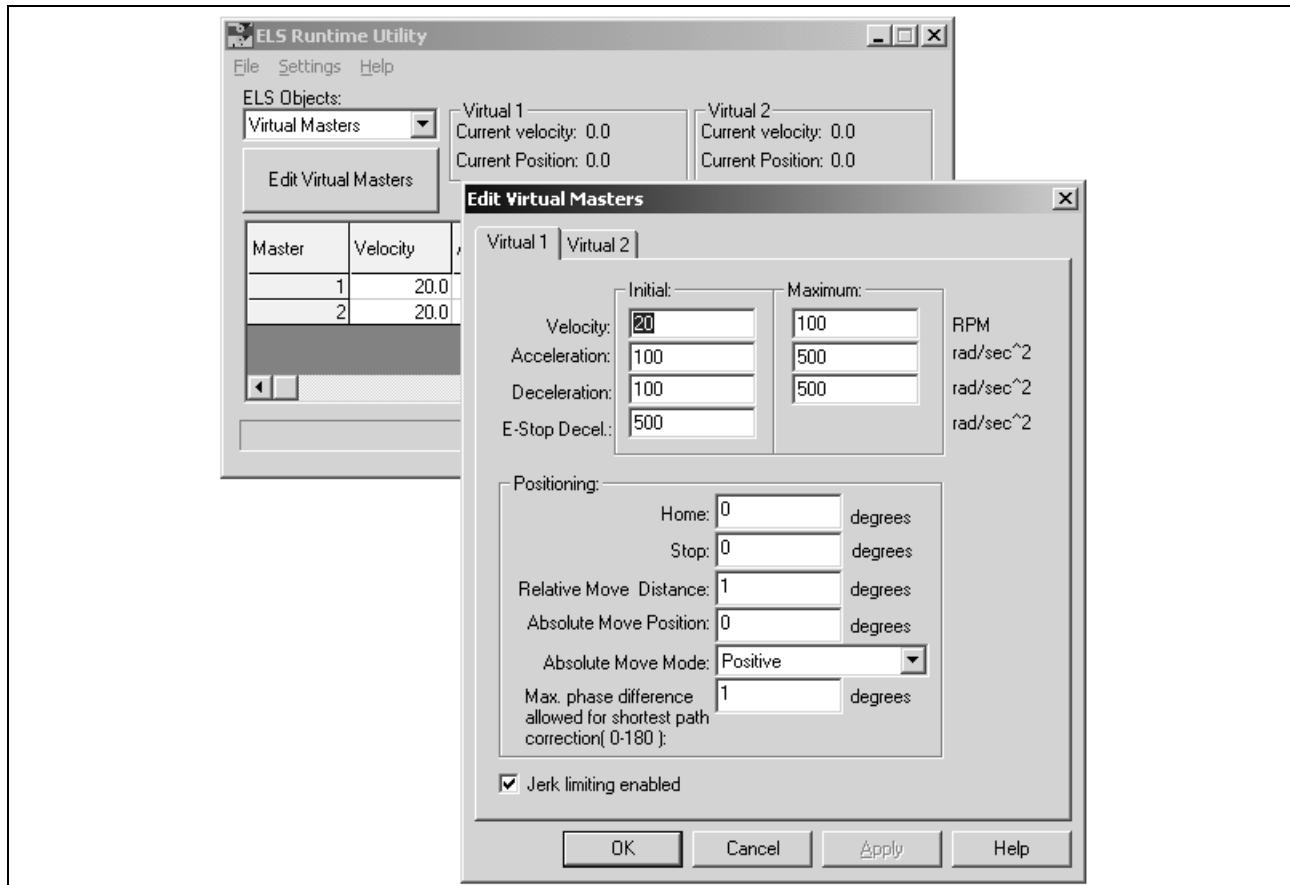


Fig. 7-82: Edit Virtual Masters

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**.

PID

Selecting **On-Line Data** ⇒ **PID** opens the *PID Monitor* window below. This menu item spawns a window for monitoring and tuning the PID of the active program on the control.

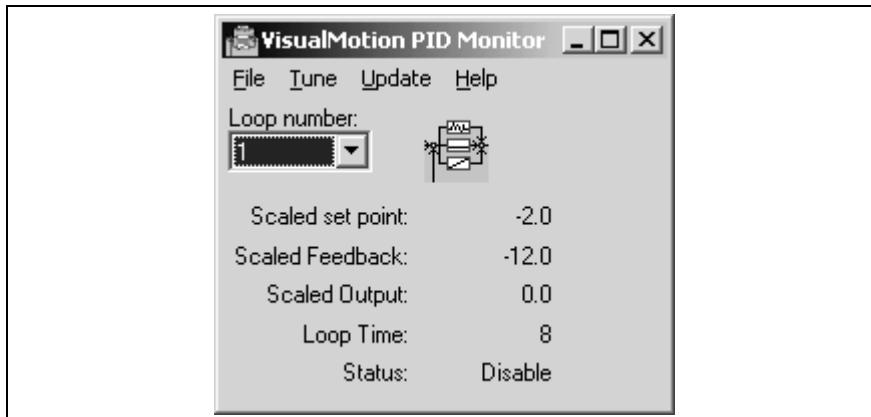


Fig. 7-83: PID Monitor Window

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.

Note: The available number of PID loop is dependent upon the VisualMotion software release.

VisualMotion release 08V08 and earlier:

Up to 10, PID loops can be configured.

VisualMotion release 08V09 and later:

Up to 32, PID loops can be configured.

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 ().

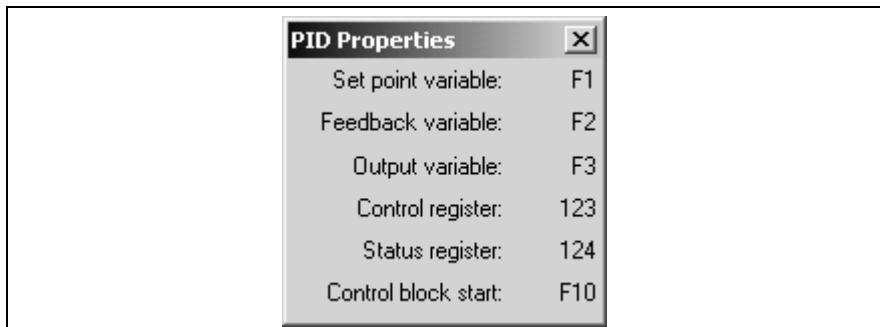


Fig. 7-84: PID Properties

Tune PID Control Block

Selecting *Tune* opens the PID Properties window below. 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.

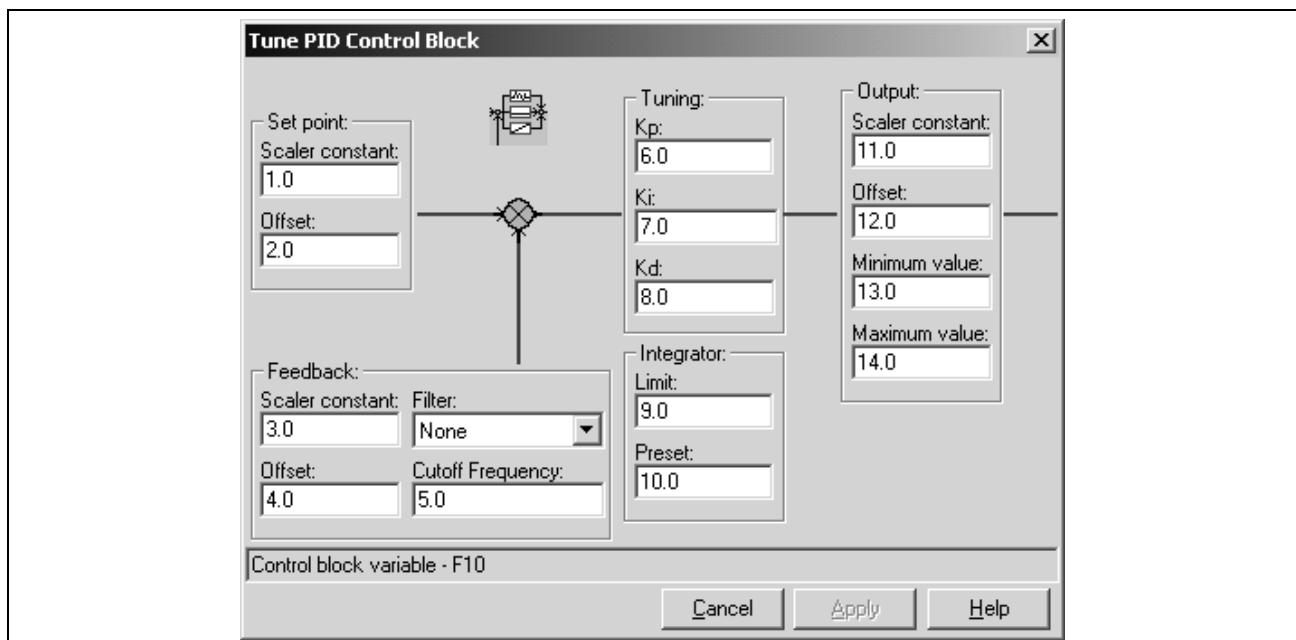


Fig. 7-85: Tuning PID

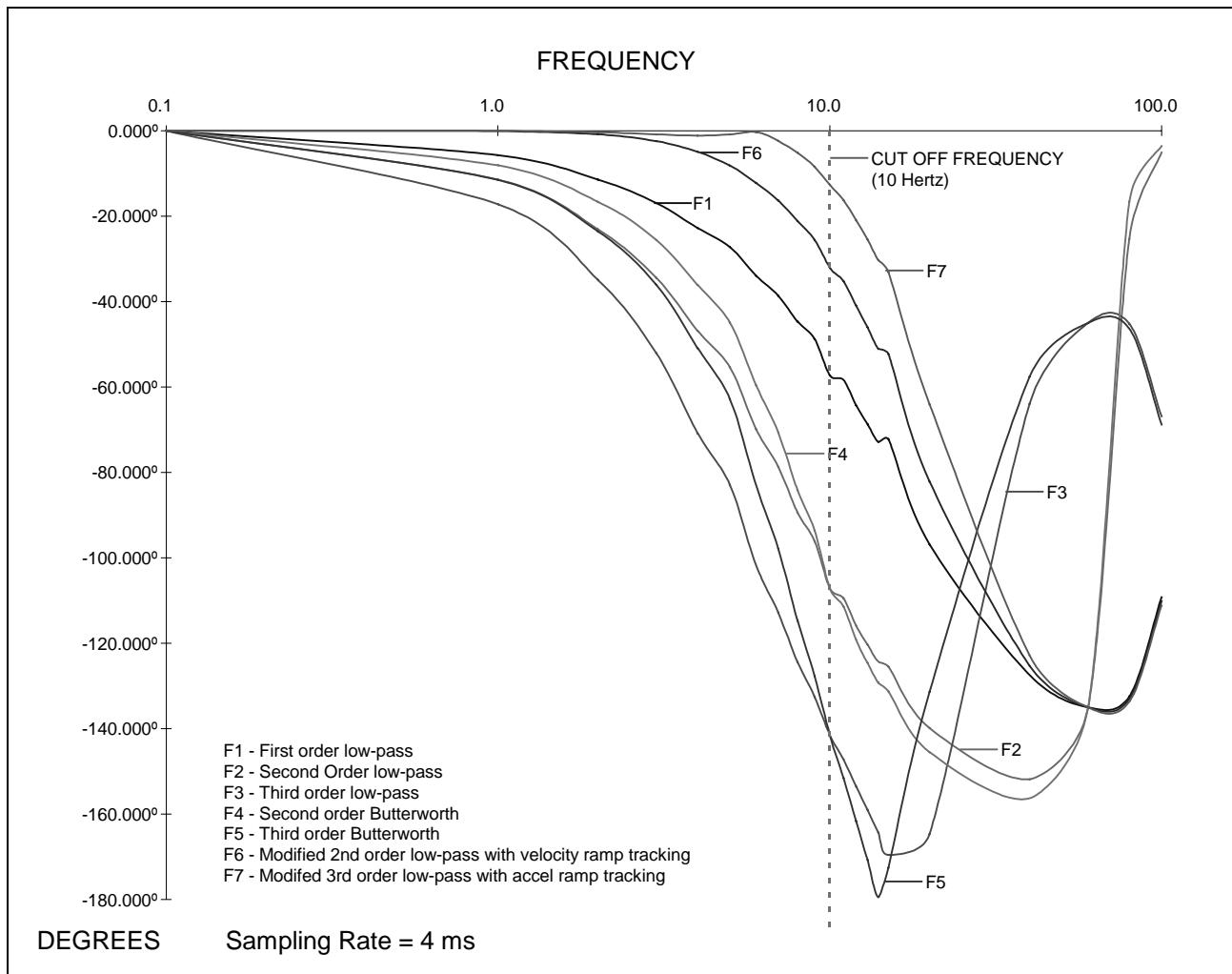
A grouping adjusting the Kd, Kp, Ki, integral preset, and integral limit are provided. The output grouping also has min, max limits for it. The feedback has an optional digital filter to condition the signal.

The feedback filter choices are:

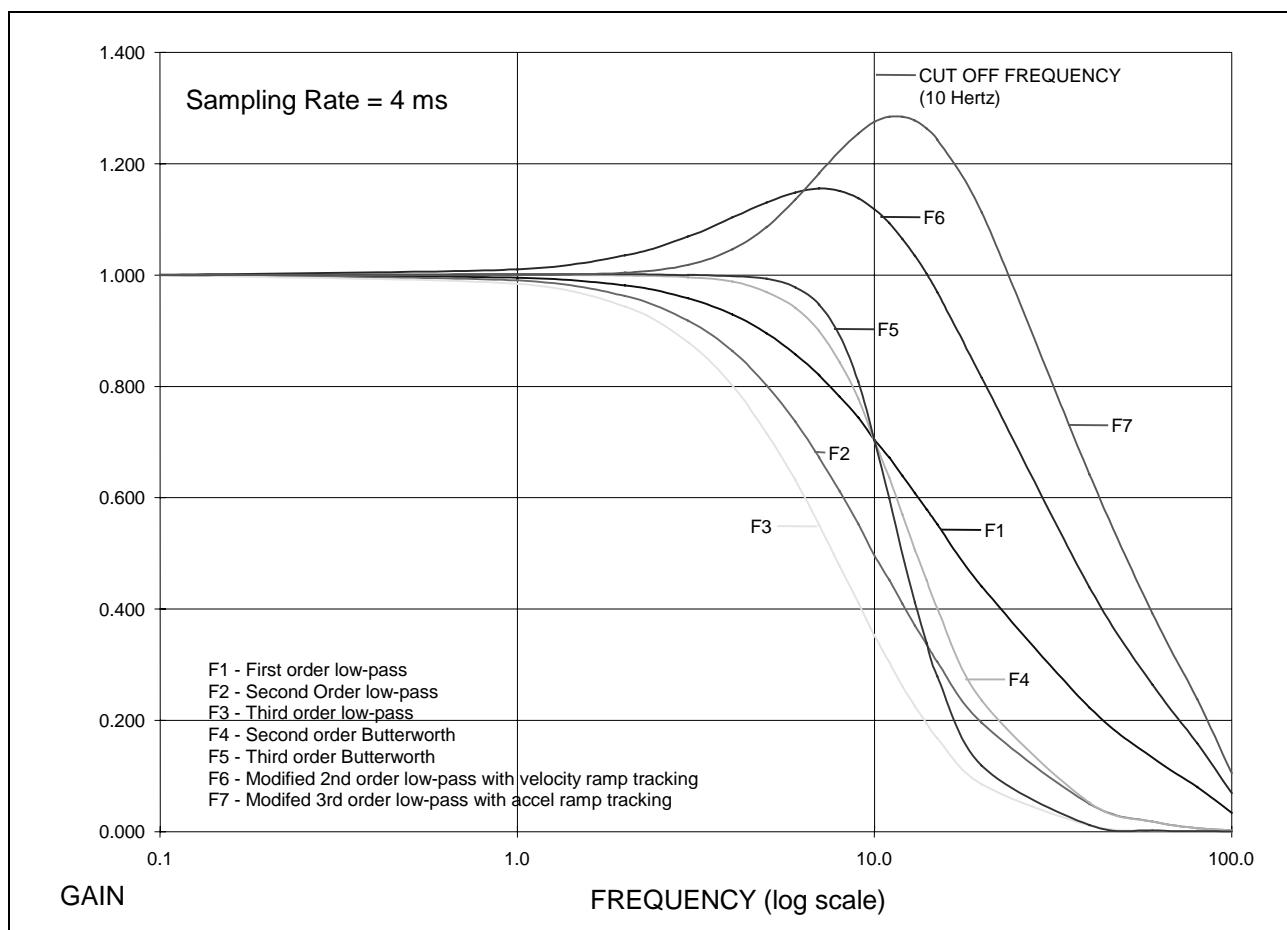
- None
- First order low-pass, $G(s)=1/(s+1)$
- Second order low-pass, $G(s)=1/(s^2 +2s +1)$
- Third order low-pass, $G(s)=1/(s^3 +3s^2 +3s+1)$
- Second order Butterworth, $G(s)=1/(s^2 +2^{1/2}s +1)$

- 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, $G(s)=(3s^2+3s+1)/(s^3+3s^2+3s+1)$

The following charts illustrate the different filter types.



The chart above shows the frequency vs. degrees for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

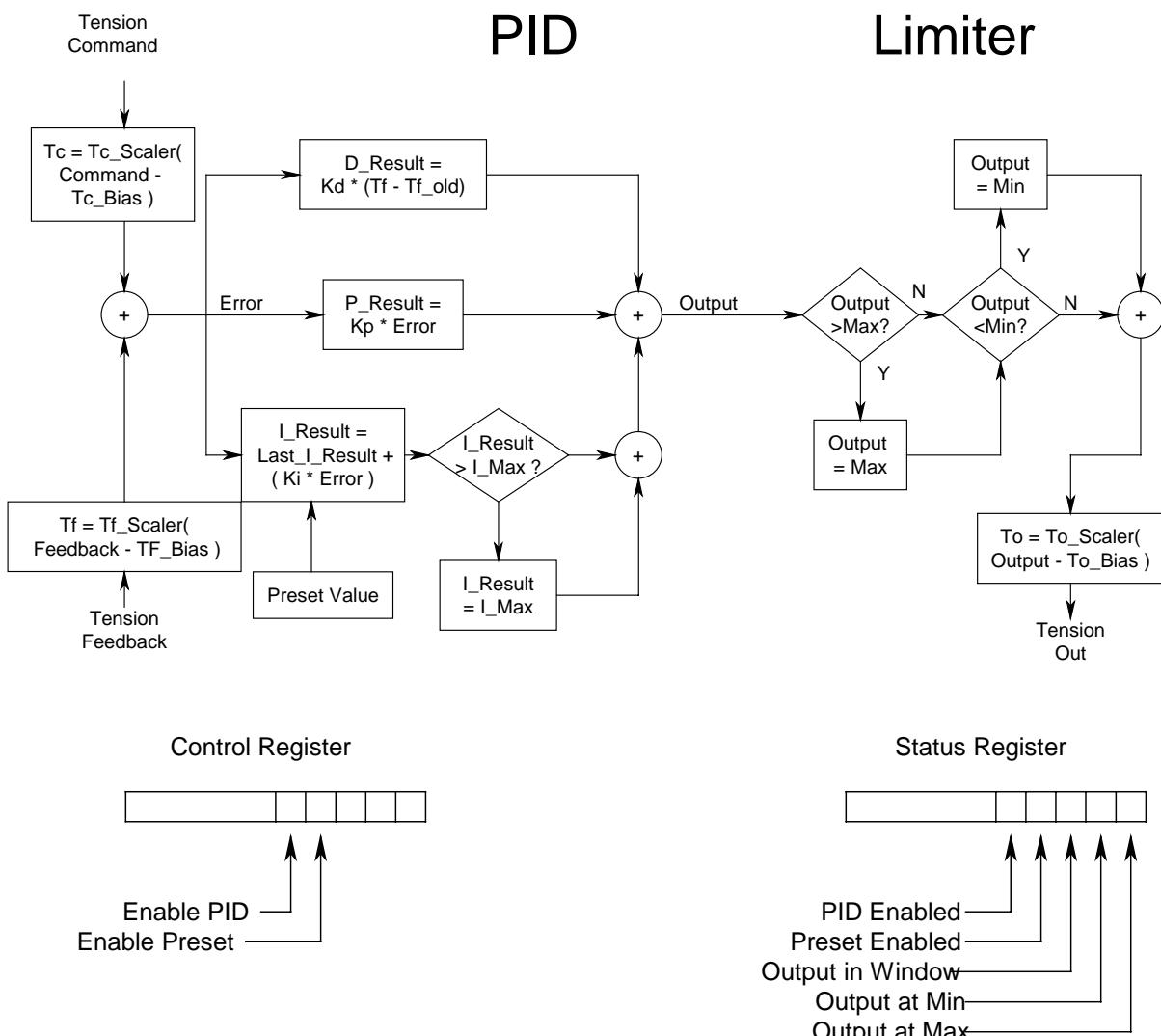


The chart above shows the gain vs. frequency for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

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

F20	Command scaler value, default 1.0.	4
1	Command bias value, default 0.0.	4
2	Feedback scaler value, default 1.0.	4
3	Feedback bias value, default 0.0.	4
4	Kp value, default 1.0.	4
5	Ki value, default 0.0.	4
6	Kd value, default 0.0.	4
7	Ki limit value, default 0.0.	4
8	Minimum output value, default -10.0.	4
9	Maximum output value, default 10.0.	4
F30	Preset value, default 0.0.	4
1	Output scaler value, default 1.0.	4
2	Output bias value , default 0.0.	4
3	Feedback cutoff frequency (Hz), default 0	4
4	Feedback filter type, default 0	4
	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	
5	reserved	4
6	reserved	4
7	reserved	4
8	reserved	4
9	reserved	4

Registration

Registration is the process of referencing a product's edge 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 positionally corrected. 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 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 4 axes can use the registration function.

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*. 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 Indramat drive firmware versions:

DIAX04 (ELS-05VRS or later)

ECODRIVE03 (SGP-01VRS or later)

Selecting **On-Line Data ⇒ Registration** in VisualMotion Toolkit opens the Registration runtime utility window. Refer to the **Registration** icon for more information about the settings in this window.

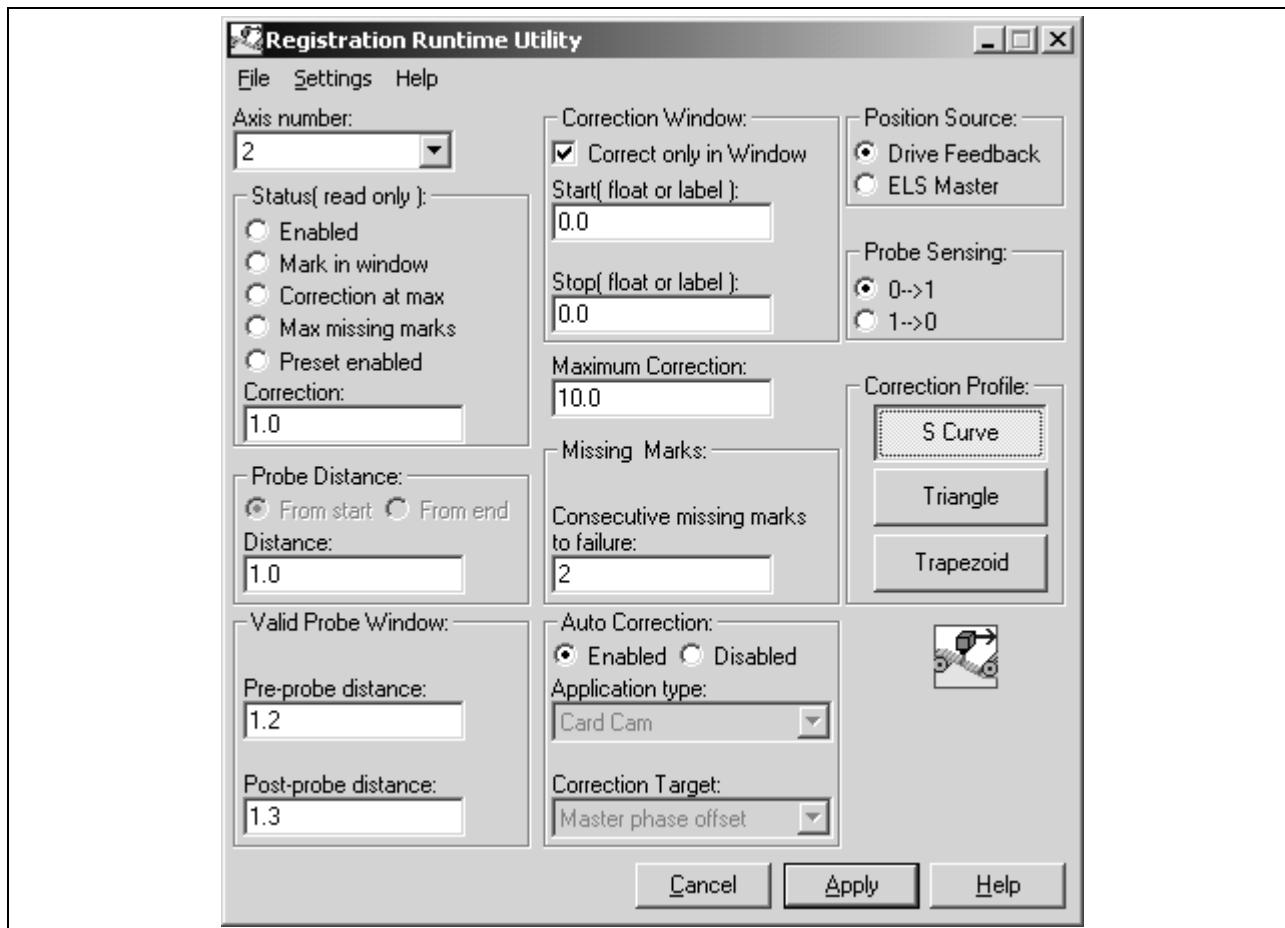


Fig. 7-86: Registration Utility

Note: The Auto Correction functionality for *Application type* and *Correction Target* is in development.

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).

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 are 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:

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

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 below. 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.

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.

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. 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-0-0157)
0	1	invalid	ratio fine adjust (P-0-0083)	slave phase offset (S-0-0048)	slave phase offset (A-0-0162)
1	0	invalid	invalid	cam shaft distance (S-0-0093)	Cam H factor (A-0-0033)

Position Source

The user select the source to be used for positional data. The selections are either a Drive Feedback or an ELS Master.

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.

Conveyor Tracker

Selecting **On-Line Data** ⇒ **Conveyor Tracker** opens the Conveyor runtime tool below. Its purpose is to assist the operator in the setup, operation and management of a robotic conveyor tracking system.

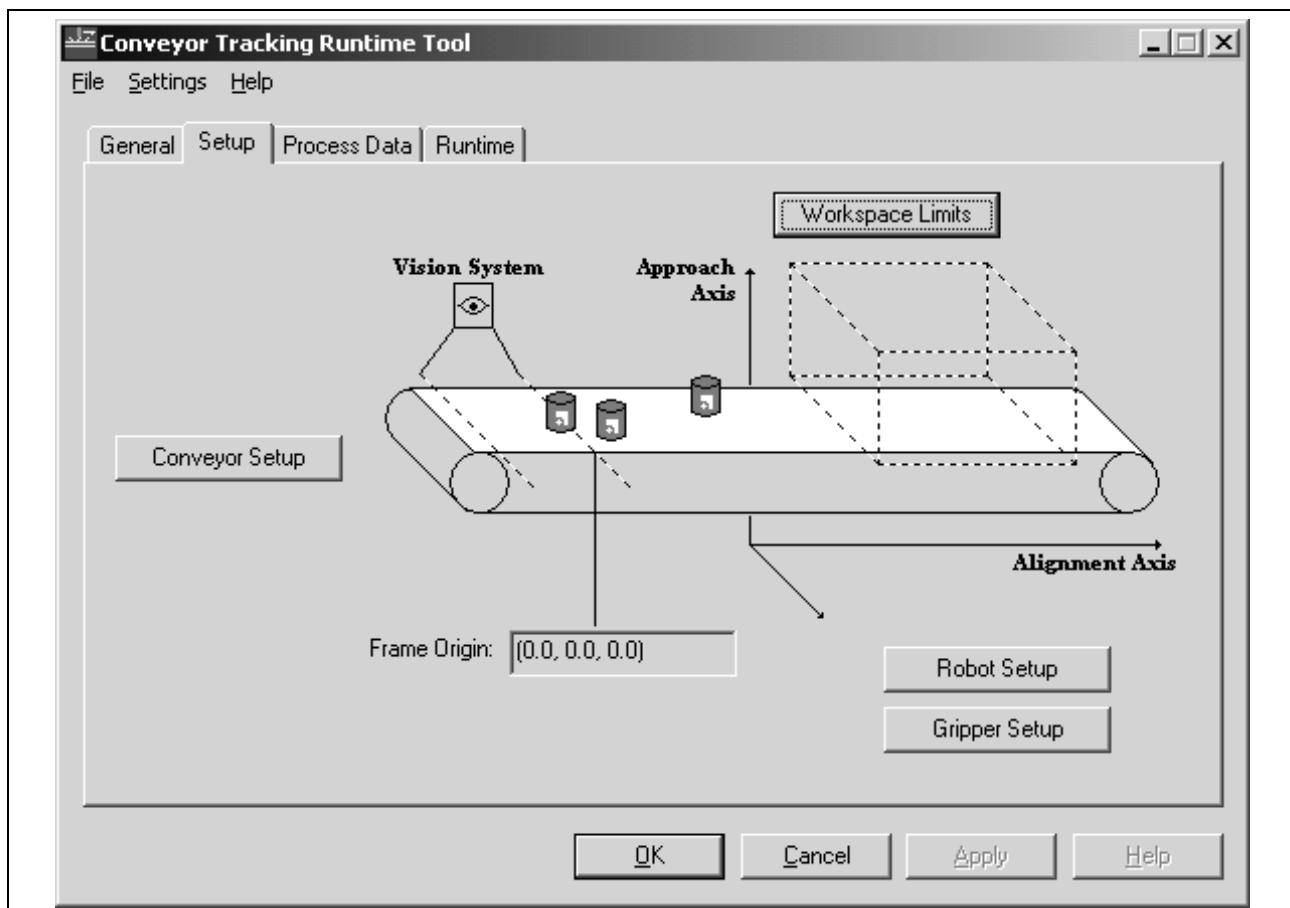


Fig. 7-87: Conveyor Tracking Runtime Tool

Note: The Conveyor Tracker tool requires the following special control firmware in order to function, otherwise an error window will be displayed.

FWC-PSM01*-GP*-98VRS-MS-FLASH

This tool displays fields for all integer and float block variables. Any field that may be changed is displayed with a white background. Read only fields are displayed as grayed out. Read/write fields that change to read only fields in phase 4 are grayed out in phase 4. A white background colored field may be changed by placing the cursor within it and typing a new value then pressing the enter key. Fields that have a discrete number of possible values use a drop-down selection box that lists all the possibilities by name.

This tool displays a graphical view of the conveyor and the objects on it. Different symbols are used to represent different states of the product. Such as: invalid, ready, acquired, workspace limited, move time-limited gripper failed at hover point, gripper failed at pickup point. This view is updated dynamically as product enters and leaves the workspace boundaries.

This tool displays a table view of the conveyor with real world coordinates of the products on it. The following fields of the table are defined: x, y, z,

target type, target status, and product type. The table is updated dynamically as product enters and leaves the workspace boundaries.

Sequencer

Selecting **On-Line Data** ⇒ **Sequencer** opens the Sequencer window below. The Sequencer provides the user with a facility for making machine operational changes without having to edit, recompile and download a program to the control. A Sequencer is a list of steps which contain one or more functions, with up to five arguments per function.

When the Sequencer is selected from the On-Line Data menu, VisualMotion reads the Sequencer information from the control and displays it in the Sequencer editor. This allows the user to view and edit a Sequencer list on-line. An error will occur if there are no Sequencer lists in the active program. A Sequencer List needs to be initialized using the Sequencer icon before it can be edited on-line. See [**Sequencer Icon**](#) for more information.

Sequencer Editor

The initial screen provides a drop-down box, which lists all the Sequencer list names found in the selected program and to the right shows the steps of the selected sequencer.

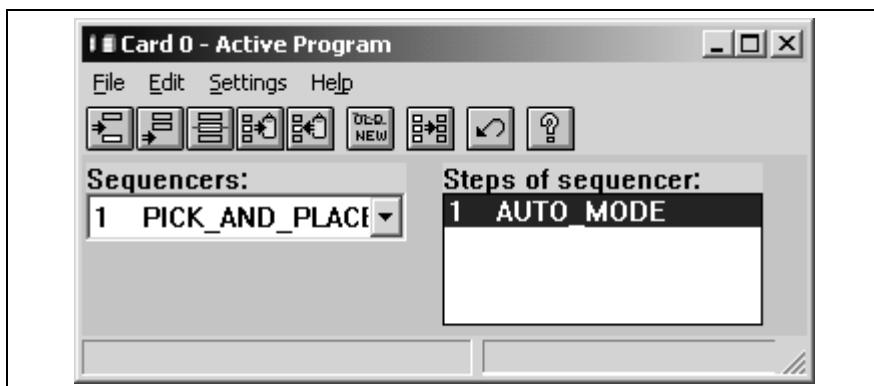


Fig. 7-88: Sequencer

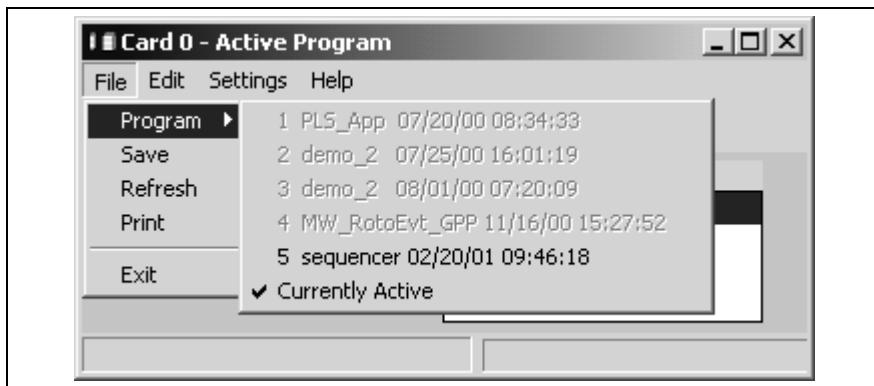


Fig. 7-89: Choosing a Program

Select “Program” under the “File” menu to list programs currently on the control. When a different program is selected, the associated Sequencer tables will be uploaded and the program name will appear in the window caption. The “File” menu also allows the user to save refresh or print the Sequencer table.

The “Edit” menu items allow the user to insert, append, delete, copy, paste and rename steps within a Sequencer list. This editing is also available through icons on the tool bar.



The steps can also be moved by selecting and dragging them with the mouse. Select Step List to show all available Steps.

The **Card Selection Setup** item under the **Settings** menu opens the Card Selection Setup window.

Selecting "Configuration" allows the user to adjust the DDE Synchronous Time-out. This is the amount of time in seconds the Sequencer Editor will wait for a response from the DDE server before proceeding. Refer to [DDE](#) for more information. Sequencer Configuration also allows the user to monitor the Sequence System and/or Active Program for change.

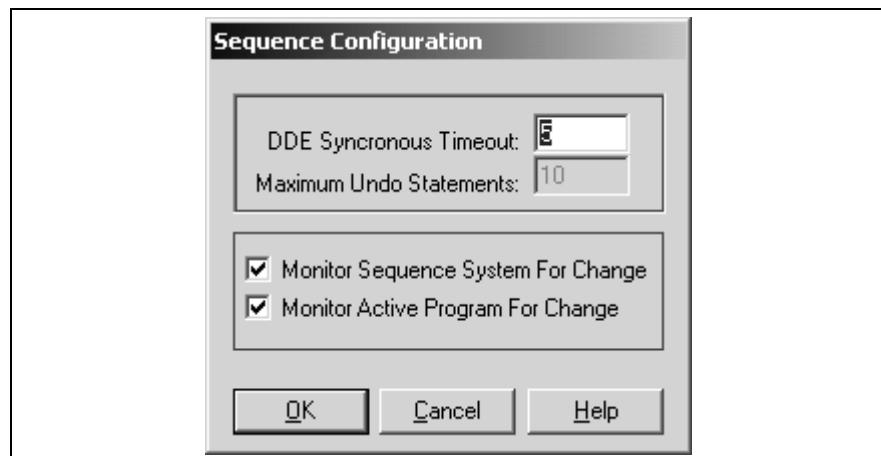


Fig. 7-90: Sequence Configuration

Functions and arguments can be edited by double clicking on a selected step. A Step List window will open with the same edit and icon tools on top. Individual arguments can be edited by double clicking on them. After making any changes, the Step List needs to be saved to retain the current edits in the Sequencer.

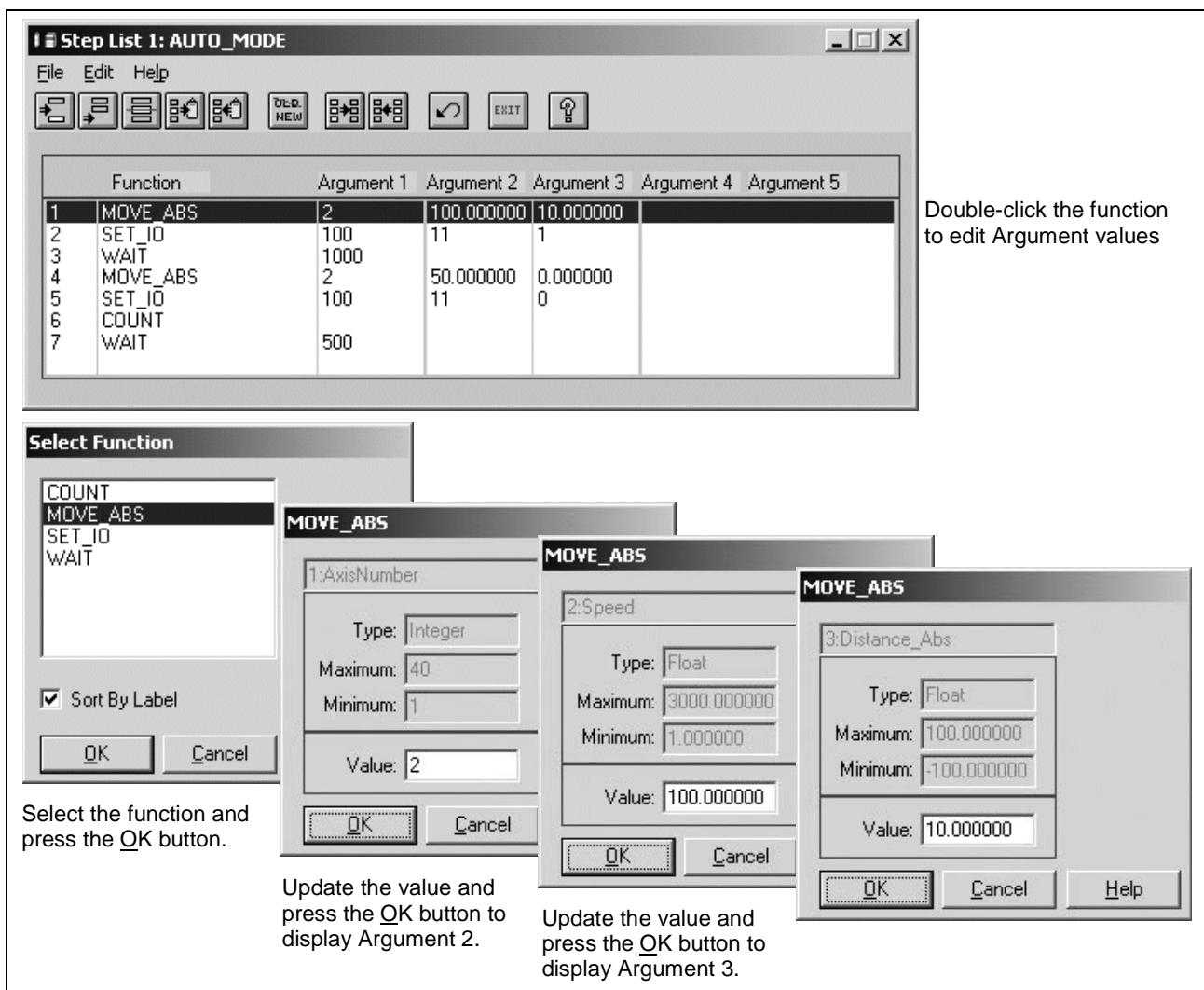


Fig. 7-91: Step List Window

Argument value adjustments can also be sent directly to the control while the program is running by selecting Send. This can only be done if the Step List order hasn't changed since the last time it was saved. If a change has been made to the function order, this option will gray out. A Step List can not be saved to a Sequencer if it is running.

An entire function can be edited or replaced by double clicking on the function name. A list of available functions will open. These are all the predefined subroutines within the VisualMotion program. After selecting a function, the user will have to enter a value for each argument. This value will have to be within the programs predefined limits. Selecting a function this way changes the order of the Step List. The user will not be able to *send* the resulting argument values to the control without first saving the Step List.

Zones

Selecting **On-Line Data** \Rightarrow **Zones** opens the Zones utility. 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 a event, select it in the list box, and press the **Edit** button, or double-click on the list item. In either case, another window will open with an edit field. Change the fields and press the **Save** button to send to control.

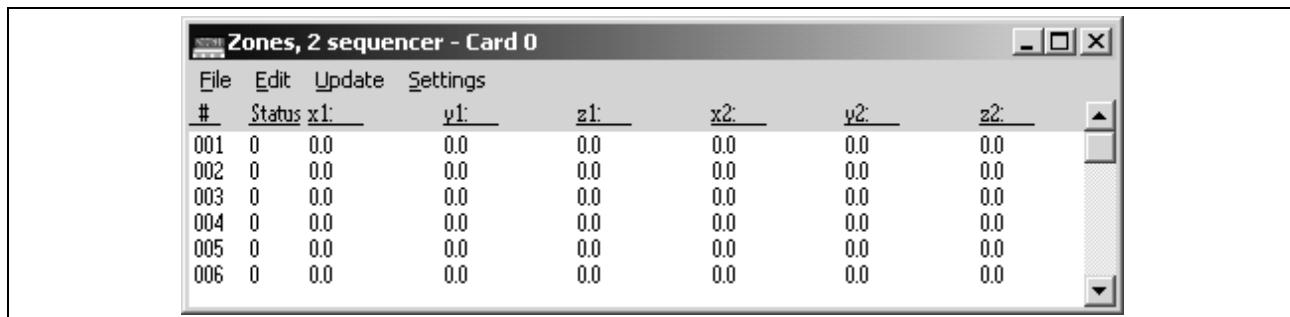


Fig. 7-92: Zones List

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.

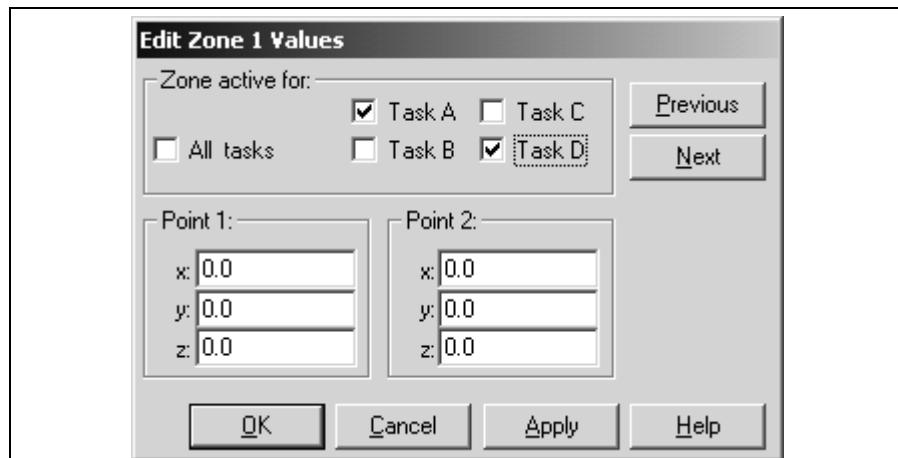


Fig. 7-93: Edit Zone Window

7.8 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, a portion of the program flow by enabling a Break Point or monitor the flow of a running program.

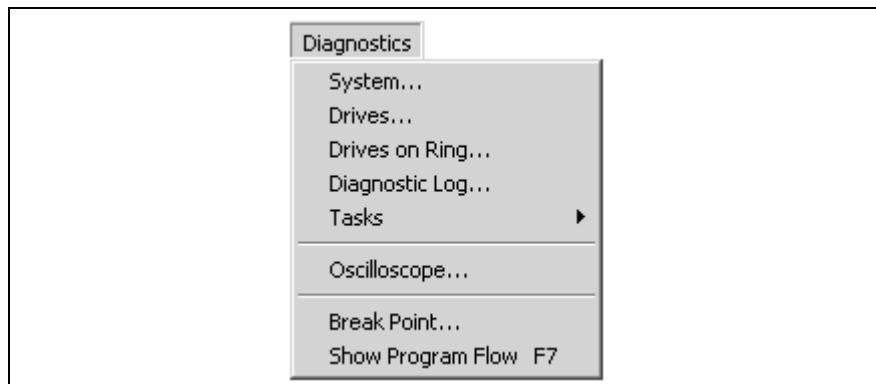


Fig. 7-94: The Diagnostics menu

System

Selecting **Diagnostics** ⇒ **System** opens the *System Parameters* window. System Parameters displays information about the current control hardware and software for the indicated unit number; and the total memory and free memory on the control.

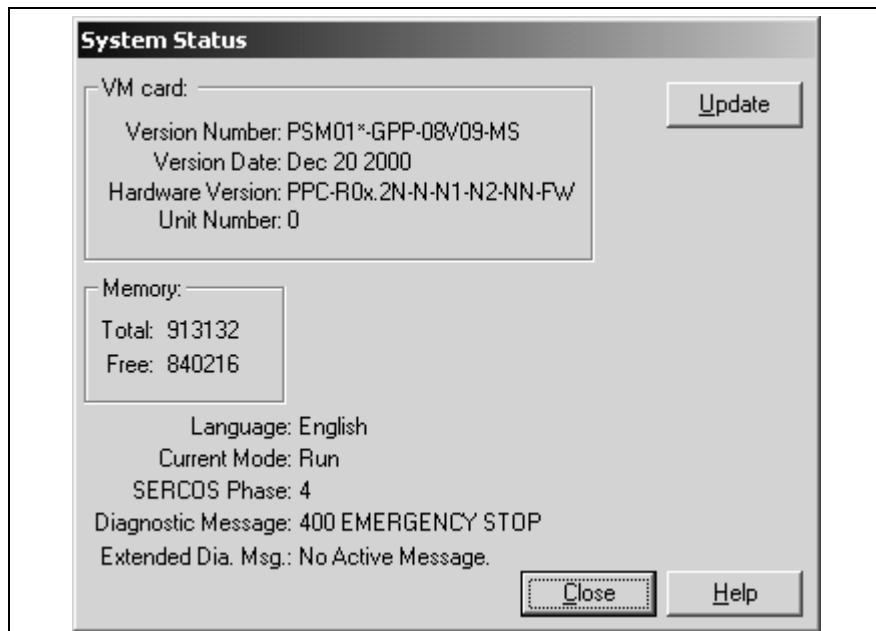


Fig. 7-95: System

Drives

Selecting **Diagnostics** \Rightarrow **Drives** opens the Drive Parameter Editor Window below.

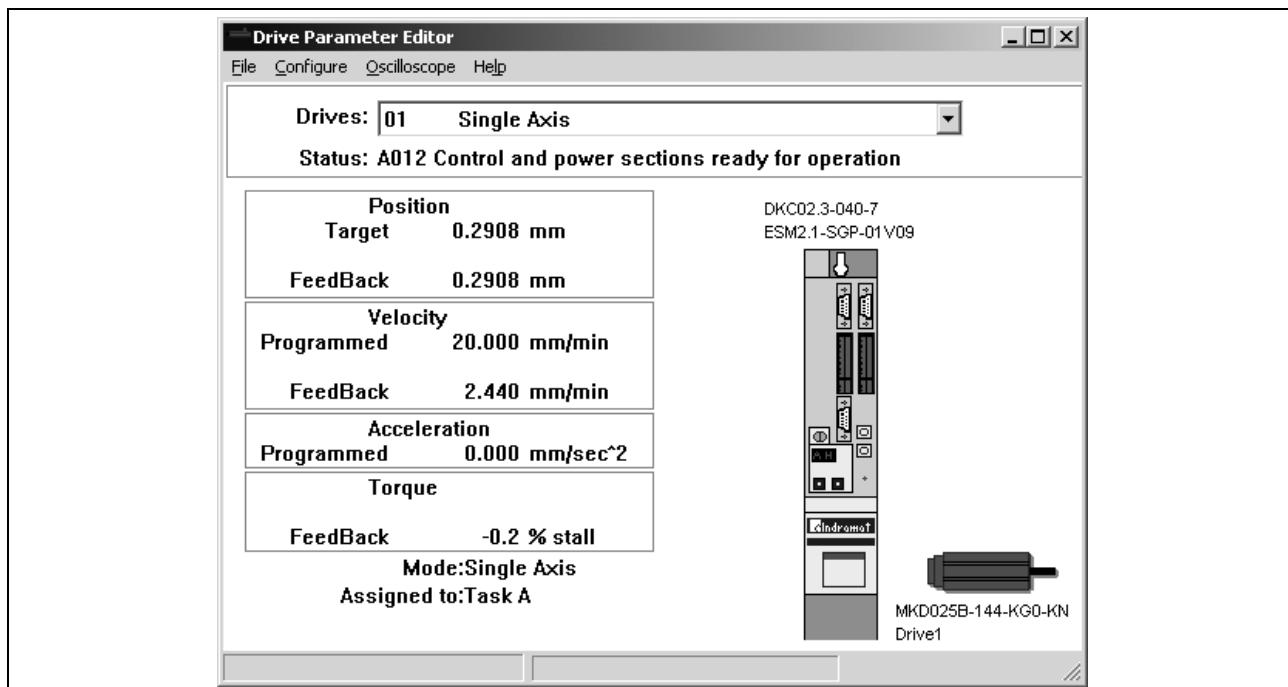


Fig. 7-96: Drive Parameter Editor

For instructions on how to use this utility program, refer to [VisualMotion 8 \(GPP\) Application Manual, Drive Parameter Editor.](#)

Drives on Ring

Selecting **Diagnostics** \Rightarrow **Drives** opens the Visible Drives on SERCOS Ring window. This window displays the addresses, uses and identification of the drives and motors the control sees on the SERCOS ring.

Visible Drives on SERCOS Ring(Read Only)				
#	Used As	Controller Type	Controller Version	Motor Type
01	Single Axis	DKC02.3-040-7	ESM2.1-SGP-01V09	MKD025B-144-KG0-KN
02	Single Axis	DKC02.3-040-7	ESM2.1-SGP-01V15	MKD025B-144-KG0-KN

Fig. 7-97: Drives on Ring

Diagnostic Log

Selecting **Diagnostics** ⇒ **Diagnostic Log** opens the *Diagnostic Log* window. This window lists the last 100 errors the control has encountered. Along with the error messages, the date, time and extended error codes are displayed.

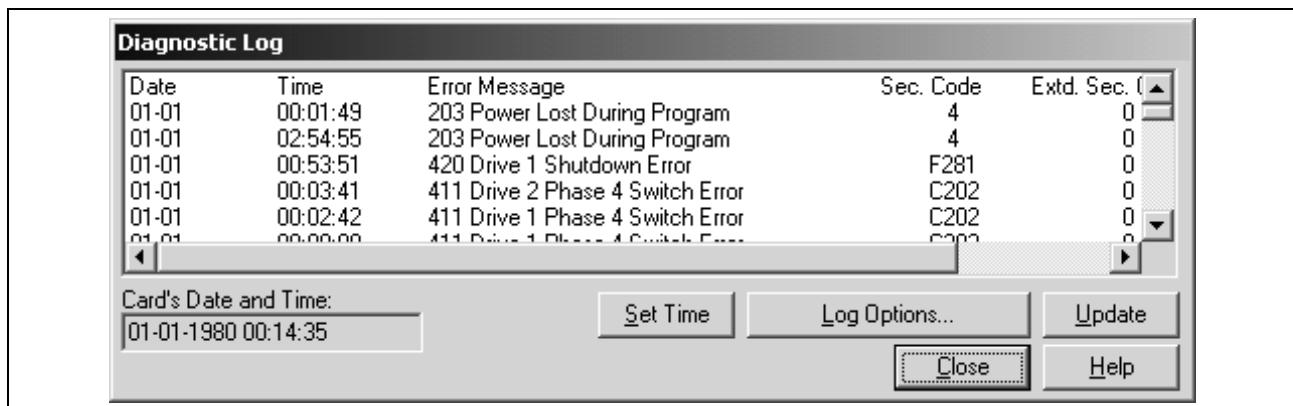


Fig. 7-98: Diagnostic Log

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

The Log Options button opens an options window. It allows some common errors to be ignored and saving the diagnostic log to a file. Also see card parameter [C-0-2020](#).

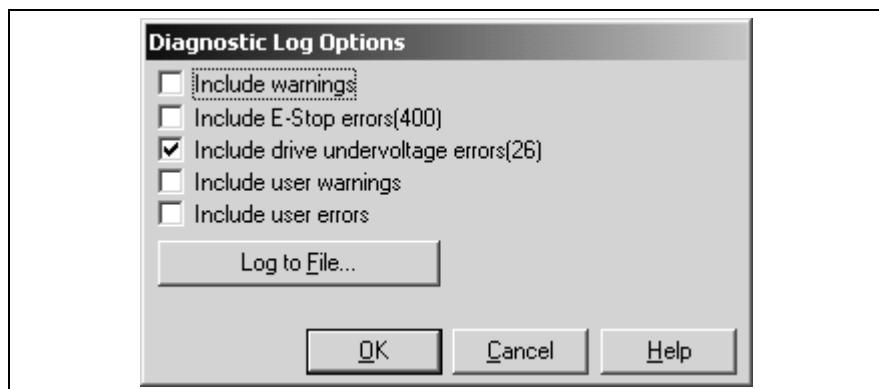


Fig. 7-99: Diagnostic Log Options

Tasks

Selecting **Diagnostics** ⇒ **Task** ⇒ **Task (A, B, C or D)** opens the **Task_x Parameter** window and uploads data regarding the active VisualMotion task. The other tasks may be viewed by clicking on the Previous or Next buttons.

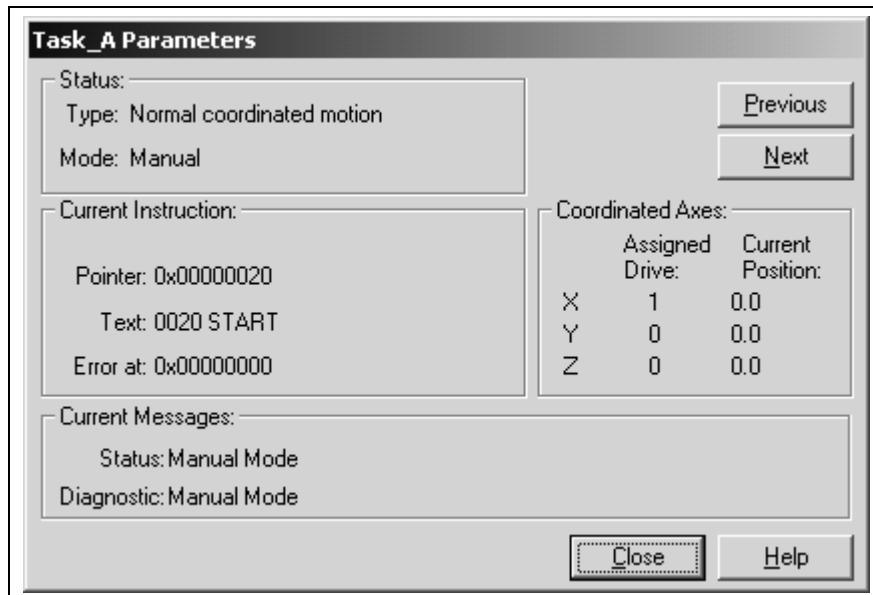


Fig. 7-100: Tasks

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).

Current Instruction - displays the instruction executing and its pointer, and a pointer to a run-time error if one has occurred. 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 SERCOS cycle sampled instruction execution, which may appear to be random.

Current Messages displays the last messages encountered in the program.

Coordinated Axes displays the axes in the active task that are assigned to coordinated motion and their current position.

Oscilloscope

Selecting **Diagnostics** ⇒ **Oscilloscope** opens the *Oscilloscope* window.

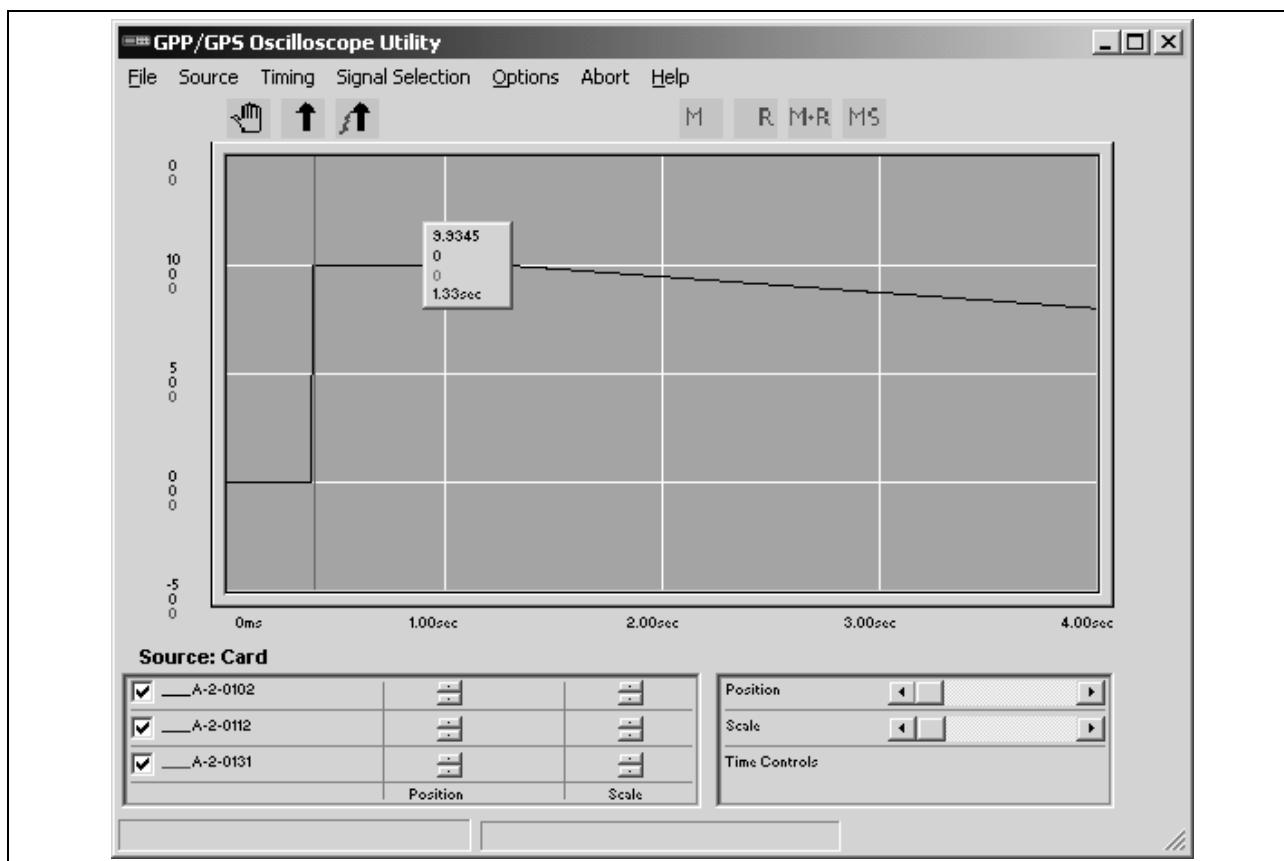


Fig. 7-101: Oscilloscope

For information on the Oscilloscope function, refer to [VisualMotion 8 \(GPP\) Application Manual - Drive Parameter Editor](#).

Break Point

Selecting **Diagnostics** ⇒ **Break Point** opens the *Break Point Control* window. The Breakpoint Control utility is a debugging tool. The window is opened with four separate windows, one for each task, as shown below.

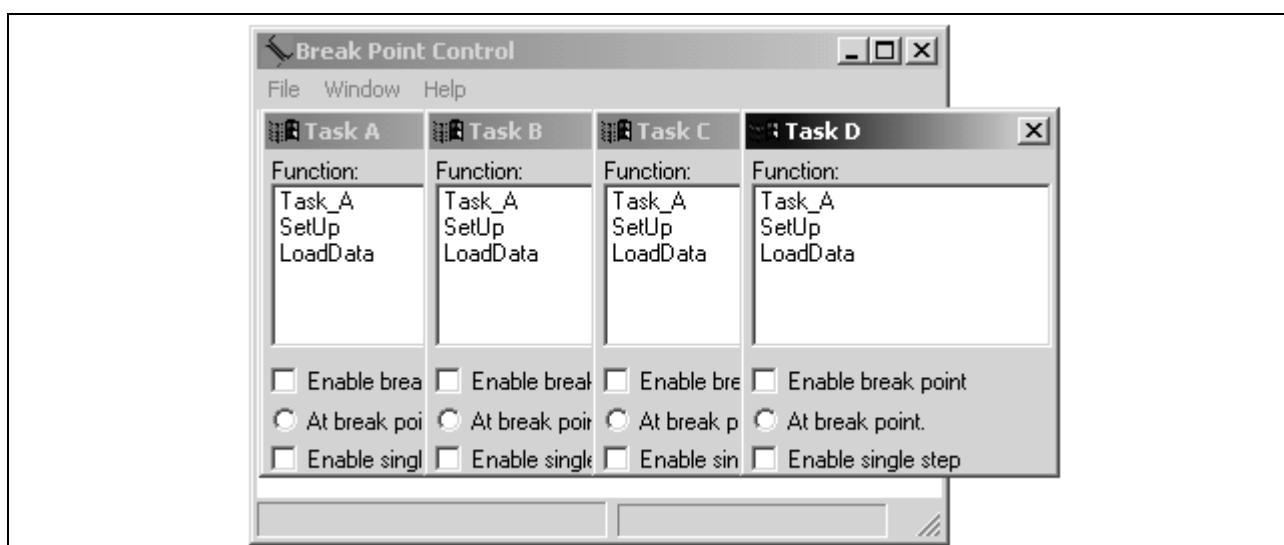


Fig. 7-102: "Break Point Control" Window

Each grouping contains a list of all functions of the active program, an enable checkbox, and an indicator of when the breakpoint was reached.

- Enable break point checkbox
- At break point
- Enable single step checkbox

When enabled, this utility stops the flow of the running program flow of a selected task at the start of a selected function. Toggling the Cycle_Start_Resume (bit 6) of the task control register (2-5) of the selected task resumes program flow.

To stop at a function:

1. Select the function from the list box of the task to be stopped. The list-box contains a listing of all functions, although some functions are never run from a given task (e.g. Task B is never run from Task A).
2. Arm the function by clicking on the enable checkbox.
3. Wait for dot in the indicator.

Show Program Flow

Selecting **Diagnostics** \Rightarrow **Show Program Flow F7** highlights the currently executing icon in a running program, as shown below:

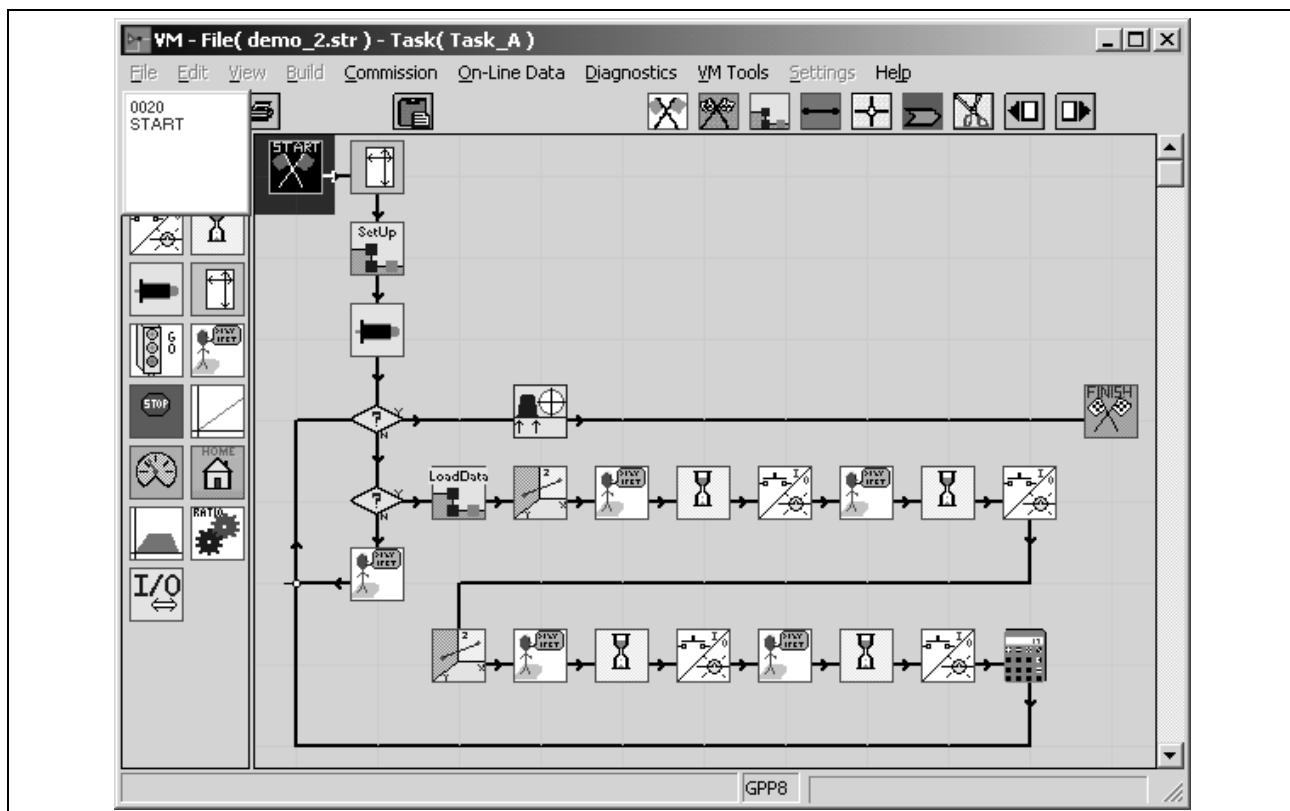


Fig. 7-103: Show Program Flow View

Many icons operate quickly and may appear to be skipped over. To see the program flow for subroutines, the operator must identify the task controlling the subroutines.

Show Program Flow utilizes 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). If changes have been made to the previously compiled and downloaded icon program, erroneous program flow may appear.

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.

7.9 The VM Tools Menu

The **VM Tools** menu contains the Cam Builder and Jogging utilities. The user can also launch additional Indramat specific utility that may appear below Jogging. Items such as different releases of the DDE Server, IoBox and even VisualMotion Toolkit can be launched.

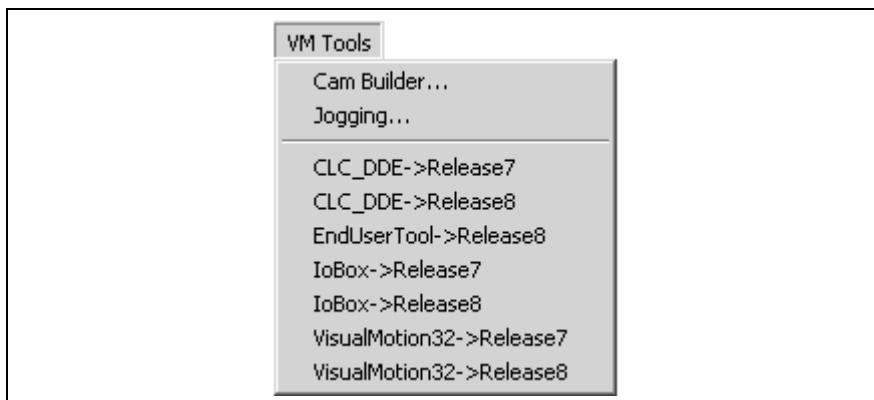


Fig. 7-104: The VM Tools Menu

CAM Builder

Selecting **VM Tools** ⇒ **Cam Builder** opens the *CAM Building Utility* window below. This utility is a Windows-based application used to build cam tables for Indramat's motion controls.

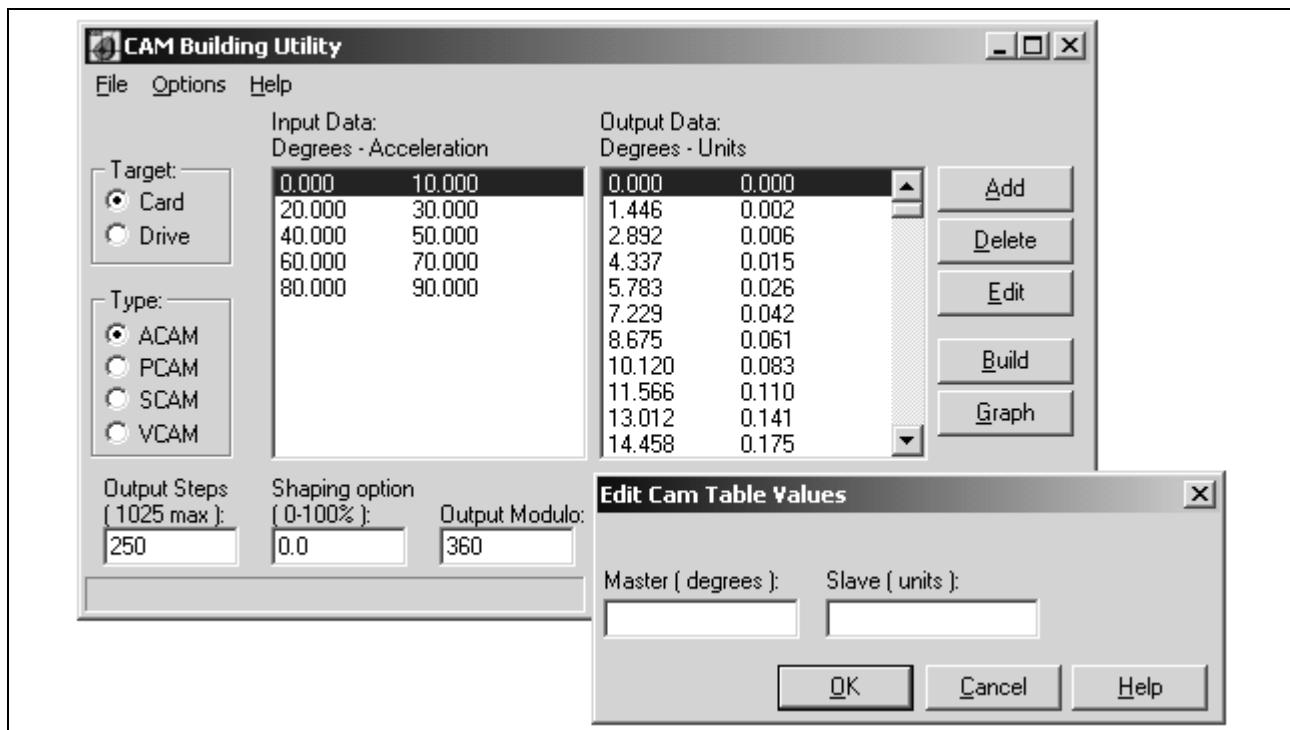


Fig. 7-105: "Cam Building Utility" Window

Controls

Target - selects the format of the output data.

Card - the output file will be in Degrees - Units.

Drive - the output file will be in Degrees - Percent.

Note: Units and Percent depend on the axis modulo.

Type - selects the algorithm used to build output data from input data (ACAM, PCAM, SCAM or VCAM).

Output Steps - selects the number of lines of output data to be built.

S Shaping (Jerk limiting) - select the percentage of S-shaping to be used on the velocity curve to prevent a jump in acceleration.

Output Modulo - for scaling the range of ACAM and VCAM output calculations. By default, this scaling is 360.

Input Data - list box containing input data. Double click left mouse button to edit or click the **Edit** button.

Output Data - list box used to display output data. This data is read-only.

Buttons

Add - add an entry to the input data list box.

Edit - edits selected entry in input data list box.

Delete - deletes selected entry in input data list box.

Build - builds position, velocity, and acceleration profiles and displays position data in the output data list box. Uses the algorithm selected in Build Types to build the number of data sets chosen in Output Steps using the Input Data as input and S Shaping to modify.

Graph - creates/refreshes a graph from data in the output data list box.

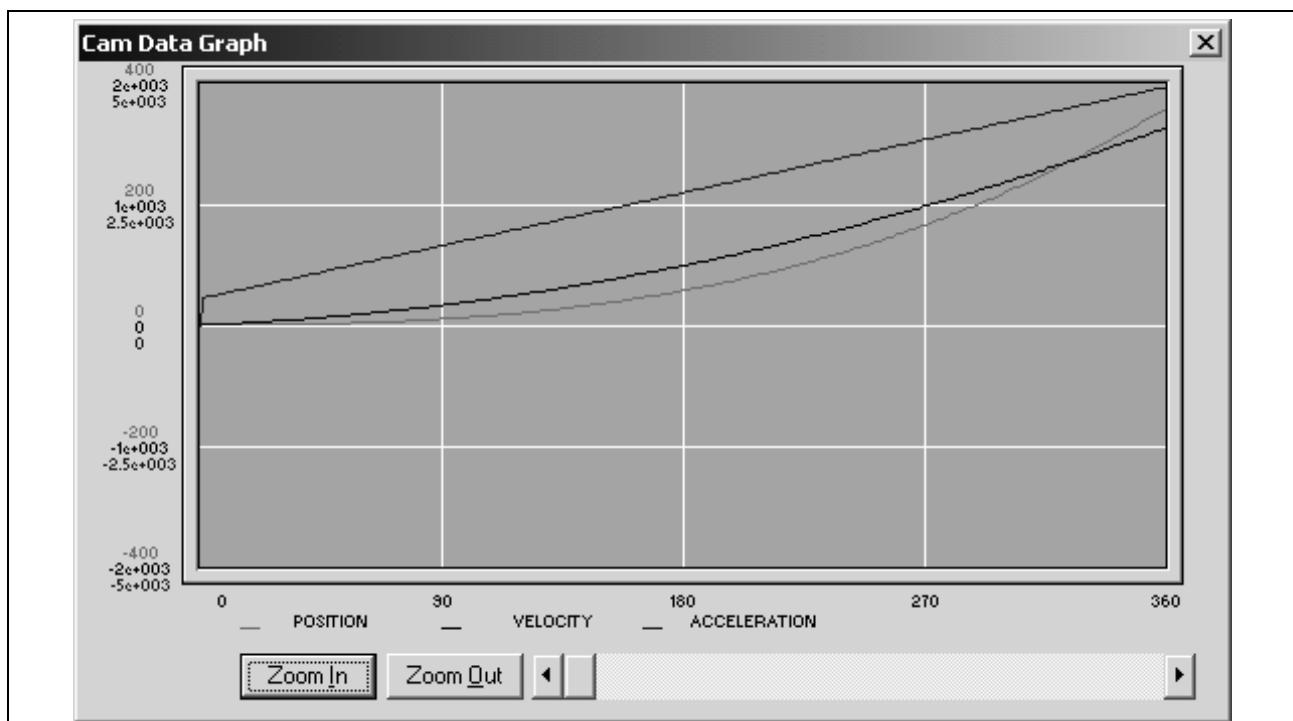


Fig. 7-106: Cam Data Graph

File Menu

The File Menu is used to retrieve file data, save data to a file, print, and exit.

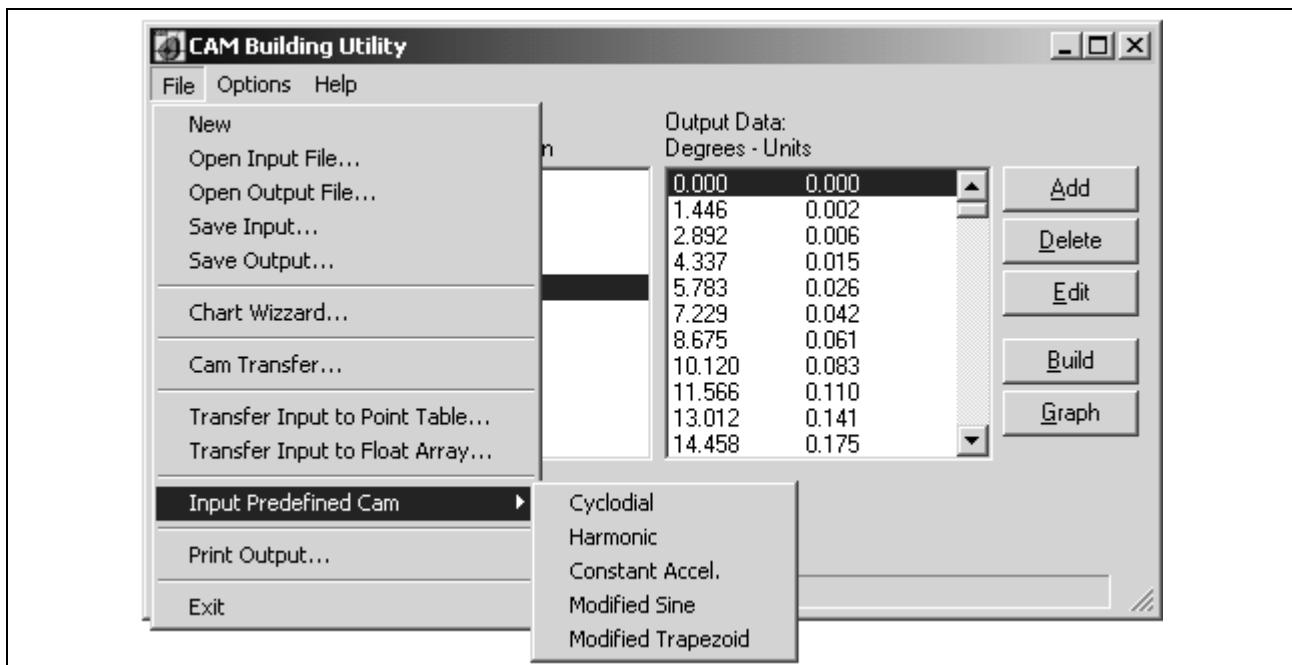


Fig. 7-107: Cam Building Utility, File Menu

New - clears input data list-box.

Open Input File – loads data from user-selected input file into input data list-box.

- *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   1024   500   1024
```

Open Output File – loads data from user-selected output into output data list-box.

- *User-selected output file*

Output files are assumed to have the following format. This is commonly referred as CSV format and supported by Excel. 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 - saves values from input data list-box to a user-selected input file.

Save Output - saves values from output data list-box to a user-selected output file.

Chart Wizard – follows a series of setup screens to help setup selections on the main screen. More detailed explanations for each selection are given.

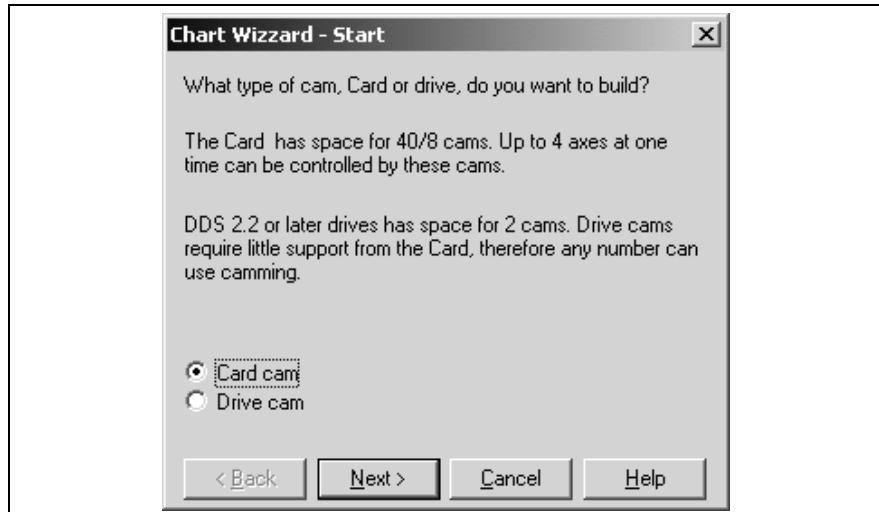


Fig. 7-108: Chart Wizard

Cam Transfer - cam transfer function to move a cam file to the control or drive cam table.

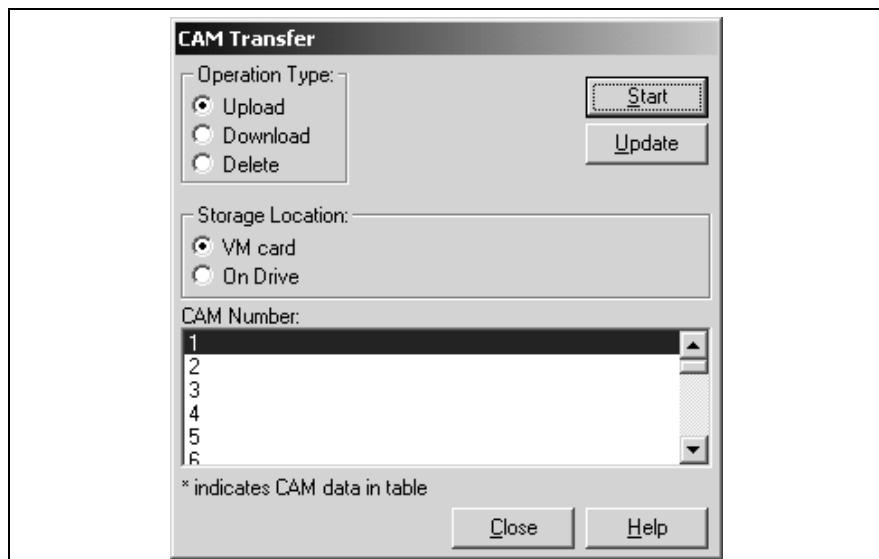


Fig. 7-109: Cam Transfer

Transfer Input to Point Table - transfers the contents of the input data list-box to points in a VisualMotion 8 program.

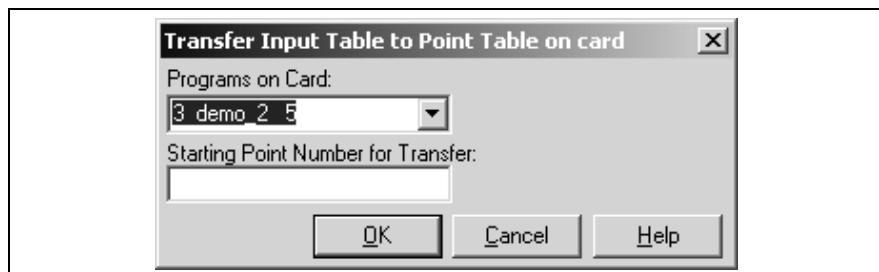


Fig. 7-110: Transfer Input Table to Point

Starting Point Number for Transfer: Enter the beginning point number to which the data should be copied. The values of the first line of the "input

“data list-box” will be copied to this point. The values from each additional line of the “input data list-box” will be moved to successive point locations.

- **Transfer Input to Point Table**

This screen is used to transfer the contents of the ‘input data list-box’ to points in a VisualMotion 8 program. A set of input values used for the CAM Build icon or instruction can be evaluated with the **Build** and **Graph** buttons of the previous screen. When the resulting cam is acceptable, the set of input values is transferred to the point table of the selected program on the control.

Transfer Input to Float Array

This screen acts exactly like the Transfer Input to Point Table menu item, except the data are copied into floats instead of points.

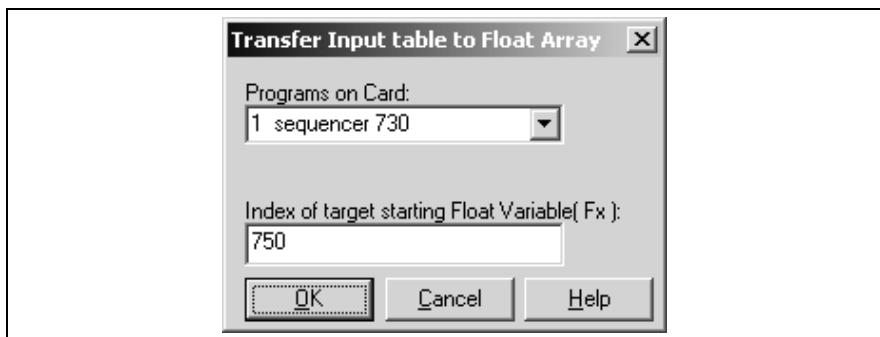


Fig. 7-111: Transfer Input to Float Array

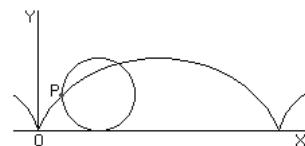
Programs on Card: This combo-box lists the programs on the control. The list is uploaded when the screen is opened. Select the program containing the point table to which to write.

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 screen.

Input Predefined Cam - inputs data from a predefined cam file.

The choices are:

- 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 - prints graph and output data to a Windows printer.

Exit - closes the Cam Building Utility window.

Options Menu

This *Options* menu is used for graphing and printout choices.

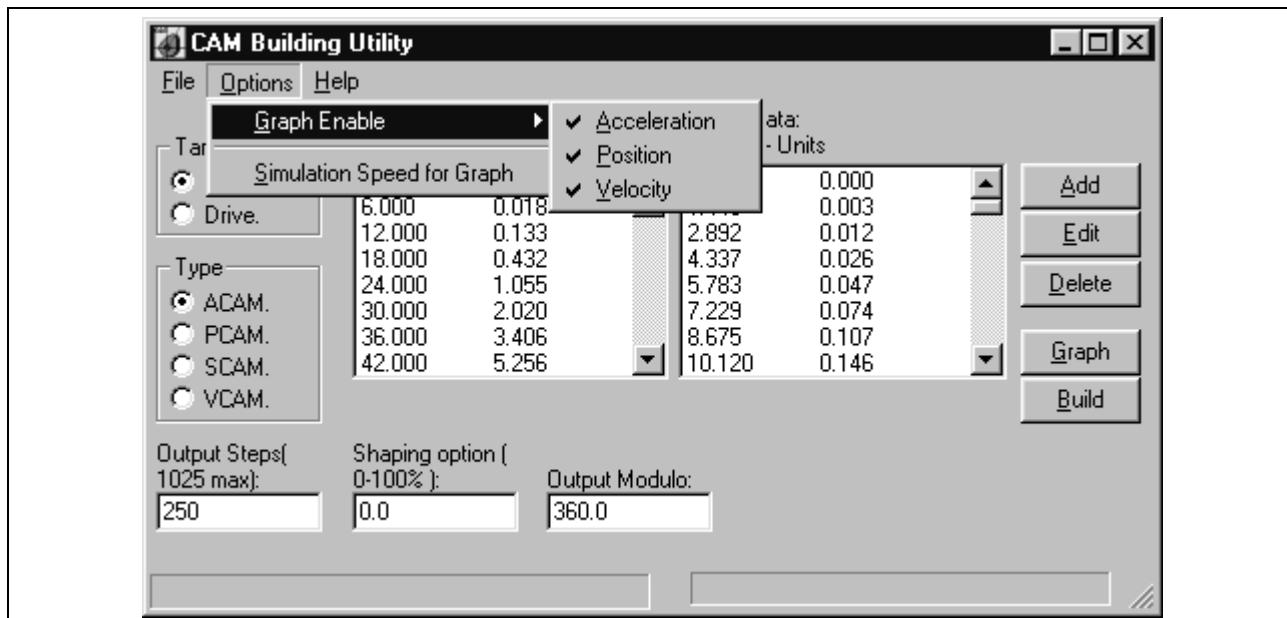


Fig. 7-112: Cam Building Utility, Options Menu

Graph Enable – allows the user to choose between Acceleration, Position, and Velocity graphs.

- *Acceleration Graph* - when checked, the acceleration graph is visible on the graph and print out.
- *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 print out.

Simulation Speed for Graph - selects the speed used in graphing the velocity and acceleration.

Help Menu

The Help menu is used for accessing help system and identifying the product.

Topic - accesses this help system.

About - identifies product, version and date of build.

Jogging

Selecting **VM Tools** ⇒ **Jogging** opens the following **Jog Control** window used for jogging Rexroth Indramat's digital drives attached to a control.

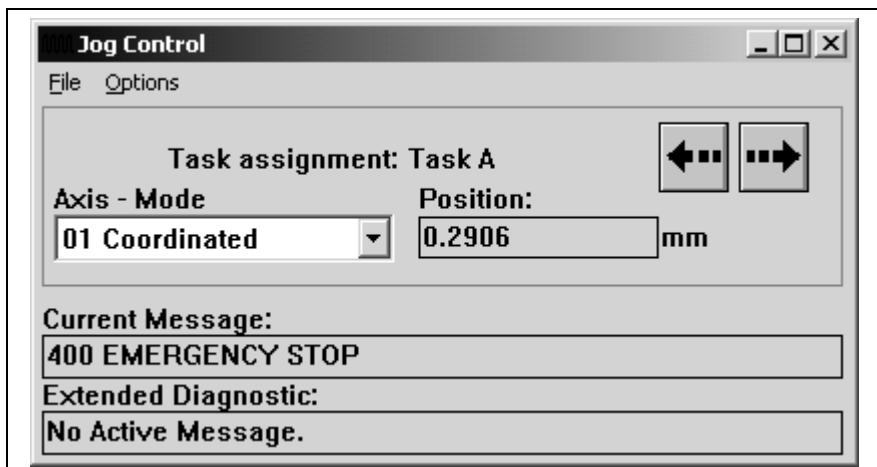


Fig. 7-113: Jog Control Window

When the control is in manual mode, the **Jog Control** tool allows jogging of an axis that is in *Velocity* or *Single Axis* mode. For *Coordinated motion* all jogging functions are controlled by the Axis(*n*)_Control and Task(A-D)_Jog Registers:

Registers 11-18, and 209-240: Axis *x* Control (where *x* = 1-8)

Registers 7-10: Task *x* Jog Control (where *x* = A-B)

Refer to [**C-0-0010 Control Input/Output Systems**](#) for more information.

When system parameter C-0-0010, bit 11 is set to 1 (0000010000000000), jogging can also be performed in Automatic Mode or while a task is running.

The window displays the current task assignment, mode and position for the active axis along with status messages and extended diagnostics.

Clicking 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. Deselecting these buttons changes 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.

Axis Jogging Options

Selecting **Options** ⇒ **Axis** opens the following window:

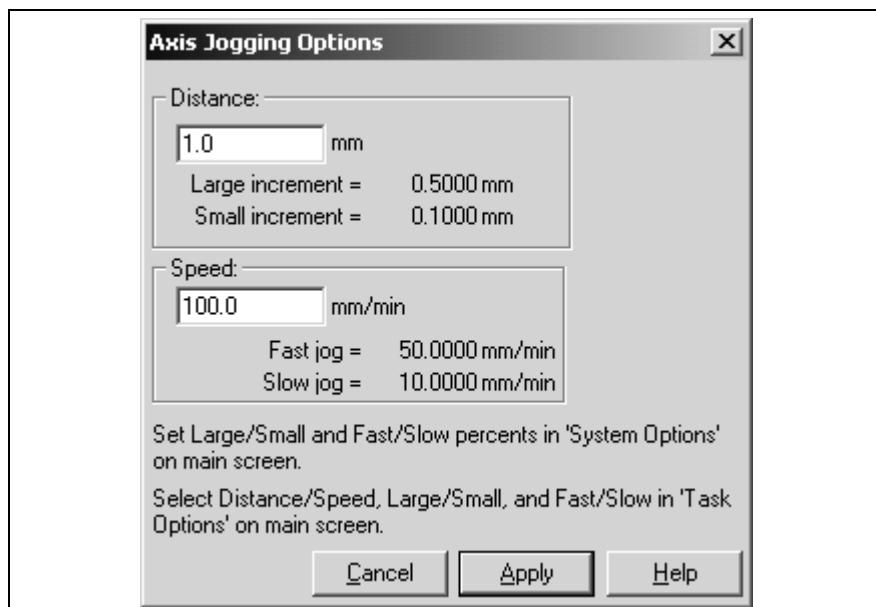


Fig. 7-114: "Axis Jogging Options" Window

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 selection described below.

System Jogging Options

Selecting **Options** ⇒ **System** opens the following window:

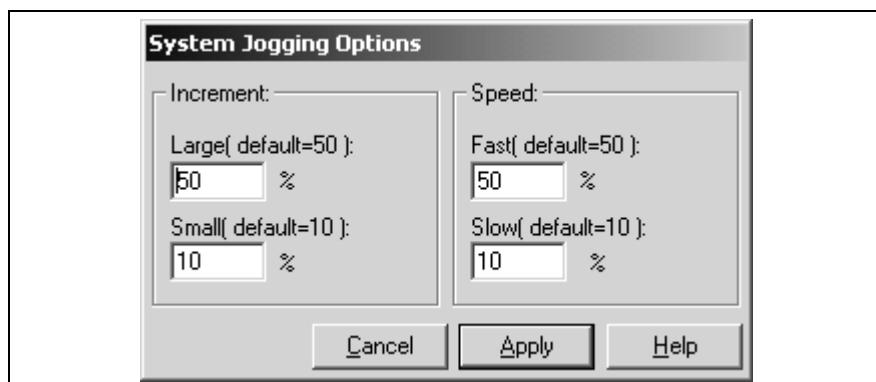


Fig. 7-115: "System Jogging Options" Window

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 Jogging Options

Selecting **Options** ⇒ **Task** opens the following window:

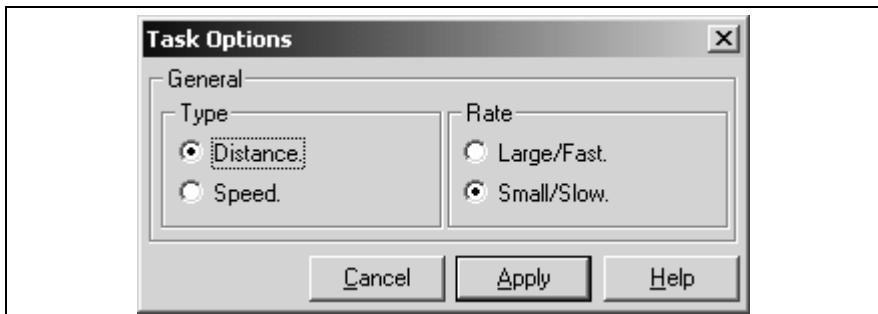


Fig. 7-116: "Task Options" Window

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 deselected or the travel limit is reached. [Refer to VisualMotion 8 \(GPP\) Application Manual, Drive Parameter Editor - Drive Limits.](#)

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.](#)

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.

CLC_DDE Release7 / CLC_DDE Release8

Selecting **VM Tools** ⇒ **CLC_DDE Release x** opens the DDE Server, if not already open.

IoBox Release7 / IoBox Release8

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 Release 7 / VisualMotion32 Release 8

Selecting **VM Tools** ⇒ **VisualMotion32 Release x** opens a separate instance of the selected VisualMotion release version.

7.10 The Settings Menu

The **Settings** Menu allows the user to enter basic setup information for VisualMotion and the currently open program.

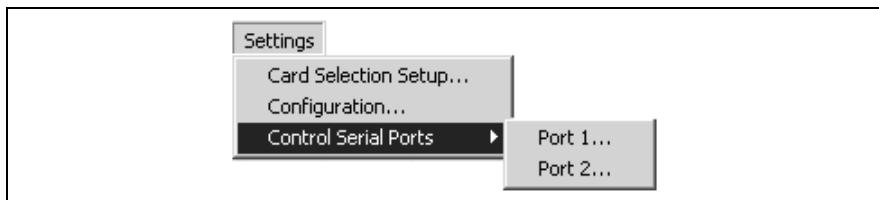


Fig. 7-117: The Setting Menu

Card Selection

Selecting **Settings** ⇒ **Card Selection** opens the following window:

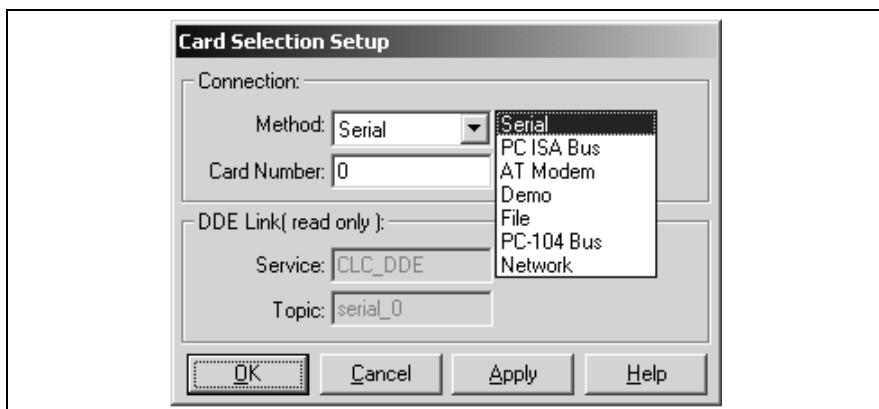


Fig. 7-118: "Card Selection Setup" Window

Possible selections are:

Serial – connection between the PC and control using an IKB0005 serial interface cable.

PC ISA Bus

AT Modem

Demo – no connection between the PC and the control. This setting allows some offline setup, but none of the runtime utilities or auto-detection functions can be used.

File – after saving and compiling an icon program file, the user can setup variables, events, point tables and zones without being connected to a control.

To setup the off-line programming:

1. Select **File** from the connection method drop-down box. This tells VisualMotion to get setup information from a file on the host computer instead of from the serial port, AT bus, etc.
2. To select a file, click on the **File** button and select the desired .exc file from the **Open** window.
3. Click the **Open** button, then click **Save** in the **Card Selection** window. Now, the programmer can enter values for variables and point tables and configure events that will be downloaded to the control with the *.exc file.

PC-104 Bus

Network – connection to a GPP control across a TCP/IP connection.

Configuration

Selecting **Settings** ⇒ **Configuration** opens the following window and allows the user to set the following:

- Windows Editor
- default Project Directory for saving files
- target firmware
- programming language
- language for the BTV and drives

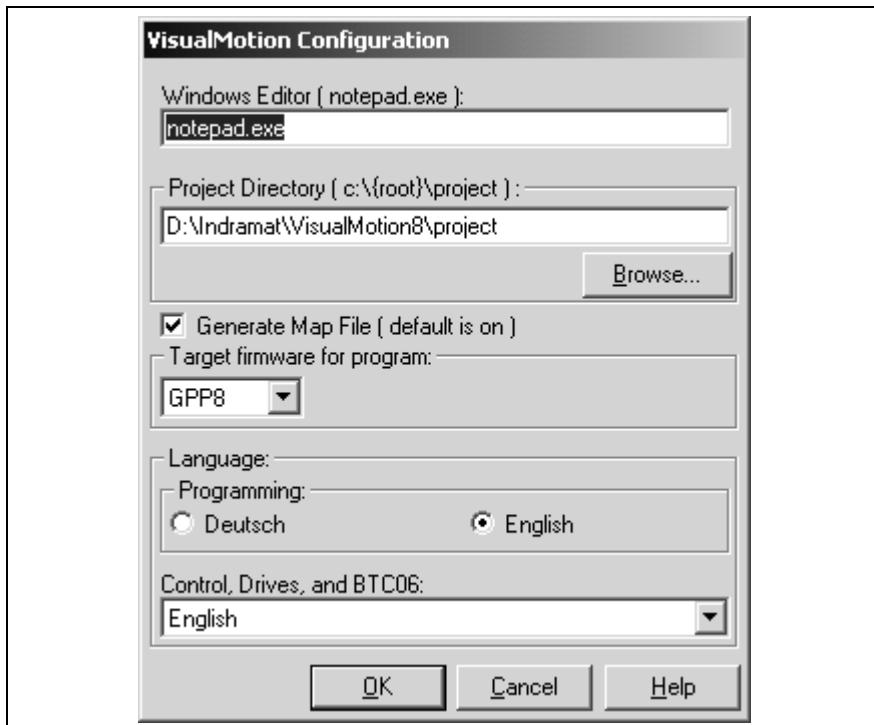


Fig. 7-119: "VisualMotion Configuration" Window

Windows Editor - Select the Windows compatible text editor to be used for non-graphic displays. The default is Notepad.exe.

Project Dir. - Select the directory for file storage. The default is \VisualMotion 8\project. The **Browse** button allows the selection of a different project directory.

Generate Map File - When icon-based programs are compiled, a map file can be generated to be used to show program flow. On larger programs or older PCs, the generation may take several minutes. The default is on.

Target firmware for program - Choose GPP 7 and GPP 8.

Language:

Programming – Choose between Deutsch (German) and English. This is the language for the menus and utilities in VisualMotion 8.

Control, Drives and BTC06 - Choose between Deutsch (German), English, French, Italian and Spanish. French, Italian and Spanish are supported for the Drives and BTC06 only. This is the language for the drive-related diagnostic messages and the BTC.

To save and exit the window, first click **OK**.

Control Serial Ports

Selecting **Settings** ⇒ **Control Serial Ports** ⇒ **Port x** permits setup of the control's two serial ports. The following window opens:

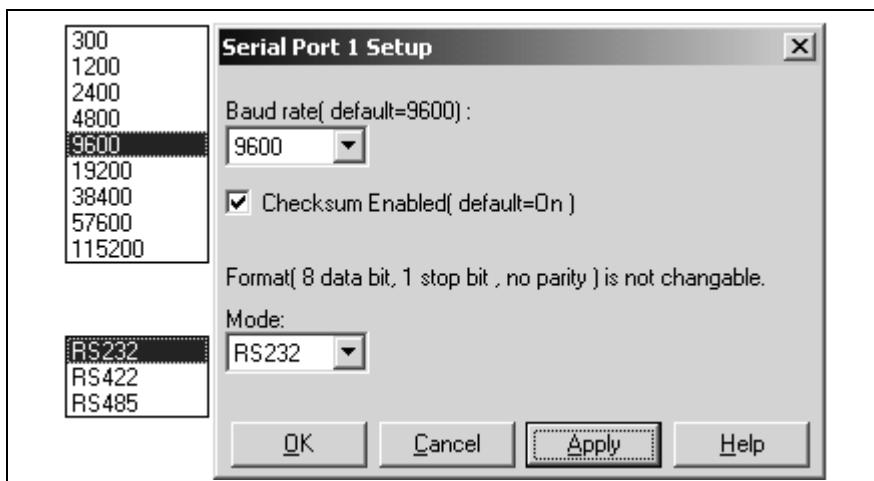


Fig. 7-120: "Serial Port x Setup" Window

Port 1 is typically used for communication with the host system and defaults to 9600 baud. This baud rate can be increased.

Port 2 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. Before saving any serial port modifications, VisualMotion informs the user that changes take effect on a reboot of the system and displays the following message:



Clicking Apply or OK downloads the changed values to the control.

7.11 The Help Menu

The **Help** menu provides assists to users in the form of an online help system. The user can also setup any drive related parameter help files installed on the hard drive to provide online help for drive parameters. A version change log is included with the installed version of VisualMotion indicating the current modifications and enhancements.

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.

Drives Help Directories

The drives help directories are used to set the path(s) to the drive parameter help file(s). Selecting **Help** ⇒ **Drives Help Directories** opens the following window:

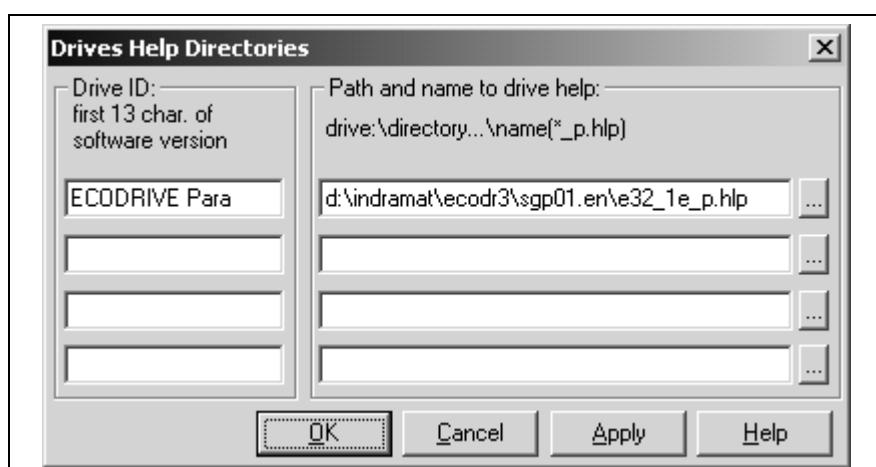


Fig. 7-121: "Drives Help Directories" Window

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 an Indramat office.

The left edit field is for the first 13 characters of the drive firmware typecode. It can be read from the drive editor screen or the list of drives on SERCOS ring windows.

The right edit field is for the complete path of the drive help file containing the parameter descriptions. The last character in the filename is '.'

Version Change Log

Selecting **Help** ⇒ **Version Change Log** displays a list of new features that appear in the installed version of VisualMotion Toolkit.

About VisualMotion

Selecting **Help** ⇒ **About** displays version information of VisualMotion 8 software and firmware.

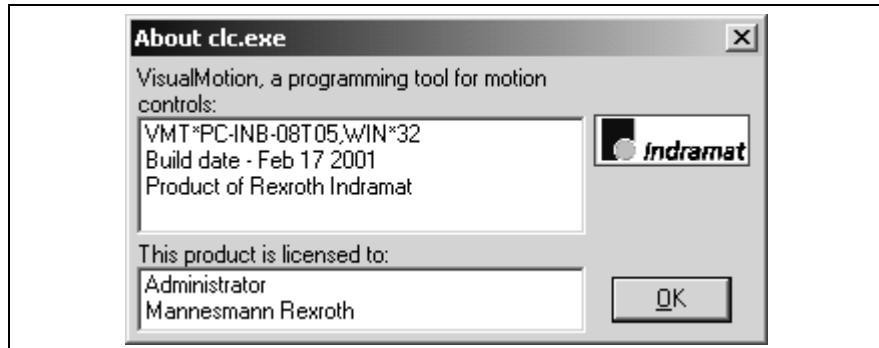


Fig. 7-122: About VisualMotion

8 Icon Programming

8.1 Introduction

This section describes how to use icons and other VisualMotion commands to create motion programs, a process briefly illustrated in the flow diagram below.

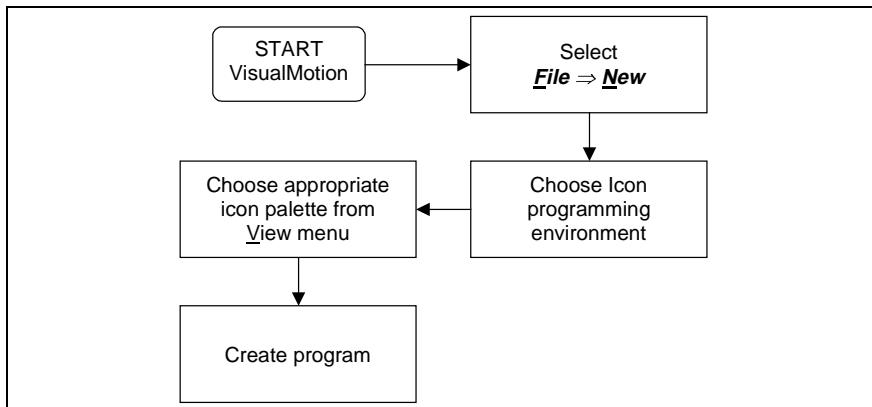


Fig. 8-1: Icon Programming Overview

The first section describes how icons are placed and connected in motion control programs. The second section contains descriptions of each icon, presented in alphabetical order.

Working with VisualMotion Toolkit's Icon Palettes

A VisualMotion Toolkit icon palette is displayed to the left of the program workspace when selected from the View menu. Four standard palettes are provided for single, coordinated, ELS and Utility icons. An icon in the palette is selected for placement by clicking on it with the left mouse button. The selected icon is placed on the VisualMotion workspace by positioning the cursor where you want the icon to appear and clicking once. A different icon may be selected for placement simply by clicking a new icon in the palette.

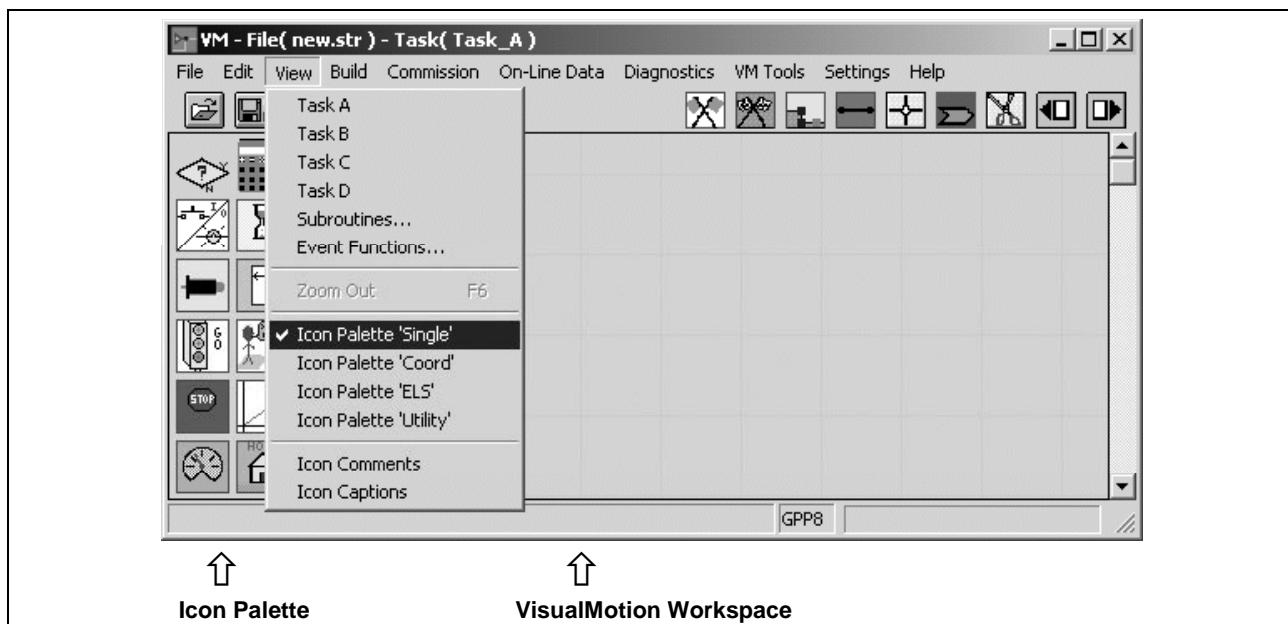


Fig. 8-2: VisualMotion Toolkit Environment

The VisualMotion window also displays a set of frequently used operational icons. These icons are always visible above the workspace regardless of the different icon palettes that might be selected.

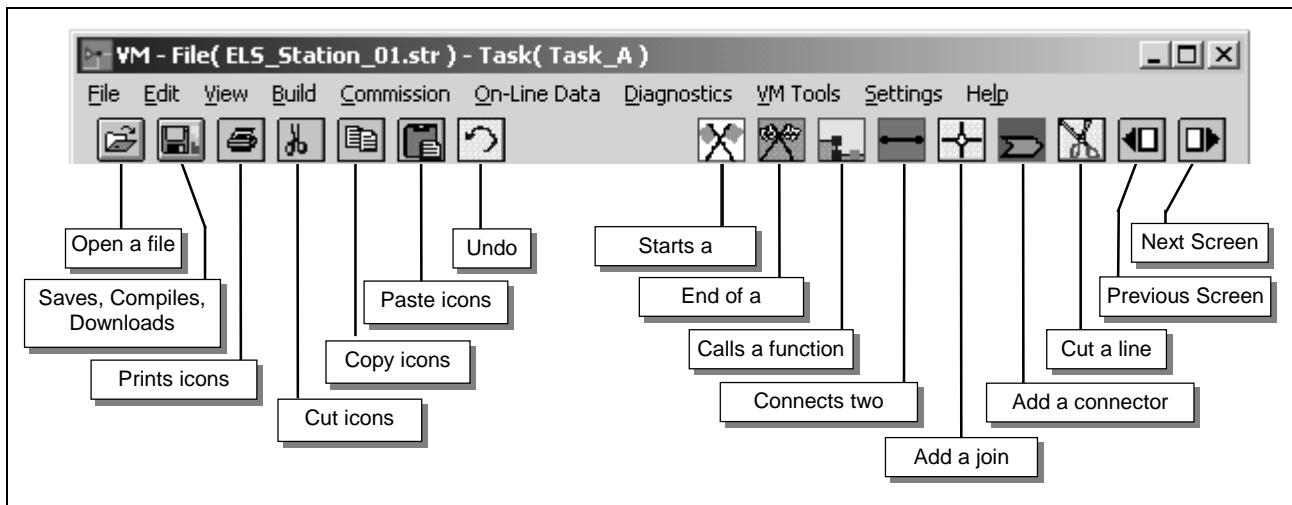
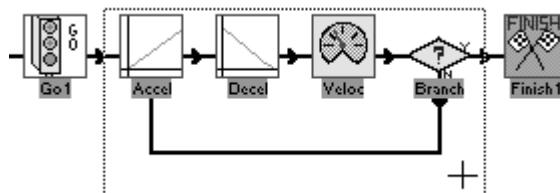


Fig. 8-3: VisualMotion Toolkit Program Menu and Icon Button Bar

Only left and right mouse buttons are recognized by VisualMotion. Center and scroll wheel buttons are not supported in VisualMotion. The left button is used for selecting single icons or lines; selecting commands from the pull-down menus; and positioning the cursor in windows for keyboard entry. In order to select multiple icons or lines, click and hold the left mouse button while drawing a selection box around a program section. The current icon may be de-selected, freeing the workspace cursor, by re-clicking on the selected palette icon.



Cut, Paste and Copy an Icon

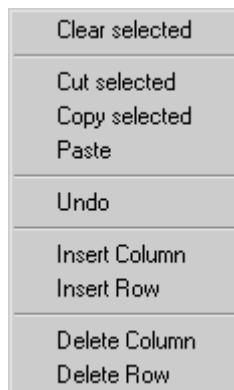
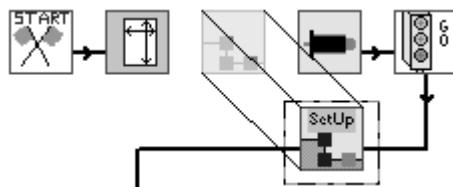
Once a selection has been made, the cut and paste menu selections under Edit can be used. Once a selection has been cut or copied, it is saved to the Windows clipboard. If the paste button is pressed, the selection will appear on the screen and follow the cursor until a position for it is selected with the left mouse button.

Delete an Icon

To delete or clear a selection or icon, press the delete key or select delete from the Edit menu.

Moving an Icon or Selection

To move an icon or selection, click on the icon(s), release the mouse button then click and hold the icon(s) again. The icon(s) can now be dragged to a different position and placed by releasing the mouse button.



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.

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



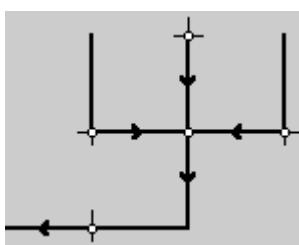
After you have placed a number of icons on the workspace, they must be connected to indicate the program flow. Most icons have a maximum of three possible inputs and one output. The exception is the Branch icon, which has two outputs.

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.

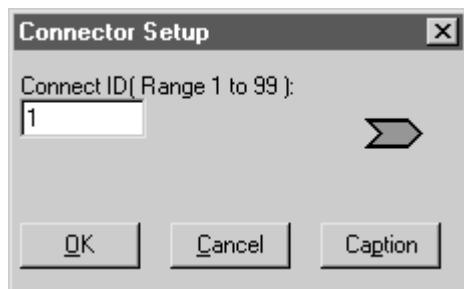




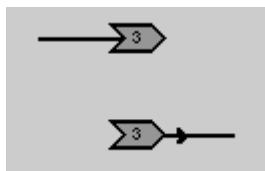
A line connecting two icons may be removed by using the Scissors icon from the icon toolbar. Simply select the scissors icon, position the scissors over the line to be removed, and press the left mouse button.



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 logically fit in a relatively small workspace.



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.



The **Previous screen** and **Next screen** buttons provide a shortcut to jump between Program Tasks, Subroutines and function screens.

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 pop-up window as the cursor passes over the icon.

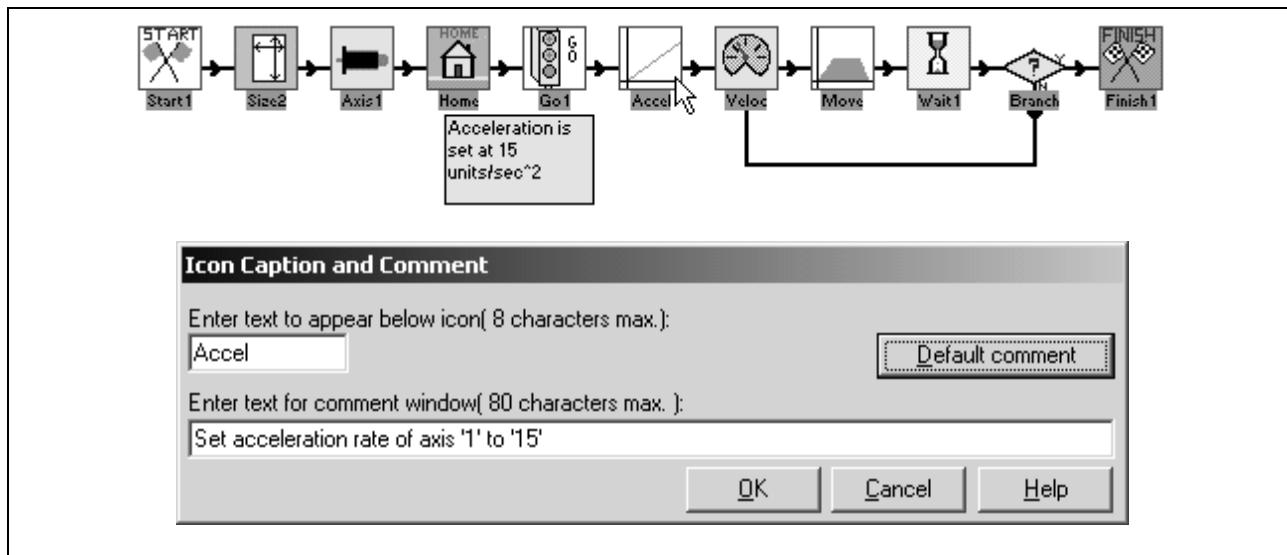


Fig. 8-4: Icon Captions and Comments

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.

8.2 VisualMotion Toolkit Icons

Four standard palettes are provided for single, coordinated, ELS and Utility icons. The following tables will list each icon palette along with their names, descriptions and the page number where the functional description is located.

Single Axis Icon Palette

Icon	Name	Function	Location
	Branch	Directs program flow	8-17
	Calc	Calculation and initialization	8-19
	I_O	Sets an I/O register bit	8-70
	Wait	Conditionally holds program execution	8-104
	Axis	Defines an axis for use	8-11
	Size	Allocates control memory for data types	8-95
	Go	Enables an axis (axes)	8-68
	Msg	Sets status or diagnostic message	8-74
	Stop	Halts motion	8-97
	Accel	Sets the acceleration of an axis	8-10
	Veloc	Sets velocity for non-coordinated motion	8-99
	Home	Initiates drive homing procedure	8-69
	Move	Axis position move	8-73
	Ratio	Sets the ratio between two axes	8-90
	Reg	Transfer data between registers and variables	8-91

Table 8-1: Icon Palette "Single"

Coordinated Motion Icon Palette

Icon	Name	Function	Location
	Branch	Directs program flow	8-17
	Calc	Calculation and initialization	8-19
	I_O	Sets an I/O register bit	8-70
	Wait	Conditionally holds program execution	8-100
	Axis	Defines an axis for use	8-11
	Size	Allocates control memory for data types	8-95
	Go	Enables an axis (axes)	8-68
	Msg	Sets status or diagnostic message	8-74
	Path	Coordinated straight line move	8-77
	Circle	Coordinated circular move	8-45
	Position	Gets position of coordinated move	8-82
	Event	Control system events	8-67
	Stop	Halts motion	8-97
	Joint	Six axis coordinated move	8-71
	Param	Transfers a control or drive parameter	8-75
	CvyrInit	Conveyor Tracker Setup (GPP 8 only)	8-47
	PathCalc	Complex Coordinated Move Block	8-78

Table 8-2: Icon Palette "Coordinated"

ELS Icon Palette

Icon	Name	Function	Location
	Branch	Directs program flow	8-17
	Calc	Calculation and initialization	8-19
	I_O	Sets an I/O register bit	8-70
	Wait	Conditionally holds program execution	8-100
	Size	Allocates control memory for data types	8-95
	Go	Enables an axis (axes)	8-68
	Msg	Sets status or diagnostic message	8-74
	AxisEvt	Sets rotary event for an axis	8-16
	ELSMODE	Switches phase or velocity of ELS axis	8-66
	Cam	Assigns a CAM to an axis	8-28
	ELSAJAD	Shifts phase or velocity of ELS axis	8-51
	Decel	Sets deceleration rate of an axis	8-50
	CAMADJ	Phase shift a CAM axis	8-29
	CMBLD	Builds a CAM table	8-30
	INDEX	Initializes a CAM indexer	8-34
	ELSMSTR	Assignment of ELS Masters	8-62
	ELSGRP	Assigns axes to an ELS Group	8-52
	VM	Initialization of Virtual Masters	8-100

Table 8-3: Icon Palette "ELS"

Utility Icon Palette

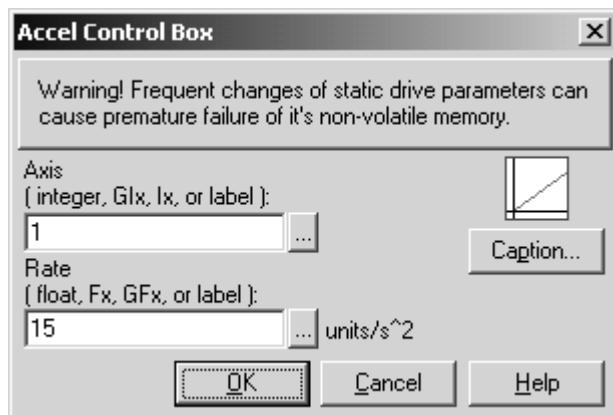
Icon	Name	Function	Location
	Param	Transfers a control or drive parameter	8-75
	PrmInt	Sets control or drive parameter at program activation	8-86
	PrmBit	Sets bit(s) in control or drive parameter	8-83
	Reg	Transfers data between registers and variables	8-91
	Command	Initiates drive resident commands	8-46
	Probe	Enables drive based position capture	8-88
	ProbeEvt	Enables event based on drive probe transition	8-89
	Go	Enables an axis (axes)	8-68
	Wait	Conditionally holds program execution	8-104
	Tool	Reference coordinated tool tip to new point	8-92
	Origin	Reference coordinated origin to new point	8-92
	PLS	Compile time initialization of PLS	8-81
	Sequencer	Initiates and or runs sequencer list	8-93
	PID	Installs and initializes a PID loop	8-79

Table 8-4: Icon Palette "Utility"

Accel



The Accel icon is used to set or change the acceleration of a single non-coordinated axis.



Axis indicates the axis to accelerate. The axis may be entered as an integer constant, integer variable, global integer variable or an equivalent label.

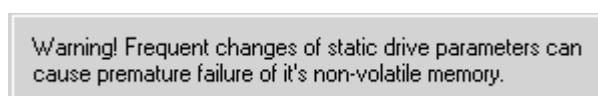
Rate specifies the acceleration rate in units per second squared. The entry may be a float constant, variable, global variable or an equivalent label. The variable must be greater than 0. A value ≤ 0 will generate an error "469 Axis D accel ≤ 0 or > maximum."

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

The *User Defined Labels* pop-up window is accessible by clicking on the button to the right of each data entry field.



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. This 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
-

Axis



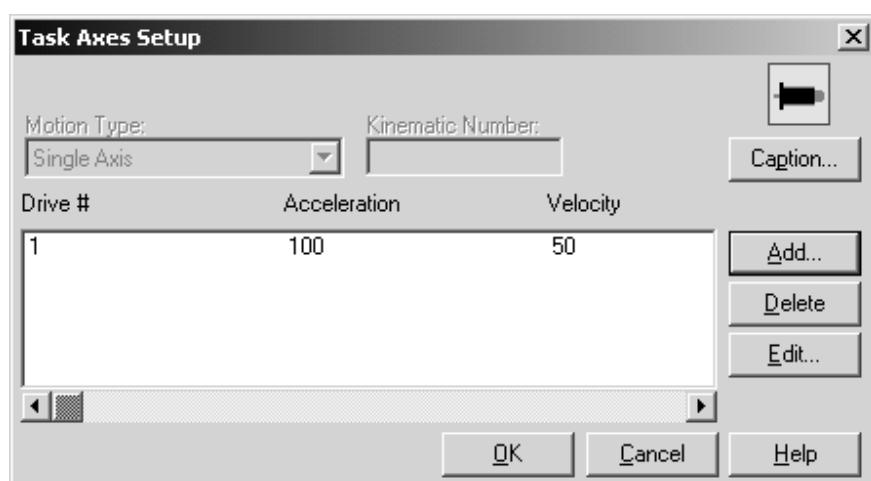
The Axis icon is used to specify drives that are assigned to the current task. The Axis icon can only be used to assign a drive within the four Tasks; it should not be used in a subroutine or event function.

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.

Kinematic Number permits entry of a kinematics library routine number or label identifying the kinematics to be used for **coordinated** motion. Kinematic routines are unique to hardware, consult your Indramat 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:

Axis Definition	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 Single Axis the drive list may be scrolled to display:

Drive #	Trigger 1
Acceleration	Trigger 1 Event
Velocity	Trigger 2
Halted?	Trigger 2 Event
Mode	

In Velocity Mode the drive list may be scrolled to display:

Drive #	Change
Acceleration	Trigger 1
Velocity	Trigger 1 Event
Halted?	Trigger 2
Mode	Trigger 2 Event

In Ratioed Axes the drive list may be scrolled to display:

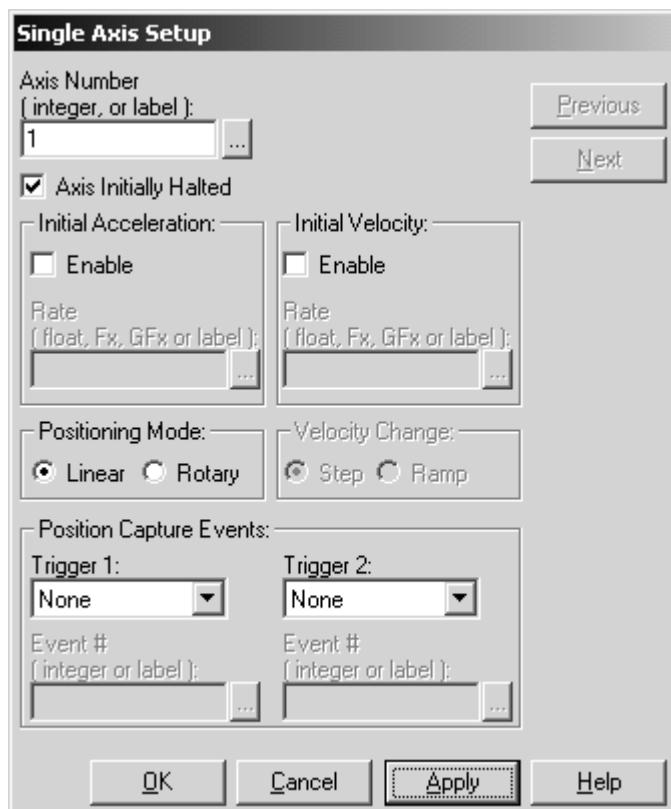
Drive #	Mode
Master #	Trigger 1
Slave Ratio	Trigger 1 Event
Master Ratio	Trigger 2
Halted	Trigger 2 Event

In Torque Mode, the list displays only the drive number. In Coordinated mode, the list displays the drive number and its designation as an x-, y-, z-, roll, pitch or yaw axis.

A pop-up User Defined Labels window is available for the data entry boxes by placing the focus ("I-beam" cursor) in an entry box, then double clicking the left mouse button on an empty area outside of the field.

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.

By default, the Axis **Initially Halted** option is checked. This disables the axis at the start of the task; it can be enabled using a Go icon later in the task. Clicking to uncheck the Axis Initially Halted option enables the axis to go at the start of a task.



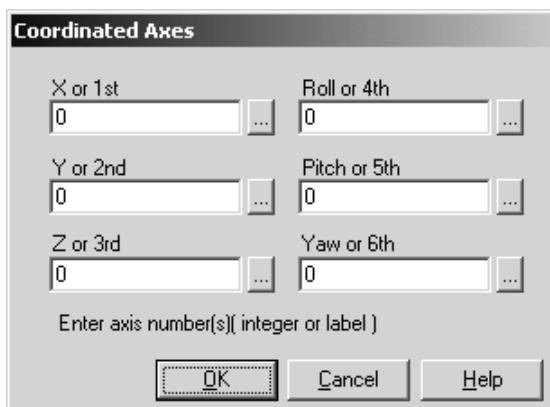
You may enable and enter an **initial acceleration** and **velocity** as a float constant, global float variable(GF1- GF256), program float variable (Fx), or an equivalent label. If the initial accel and velocity are not defined they must be subsequently specified in the program using the **Acce** and **Veloc** icons. Both Single and Velocity Mode axes may be assigned linear (units) or rotary (degrees) positioning.

Trigger 1, Trigger 2 - enables drive (non-RAC) based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

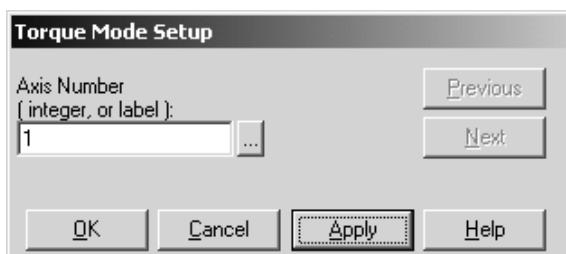
Trig1 Event, Trig 2 Event - event number to execute on position capture occurrence. Event must have been allocated in **Size** icon and initialized prior to occurrence. Captured position data can be read in the event's argument field(EVT[x].t) or from the drive parameter utilizing the **Param** icon. For more information, refer to the **ProbeEvt** icon.

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.

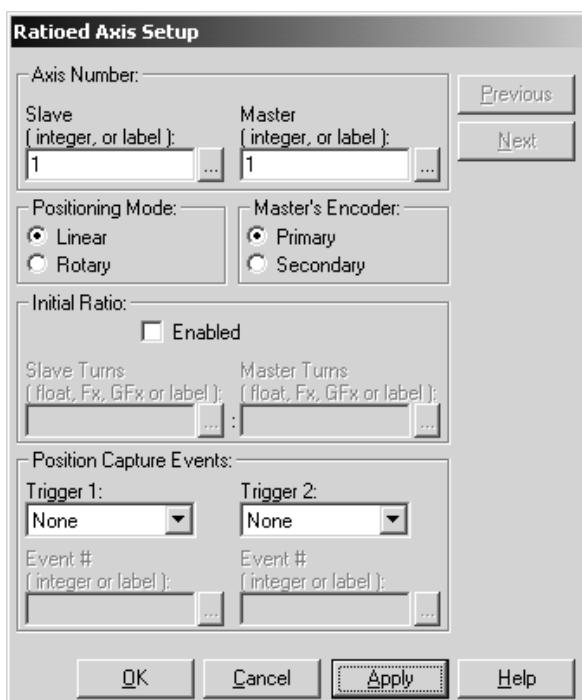
Selecting **Coordinated** motion type in the Task Axes Setup window and clicking the Add button opens a Coordinated Setup window. Axes are assigned by entering an integer or equivalent label for each axis and clicking the OK button. The Coordinated Setup window remains open until it is explicitly closed using the Cancel button. Although you may assign up to 99 axes using one coordinated motion Axis 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 data entry box.



Selecting **Torque Mode** allows entering axes in the same manner as coordinated mode; however, any number of axes may be assigned up to the maximum number of axes in the system.



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.



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).

AxisEvt



The AxisEvt icon is used to enable up to four rotary axis events for a specified axis. A maximum of four events may be enabled by any combination of AxisEvt or Move icons.

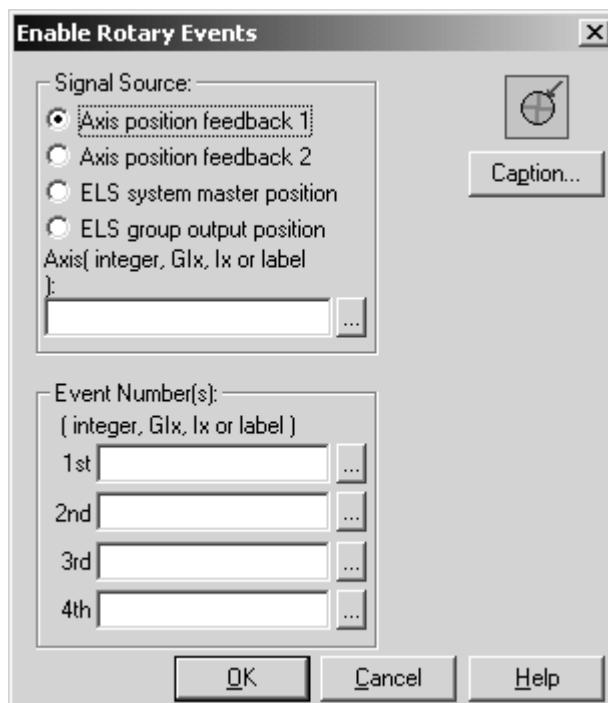
The Axis and Event numbers are specified using integer constants, global integer variable (Glx), program integer variable (Ix), or an equivalent label.

Note: A pop-up User Defined Labels window is available for the data entry boxes by placing the focus ("I" beam cursor) in an entry box, then clicking the cursor on an empty area of the window.

Each of the four edit fields permits entry of one event (1 - n) that may occur during movement on this path segment. A selected event must be configured in the event table prior to run time. The number of events is scaleable in the **Size** Icon.

For an event enable to have an effect, the event function must already be entered and setup as a type 5 (Rotary - Repeating Axis Position) event.

Note: Rotary events cannot be attached to an axis that has a PLS (Programmable Limit Switch) attached to it.

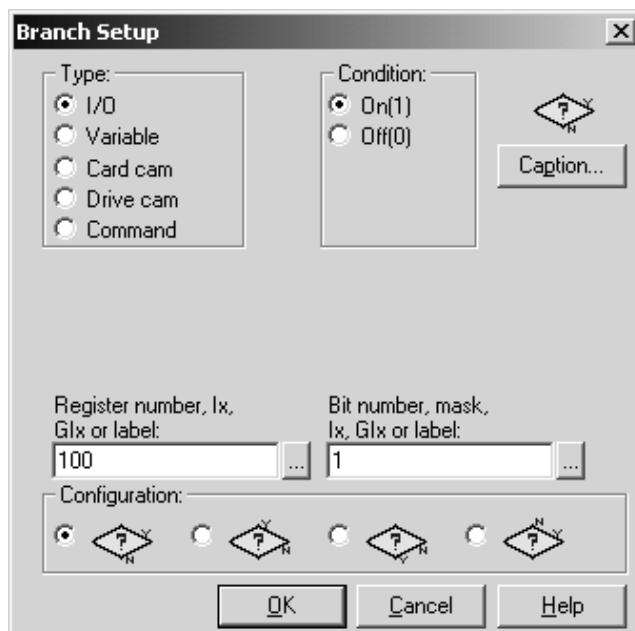


Branch

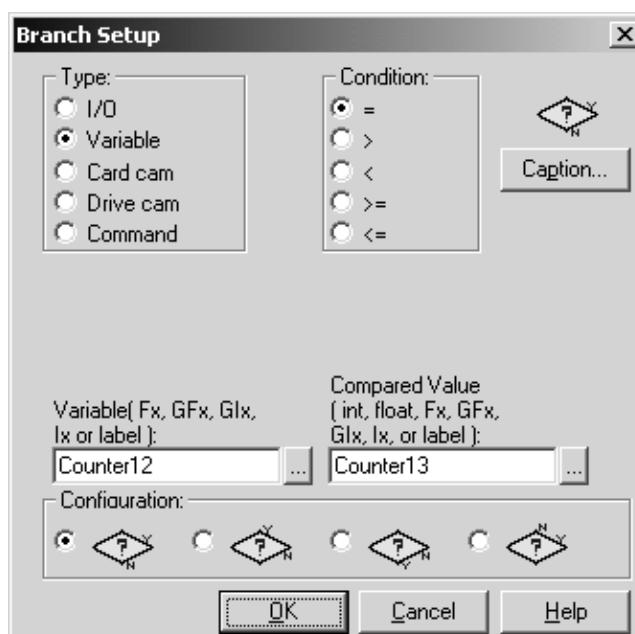


A Branch icon re-directs the program flow depending upon a true/false logical value.

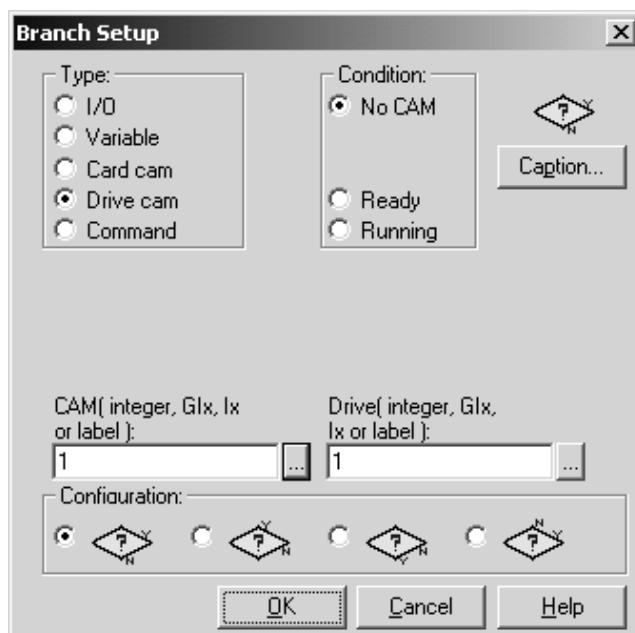
Four radio buttons at the bottom of the window permit selection of one of four Branch icon graphics, each with a different positioning of the branch test outputs.



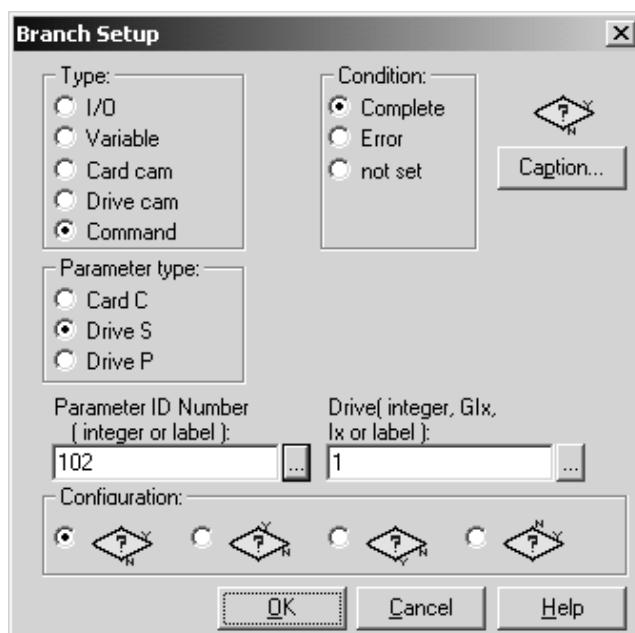
Selecting an I/O type branch permits testing a bit in a specified I/O register for a logical (true/false) condition. The I/O register for comparison is specified by an integer or an equivalent label. A bit mask only permits testing of selected bits within the specified register to be tested. The bit 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.



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). User defined labels are accessible through a pop-up window.



Selecting Card 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.



Selecting Command enables the Branch icon to monitor the condition of a Card, 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).

Calc

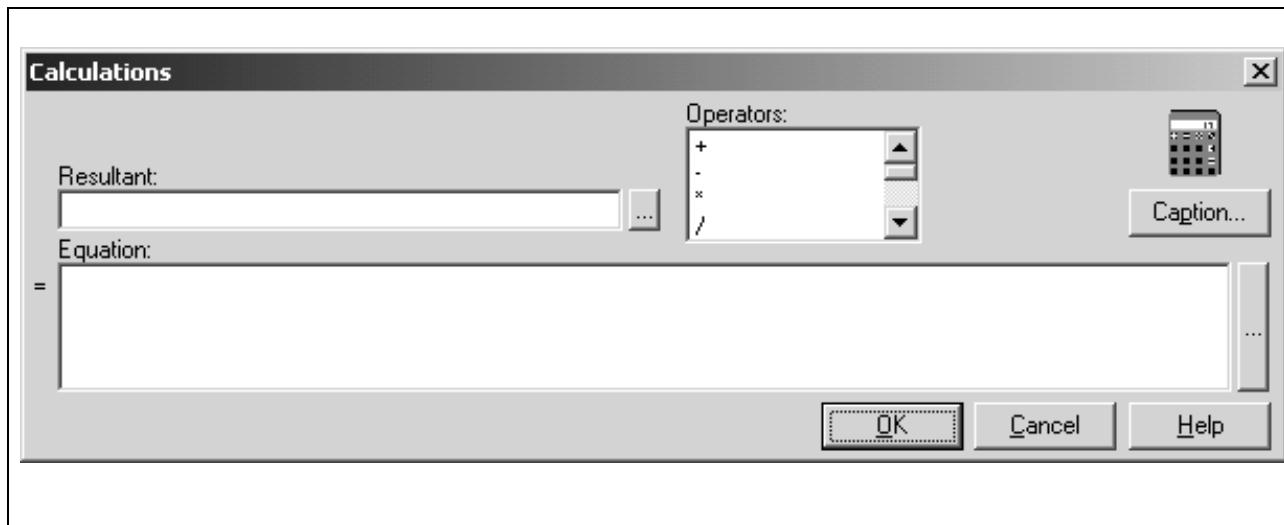


The Calc icon can be used for placing calculations within the program flow or to initialize variables, points, events, zones or PLS elements. By using Calc icons in this fashion, the user program is initialized with operational values, thereby eliminating the need to specify them after downloading the 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 *User Defined Labels* window by clicking on the button to the right of the Resultant and Equation fields.



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 Calc icon are listed in the following table:

<i>Arithmetic operators</i>	
+	addition (floating point or integer)
-	subtraction (floating point or integer)
*	multiplication (floating point or integer)
/	division (floating point or integer)
<i>Logical operators</i> Note: <i>Logical operations are limited to unsigned integers.</i>	
&	logical bitwise AND of two integers
	logical bitwise Inclusive OR of 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
<i>Transcendental functions</i> Note: <i>All transcendental functions are in radians.</i>	
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
<i>Exponential functions:</i>	
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
<i>Conversion functions:</i>	
%	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 **On-Line Data** \Rightarrow **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 its 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 ↓ F[2] = 5.0	First, the value in I5 is used as the entry number for the float variable If I5=2, then the 2 nd entry (row) of the float table is accessed and a value of 5.0 is written
ABS[I2].x = 4 ↓ ABS[3].x = 4	First, the value in I2 is used as the point in the absolute table If I2=3, then the 3 rd point of the absolute points table is accessed and 4 is written to its x element
Using a Label Assigned to an Integer Variable Indirectly	
F[move] = 3.2 ↓ F[2] = 3.2	First, the integer's value assigned to the label "move" is used as the entry number for the float variable If move=I5=2, then the 2 nd entry (row) of the float table is accessed and a value of 3.2 is written
ABS[set].x = 4 ↓ ABS[3].x = 4	First, the integer's value assigned to the label "set" is used as the point in the absolute table If set=I2=3, then the 3 rd point of the absolute points table is accessed and 4 is written to its x element

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,el}
REL[n]	= {x,y,z,b,s,a,d,j,e1,e2,e3,e4,r,p,ya,el}

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
el	elbow state/coordinated axis mask

Examples of data structure initialization:

The Calc 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 Calc 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 Calc 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 Calc 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 Calc 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 Calc 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[1].f = IO_On      ; indirect addressing
```

Example of data transfer between like structures

The Calc 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)
<code>ZONE[n]</code>	= { <code>s,a,b,c,d,e,f</code> }

Note: Elements in the data structure are separated by a comma(,) and no spaces are allowed between element values.

Element	Description	Selections
<code>s</code>	Status	1 = All Task 2 = Task A 4 = Task B 8 = Task C 16 = Task D
<code>a</code>	x Cartesian coordinate, Point 1	
<code>b</code>	y Cartesian coordinate, Point 1	
<code>c</code>	z Cartesian coordinate, Point 1	
<code>d</code>	x Cartesian coordinate, Point 2	
<code>e</code>	y Cartesian coordinate, Point 2	
<code>f</code>	z Cartesian coordinate, Point 2	

Example of data structure initialization:

The Calc 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 Calc 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 Calc 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 Calc 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 Calc icon can be used to modify different elements in the configuration. The following three PLS types can have their elements written to using the Calc icon.

- Card PLS
- Control PLS
- Drive PLS

Card PLS

A user program can contain only one active Card PLS. The following tables list the elements of a 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
<i>x</i> = Master range 1-8				

Switch Elements

The following switch elements are available for modification using the Calc icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].on <i>x</i>	On Position	PLSP[1].on3=40	switch 3 On position is 40	C-0-2920
PLSP[1].off <i>x</i>	Off Position	PLSP[1].on3=60	switch 3 Off position is 60	C-0-2921
PLSP[1].out <i>x</i>	Assigned Output	PLSP[1].out3=2	switch 3 is assigned to output 2	C-0-2922
<i>x</i> = switch range 1- 96				

Output Elements

The following Output elements are available for modification using the Calc icon.

Syntax	Description	Example	Comment	Parameter
PLSP[1].lt <i>x</i>	Lead Time	PLSP[1].lt2=500	add a lead time of 500µs to output 2	C-0-2931
PLSP[1].lg <i>x</i>	Lag Time	PLSP[1].lg2=500	add a lag time of 500µs to output 2	C-0-2932
PLSP[1].ss <i>x</i>	PT (Time Duration)	PLSP[1].ss2=1000	Maintain output 2 On for 1000µs	C-0-2933
PLSP[1].hx	Hysteresis	PLSP[1].h2=0.1	Add a Hysteresis of 0.1 to the output 2	C-0-2936
PLSP[1].md <i>x</i>	Mode	PLSP[1].md2=0	Set mode of output 2 to Lag Time (0=Lag Time, 1=PT)	C-0-2934
PLSP[1].dr <i>x</i>	Direction	PLSP[1].dr2=0	Set direction of output 2 to positive (0=pos, 1=neg, 2=pos/neg)	C-0-2935
<i>x</i> = 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 Card PLS that can be easily modified using the Calc icon.

Syntax	Description	Variable Range	Example	Comment
PLS[x].a	Number	1-32 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		PLS[1].lt1=5	switch 1 lead time is 5
x = PLS number 1-2				

Drive PLS

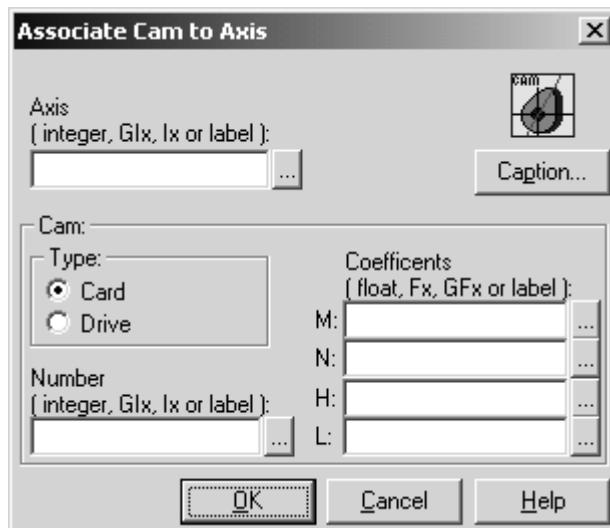
VisualMotion 8 supports drive based PLS configurations for DIAX04 and ECODRIVE03 digital drives. The drive based PLS is stored on the drive in parameter lists. The elements of this list cannot be modified individually. Modifying one element of the list requires sending the entire list over the service 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	PLSD[1].lt1=5	switch 1 lead time for drive 1 is 5	P-0-0134
x = drive number range 1-32				

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 Card resident.



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.

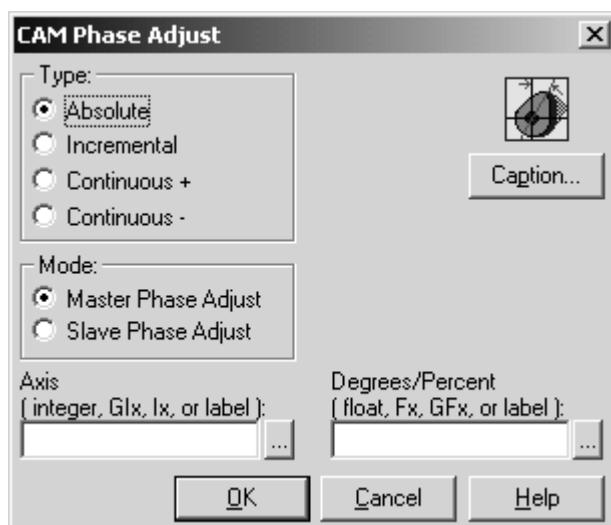
CAM Number: Specifies which CAM table to use. The entry may be an integer, global integer variable (Glx), program integer variable (Ix) or an equivalent label (Range 1 to 8 for control CAMs and 1 to 2 for drive CAMs.)

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).

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.



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 writes to axis parameter A-0-0151. For drive based CAMs, the Master Phase Adjust writes to drive parameter P-0-0061.

The Slave Phase adjusts the phase adjust value to the output position of the CAM table. For control CAMs, the Slave Phase Adjust writes to axis parameter A-0-0161. For drive based CAMs, the Slave Phase Adjust writes 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.

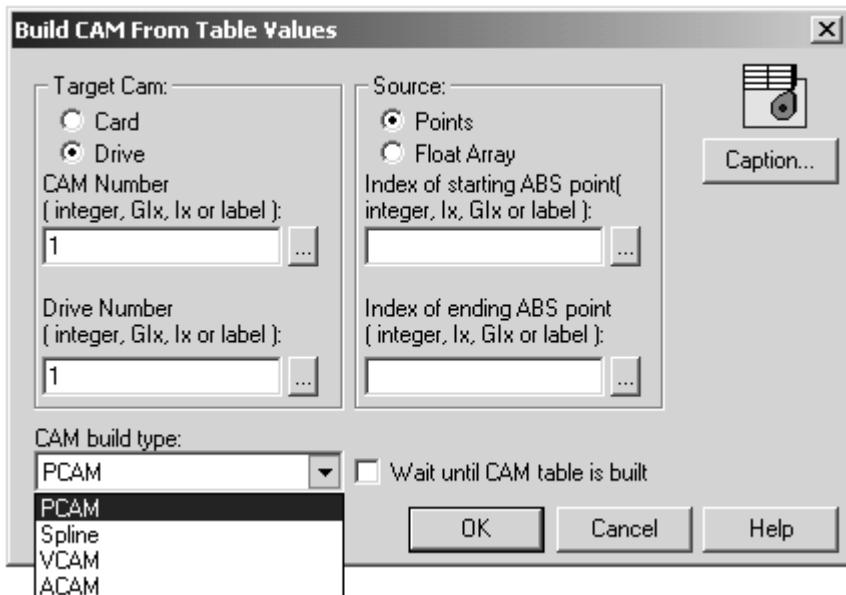
If **continuous +** or **continuous -** (phase sync mode only), a velocity dependent amount is added to the degree offset.

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.

CAMBuild



The control can build a CAM on-line based on a set of input positions. Program instructions use the control's ABS point table and an internal utility to build an internal control CAM which is stored on the Card or Drive as the CAM number indicated.



Target Cam:

From the Target Cam section, the user selects the location to store the control CAM. When **Card** 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 8 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 and outputs a normalized profile. 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.

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

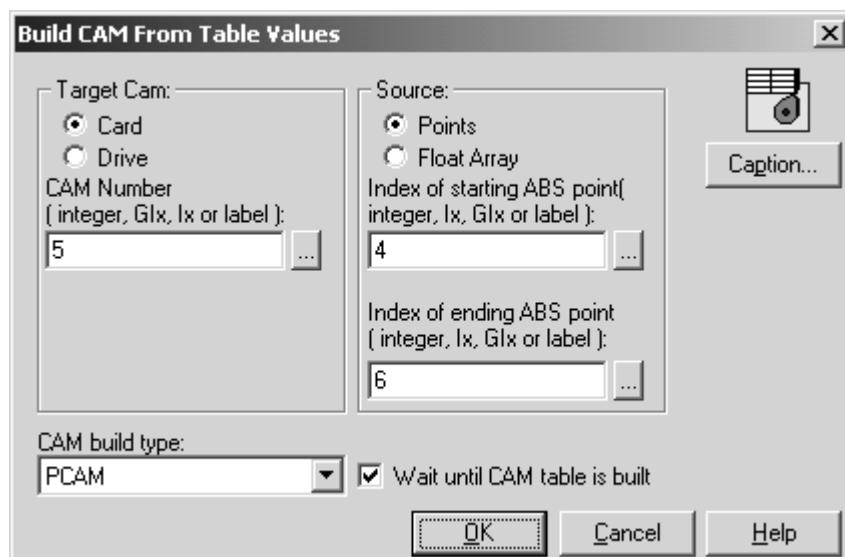
Specifies the starting ABS point number used for CAM generation. The entry may be an integer, global integer variable(Glx), program integer variable(Ix), or an equivalent label. The number must be in the range of points defined in the **Size** Icon.

ABS Ending Point

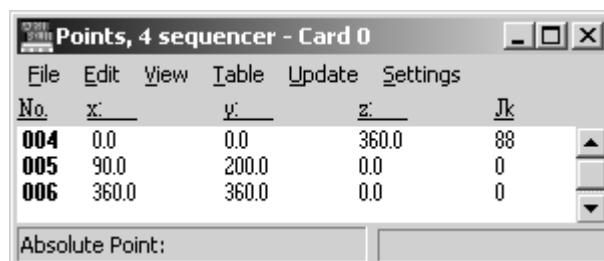
Specifies the ending ABS point number used for CAM generation. The entry may be an integer, global integer variable(Glx), program integer variable(Ix), or an equivalent label. The number must be in the range of points defined in the **Size** icon.

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.



Once compiled and downloaded to the control, the user can write to the points table by selecting **On-Line Data** \Rightarrow **Points** and modifying the points directly, or within the user program using the Calc icon.



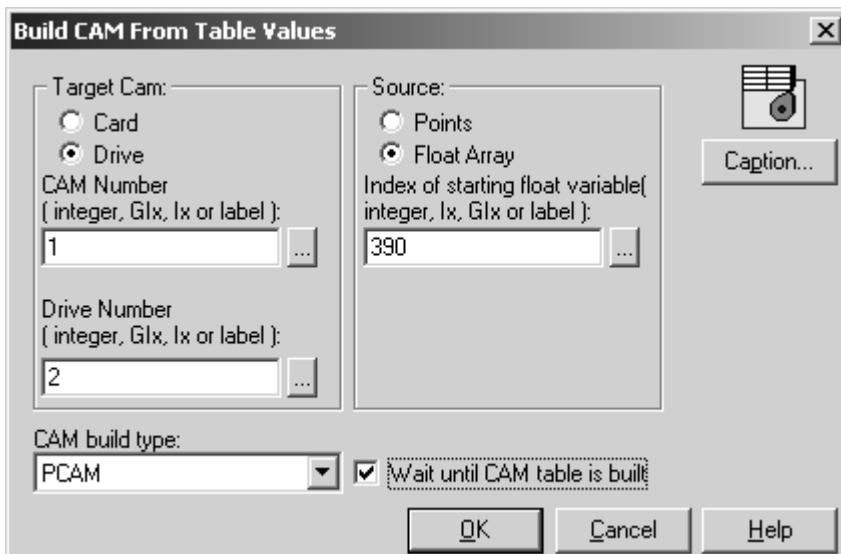
No.	X	Y	Z	Jk (Jerk)
004	First master position	First slave position	Modulo setting	Shaping Factor, 0 –100%
005	Second master position	Second slave position	not applicable	not applicable
006	Last master position	Last slave position	not applicable	not applicable

Table 8-5: Example Points Table Description

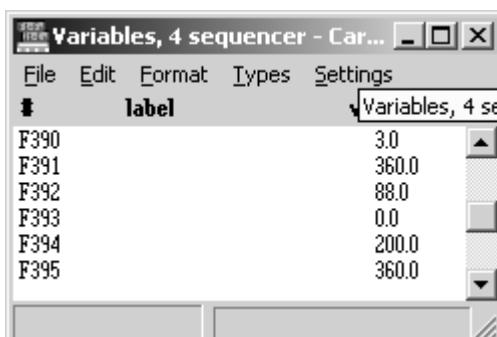
When selecting **Float Array**, the user must specify the *Index of starting float variable*. This number identifies the first float variable number that will be used to store the data for the float array table.

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.



Once compiled and downloaded to the control, the user can write to the float variables by selecting **On-Line Data** \Rightarrow **Variables** and modifying the variables directly, or within the user program using the Calc icon.



Float Variable	Description
F390	Number of equally spaced CAM values
F391	Modulo setting
F392	Shaping factor, 0 – 100%
F393	1 st CAM value
F394	2 nd CAM value
F395	3 rd CAM value

Table 8-6: Example Float Table Description

General Information

The ABS point elements may be changed from within the program using CALC statements. Changes in the point table do not affect the CAM until the CAM BUILD 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 CAM BUILD 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.

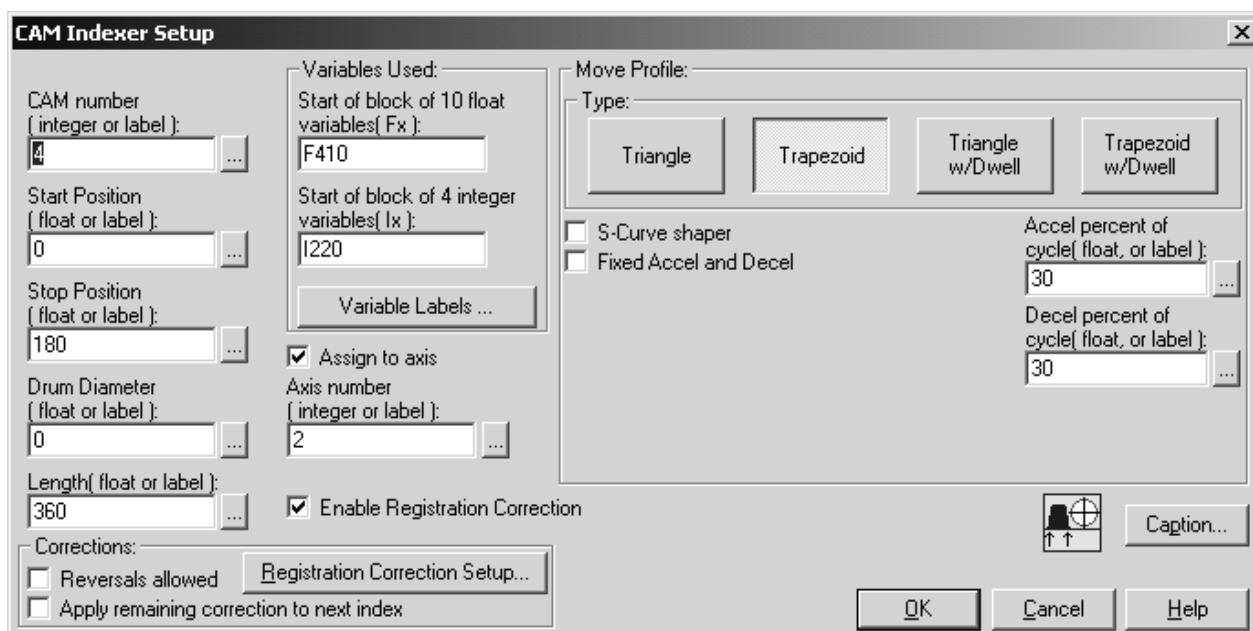
The x position exceeds the modulo of the master.

CAM Indexer



NOTE: When using Indexer CAMs, increase the SERCOS cycle time (C-0-0099) to 4ms or more to allow time for the equation calculations.

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.



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.

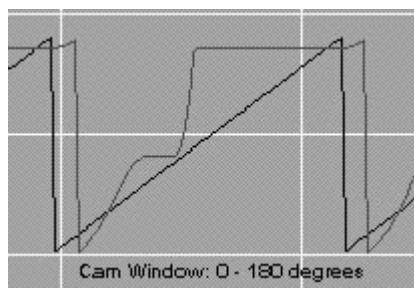
- GPP controls (PPC-R) limit the number of active running CAM Indexers to 4. GPP control systems can have a maximum of 40 built CAMs.

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, **Tc**,

$$Tc = ABS(Pe-Ps)/Vm$$

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:

$$\text{Angle} = (D/\text{Dia})(360/\pi)$$

Internal CAM units (degrees) are computed automatically through this parameter.

NOTE: When the Drum Diameter is set to zero, its default value is 360°/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:

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

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.

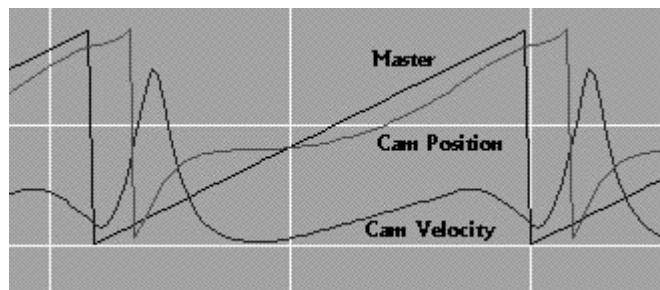
The Triangle profile without the S-Curve shaper has the simplest equation, thus it takes the least amount of time to execute, but has maximum jerk. Selecting the S-Curve shaper takes longer but minimizes jerk.

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.

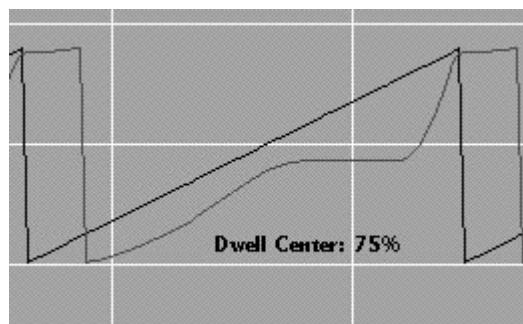
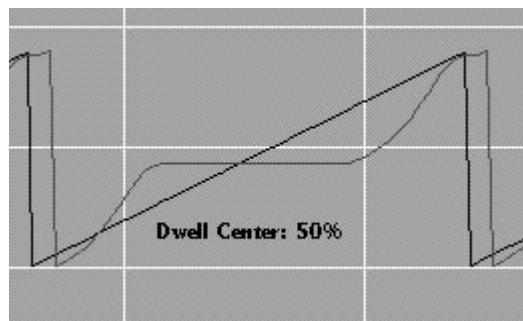
CAM Acceleration/Deceleration(Acc/Dec) - Some CAM types (see table) use these parameters to define the percent of their cycle time devoted to acceleration and deceleration.

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, slew for 30% and decelerate for 50%.

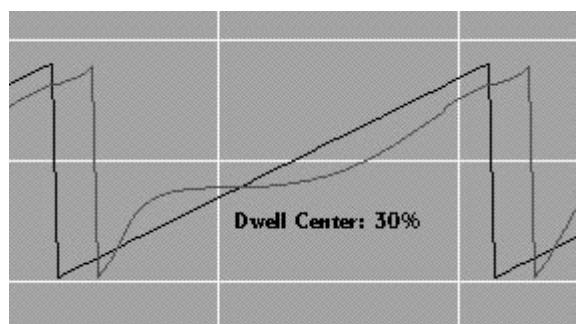


CAM Dwell (Cw) - Some CAMs (see table) use this parameter to determine the center of the dwell. This is defined as a percent, which is limited to 25% to 75% of the CAM cycle.

Example1: $C_w = 50\% \quad P_s = 0 \quad P_e = 360$ then dwell center
 $P_w = (360-0)*0.5 + 0 = 180 \text{ deg.}$



Example2: $C_w = 30\% \quad P_s=90 \quad P_e=270 \text{ then dwell center}$
 $P_w = (270-90)*0.3+90 = 144 \text{ deg.}$



CAM Dwell (Tw) - 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. Tw 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

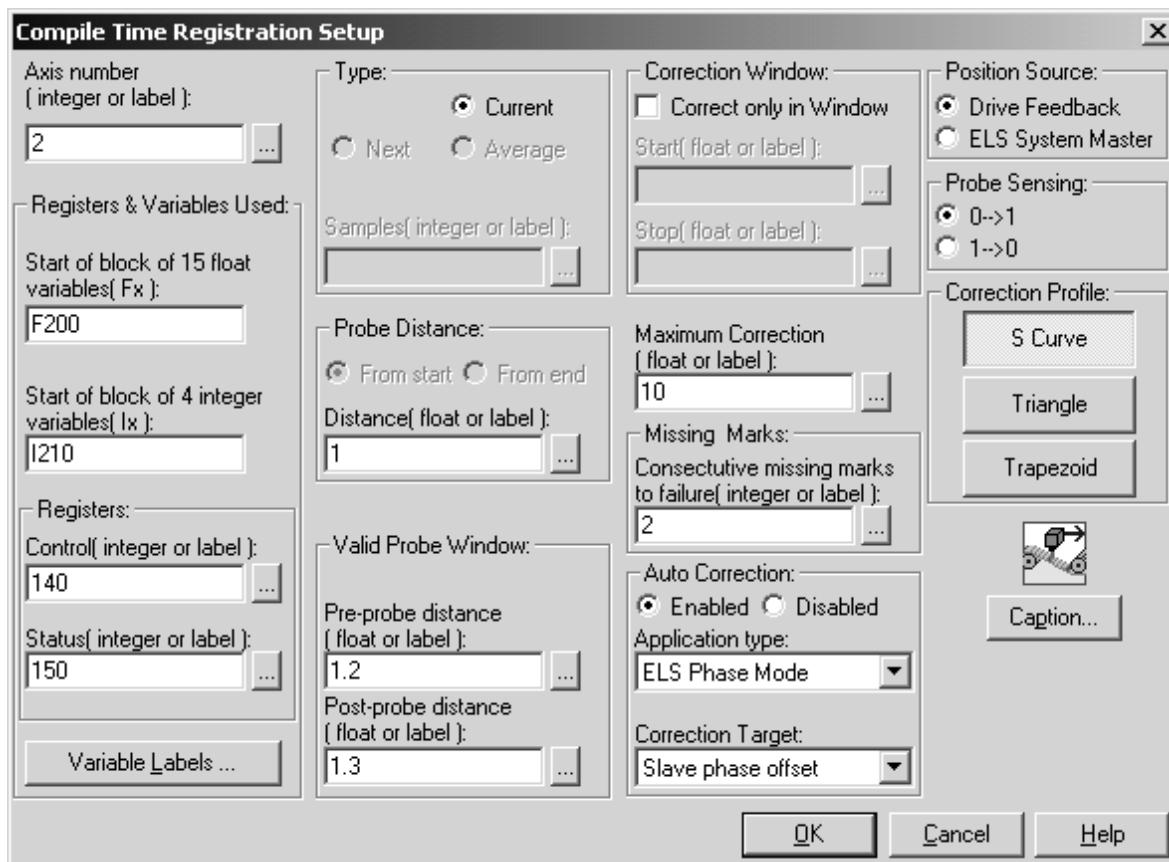
This enables the auto-registration function for the indexed CAM.

[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.



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 4 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 Indramat 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 card 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

Parameters	Description	Variable	Written by
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 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.

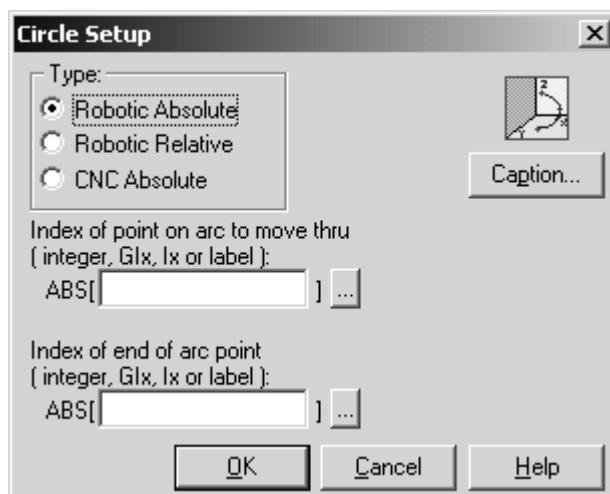
Max Missed Marks

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.



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.

CNC Move (Reserved for future use)

Move in a circular arc in the defined plane. Start of the arc is the current position. Center of the arc is C (I,J) in the defined plane. End of the arc is the intersection of the arc and a line from the center to end point E (X,Y), or the number of degrees specified from the start.

The center point, end point, degrees, direction, feed rate, and defined plane are defined in the ABS point structure as follows:

X	Y....	Z.....	S	R	P	W	EI
I	J.....	DIR...	F	DEG	X	Y	1=x-y plane
I	J.....	DIR...	F	DEG	X	Y	2=z-y plane
I	J.....	DIR...	F	DEG	X	Y	3=y-z plane

The arc's radius is the distance from start to center.

The arc's direction is:

DIR > 0 - Clockwise.

DIR <= 0 - Counterclockwise.

The end of the arc is:

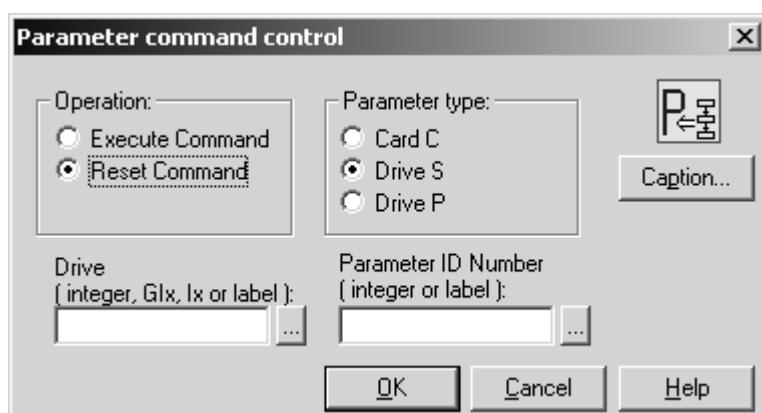
DEG=0 -The intersection of the arc and a line from the center to end point.

DEG>0 - The number of degrees from a line through start and center points.

Command



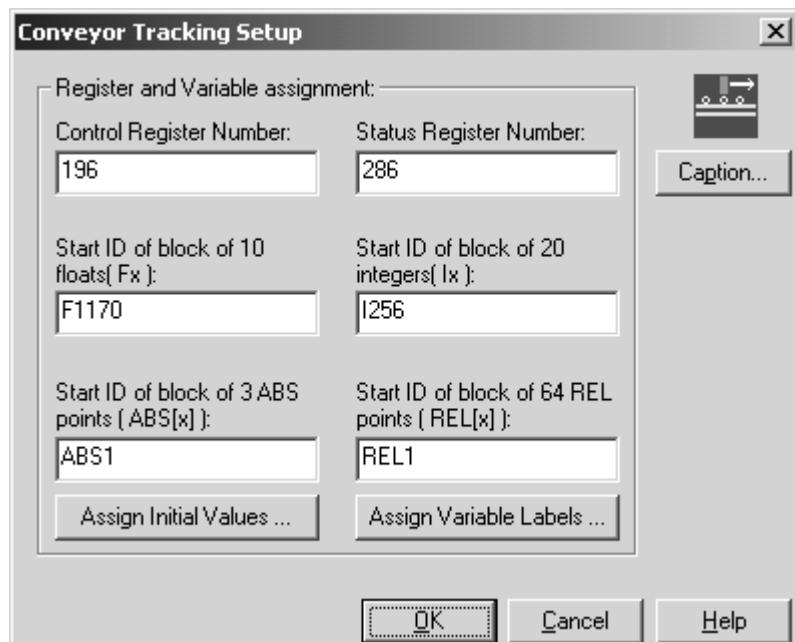
The Command icon is used to initiate drive-resident commands.



CvyrInit (Conveyor Tracker Setup)

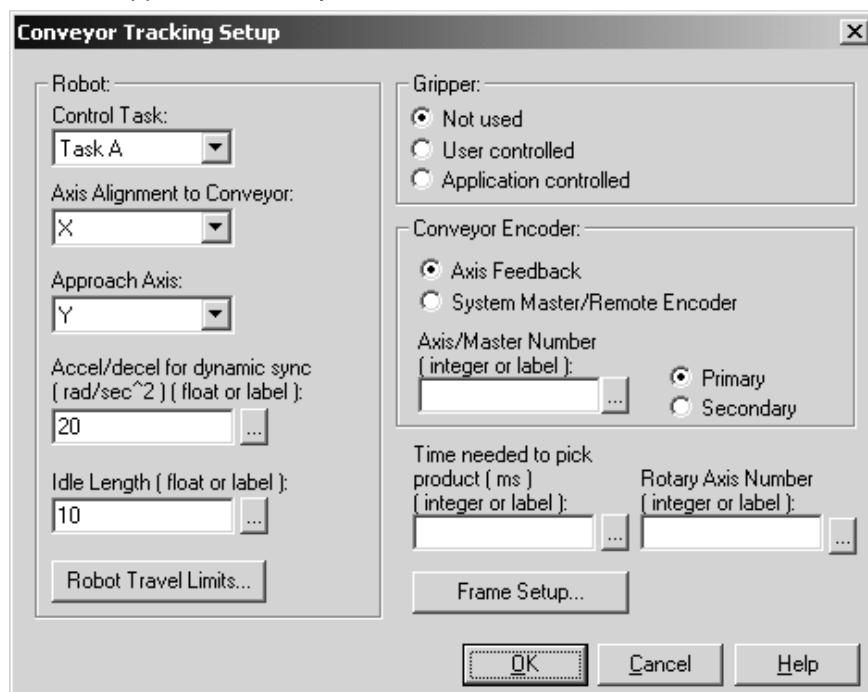


The CvyrInit icon is used to setup a conveyor tracker. Refer to the Conveyor Tracker description in the “VisualMotion Toolkit Menu Commands” section for details. The Conveyor Tracker Setup window is used to enter register and variable assignment for Control and Status.



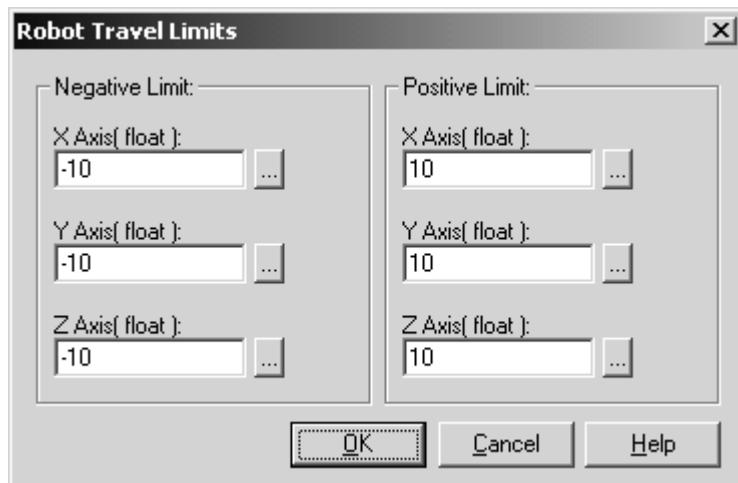
Assign Initial Values ...

The **Assign Initial Values** button is used to initiate runtime data for the Robot, Gripper and Conveyor Encoder.

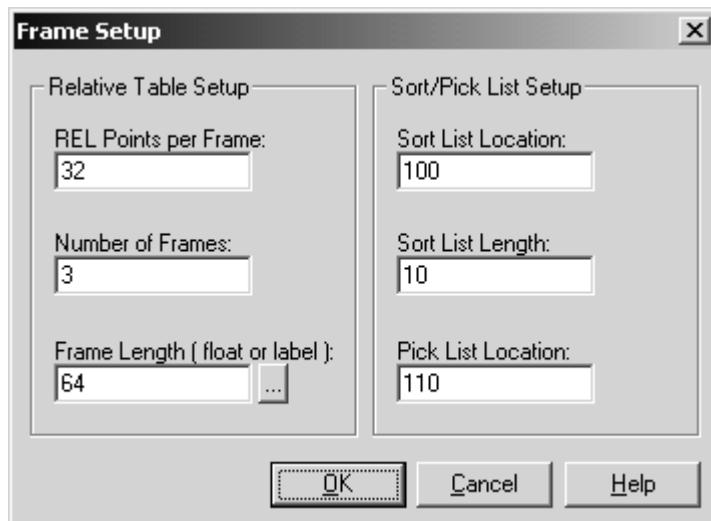


Robot Travel Limits...

The **Robot Travel Limits** button opens the Robot Travel Limits window, which contains menu selections for configuring the negative and positive travel limits of the robot.

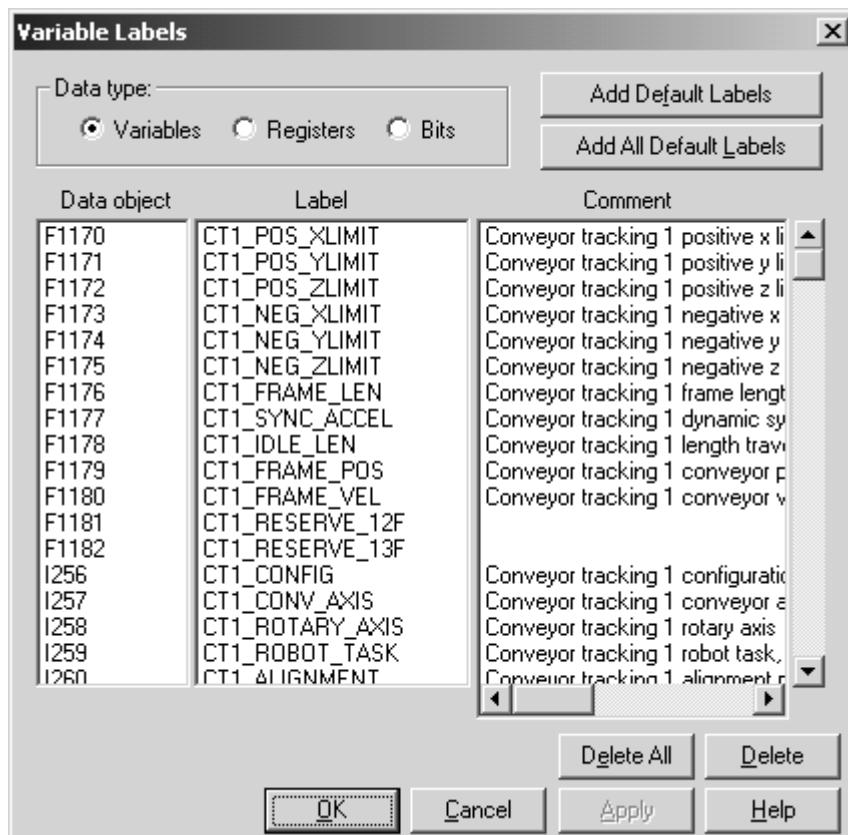
**Frame Setup...**

The **Frame Setup** button is used to setup the relative table and sort/pick list.



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.



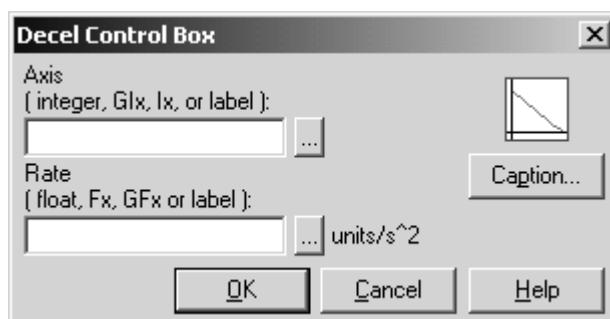
Decel



Note: The Decel icon is used in GPS firmware to decelerate a Virtual Master (Axis 0) in ELS mode. This icon remains in VisualMotion for support of GPS firmware types. GPP7 and GPP8 do not support the Decel icon. Deceleration of ELS System Masters is performed via predefined variables.

Deceleration rate - specifies the deceleration rate in units per second squared for a Virtual Master.

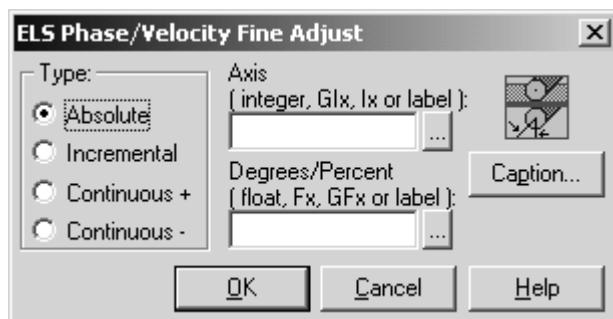
A pop-up User Defined Labels window is available for the data entry fields by placing the focus ("I" beam cursor) in an entry field, then clicking the User Defined Labels button to the right.



ELSAdj



The ELSAdj 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. A pop-up User Defined Labels window is available from both data entry boxes.



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:

$$\emptyset_S = (\emptyset_m * (K_s/K_m)) + \text{adjust}$$

For velocity synchronization the resulting slave velocity is:

$$V_S = V_m * ((1 + \text{adjust}) * (K_s/K_m))$$

Where:

\emptyset_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.

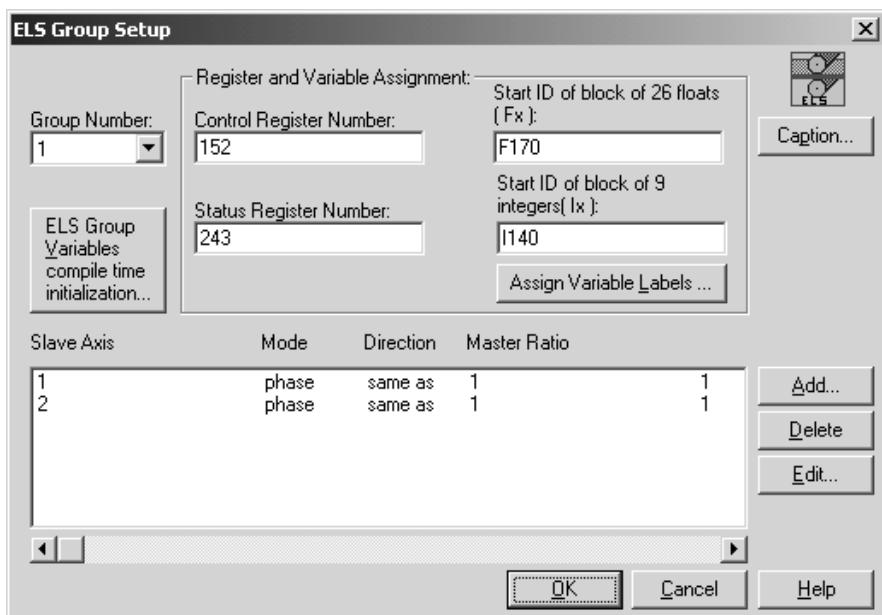
Degrees or Percent

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%.

ELSGrp



GPP supports a maximum of eight ELS Groups. The initialization of the ELS Group's registers and program variables is defined in the ELSGrp setup icon. The following window appears when the ELSGrp icon is placed:



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 programs 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.

ELS Group Control Register Default Label	ELS Group Default Control Register and Bit								ELS Group Control Register Default Comment (80 character limit)
	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

Each # symbol represents an entry for the number of the ELS Group

Table 8-7: 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.

ELS Group Status Register Default Label	ELS Group Default Status Register and Bit								ELS Group Status Register Default Comment (80 character limit)
	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

Each # symbol represents an entry for the number of the ELS Group

Table 8-8: ELS Group 1-8 Default Status Register Bits

ELS Group Program Variable Start ID Blocks

Start ID of block of 26 floats (Fx):

This number represents the first float in a block of 26 floats set aside for the selected ELS Group. The float number must be preceded with an "F".

Example: F170

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	F170	F200	F230	F260	F290	F320	F350	F380	Group #, dynamic sync acceleration	Phase 4
G#_SYNC_VEL	F171	F201	F231	F261	F291	F321	F351	F381	Group #, dynamic sync velocity	Phase 4
G#_M1	F172	F202	F232	F262	F292	F322	F352	F382	Group #, M factor	Phase 4 & Forcing
G#_N1	F173	F203	F233	F263	F293	F323	F353	F383	Group #, N factor	Phase 4 & Forcing
G#_REL_M_PH	F174	F204	F234	F264	F294	F324	F354	F384	Group #, relative master phase adjust	Phase 4
G#_REL_S_PH	F175	F205	F235	F265	F295	F325	F355	F385	Group #, relative slave phase adjust	Phase 4
G#_ABS_M_PH	F176	F206	F236	F266	F296	F326	F356	F386	Group #, absolute master phase adjust	Phase 4 (read-only)
G#_ABS_S_PH	F177	F207	F237	F267	F297	F327	F357	F387	Group #, absolute slave phase adjust	Phase 4 (read-only)
G#_H_LOCKON	F178	F208	F238	F268	F298	F328	F358	F388	Group #, H factor lock on cam profile	Phase 4 & Forcing
G#_H_RUN	F179	F209	F239	F269	F299	F329	F359	F389	Group #, H factor 1:1 cam profile	Phase 4 & Forcing
G#_H_LOCKOFF	F180	F210	F240	F270	F300	F330	F360	F390	Group #, H factor lock off cam profile	Phase 4 & Forcing
G#_H_USER	F181	F211	F241	F271	F301	F331	F361	F391	Group #, H factor user cam profile	Phase 4
G#_LOCK_WIN	F182	F212	F242	F272	F302	F332	F362	F392	Group #, shortest path window for dynamic sync. phase correction	Phase 4
G#_STOP_DECEL	F183	F213	F243	F273	F303	F333	F363	F393	Group #, stop ramp deceleration	Phase 4
G#_JOG_ACCEL	F184	F214	F244	F274	F304	F334	F364	F394	Group #, jog acceleration	Phase 4
G#_JOG_VEL	F185	F215	F245	F275	F305	F335	F365	F395	Group #, jog velocity	Phase 4
G#_JOG_INC	F186	F216	F246	F276	F306	F336	F366	F396	Group #, relative position distance (incremental jog)	Phase 4
G#_JOG_ABS	F187	F217	F247	F277	F307	F337	F367	F397	Group #, absolute position target (absolute jog)	Phase 4
G#_JOG_WIN	F188	F218	F248	F278	F308	F338	F368	F398	Group #, shortest path window for absolute jog	Phase 4
G#_LOCKON_OFFSET	F189	F219	F249	F279	F309	F339	F369	F399	Group #, offset added to the output when lock on cam profile is being forced	Phase 4
G#_IN_POS	F190	F220	F250	F280	F310	F340	F370	F400	Group #, input position	Phase 4 & Forcing
G#_IN_VEL	F191	F221	F251	F281	F311	F341	F371	F401	Group #, input velocity (read only)	Phase 4
G#_OUT_POS	F192	F222	F252	F282	F312	F342	F372	F402	Group #, output position (read only)	Phase 4 & Forcing
G#_OUT_VEL	F193	F223	F253	F283	F313	F343	F373	F403	Group #, output velocity (read only)	Phase 4
G#_OUT_ACC	F194	F224	F254	F284	F314	F344	F374	F404	Group #, output acceleration (read only)	Phase 4
G#_CAM_INPUT	F195	F225	F255	F285	F315	F345	F375	F405	Group #, group cam profile ID input position	Phase 4 & Forcing

Table 8-9: 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 the selected ELS Group. The integer number must be preceded with an "I". **Example:** I170

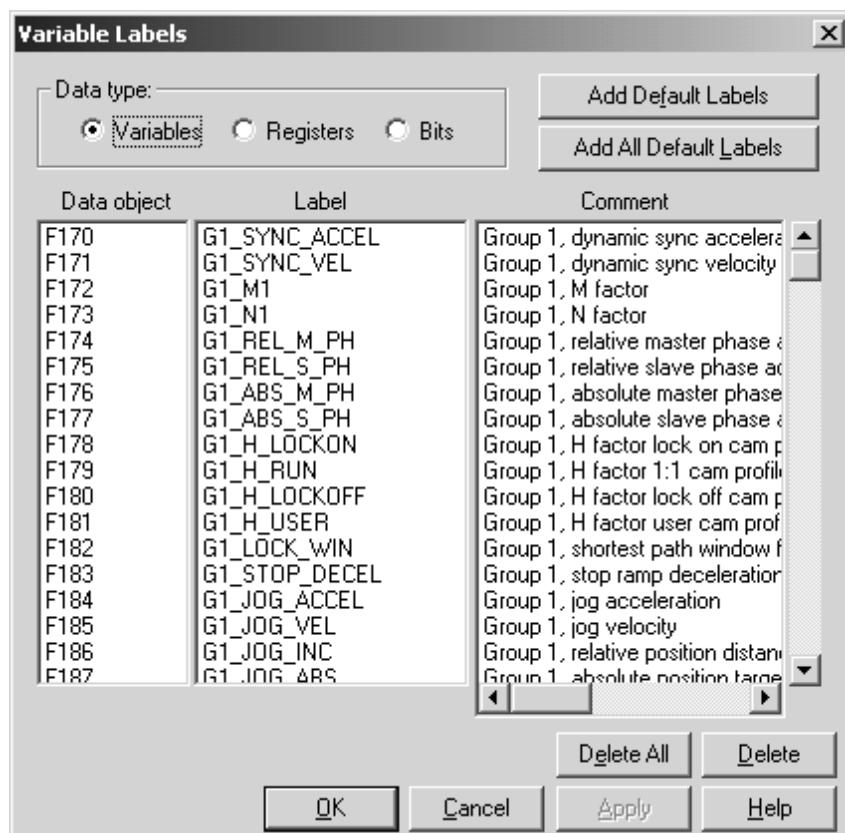
Default integer labels and numbers for all 8 ELS Groups are listed in the table below.

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	Refer to Fig. 5-2
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

Table 8-10: ELS Group 1-8 Default Integer Variables

Assign Variable Labels

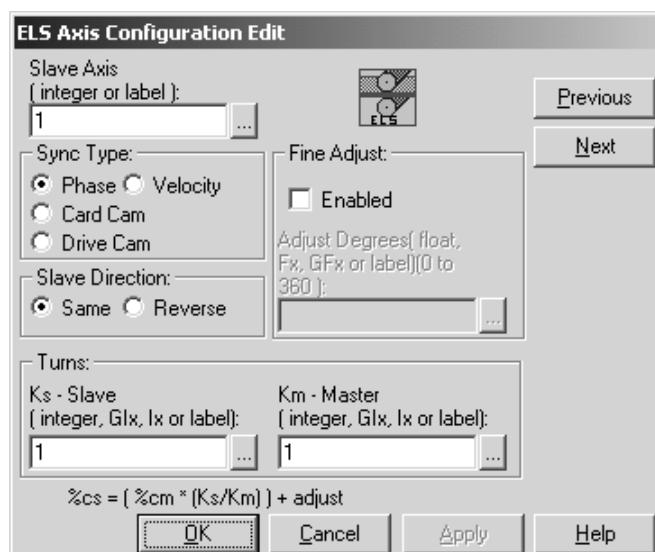
Default variable labels and comments can be added individually for Variable, Registers and Bits by selecting the appropriate Data Type radio button and clicking the **Add Default Labels** button. Clicking the **Add All Default Labels** button will add the default variable labels and comments to all Data Types at one time.



ELS Slave Axis Configuration

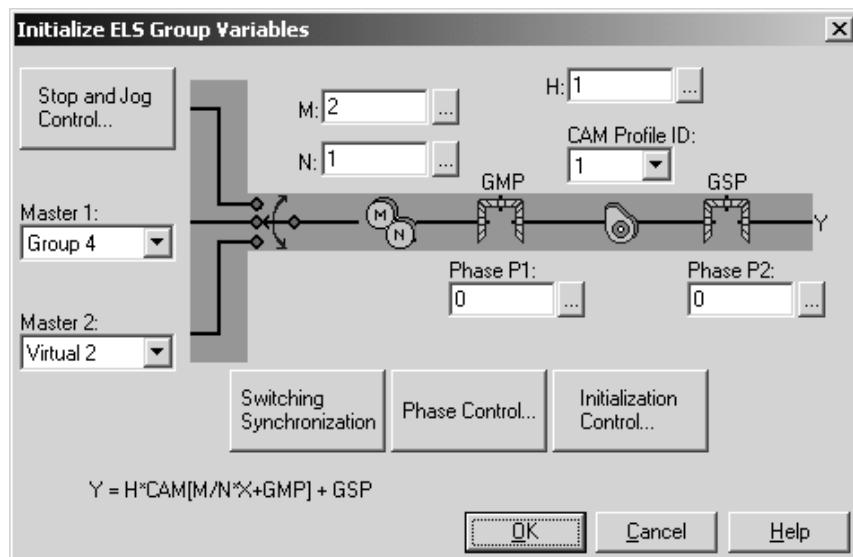
Clicking on the **Add** button in the *ELS Group Setup* window opens the *ELS Axis Configuration Edit* window.

Refer to [**ELS Axis Configuration**](#) in chapter 5 of the VisualMotion 8 Application Manual for a functional description.



ELS Group Compile Time Initialization

Clicking the "ELS Group Variables and Compile-Time Initialization" button opens the **Initialize ELS Group Variables** window below:



Refer to [**ELS Group**](#) in chapter 5 of the VisualMotion 8 Application Manual for a functional description.

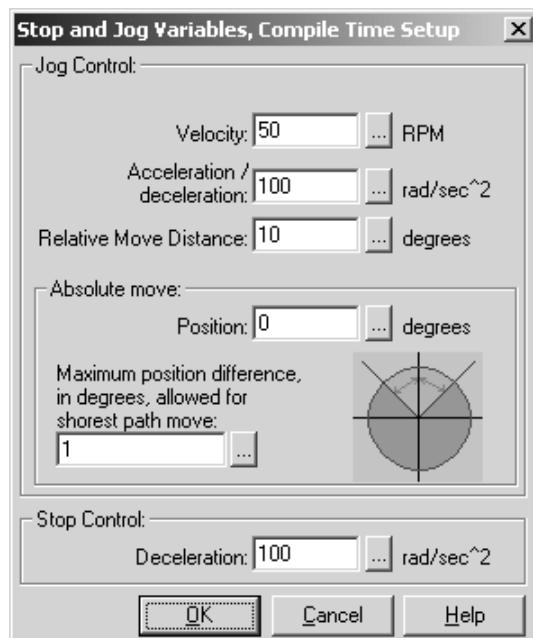
Clicking the variable label button to the right of M, N, H, Cam Profile ID, Phase P1 or Phase P2 allows selection of a variable for those values.

From this window, four main buttons are provided to access setup screens:

- Stop and Jog Control
- Switching Synchronization
- Phase Control
- Initialization Control

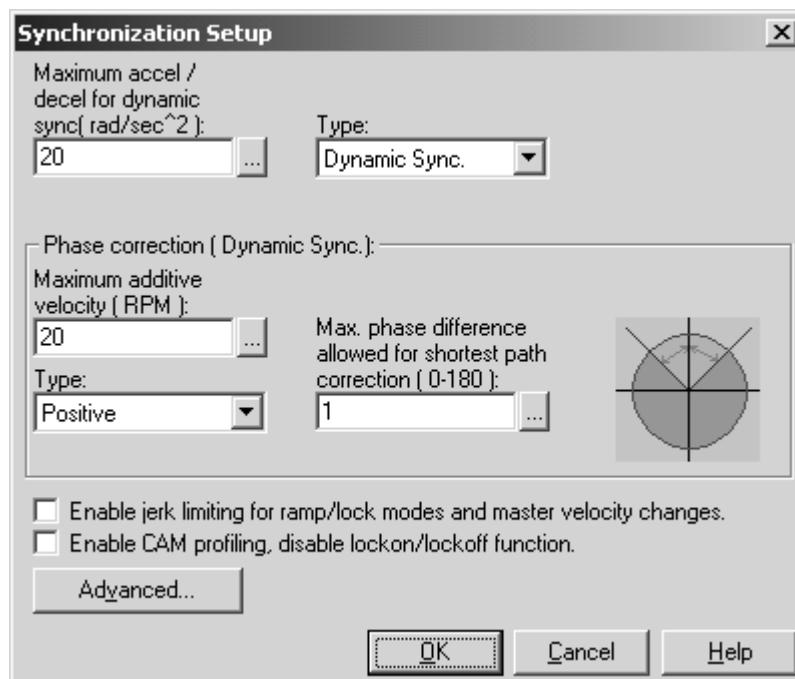
Stop and Jog Control

Refer to [Stop and Jog Control](#) in chapter 5 of the VisualMotion Application Manual for a functional description.



Synchronization Setup

Refer to [Synchronization Setup](#) in chapter 5 of the VisualMotion Application Manual for a functional description.



Maximum Acceleration/Deceleration for Dynamic Sync. This value is the maximum acceleration or deceleration that the ELS Group will use to ramp up to the new master's velocity and perform any phase corrections with a trapezoidal velocity profile.

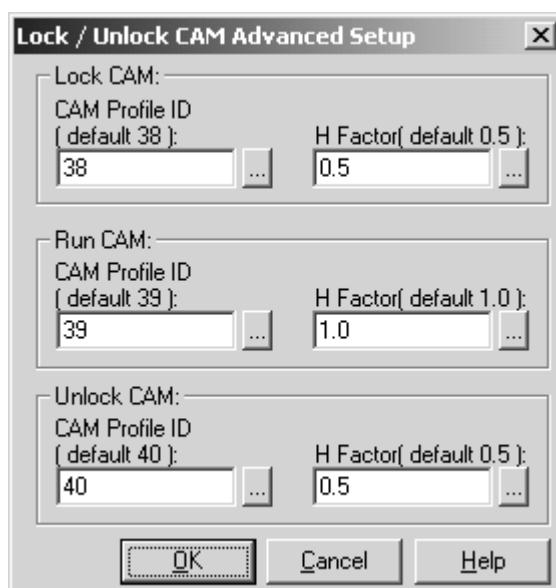
Note: The maximum acceleration and deceleration value is only used for dynamic synchronization.

Type: This selection determines the method for switching between ELS Group input masters. GPP supports two switching methods as follows:

- Immediate Switching
- Dynamic Synchronization

Lock / Unlock CAM Advanced Setup

This window displays the default CAM numbers used for the ELS Lock On / Lock OFF function. The user can modified the default settings with CAM numbers and H factors that have been designed for their specific application. Refer to Synchronized "Lock On / Lock Off" of ELS Group Master in chapter 5 of the VisualMotion 8 Application Manual for a functional description of this feature.



Phase Correction

Note: The following phase corrections are only available for dynamic synchronization.

Maximum Additive Velocity: Specifies the maximum increases or decreases in velocity allowed for matching the phase (position) of target master.

Type: Specifies the direction in which the phase correction will be made. The user can select *shortest path*, *positive*, *negative* or *no phase correction*.

**Maximum Phase Difference
"Monitoring Window"** When selecting a positive or negative direction, a value ($\pm 0\text{-}180$ degrees) is entered which creates a range (monitoring window) around the position of the target master. If any phase errors are within this window, shortest path will be used for the correction.

This allows the user to eliminate large phase corrections depending on the size of the window.

Phase Control

The group master and slave phase adjust defaults to a trapezoidal velocity profile using:

- dynamic synchronization acceleration
- dynamic synchronization velocity

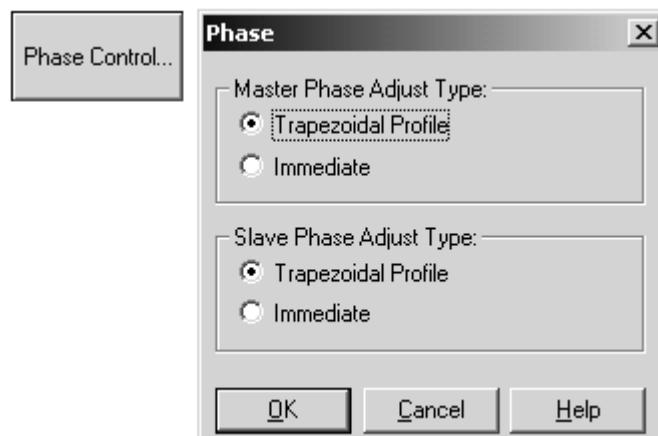
The desired relative phase adjust are written to the following ELS Group float variables:

- G#_REL_M_PH (group master relative master phase adjust)
- G#_REL_S_PH (group master relative slave phase adjust)

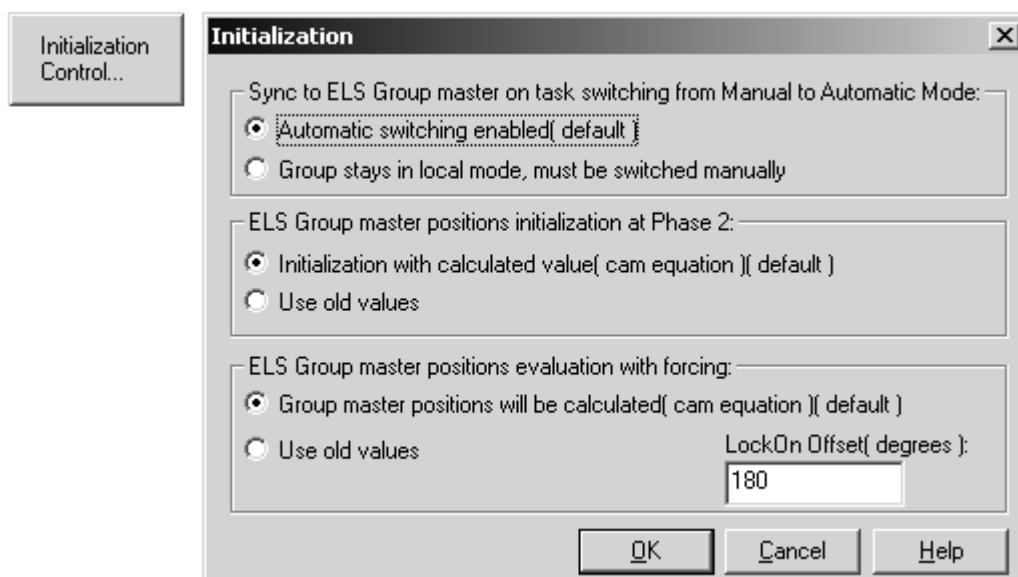
and triggered with bit 2 or 3 in the ELS group control register. After execution of the velocity profile, the absolute group master or group slave phase adjust is updated. The phase adjust can also be configured to be executed in one step.

The absolute phase adjust values can only be read. The exception would be when the stop ramp is active and the group master is at standstill. In this case the absolute phase adjust values can be overwritten and forced.

If an ELS group is switched to local, manual or parameter mode during a phase adjust with a trapezoidal velocity profile, the phase adjust will be completed.



Initialization Controls

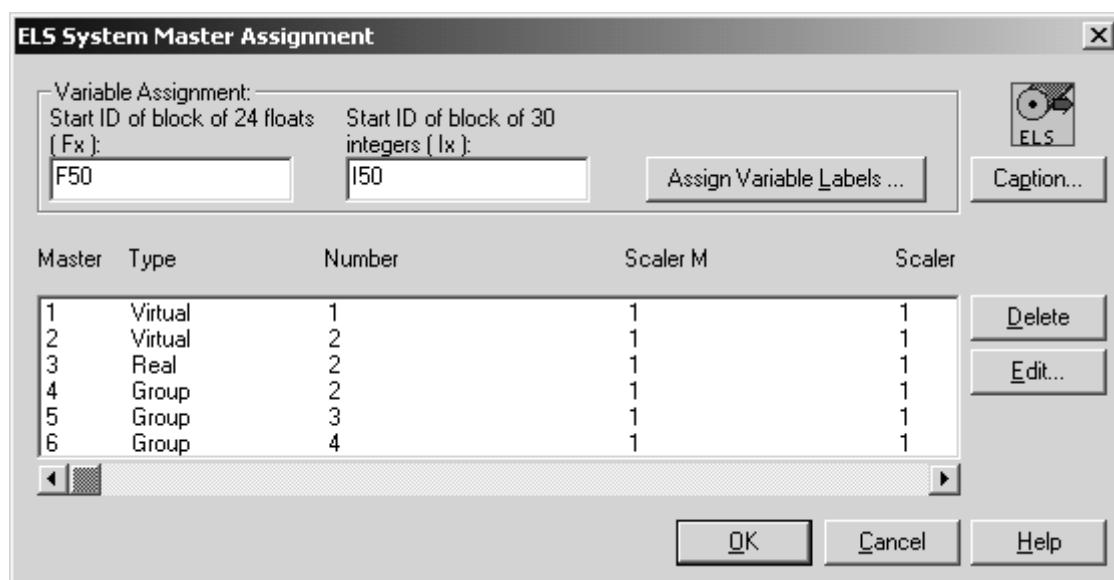


ELSMstr



The ELSMstr icon allows for assignment of up to six master axes.

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.



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.

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 programs an easier task over the scope of the project.

ELS System Master Program Variable Start ID Blocks

Start ID of block of 24 floats (Fx):

This number represents the first float in a block of 24 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 9 integers set aside for the selected ELS Group. 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.

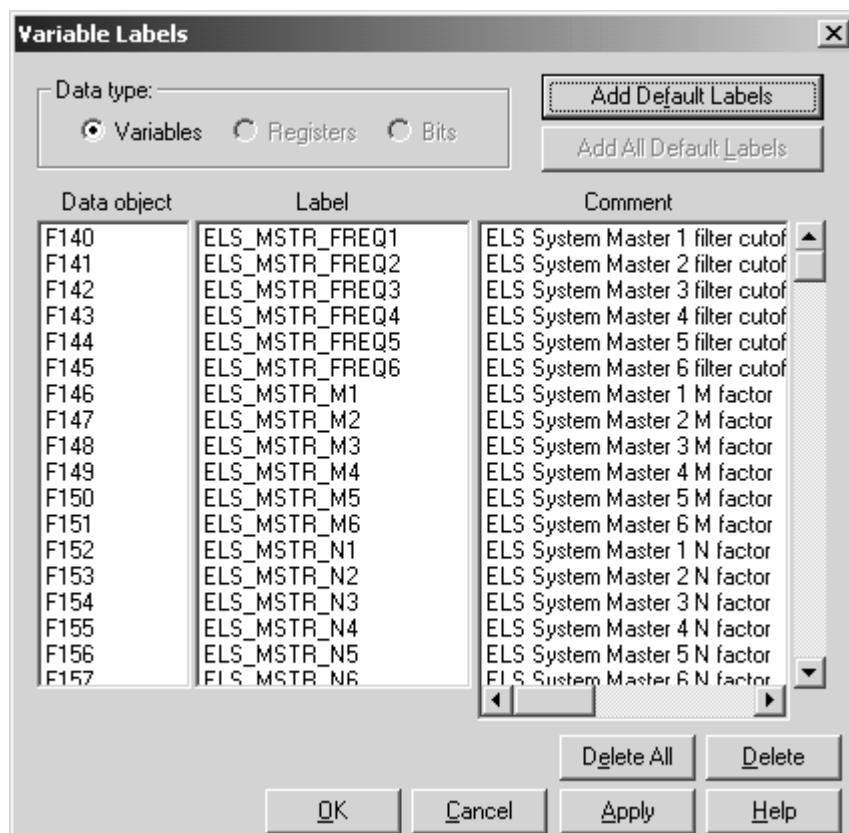
ELS System Master Program Variable Default Label	ELS System Master Program Variable						ELS System Master Program Variable Default Comment (80 character limit)	Update Mode
	1	2	3	4	5	6		
ELS_MSTR_FREQ#	F140	F141	F142	F143	F144	F145	ELS Master # filter cutoff frequency	Phase 2
ELS_MSTR_M#	F146	F147	F148	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_RS_FT#	F158	F159	F160	F161	F162	F163	ELS Master # reserve float	Phase 2
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_RSVD#	I134	I135	I136	137	I138	I139	ELS Master reserve integer	Phase 2

Table 8-11: ELS System Master Program Variables

Refer to [ELS Master Variable Definition](#) in chapter 5 of the VisualMotion 8 Application Manual for details.

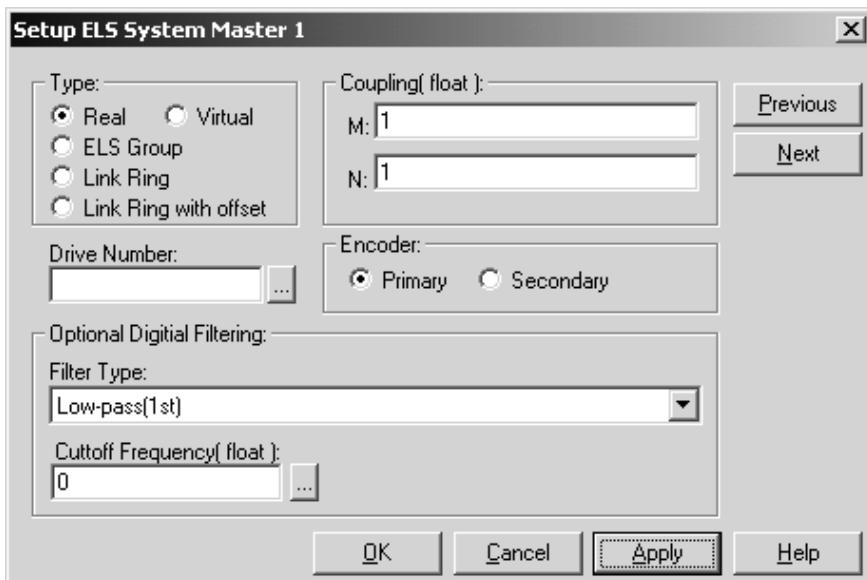
Assign Variable Labels

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.



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 Master* window are dependent upon the type of master type selected.



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: Real Masters can be primary encoders that are not slaves of an ELS Group or secondary encoders.

- **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, geared by an M/N ratio, 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.
- **Link Ring with offset:** This feature is not supported in GPP8.

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.
- **Link Ring with Offset** – this feature is not supported in GPP8.

Coupling (float) (for Real Master and ELS Group only)

The M/N ratio is only available for Real Masters and ELS Group outputs. The positional value for either a Real Master or ELS Group 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 receive 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 receive a position value of 90°.

Encoder

Select the encoder type when setting up a Real Master.

Optional Digital Filtering

Digital filtering is only applicable for 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.

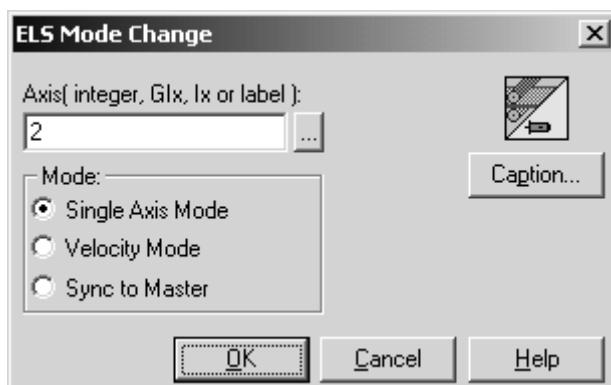
ELSMode



The ELSMode icon is used to switch between ELS, Single Axis and Velocity 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 ELS Mode Change icon is used to change an axis from Single Axis mode to Sync to Master mode, a second ELS 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 put command velocity into the cyclic data. Axis parameter **A-0-0004, bit 7** must be set to 1 to enable acceleration. The Sync to Master mode is then ignored.



The axis is specified by an integer constant, variable, global variable or an equivalent label. Axes must be configured as ELS axes. A pop-up User Defined Labels window is available from both data entry boxes. Entering "-1" in the axis box will send the mode change command to all ELS axes defined in the Task where the command is issued.

When an axis that is configured for ELS 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.

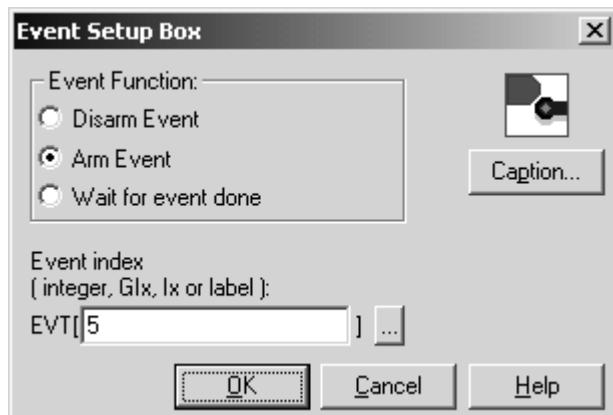
Care should be taken when switching an axis into synchronization with a moving master. DIAX04 drives have the "ramp up and lock on" feature that assures smooth acceleration when synchronizing to a moving master.

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 decel and stop icons.

Event



The Event icon is used to control the way that the system handles events; and may be used to control program flow by suspending a task's program execution pending the completion of an event. For the event icon to have an effect upon system operation, an event must have been previously entered in the control event table. Placing an Event icon on a task or subroutine workspace automatically opens an Event window.



The event is specified by entering an event ID number indicating a valid event in control's event table. The event may be entered as an integer constant, variable (Ix), global variable (Glx), or an equivalent label.

An event icon may have one of three effects:

Disarm Event de-activates the specified event. If the event has already been made inactive, the icon has no effect.

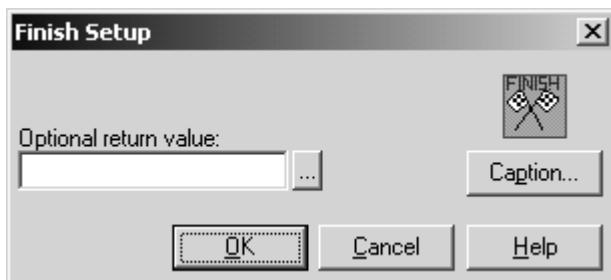
Arm Event is valid only for timed events. An Arm Event icon starts the specified event's timer. If the event is currently active, the icon has no effect.

Wait for event done suspends the execution of program flow until the specified event has completed.

Finish



Each program task, subroutine and event function must end with a single Finish icon. Subroutines can return an optional single argument 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 utilizing a common subroutine from more than one task. The return argument only works when the **Sub** icon requests it.

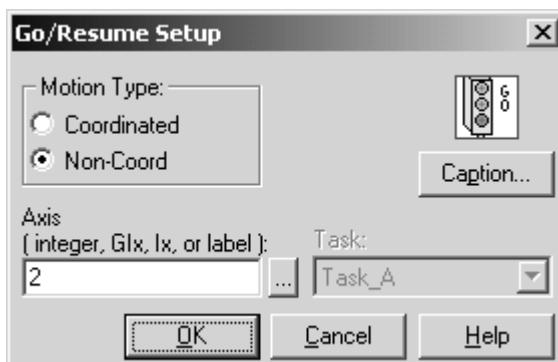


Go



The Go 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 Move icon. The Go icon can also be used to resume multi-axis coordinated motion that has been stopped.

Placing a Go icon automatically opens the Go Setup window. The **Motion Type** radio button specifies the type of motion to start. Choosing Single Axis Motion Type requires an entry in the **Axis** data entry box specifying the control axis to start. The axis is specified by a valid integer constant, variable, global variable or an equivalent label. Specifying "-1" as the single axis enables all the single axes assigned to the current task in the Axis icon. Specifying "0" starts the virtual master of an ELS system.



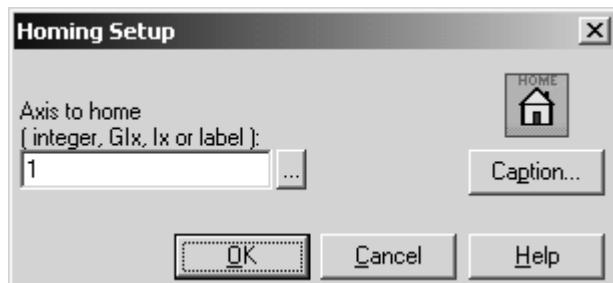
Choosing Coordinated Motion Type to resume motion requires selection of one of the four control tasks from the pull-down **Task** pick list. The coordinated motion must have been halted by a coordinated stop. Aborted coordinated motion should not be simply resumed.

A pop-up window of User Defined Labels is available.

Home



The Home icon commands the control to send an axis home signal to the specified drive. This is a single-axis non-coordinated motion command. Placing a Home icon on a task or subroutine workspace automatically opens the Homing Setup window.



The axis to home may be specified by a valid integer constant, variable (Ix), global variable (Glx), or an equivalent label. A pop-up window of User Defined Labels is available.

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 operation are reported to the control. The user program waits in this icon until the homing sequence is complete.

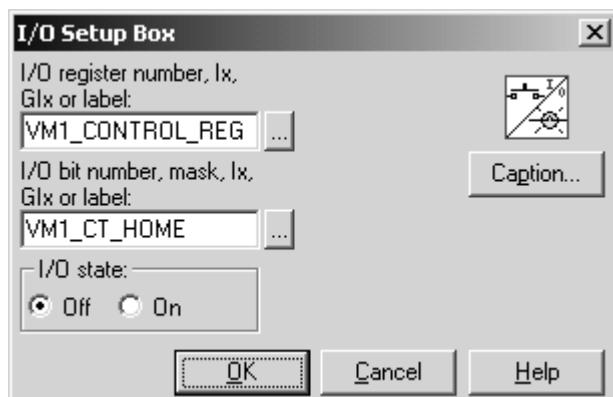
The Home icon uses the internal homing capability of the intelligent digital drive to perform the homing operation. For homing to occur, the homing parameters in the specified drive must have been setup prior to executing the Home icon. The homing parameters may be set using the Homing Setup window accessed through the Drives menu item under Setup. Refer to [Drive Reference](#) menu selection under [Parameters](#) in the *control Drive Parameter Editor*. Refer to the respective Drive manual for more information about homing and the homing parameters.

The Home icon cannot be used with multi-turn feedback. To reference multi-turn feedback, refer to the [Drive Parameter Editor](#).

I/O Setup



The I/O icon is used to control the state of I/O register bits. Placing an I/O icon on a task or subroutine workspace automatically opens an I/O Setup window. Entering a number as a register ID code or an equivalent label in the I/O Register data entry box selects the target I/O register.



The **I/O Register** data entry box specifies the target register number. The entry can be a number, an integer variable, or a defined register label.

The **I/O Bit mask** data entry box 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). A pop-up window of Bit Labels is available for adding, editing, or deleting register bit labels.

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).

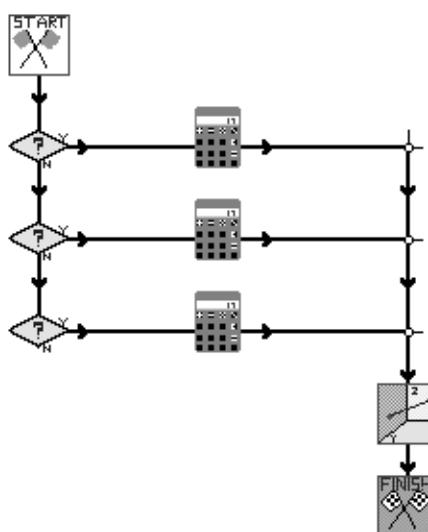
A pop-up window of Register Labels is accessible, and permits adding, editing or deleting register labels. The Register Labels window provides a scrolling list of default labels for the standard control system, axis, task, and digital drive I/O card control and status registers.

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

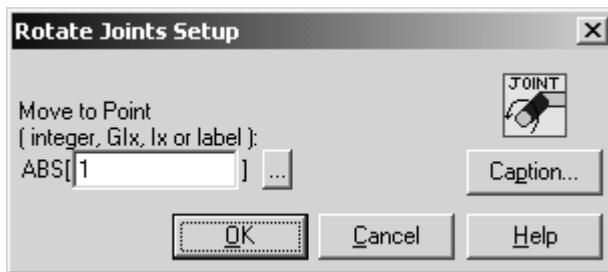


The Joint icon is used for point-to-point movement and joint (elbow) positioning, typical to robotic motion. It changes one or more motor angles from the current set J0 to a new set J1 defined by a absolute(ABS[x]) point. This instruction is only for coordinated motion whose kinematic supports joint 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 accel and decel 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.



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.

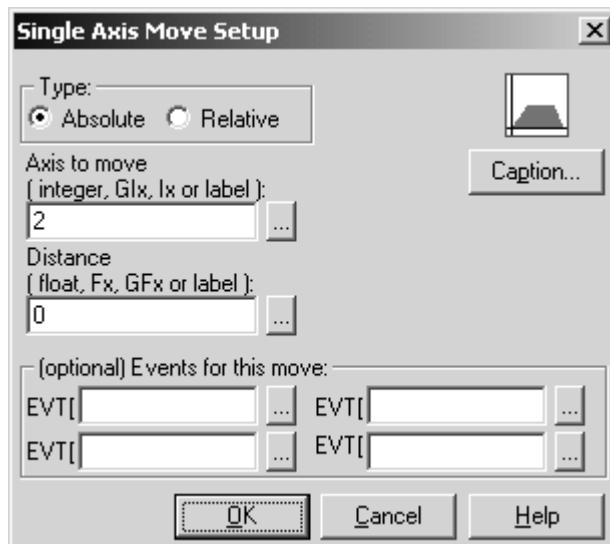
Under some circumstances, VisualMotion may be unable to route a line and displays a "Connection could not be made, try connecting adjacent blocks" window. Lines may be manually routed by clicking adjacent empty squares on the invisible workspace grid from the first icon to the second. A manually placed 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 from the Utility palette. Position the Scissors over the line and click the left mouse button.

Move



The Move icon is used to program movement on any single non-coordinated axis from any task. The Move icon initiates motion only if the axis has been enabled previously with a "GO" icon. Placing a Move icon on a task or subroutine workspace automatically displays a Single Axis Move Setup window.



The **Type** radio buttons specifies either an incremental move distance added to the current position (Relative) or a move to an absolute position (Absolute).

The **Axis to move** data entry box specifies the axis to move. It accepts an integer number, constant or variable.

The **Distance** data entry box specifies the incremental move distance or absolute target position. It accepts a float number, constant or variable.

Up to four optional events (event functions) may be associated with the move. A data entry box for each of four possible events allows you to specify an event by a valid integer constant, variable (Ix), global variable (Glx), or an equivalent label, identifying the event. Any specified event functions must be programmed before running the program. The AxisEvt icon may also be used to enable events on an axis. The limit of four enabled events includes both Move and AxisEvt enabled events. The event types available for the "Move" icon are "Single Axis Distance from Start" and "Single Axis Distance from End."

Axis Definition Units

Single Axis Linear	inches
Single Axis Linear	mm
Single Axis Rotary	degrees (displayed as grads)
Coordinated	inches
Coordinated	mm
Ratioed (master)	see single axis definitions
Ratioed (slave)	same as master

Msg

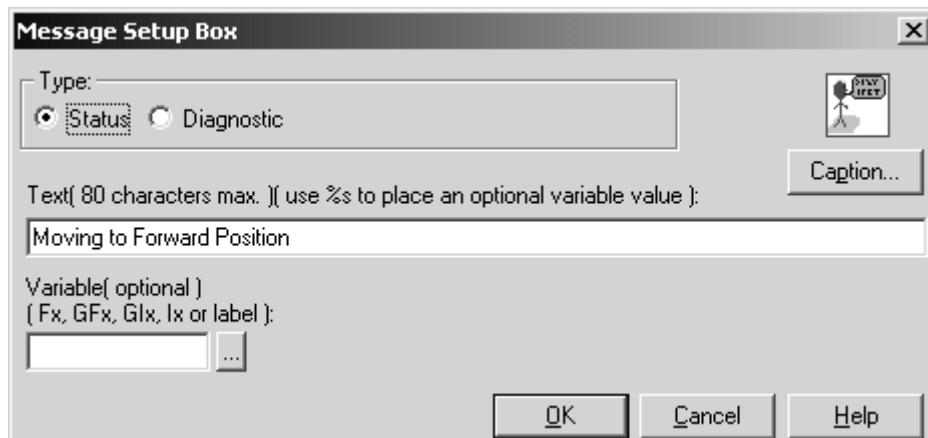


The Message icon is used to select a status or diagnostic message (up to eighty characters) at a specified point in the program flow. Placing a Msg icon on a task or subroutine workspace automatically displays a Message Setup window.

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 run time, the current status and diagnostic messages can be viewed by selecting Tasks from VisualMotion's Status 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.



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 displayed variable.

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 label window will be displayed when focus is on the variable field and cursor is doubled-clicked within the Message Setup Box.

It is also a good idea to copy the message into the comment box under the icon label. When the cursor is over the icon, the pop-up window will display the text message.

Param



The Param icon is used to transfer specified control or drive-related parameters or variables between the PC Host system and an array of control variables. 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 its 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:

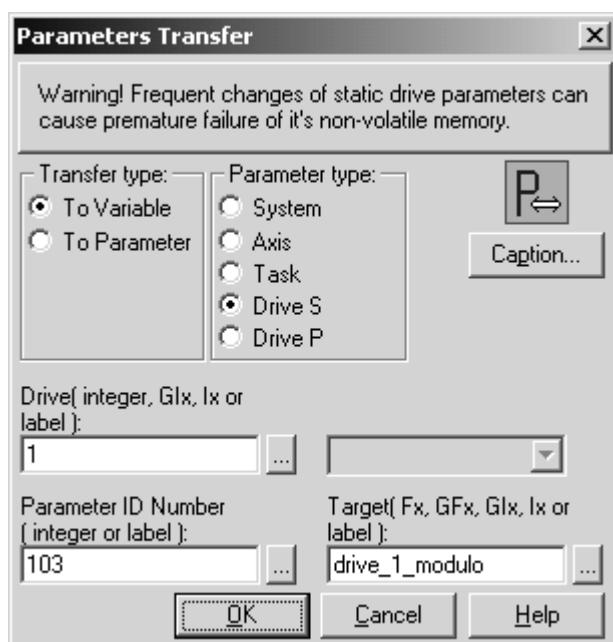
- ECODRIVE01
- DIAx04
- DIAx03

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
-

Subsequently downloading the modified control variables permits dynamic modification of parameters during system operation. Placing a Param icon on a task or subroutine workspace automatically displays a Parameter Transfer window.



Example 1: To read the drive 1 modulo value (S-x-0103) into the variable "drive_1_modulo"

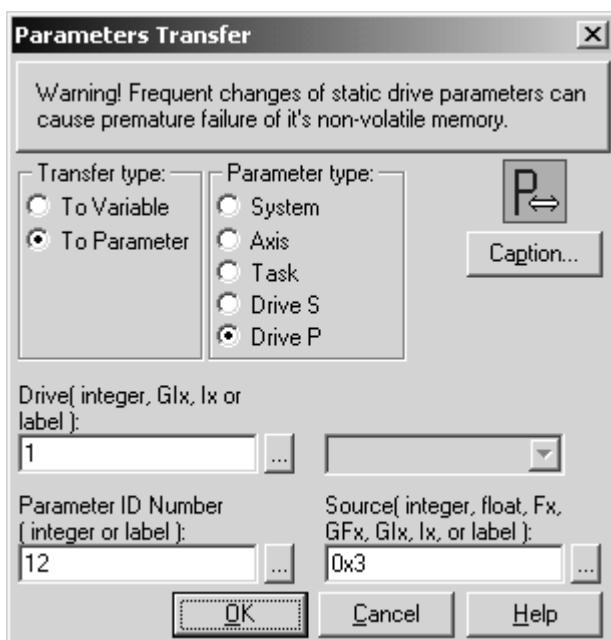
The radio buttons for Transfer type determine the direction of the transfer, from a specified control constant or variable to the parameter, or from the parameter to a control variable. The Parameter type radio buttons select from System, Axis, Task or Drive associated parameters.

Specify the parameter to be transferred in the Parameter ID Number data entry box as an integer or equivalent label.

If the **System** 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 data entry box 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).



Example 2: To write to the drive 1 "Set Absolute Measuring Procedure Command" (P-x-0012)

A pop-up User Defined Labels window may be opened by clicking once within the appropriate data entry box, then double-clicking immediately outside the entry box.

Note: Certain system 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.

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.

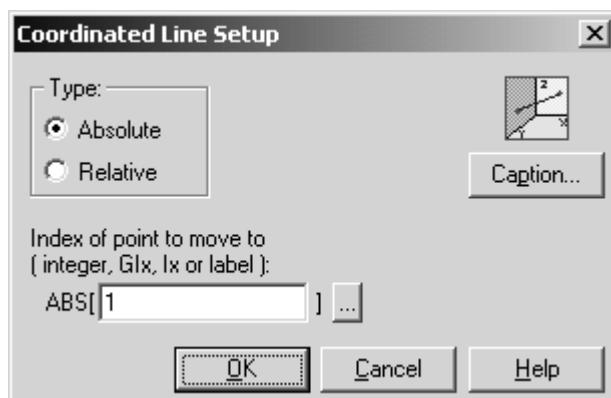
Motion may be Absolute or Relative and is defined by the two endpoints of the line of motion. Points are specified by an index into the point table using an integer constant, variable, global variable or an equivalent label. Refer to the **Calc icon**. The second point, or target point, must be a point 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.

A pop-up window listing the User Define Labels is available for the ABS and REL data entry boxes.

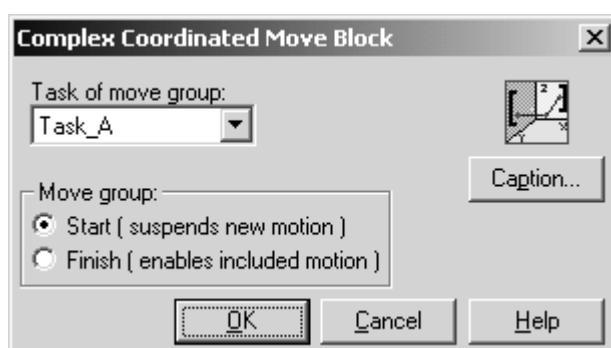
A Go icon is not required when using the Path icon. When executed, the motion will immediately be sent to the path planner. Stepping to the next icon will take place immediately.



PathCalc



The PathCalc icon is used to set up a complex coordinated move block in the Trans 01D product.



PID



The PID icon is used to initialize and install a proportional control loop on the control card, up to 32 loops are possible. The location of the PID 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.

Note: The available number of PID loop is dependent upon the VisualMotion software release.

VisualMotion release 08V08 and earlier:

Up to 10, PID loops can be configured.

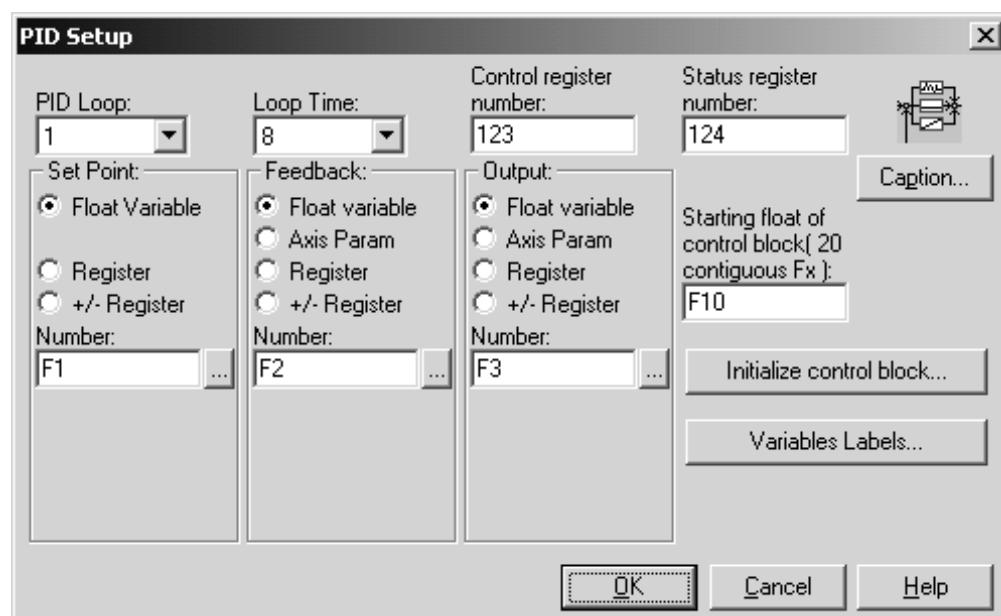
VisualMotion release 08V09 and later:

Up to 32, PID loops can be configured.

Set point can be a program or global variable (Fx, GFx, Glx, Ix, or label), or signed or unsigned register. Internally, the value is converted to a float. Its offset is then subtracted and the resultant multiplied by its scalar.

Feedback and **output** may be a program or global variable (Fx, GFx, Glx, Ix, 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. Its offset is then subtracted and the resultant multiplied by its scalar. A list of valid axis parameters can be viewed in drive parameter S-0-0188 "List of configurable Data in the MDT."

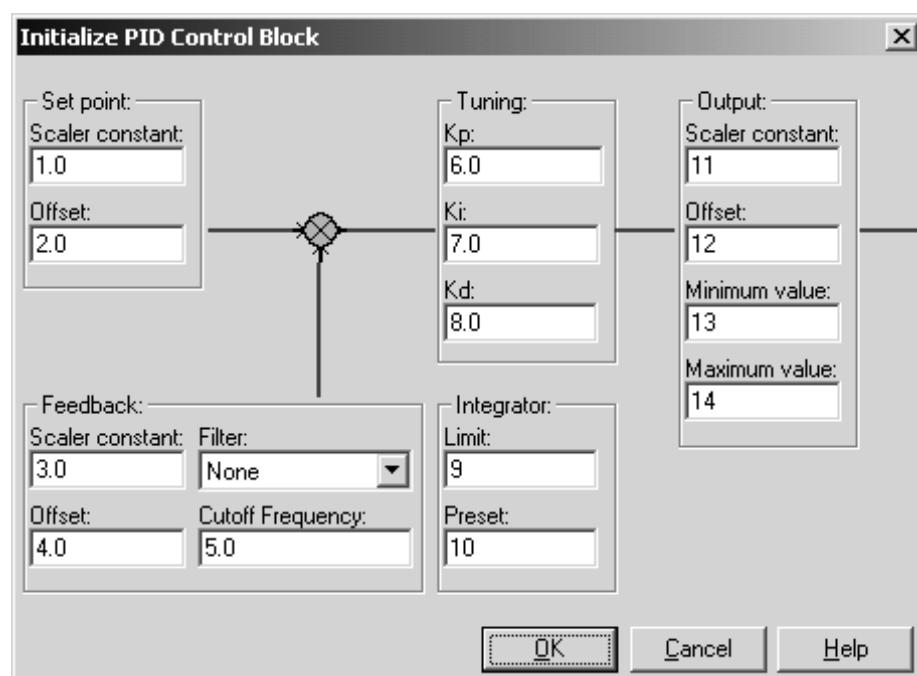


The Starting float of the control block is the first variable used to store the initialization values. The remaining variables are sequentially assigned to each control block value. (F11, F12, F13, F14.....)

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.

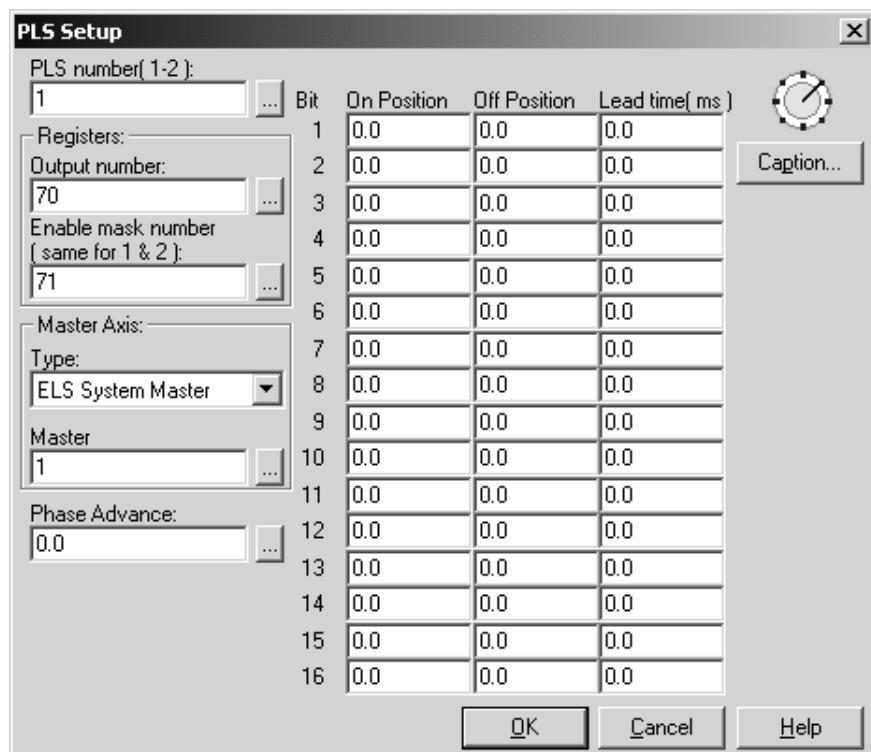
Control block values, default or user selected, are added at compile time. If program floats are used, they must have been allocated in the **Size** Icon.



PLS



The PLS icon initializes the control's programmable limit switch data structure when the program is compiled. The location of the PLS icon in the program is not important since it is not executed when the program is run. A maximum of 2 Control PLSSs can be compiled in a user program, whether active or not. 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 phase advance is added to all positions in the PLS table. The position input for the PLS can be assigned to an ELS, Virtual, Real or Drive Based master. The outputs are updated every SERCOS cycle. It is possible to write to the positions in the PLS table in the user program. Refer to the **Calc** icon for details.

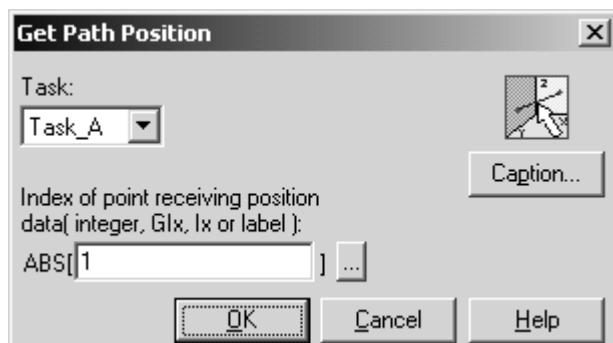
Position



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.

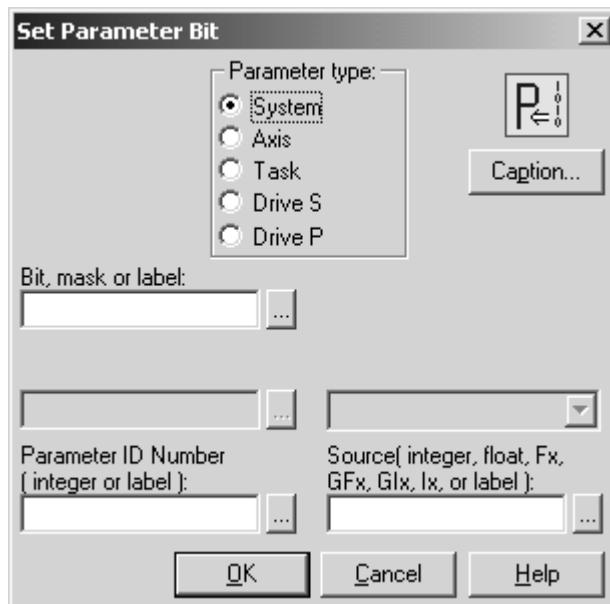
Placing a Position icon on a task or subroutine workspace automatically displays a Get Path Position window. Four control tasks (A-D) can be selected from the Task menu. The absolute point table destination for the position data may be specified by an integer constant, variable, global variable or an equivalent label. A pop-up User Defined Labels window is available.



PrmBit



The PrmBit icon is used to change the state of a specific bit(s) of a binary parameter. The state of the bit is changed by creating a mask and writing to it a value of either 0 or 1. A bit mask can be created for single or multiple bits.



Note: Any binary parameter that can be written to can be used in the PrmBit icon. An error is issued if a read-only binary parameter is used.

The PrmBit icon is initialized during the SERCOS phase 2 to 3 transition, regardless of where the icon appears in the user program flow.

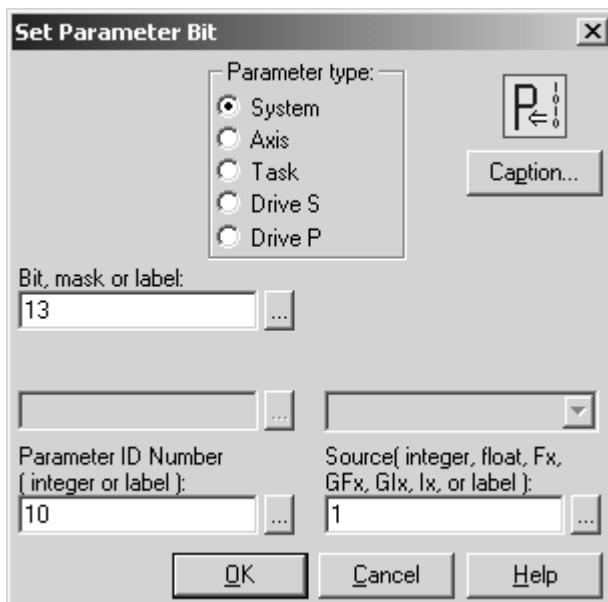
Different entry fields in the *Set Parameter Bit* window are displayed based on the Parameter type radio button selected.

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, Ix or label	Axis Parameter Type
Drive	Enter a drive number as an integer, Glx, Ix 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, Ix or label	All Parameter Types

Table 8-12: PrmBit 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 PrmBit icon used to change the state of bit 13 for system parameter C-0-0010 from its current state to 1.



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.

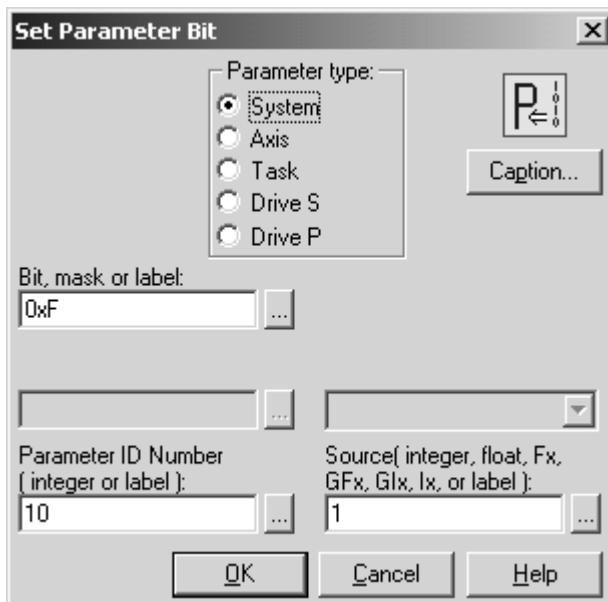
00000000000001111 (binary) = 0xF (hexadecimal)

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: 00000000000010101. 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 PrmBit icon used to change the states of bits 1, 2, 3 and 4 for system parameter C-0-0010 from its current state to 1.



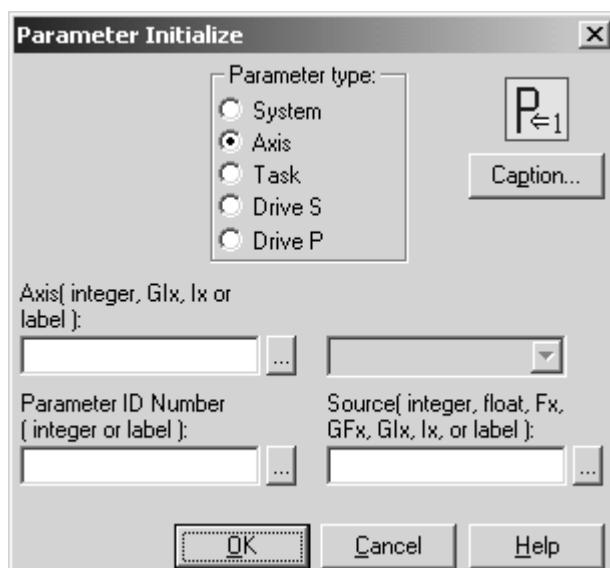
PrmInt



The PrmInt icon is used to initialize a system, axis, task, or drive parameter with a specific value.

Note: Any parameter that can be written to can be used in the PrmInt icon. An error is issued if a read-only binary parameter is used.

The PrmInt icon is initialized during the SERCOS phase 2 to 3 transition, regardless of where the icon appears in the user program flow.



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, Ix or label	Axis Parameter Type
Drive	Enter a drive number as an integer, Glx, Ix 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, Ix or label	All Parameter Types

Table 8-13: PrmInt Icon Entry Fields

A parameter is selected by clicking on the desired parameter type radio button and entering its number in the "Parameter ID Number" field. Enter the appropriate axis, task or drive number.

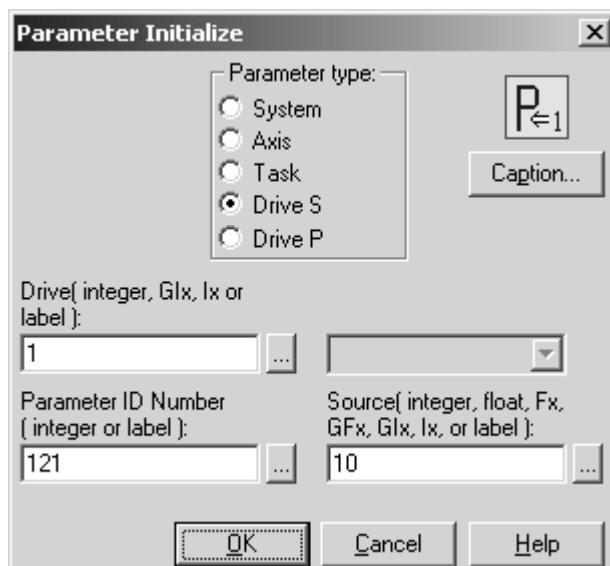
Selecting "Axis" or "Drive" enables a data entry field, permitting entry of an integer constant, variable (Ix), global variable (Glx) or equivalent label specifying the axis or drive.

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 of the same data type. For Drive, an integer Glx, Ix or label must be used and not a different data type (such as Fx or a label created for a float).



Example: The gear ratio input revolutions of drive 1 is initialized to 10.

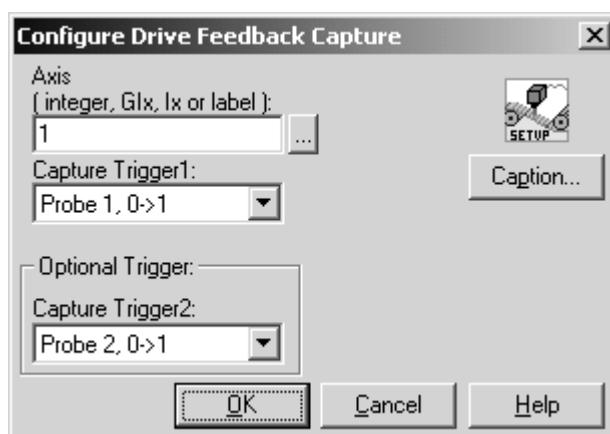
Note: When using variables that are initialized with a zero as the parameter source, a SERCOS error may occur if no value has been entered for the variable as the system switches from phase 2 to phase 3.

Probe



The Probe icon is used to enable drive based position capture inputs (Probe 1 and Probe2) for a drive (non-RAC) assigned to the specified axis. On the programmed transition, a flag is set in the cyclic SERCOS data informing the control of its happening. The ProbeEvt icon (Refer to ProbeEvt) assigns an event to execute when this flag is set.

This icon is necessary to put the captured probe position in the SERCOS telegram and it executes during the transition from phase 2 to phase 3. This icon should only be used once for each drive.



Axis - indicates the drive whose position will be captured on its Probe input transition. The entry may be an integer, global integer variable (Glx), program integer variable (Ix), or an equivalent label.

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, Probe 1 or Probe 2. Capture can be on a 0->1 or 1->0 transition.

One or two Capture Triggers may be enabled. Each may be assigned to none, or one of the two drive probe signals, with selection of the trigger on a rising or falling edge:

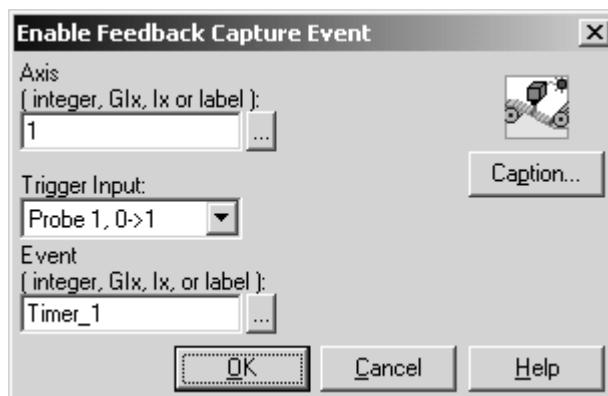
Data stored in this Parameter

Probe 1 low (0) to high (1)	S-x-0130
Probe 1 high (1) to low (0)	S-x-0131
Probe 2 low (0) to high (1)	S-x-0132
Probe 2 high (1) to low (0)	S-x-0133

ProbeEvt



The ProbeEvt icon is used to enable a control event for triggering by a digital drive probe transition. The event function must have been entered by choosing the Add Event Function from Visual Motion's Edit menu.



The Axis triggering the event is specified by an integer constant, variable (Ix), global variable (Glx), or an equivalent label. The corresponding probe trigger must be assigned to the same axis using an Axis or Probe icon.

The drive probe event used as the Event Trigger Input is selected from the pop-down menu. Choices include none (trigger disabled), Probe 1 low-to-high, Probe 1 high-to-low, Probe 2 low-to-high, and Probe 2 high-to-low.

The Event to be triggered on position capture occurrence is specified by an integer constant, variable (Ix), global variable (Glx), or an equivalent label that provides an index into the event table. The Event must have been allocated in the Size icon and initialized prior to occurrence.

The axis position at the time of the probe trigger is immediately available by reading the event's "a" field from the event table, (i.e., EVT[x].a), or by using the Param icon to read the corresponding probe's position capture from the drive as a drive parameter (Dx.00130 - Dx.00133). Because this instruction does not enable a repeating event, it must be executed for each new edge to be scanned.

S-0-0130	Probe 1 0->1
S-0-0131	Probe 1 1->0
S-0-0132	Probe 2 0->1
S-0-0133	Probe 2 1->0

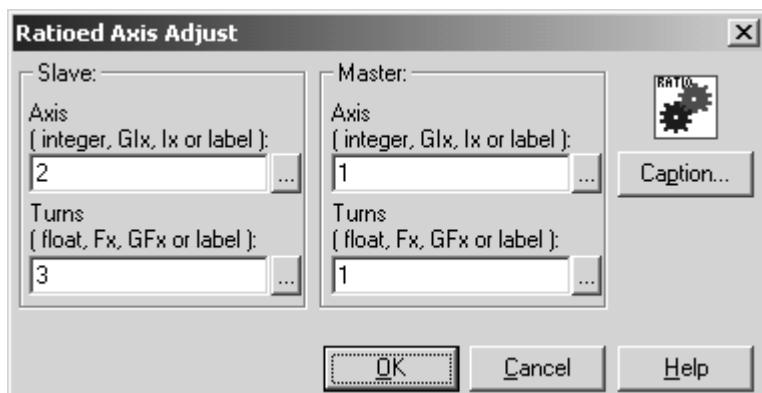
Ratio



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 Axis icon as ratioed axes. Refer to the **Axis Icon** for details.



The **Master** and **Slave** axes are selected by entering an integer constant, variable, global variable or an equivalent label in the Axis data field. The ELS Virtual Master 1 (axis 0) may be used as the Master Axis.

A pop-up User Defined Labels window is available for all four data entry boxes. 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** data entry boxes 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.

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.

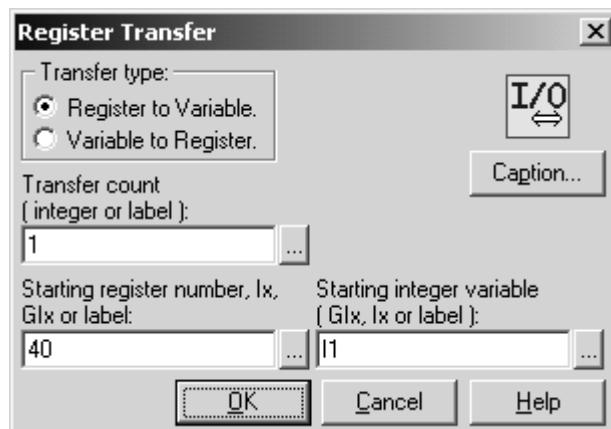
A ratio mode slave can now follow the ELS master (virtual or real). On a system without ELS drives, all slaves can follow the ELS master so that there is no lag between slaves. To follow the ELS master, enter a 0 for the master axis in the Axis icon or Ratio icon.

Note: When assigning the Master axis as an ELS Virtual Master 1 (axis 0), it must also be setup using both VM (Virtual Master) and ELSMstr icons. Only Virtual Master 1 can be used as a master in a Ratioed Axes Setup.

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.



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.

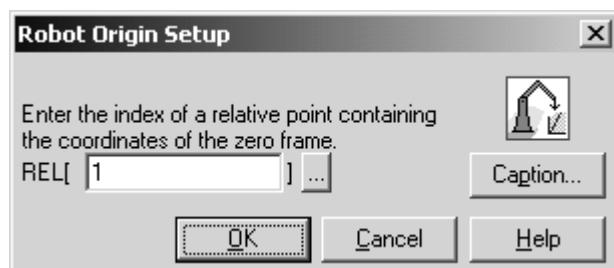
The **Transfer count**, Starting register number and starting integer variable must be an integer constant, variable (Ix), global variable (Glx), or an equivalent label.

RobotOrg



The RobotOrg icon 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 $\text{REL}[3] = \{1, 2, 3, 0, 0, 0, \dots\}$ and this point is specified as the robot origin, 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 icon instruction is executed, all jogging, teaching and path locations are affected by the new origin.

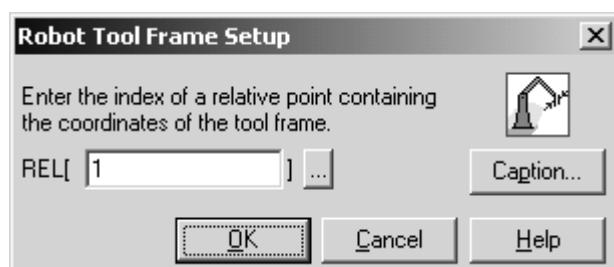


RobotTool



The RobotTool icon 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 $\text{REL}[4] = \{0, 0, 10, 0, 0, 0, \dots\}$ 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 icon instruction is executed, all jogging, teaching and path locations are affected by the end-of-arm tool location.



Scissors



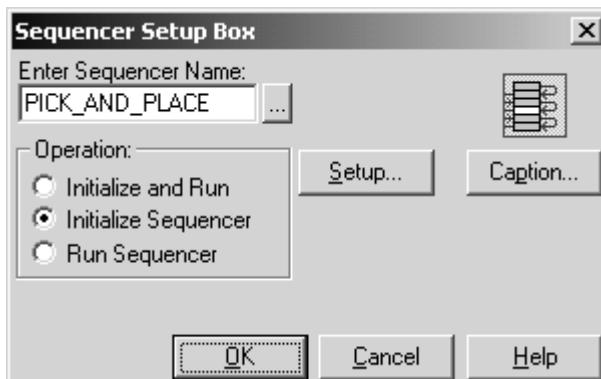
The Scissors icon is used to delete a line between two icons. Select the Scissors on the palette, position the tip of the Scissors cursor over the line to delete, press the left mouse button to delete.

Sequencer



The Sequencer icon is used to initialize and/or run a list. A Sequencer list contains user-defined steps that can be edited "on-line" without having to edit, compile and download a program. Each step contains a list of functions that will be executed sequentially following the order of the list. Each step also can have up to five-function argument, which will be passed to the subroutine when the function is executed.

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 sequences defined in the Size icon. If the Sequencer is only running, its number can be specified by a program or global integer (Ix, Glx).

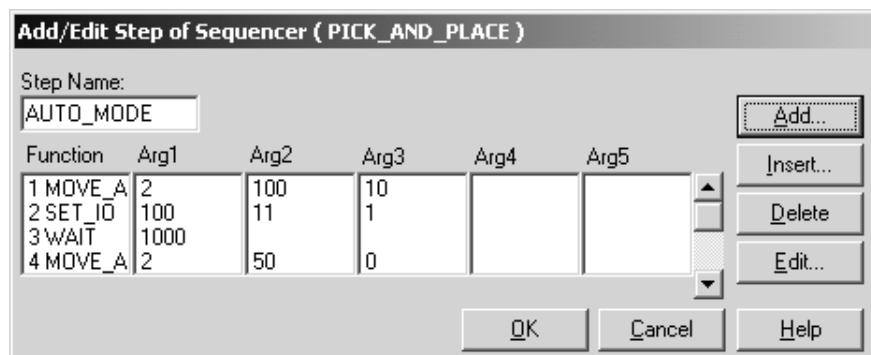
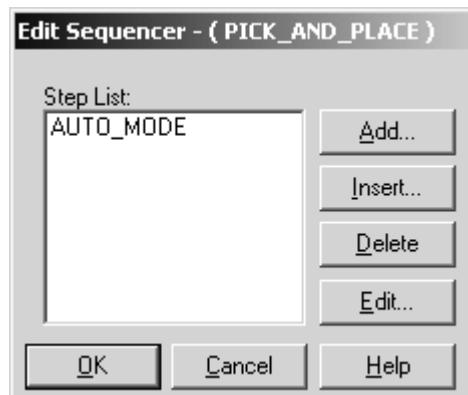


If Run Sequencer is selected the Setup... option is inactive and the icon can run a Sequencer that has previously been setup using the Sequencer Editor. Select "Run Sequencer" to just run the Sequencer without initializing it back to a default.

If *Initialize Sequencer* or *Initialize and Run* is selected, the Sequencer can be Setup... by the Sequencer icon according to the following icon windows. This is a good tool to use for restoring your program to a working default. When this icon is executed, the setup information will overwrite any previous "on-line" changes that may have been made using the Sequencer Editor. The system will stay on the sequencer icon until the sequence is complete.

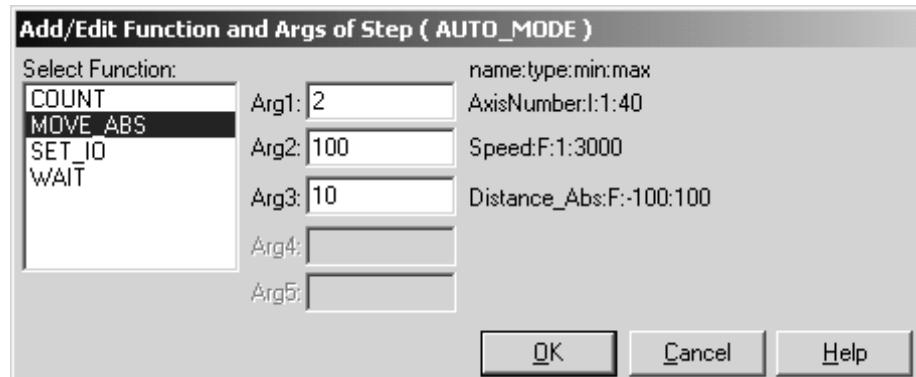
When Setup... is selected the following windows can be used to add, edit, delete or insert step lists within a Sequencer.

After selecting Add, a Step Name can be entered and edited as illustrated below. Each step has room for a twenty-character name to uniquely identify it.



Functions need to have the "Callable from Sequence" access flag set in the **Start** icon of the subroutine before they can be added to a Step List. Function arguments must also be defined in the Start icon for them to be used.

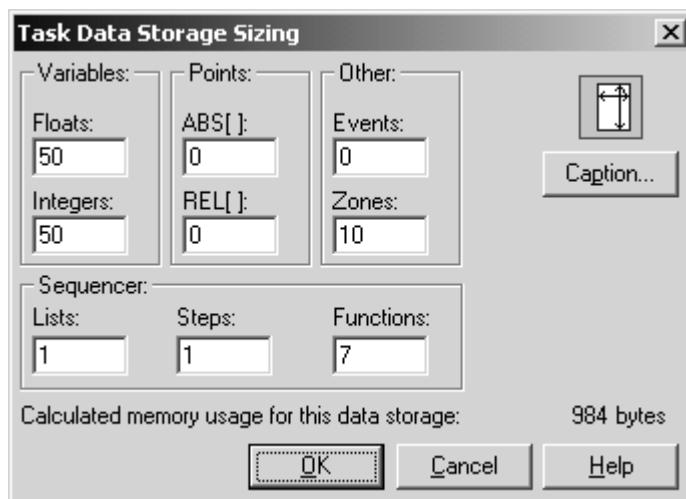
After selecting Add, a function can be entered and edited as illustrated below. The function argument values must be within the limits predefined in the START icon of the subroutine.



Size



The Size icon allocates control memory for the different types of data storage each task requires. Placing a Size icon in the workspace opens a Task Data Storage Sizing window. The entered values are the maximum number of variables, points, events, zones and Sequencer components that can be used in all the tasks.



The Sequencer components consist of lists, steps and functions. The values entered for lists and steps represent the total number of *different* lists or steps that are allowed in a program. If the same step is used more than once, in one or more lists, it still only counts as one towards the total. The total number of functions represents every time a function is executed in a step, even if the same function is used more than once. Each function will add to the total every time it is used.

The Size icon can only be used in the four main tasks; it cannot be used within a subroutine or event. In most cases, only one Size icon needs to be used per program. A size icon should be used in each Task that contains coordinated motion. When used, the Size icon should be placed immediately after the Start icon.

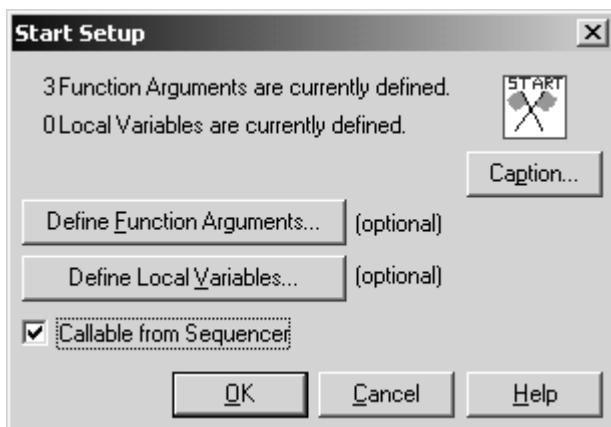
A program has a default value of fifty entries for integer and float variables. If any task uses more than 50 entries of any type, the Size icon must be used to allocate additional space. If a program uses less than fifty entries of any type, using the Size icon to reduce the allocation from the default to the actual number of data entries saves control memory space for other programs.

Variables each require 4 bytes, points 42 bytes, and events 118 bytes. The byte total of allocated memory is shown at the bottom of Tasks Data Storage Sizing window. After a new value is entered and the cursor is moved outside the window, a new byte total is calculated and displayed.

Start



The four control tasks (A through D), subroutines and event functions must all contain a Start icon. The Start icon indicates the beginning of program flow to the control compiler and declares function arguments and local variables. The Start icon is equivalent to the control Text Language TASK/START command statement.



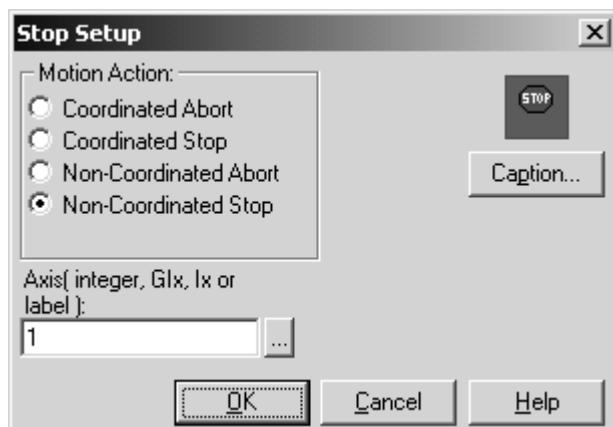
Stack based variables (function arguments and local variables) exist only while in the function (task, subroutines, or event) where they are declared. *Function arguments are those called from a subroutine icon or sequencer, and local variables are used within the subroutine for local data only they don't exist outside the subroutine.* This type of variable is useful for temporary results within a function or passing values to a function.

The same label may be used for a function argument or local variable in different functions. The label may not be a keyword or other label (user, register, or bit). The total number of function arguments and local variables in a function is limited to 16. Up to 5 function arguments may be passed to a subroutine.

Stop



The Stop 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 Stop icon on a task or subroutine workspace automatically displays a Stop Setup window.



A pop-up "User Defined Labels" window is available by highlighting variables (F1) and double clicking inside the "Stop Setup" window.

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 Go 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. 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

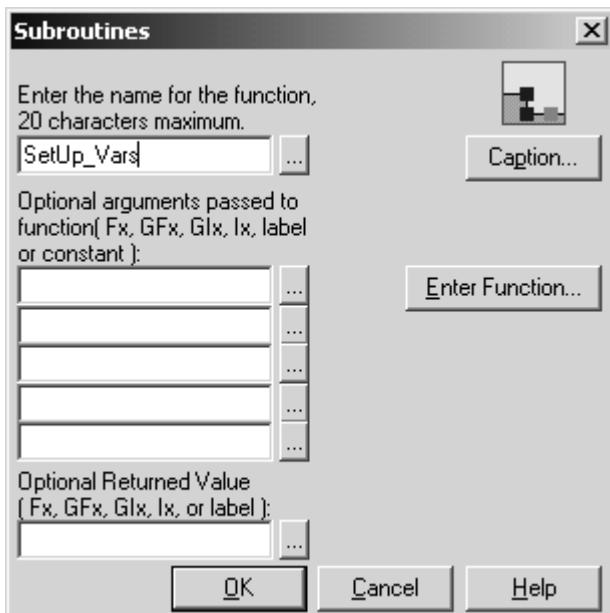
Note: After an **Abort**, all queued events and the "look-ahead" motion calculated by the path planner is lost and the current move is aborted. The target position is set equal to the current feedback position. After a **Stop**, both queued events and the calculated path are retained. The target position is not set equal to the feedback position. When the axis is again enabled, it will complete the last commanded move.

Motion Action Non-Coordinated and enter a "0" for the axis number.

Sub



The Sub icon is used to enter a subroutine function from the normal program flow. Placing a Sub icon on a task or subroutine workspace automatically displays a Subroutine window.



After you enter a name for your subroutine, clicking on the Enter Function button automatically replaces your current task or subroutine workspace with a new subroutine workspace in the main Visual Motion window. The title bar of the main window changes to show the name of the new subroutine. You can now use Visual Motion icons to build a subroutine in exactly the same manner as building a main Task. To return from a subroutine workspace to one of the four tasks, select the task from the View menu.

Function arguments are stack based variables passed to a subroutine. The setup window for the Sub icon has entries for the function arguments. Up to 5 arguments may be passed to a subroutine. The arguments can be constants or variables (float, integer, ABS index or REL index). The function arguments need to be declared in the subroutine Start icon window. A minimum and maximum value must also be entered for each of these arguments. This limits the range of argument values used during Sequencer editing from the teach pendant or the Sequencer interface.

Function arguments are not allowed in tasks or events; however, an optional return argument may be passed back to a task from a subroutine. A return argument is a good way of getting a return value

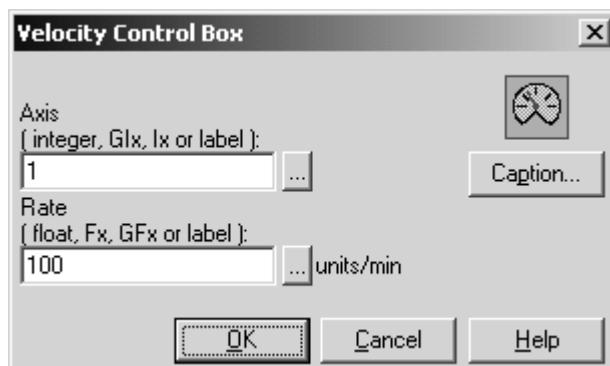
from a subroutine. A common subroutine can then be utilized from more than one task. Multiple variables have similar states depending on conditions. You can call a subroutine with a variable being the return value to set its state.

Local variables are stack based variables used in a function. The total number of local variables and function arguments is limited to 16. Local variables, floats and integers are initialized to zero on each entry into the function. This type of variable can be used to store temporary results in a re-entrant function called on by more than one task. Local variables can also be used to avoid possible conflicts with program or global variables between tasks.

Velocity



The Velocity icon is used to specify a rate for motion on a single non-coordinated axis or the ELS Virtual Master (Axis 0). 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. A pop-up window of User Defined Labels is available for both data entry boxes.



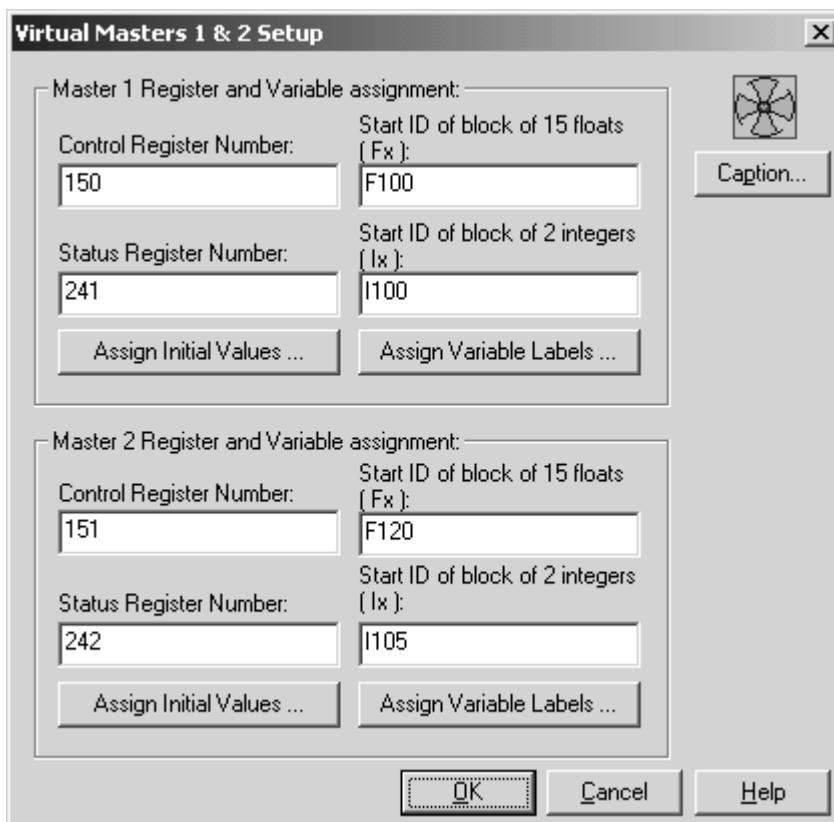
Units:

Single Axis Linear inches:	inches/min
Single Axis Linear mm:	mm/min
Single Axis Rotary:	RPM
Velocity Mode:	RPM
Virtual Master:	RPM

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 of the 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 (see below). They can be changed if desired.



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 programs 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 8-14: 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 8-15: 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.

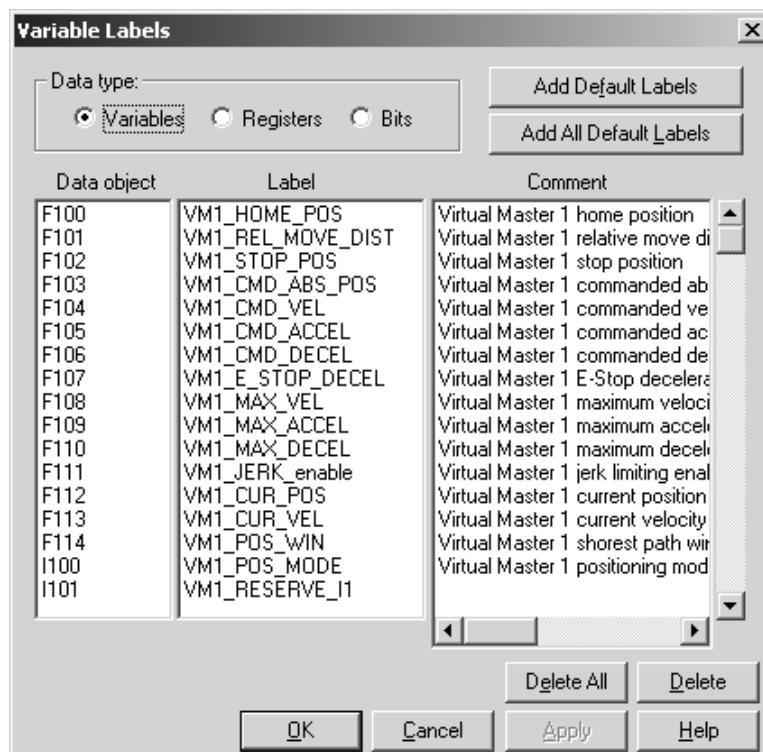
Virtual Master 1 & 2 Program Variable Default label	Virtual Master 1 & 2		Virtual Master 1 & 2 Program Variable Default Comment (80 character limit)	Default Initial Value	Units
VM#_HOME_POS	F100	F120	Virtual Master # home position	0	Degrees
VM#_REL_MOVE_DIST	F101	F121	Virtual Master # relative move distance	1	Degrees
VM#_STOP_POS	F102	F122	Virtual Master # stop position	0	Degrees
VM#_CMD_ABS_POS	F103	F123	Virtual Master # commanded absolute position	0	Degrees
VM#_CMD_VEL	F104	F124	Virtual Master # commanded velocity	20	RPM
VM#_CMD_ACCEL	F105	F125	Virtual Master # commanded acceleration	100	Rad/sec ²
VM#_CMD_DECEL	F106	F126	Virtual Master # commanded deceleration	100	Rad/sec ²
VM#_E_STOP_DECEL	F107	F127	Virtual Master # E-Stop deceleration	500	Rad/sec ²
VM#_MAX_VEL	F108	F128	Virtual Master # maximum velocity	100	RPM
VM#_MAX_ACCEL	F109	F129	Virtual Master # maximum acceleration	500	Rad/sec ²
VM#_MAX_DECEL	F110	F130	Virtual Master # maximum deceleration	500	Rad/sec ²
VM#_JERK_ENABLE	F111	F131	Virtual Master # jerk limiting enable	1	Degrees
VM#_CUR_POS	F112	F132	Virtual Master # current position		Degrees
VM#_CUR_VEL	F113	F133	Virtual Master # current velocity		RPM
VM#_POS_WIN	F114	F134	Virtual Master # shortest path window	1	Degrees
VM#_POS_MODE	I100	I105	Virtual Master # positioning mode	0	Note 1.)
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 8-16: Virtual Master Program Variables

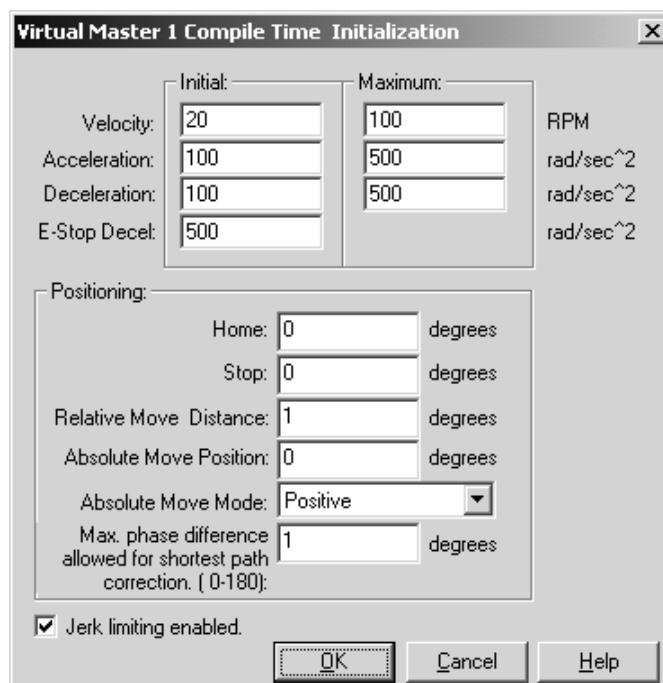
Assign Variable Labels

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.



Assign Initial Values

Refer to [Virtual Master Compile Time Initialization](#) in chapter 5 of the VisualMotion Application Manual for details.



Wait

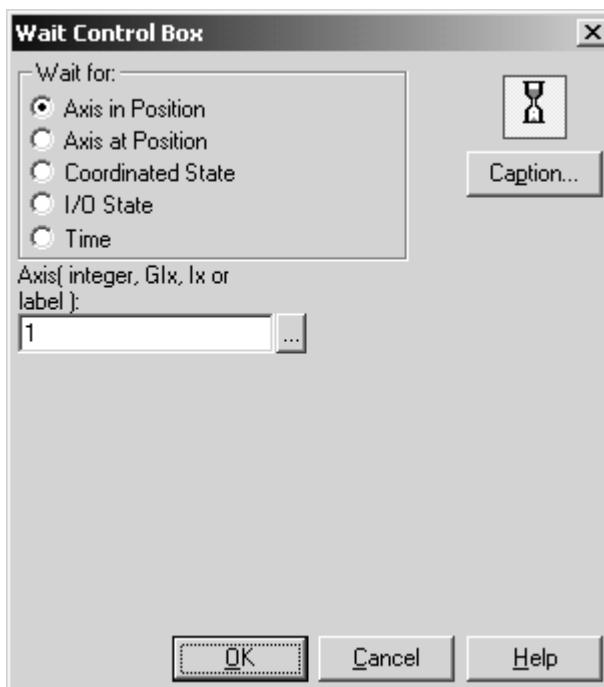


The Wait icon is used to hold the execution of the program flow at the Wait icon until a specified condition has been satisfied. The condition may be related to a single axis' position, time, I/O state, or path planner state (for coordinated motion). Placing a Wait icon on a task or subroutine workspace 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.



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



The *At point state* pop-down menu 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.

Slew (constant speed) - 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.

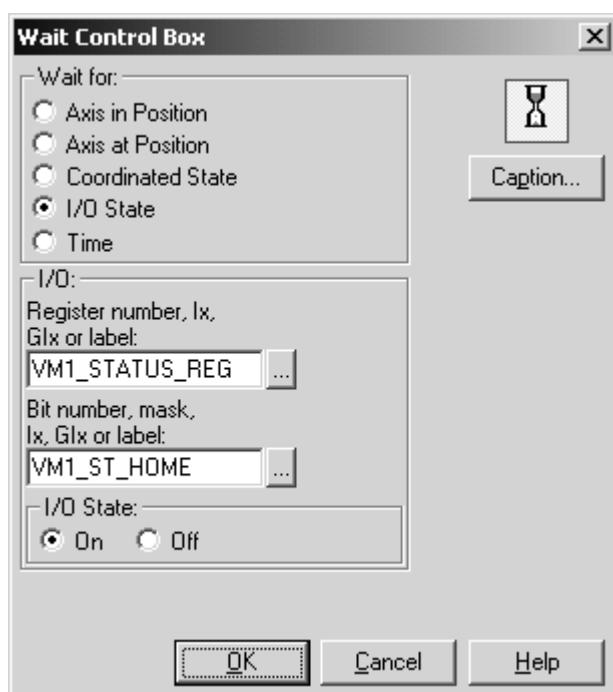
At Target - Task's coordinated motion is at the specified point.

Done - Task's coordinated motion has completed for the specified point.

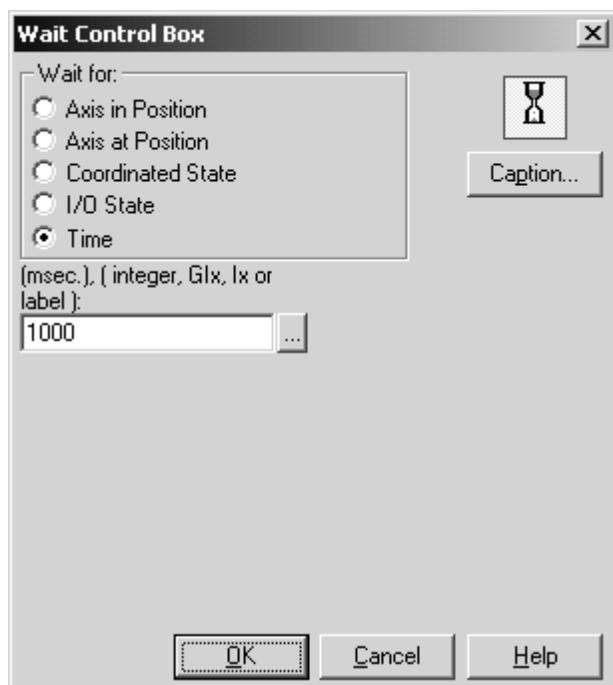
I/O State pauses program execution until the specified I/O condition for the specified I/O register is satisfied. The I/O register is specified by entering a valid I/O register ID number or equivalent label. A pop-up Register Labels window is available

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). A pop-up Bit Labels window is available.

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.



Time pauses program execution for a specified time delay. Enter the number of milliseconds in the Time Delay data entry box. The delay may be specified by an integer constant, variable (Ix), global variable (Glx), or an equivalent label. A pop up window of User Defined Labels is available for both data entry boxes.



9 Kinematics

9.1 Description

VisualMotion provides predefined kinematic libraries for controlling industrial robots in coordinated motion applications to achieve accurate high-speed positioning over geometric paths. Currently, GPP supports 9 kinematic libraries for use in VisualMotion programs.

The following kinematic libraries are available for assigning to axes:

- Kinematic #1 (XYZ Gantry Robot)
- Kinematic #2 (Parallelogram Robot)
- Kinematic #3 (XY Tandem Motor Robot)
- Kinematic #4 (Parallelogram Robot)
- Kinematic #5
- Kinematic #8 (XYZ Gantry Robot with Velocity Precision)
- Kinematic #9 (2 Axes Articulated Arm Robot)
- Kinematic #10 (XY Wrist Robot)
- Kinematic #12 (X and Z Loader Robot)

The assignment of a kinematic library number is done in the *Task Axes Setup* icon within a VisualMotion program, as shown in Fig. 9-1.

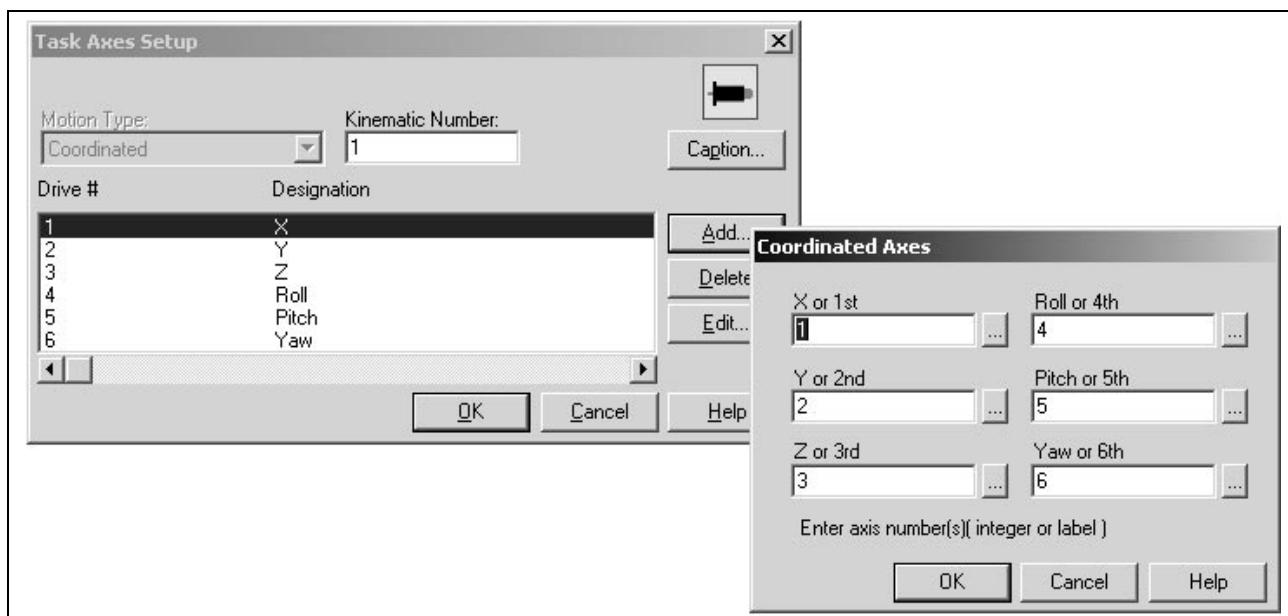


Fig. 9-1: Assigning a Kinematic Library

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 library (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 ***On-Line Data*** \Rightarrow **Parameters** from VisualMotion's main menu and selecting the Task tab.

Note: In addition to the mentioned task parameters, unit parameters such as A-0-0005 and T-0-0005 must be setup manually and match the units used in the active kinematic library.

As part of the user program setup, make sure to configure the mechanical setting for each axis under **Parameters** \Rightarrow **Mechanical** from within the Drive Parameter Editor Window (**Diagnostics** \Rightarrow **Drives**).

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. Refer to Fig. 9-2 for details. 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 (2π)	S-0-0123

Table 9-1: Mechanical Settings

To make these settings, select **Diagnostics** \Rightarrow **Drives** from VisualMotion Toolkit's main menu. Next, select **Parameters** \Rightarrow **Mechanical** to display the window in Fig. 9-2.

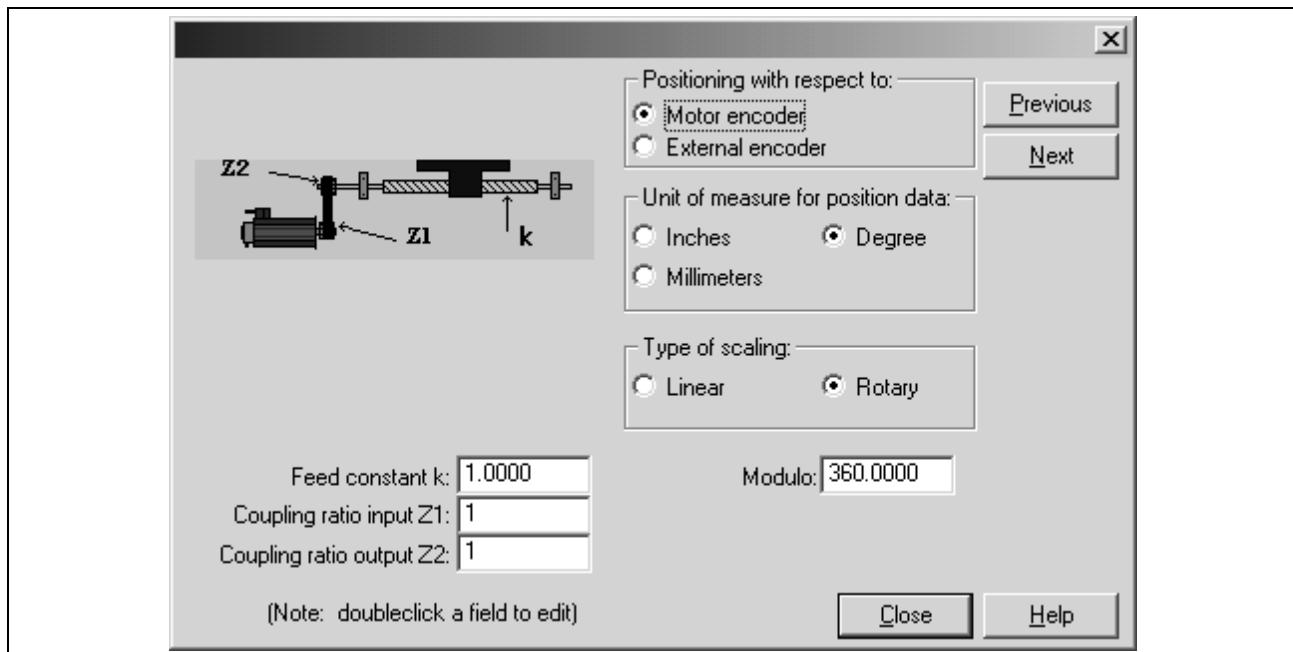


Fig. 9-2: Drive n Mechanical

Note: Gear ratios should be set according to the physical gearing of each axis.

9.2 Kinematic Libraries

Kinematic #1

This kinematic library 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.

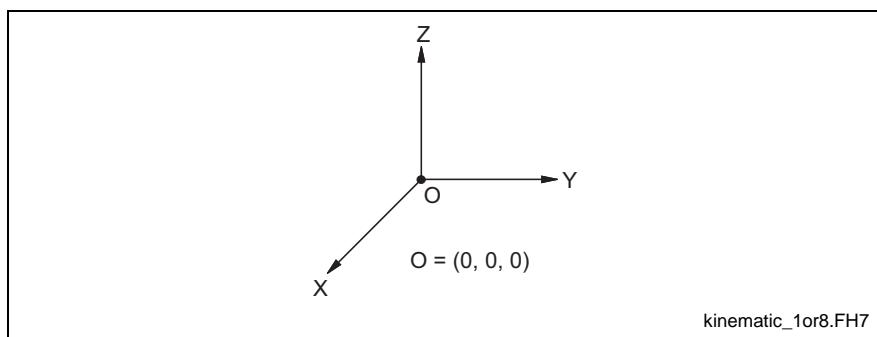


Fig. 9-3: Kinematic Library 1 Graphic

Kinematic #2

This kinematic library 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. 9-4: Kinematic Library 2 Task Parameters Values

Note: Kinematic #2 requires special mechanical setup. Refer to Special Case Kinematics on page 9-2 for details.

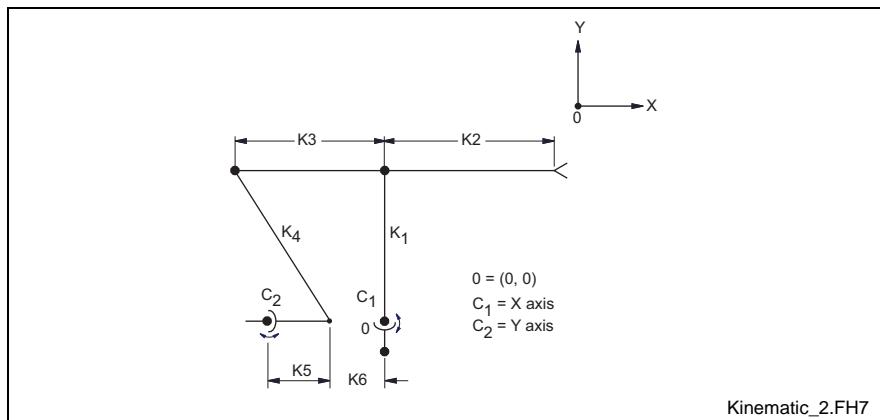


Fig. 9-5: Kinematic Library 2 Graphic

Kinematic #3

This kinematic library 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.

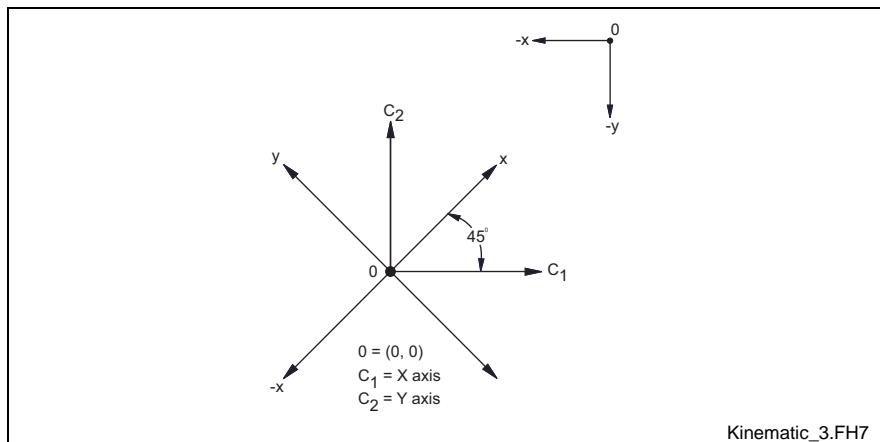


Fig. 9-6: Kinematic Library 3 Graphic

Kinematic #4

This kinematic library represents a 2 axes (C1 and C2) parallelogram robot.

Task Parameters	Kinematic 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. 9-7: Kinematic Library 4 Task Parameters Values

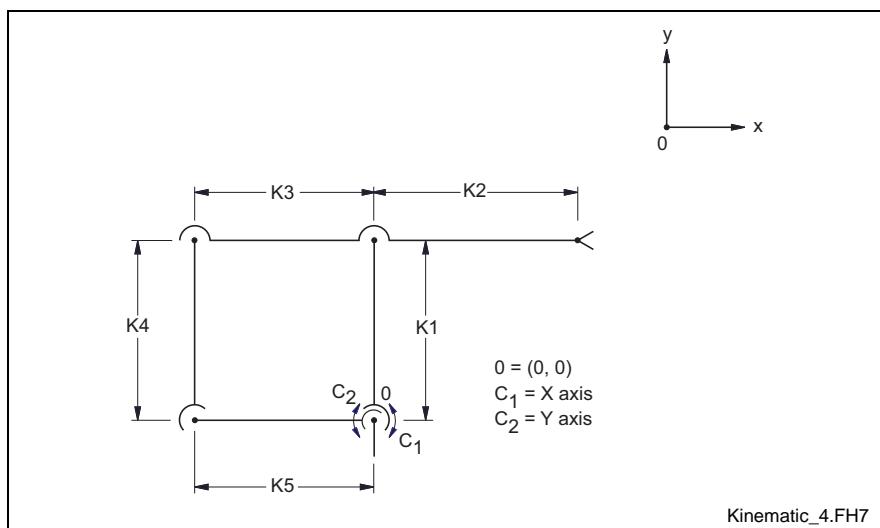


Fig. 9-8: Kinematic Library 4 Graphic

Note: Kinematic #4 requires special mechanical setup. Refer to Special Case Kinematics on page 9-2 for details.

Kinematic #5

This kinematic library represents a 2 axes (C1 and C2) XY tandem motor robot.

Task Parameters	Kinematic 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. 9-9: Kinematic Library 5 Task Parameters Values

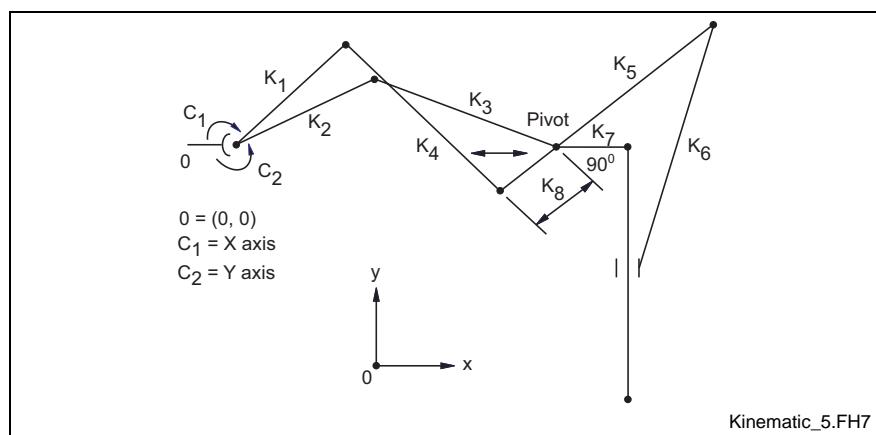


Fig. 9-10: Kinematic Library 5 Graphic

Note: Kinematic #5 requires special mechanical setup. Refer to Special Case Kinematics on page 9-2 for details.

Kinematic #8 with Velocity Precision

This kinematic library 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.

Active Program, Points - Card 0										
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	0.0	0.0	0	0	0	0	0.0	
004	0.0	0.0	0.0	0.0	0	0	0	0	0.0	
005	0.0	0.0	0.0	0.0	0	0	0	0	0.0	
006	0.0	0.0	0.0	0.0	0	0	0	0	0.0	
007	0.0	0.0	0.0	0.0	0	0	0	0	0.0	
008	0.0	0.0	0.0	0.0	0	0	0	0	0.0	
009	0.0	0.0	0.0	0.0	0	0	0	0	0.0	

Fig. 9-13: Example Points Table

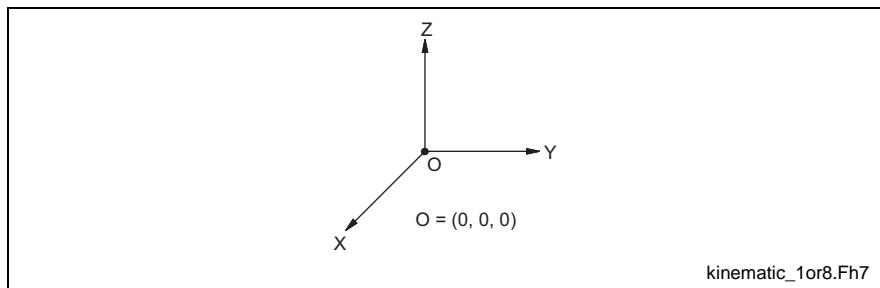


Fig. 9-14: Kinematic Library 8 Graphic

Kinematic #9

This kinematic library represents a 2 axes (C1 and C2) articulated arm robot.

Task Parameters	Kinematic Constants	Default Values	Units
T-0-0050	K1	35.669	Inches
T-0-0051	K2	35.669	Inches

Fig. 9-15: Kinematic Library 9 Task Parameters Values

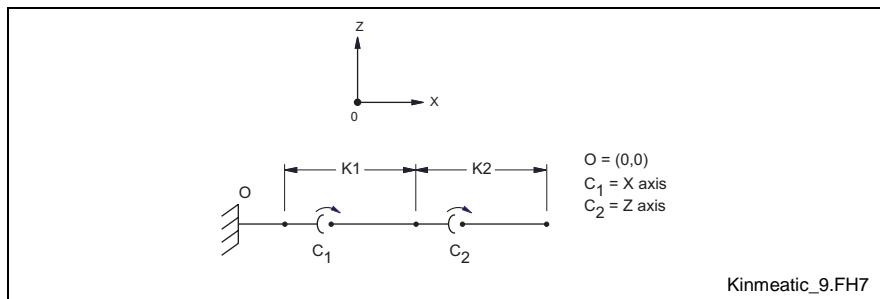


Fig. 9-16: Kinematic Library 9 Graphic

Note: Kinematic #9 requires special mechanical setup. Refer to Special Case Kinematics on page 9-2 for details.

Kinematic #10

This kinematic library 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.

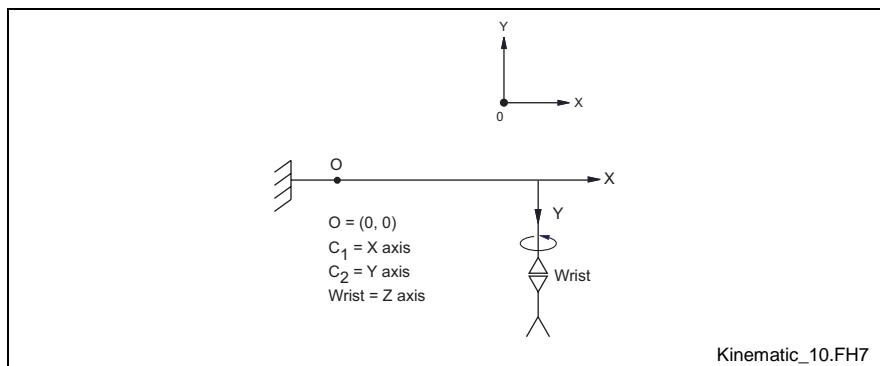


Fig. 9-17: Kinematic Library 10 Graphic

-
- 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 library represents a 2 axes (X and Z) Loader robot.

Task Parameters	Kinematic 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. 9-18: Kinematic Library 12 Task Parameters Values

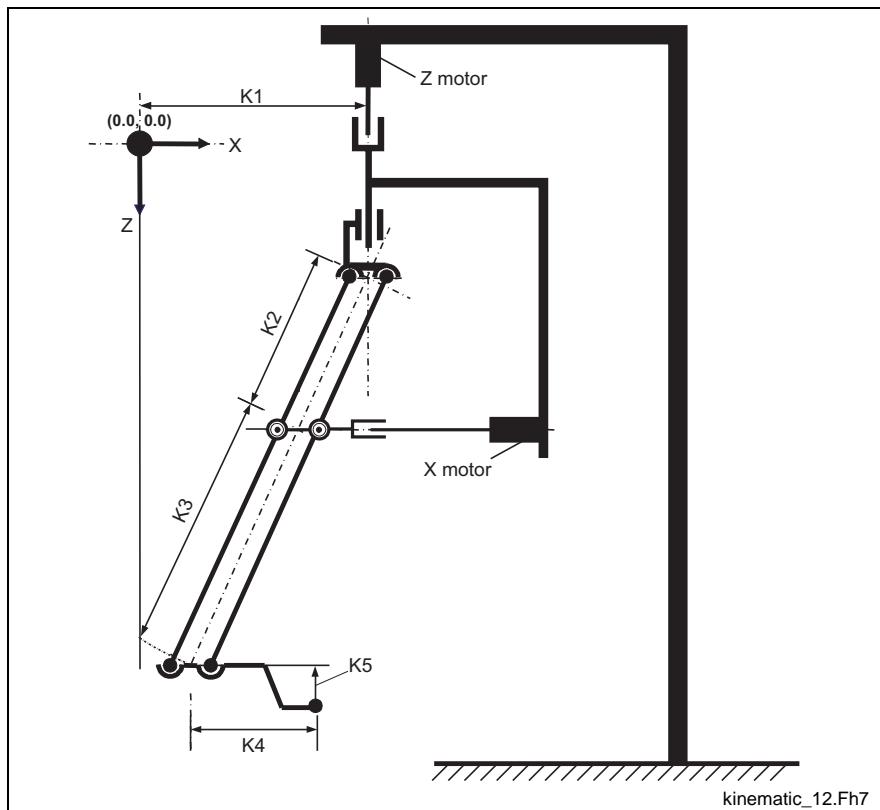


Fig. 9-19: Kinematic Library 12 Graphic

10 VisualMotion Human Machine Interface

10.1 Overview

Rexroth Indramat'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.

-
- Note:** VisualMotion 8 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
-

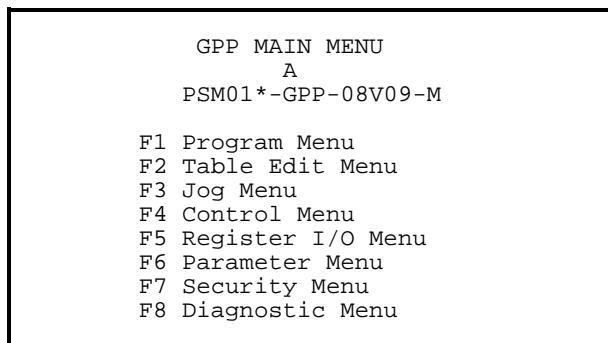
This chapter will focus on the BTC06 using terminal emulation software. Refer to the *VisualMotion 8 Project Planning Manual* for details regarding hardware setup.

10.2 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 Indramat'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:



-
- Note:** The first screen that is displayed on the Teach Pendant is the Control 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.

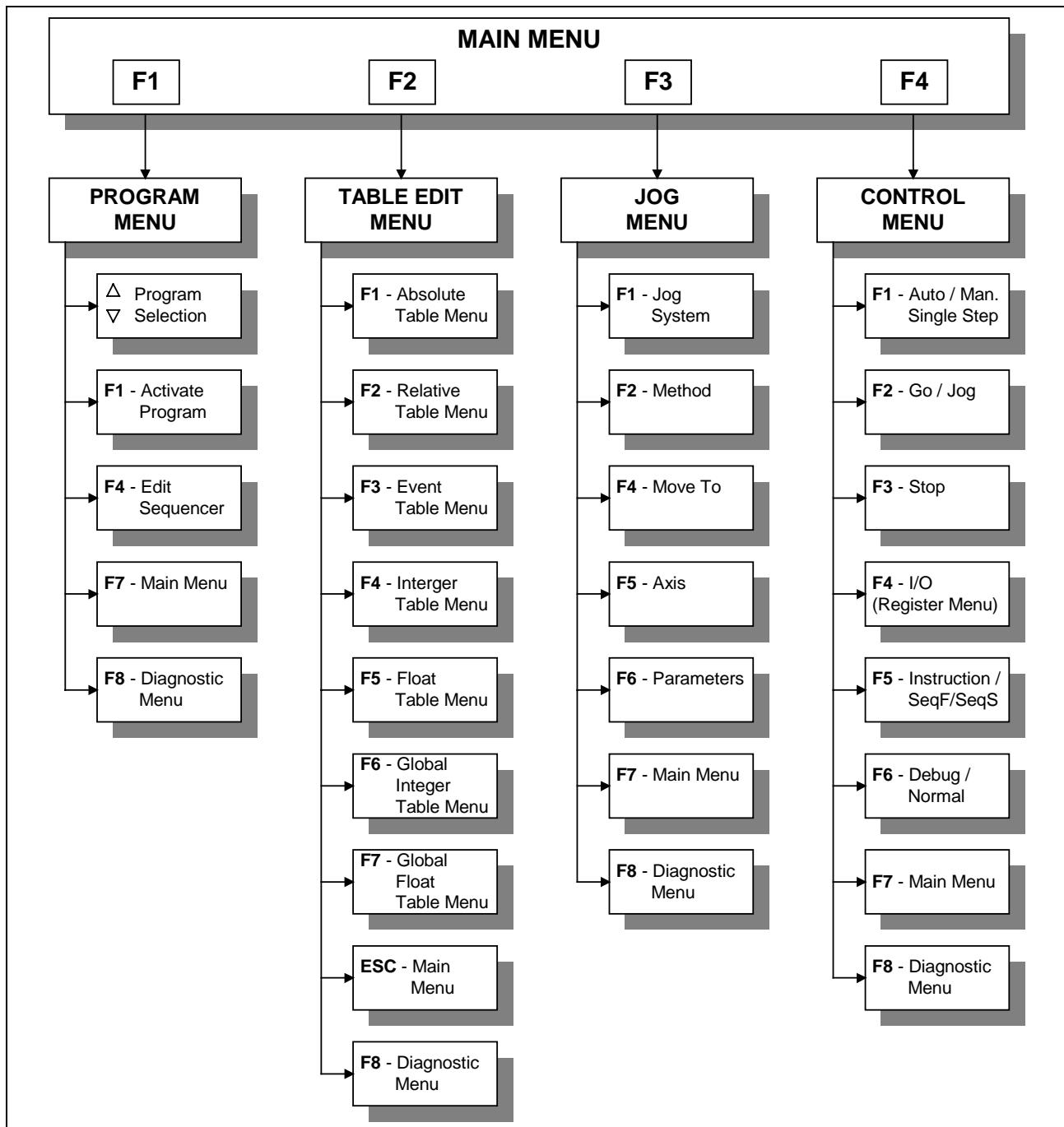


Figure 10-1: Menu Map (F1-F4)

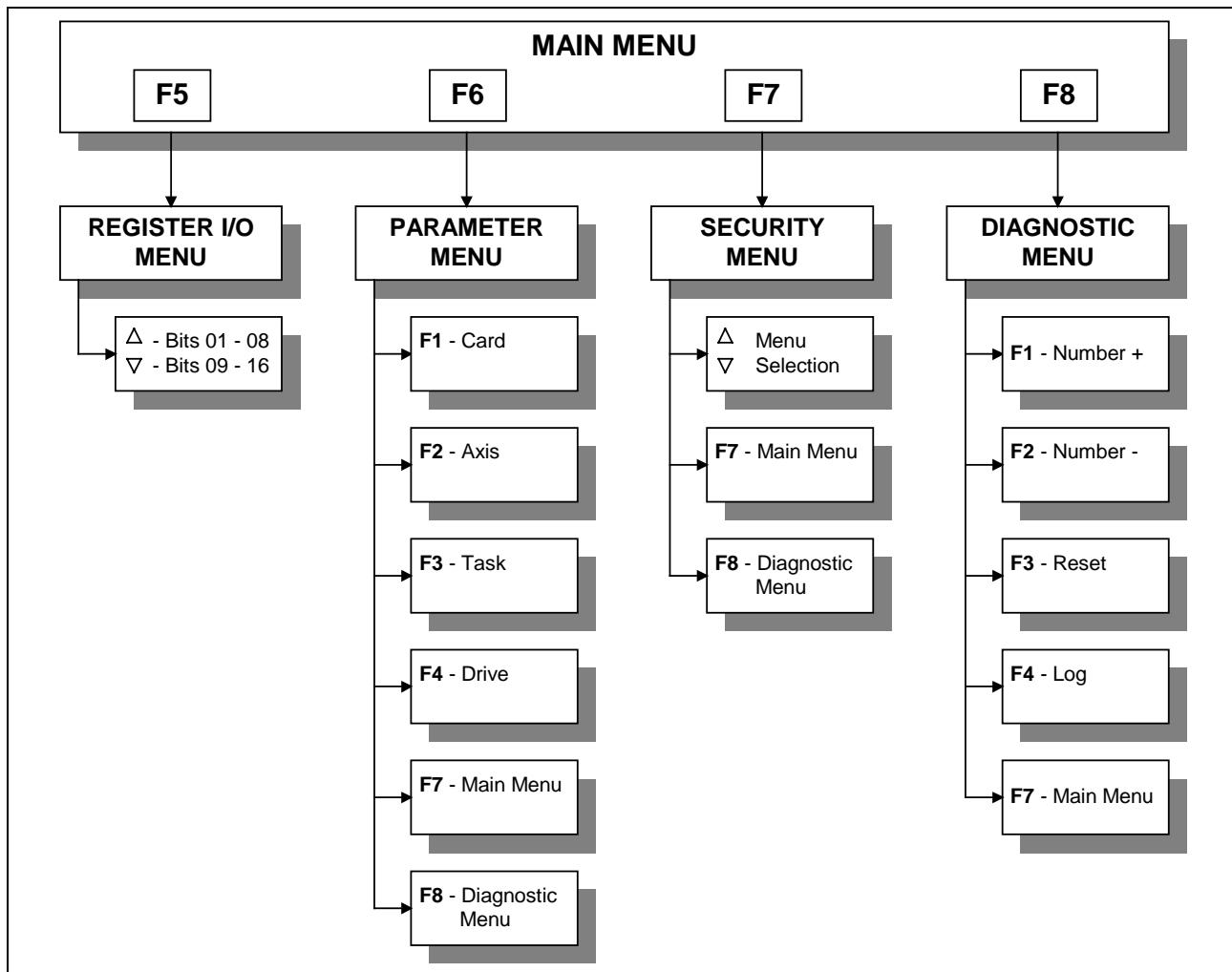


Figure 10-2: Menu Map (F5-F8)

10.3 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.

10.4 BTC06 Keyboard Operation

The following defines the keys for the Teach Pendant:

Key	Action
F1	Soft key defined by active menu
F2	Soft key defined by active menu
F3	Soft key defined by active menu
F4	Soft key defined by active menu
F5	Soft key defined by active menu
F6	Soft key defined by active menu
F7	Soft key defined by active menu
F8	Soft key defined by active menu
L1 R1	left and right refresh of the BTC06 screen
HELP	First press function key help, then press item help (only available for Parameter Menu)
MAIN MENU	Display all main menu functions
A+ A-	Jog A coordinate plus/minus
B+ B-	Jog B coordinate plus/minus Main Menu: respectively turns backlighting on and off
C+ C-	Jog C coordinate plus/minus
X+ X-	Jog X coordinate plus/minus
Y+ Y-	Jog Y coordinate plus/minus
Z+ Z-	Jog Z coordinate plus/minus
0	numeric key
STU 1	numeric key and letter combination (use the shift key to access letters)
VW 2	numeric key and letter combination
XYZ 3	numeric key and letter combination
JKL 4	numeric key and letter combination
MNO 5	numeric key and letter combination
PQR 6	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.

F1	F2	F3	F4	F5	F6	F7	F8	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.

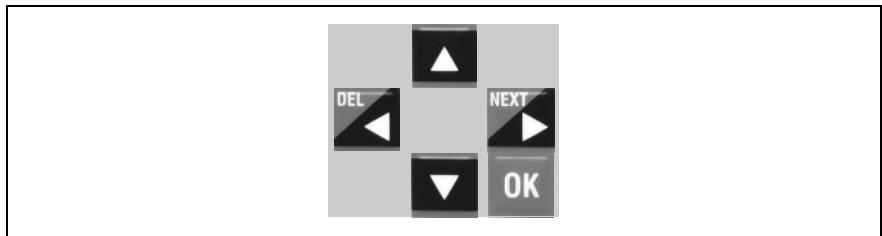


Fig. 10-1: Cursor Control and Editing

Number or Letter Selection

The number keys labeled 1 - 9 also double as letter keys when the shift  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.

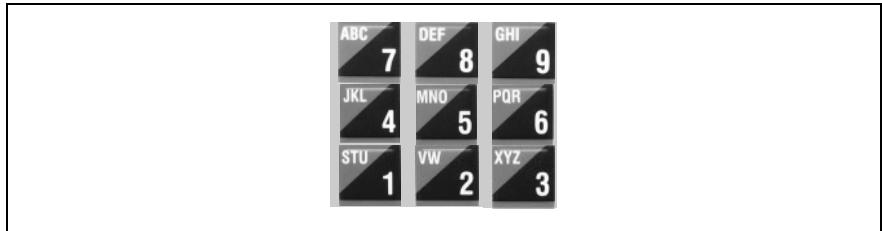


Fig. 10-2: Using the Shift key

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.

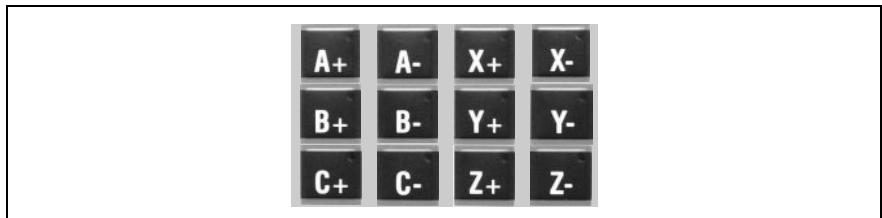


Fig. 10-3: Jogging Control

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.

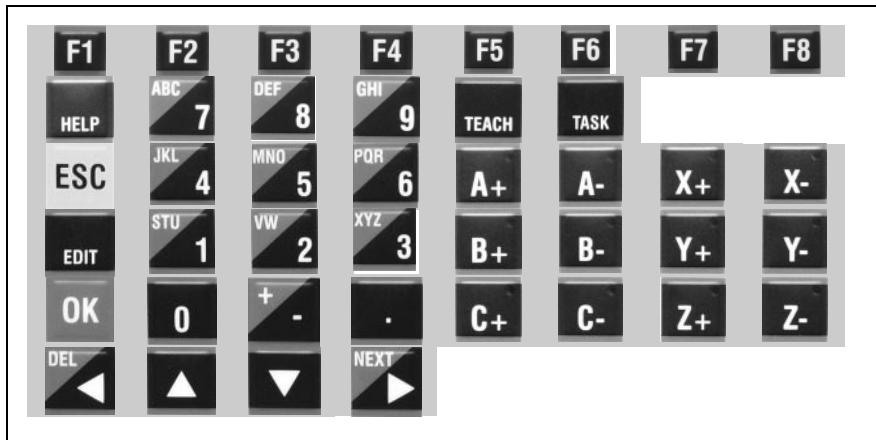


Fig. 10-4: Teach Control

10.5 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
Actv			Edit	
			F6	F7
			Main	Diag

The up and down arrow keys move the cursor to select a program. Pressing **F1** activates the selected program.

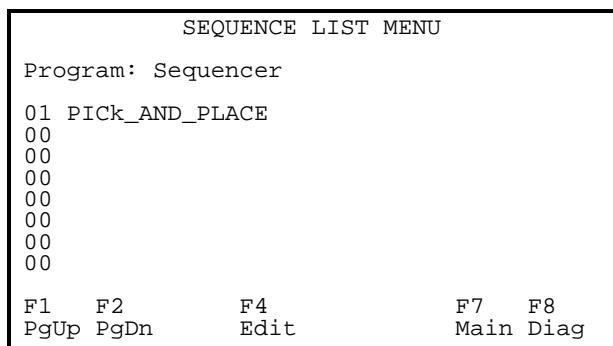
Note: The currently active program must not be running when activating another program.

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*.



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 10-9 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

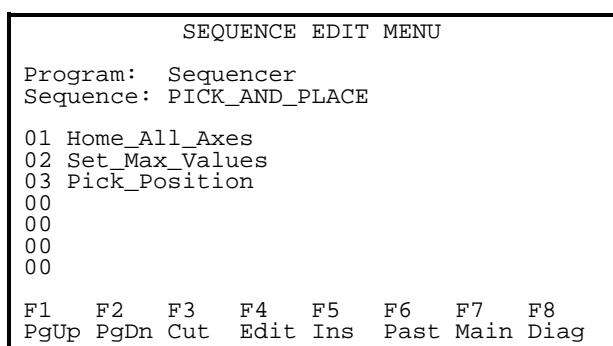
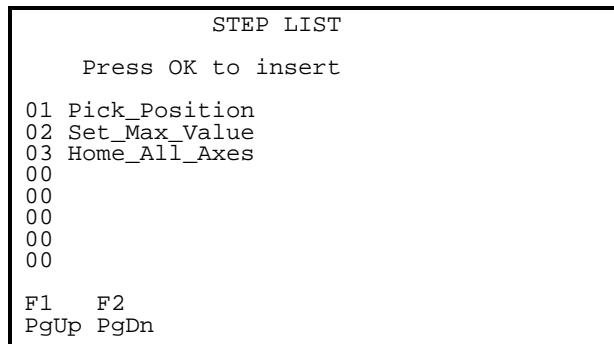


Table List Menu

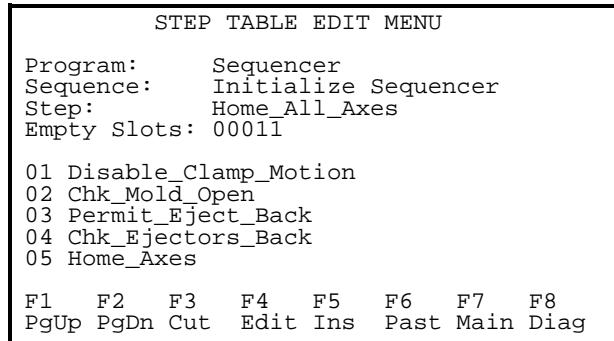
Pressing **F5** (Ins) from the *Sequence Edit Menu* will open the *Step List Menu*. This menu contains a list of all the step tables available within the selected Sequence. Use the arrow keys to navigate the cursor to the

desired *Step Table*. Press OK to insert that function into the previous *Sequence*.



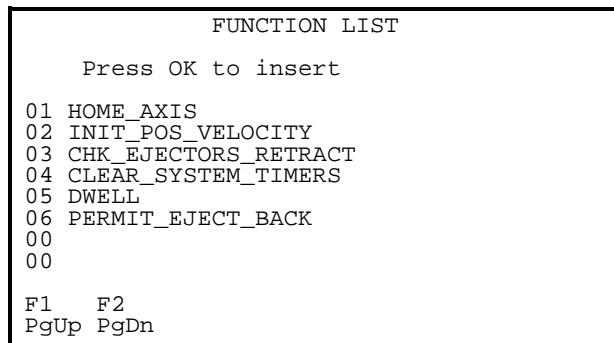
The Step Table Edit Menu

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



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 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

10.6 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 Inpt	F2 DcPt	F3 Home	F4 End	F6 Jog	F7 Main	F8 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 10-16

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 PgUp	F2 PgDn	F3 Home	F4 End	F5 Edit	F6 Jog	F7 Main	F8 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 Inpt	F2 DcPt	F3 Home	F4 End	F6 Jog	F7 Main	F8 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 10-16

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 PgUp	F2 PgDn	F3 Home		F5 Repl	F7 Main	F8 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
PgUp	PgDn	Fmt
F7	F8	
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
PgUp	PgDn	Fmt
F7	F8	
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
PgUp	PgDn	Fmt
F7	F8	
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
PgUp	PgDn	Fmt
F7	F8	
Main	Diag	

Display Format

Pressing **F3** toggles the display between float fixed (100.000) and scientific (1.000e+02) notation.

10.7 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 Syst	F2 Meth	F4 MvTo	F5 Axis
		F6 Para	F7 Main
			F8 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, World or Tool jog system. **F2** selects the jog method which can be continuos 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.

Tool Jog Menu

The **Tool** Jog menu allows jogging of the position of the end of a robotic arm.

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.

10.8 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:

Manual: Jog \Rightarrow Auto: Step \Rightarrow Auto: Run/Hold \Rightarrow Auto: Step \Rightarrow



F1 = Auto
F6 = Debug/Norm

F2 = Go
F7 = Main Menu

F3 = Stop **F4** = I/O
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	
INST: 00FC AXIS_WAIT	
STAT: Task Running	
DIAG: Task Running	

Screen name

VisualMotion instruction being processed

Task status

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							
INST:	0194 SET						
STAT:	In return position						
DIAG:	Instruction Sin			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
Step	Go	Stop	I/O	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.

10.9 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.

10.10 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

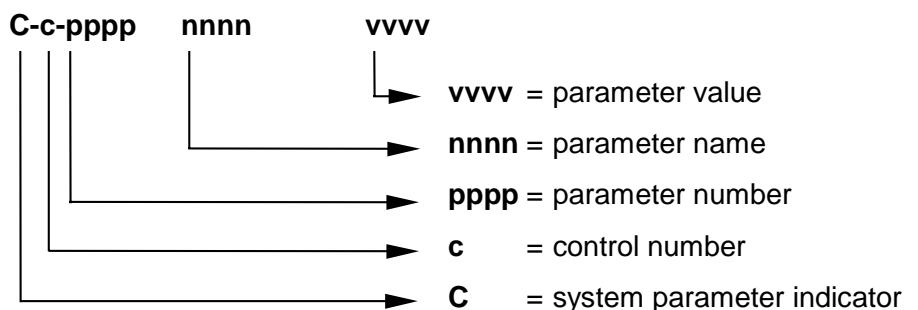
```

Selecting **F1-F4** will open one of the following Parameter screens.

F1 - Card Parameter Screen

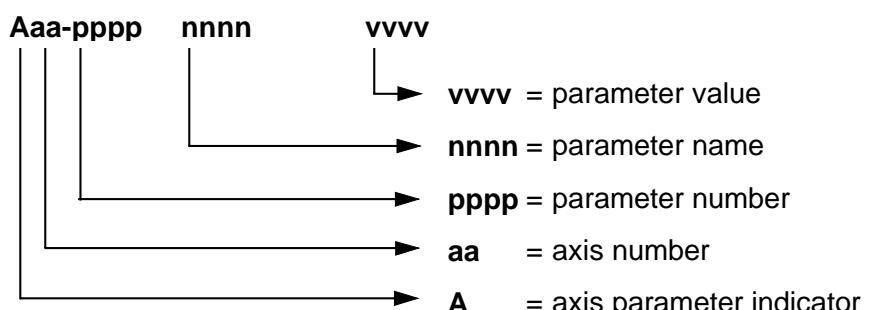
CARD PARAMETER MENU

C-c-ppppp	nnnnn	vvvvv			
C-0-00001	Language Selection	1			
...					
...					
...					
...					
...					
...					
...					
...					
C-0-00042	World Large Increment	50			
F1	F2	F3	F4	F7	F8
Home	End	PqUp	PqDn	Main	Diaq



For changing of Card parameter numbers and editing of values, refer to **F2 - Axis Parameter Screen** on page 10-26.

F2 - Axis Parameter Screen



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.

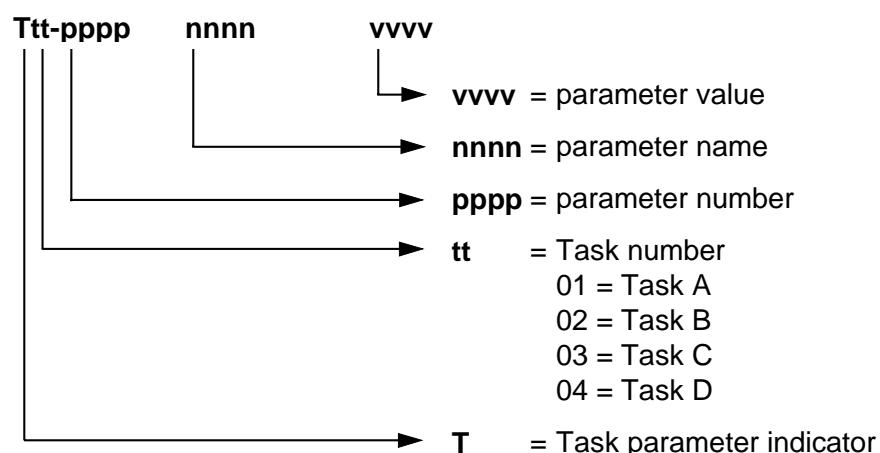
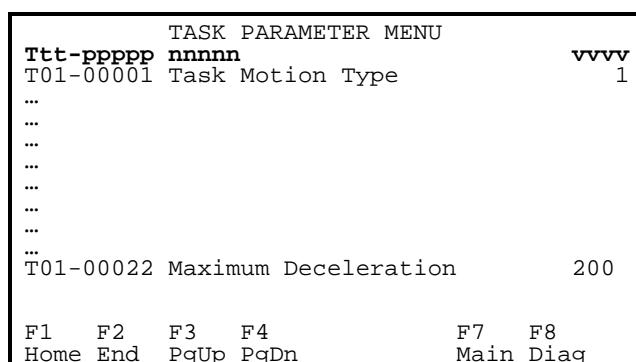
To jump to a given parameter number:

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.

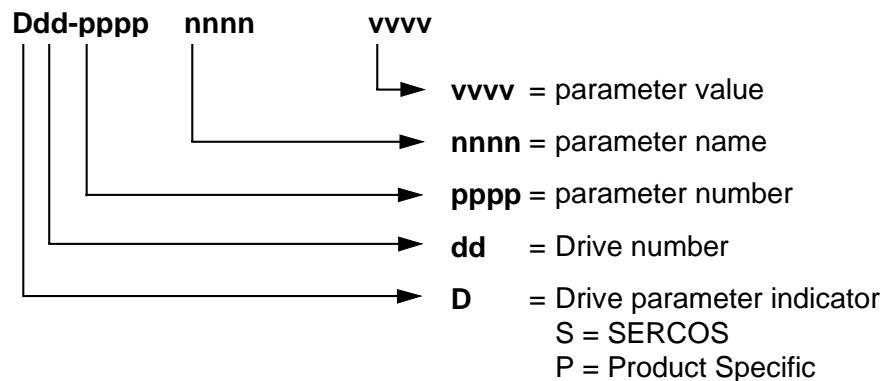
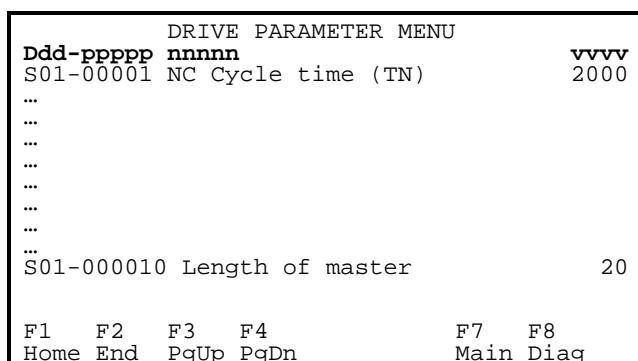
To modify the value for a given parameter, press the **F2 End** key to jump to the value field. Press the Edit key, enter the new value and press OK.

F3 - Task Parameter Screen



For changing of Task parameter numbers and editing of values, refer to **F2 - Axis Parameter Screen** on page 10-26.

F4 - Drive Parameter Screen



For changing of Drive parameter numbers and editing of values, refer to **F2 - Axis Parameter Screen** on page 10-26.

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.

10.11 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	

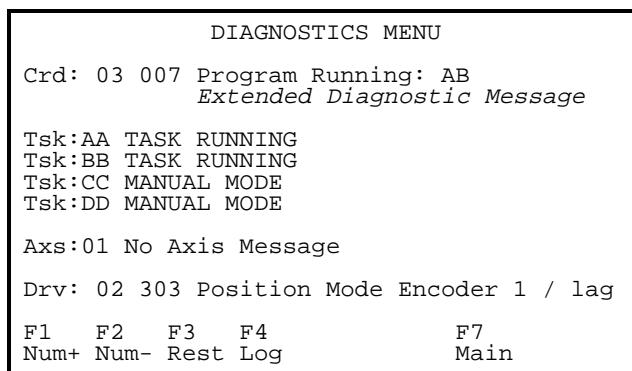
To alter a menu protection level, place the cursor over the protection level field and press the EDIT button. Key-in the appropriate code (0, 1 or 2) and press OK. The Security Level Menu has a default of -1 to allow initial access to all users.

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

10.12 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.



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

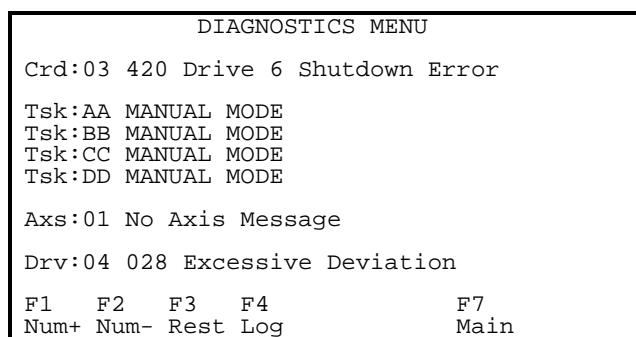
For a complete listing of Diagnostics, refer to the *VisualMotion 8 Trouble Shooting Guide, Monitoring and Diagnostics*.

10.13 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.



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.

11 Textual Language Programming

11.1 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.

11.2 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

11.3 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 GIx, 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 GIx, 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 equ1
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 Ix, I[x] - global variable GIx, 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 Ix, I[x] - global variable GIx, 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:

$$\text{Slave axis velocity} = \text{Master axis velocity} * (\text{Sfactor/Mfactor})$$

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 GIx, GI[x] - label	1 to the maximum number of valid axes	master axis number
axis2	integer - constant - variable Ix, I[x] - global variable GIx, 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 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      =      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, Ix, GFx, GIx, integer label, float label or point label		values passed to function
retval (optional)	Fx, Ix, GFx, GIx, 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 GIx, GI[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, GIx, 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 GIx, 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 GIx, 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 checked. 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
```

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 ; 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 GFx	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: - G#_SYNC_ACCEL - G#_H_LOCKOFF - G#_SYNC_VEL - G#_H_USER - G#_M1 - G#_LOCK_WIN - G#_N1 - G#_STOP_DECEL - G#_REL_M_PH - G#_JOG_ACCEL - G#_REL_S_PH - G#_JOG_VEL - G#_ABS_M_PH - G#_JOG_INC - G#_ABS_S_PH - G#_JOG_ABS - G#_H_LOCKON - G#_JOG_WIN - G#_H_RUN - G#_LOCK_OFFSET
integer_block	integer - variable Ix, I[x]	use default values	Starting block for the following 9 integers: - G#_CONFIG - G#_LOCKON_CAM - G#_MSTR1_AXIS - G#_RUN_CAM_ID - G#_MSTR2_AXIS - G#_LOCKOFF_CAM - G#_ACTIVE_STATE - G#_USER_CAM - G#_ACTIVE_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 GIx, GI[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 Ix, I[x], GIx, GI[x] - label	1 to the maximum number of valid axes	axis number of slave
mode	integer - constant Ix, I[x], GIx, GI[x] - label	1 = single-axis mode 2 = synch. to master 3 = velocity	

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
.
ELS/INIT      1, 1, 2, 1, 1 ; ELS axis initialization
ELS/MODE      1, 1           ; Single Axis mode
ELS/DECEL    F1             ; Sets deceleration rate
```

ELS/MASTER

The *ELS/MASTER* instruction is used to identify the starting blocks for the required floats and integers that will be used by VisualMotion's 6 ELS Masters.

Syntax:

```
ELS/MASTER   float_block, integer_block
VAR/INIT       (required for floats)
VAR/INIT       (required for integers)
```

Note: In order to compile the 18 floats and 24 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
float_block	float - variable Fx, F[x]	use default values	Labels for the ELS Master's 18 floats: (# = 1,2,3,4,5,6) - ELS_MSTR_FREQ# - ELS_MSTR_M# - ELS_MSTR_N#
integer_block	integer - variable Ix, I[x]	use default values	Labels for the ELS Master's 24 integers: - ELS_MSTR_A# - ELS_MSTR_E# - ELS_MSTR_FLTR# - ELS_MSTR_TYPE#

Example:

```
ELS_MASTER   F140,I110
VAR_INIT     I110, 1, 2, 1, 4, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0,
              0, 0, 0, 0, 3, 3, 0, 0, 1, 1
VAR_INIT     F140, 0, 0, 0, 0, 0, 0, 0, 2, 5, 7, 9, 0, 0,
              3, 6, 8, 10
```

EQU (Equate)

The *EQU* directive permits you to assign a symbolic name to a literal constant. Equating is most beneficial when using a constant repeatedly. An equated symbol name is global to all VisualMotion tasks, subroutines and events within your user program. The name may have up to 20 ASCII characters. Remember that no spacing is allowed within the name.

EQU instructions should be placed at the very beginning of your program and must appear before the name is referenced. If you need to change

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
.
.
.
EVENT/TRIGGER 3      ;Trigger event 3
EVENT/WAIT 3          ;Wait until done
.
.
.
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 Ix, I[x] - global variable GIx, GI[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 Ix, I[x] - global variable GIx, GI[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 (optional)	integer - constant - label label		return value from function

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, GI[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
.
.
.
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
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/STAUTS 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 Ix - global variable GIx - label	a valid absolute or relative point table entry	defines the mid-point of the circular arc
index2	integer - constant - variable Ix - global variable GIx - 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 slew 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 Ix - 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 GIx - 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], GIx, 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], GIx, 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 or float - constant - variable Fx, F[x] - global variable GFx, GF[x], GIx, GI[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, Gl[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 Ix - global variable GIx - 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 then
          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 = slew (constant speed) 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, Ix, 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 its scalar.

Feedback and output may be a program or global variable(Fx, GFx, Glx, Ix, 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], Ix, I[x] - global variable GFx, GF[x], Glx, GI[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], Ix, I[x] - global variable GFx, GF[x], GIx, GI[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], Ix, I[x] - global variable GFx, GF[x], GIx, GI[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_crtl_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, Glx, I[x], Gl[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, Glx	a valid I/O register	source register to test
bit	integer - constant - label - variable Ix, Glx	1 to 16	bit within register to test
target	integer - variable Ix, I[x], Glx, 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 Ix	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 Ix	0-360	position in which switch element turns on
off position	integer - constant - variable Ix	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].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_POSITION	;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, ...} 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 $\text{REL}[4] = \{0, 0, 10, 0, 0, 0, \dots\}$ 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 GIx - 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 its 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 A's program instructions  
    ;  
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 GIx, GI[x] - label	1 and 2	Up to 2 Virtual Masters can be configured Each Virtual Master requires its 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: - VM#_HOME_POS - VM#_REL_MOVE_DIST - VM#_STOP_POS - VM#_CMD_ABS_POS - VM#_CMD_VEL - VM#_CMD_ACCEL - VM#_CMD_DECEL - VM#_E_STOP_DECEL - VM#_MAX_VEL - VM#_MAX_ACCEL - VM#_MAX_DECEL - VM#_JERK_ENABLE - VM#_CUR_POS - VM#_CUR_VEL - VM#_POS_WIN
integer_block	integer - variable Ix, I[x]	use default values	Starting block for the following 9 integers: - 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
.
```


12 Direct ASCII Communication

12.1 Overview

Rexroth Indramat's VisualMotion control can send and receive drive parameters, system parameters, user programs, and tables by means of the serial port. By using the text-based protocol described in this chapter, a variety of devices and programs can communicate with the PPC-R. The protocol also supports ASCII PC ISA/ESIA communication.

12.2 VisualMotion Communication Protocol

The ASCII protocol format for VisualMotion is designed so that all serial transmissions are similar in structure, facilitating simple coding/decoding routines. The protocol is the same for sending or receiving, making the handling and tracking of data easier.

All serial communications use the following standard protocol template. Information is structured using a three level system:

- command class
 - subclass
 - data identifier

```
>1 CS s.n 0123...CDEF $cs\r\n
||||||||| | | | |
||||||||| | | |_ End of Message(i.e., cr/lf)
||||||||| | | | (\r=0x0D and \n=0x0A)
||||||||| | | |_ Two Digit ASCII Checksum
||||||||| | _____|_ Variable Length Data Field
|||||||_| Number Within Set Data Identifier
|||||||_| ASCII period (0x2E)
|||||||_| Set Type Data Identifier
|||||||_| optional ASCII space character (0x20)
|||||_| Command Subclass
||||_| Command Class
|||_| required ASCII space (0x20)
||_| Network Address (i.e. card position identifier)
|_| Start Character (0x3E)
```

Start Character	The start of each message is identified by the ">" character (ASCII 62 decimal, 3E hex).
Network Address	The network address can be used to support data transfer across a communication system to multiple controls. The network address must be followed by an ASCII space character (0x20).
Data Identifier	<p>The Data Identifier field is a variable length field used to identify the data set being sent or requested.</p> <p>This identifier is in the following format:</p> <ul style="list-style-type: none">• s.n• s.n.x <p>where,</p> <p>s = the set identifier; this may be a program handle, drive number, task ID, etc.</p>

n = the numeric identifier used for items such as parameter numbers, table indexes and register numbers.

x = the Step x identifier used for parameter lists. No white space is allowed between the identifiers and the separating dot operators.

Variable Length Data	This field contains the actual data being sent to or received from VisualMotion. All subclasses have read and write capability. The same strings can be used for both responses and downloads.
-----------------------------	--

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 a 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 = FFFFFFFD 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.

12.3 Command Classes/Subclasses

The tables below list the command class identifier and the subclass identifier for each of the available subclasses within the command class.

Parameters

Command Class	Command Subclass
A - VisualMotion Axis Parameters	A - Attributes
	B - Block List Parameter
	D - Lists/Tables
	H - Upper Limit
	L - Lower Limit
	P - Parameter Data
	T - Name Text
	U - Units Text
C – VisualMotion Control Parameters	(Same Subclasses as VisualMotion Axis Parameters)
D - SERCOS Drive Parameters	(Same Subclasses as VisualMotion Axis Parameters)
T - Task Parameters	(Same Subclasses as VisualMotion Axis Parameters)

Variables

Command Class	Command Subclass
F - Float Variables	P - Data
	T - Text Label
G - Global Integer Variables	(Same Subclasses as Float Variables)
H - Global Float Variables	(Same Subclasses as Float Variables)
I - Integer Variables	(Same Subclasses as Float Variables)

PID

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

Point Tables

Command Class	Command Subclass
X - Absolute 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
Y - Relative Point Table	(Same Subclasses as X Absolute Point Table)

Event Tables

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

Program Communication

Command Class	Command Subclass
P - Program	A - Activate Program / Request Current Program
	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

Functions

Command Class	Command Subclass
K - ELS Functions	G - Group Registers and Variables
	L - List of Drives Assigned to Group
	M - Master Variables
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

I/O Registers

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
	S - Binary Forcing State
	T - Name Text
	X - Current State in Hexadecimal

Sequencer Data

Command Class	Command Subclass
L - Sequencer List	T - Name of Sequence List
	P - Print or Store Sequence Tables
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

PLS

Command Class	Command Subclass
W - PLS	A – PLS Number
	E - An Element's On Value
	F - An Element's Off Value
	G - Switch Lead Time
	H - On & Off Value and Lead Time
	M - Mask Register
	O - Phase <u>Offset</u>
	R - Assigned <u>Register</u>
	T - Master <u>Type</u>

Zones

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

12.4 VisualMotion and Drive Parameters and Subclasses

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.

VisualMotion control parameters are accessed using parameter set C1.

- VisualMotion axis parameters use the parameter sets A1 through A4.
- Drive parameters are accessed using parameter sets D1 through D4.
- VisualMotion task parameters use the parameter set Task C.

Note: Additional information on the parameter sets can be found in chapter 4, Parameters, of this manual, and in Rexroth Indramat's digital drives and SERCOS manuals.

Example:

```
>1 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
```

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.

Upper Limit, L: Lower Limit 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.

Attribute Subclass

The A subclass request 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 Lists Subclasses

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. Parameter list formats are described in Parameter Lists.

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., 1.95). 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
```

12.5 Parameter 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.

List Parameter Command:

```
>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= card, 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 of Parameter List request:

- 1) The Host requests a List Parameter:

```
>1 DD 1.17.0 \r\n ;Parameter S-0-17
```

VisualMotion responds with length of the list:

```
>1 DD 1.17.0 180 $cs\r\n ;180 parameters in list
```

- 2) The Host requests the first parameter in the list:

```
>1 DD 1.17.1 \r\n
```

VisualMotion responds with the first parameter:

```
>1 DD 1.17.1 44 $cs\r\n
```

- 3) The Host requests the second parameter in the list:

```
>1 DD 1.17.2 \r\n
```

VisualMotion responds with the second parameter:

```
>1 DD 1.17.2 104 $cs\r\n
```

.

.

.

- 180) The Host continues to sequentially request items in list.

- 181) 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
```

Parameter List Block Transfer

Classes: C, A, D, T

Subclass: B

Data Type: List of space-delimited strings with ASCII integers or floats

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.

```
>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 ((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 "!55 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:

- 0) Host requests a list parameter using block transfer

```
>1 DB 1.32840.0 \r\n ;Parameter P-0-72 (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
```

- 1) Host requests first 16 elements in list:

```
>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
```

- 2) Host requests elements 17-32:

```
>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
```

- 3-64) Host continues to request items in list as above.

- 65) 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 ((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:

- 0) Host starts sending a list parameter using block transfer

```
>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
```

- 1) Host sends first 16 elements in list:

```
>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
```

- 2-64) Host continues to send items in list as above

- 65) 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 ((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:

- 0) Host requests control cam #1

```
>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)
```

- 1-129) Host requests data and VisualMotion sends

```
>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
```

130) Host closes list

```
>1 CB 1.3101.130 \r\n
>1 CB 1.3101.130 !19 List is finished $cs\r\n
```

12.6 User Program Variables

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.

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.

'P': Data

Type: Float ("F") or Integer ("I")

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.

'T': Label Text

Type: String

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.

Examples:

The Host requests the number of integer variables used by program 1:

```
>1 IP 1.0 \r\n
```

VisualMotion responds with:

```
>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 with:

```
>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
```

12.7 PID

Proportional Integral Derivative (PID) information is only available from the active program.

Getting the assigned variables

The host request 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 request 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 request 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 request 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 not used

```
>0 ML 0.0 0 $cs\r\n
```

Control Blocks Variables

The control block variables are defined by the type of PID loop. The usage for type 1 is given above.

12.8 Point Tables

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. All the elements of a point in either the absolute or 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 12-5.

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

Examples:

```
>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, each of 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.

Examples:

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\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
;where X Y Z B S A D J 1 2 3 4 R P W E
```

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 (V)

Label:	integer followed by string of <= 21 byte
Set:	program handle
Number:	index for list
Step:	not used

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.

12.9 Event Tables

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.

General Format:

```
>1 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 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	Unsigned Integer	Read/write

Examples:

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 with:

```
>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
```

12.10 VisualMotion's User Program Communication

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.

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 C1.50 (Download Block Size) and defaults to 16 bytes. The Download Block Size parameter may be changed from 1 to 512 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 with:

```
>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
```

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
```

VisalMotion 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 with:

```
>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 card:

```
>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 12-1: 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 256K of available unused memory remains. VisualMotion requests a compression following a program download, program delete or program activation.

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 Istart_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 12-2: 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 with:

```
>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.)

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 new program's tables are set to a 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".

Format:

```
>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 110000000000000000 \r\n
```

VisualMotion responds with:

```
>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
```

Response Format:

```
>1 PV h.s Ixx Label
| | |_ variable label
| | |_ variable id number (index number)
|_ variable type identifier (F = float, I =
integer)
```

Example:

The Host requests the variables label list for program 1:

```
>1 PV 1.0 \r\n
```

VisualMotion responds with the count of named variables for program 1:

```
>1 PV 1.0 2 $cs\r\n
|_ 2 labels are defined
```

Host requests the first label:

```
>1 PV 1.1 \r\n
```

VisualMotion responds with:

```
>1 PV 1.1 F11 axis1_velocity $cs\r\n
```

The Host requests the next label

```
>1 PV 1.2 \r\n
```

VisualMotion responds with:

```
>1 PV 1.2 I1 part_count $cs\r\n
```

Host closes the list:

```
>1 PV 1.3 \r\n
```

VisualMotion closes the list and acknowledges:

```
>1 PV 1.3 !19 List is finished $cs\r\n
```

Initializing a Download (PW)

To start a download from the Host to the control, the Host sends a Header Record with the "PW" command. If the available control memory is insufficient to contain the program, VisualMotion returns an error message. If enough memory is available, VisualMotion responds with a numeric program handle. The program handle returned by VisualMotion must be used to identify the program with subsequent commands. VisualMotion will always respond with a new handle, allowing more than one program with the same name.

Example:

The Host starts a download:

```
|<-----Header Record----->|
>1 PW 0.0 1592 ABCD1234 12/25/92 Program_1 $cs\r\n
|           |           |           |           |__ Program Name
|           |           |           |           |__ (variable length)
|           |           |           |           |__ Date (mm/dd/yy)
|           |           |           |           |__ Program checksum (hex)
|           |           |           |           |__ Program size in bytes (decimal)
|__ Command: Start Download
```

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".

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 with:

```
>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 with:

```
>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()

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. Each time data is stored, a counter is incremented. 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.

12.11 Function Classes

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.

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 the staring float and integer variables:

```
>0 KM 0.0 F140 I110 \r\n
```

S Class

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.

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)

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.

For 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.

12.12 Input/Output Registers

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.

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 (RM, 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.

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:

```
>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 Read

Example:

Host requests the contents of register 1 in binary:

```
>1 RB 0.1 \r\n
```

VisualMotion responds with:

```
>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 all the 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 000000001001000
      |__|_ 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
```

12.13 Sequencer Data

VisualMotion's serial protocol includes the following classes and subclasses to handle the sequencer and functions with arguments.

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

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)**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)

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
```

12.14 PLS

Class	Subclass
W: PLS(Programmable Limit Switch)	A - Axis Number
	E - An Element's On Value
	F - An Element's Off Value
	G - Switch Lead Time
	H - On & Off Value and Lead Time
	M - Mask Register
	O - Phase Offset
	R - Assigned Register
	T - Master Type

Table 12-3: PLS Class and Subclass

Format:

```
>1 Wn h.n\r\n
    || | _number: Switch number (1 or 2)
    || | _set: Program handle
    || _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 4:

```
>1 WR 4.1 \r\n
```

VisualMotion responds register 100:

```
>1 WN 4.1 100
```

Format of Configuration Data

A - Axis	Axis Number(1 - 32)	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

```
>1 WH h.n.m\r\n
    || | |_element
    || | |_number: Switch number
    || | _set: Program handle
    || |_subclass: Switch Values
    |_class: PLS
```

The number of elements per switch can be requested by sending a request with element equal to 0.

Format of Switch Values

Example:

Host requests the on position for PLS 1, element 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 PLS 1, element 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 PLS 1, element 6, of program 4:

```
>0 WG 4.1.6 \r\n
```

VisualMotion responds:

```
>0 WG 4.1.6 65 \r\n\
```

Host requests on/off positions and lead time of element 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 requests on/off positions and lead time for PLS 1, element 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 Rexroth Indramat digital drives using the specified firmware have Drive PLS functionality:

- DIAX04 using ELS05VRS or greater allow up to 8 PLS switches.
- ECODRIVE03 using SMT01VRS allow up to 8 PLS switches.
- ECODRIVE03 using SMT02VRS or SGP01VRS or greater allow up to 16 PLS switches.

Elements of the PLS can be accessed individually or as a list. The available elements are described in the PLS Help in Visual Motion.

List of PLS Elements

Host requests the count of On element's for drive 1:

```
>1 DD 1.32900.0 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.0 8
```

Host requests the first On element for drive 1:

```
>1 DD 1.32900.1 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.1 25
```

Host requests the last On element for drive 1:

```
>1 DD 1.32900.8 \r\n
```

VisualMotion responds:

```
>1 DD 1.32900.8 350
```

Host requests one more element 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 elements

The subclass 'E' of the ASCII protocol for drive parameters allows individual elements of the list to be accessed.

Host requests the 5th Off element for drive 1:

```
>1 DE 1.32901.5 \r\n
```

VisualMotion responds:

```
>1 DE 1.32901.5 270
```

Host writes the 4th Offset element for drive 3:

```
>1 DE 3.32902.4 220\r\n
```

VisualMotion responds:

```
>1 DE 3.32902.4
```

12.15 Zone Tables

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.

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

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 Index

A

- A-0-0001 Task Assignment 6-116
- A-0-0002 Type of Positioning 6-116
- A-0-0003 Axis Motion Type 6-117
- A-0-0004 Axis Options 6-118
- A-0-0005 Linear Position Units .. 6-123
- A-0-0006 Reference Options..... 6-124
- A-0-0007 Configuration Mode.... 6-125
- A-0-0009 Drive PLS Register 6-125
- A-0-0020 Maximum Velocity 6-126
- A-0-0021 Maximum Acceleration6-126
- A-0-0022 Maximum Deceleration6-126
- A-0-0023 Jog Acceleration..... 6-127
- A-0-0031 Control Cam/Ratio Master Factor (N)..... 6-128
- A-0-0032 Control Cam/Ratio Slave Factor (M) 6-129
- A-0-0033 Control Cam Stretch Factor (H)..... 6-129
- A-0-0034 Control Cam Currently Active..... 6-130
- A-0-0035 Control Cam Position Constant (L) 6-130
- A-0-0036 Ratio Mode Encoder Type6-130
- A-0-0037 Ratio Mode Step Rate . 6-131
- A-0-0038 Ratio Mode Options.... 6-132
- A-0-0100 Target Position 6-134
- A-0-0101 Commanded Position.. 6-134
- A-0-0102 Feedback Position 6-135
- A-0-0110 Programmed Velocity . 6-135
- A-0-0111 Commanded Velocity.. 6-136
- A-0-0112 Feedback Velocity..... 6-136
- A-0-0120 Programmed Acceleration6-136
- A-0-0131 SERCOS Control Word6-137
- A-0-0132 SERCOS Status Word. 6-138
- A-0-0133 AT Error Count 6-138
- A-0-0140 Mfg. Class 3 Status Word6-139
- A-0-0141 Torque Mode Commanded Torque..... 6-139
- A-0-0142 Torque Feedback (cyclic)6-140
- A-0-0150 Programmed Ratio Adjust6-141
- A-0-0153 Control Phase Adjust Average Velocity..... 6-142
- A-0-0155 Control Phase Adjust Time Constant 6-142
- A-0-0157 Current Phase/ Control Cam Master Offset..... 6-143
- A-0-0159 Ratio Adjust Step Rate 6-143
- A-0-0160 Commanded Ratio Adjust6-144
- A-0-0161 Control Cam Programmed Slave Adjust..... 6-144
- A-0-0162 Control Cam Current Slave Adjust (Sph)..... 6-144
- A-0-0163 Control Cam Output Position 6-145
- A-0-0164 ELS Options 6-145
- A-0-0170 Probe Configuration Status6-147
- A-0-0171 Probe 1 Positive Captured Value 6-147
- A-0-0172 Probe 1 Negative Captured Value 6-148
- A-0-0173 Probe 2 Positive Captured Value 6-148
- A-0-0174 Probe 2 Negative Captured Value 6-148
- A-0-0180 Optional Command ID #16-148
- A-0-0181 Optional Command ID #26-149
- A-0-0182 Optional Command ID #36-149
- A-0-0185 Optional Feedback ID #16-149
- A-0-0186 Optional Feedback ID #26-149
- A-0-0190 Command Data #1 6-150
- A-0-0191 Command Data #2 6-150
- A-0-0192 Command Data #3 6-150
- A-0-0195 Feedback Data #1 6-150
- A-0-0196 Feedback Data #2 6-150
- A-0-0200 MDT Multiplex Selection List (DKC 2.3 only)..... 6-151
- A-0-0201 AT Multiplex Selection List (DKC 2.3 only)..... 6-152
- A-0-0202 MDT Multiplex Ident List (DKC 2.3 only)..... 6-152
- A-0-0203 AT Multiplex Ident List (DKC 2.3 only)..... 6-153
- A-0-2000 List of All Parameters ..6-153
- A-0-2001 List of Required Parameters 6-154
- About VisualMotion 7-96
- Absolute Point Edit 10-15
- absolute points table
 - data structure..... 8-23
- Absolute Table Menu 10-14
- Accel 8-10
- access PLS data via the Calc icon
 - Control PLS 4-35
 - Drive PLS 4-36
 - Option Card PLS 4-36
- Activating a Program (PA)..... 12-23
- Add a Drive 2-4
- Add a Drive I/O Card 2-4
- Add a RECO Module 2-4
- Add an I/O Modules..... 2-10
- Add an I/O Station 2-4
- Advanced restore
 - Add>>..... 7-31
 - All>>..... 7-31
 - Clear all prorams first 7-32
 - Remove 7-31
 - Remove all 7-32
 - Restoring a drive parameter 7-32
- Allowable Drive Address Range ...6-34
- Archive 7-30
- Assign registers to “User Configuration” I/O..... 2-5

Assigning a Register Starting Block2-
 16
 Assigning Register Numbers 2-15
 AT Error Count 6-138
 AT Multiplex Ident List (DKC 2.3
 only) 6-153
 AT Multiplex Selection List (DKC 2.3
 only) 6-152
 Attribute Subclass 12-10
 Available Program Memory 6-30
 Axis 8-11
 Axis Fast Jog Velocity 6-28
 Axis Large Increment 6-28
 Axis Motion Type 6-117
 Axis Options 6-118
 Axis Parameter Menu 10-26
 Axis Parameters 6-14
 Axis Feedback Capture
 (Registration) 6-16
 Axis Setup 6-14
 A-0-0025 Maximum Jog
 Increment 6-127
 A-0-0026 Maximum Jog Velocity
 6-128
 A-0-0030 Master Axis for Ratio
 Function 6-128
 Axis Status 6-15
 A-0-0145 Current Motion Type6-
 140
 Electronic Line Shaft 6-141
 A-0-0151 ELS Programmed
 Phase Offset 6-141
 Electronic Line Shafting 6-15
 Multiplexing Parameters (DKC 2.3
 only) 6-16
 Optional SERCOS Data 6-16
 Parameter Lists 6-16
 Axis Parameters: 6-116
 Axis Slow Jog Velocity 6-29
 Axis Small Increment 6-28
 AxisEvt 8-16

B

Backspaces and White spaces 12-4
 Bit Definitions
 Register 026
 Current Miss Counter 5-18
 Fieldbus Timeout Counter 5-18
 Peak Miss Counter 5-18
 Bootloader Firmware Version 6-35
 Branch 8-17
 Break Point 7-80
 BTC06 Data Transaction Word 6-56
 BTC06 Error Screen 10-31
 BTC06 jog method 10-20
 BTC06 jog system
 axis jog menu 10-20
 joint jog menu 10-20
 tool jog menu 10-20
 world jog menu 10-20
 BTC06 keyboard I/O map 5-25, 10-8
 BTC06 keyboard operation
 cursor control and editing 10-9
 jogging control 10-9
 number or letter selection 10-9
 task control 10-10
 teach control 10-10

BTC06 menus
 F1 Program Menu 10-11
 F2 Table Edit Menu 10-14
 F3 Jog Menu 10-19
 F4 Control Menu 10-21
 F5 Register I/O Menu 10-24
 F6 Parameter Menu 10-25
 axis parameter menu 10-26
 card parameter menu 10-26
 drive parameter menu 10-28
 task parameter menu 10-27
 F7 Security Menu 10-29
 F8 Diagnostic Menu 10-30
 BTC06 Message and Prompt Control
 Word 6-51
 BTC06 screen map 10-2
 BTC06 Teach Pendant
 Keyboard Operation 10-6
 Setup 10-4
 Build menu
 compile 7-20
 Display code 7-20
 Program management 7-21
 Save, compile, download 7-20

C

C-0-0001 Language Selection 6-17
 C-0-0002 Unit Number 6-18
 C-0-0003 Serial Port A Setup 6-18
 C-0-0004 Serial Port B Setup 6-19
 C-0-0009 Error Reaction Mode 6-19
 C-0-0010 System Options 6-20
 C-0-0012 Serial Port B Device Type6-
 22
 C-0-0013 Serial Port A Mode 6-23
 C-0-0014 Serial Port B Mode 6-23
 C-0-0016 Communication Time-out
 Period 6-24
 C-0-0020 Transmitter Fiber Optic
 Length 6-24
 C-0-0021 User Watchdog Timer 6-25
 C-0-0022 User Watchdog Task ID.6-25
 C-0-0035 PLC Communication Option
 6-26
 C-0-0042 World Large Increment .. 6-26
 C-0-0043 World Small Increment .. 6-27
 C-0-0045 World Fast Jog Speed ... 6-27
 C-0-0046 World Slow Jog Speed... 6-27
 C-0-0052 Axis Large Increment.... 6-28
 C-0-0053 Axis Small Increment.... 6-28
 C-0-0055 Axis Fast Jog Velocity... 6-28
 C-0-0056 Axis Slow Jog Velocity.. 6-29
 C-0-0090 Download Block Size 6-29
 C-0-0091 Total Program Memory.. 6-29
 C-0-0092 Available Program Memory
 6-30
 C-0-0093 Contiguous Program
 Memory 6-30
 C-0-0094 Maximum Executable
 Program Size 6-31
 C-0-0095 Path Planner to SERCOS
 Time Factor 6-31
 C-0-0098 Initialization Delay 6-32
 C-0-0099 Minimum SERCOS Cycle
 Time 6-32
 C-0-0100 Control Firmware Version6-
 33

C-0-0101 Control Hardware Version	6-33
C-0-0102 Control Version Date	6-34
C-0-0103 Allowable Drive Address Range	6-34
C-0-0104 Bootloader Firmware Version	6-35
C-0-0120 Operating Mode	6-35
C-0-0121 SERCOS Communication Phase	6-36
C-0-0122 Diagnostic Message	6-36
C-0-0123 Diagnostic Code	6-37
C-0-0124 Extended Diagnostic	6-37
C-0-0125 System Timer Value	6-38
C-0-0126 Date and Time	6-38
C-0-0127 Current PPC-R Temperature	6-39
C-0-0142 Card Label String	6-39
C-0-0200 Current Load due to Motion	6-39
C-0-0201 Peak Load due to Motion	6-40
C-0-0202 Current Load due to I/O	6-40
C-0-0203 Peak Load due to I/O	6-41
C-0-0300 Link Ring Control Word	6-41
C-0-0301 Link Ring Primary Fiber Optic Length	6-42
C-0-0302 Link Ring Secondary Fiber Optic Length	6-42
C-0-0303 Link Ring MDT Error Counter	6-43
C-0-0400 Card IP Address	6-43
C-0-0401 Card Subnet Mask	6-44
C-0-0402 Card Gateway IP Address	6-44
C-0-0403 Card CIF Network Control	6-44
C-0-0404 Card Access Network Control	6-45
C-0-0405 Card Network Password	6-46
C-0-0406 CIF Ethernet Card Hardware ID	6-47
C-0-0407 CIF Ethernet Card Firmware Version	6-47
C-0-0408 CIF Driver Version	6-47
C-0-0801 Pendant Protection Level 1 Password	6-48
C-0-0802 Pendant Protection Level 2 Password	6-48
C-0-0803 Pendant User Accessible Floats Section	6-48
C-0-0804 Pendant User Accessible Integers Section	6-49
C-0-0805 Pendant Start of User Accessible Registers	6-49
C-0-0806 Pendant End of User Accessible Registers	6-50
C-0-0807 Pendant Password Timeout	6-50
C-0-0810 BTC06 Message and Prompt Control Word	6-51
C-0-0811 User Task Controlled Menu ID for BTC06	6-53
C-0-0812 User Task Controlled Task ID for BTC06	6-54
C-0-0813 User Task Controlled Axis Number for BTC06	6-55
C-0-0814 BTC06 Data Transaction Word	6-56
C-0-0990 Exit to Monitor Prompt	6-58
C-0-0994 Shutdown Command for Flash Programming	6-59
C-0-0996 Clear Program and Data Memory	6-59
C-0-0997 Clear Diagnostic Log	6-60
C-0-2000 List of All Parameters	6-60
C-0-2001 List of Required Parameters	6-60
C-0-2010 List of SERCOS Devices	6-61
C-0-2011 List of SERCOS Drives	6-62
C-0-2012 List of SERCOS I-O Stations	6-62
C-0-2013 I/O Configuration List	6-62
C-0-2016 List of Virtual Axes	6-63
C-0-2017 I/O User Configuration List	6-64
C-0-2020 Diagnostic Log List	6-64
C-0-2021 Diagnostic Log Options	6-65
C-0-2501 Oscilloscope Signal 1 Type	6-66
C-0-2502 Oscilloscope Signal 2 Type	6-67
C-0-2503 Oscilloscope Signal 3 Type	6-67
C-0-2504 Oscilloscope Signal 1 ID Number	6-67
C-0-2505 Oscilloscope Signal 2 ID Number	6-68
C-0-2506 Oscilloscope Signal 3 ID Number	6-68
C-0-2507 Oscilloscope Signal 1 Axis Number	6-68
C-0-2508 Oscilloscope Signal 2 Axis Number	6-69
C-0-2509 Oscilloscope Signal 3 Axis Number	6-69
C-0-2510 Oscilloscope Sampling Rate	6-69
C-0-2511 Oscilloscope Signal 1 List	6-70
C-0-2512 Oscilloscope Signal 2 List	6-70
C-0-2513 Oscilloscope Signal 3 List	6-70
C-0-2514 Oscilloscope Sample Count	6-70
C-0-2515 Oscilloscope Trigger Post-count	6-71
C-0-2516 Oscilloscope Trigger Type	6-71
C-0-2517 Oscilloscope Trigger ID Number	6-72
C-0-2518 Oscilloscope Trigger Axis or Mask	6-72
C-0-2519 Oscilloscope Trigger Level or Mask	6-73
C-0-2520 Oscilloscope Trigger Mode	6-73
C-0-2521 Oscilloscope Trigger Source	6-73
C-0-2522 Oscilloscope Trigger Control Word	6-74
C-0-2523 Oscilloscope Trigger Status Word	6-74

C-0-2600 Fieldbus Mapper (cyclic channel) To PLC	6-75
C-0-2601 Fieldbus Mapper (cyclic channel) From PLC	6-76
C-0-2630 Fieldbus Slave Device Address	6-76
C-0-2631 Fieldbus Parameter Channel Length	6-77
C-0-2632 Fieldbus Multiplex Method	6-77
C-0-2633 Fieldbus Baud Rate (DeviceNet only).....	6-77
C-0-2635 Fieldbus Error Reaction	6-78
C-0-2636 Fieldbus Word Swap (DeviceNet only).....	6-78
C-0-2637 Fieldbus Slave Firmware Version.....	6-79
C-0-2638 Fieldbus Available Cyclic IN Parameters.....	6-80
C-0-2639 Fieldbus Available Cyclic OUT Parameters.....	6-80
C-0-2641 Indramat/PLC Input Register List	6-80
C-0-2642 Indramat/PLC Output Register List	6-81
C-0-2700 Fieldbus Mapper (non-cyclic channel) To PLC	6-81
C-0-2701 Fieldbus Mapper (non-cyclic channel) From PLC	6-82
C-0-2901 PLS 1 Start Output Register	6-83
C-0-2902 PLS 1 Start Mask Register	6-83
C-0-2903 PLS 1 Build Command ..	6-84
C-0-2904 PLS1 Build Status	6-84
C-0-2905 PLS1 Activate Command	6-85
C-0-2906 PLS1 Activate Status.....	6-85
C-0-2907 PLS1 Error Code.....	6-85
C-0-2908 PLS1 Extended Error Code	6-86
C-0-2909 PLS1 Hardware ID	6-86
C-0-2910 PLS1 Software ID	6-86
C-0-2920 PLS1 Switch On List.....	6-87
C-0-2921 PLS1 Switch Off List	6-87
C-0-2922 PLS1 Switch Output List	6-88
C-0-2930 PLS1 Output Master List	6-88
C-0-2931 PLS1 Output Lead Time List	6-88
C-0-2932 PLS1 Output Lag Time List	6-89
C-0-2933 PLS1 Output One Shot List	6-90
C-0-2934 PLS1 Output Mode List	6-90
C-0-2935 PLS1 Output Direction List	6-91
C-0-2936 PLS1 Output Hysteresis List	6-91
C-0-2940 PLS1 Master Type List..	6-92
C-0-2941 PLS1 Master Number List	6-92
C-0-2942 PLS1 Master Encoder List	6-93
C-0-2943 PLS1 Master Phase Offset List	6-93
C-0-3000 I/O Mapper Program	6-94
C-0-3001 I/O Mapper Options	6-94
C-0-3003 I/O Mapper Total Operations	6-95
C-0-3004 I/O Mapper File Size.....	6-95
C-0-3005 I-O Mapper Executable Size	6-96
C-0-3100 Cam Tags	6-96
C-0-3101 CAM Table 1 through C-0-3108 CAM Table 8	6-96
C-0-3109 CAM Table 9 through C-0-3140 CAM Table 40	6-97
Calc	8-19
card PLS	8-26
control PLS.....	8-27
direct addressing	8-21
drive PLS	8-27
events table	8-24
indirect addressing	8-21
operators	8-20
points table.....	8-23
zone table.....	8-25
Cam	
Number	8-30
CAM Builder	7-82
CAM Indexer	7-58
CAM Indexer Function Block Variables (PJ)	12-27
CAM Table 1 through C-0-3108 CAM Table 8	6-96
CAM Table 9 through C-0-3140 CAM Table 40	6-97
Cam Tags	6-96
Camming	8-28
Card Access Network Control.....	6-45
Card CIF Network Control.....	6-44
Card Gateway IP Address	6-44
Card IP Address	6-43
Card Label String	6-39
Card Network Password.....	6-46
Card Parameter Menu	10-26
Card Selection.....	7-92
Card Subnet Mask	6-44
CIF Driver Version	6-47
CIF Ethernet Card Firmware Version	6-47
CIF Ethernet Card Hardware ID	6-47
Circle	8-45
CLC Cycle Control Considerations..	5-8
CLC Table Edit Menu	10-14
Clear Diagnostics Log	6-60
Clear Program and Data Memory ..	6-59
Command	8-46
Command Classes/Subclasses Event Tables	12-7
Functions	12-8
I/O Registers	12-8
Parameters	12-5
PID.....	12-6
PLS	12-9
Point Tables	12-6
Program Communication	12-7
Sequencer Data	12-8
Variables.....	12-5
Zones	12-9
Command Data #1.....	6-150
Command Data #2.....	6-150
Command Data #3.....	6-150
Commanded Position	6-134
Commanded Ratio Adjust	6-144

Commanded Velocity	6-136
Commission	
Coordinated motion	7-27
Drives	7-24
Fieldbus mapper	7-25
I/O mapper	7-25
I/O setup	7-24
PLS	7-26
Transfer	7-32
Communication Errors	12-2
Communication Protocol for	
VisualMotion	12-1
Communication Time-out Period ..	6-24
Composite Instruction Pointer.....	6-112
Configuration	7-93
Configuration Mode	6-125
configure a Control PLS	
PLS Master configuration for a	
Control PLS	4-9
PLS register assignment for a	
Control PLS	4-10
switch configuration for a Control	
PLS	4-7
configure a Drive PLS	
PLS Master configuration for a	
Drive PLS	4-14
PLS register assignment for a Drive	
PLS	4-14
switch configuration for a Drive PLS	
.....	4-13
configure an Option Card PLS	
PLS Master configuration for an	
Option Card PLS	4-19
PLS outputs.....	4-23
PLS register assignment for an	
Option Card PLS	4-21
switch configuration for an Option	
Card PLS	4-18
Configure RECO02 Modules	2-12
Configuring I/O Devices Offline	2-2
Configuring I/O Devices Online.....	2-3
Connecting Icons.....	8-3
Contiguous Program Memory	6-30
Control Cam Current Slave Adjust	
(Sph)	6-144
Control Cam Currently Active....	6-130
Control Cam Output Position.....	6-145
Control Cam Position Constant (L)6-	
130	
Control Cam Programmed Slave Adjust	
.....	6-144
Control Cam Stretch Factor (H) ..	6-129
Control Cam/Ratio Master Factor (N)6-	
128	
Control Cam/Ratio Slave Factor (M)6-	
129	
Control Firmware Version.....	6-33
Control Hardware Version	6-33
Control Menu	
auto run/hold mode	10-22
auto step mode	10-23
manual mode	10-24
Control Parameters	6-5
BTC06 Teach Pendant	6-8
Cam Table	6-11
Control Processor Usage Status	
(GPP only).....	6-7
Fieldbus Interface.....	6-10
I/O Mapper	6-11
Internal System Monitoring	6-8
Jogging and Display.....	6-6
Oscilloscope.....	6-9
PLC Interface.....	6-5
Program Management.....	6-6
System Memory (GPP only).....	6-8
System Parameter Lists.....	6-9
System Setup	6-5
System Status.....	6-6
Control Phase Adjust Average Velocity	
.....	6-142
Control Phase Adjust Time Constant6-	
142	
Control PLS	
direct ASCII instructions	4-2
specifications	4-2
Control Serial Ports.....	7-94
Control Version Date	6-34
Conveyor Tracker.....	7-71
Coordinated Jogging	5-9
Coordinated motion	
Advanced restore	7-31
Archive	7-30
Jogging	7-28
Jogging acceleration	7-28
Pendant security.....	7-29
Task limits	7-27
Coordinated X Axis	6-101
Coordinated Y Axis	6-102
Coordinated Z Axis.....	6-102
Copy "Visible Configuration" to "User	
Configuration.....	2-5
Current Instruction	6-111
Current Instruction Pointer	6-111
Current Load due to I/O	6-40
Current Load due to Motion.....	6-39
Current Motion Type	6-140
Current Phase/ Control Cam Master	
Offset	6-143
Current PPC-R Temperature	6-39
Current Subroutine.....	6-113
Current X Position	6-109
Current Y Position	6-109
Current Z Position.....	6-109
custom list	
modify.....	7-48
custom list group	
create.....	7-48
D	
DAE02.1	2-15
Date and Time	6-38
DDE Server	7-91
DEA Digital Drive I/O Cards	2-21
DEA04	2-14
DEA04 Digital I/O Pin-out	2-21
DEA08	2-14
Diagnostic Code.....	6-37
Diagnostic Log	7-78
Diagnostic Log List	6-64
Diagnostic Log Options	6-65
Diagnostic Menu	10-30
error screen	10-31
Diagnostic Message	6-36
Diagnostics	
Break point	7-80

Diagnostic log	7-78	ELS System Masters	7-60
Drives.....	7-77	ELSAdj	8-51
Drives on ring	7-77	ELSGrp	8-52
Oscilloscope.....	7-80	ELSMode	8-66
Show program flow.....	7-81	ELSMstr	8-62
System.....	7-76	End of Message.....	12-3
Tasks	7-79	Enhanced I/O mapper.....	7-25
direction		EQU (Equate).....	11-29
negative.....	4-26	Erase All Forcing Masks (RE)	12-37
positive.....	4-26	Erasing (Deleting) a Program (PE).....	12-25
positive/negative	4-26	Error Reaction Mode.....	6-19
Directive		Event	8-67
EVENT/END	11-31	Event Table	10-16
EVENT/START	11-31	Event Table Data	12-21
PLS/INIT.....	11-60	Event Table Data, Row Format	12-22
TASK/END	11-69, 11-70	Event Tables.....	12-21
VAR/INIT	11-71	Events.....	7-54
Directives.....	11-1	events table	
DEFINE	11-23	data structure.....	8-24
EQU (Equate).....	11-29	Executing a Download (PD)	12-24
Download Block Size.....	6-29, 12-23	Executing an Upload (PD)	12-25
Drive I/O Configuration	2-14	Exit to Monitor Prompt.....	6-58
Drive Parameter Editor.....	7-77	Extended Diagnostic	6-37
Drive Parameter Menu.....	10-28		
Drive PLS			
DIAX specifications.....	4-3		
ECODRIVE specifications.....	4-4		
parameters	4-4		
Drive PLS Register.....	6-125		
Drives	7-24, 7-77		
Drives Help Directories	7-95		
Drives on Ring.....	7-77		
E			
Edit labels			
Bit labels	7-13	F	
I/O bit function labels	7-14	F1 program menu	
Register labels	7-11	F4 editing sequencer	10-12
Variable labels	7-10	F2 Table Edit Menu	
Edit menu		Absolute Point Table	10-14
Add event function	7-8	Event Table Menu	10-16
Add subroutine.....	7-8	Floating Table	10-17
Clear current task	7-7	Global Floating Table	10-18
Copy (Ctrl+C)	7-6	Global Integer Table	10-18
Cut.....	7-6	Integer Table Menu	10-17
Delete (Del).....	7-7	Relative Point Table	10-15
Edit labels	7-9	F3 Jog Menu	10-19
Find	7-7	F4 Control Menu	
Find next	7-7	control menu	
Paste (Ctrl+V)	7-6	auto run/hold mode	10-22
Replace.....	7-7	auto step mode	10-23
Select all.....	7-7	manual mode	10-24
Undo (Ctrl+Z)	7-6	F5 Register I/O Menu	10-24
Edit Menu		F6 Parameter Menu	10-25
Labels		F6 Security Menu	10-29
User Labels	7-10	F8 Diagnostic Menu	10-30
edit parameter	7-40	Feedback Data #1	6-150
Electronic Line Shaft.....	6-141	Feedback Data #2	6-150
ELS.....	7-59	Feedback Position	6-135
ELS Groups.....	7-60	Feedback Velocity.....	6-136
ELS System Masters	7-60	Fieldbus Available Cyclic IN	
Virtual Masters.....	7-61	Parameters	6-80
ELS Functions	12-8	Fieldbus Available Cyclic OUT	
ELS Group Master.....	8-64	Parameters	6-80
ELS Groups	7-60, 8-52	Fieldbus Baud Rate (DeviceNet only).....	6-77
ELS Options	6-145	Fieldbus Error Reaction	6-78
ELS Programmed Phase Offset ...	6-141	Fieldbus Mapper	7-25
		Fieldbus Mapper (cyclic channel) From PLC	6-76
		Fieldbus Mapper (cyclic channel) To PLC	6-75
		Fieldbus Mapper (non-cyclic channel)	
		From PLC	6-82
		Fieldbus Mapper (non-cyclic channel)	
		To PLC	6-81
		Fieldbus Multiplex Method.....	6-77

Fieldbus Parameter Channel Length-
77
Fieldbus Slave Device Address 6-76
Fieldbus Slave Firmware Version . 6-79
Fieldbus Word Swap (DeviceNet only)
..... 6-78

File menu
Exit..... 7-3
Export label file..... 7-4
Import label file..... 7-3
New 7-3
Open..... 7-3
Print..... 7-5
Sample programs..... 7-6
Save..... 7-3
Save as 7-3
Find, Find Next, Replace..... 7-7
Finish..... 8-68
Floating Table Menu 10-18
Format of Data Sent to VisualMotion
..... 12-4
Function Classes
K - ELS Functions 12-34
S Class..... 12-34
Function Edit Menu..... 10-13
Function List 10-13

G

Getting Search 7-95
Getting started 7-95
Global Floating Table 10-18
Global Integer Table 10-18
Go 8-68
GPP Flash Compression (PK) 12-28
GPP I/O Configuration Menus 2-3
Edit Menu 2-4
File Menu 2-3
Help Menu 2-8

H

Help
About VisualMotion 7-96
Drives Help Directories..... 7-95
Getting started 7-95
Search..... 7-95
Version change log..... 7-96
Home 8-69
hysteresis 4-25

I

I/O 8-70
I/O Binary Forcing State (RS) 12-38
I/O Configuration 2-1
I/O Configuration List 6-62
I/O Configuration Menus
Settings Menu 2-8
View Menu
Display Register Usage..... 2-6
Error and Status Information .. 2-7
Status Bar..... 2-6
Toolbar 2-6
I/O configuration tool 7-24
I/O Devices..... 2-1
I/O Forcing Selection (RF) 12-37

I/O Forcing State Change (RC) 12-37
I/O Mapper..... 3-1
Binary Shift Register 3-15
Card Selection Setup 3-8
Check rungs and convert to Boolean
strings 3-22
coil 3-12
Latch..... 3-13
normal coil 3-12
One Shot..... 3-12
contacts 3-10
Copy 3-22
Counter 3-17
create new file 3-5
cross reference 3-21
Cut 3-22
Delete 3-22
Delete row 3-21
Forcing icons 3-18
forcing options 3-8
input logic 3-10
Insert row 3-21
Ladder logic icons 3-10
menu selection 3-6
Edit menu 3-7
File menu..... 3-6
Help menu 3-9
Setting menu..... 3-7
View menu 3-7
Window menu 3-9
open default *.iom file 3-5
Forcing 3-18
output logic functions 3-12
Paste 3-22
Properties 3-10
contact selection 3-11
contact setup 3-11
register and bit 3-11
scan time 3-8
specifications 3-2
Boolean strings 3-3
maximum rows 3-4
Rung 3-2
total operations 3-4
Timer 3-16
Undo 3-21
Forcing 3-19
I/O Mapper Executable Size 6-96
I/O Mapper File Size 6-95
I/O Mapper Options 6-94
I/O Mapper Program 6-94
I/O Mapper Total Operations 6-95
I/O Register Access (RB), (RX), (RD)
..... 12-36
I/O RegisterAccess
I/O Register Read 12-36
I/O Register Write..... 12-36
I/O User Configuration List 6-64
icon palette 8-1
Icon Programming..... 8-1
Icons
Accel..... 8-10
Axis..... 8-11
AxisEvt 8-16
Branch..... 8-17
Calc..... 8-19
Cam..... 8-28
CamBuild..... 8-30

Circle.....	8-45	Kinematic #10.....	9-7
Command.....	8-46	Kinematic #12.....	9-8
Connecting Icons	8-3	Kinematic #2.....	9-4
ELSAdj	8-51	Kinematic #3.....	9-4
ELSGrp	8-52	Kinematic #4.....	9-5
ELSMode	8-66	Kinematic #5.....	9-5
ELSMstr.....	8-62	Kinematic #8 with Velocity Precision.....	9-6
Event	8-67	Kinematic #9.....	9-7
Finish	8-68	Normal Case	9-2
Go.....	8-68	Special Case.....	9-2
Home.....	8-69		
I/O	8-70		
Join.....	8-71		
Joint.....	8-71		
Line	8-72	L	
List of control Icons	8-6	Label too long	5-2
Move	8-73	lag time.....	4-25
Msg	8-74	Language Selection	6-17
Param	8-75	Last Active Event Number	6-115
Path	8-77, 8-78	lead time.....	4-24
PLS	8-81	Line	8-72
Position	8-82	Linear Position Units	6-123
Prmbit.....	8-83	Link Ring Control Word	6-41
Prmint.....	8-86	Link Ring MDT Error Counter	6-43
Probe	8-88	Link Ring Primary Fiber Optic Length	6-42
ProbeEvt.....	8-89	Link Ring Secondary Fiber Optic Length.....	6-42
Ratio.....	8-90	List Control Resident Programs (PH)	12-27
Reg	8-91	List of All Parameters6-60, 6-115, 6- 153	
RobotOrg	8-92	List of Event Function Marks (PF)12- 26	
RobotTol	8-92	List of Required Parameters6-60, 6- 115, 6-154	
Scissor.....	8-93	List of SERCOS Devices	6-61
Sequencer.....	8-93	List of SERCOS Drives.....	6-62
Size:	8-95	List of SERCOS I-O Stations.....	6-62
Start.....	8-96	List of Virtual Axes.....	6-63
Stop	8-97	List Variable Labels (PV)	12-31
Sub	8-98	Local RECO02 I/O Configuration (Offline).....	2-10
Veloc.....	8-99	Local RECO02 I/O Configuration (Online).....	2-11
VM.....	8-100	Local RECO02 I/O Modules.....	2-8
Wait.....	8-104	Look Ahead Distance	6-104
Axis at Position.....	8-104		
Axis in Position	8-104		
Coordinated State	8-104		
Indramat/PLC Input Register List ..	6-80	M	
Indramat/PLC Output Register List	6-81		
Initialization Delay	6-32	Master Axis for Ratio Function....	6-128
Initializing a Download (PW).....	12-32	Maximum Acceleration....	6-103, 6-126
Initializing an Upload (PR)	12-30	Maximum Deceleration....	6-103, 6-126
Input/Output Registers	12-35	Maximum Executable Program Size6- 31	
Instruction Format	11-2	Maximum Jog Increment..	6-105, 6-127
Instruction Pointer at Error.....	6-112	Maximum Jog Velocity....	6-105, 6-128
		Maximum Number of Axes Allowed6- 34	
		Maximum Path Jerk	6-106
		Maximum Path Speed	6-103
		Maximum Velocity	6-126
		MDT Multiplex Ident List (DKC 2.3 only).....	6-152
		MDT Multiplex Selection List (DKC 2.3 only).....	6-151
		Menu Map (F1-F4)	10-2
		Menu Map (F5-F8)	10-3

J

Jog Acceleration	6-127
Jog Fine Adjustments	10-21
Jogging	7-88
Join	8-71
Joint	8-71

K

Kinematic Number	6-101
Kinematic Value 1 through T-0-0059 Kinematic Value 10	6-107
Kinematics.....	9-1
Associated Task Parameters	9-2
Kinematic #1	9-3

Mfg. Class 3 Status Word.....	6-139
Minimum SERCOS Cycle Time....	6-32
mode	
lag time.....	4-26
PT.....	4-26
monitoring a PLS status	
PLS Message.....	4-34
Move	8-73
Msg.....	8-74
Multiplexing Parameters.....	6-151

N

Name Text Subclass	12-10
Numeric Data Formats.....	12-4

O

On-Line Data	
CAM indexer.....	7-58
ELS	7-59
Events.....	7-54
Parameters.....	7-38
Points	7-56
Registers.....	7-51
Variables	7-54
Online-Data	
Conveyor Tracker.....	7-71
PID	7-62
Registration	7-67
Sequencer.....	7-72
Zones.....	7-75
Operating Mode.....	6-35
Operators	8-20
Option Card PLS	
parameters	4-5
specifications.....	4-5
Optional Command ID #1	6-148
Optional Command ID #2	6-149
Optional Command ID #3	6-149
Optional Feedback ID #1.....	6-149
Optional Feedback ID #2.....	6-149
Oscilloscope	7-80
Oscilloscope Sample Count.....	6-70
Oscilloscope Sampling Rate.....	6-69
Oscilloscope Signal 1 Axis Number6-68	
Oscilloscope Signal 1 ID Number .	6-67
Oscilloscope Signal 1 List	6-70
Oscilloscope Signal 1 Type	6-66
Oscilloscope Signal 2 Axis Number6-69	
Oscilloscope Signal 2 ID Number .	6-68
Oscilloscope Signal 2 List	6-70
Oscilloscope Signal 2 Type	6-67
Oscilloscope Signal 3 Axis Number6-69	
Oscilloscope Signal 3 ID Number .	6-68
Oscilloscope Signal 3 List	6-70
Oscilloscope Signal 3 Type	6-67
Oscilloscope Trigger Axis or Mask6-72	
Oscilloscope Trigger Control Word6-74	
Oscilloscope Trigger ID Number ..	6-72
Oscilloscope Trigger Level or Mask6-73	
Oscilloscope Trigger Mode	6-73

Oscilloscope Trigger Post-count	6-71
Oscilloscope Trigger Source	6-73
Oscilloscope Trigger Status Word .	6-74
Oscilloscope Trigger Type	6-71
output # data	
hysteresis	4-25
lag time	4-25
lead time	4-24
PT (Time Duration)	4-25
output # switches	
edit a switch's on/off position	4-28
output configuration	
direction	4-26
mode	4-26
output # data	4-24
output # PLS Master	4-27
output # switches	4-28
output index.....	4-27

P

P Data.....	12-16
palette	8-1
Param	8-75
parameter access.....	7-39
Parameter Data Subclass	12-10
Parameter Identification	6-2
parameter list	7-43
Parameter List Block Transfer.....	12-13
Parameter Lists.....	12-12
Parameter Lists Subclasses.....	12-11
Parameter Menu	10-25
parameter overview	
system configuration	7-50
Parameter Transfer Commands	6-4
Parameters	6-1, 7-38
Parameters List	6-5
Path	8-77, 8-78
Path Planner to SERCOS Time Factor	6-31
Peak Load due to I/O	6-41
Peak Load due to Motion	6-40
Pendant End of User Accessible	
Registers	6-50
Pendant Password Timeout	6-50
Pendant Protection Level 1 Password6-48	
Pendant Protection Level 2 Password6-48	
Pendant Security	7-29
Pendant Start of User Accessible	
Registers	6-49
Pendant User Accessible Floats Section	6-48
Pendant User Accessible Integers	
Section	6-49
Phase Correction	8-60
PID	7-62, 12-17
PLC Communication Option.....	6-26
PLS	7-26, 8-81, 12-42
data structure.....	8-26
PLS 1 Build Command	6-84
PLS 1 Start Mask Register	6-83
PLS 1 Start Output Register.....	6-83
PLS ASCII Protocol.....	12-44
PLS Functionality	4-1, 7-26
access PLS data via the Calc icon4-35	

configure a Control PLS	4-7	Probe Configuration Status	6-147
configure a Drive PLS.....	4-12	ProbeEvt.....	8-89
configure an Option Card PLS ..	4-17	Program management	
Drive PLS.....	4-3	Activate.....	7-21
editing PLS configurations.....	4-31	Clear all	7-21
monitoring a PLS status	4-34	Close	7-22
Option Card PLS.....	4-5	Data transfer.....	7-22
saving and downloading PLS configurations	4-30	Delete	7-21
uploading PLS configurations... 	4-31	Download	7-22
PLS Master configuration for a Control PLS		Upload	7-21
ELS Group	4-10	Program Menu	10-11
ELS Master	4-9	Programmed Acceleration.....	6-136
PLS Master configuration for an Option Card PLS		Programmed Ratio Adjust.....	6-141
ELS Group	4-20	Programmed Velocity	6-135
ELS Master	4-20	PT (Time Duration).....	4-25
PLS outputs			
output configuration.....	4-23		
PLS register assignment for a Control PLS		R	
graph limits	4-11	Ratio.....	8-90
mask register	4-11	Ratio Adjust Step Rate	6-143
output register	4-11	Ratio Mode Encoder Type	6-130
PLS register assignment for a Drive PLS		Ratio Mode Options	6-132
graph limits	4-15	Ratio Mode Step Rate	6-131
output register	4-15	Reading Data from VisualMotion ..	12-2
PLS register assignment for an Option Card PLS		RECO02 Bit Access	2-18
graph limits	4-22	16-Bit I/O Modules.....	2-18
mask register	4-22	32-Bit I/O Modules.....	2-18
output register	4-22	RMC02.2 Analog I/O Module ..	2-20
PLS1 Activate Command	6-85	Reference Options	6-124
PLS1 Activate Status.....	6-85	Reg	8-91
PLS1 Build Status	6-84	Register Labels, Bit Labels (RT)..	12-39
PLS1 Error Code	6-85	Register Menu	10-24
PLS1 Extended Error Code	6-86	Register Usage	2-16
PLS1 Hardware ID	6-86	Registers.....	5-1, 7-51
PLS1 Master Encoder List.....	6-93	Reg. 001 - System Control.....	5-3
PLS1 Master Number List	6-92	Reg. 002-005 - Task Control	5-5
PLS1 Master Phase Offset List.....	6-93	Reg. 006 - System Diagnostic Code	5-8
PLS1 Master Type List.....	6-92	Reg. 007-010 - Task Jog Control ..	5-9
PLS1 Output Direction List.....	6-91	Reg. 011-018 - Axis Jog Control ..	5-11
PLS1 Output Hysteresis List	6-91	Reg. 019 - Fieldbus Status	5-12
PLS1 Output Lag Time List	6-89	Reg. 020 - Fieldbus Diagnostics ..	5-14
PLS1 Output Lead Time List.....	6-88	Reg. 021 - System Status	5-15
PLS1 Output Master List.....	6-88	Reg. 022-025 - Task Status	5-16
PLS1 Output Mode List	6-90	Reg. 026 - Fieldbus Resource Monitor	5-17
PLS1 Output One Shot List.....	6-90	Reg. 031-038 - Axis Status	5-19
PLS1 Software ID.....	6-86	Reg. 040 – Link Ring Status	5-21
PLS1 Switch Off List	6-87	Reg. 041 – Link Ring Data 1	5-22
PLS1 Switch On List.....	6-87	Reg. 042 – Link Ring Data 2	5-22
PLS1 Switch Output List.....	6-88	Reg. 050 – Ethernet Status	5-23
Point Table Data, Row Format ...	12-20	Reg. 051 – Standard Message Count	5-23
Point Tables.....	12-18	Reg. 052 – Cyphered Message Count	5-23
Point Table Data	12-19	Reg. 053 – Invalid Protocol Count ..	5-23
Point Table Data, Row Format	12-20	Reg. 088 and 089 - Task A Extend Event Control	5-24
Points.....	7-56	Reg. 090 and 091 - Latch and Unlatch	5-24
Position.....	8-82	Reg. 092-094 - Mask Pendant Key Functionality	5-24
PrmBit	8-83	Reg. 095-097 - BTC06 Teach Pendant Status	5-25
Prmint.....	8-86	Reg. 098 - Pendant Control Task A, B	5-26
Probe	8-88		
Probe 1 Negative Captured Value	6-148		
Probe 1 Positive Captured Value.	6-147		
Probe 2 Negative Captured Value	6-148		
Probe 2 Positive Captured Value	6-148		

Reg. 099 - Pendant Control Task C, D	5-26	Configuration.....	7-93
Reg. 150 and 151 - Virtual Master 1 and 2 Control.....	5-27	Control serial ports	7-94
Reg. 152-159 - ELS Group Control	5-29	Show Program Flow.....	7-81
Reg. 209-232 - Axis Jog Control5-11		Shutdown Command for Flash	
Reg. 241 and 242Virtual Master 1 and 2 Status	5-31	Programming	6-59
Reg. 243-250 - ELS Groups 1- 8 Status	5-32	Size.....	8-95
Reg. 309-332 - Axis Status	5-19	Stack Variable Data.....	6-113
Registration	7-67, 8-39	Start.....	8-96
Registration Block Information (PM)	12-29	Step List	10-12
registration error.....	8-40	Step Table Edit Menu	10-13
Relative Point Used for Origin	6-106	Stop	8-97
Relative Point Used for Tool Frame6-106		Sub	8-98
relative points table		Function arguments	8-98
data structure.....	8-23	Local variables.....	8-99
Relative Table.....	10-15	Subroutine Breakpoint	6-114
Remote Analog Input Drive Cards	2-15	System.....	7-76
Remote I/O Stations	2-13	System Control Parameters	6-17
Configuration	2-13	System Options	6-20
Request Currently Active Program (PA)	12-24	System Timer Value.....	6-38
Request Name of Program (PN)..	12-29		
Required Registers Based on I/O Device	2-17		
Robot Jog Menu	10-19		
RobotOrg.....	8-92		
RobotTol	8-92		
T			
T Label Text.....	12-16		
T-0-0001 Task Motion Type	6-97		
T-0-0002 Task Options	6-98		
T-0-0005 World Position Units....	6-100		
T-0-0010 Kinematic Number.....	6-101		
T-0-0011 Coordinated X Axis	6-101		
T-0-0012 Coordinated Y Axis	6-102		
T-0-0013 Coordinated Z Axis.....	6-102		
T-0-0020 Maximum Path Speed ..	6-103		
T-0-0021 Maximum Acceleration	6-103		
T-0-0022 Maximum Deceleration	6-103		
T-0-0023 Look Ahead Distance...6-104			
T-0-0024 Velocity Override.....6-104			
T-0-0025 Maximum Jog Increment6-105			
T-0-0026 Maximum Jog Velocity 6-105			
T-0-0027 Maximum Path Jerk	6-106		
T-0-0035 Relative Point Used for Origin.....	6-106		
T-0-0036 Relative Point Used for Tool Frame	6-106		
T-0-0050 Kinematic Value 1 through T-0-0059 Kinematic Value 10.6-107			
T-0-0100 Target Point Number....6-107			
T-0-0101 Segment Status.....	6-108		
T-0-0111 Current X Position	6-109		
T-0-0112 Current Y Position	6-109		
T-0-0113 Current Z Position.....	6-109		
T-0-0120 Task Operating Mode...6-110			
T-0-0122 Task Diagnostic Message6-110			
T-0-0123 Task Status Message6-111			
T-0-0130 Current Instruction Pointer6-111			
T-0-0131 Current Instruction	6-111		
T-0-0132 Instruction Pointer at Error6-112			
T-0-0133 Composite Instruction Pointer	6-112		
T-0-0135 Current Subroutine.....6-113			
T-0-0136 Stack Variable Data.....6-113			
T-0-0137 Subroutine Breakpoint .6-114			
T-0-0138 Sequencer Information.6-114			
T-0-2000 List of All Parameters ..6-115			
T-0-2001 List of Required Parameters	6-115		

Target Point Number	6-107	MESSAGE/DIAG.....	11-41
Target Position	6-134	MESSAGE/STATUS.....	11-42
Task Assignment	6-116	MOVE/CIRCLE.....	11-43
Task Diagnostic Message	6-110	MOVE/JOINT.....	11-44
Task Jog Control Registers Coordinated Jogging:	5-9	MOVE/LINE.....	11-45
Task Motion Type	6-97	PARAMETER/BIT.....	11-46
Task Operating Mode	6-110	PARAMETER/GET.....	11-47
Task Options.....	6-98	PARAMETER/INIT.....	11-48
Task Parameter Lists.....	6-115	PARAMETER/SET.....	11-49
Task Parameter Menu.....	10-27	PATH/ABORT.....	11-50
Task Parameters.....	6-12	PATH/POSITION.....	11-51
Coordinated Motion	6-12	PATH/RESUME.....	11-52
Coordinated Motion Status	6-13	PATH/STOP.....	11-52
Robotics	6-12	PATH/WAIT.....	11-53
Task Parameter Lists	6-13	PID/CONFIG.....	11-54
Task Setup.....	6-12	PLC/CLEAR.....	11-56
Task Status	6-13	PLC/READ.....	11-56
T-0-0200 Last Active Event Number.....	6-115	PLC/SET.....	11-57
Task Status Message.....	6-111	PLC/TEST.....	11-58
Tasks.....	7-79	PLC/WAIT.....	11-59
Teach Pendant screens.....	10-1	PLC/WRITE.....	11-59
Teaching Points	10-21	PLS/INIT.....	11-60
Text Language Programming Directives	11-1	REGISTRATION.....	11-62
VisualMotion Textual Instructions Instruction Format	11-2	RETURN	11-63
Textual Language Programming AXIS/EVENT	11-4	ROBOT/ORIGIN.....	11-63
AXIS/HOME	11-5	ROBOT/TOOL.....	11-64
AXIS/INITIALIZE.....	11-6	ROTARY/EVENT.....	11-64
AXIS/MOVE	11-7	SEQ/LIST.....	11-66
AXIS/RATIO	11-8	SEQ/STEP.....	11-67
AXIS/SPINDLE.....	11-9	SEQUENCER.....	11-65
AXIS/START.....	11-10	TASK/AXES	11-68
AXIS/STOP	11-11	TASK/END.....	11-69, 11-70
AXIS/WAIT	11-12	VAR/INIT.....	11-71
CALL	11-13	VIRTUAL/MASTER.....	11-72
CAM/ACTIVATE.....	11-14	Torque Feedback (cyclic).....	6-140
CAM/ADJUST.....	11-15	Torque Mode Commanded Torque	6-139
CAM/BUILD	11-16	Total Program Memory	6-29
CAM/INDEX	11-18	Transfer Cams	7-36
CAM/STATUS	11-19	Events	7-34
CAPTURE/ENABLE.....	11-20	I/O mapper.....	7-33
CAPTURE/SETUP	11-21	Parameters	7-33
DATA/SIZE	11-22	Points	7-35
DEFINE	11-23	Variables.....	7-34
DELAY	11-24	Zones	7-36
ELS/ADJUST.....	11-24	Transfer Tables Between Programs (PX)	12-32
ELS/GROUPM	11-26	Transmitter Fiber Optic Length.....	6-24
ELS/GROUPS.....	11-27	Type of Positioning	6-116
ELS/MASTERS	11-29	U	
ELS/MODE.....	11-28	Unit Number	6-18
EQU (Equate).....	11-29	Units Text Subclass.....	12-10
EVENT/DONE	11-30	Upper Limit, L: Lower Limit Subclasses	12-10
EVENT-END	11-31	User Labels	7-10
EVENT/START	11-31	User Program Header Record.....	12-23
EVENT/TRIGGER	11-31	User Program Variables	12-16
EVENT/WAIT	11-32	User Task Controlled Axis Number for BTC06	6-55
FUNCTION/ARG	11-33	User Task Controlled Menu ID for BTC06	6-53
FUNCTION-END	11-34	User Task Controlled Task ID for BTC06	6-54
FUNCTION/START	11-34	User Watchdog Task ID	6-25
GOSUB	11-35	User Watchdog Timer	6-25
GOTO	11-36		
IF (If-Else-Endif)	11-37		
KINEMATIC	11-39		
LOCAL/VARIABLE	11-40		

V

Variables.....	7-54	Initializing a Download (PW).....	12-32
Veloc	8-99	Initializing an Upload (PR).....	12-30
Velocity Override	6-104	Input/Output Registers.....	12-35
Version change log.....	7-96	List Control Resident Programs (PH)	12-27
View menu		List of Event Function Marks (PF)	12-26
Event functions	7-17	List Variable Labels (PV).....	12-31
Icon captions	7-20	Register Labels, Bit Labels (RT).....	12-39
Icon comments	7-19	Registration Block Information (PM)	12-29
Icon palette.....	7-19	Request Currently Active Program (PA).....	12-24
Subroutines	7-16	Request Name of Program (PN).....	12-29
Task [x]	7-16	Selective Table Transfer Between Programs (PT)	12-30
Zoom out F6.....	7-18	Sequencer/Subroutine Related Subclasses (PI, PL, PQ, PS)	12-33
Virtual Master	5-27	Set Current I/O State with Mask (RM).....	12-38
Virtual Masters.....	7-61	Transfer Tables Between Programs (PX).....	12-32
VisualMotion and Drive Parameters and Subclasses	12-10	User Program Header Record	12-23
VisualMotion menus		VM.....	8-100
Build	7-20	VM Tools	
Commission	7-23	CAM Builder	7-82
Diagnostics.....	7-76	DDE Server.....	7-91
Edit menu	7-6	Jogging	7-88
File menu	7-2	VisualMotion32	7-91
Help.....	7-95		
On-Line Data.....	7-38		
Settings.....	7-92		
View Menu.....	7-15		
VM Tools	7-82		
VisualMotion System Overview.....	1-1		
VisualMotion User's Program			
Communication			
I/O Binary Forcing State (RS). 12-38			
VisualMotion's User Program		Wait.....	8-104
Communication	12-23	World Fast Jog Speed	6-27
Activating a Program (PA).....	12-23	World Large Increment	6-26
CAM Indexer Function Block		World Position Units.....	6-100
Varaiables(PJ)	12-27	World Slow Jog Speed	6-27
Download Block Size.....	12-23	World Small Increment	6-27
Erase All Forcing Masks (RE) 12-37		Writing Data to VisualMotion	12-2
Erasing (Deleting) a Program (PE)			
.....	12-25		
Executing a Download (PD) ...	12-24		
Executing an Upload (PD)	12-25		
GPP Flash Compression (PK). 12-28			
I/O Forcing Selection (RF).....	12-37		
I/O Forcing State Change (RC)12-37			
I/O Register Access (RB), (RX), (RD).....	12-36		

.....	12-27	Wait.....	8-104
.....	12-27	World Fast Jog Speed	6-27
.....	12-26	World Large Increment	6-26
.....	12-100	World Position Units.....	6-100
.....	6-27	World Slow Jog Speed	6-27
.....	6-27	World Small Increment	6-27
.....	12-2	Writing Data to VisualMotion	12-2

W

zone tables	
data structure.....	8-25
Zone Tables.....	12-45
Zones.....	7-75

Z

14 Service & Support

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@indramat.de**

Our service helpdesk in our headquarters in Lohr am Main, Germany can assist you in all kind of inquiries.

Contact us

- by phone: **+49 (0) 9352 40 50 60**
via Service-Call Entry Center Mo-Fr 7:00 a.m. - 6:00 p.m.
- by fax: **+49 (0) 9352 40 49 41**
- by e-mail: **service@indramat.de**

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

+49 (0) 171 333 88 26
+49 (0) 172 660 04 06

Internet - Worldwide Web

Weitere Hinweise zu Service, Reparatur und Training finden Sie im Internet unter

www.indramat.de

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem lokalen Ansprechpartner auf. Die Adressen sind im Anhang aufgeführt.

Additional notes about service, repairs and training are available on the worldwide web at

www.indramat.de

Please contact the sales & service offices in your area first. Refer to the addresses on the following pages.

Vor der Kontaktaufnahme...

...Before Contacting Us

Wir können Ihnen schnell und effizient helfen wenn Sie folgende Informationen bereithalten:

1. detaillierte Beschreibung der Störung und der Umstände.
2. Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern.
3. Tel.-/Faxnummern und e-Mail-Adresse, unter denen Sie für Rückfragen zu erreichen sind.

For quick and efficient help, please have ready the following information:

1. Detailed description of the failure and circumstances.
2. Information on the nameplate of the affected product(s), especially typecode(s) and serial number(s).
3. Your phone/fax numbers and e-mail address, so we can contact you in case of questions.

Kundenbetreuungsstellen Sales & Service Facilities

 Verkaufsniederlassungen
Niederlassungen mit Kundendienst

 sales offices
 offices providing service

Deutschland – Germany

vom Ausland:

from abroad:

(0) nach Landeskennziffer weglassen!!

don't dial (0) after country code!

Vertriebsgebiet Mitte Central Region	SERVICE	SERVICE	SERVICE
Rexroth Indramat GmbH 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 7 am – 6 pm Tel. +49 (0) 9352 40 50 60 service@indramat.de	HOTLINE MO – FR von 17:00 - 07:00 Uhr 5 pm - 7 am + SA / SO Tel.: +49 (0)172 660 04 06 oder / or Tel.: +49 (0)171 333 88 26	ERSATZTEILE / SPARE PARTS verlängerte Ansprechzeit - extended office time - ◆ nur an Werktagen - business days only - ◆ von 07:00 - 18:00 Uhr - 7 am - 6 pm - Tel.: +49 (0) 9352 40 42 22
Vertriebsgebiet Süd Southern Region	Gebiet Südwest Southwest Region	Vertriebsgebiet Ost Eastern Region	Vertriebsgebiet Nord Northern Region
Rexroth Indramat GmbH Ridlerstraße 75 80339 München Tel.: +49 (0)89 540138-30 Fax: +49 (0)89 540138-10 indramat.mue@t-online.de	Mannesmann Rexroth AG Vertrieb Deutschland – VD-BI Geschäftsbereich Rexroth Indramat Regionalzentrum Südwest Ringstrasse 70 / Postfach 1144 70736 Fellbach / 70701 Fellbach Tel.: +49 (0)711 57 61-100 Fax: +49 (0)711 57 61-125	Rexroth Indramat GmbH Beckerstraße 31 09120 Chemnitz Tel.: +49 (0)371 35 55-0 Fax: +49 (0)371 35 55-333	Mannesmann Rexroth AG Vertriebsniederlassung Region Nord Gesch.ber. Rexroth Indramat Walsroder Str. 93 30853 Langenhagen Tel.: +49 (0) 511 72 66 57-0 Fax: +49 (0) 511 72 66 57-93
Vertriebsgebiet West Western Region	Vertriebsgebiet Mitte Central Region	Vertriebsgebiet Ost Eastern Region	Vertriebsgebiet Nord Northern Region
Mannesmann Rexroth AG Vertrieb Deutschland Regionalzentrum West Borsigstrasse 15 40880 Ratingen Tel.: +49 (0)2102 409-0 Fax: +49 (0)2102 409-406	Mannesmann Rexroth AG Gesch.ber. Rexroth Indramat Lilistraße 14-18 63067 Offenbach Tel.: +49 (0) 69 82 00 90-0 Fax: +49 (0) 69 82 00 90-80	Mannesmann Rexroth AG GB Rexroth Indramat GmbH Holzhäuser Str. 122 04299 Leipzig Tel.: +49 (0)341 86 77-0 Fax: +49 (0)341 86 77-219	Rexroth Indramat GmbH Kieler Straße 212 22525 Hamburg Tel.: +49 (0)40 85 31 57-0 Fax: +49 (0)40 85 31 57-15

Europa – Europe

vom Ausland: (0) nach Landeskennziffer weglassen,
from abroad: don't dial (0) after country code,
Italien: 0 nach Landeskennziffer mitwählen
Italy: dial 0 after country code

Österreich – Austria	Österreich – Austria	Belgien – Belgium	Dänemark – Denmark
Mannesmann Rexroth Ges.m.b.H. Gesch.ber. Rexroth Indramat Hägelingasse 3 1140 Wien Tel.: +43 (0)1 9852540-400 Fax: +43 (0)1 9852540-93	Mannesmann Rexroth G.m.b.H. Gesch.ber. Rexroth Indramat Industriepark 18 4061 Pasching Tel.: +43 (0)7221 605-0 Fax: +43 (0)7221 605-21	Mannesmann Rexroth N.V.-S.A. Gesch.ber. Rexroth Indramat Industrielaan 8 1740 Ternat Tel.: +32 (0)2 5830719 Fax: +32 (0)2 5830731 indramat@rexroth.be	BEC AS Zinkvej 6 8900 Randers Tel.: +45 (0)87 11 90 60 Fax: +45 (0)87 11 90 61
Czech Republic	England	Finnland – Finland	Frankreich – France
Mannesmann-Rexroth, spol.s.r.o. Hviezdoslavova 5 627 00 Brno Tel.: +420 (0)5 48 126 358 Fax: +420 (0)5 48 126 112	Mannesmann Rexroth Ltd. Rexroth Indramat Division Broadway Lane, South Cerney Cirencester, Glos GL7 5UH Tel.: +44 (0)1285 863000 Fax: +44 (0)1285 863030	Rexroth Mecman Oy Rexroth Indramat division Ansatie 6 017 40 Vantaa Tel.: +358 (0)9 84 91-11 Fax: +358 (0)9 84 91-13 60	Mannesmann Rexroth S.A. Division Rexroth Indramat Parc des Barbanniers 4, Place du Village 92632 Gennevilliers Cedex Tel.: +33 (0)141 47 54 30 Fax: +33 (0)147 94 69 41 Hotline: +33 (0)608 33 43 28
Frankreich – France	Frankreich – France	Ungarn – Hungary	Italien – Italy
Mannesmann Rexroth S.A. Division Rexroth Indramat 270, Avenue de Lardenne 31100 Toulouse Tel.: +33 (0)5 61 49 95 19 Fax: +33 (0)5 61 31 00 41	Mannesmann Rexroth S.A. Division Rexroth Indramat 91, Bd. Irène Joliot-Curie 69634 Vénissieux – Cedex Tel.: +33 (0)4 78 78 53 65 Fax: +33 (0)4 78 78 53 62	Mannesmann Rexroth Kft. Angol utca 34 1149 Budapest Tel.: +36 (1) 364 00 02 Fax: +36 (1) 383 19 80	Mannesmann Rexroth S.p.A. Divisione Rexroth Indramat Via G. Di Vittorio, 1 20063 Cernusco S/N.MI Tel.: +39 02 2 365 270 Fax: +39 02 700 408 252378
Italien – Italy	Italien – Italy	Italien – Italy	Italien – Italy
Mannesmann Rexroth S.p.A. Divisione Rexroth Indramat Via Borgomanero, 11 10145 Torino Tel.: +39 011 7 50 38 11 Fax: +39 011 7 71 01 90	Mannesmann Rexroth S.p.A. Divisione Rexroth Indramat Via del Progresso, 16 (Zona Ind.) 35020 Padova Tel.: +39 049 8 70 13 70 Fax: +39 049 8 70 13 77	Mannesmann Rexroth S.p.A. Divisione Rexroth Indramat Via Mascia, 1 80053 Castellamare di Stabia NA Tel.: +39 081 8 71 57 00 Fax: +39 081 8 71 68 85	Mannesmann Rexroth S.p.A. Divisione Rexroth Indramat Viale Oriani, 38/A 40137 Bologna Tel.: +39 051 34 14 14 Fax: +39 051 34 14 22
Niederlande – Netherlands	Niederlande – Netherlands	Norwegen – Norway	Polen – Poland
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	Rexroth Hydrocare 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	Rexroth Mecman AS Rexroth Indramat Division Berghagan 1 or: Box 3007 1405 Ski-Langhus 1402 Ski Tel.: +47 (0)64 86 41 00 Fax: +47 (0)64 86 90 62	Mannesmann Rexroth Sp.zo.o. Biuro Poznan ul. Dabrowskiego 81/85 60-529 Poznan Tel.: +48 061 847 67 99 Fax: +48 061 847 64 02
Rumänien – Rumania	Rußland – Russia	Spanien – Spain	Spanien – Spain
Mannesmann Rexroth Sp.zo.o. Str. Drobetei 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	Tschudnenko E.B. Arsenia 22 153000 Ivanovo Tel.: +7 093 223 96 33 oder/or +7 093 223 95 48 Fax: +7 093 223 46 01	Mannesmann Rexroth S.A. División Rexroth Indramat 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	Goimendi S.A. División Rexroth Indramat Parque Empresarial Zuatzu C/ Francisco Montagne 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@adegi.es
Schweden – Sweden	Slowenien – Slovenia	Schweiz - Ost – Switzerland -East	Schweiz - West – Switzerland –West
Rexroth Mecman Svenska AB Rexroth Indramat Division Varuvägen 7 125 81 Stockholm Tel.: +46 (0)8 727 92 00 Fax: +46 (0)8 647 32 77	Rexroth Indramat elektromotorji d.o.o. Otoki 21 64 228 Zelezniki Tel.: +386 64 61 73 32 Fax: +386 64 64 71 50	Mannesmann Rexroth Schweiz AG Gesch.ber. Rexroth Indramat Gewerbestraße 3 8500 Frauenfeld Tel.: +41 (0)52 720 21 00 Fax: +41 (0)52 720 21 11	Mannesmann Rexroth Suisse SA Département Rexroth Indramat Rue du village 1 1020 Renens Tel.: +41 (0)21 632 84 20 Fax: +41 (0)21 632 84 21
Türkei – Turkey			
Mannesmann Rexroth Hidropar A.S. Fevzi Cakmak Cad No. 3 34630 Sefaköy İstanbul Tel.: +90 212 541 60 70 Fax: +90 212 599 34 07			

Afrika, Asien, Australien – inkl. [Pacific Rim]

Africa, Asia, Australia – incl. Pacific Rim

vom Ausland:
from abroad:

(x) nach Landeskennziffer weglassen!
don't dial (x) after country code!

Australien – Australia	Australien – Australia	China	China
AIMS - Australian Industrial Machinery Services Pty. Ltd. Unit 3/45 Horne ST Campbellfield , VIC 3061 Melbourne Tel.: +61 (0)3 93 59 02 28 Fax: +61 (0)3 93 59 02 86	Mannesmann Rexroth Pty. Ltd. No. 7, Endeavour Way Braeside Victoria, 3195 Melbourne Tel.: +61 (0)3 95 80 39 33 Fax: +61 (0)3 95 80 17 33 mel@rexroth.com.au	Shanghai Mannesmann Rexroth Hydraulics & Automation 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	Mannesmann 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
China	China	Hongkong – Hong Kong	Indien – India
Mannesmann 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	Mannesmann Rexroth (China) 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	Rexroth (China) Ltd. 1/F., 19 Cheung Shun Street Cheung Sha Wan, Kowloon, Hongkong Tel.: +852 22 62 51 00 Fax: +852 27 41 33 44	Mannesmann Rexroth (India) Ltd. Rexroth Indramat Division 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
Indien – India	Indonesien – Indonesia	Japan	Japan
Mannesmann Rexroth (India) Ltd. Rexroth Indramat Division 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	PT. Rexroth Wijayakusuma Jl. Raya Bekasi Km 21 Pulogadung Jakarta Timur 13920 Tel.: +62 21 4 61 04 87 +62 21 4 61 04 88 Fax: +62 21 4 60 01 52	Rexroth Automation Co., Ltd. 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	Rexroth Automation Co., Ltd. Rexroth Indramat Division 1F, I.R. Building Nakamachida 4-26-44, Suzuki-ku YOKOHAMA 224-0041, Japan Tel.: +81 (0)45 942 72 10 Fax: +81 (0)45 942 03 41
Korea	Südafrika – South Africa	Taiwan	Thailand
Mannesmann Rexroth-Korea Ltd. Rexroth Indramat Division 1500-12 Dadae-Dong- Saha-Ku Pusan, 604-050 Republic of South Korea Tel.: +82 (0)51 26 00 741 Fax: +82 (0)51 26 00 747 gyhan@rexrothkorea.co.kr	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	Rexroth Uchida Co., Ltd. No.17, Lane 136, Cheng Bei 1 Rd., Yungkang, Tainan Hsien Taiwan, R.O.C. Tel.: +886 (0)6 25 36 565 Fax: +886 (0)6 25 34 754	NC Advance Technologies Co. Ltd. 59/76 Moo 9 Soi Ramintra 34 Ramintra Road, Tharang, Bangkhen Bangkok 10220 Tel.: +66 2 943 70 62 +66 2 943 71 21 Fax: +66 2 509 23 62 sonkawin@hotmail.com

Nordamerika – North America

USA Hauptniederlassung – Headquarters	USA Zentral – Central Region	USA Südwest – Southeast Region	USA SERVICE-HOTLINE
Mannesmann Rexroth Corporation Rexroth Indramat Division 5150 Prairie Stone Parkway Hoffman Estates, IL 60192-3707 Competence Center America Tel.: +1 847 6 45 36 00 Fax: +1 847 6 45 62 01 service@indramat.com	Mannesmann Rexroth Corporation Rexroth Indramat Division Central Region Technical Center Auburn Hills, MI 48326 Tel.: +1 248 3 93 33 30 Fax: +1 248 3 93 29 06	Mannesmann Rexroth Corporation Rexroth Indramat Division Southeastern Technical Center 3625 Swiftwater Park Drive Suwanee, Georgia 30174 Tel.: +1 770 9 32 32 00 Fax: +1 770 9 32 19 03	- 7 days x 24hrs - +1-800-860-1055
USA Nordost – Northeast Region	USA Nordost – Northeast Region	USA West – Western Region Mannesmann Rexroth Corporation Rexroth Indramat Division Northeastern Technical Center 99 Rainbow Road East Granby, Connecticut 06026 Tel.: +1 860 8 44 83 77 Fax: +1 860 8 44 85 95	Mannesmann Rexroth Corporation Rexroth Indramat Division Western Regional Sales Office 7901 Stoneridge Drive, Suite 220 Pleasanton, CA 94588 Tel.: +1 209 815 51 74 Fax: +1 925 227 10 81
Kanada Ost – Canada East	Kanada West – Canada West	Mexiko – Mexico Mannesmann Rexroth Mexico 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

Argentinien – Argentina	Argentinien – Argentina	Brasilien – Brazil	Brasilien – Brazil
Mannesmann Rexroth S.A.I.C. Division Rexroth Indramat Acassuso 48 41/7 RA - 1605 Munro (Buenos Aires) Tel.: +54 (0)11 4756 01 40 Fax: +54 (0)11 4762 6862 mannesmann@mannesmannsaic.com.ar	NAKASE Servicio Tecnico CNC Calle 49, No. 5764/66 RA - 1653 Villa Balester Prov. - Buenos Aires Tel.: +54 (0) 11 4768 36 43 Fax: +54 (0) 11 4768 24 13 nakase@usa.net nakase@infovia.com.ar	Mannesmann Rexroth Automação Ltda. Divisão Rexroth Indramat Rua Georg Rexroth, 609 Vila Padre Anchieta BR - 09951-270 Diadema-SP [Caixa Postal 377] [BR-09901-970 Diadema-SP] Tel.: +55 (0)11 4075 90 60 +55 (0)11 4075 90 70 Fax: +55 (0)11 4075 35 52 awittwer@rexroth.com.br	Mannesmann Rexroth Automação Ltda. Divisão Rexroth Indramat R. Dr.Humberto Pinheiro Vieira, 100 Distrito Industrial BR - 89220-390 Joinville - SC [Caixa Postal 1273] Tel./Fax: +55 (0)47 473 58 33 Mobil: +55 (0)47 974 66 45 prochnow@zaz.com.br



2 8 9 8 7 1

Printed in the U.S.A.

