



VisualMotion 8 (GPP) Multi-Axis Motion Control

Application Manual

Title	VisualMotion 8 (GPP) Multi-Axis Motion Control												
Type of Documentation	Application Manual												
Document Typecode	DOK-VISMOT-VM*-08VRS**-AW02-AE-P												
Internal File Reference	<ul style="list-style-type: none"> • Document Number: 120-2300-B313-02/AE 												
Purpose of Documentation	<p>This documentation describes ...</p> <ul style="list-style-type: none"> • the PPC-R control using GPP firmware with non-coordinated, coordinated and electronic line shafting motion capabilities • VisualMotion Toolkit used for the creation of motion programs and system management • the control of grouped axes by multiple masters for electronic line shafting applications • fieldbus interfaces for Profibus, DeviceNet, ControlNet and Interbus • VisualMotion networking functionality using an Ethernet interface and DDE server application • VisualMotion's Drive Parameter Editor tool with Oscilloscope function • and program debugging and monitoring descriptions 												
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: left; padding: 2px;">Release Date</th><th style="text-align: left; padding: 2px;">Notes</th></tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">01</td><td style="text-align: left; padding: 2px;">05/2001</td><td style="text-align: left; padding: 2px;">Initial release</td></tr> <tr> <td style="text-align: left; padding: 2px;">02</td><td style="text-align: left; padding: 2px;">10/2001</td><td style="text-align: left; padding: 2px;">Updated release</td></tr> <tr> <td style="text-align: left; padding: 2px;"></td><td style="text-align: left; padding: 2px;"></td><td style="text-align: left; padding: 2px;"></td></tr> </tbody> </table>	Description	Release Date	Notes	01	05/2001	Initial release	02	10/2001	Updated release			
Description	Release Date	Notes											
01	05/2001	Initial release											
02	10/2001	Updated release											
Copyright	<p>© 2002 Rexroth Indramat GmbH</p> <p>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).</p>												
Validity	The specified data is for product description purposes only and may not be deemed to be guaranteed unless expressly confirmed in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.												
Published by	<p>Rexroth Indramat GmbH Bgm.-Dr.-Nebel-Str. 2 • 97816 Lohr am Main • Germany Tel.: +49 (0) 93 52/40-0 • Fax: +49 (0) 93 52/40-48 85 • Telex: 68 94 21</p> <p>Bosch Rexroth Corporation • Electric Drives and Controls 5150 Prairie Stone Parkway • Hoffman Estates, IL 60192 • USA Tel.: 847-645-3600 • Fax: 847-645-6201</p> <p>http://www.boschrexroth.de Dept. ESG (DPJ)</p>												
Note	This document has been printed on chlorine-free bleached paper.												

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 Motion Capabilities	2-1
2.1 Non-Coordinated Motion	2-1
2.2 Coordinated Motion.....	2-1
2.3 Electronic Line Shafting (ELS)	2-2
3 VisualMotion Toolkit	3-1
3.1 Introduction.....	3-1
3.2 Installation and Setup.....	3-1
System Requirements.....	3-1
Installing VisualMotion Toolkit 8.0	3-2
Establish Communication using VisualMotion Toolkit	3-8
3.3 Create and Download a Program	3-10
Program Example	3-10
Save, Compile and Download Program.....	3-19
Program Variables	3-21
File Types.....	3-25
3.4 Program Execution.....	3-26
Initial Setup Prior to Operation.....	3-26
Control Registers	3-27
Parameter Mode	3-27
I/O Mapper	3-29
Bit Labels	3-31
3.5 Run VisualMotion Program	3-32
Program Operation	3-35
3.6 Demo Programs	3-36
Linear Motion Application Demos	3-36
Setup.....	3-36
VisualMotion Icon Program – Position Mode (Single Axis)	3-37
VisualMotion Icon Program - Coordinated Mode.....	3-40
4 Multiple Master Overview	4-1
4.1 VisualMotion GPP Overview.....	4-1
4.2 Multiple Master Overview.....	4-1

Virtual Master	4-1
Real Master.....	4-2
ELS Group Master	4-2
ELS System Master	4-2
ELS Group	4-2
Multiple Master Functionality in VisualMotion 8 (GPP)	5-1
5.1 Initialization of VisualMotion's Multiple Master Functionality	5-1
Assigning Registers in VisualMotion.....	5-1
Assigning Program Variables.....	5-3
Assigning Default Labels to Registers and Program Variables	5-3
ELS Group Configuration Word	5-12
ELS Runtime Utility	5-15
Modifying ELS Masters at Runtime	5-16
Modifying ELS Group Masters at Runtime	5-17
Modifying Virtual Masters at Runtime	5-18
5.2 Virtual Master	5-19
Virtual Master Compile Time Initialization.....	5-19
Virtual Master Modes of Operation	5-22
5.3 Electronic Line Shafting (ELS) Master Assignment	5-23
Cascading ELS Groups	5-24
Assigning ELS Master Types.....	5-25
ELS Master Connection Box (System Masters)	5-26
5.4 Electronic Line Shafting (ELS) Group	5-27
ELS Axis Configuration	5-28
Stop and Jog Variables, Compile Time Setup	5-29
Switching Synchronization between Group Input Masters	5-30
Synchronized "Lock On / Lock Off" of ELS Group Master	5-36
Profibus Fieldbus Interface	6-1
6.1 General Information	6-1
PPC-R System Description with a Fieldbus.....	6-1
The VisualMotion Fieldbus Mapper	6-2
Data Transfer Direction (Output vs. Input).....	6-2
Fieldbus Data Channel Descriptions	6-2
6.2 Fieldbus Mapper Functionality	6-6
Initializing the Fieldbus Mapper from VisualMotion 8	6-6
Creating a Fieldbus Mapper File.....	6-6
Fieldbus Slave Definition	6-8
Fieldbus Slave Configuration	6-9
Cyclic Data Configuration	6-10
Additional Functions.....	6-13
6.3 Information for the GPP Programmer	6-15
Register 19 Definition (Fieldbus Status)	6-15
Register 20 Definition (Fieldbus Diagnostics).....	6-16
Register 26 Definition (Fieldbus Resource Monitor)	6-16

Fieldbus Error Reaction	6-17
6.4 Information for the PLC Programmer.....	6-19
*.gsd File	6-19
Multiplexing	6-19
Non-Cyclic Data Access via the Parameter Channel	6-24
7 DeviceNet and ControlNet Fieldbus Interfaces	7-1
7.1 General Information	7-1
PPC-R System Description with a Fieldbus.....	7-1
The VisualMotion Fieldbus Mapper	7-2
Data Transfer Direction (Output vs. Input).....	7-2
Fieldbus Data Channel Descriptions	7-2
7.2 Fieldbus Mapper Functionality	7-7
Initializing the Fieldbus Mapper from VisualMotion 8	7-7
Creating a New Fieldbus Mapper File	7-7
Editing an Existing Fieldbus Mapper File.....	7-7
Fieldbus Slave Definition	7-9
Fieldbus Slave Configuration	7-10
Cyclic Data Configuration	7-11
Additional Functions.....	7-15
7.3 Information for the GPP Programmer	7-17
Register 19 Definition (Fieldbus Status)	7-17
Register 20 Definition (Fieldbus Diagnostics).....	7-18
Register 26 Definition (Fieldbus Resource Monitor).....	7-18
Fieldbus Error Reaction	7-19
7.4 Information for the PLC Programmer.....	7-20
*.eds File	7-20
Word and Byte Swapping	7-20
Multiplexing	7-21
Non-Cyclic Data (Explicit Messaging).....	7-26
8 Interbus Fieldbus Interface	8-1
8.1 General Information	8-1
PPC-R System Description with a Fieldbus.....	8-1
The VisualMotion Fieldbus Mapper	8-2
Data Transfer Direction (Output vs. Input).....	8-2
Fieldbus Data Channel Descriptions	8-2
8.2 Fieldbus Mapper Functionality	8-6
Initializing the Fieldbus Mapper from VisualMotion 8	8-6
Creating a New Fieldbus Mapper File	8-6
Editing an Existing Fieldbus Mapper File.....	8-6
Fieldbus Slave Definition	8-8
Fieldbus Slave Configuration	8-8
Cyclic Data Configuration	8-9
Additional Functions.....	8-12
8.3 Information for the GPP Programmer	8-15

Register 19 Definition (Fieldbus Status)	8-15
Register 20 Definition (Fieldbus Diagnostics).....	8-16
Register 26 Definition (Fieldbus Resource Monitor).....	8-16
Fieldbus Error Reaction	8-17
8.4 Information for the PLC Programmer.....	8-19
Multiplexing	8-19
Non-Cyclic Data Access via the Non-Cyclic (PCP) Channel.....	8-23
9 VisualMotion Networking Functionality	9-1
9.1 Ethernet Interface.....	9-1
Hardware Requirements	9-1
DDE Server Communication.....	9-2
VisualMotion Ethernet Registers	9-2
Ethernet Card Setup	9-3
VisualMotion DDE Server Settings	9-4
Establishing Communication using VisualMotion Toolkit	9-6
9.2 Dynamic Data Exchange.....	9-8
The Dynamic Data Exchange Server	9-8
Dynamic Data Exchange Interface	9-9
9.3 The VM DDE Server	9-10
The Settings Menu.....	9-11
The Monitor Menu.....	9-20
9.4 Modem Communication using TAPI Interface	9-26
Technical Requirements	9-26
Modem Configuration.....	9-27
TAPI Control String	9-28
VisualMotion Modem Error Messages.....	9-29
9.5 SERVER Topic Name	9-30
9.6 DDE Client Interfaces.....	9-32
Creating and Customizing a DDE Client Interface in Microsoft® Excel	9-32
Wonderware.....	9-37
10 Drive Parameter Editor	10-1
10.1 Overview	10-1
10.2 File Menu.....	10-2
Load Default Parameters	10-2
Minimize Parameter Lists.....	10-2
Transfer Parameters	10-3
Scan for SERCOS Drives	10-4
10.3 Configure Menu.....	10-4
Analog Outputs	10-4
Drive Direction.....	10-7
Drive Language Selection.....	10-7
Drive Name	10-8
Drive Monitoring.....	10-8
Drive Tuning.....	10-9

Drive Limits	10-10
Drive Reference	10-11
Encoder 2.....	10-13
Encoder 2 Measuring Wheel Example	10-15
Activating the Measuring Wheel	10-17
Mechanical	10-18
Parameters	10-20
Synchronization	10-30
10.4 Oscilloscope	10-37
File Menu	10-37
Source Menu.....	10-37
Timing	10-38
Signal Selection	10-38
Options Menu.....	10-40
Abort, Upload and Enable Trigger	10-42
Oscilloscope memory buttons.....	10-42
Manipulating trace signals	10-43
Time Controls.....	10-44
11 Programming Concepts	11-1
11.1 Overview	11-1
Programming Environment	11-1
Instruction Execution.....	11-1
11.2 Events	11-1
General Information	11-2
Event Setup.....	11-2
Event Arming/Disarming	11-7
Event Processing	11-7
11.3 Event Types	11-8
Percentage of Coordinated Path from Start / before End Event.....	11-8
Time In Coordinated Path from Start / before End Event.....	11-10
Single Axis Distance from Start / before End Event	11-12
Repeating Timer Event	11-14
Rotary (Repeating Axis Position) Event.....	11-16
Task Input Transition Event	11-18
Feedback Capture	11-20
I/O Register Event.....	11-26
PPC-R X1 Input Event	11-28
11.4 Summary of Event Types.....	11-30
12 Program Debugging and Monitoring	12-1
12.1 Finding Program Problems	12-1
Test Code.....	12-1
12.2 Control Compiler Base Code	12-2
Base Code instruction mnemonics and valid arguments.....	12-2
12.3 Icon Language Warnings and Error Messages.....	12-15

12.4 Text Language Error Messages.....	12-16
First Pass Errors	12-17
Second Pass Compiler Errors.....	12-17
13 Index	13-1
14 Service & Support	14-1
14.1 Helpdesk	14-1
14.2 Service-Hotline.....	14-1
14.3 Internet	14-1
14.4 Vor der Kontaktaufnahme... - Before contacting us.....	14-1
14.5 Kundenbetreuungsstellen - Sales & Service Facilities	14-2

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.

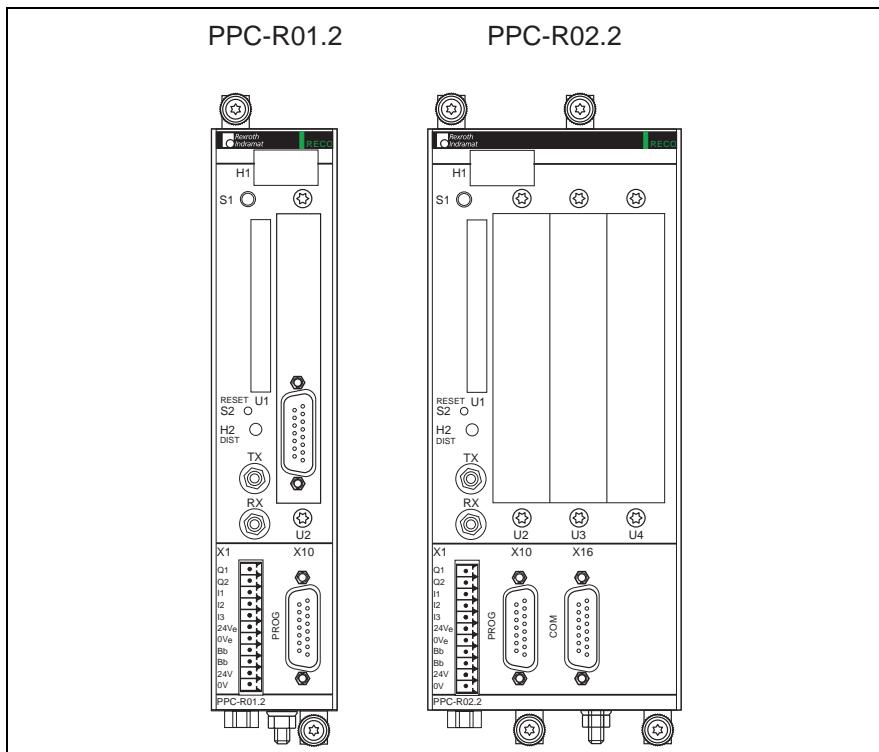


Fig. 1-1: PPC-R Motion Control

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 Motion Capabilities

VisualMotion supports three types of motion:

- Non-Coordinated
- Coordinated
- Electronic line Shafting

2.1 Non-Coordinated Motion

Non-coordinated motion is primarily used to control a single independent axis. There are two modes of non-coordinated motion:

- Single axis
- Velocity mode

Single axis Single axis motion commands within a VisualMotion user program are processed by the control and sent to the digital drive. The user program communicates to the drive the target position (travel distance), the velocity and acceleration. This information is used to develop a velocity profile that is maintained and controlled within the intelligent digital drive. As a result, single axis motion does not require continuous calculation by the control and consumes minimum CPU resources.

Velocity Mode Velocity mode controls the speed of the axis, with no position control loop. Rexroth Indramat's intelligent digital drives maintain torque and velocity loops internally.

A special form of non-coordinated motion called *ratioed axes* permits linking two axes by relating the number of revolutions of a slave axis to a master axis. For example, a ratio might be required when the positioning axis of a gantry robot has a motor on each side of its supporting track.

2.2 Coordinated Motion

The VisualMotion control defines multi-axis coordinated motion in terms of a path composed of standard straight line and circular geometry segments. Point positions, (x, y, z), are used to establish the start, middle or end of a geometry segment. Two points define a line; three points define a circle. The path combines these standard geometry segments so that the start of the next segment begins at the end of the previous segment. A path, therefore, is nothing more than a collection of connected segments.

Since each segment has an end point specifying speed, acceleration, deceleration and jerk, each segment can have a unique rate profile curve. A special type of segment, called a blend segment, can be used to join two standard geometry segments. Blend segments provide the capability of continuous smooth motion from one standard segment to another without stopping. They reduce calculation cycle time as well as provide a means of optimal path shaping.

A VisualMotion system is capable of calculating a path in any of several different modes:

Constant Speed	Constant Speed mode is always active and tries to maintain a constant speed between any two connecting segments in the path. The system and axes acceleration and deceleration limit this mode. Constant speed is the optimum path motion for applying adhesives or paint, welding and some forms of cutting such as laser or water-jet, etc..
Linear Interpolation	Two points define a coordinated motion straight-line segment. The motion is calculated from the end point of the last segment, or the current position if the system is not in motion, to the new end point.
Circular Interpolation	Three points define a coordinated motion circular segment. Circular motion begins with the end point of the last segment executed, or the current system position if the system is not in motion, moves in a circular arc through an intermediate point, and terminates at the specified endpoint.
Kinematics	<p>In addition to the standard x, y, z kinematics, the control has the capability of executing several forward and inverse kinematic movements by using an application-specific library of kinematic functions.</p> <p>Kinematics can be developed to customer specifications. Contact Rexroth Indramat's Application Engineering to inquire about applications which could benefit from kinematics.</p>

2.3 Electronic Line Shafting (ELS)

Electronic Line Shafting is used to synchronize one or more slave axes to a master axis. Using GPS firmware, an ELS master can be a real or virtual axis. GPP firmware introduces multiple master functionality. Refer to *Chapter 4, Multiple Master Overview*. Each slave axis can use either velocity, phase or cam synchronization. ELS has the capability to jog each axis synchronously or independently, and to adjust phase offset and velocity while the program is running.

Velocity synchronization	Velocity synchronization relates slave axes to a master in terms of rotational rate. It is used when axis velocities are most critical, as in paper processing operations in which two or more motors act on a single piece of fragile material.
Phase synchronization	Phase synchronization maintains the same relative position among axes, but adjusts the lead or lag of the slaves to the master in terms of degrees. It is used when the positions of axes are most critical. For example, to achieve proper registration in printing operations, the axis controlling the print head may be programmed for a particular phase offset relative to some locating device, such as a proximity switch.
Cam synchronization	<p>Cam synchronization is used when custom position profiles are needed at a slave axis. A cam profile can be executed either in the control (control cam) or in the drives (drive cams).</p> <p>A cam is an (x, y) table of positions that relate a master axis to a slave. Cams can be stored in the control or in the digital drive. Control cams have more adjustment options and can work with SERCOS drives that do not support the ELS functionality (e.g., SMT or SSE firmware). The same programming commands and utilities are used for both control and drive cams.</p> <p>The number of control cams that can be active at the same time is limited to 4.</p>

3 VisualMotion Toolkit

3.1 Introduction

VisualMotion Toolkit (VMT) is a Windows-based program used for motion control programming, parametrization, system diagnostics and motion control management. With the use of icon driven instructions, motion control programs are created and downloaded to the control for activation. This chapter presents basic installation procedures and instructions on how to create, download, activate and run a VisualMotion program.

3.2 Installation and Setup

VisualMotion Toolkit is supplied on CD-ROM format. VMT is installed with dual language support in English and German. A complete help system is available as part of the installation which, contains detailed information on the use of VisualMotion along with diagnostics and context sensitive help by depressing F1 or using the Help button.

System Requirements

Computer

VisualMotion Toolkit can be installed on any IBM™ PC compatible Pentium computer running...

- Windows98, Windows NT 4 or Windows 2000
- Internet Explorer 4.1 or later
- 16 Mb of RAM system memory
- Complete dual language (*English and German*) installation including help system requires **36.6 MB** of hard disk space. Additional space is required for user files.

Display

A VGA display is required. A color monitor display makes it possible to take full advantage of VMT's graphic interface.

Printer

VMT uses the default printer installed on your computer. For optimal resolution, especially when printing icon programs, use a high-resolution (300-dpi) laser or ink jet printer.

Mouse

A serial or PS2 mouse is required to use the VMT's Icon programming environment.

Serial I/O

VMT can be configured to use the PC's serial port for communication between the host PC and the PPC-R. An IKB0005 RS-232 serial cable is required between the host PC and the PPC-R X10 or X16 communication ports. Hardware handshaking is not used.

Installing VisualMotion Toolkit 8.0

To install VisualMotion Toolkit in a host PC, proceed with the following steps.

1. Insert the VisualMotion CD into the CD-Rom drive. VisualMotion will automatically start.

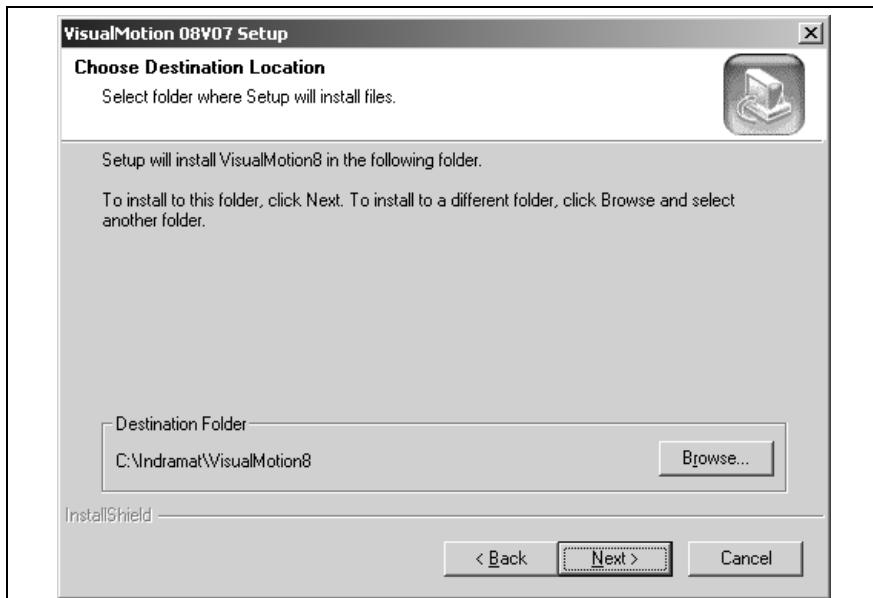
The install program will prompt you to select either the English or German installation language version from the drop down menu and click *OK*. This option can be changed at any time after installation.

Note: After the initial installation, the language can be changed by selecting **Settings** ⇒ **Configuration** from VisualMotion Toolkit's main menu.

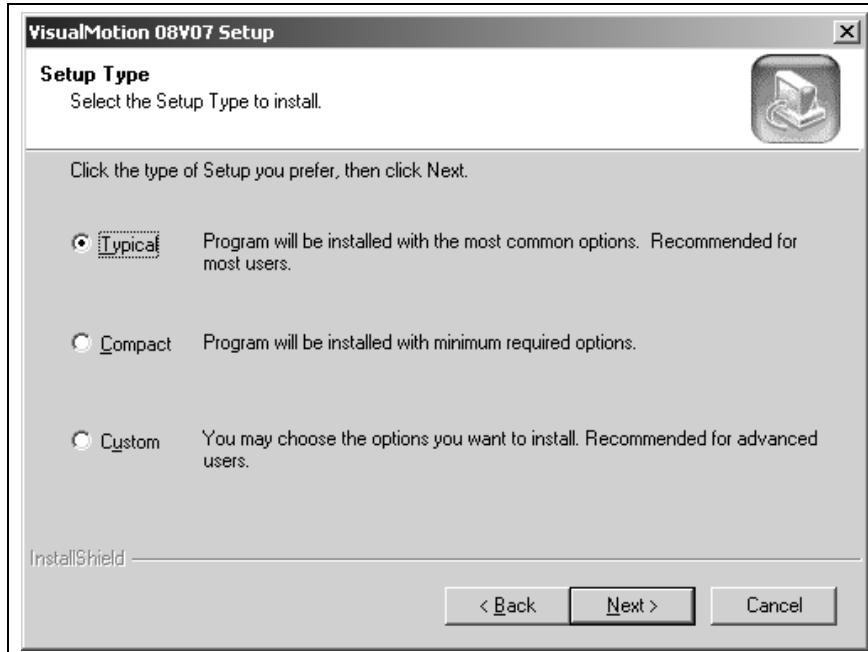


VisualMotion's splash screen will be displayed while an InstallShield® Wizard launches and guides you through the rest of the setup process. Select *Next* to continue with the InstallShield® Wizard.

2. When the *Choose Destination Location* screen appears, decide where to install the VisualMotion program on your system. The default directory is *c:\Indramat\VisusalMotion8*.



3. The *Setup Type* screen allows you to choose from 3 different installation types. The amount of available hard disk space required is dependent upon the setup type selected.



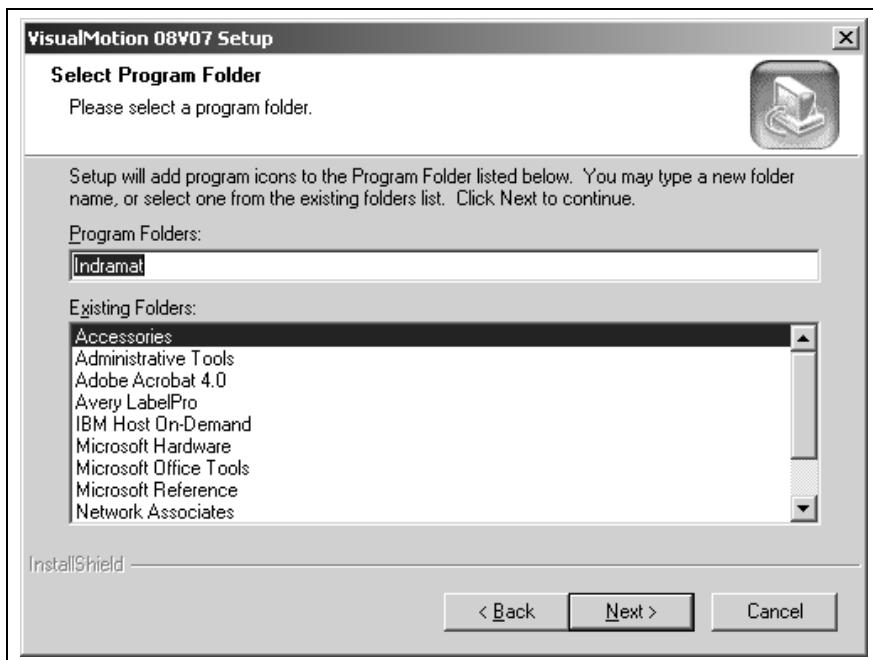
Note: When Custom is selected, the user has the option to choose all or only those components to install.

The following table outlines how much hard disk space is needed per setup type.

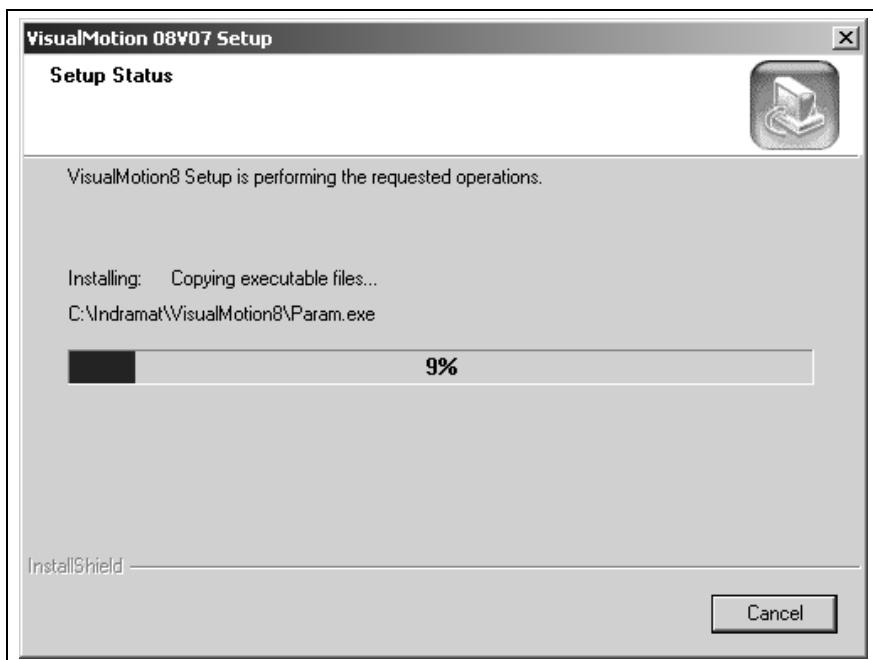
Type of Setup	Description	Required Hard disk Space
Compact	Required files, no help files	7.2 MB
Custom	User-selectable installation	depends on selection, 7.2 – 36.6 MB
Typical (English)	Required files and English help files	36.6 MB
Typical (German)	Required files and German help files	36.6 MB

Table 3-1: Setup Types

4. Select the Program Folder where you would like the program icons to appear. Indramat is the default folder.



5. The Setup Status screen will automatically perform the requested operations.

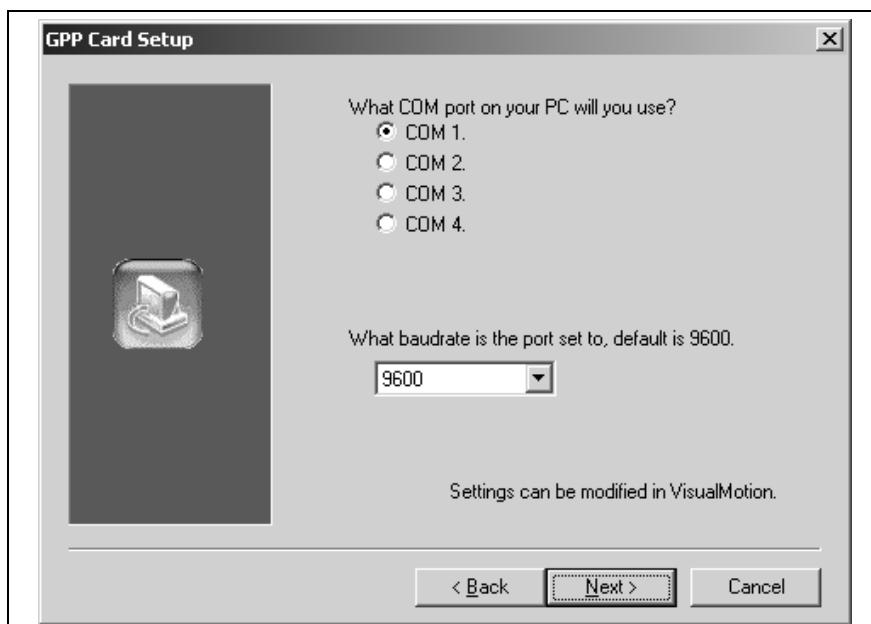


Note: The remaining installation screens vary based on the type of VisualMotion communication selected. The following figures will illustrate these screens along with any additional information necessary to complete the installation.

6. The following three communications links can be used to connect VisualMotion Toolkit with the control.
- Serial port: a communication interface connection between the PC and the controller.
 - AT Modem: an Advanced Technology modem, facilitating connection through a common phone line.
 - Ethernet: an optional pre-configured communication card that is ordered prior to delivery along with the control.

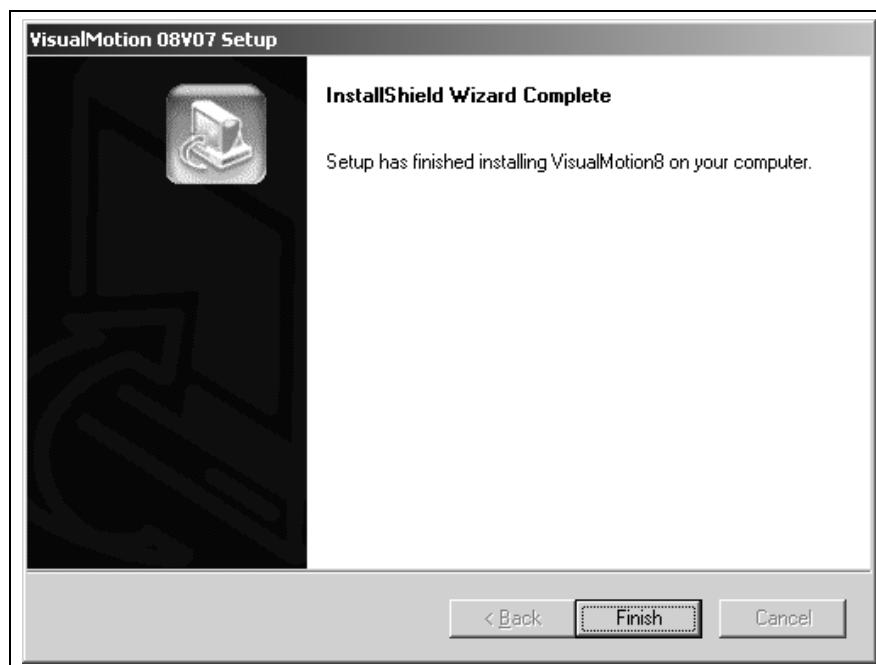


Using a serial port communication link:

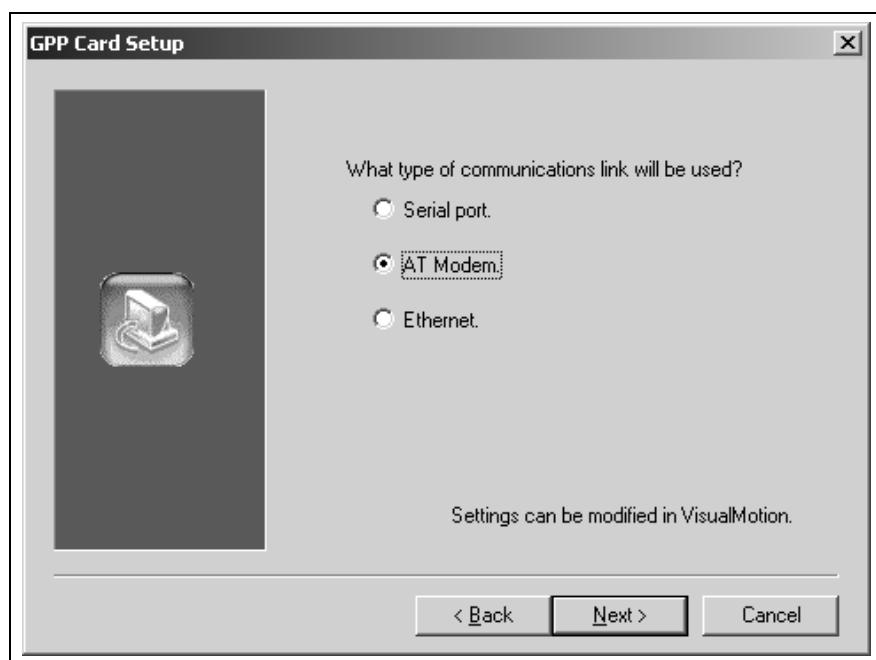


Note: Changes to VisualMotion Toolkit communication can be made from the main menu by selecting **Settings** ⇒ **Card Selection Setup**.

The InstallShield Wizard Complete screen indicates the end of the installation setup.

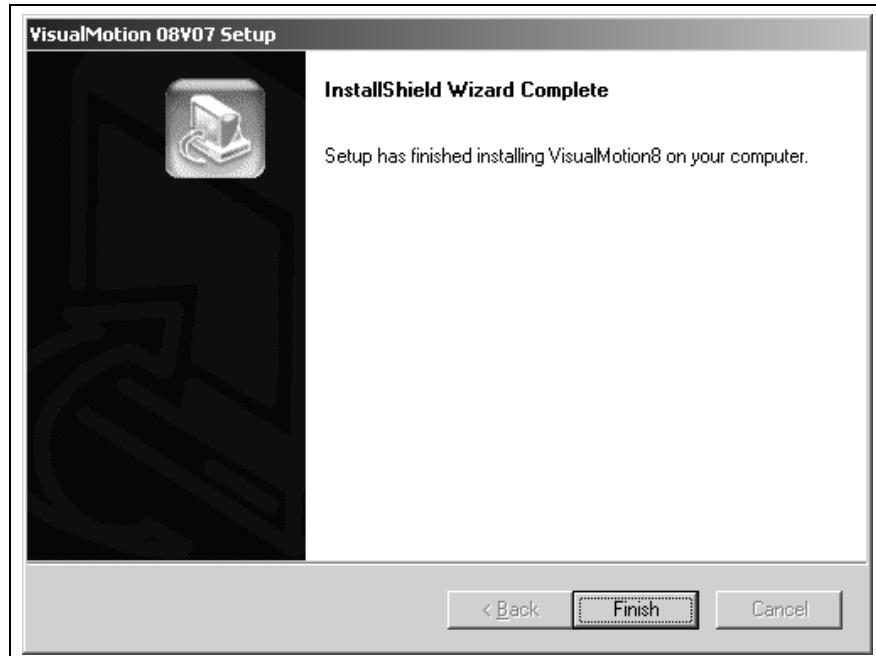


Using an AT Modem communication link:



Once the AT Modem is selected, the VisualMotion Toolkit installation is finished as indicated by the InstallShield Wizard Complete screen.

Likewise, when selecting the Ethernet communication link, the InstallShield Wizard Complete screen appears when the installation has ended.

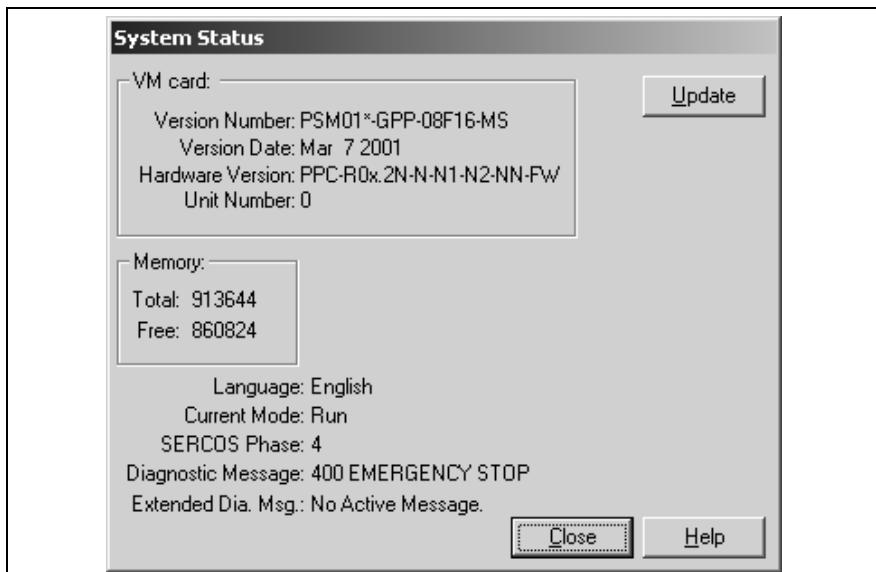


Establish Communication using VisualMotion Toolkit

Once VisualMotion Toolkit (VMT) is installed, verify communications by performing the following procedures.

- To access VMT, begin by selecting **Start** from the Start Menu on the desktop, proceed to \Rightarrow **Programs** \Rightarrow **Indramat** \Rightarrow **VisualMotion8**
- Verify that the correct VisualMotion hardware and settings are correct within **Settings** \Rightarrow **Card Selection Setup**
- Select **Diagnostics** \Rightarrow **System** from the main menu

If proper communication has been established, the System Parameter screen (shown below) will display the VisualMotion version number along with other messages.



If however, the communication link has failed, a window will appear indicating a VisualMotion Server Error as shown in Fig. 3-1. Either there is a problem with the physical connection or the communication settings do not match the settings of the DDE Server.

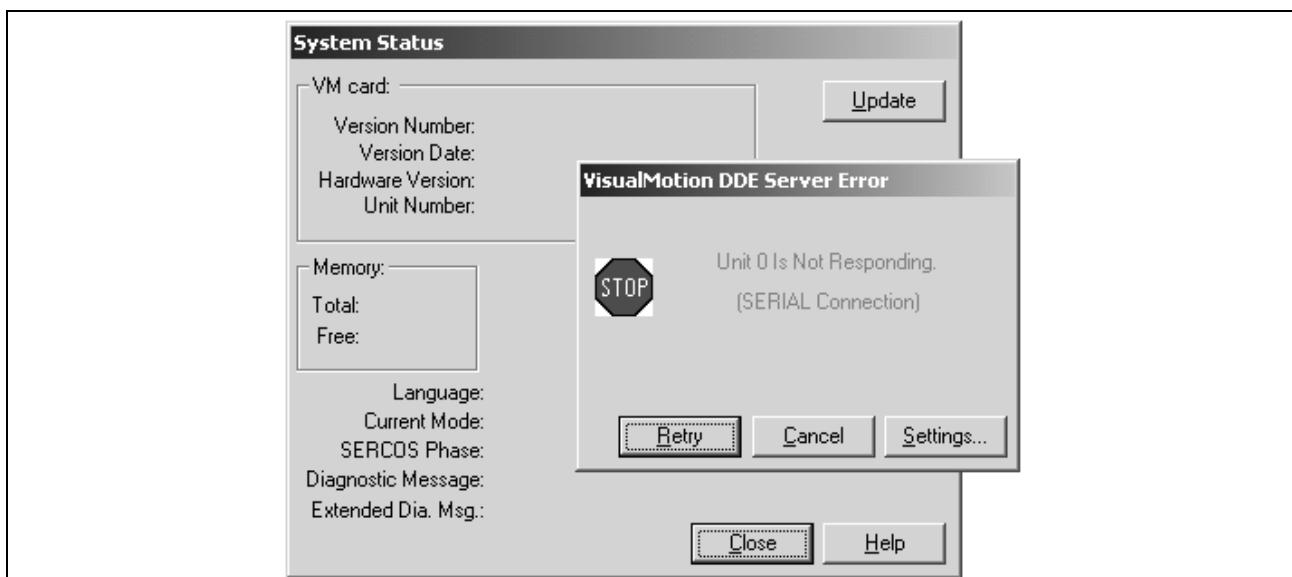


Fig. 3-1: VisualMotion DDE Server Error message

To establish a connection, click on the **Settings** button and the *Serial Communication* window in Fig. 3-2 will open.

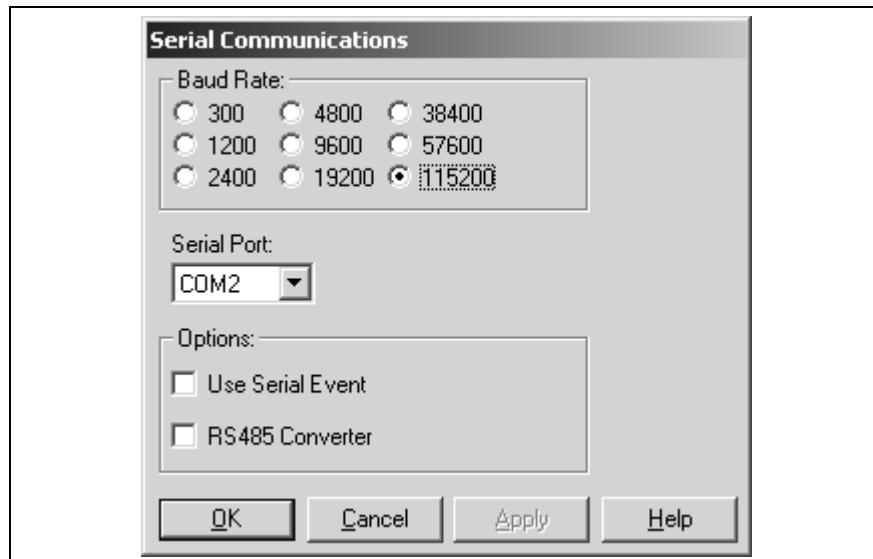


Fig. 3-2: Serial Communication

Match the baud rate setting in the *Serial Communication* window in Fig. 3-2 to that of the control's serial port.

Note: To view the current baud rate settings in the control, depress the S1 button 3 times to display the baud rate of the X10 connector. To display the baud rate of the X16 connector, depress the S1 button 5 times. The S1 button is located in the upper left corner of the control. Read the H1 display for the current baud rate.

Example: X10 = 9600

Once the control's baud rate setting is selected in the *Serial Communication* window, click on the **OK** button.

Now, select **Diagnostics** ⇒ **System** from the main menu and the *System Status* window will display the current status of the control.

3.3 Create and Download a Program

This section will cover the creation, download and activation of a basic single axis program that is not application specific. Refer to the *VisualMotion Reference Manual* for detailed icon descriptions or click on the Help button on any window to open the specific context sensitive help.

Program Example

1. Start VisualMotion Toolkit (**VMT**) by either double clicking on the VMT shortcut icon or by selecting ...

Start* ⇒ *Programs* ⇒ *Indramat* ⇒ *VisualMotion8
2. From VMT's main menu select **File** ⇒ **New** and choose **Icons** as your programming environment. The available target firmware types are as follows:
 - GPP7
 - GPP8



3. Fig. 3-3 shows a series of programming icons that will be used for this program example. As the icons are placed on the screen, specific windows will open and prompt you to enter setup information. A description of the icons along with the setup information required for each window will be provided.



Fig. 3-3: Program Icons

Note: The toolbar programming icons are located above the programming area and are visible regardless of the Icon palette selected from the **View** menu. The icons used in this example are found under **View** ⇒ **Icon Palette 'Single'**.

Toolbar Programming Icons



Start Icon

4. Select and place the **Start** icon from the set of toolbar icons onto the VisualMotion programming area. All VisualMotion *tasks* and *subroutines* must begin with a Start icon.

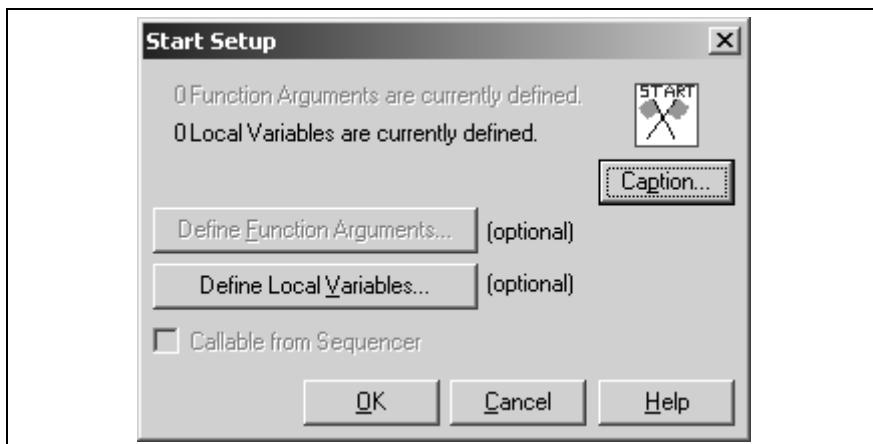
Task

A task is a process that the user runs in his machine. VisualMotion can have up to 4 separate processes or tasks running simultaneously in each program. Tasks A-D run simultaneously and are given equal priority (task A is executed first.)

Subroutine

Subroutines are basically sub-programs that are called by the main program when selected to start. They are used mainly to improve readability as well as to simplify the program.

Note: Once an icon is selected, the cursor will change to a crosshair. Move the crosshair onto the programming area and click to place. To make best use of the programming area, begin placing icons in the upper most left-hand corner of the programming area.

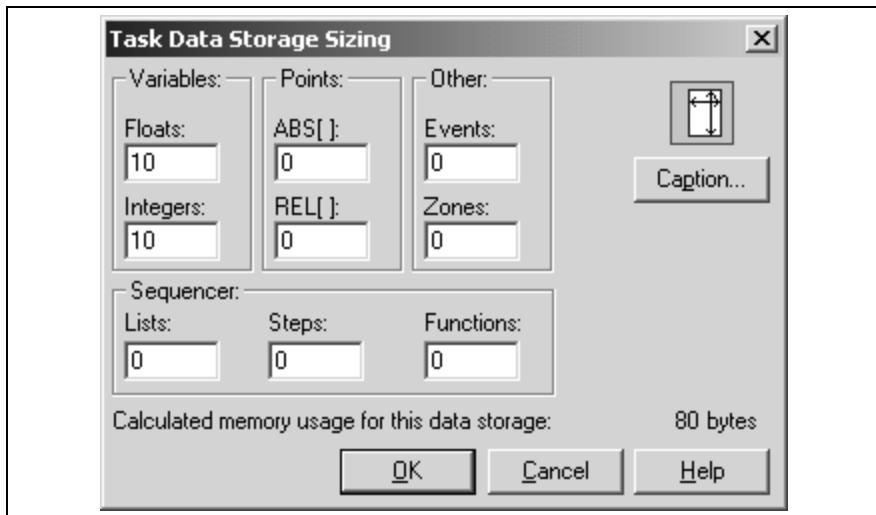


Since this program does not contain any optional function arguments or local variables select OK.

Size Icon

5. Select and place the **Size** icon from the 'Single' Icon Palette to the right of the *Start* icon. This icon determines the number of variables, points, events and zones to be used in the VisualMotion program. It also limits the number of Sequencer Lists, Steps and Functions. Refer to the *VisualMotion Reference Manual* for more information. To maximize memory on the control, place limits on the data storage space.

Note: The default size for both Floats and Integers is 730 and 300 respectively. Reduce this value to 10 for each and click on the OK button.

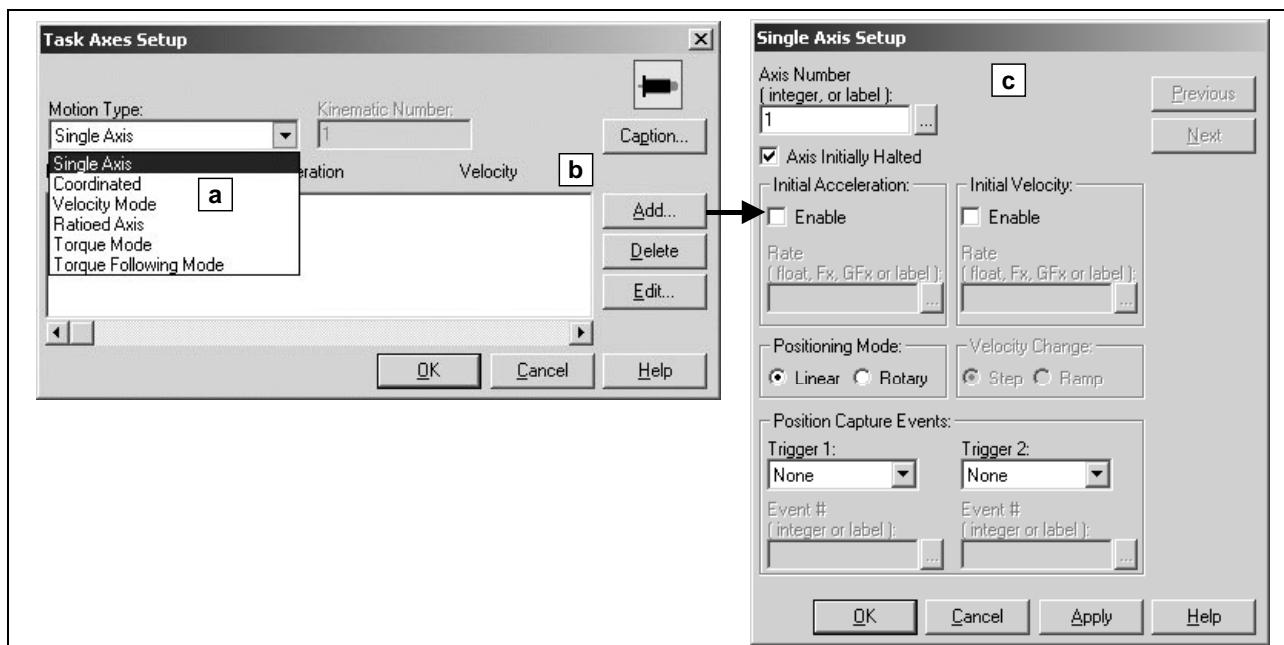
**Axis Icon**

6. Select and place the **Axis** icon to the right of the **Size** icon. This icon configures the primary operation mode for the axis to be used in this Task.

- Select *Single Axis* from the Motion Type drop-down menu box.
- Click on the Add button to setup the axis.
- Configure the axis according to the Single Axis Setup window shown below.

When done, select OK and then Cancel to close the Single Axis Setup window. If Apply is selected, click on the Cancel button.

Note: If Apply and then OK are pressed, a second axis will be displayed in the *Task Axes Setup* window.

**Home Icon**

7. Select and place the **Home** icon to the right of the Axis icon. This icon executes a drive-controlled homing procedure.

Enter a 1 and select OK



Note: Before an axis can be Homed using the *Home* icon, a homing routine must be setup. Refer to the following procedure for details.

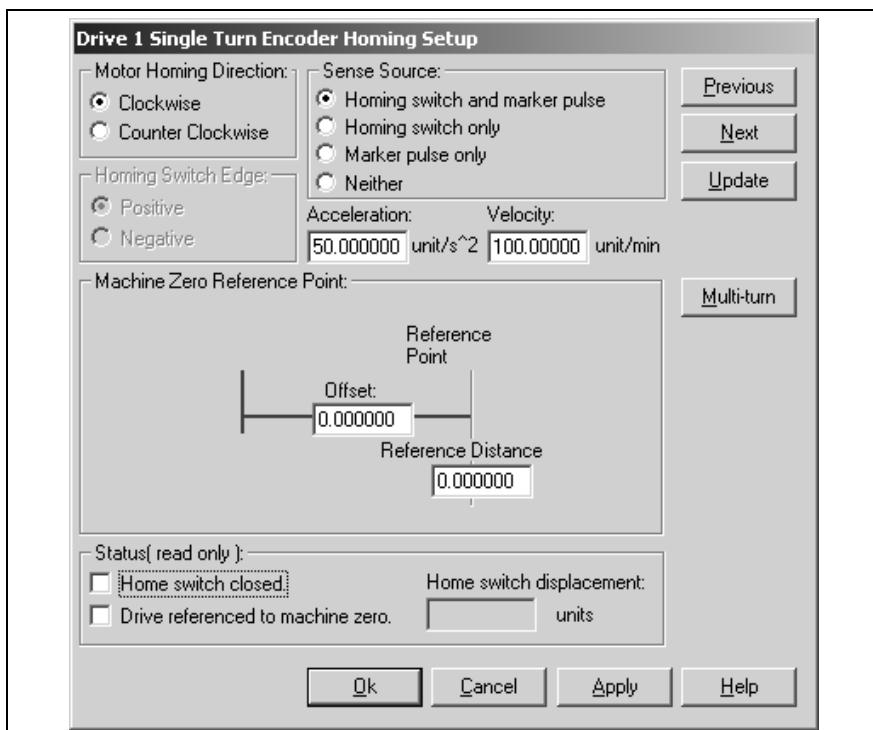
Homing Procedure

Select **Diagnostics** \Rightarrow **Drives** from VMT's main menu to open the *Drive Parameter Editor* window. Now, select **Configure** \Rightarrow **Drive Reference** and VMT will automatically sense the active drive's motor encoder type and launch either the single or multi-turn encoder setup window.

Note: Verify that the motor and drive are properly connected and powered up. Check the serial communication cable between the control and host computer for proper connection.

Drive 1 Single Turn Encoder Homing Setup

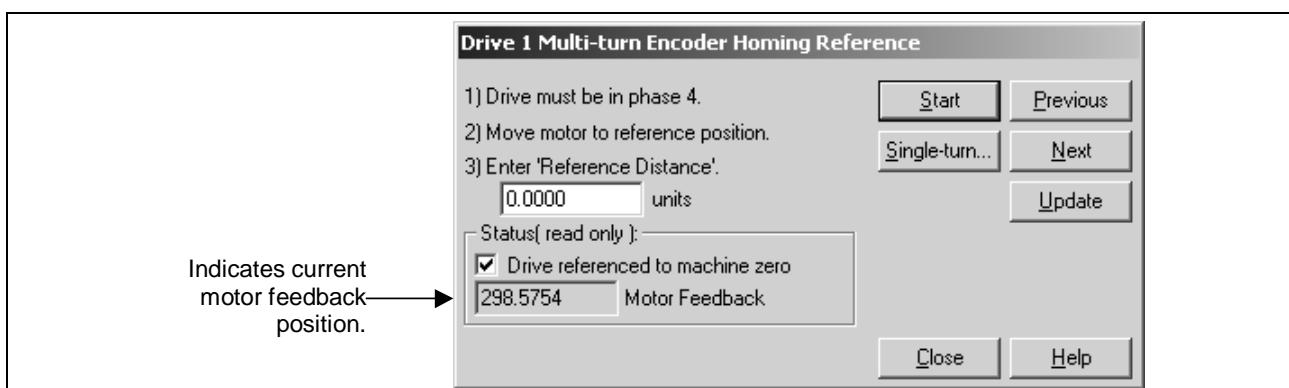
Setup the motor's homing routine according to the window below. The Acceleration and Velocity parameters should be set to a low enough value as to not cause sudden jerk movement. When done, click on **OK** and to complete the homing routine for Drive 1.



The homing procedure is an internal function of Rexroth Indramat's intelligent digital drives and requires only that VisualMotion send a home command to the drive. The actual homing procedure performed by the drive is set in drive parameters.

Drive 1 Multi-turn Encoder Homing Reference

If the motor you are using contains an Absolute encoder, the following window will automatically be displayed when **Configure** \Rightarrow **Drive Reference** is selected from the *Drive Parameter Editor* window.



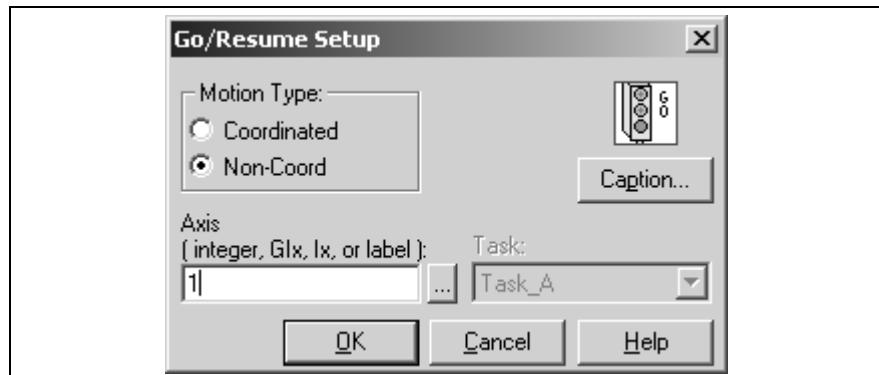
When done, click on **Close** to exit this window. To close the *Drive Parameter Editor* window and return to the Task programming screen, select **File** \Rightarrow **Exit**.

The Homing routine for the Home icon is now complete. Refer to chapter for more information.

- GO Icon** 8. Select and place the **GO** icon to the right of the *Home* icon. This icon enables the axis' drive ready signal (RF).



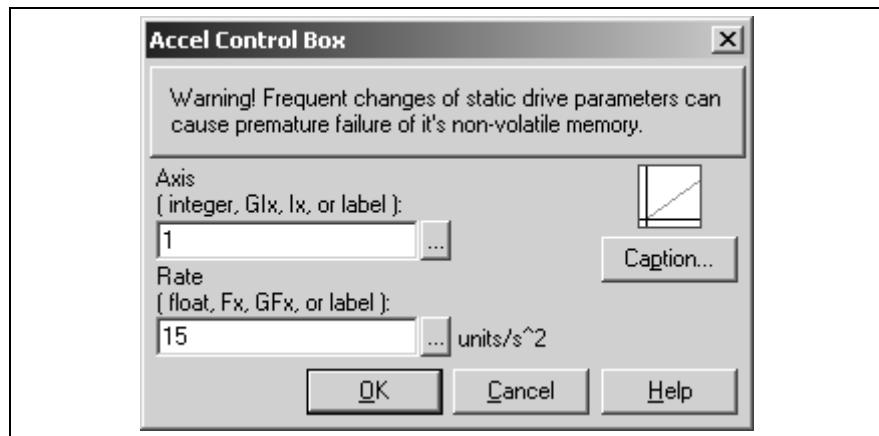
- Enter a **1** in the Axis field to specify which axis to initiate.
- Non-Coord Motion Type should be selected for a single axis.



- Accel Icon** 9. Select and place the **Acceleration** icon to the right of the **GO** icon. This icon sets the acceleration rate that will be used for the specified axis.



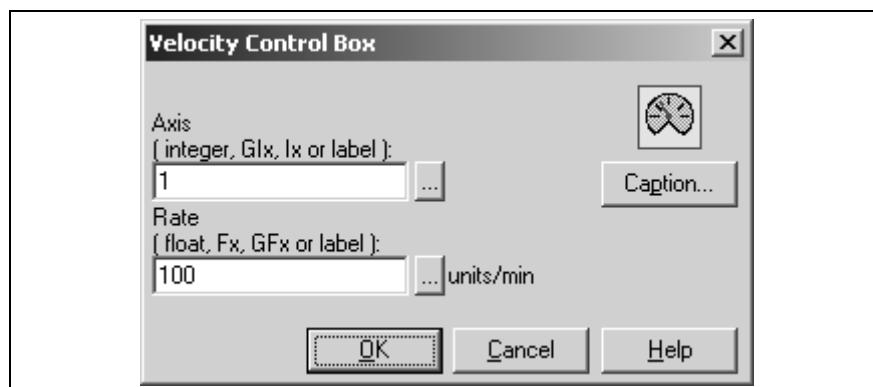
- Enter a **1** to specify the axis.
- Enter **15** for the rate of acceleration and click on **OK**.



Velocity Icon

10. Select and place the **Velocity** icon to the right of the **Accel** icon. This icon sends the velocity rate to the drive that will be used in the move calculation.

- Enter a **1** to specify the axis.
- Enter **100** for the velocity rate click on **OK**.

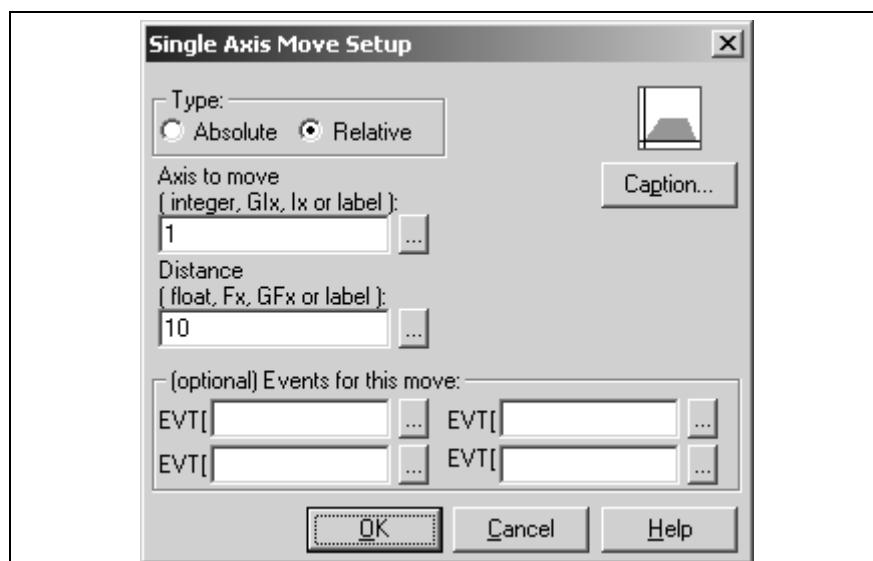
**MOVE Icon**

11. Select and place the **MOVE** icon to the right of the velocity icon. This icon sets the distance that will be traveled by the specified axis.

- Select **Relative** as the move Type.

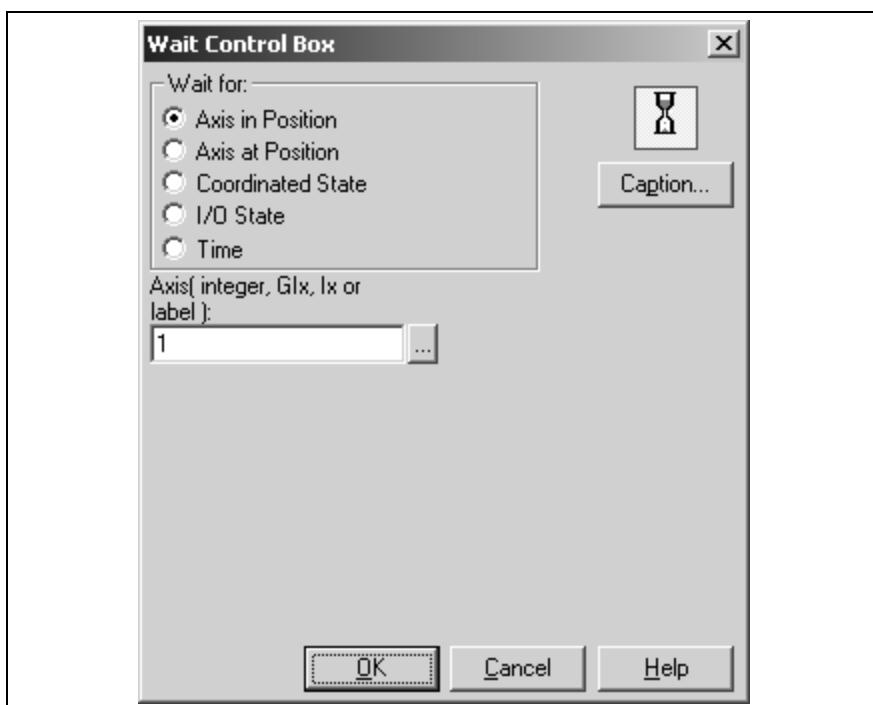
Note: A *Relative* move is an incremental distance that is moved every time the move icon is encountered in the program flow. An *Absolute* move is an exact position that is reached when the move icon is encountered and is not repeated unless the absolute position changes.

- Enter a **1** to specify the axis number.
- Enter a distance of **10** and click on **OK**.

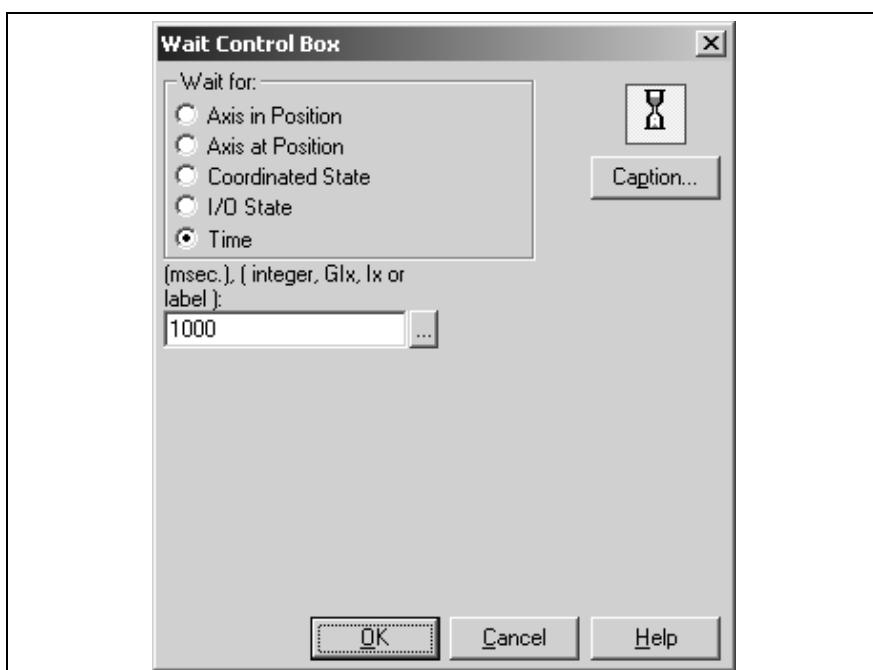


WAIT Icon

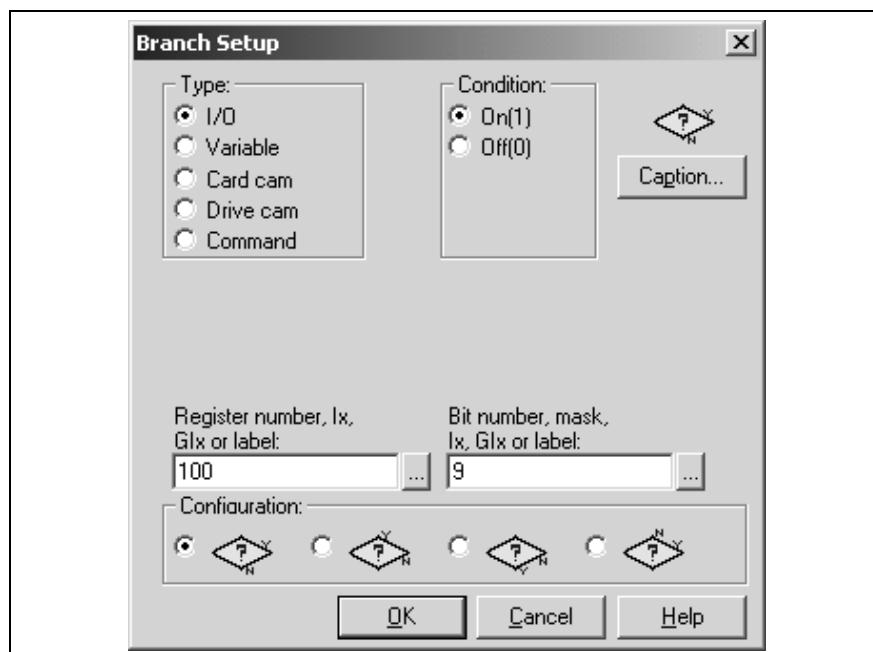
12. Select and place a **WAIT** icon to the right of the **MOVE** icon. The task execution (program flow) will *wait* at this point until the *wait condition* is true. In this case, the task waits until axis 1 is in position.



13. Add a second **WAIT** icon below the first and enter a *Time* of 1000 msec. This icon will introduce a pause of 1 sec to the program before proceeding to the next relative move.

**Branch Icon**

14. Select and place the **Branch** icon to the right of the first **WAIT** icon. This icon re-directs the program flow depending upon a true/false logical value. This creates a loop within the program depending on the value of register 100 bit 9.



Note: The Branch icon will loop back to a specified icon until the branch condition is true. In this example, the *Finish* icon will not be encountered until register 100 bit 9 is on (1.)

Finish Icon



15. Select and place the **Finish** icon to the right of the *Branch* icon. When encountered, the program will end. All *tasks* and *subroutines* must end with the Finish icon.

Line Icon



16. Use the **Line** icon to connect the icons and show program flow. To connect the icons, click once with the left mouse button on the first icon and then click on the next icon in the program flow. A line will join the icons with an arrow indicating program flow.

Note: If an error is made while connecting two program icons, use the **CUT** icon to remove the created connection line.

Afterwards, re-select the line icon to continue.

The completed program should appear as shown in Fig. 3-4.

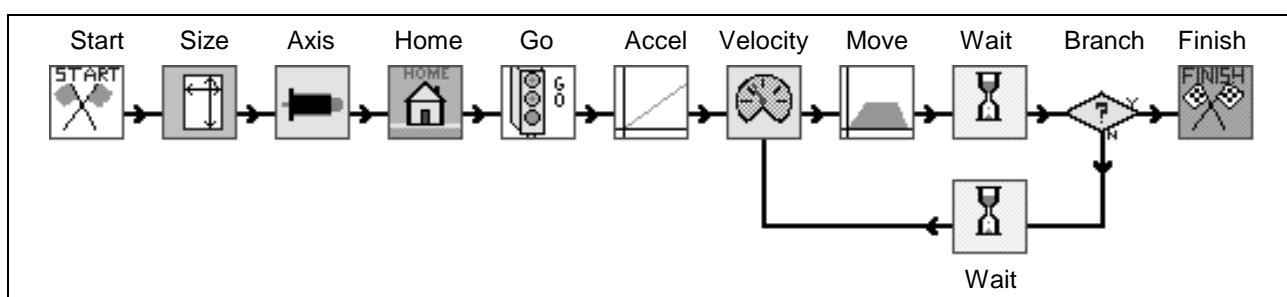


Fig. 3-4: Complete Icon Program

Note: In order to create the loop from the *Branch* icon back to the *Velocity* icon, click on the *Branch* icon first then click on the *Wait* icon. Repeat the step starting with *Wait* icon and finish with the *Velocity* icon.

Save, Compile and Download Program

Save Program

3. To save the program example to a hard drive, select **File** ⇒ **Save As** and enter the filename "sample.str."

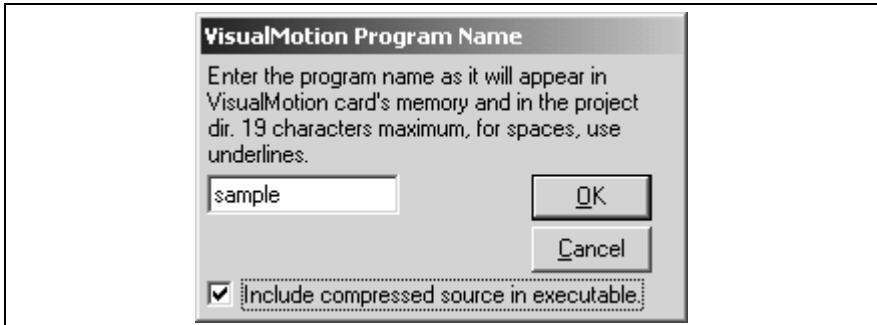
Note: Icon program files are saved with an “*.str” extension at the end of the filename. Refer to Table 3-2 on page 3-26 for other file extensions and their descriptions.

Note: Clicking on the toolbar icon , will automatically cover steps 1-3.

Compile Program

4. Compile the program by selecting **Compile** under the **Build** menu.
- The compiler will first check for a complete path from “Start” to “Finish” for each Task and subroutine.
 - The compiler will then prompt you to enter a name for the program, as it will appear in the control. Use the default name in the text box.

The compiler will then convert the program to code.



When "*Include compressed source in executable*" is selected, the source ***.str** file is compressed and appended to the ***.exc** file when compiled and download to the control. This allows a customer or service person to upload the ***.str** file from the control.

Note: Using this feature can limit the total number of programs that can be concurrently downloaded to the control.



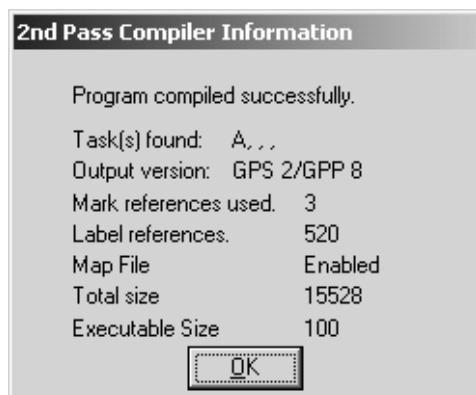
Compressed source program is not password protected.

⇒ Any person knowledgeable about this feature would be able to look at your source program if included.

Retrieving a Compressed Source Program

To retrieve a compressed source program, select **File** ⇒ **Open** and change the **Files of type:** to "*Embedded Icon Files (*.exb*.exc)*". Next, save the program to your hard drive and it will open in VisualMotion Toolkit.

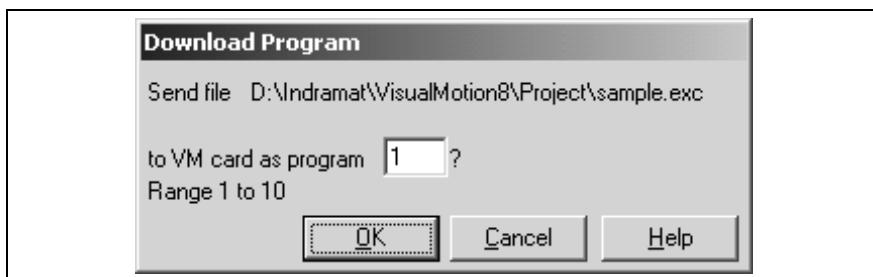
Click on **OK** and the 2nd compiler information window will appear indicating a successful compile and provide information on the compiled program.



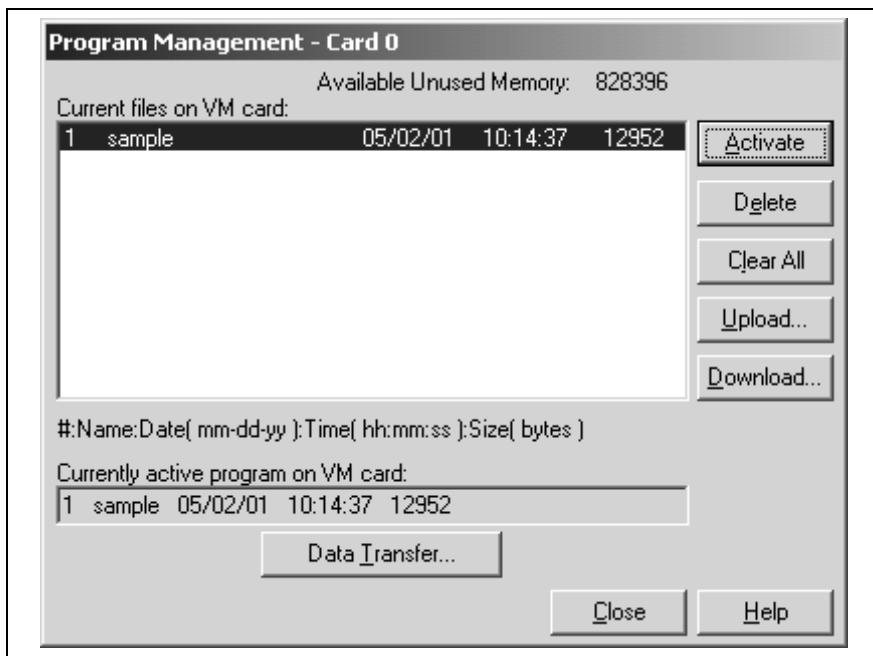
Press OK to close the Compiler Information window.

Download Program to the Control

5. Download the program to the control and activate it.
 - a. Select **Program Management** under the **Build** menu.
 - b. Click on the **Download** button.
 - c. Select the program that you just created and click on **Open**. The program will be listed as "sample.exc" under the *Project* folder.
 - d. In the *Download Program* window, enter a program number from 1 to 10 that will be used to identify the VisualMotion program when downloaded to the control.



6. After the download is complete, the program will be automatically highlighted and active in the control if it's the only program. Otherwise, to activate a different program on the control, simply click to highlight the desired program in the *Current files...* field and click on the **Activate** button.



To close the Program Management window, click on Cancel.

Program Variables

VisualMotion supports floating-point variables, integer variables and constants. There are three types of program variables:

- Global Variables
- Program Variables
- Local Variables

Global Variables

Global variables, designated GF[#] (Global Float) and GI[#] (Global Integer), are stored in the control's RAM and their values are not retained after power is disconnected. There are 256 global floats and 256 global integers and they are shared among the programs stored in the control. They can also be used to exchange values between external components of a VisualMotion system that are capable of accessing the global memory area.

Program Variables

Program variables are designated F[#] and I[#]. The Size icon in VisualMotion determines the number of program variables allocated to a VisualMotion program. Program variables retain their values during power off. The variables can be addressed in a user program by assigning a label to the variable number.

Local Variables

Local or *stack based* variables exist only while in the function (task, subroutine, or event) where they are declared. Local variables are used within a 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 to pass values to a function.

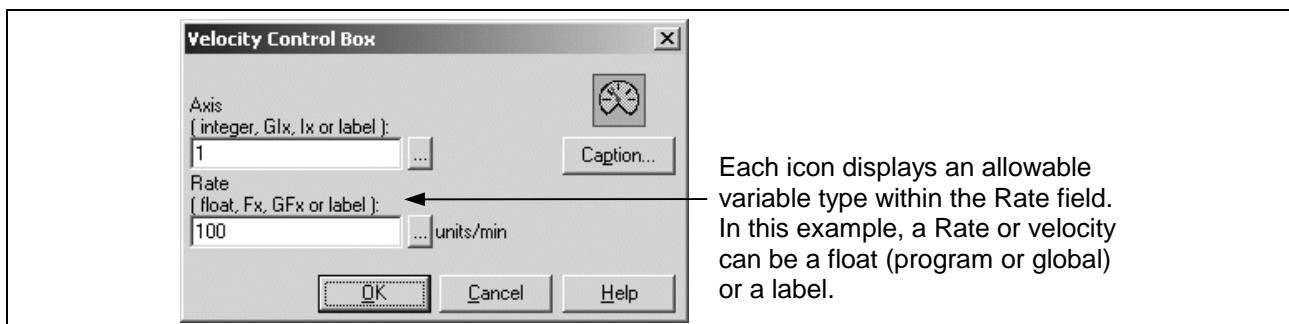
Assigning Labels to Variables

Select **Edit Labels** \Rightarrow **Variable Labels** under the **Edit** menu to assign a label to a variable. A label is simply a name given to a variable, which can help the user identify its function when programming. This can also be done directly within an icon dialog box using the following procedure:

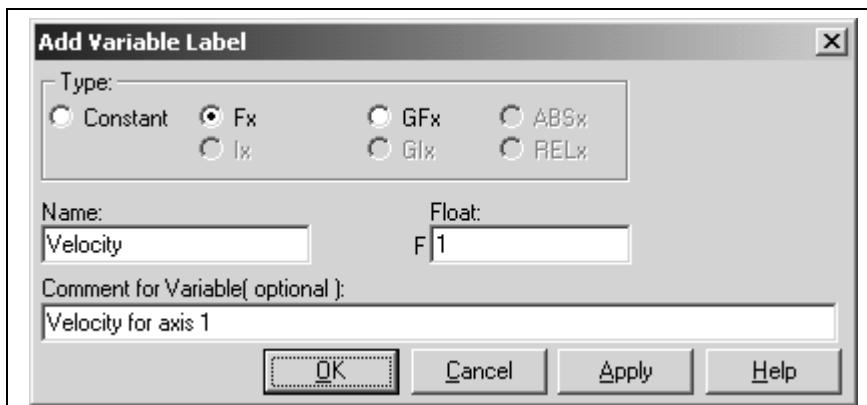
Many programming icons contain a User label button  that opens the User defined Labels window allowing the programmer the ability to create program variables.

Note: Variables can be used to replace numerical values within programming icons. Numerical values within icons that are compiled and downloaded to the control cannot be modified unless changed, re-compiled, downloaded and activated. On the other hand, once compiled and downloaded, variables can be modified within VisualMotion Toolkit by selecting **On-Line Data** \Rightarrow **Variables**. Modified variables are active the next time that specific icon is encountered in the program flow.

Example: Velocity Icon



1. Click on the User label button . The *User Defined Labels* window will open.
2. Click the **Add** button to open the *Add Variable Label* window.

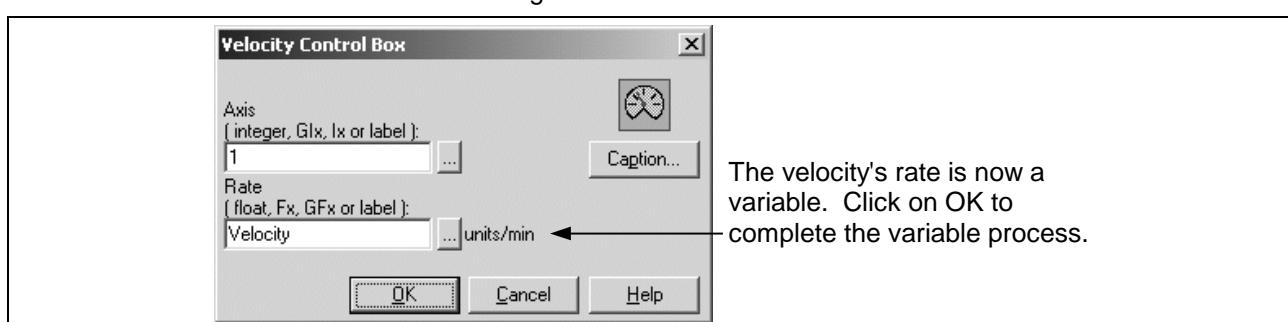


3. Select the **Type** of variable to add. Example: constant, float or integer.
4. Enter a **Name** for the variable.
5. Assign a **1** to the **Float** field.

Note: The Size icon determines the number of available variables in a VisualMotion program.



6. Click **OK** to add the variable and close the *Add Variable Label* window.
7. Highlight the new variable in the *User Defined Labels* window and click **OK**. This will place the new label in the Rate field of the icon's dialog window.



8. Now, use the same procedure to add variables for the **MOVE** icon and the second lower **WAIT** icon.

After all the variables are assigned, Save, Compile and Download the VisualMotion program and activate. To view programmed variables within a VisualMotion program select **On-Line Data** \Rightarrow **Variables**.

Note: When naming a variable, be sure not to use the icon's assigned name in VisualMotion. For example, the **WAIT** icon can be named **WAIT_1** but not **WAIT**.

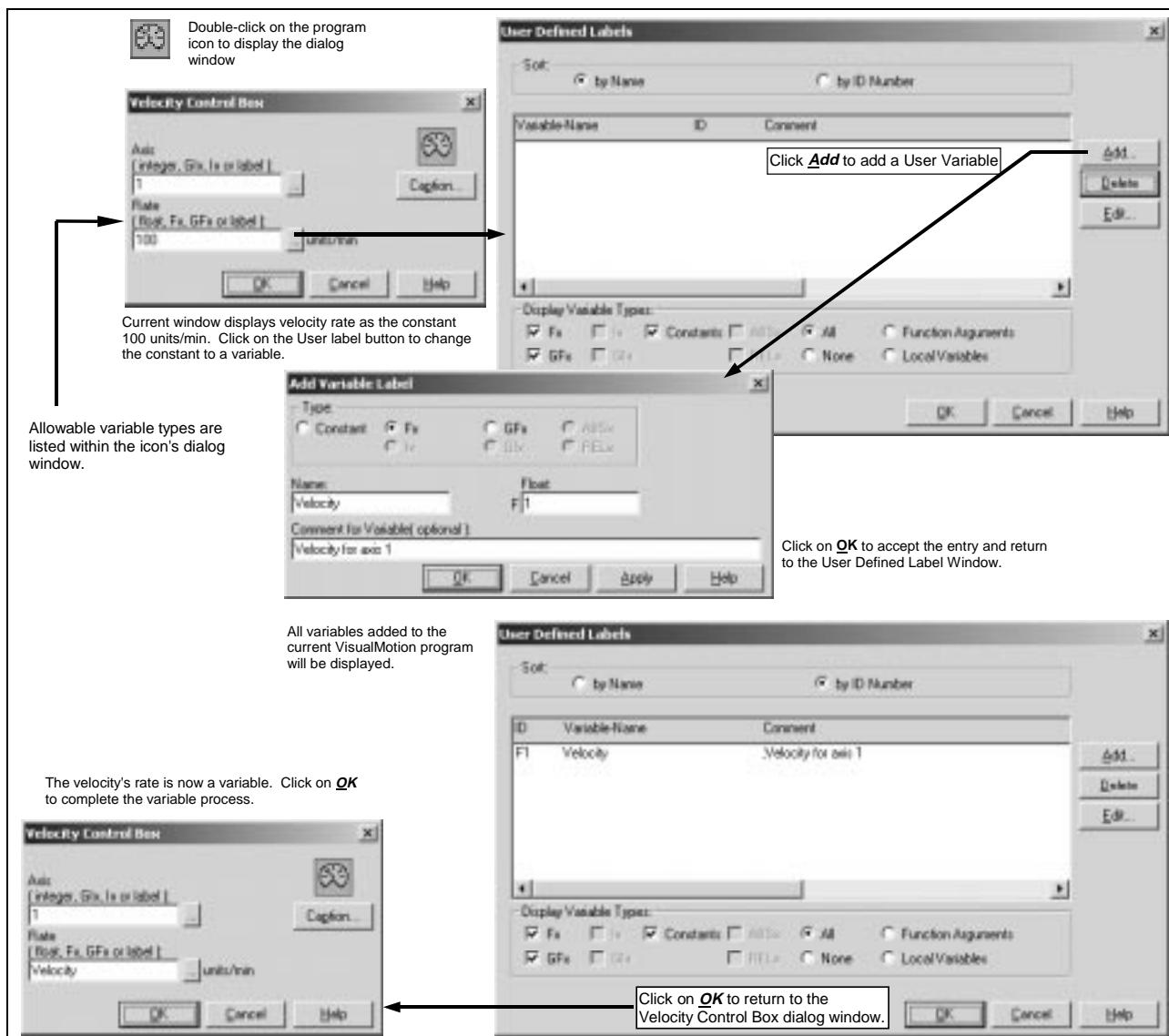


Fig. 3-5: Assigning Variables

Assigning a Value to a Variable

VisualMotion variables are defined by the programmer and are used in programs to enable the user to modify a value in the active VisualMotion program. Modified variables are active the next time the program encounters an icon instruction using that variable. Select **On-Line Data** ⇒ **Variables** to view the Active Program, Variable window.

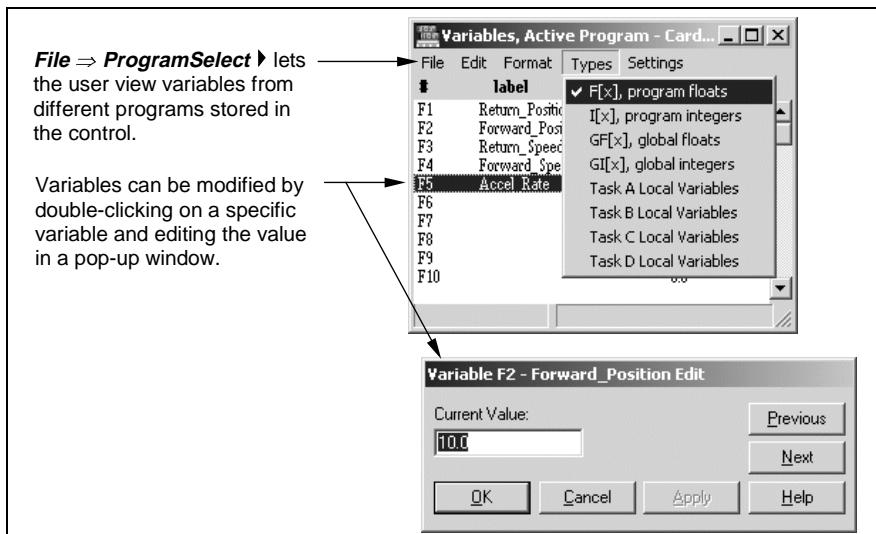


Fig. 3-6: Viewing and Editing Variables

The following variable types are available:

Float Variables (F1-Fx)

A floating-point variable is simply a number containing a decimal point. The number of 32 bit float variables is defined in the sizing icon at the beginning of the program and is stored as part of the program.

Integer Variables (I1-Ix)

Integers are signed or unsigned whole numbers, such as 5 or -3. The number of 32 bit Integer variables is defined in the sizing icon at the beginning of the program and is stored as part of the program.

Global Float and Integer Variables (GF1-Gfx;GI1-Gix)

Global variables are available to all programs stored on the control. Global variables are program independent. Multiple programs can write to the same set of global variables.

Task A-D Local Variables

Local variables are created when a task or subroutine starts and are removed when the task or subroutine execution has ended. Arguments can be passed to local variables to allow multiple applications of a common subroutine.

Event functions

Events are basically interrupt driven subroutines. They can be triggered by a variety of methods, such as transition of an input, repeating timer, position trigger, etc..

File Types

Visual Motion uses a number of different file types. Refer to the following chart to identify the file type according to its extension.

Extension	Description
.acc	Text file that ACAM utility converts to a .csv file
.csv	Comma-Separated-Variable type file used to store cam profiles.
.exb	Compiled program file that is uploaded from the control. It is ready to run and contains program data.
.exc	Compiled program file that is downloaded to and executed by the control.
.iom	I/O Mapper files. Text file consisting of Boolean strings.
.iss	Text files where Visual Motion stores register and bit labels used by the .str file.
.lst	Text file that is referred to for registers and bit labels when the registers on the control are viewed.
.map	File used by the “Show Program Flow” function to trace the flow of the program while it is executing.
.pnt	Absolute Point Table
.pos	Text file that PCAM utility converts to a .csv file
.prm	Parameter file in archived format. These files can be transferred to the control.
.str	Graphical icon program file displayed in VisualMotion Toolkit
.tbl	Text file of points created by the control's “Oscilloscope” function.
.var	Old variable file
.vel	Text file that PCAM utility converts to a .csv file
.vtr	New variable file
.mtn	Text language program source file.
.zon	Zone File

Table 3-2: VisualMotion File Extensions

3.4 Program Execution

This section covers program execution, which includes a brief description of the system and task control registers and the I/O Mapper function. Refer to the *VisualMotion Reference Manual* for a more detailed discussion on any of these topics.

Initial Setup Prior to Operation

The following procedure covers the initial setup required to run the program example that was created in Create and Download a Program on page 3-10. The program example should be opened within VisualMotion Toolkit and activated on the control.

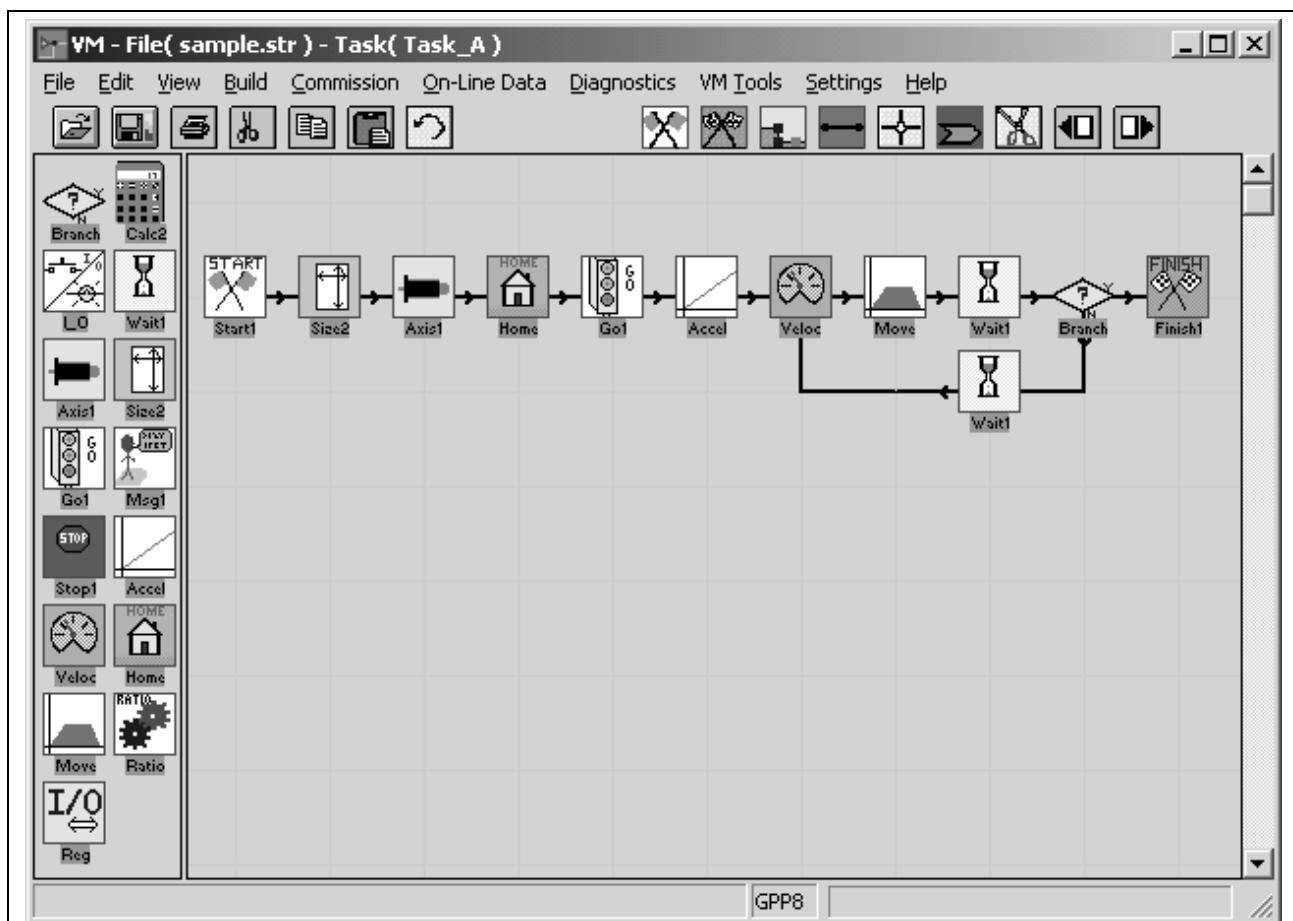
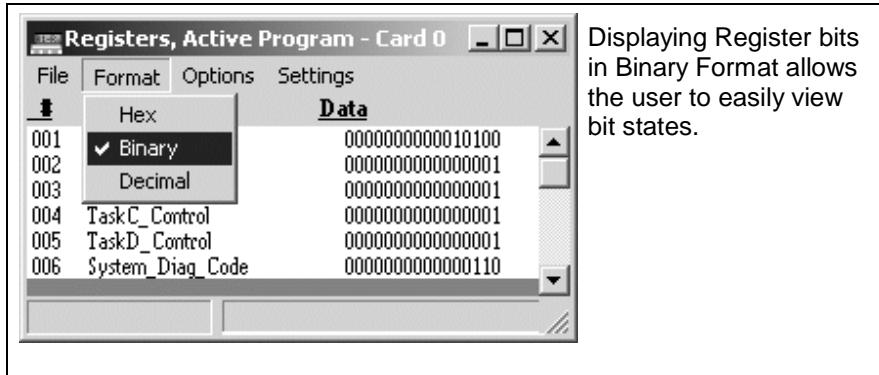


Fig. 3-7: VisualMotion Toolkit Program

Control Registers

Before a VisualMotion program can be activated, key register bits must be set in the System and Task Control registers. Registers 001, *System_Control* and 002, *TaskA_Control* are dedicated as system registers and are used to control program operation. Although the user can modify these registers, we will create an I/O Map using the I/O Mapper function to associate register 100, *User_Inputs_Reg1*, with registers 001 through 005.

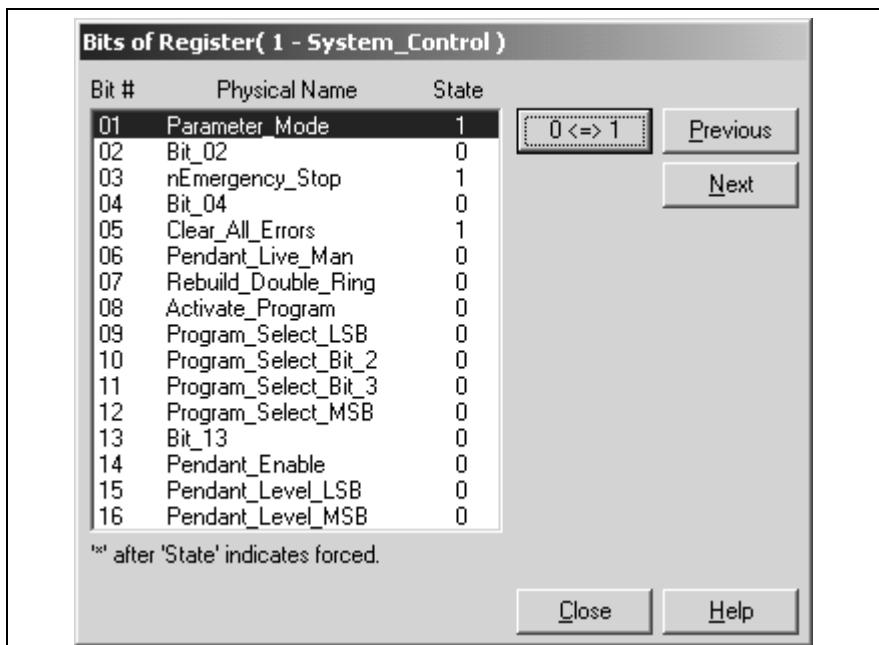
VisualMotion registers can be displayed by selecting ***On-Line Data* ⇒ *Registers*** from VisualMotion Toolkit's (VMT) main menu.



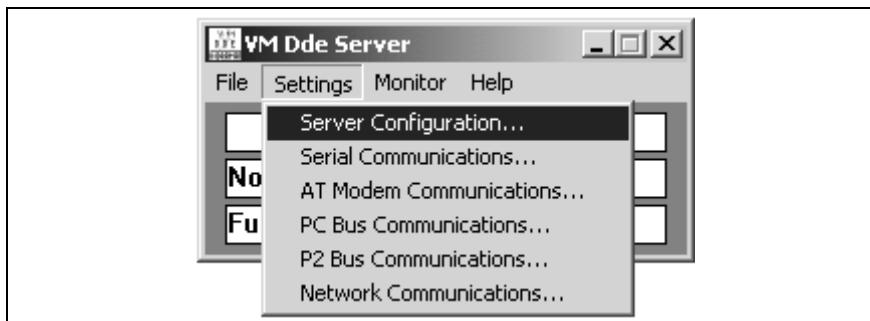
Parameter Mode

Before modifications can be downloaded or updated to the control, the system must be switch to Parameter Mode.

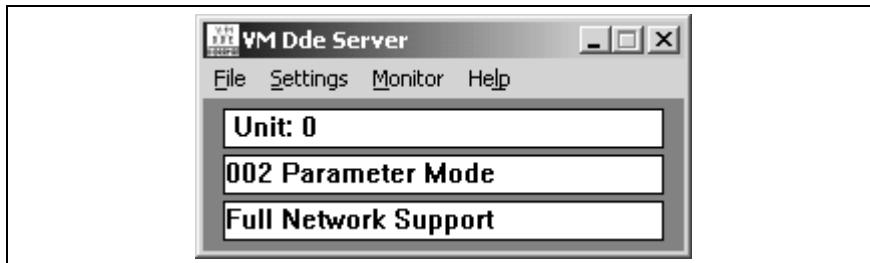
1. Select ***On-Line Data* ⇒ *Registers*** from VMT's main menu.
2. Double-click on *System_Control* register 001.
3. Click and highlight **Bit_01**, *Parameter_Mode* and select the **0 <=> 1** button to change the **State** of **Bit_01** from 0 to 1.



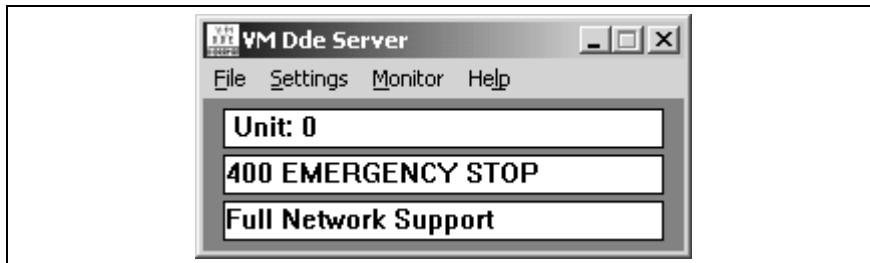
- VM DDE Server**
4. The control and drive should now be in Parameter Mode. To confirm the status of the system, use the VM DDE Server. (Use Alt-Tab to display if already running)
 - a. To display control system status on the DDE Server, select **Settings** → **Server Configuration...** and set Status Display to **SERIAL_0** and Save.



- b. If Parameter Mode was successful, the DDE Server will display...



- c. Otherwise, the display will read...



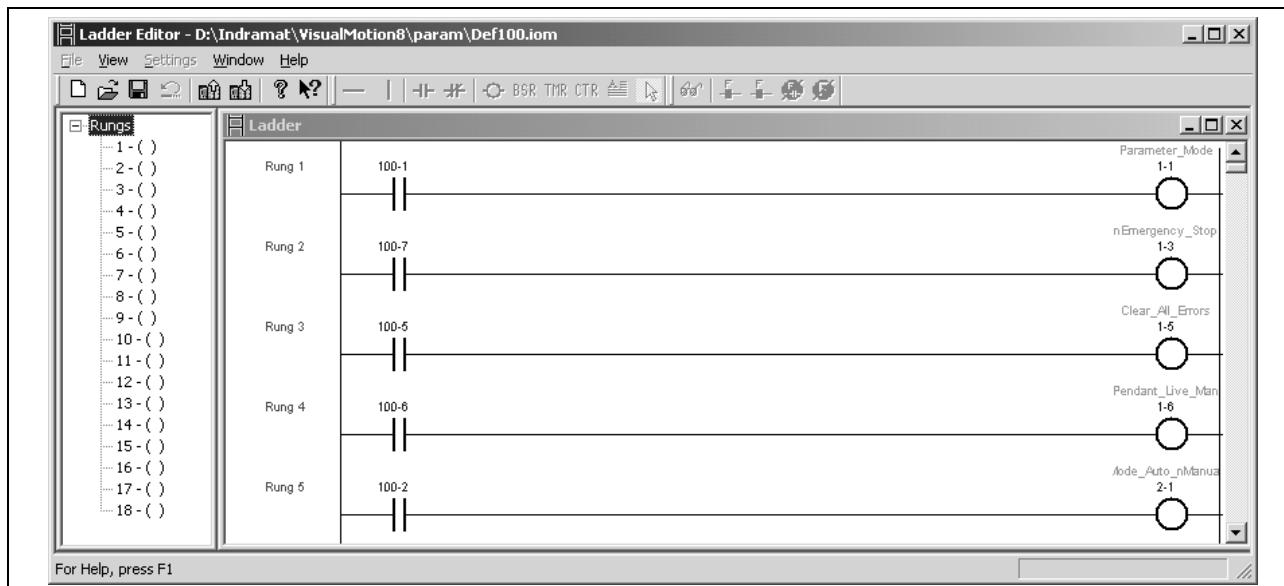
Note: If the display status of the DDE Server does not change to Parameter Mode, then register 001 bit 01 is already mapped to another register. If this case, use the following information on the I/O Mapper to correct the situation.

I/O Mapper

The I/O Mapper is a user programmed PLC logic task that automatically runs once the control has successfully completed its power-up sequence. The I/O Mapper is displayed by selecting **Commission** \Rightarrow **I/O Mapper**. VisualMotion's I/O Mapper task allows manipulation of I/O register bits that can be programmed using Boolean strings or a ladder logic interface.

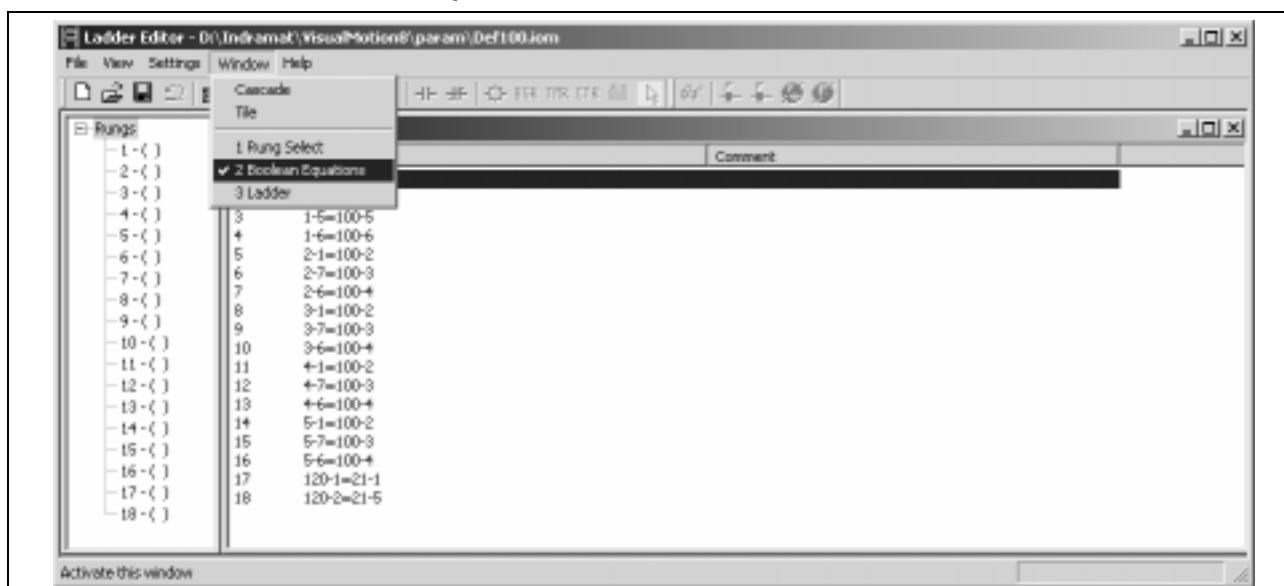
Uploading I/O Mapper from Control

Select **File** \Rightarrow **Get Ladder from control** to view existing I/O Mapper strings currently in the control.



Note: If the Ladder window is blank, then the control does not contain an I/O Mapper program. Refer to Downloading the Default I/O Mapper on page 3-31.

To view the I/O Mapper in Boolean equation form, select **Boolean Equations** from the **Windows** menu.



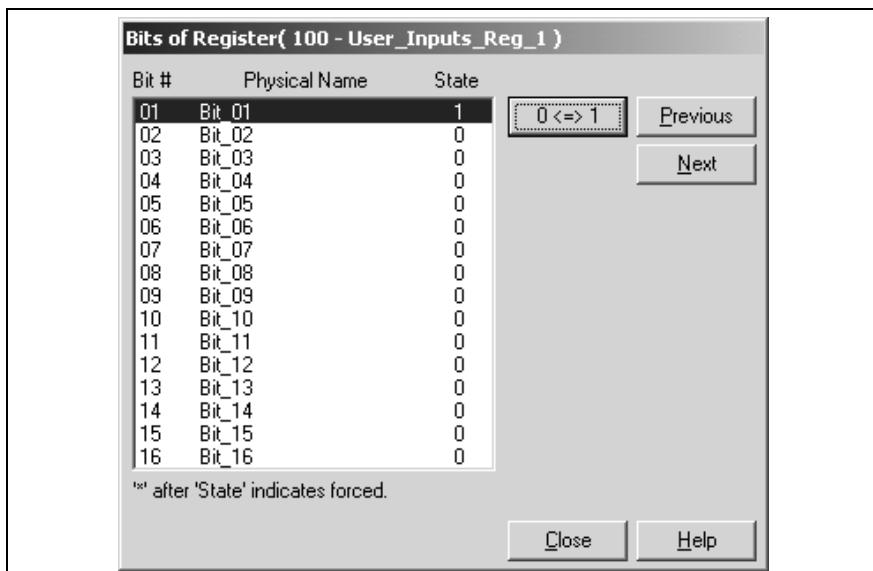
Considering the above I/O Mapper Strings, register 001 - bit 01 is map to register 100 - bit 01. In this case, the state of register 100 - bit 01 will control the Parameter Mode function of the system.

To switch the system to Parameter Mode...

1. Close all I/O Mapper windows and return to VMT's main window.
2. Select **On-Line Data** ⇒ **Registers** to open the Active Program Registers window.
3. Click and hold the scroll bar button and scroll down to register 100.

Note: Register numbers appear at the top of the window as you scroll down or up.

4. Click and highlight **Bit_01** and select the **0 <=> 1** button to change the **State** of **Bit_01** from 0 to 1.



The DDE Server should now display *002 Parameter Mode*.

Note: Refer to the Enhanced I/O Mapper chapter of the VisualMotion 8 Functional Description for an explanation on how to create an I/O Mapper File.

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.

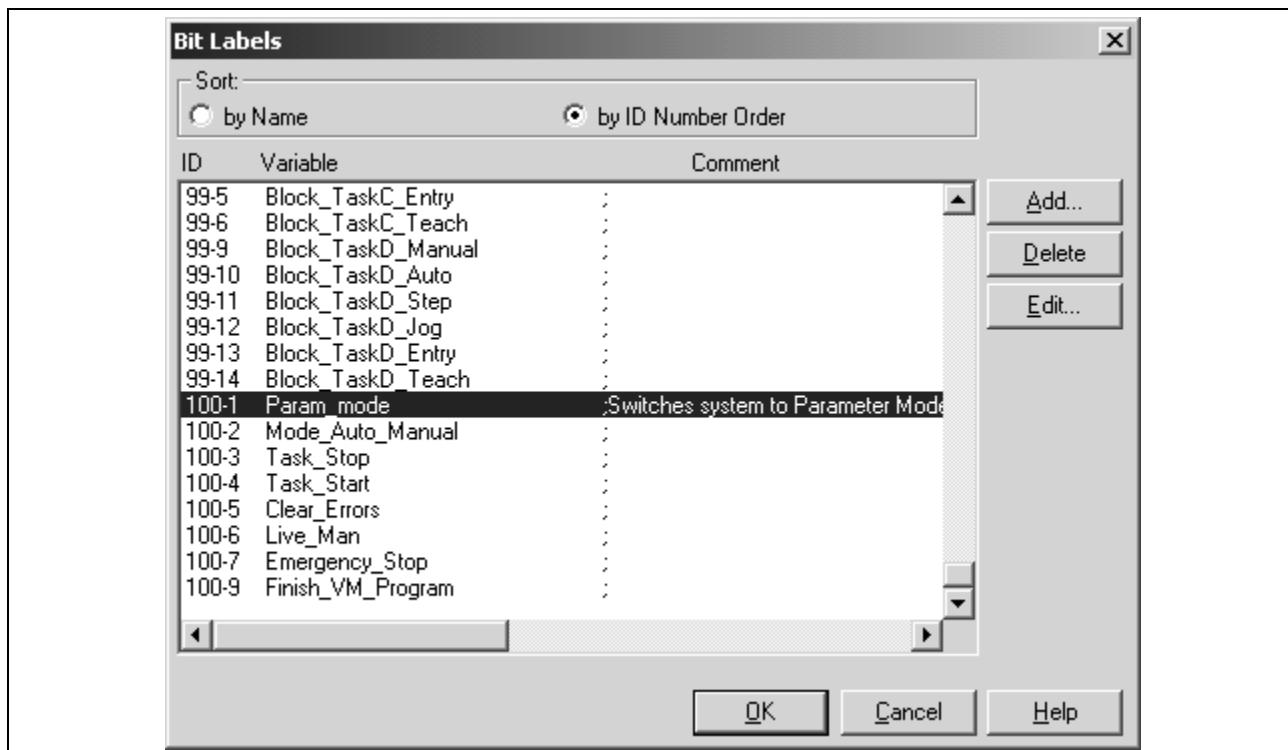
1. Start the I/O Mapper and select, **File** ⇒ **Open**, locate the *Def100.iom* file and click on **Open**. The Ladder representation of the file will open in the Ladder window.
2. To download the I/O Mapper to the control, select **File** ⇒ **Send Ladder to control** or click on the download icon (). The control must 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** ⇒ **Registers** from VisualMotion's main menu to modify registers.

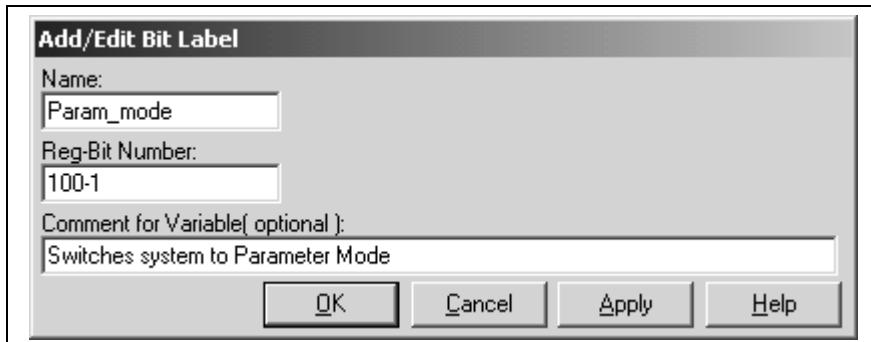
Bit Labels

Bit labels are names given to register bits making them easier to identify and use. Bit label information is saved with each VisualMotion user program file. Once the following labels have been added, the sample program must be saved, compiled, downloaded and activated in the control.

1. Select **Edit Labels** ⇒ **Bit Labels...** under the **Edit** menu.

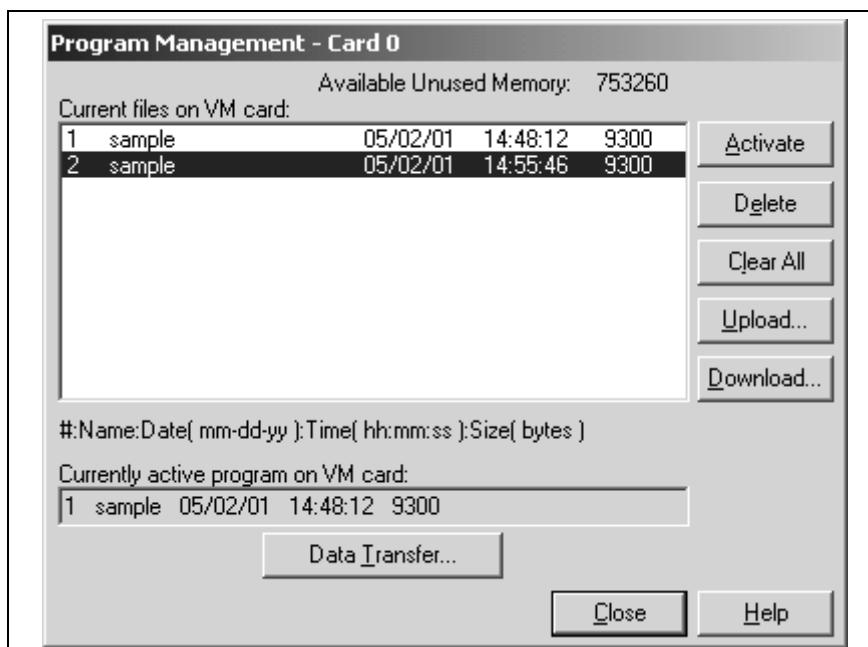


2. Select Add... and enter the labels shown above for **Reg.-Bit Number 100-1 to 100-9**.



3. After adding the last bit label, click on **OK** to close the *Add/Edit Bit Label* window.
4. Select **Save**, **Compile**, **Download** from the **Build** menu or select the icon from the Toolbar.
5. After the download is complete, select **Build** ⇒ **Program Management** and click on the Activate button. To activate a different program, highlight the file and activate.

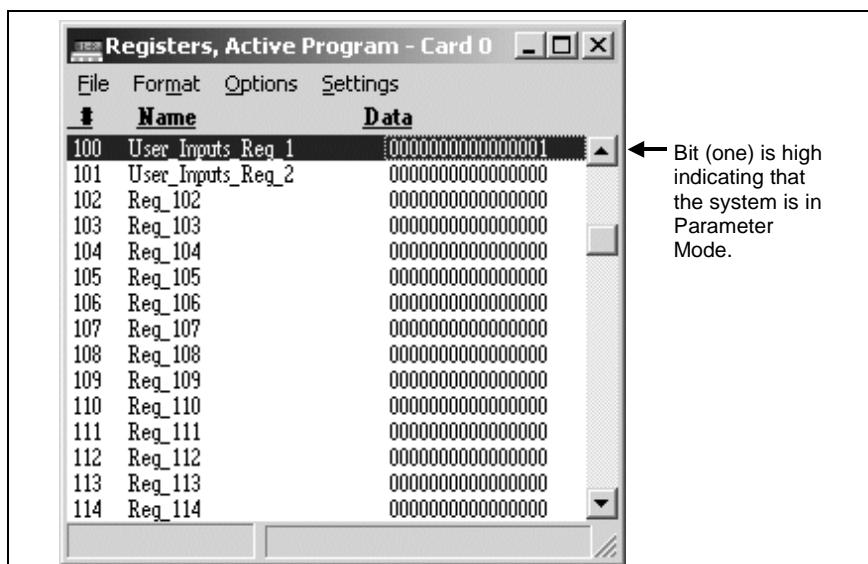
Note: If a program is activated while a different program is currently running, VisualMotion Toolkit will issue an error. Stop all motion before activating a new program.



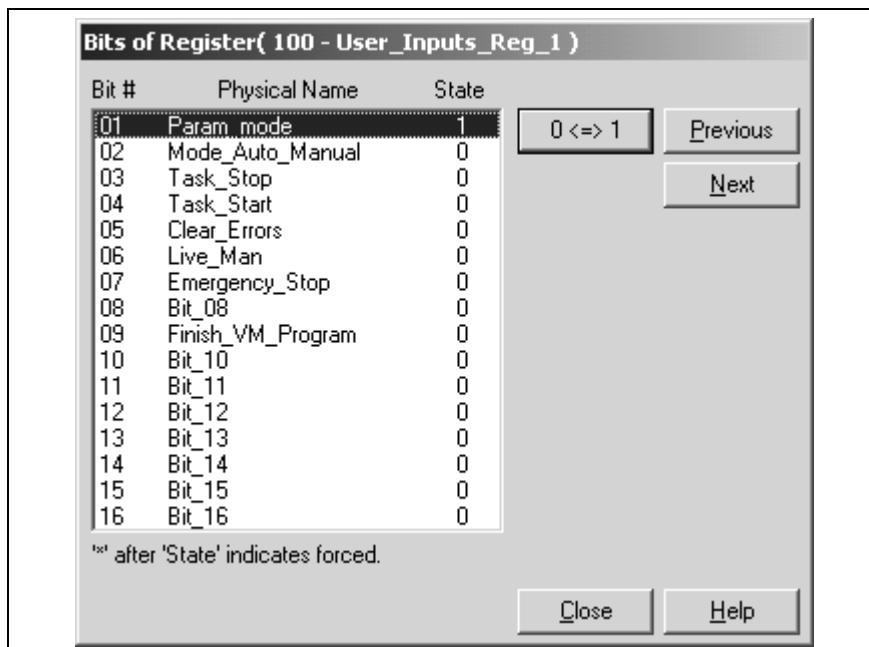
3.5 Run VisualMotion Program

The following procedure covers program execution and operation. Before following this procedure be sure that the initial setup covered in section *Program Execution* on page 3-27 has been completed. The program should be opened within VisualMotion Toolkit and activated in the control.

1. Select **On-Line Data** \Rightarrow **Registers** from the main menu. Scroll down to register 100 and double-click on it to open.

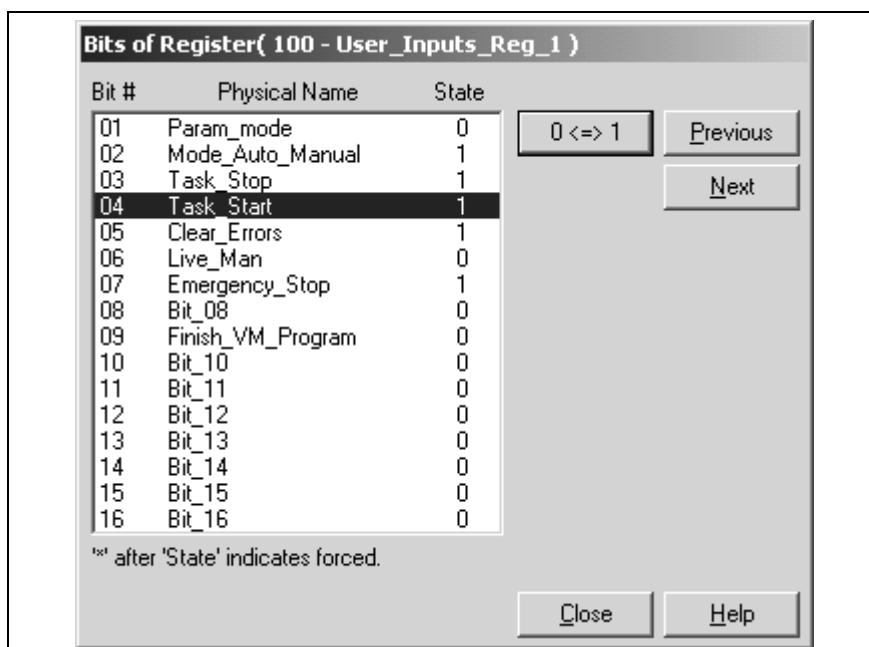


The bits should now be listed along with their corresponding labels (Physical Name).



2. Change the state of the following bits as listed in the order shown below:

Bit	Label	State
07	Emergency-Stop	0 to 1
01	Param_Mode	1 to 0
05	Clear_Errors	0 to 1
02	Mode_Auto_Manual	0 to 1
03	Task_Stop	0 to 1
04	Task_Start	0 to 1



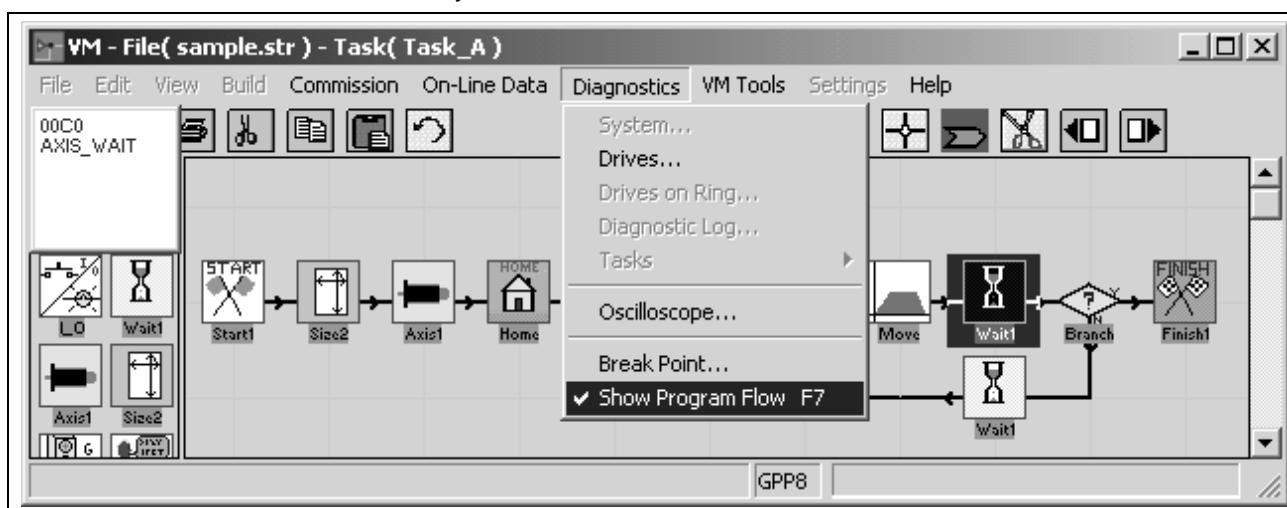
3. The program should now be running with the bits in this state.

Note: If variables were added to the sample program, make sure that they have been assigned a value before starting the program.

Refer to *Assigning a Value to a Variable* on page 3-24.

Note: If the program has been started prior to adding values to the variables, the program will not run because all variable values are initially zero. Stop the program by changing the state of bits 3 and 4 from 1 to 0. Once values have been added, reinitialize the program by changing the state of bits 3 and 4 back to 1.

To view which program icon is being processed within VisualMotion Toolkit, select **Tools** ⇒ **Show Program Flow** or press **F7** on the keyboard to turn the feature on or off.



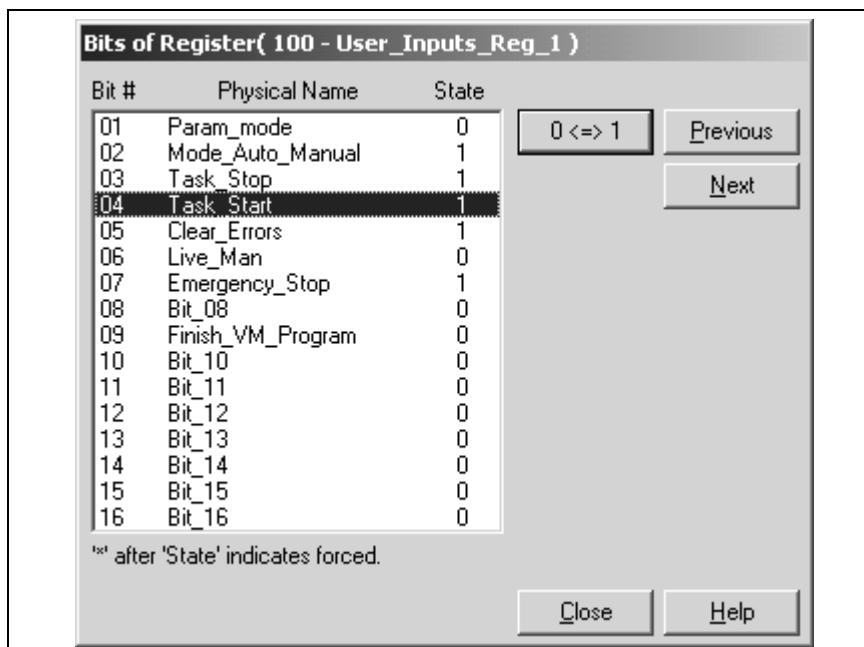
Program Operation

Changing the state () of any one of the following bits in Register 100 will stop the program from running.

Bit	Label
01	Param_Mode
02	Mode_Auto_Manual
03	Task_Stop
07	Emergency-Stop
09	Finish_VM_Program

If the program was stopped using bit 100-7, *Emergency Stop*, or from an error, bit 100-5 *Clear_Errors* must be toggled from 1 to 0 to 1 before running again.

To run the program again, toggle bit 100-4 *Task Start* from 1 to 0 to 1 with the other bits in the state shown below:



Note: In order to start the program from the very beginning, toggle bit 100-2, *Mode_Auto_Manual*, from 0 to 1 prior to toggling the *Task_Start* bit.

3.6 Demo Programs

Linear Motion Application Demos

As part of the installation of VisualMotion Toolkit 8, three sample programs are installed along with the application.

The programs are both linear motion applications and are called...

- Position Mode (filename PositionMode.str)
- Coordinated Mode (filename CoordinatedMode.str)

These programs can be opened in VisualMotion Toolkit by selecting ***File*** ⇒ ***Sample Programs***.

Note: A third program that appears under the Sample Program menu selection is called Measuring Wheel GPP. For purposes of demonstration, only the first two programs will be explained.

Both application demos automatically setup all the required variables and values. The user simply starts the programs using the appropriate user register and bits.

Setup

Use the following steps to load, compile, download and run each sample program.

1. Start the VisualMotion Toolkit program. From the main menu select ***File*** ⇒ ***Sample Programs*** ⇒ ***Position Mode*** or ***Coordinated Mode***
2. The next step will be to Save, Compile and Download the program to the control. Click on the  button. Once the file has been downloaded to the control, select it from the *Program Management* window and click on the “Activate” button.
3. Use the default I/O Mapper (Def100.iom) located in directory ***Indramat/VisualMotion8/Param***. Refer to *I/O Mapper* on page 3-30.
4. To run each demo program, select register 100 from ***On-Line Data*** ⇒ ***Registers*** and change the state of the following bits.

Bit	Label	State
07	Emergency-Stop	0 to 1
01	Param_Mode	1 to 0
05	Clear_Errors	0 to 1
02	Mode_Auto_Manual	0 to 1
03	Task_Stop	0 to 1
04	Task_Start	0 to 1
10	Motion start Bit	0 to 1

Refer to *Run VisualMotion Program* on page 3-33 for examples.

VisualMotion Icon Program – Position Mode (Single Axis)

Position Mode (PositionMode.str) is a single axis program that when started using register 100, loads and sets up the axis and required variable values. The program then waits for the activation of register 100 bit 10 to begin motion. Once motion begins, values for the variables can be modified by selecting **On-Line Data** ⇒ **Variables** and changing the program variables.

Single Axis Program Variables

The following variables are used in this program and have been assigned a corresponding label:

F1 - Return_Position	I1 - Timer_1 (Return Dwell)
F2 - Forward_Position	I2 - Timer_2 (Forward Dwell)
F3 - Return_Speed	I3 - Part_Counter
F4 - Forward_Speed	
F5 - Accel_Rate	

Note: The dash lines in the following figure are used to simply point the subroutines that are being called out in the program. They are not connection lines in the actual program

To view subroutines in a VisualMotion program select **View** ⇒ **Subroutines** and select the desired subroutine.

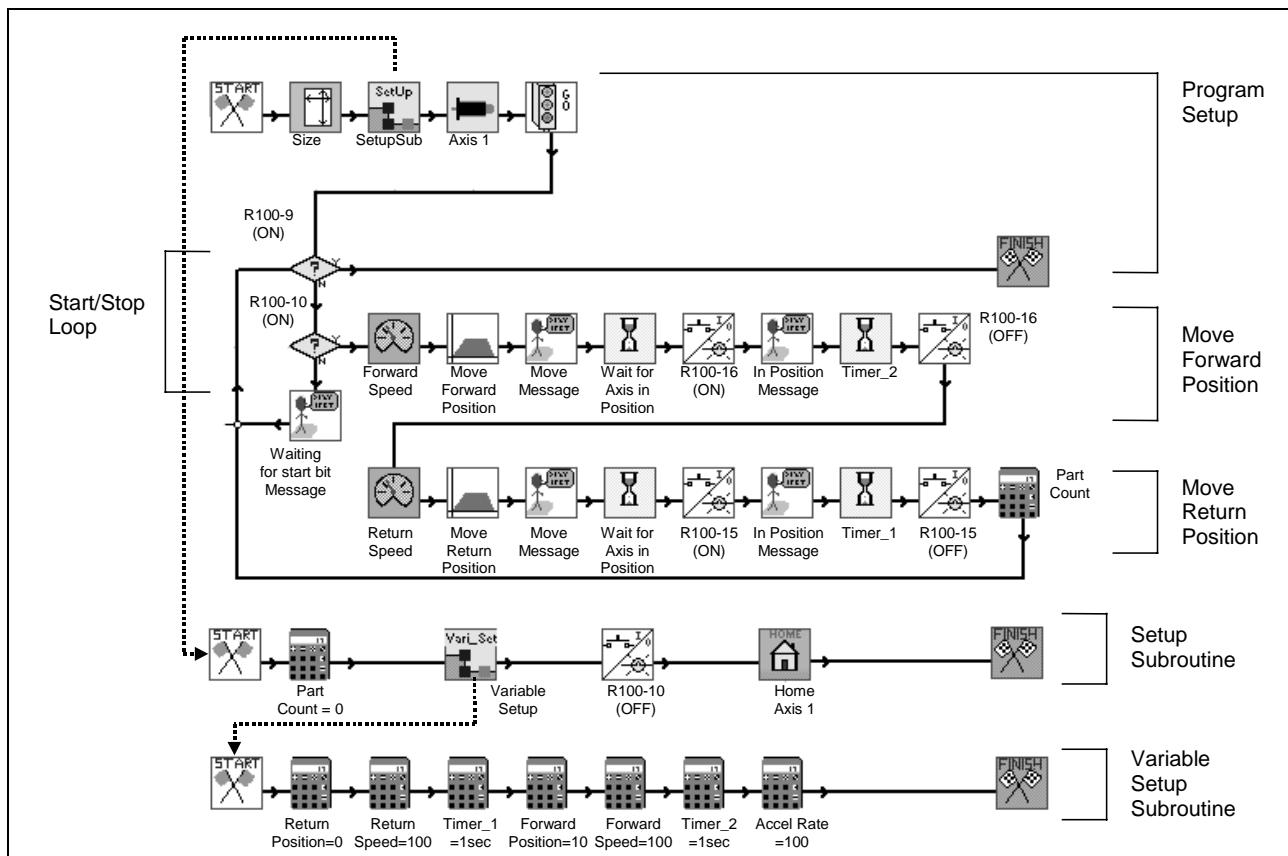


Fig. 3-8: Position Mode Sample Program

Program Setup

The section in Fig. 3-9 starts the program, allocates memory space for variables and sets up axis 1. A setup subroutine is then executed. The Go icon enables axis 1 for non-coordinated motion. The branch icon continues the program if register 100 bit 9 = 0. If the bit is high (1), the program will end.

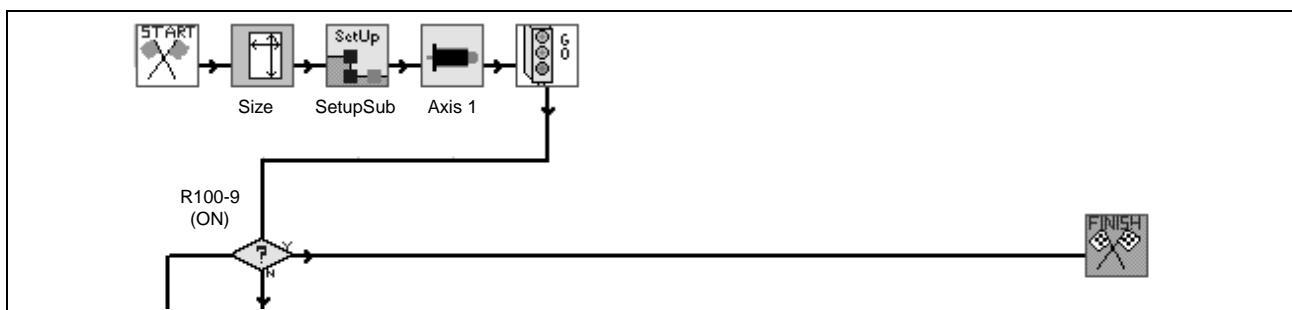


Fig. 3-9: Program Setup

Start / Stop Loop

The section in Fig. 3-10 starts and stops forward and return motion dependent on register 100 bit 10. If bit 10 is off the program will loop through the "Waiting for Start Bit" message icon. If bit 10 is on, the program will continue down to the forward and return move sections. After each cycle the program returns to this branch, adds 1 to the part count and then checks the status of bit 10 again.

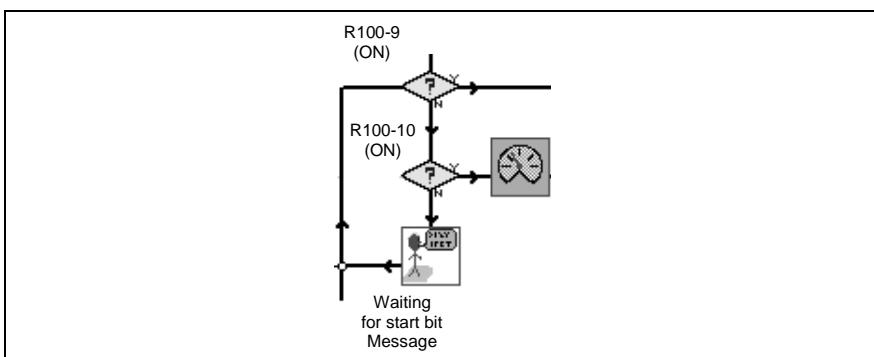


Fig. 3-10: Start / Stop Loop

Move Forward Position

The section in Fig. 3-11 sends the velocity rate value (*Forward_Speed*) and move distance value (*Forward_Position*) to the drive and initiates the forward position move. The message icon sends a "Moving to forward position" status message. The wait icon suspends further program execution until the axis has reached the forward position. Once in the forward position, register 100 bit 16 turns on the "in position" indicator light on the interface. The second wait icon is used to cause the forward dwell (*Timer_2*). After the dwell, register 100 bit 16 is turned off (reset to 0).

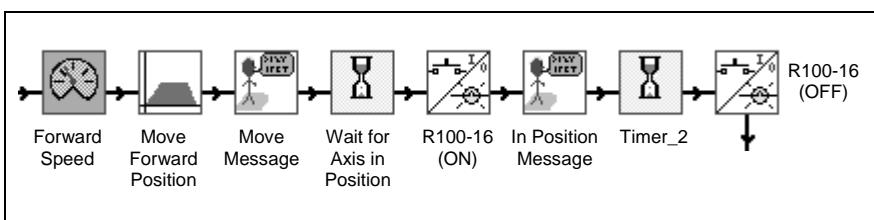


Fig. 3-11: Move Forward Position

Move Return Position

The section in Fig. 3-12 sends the velocity rate value (*Return_Speed*) and move distance value (*Return_Position*) to the drive and initiates the reverse position move. The message icon sends a “Moving to return position” status message. The wait icon suspends further program execution until the axis has reached the reverse position. Once in the return position, register 100 bit 15 turns on the “in position” indicator light on the interface. The second wait icon is used to cause the return dwell (*Timer_1*). After the dwell, register 100 bit 15 is turned off (reset to 0).

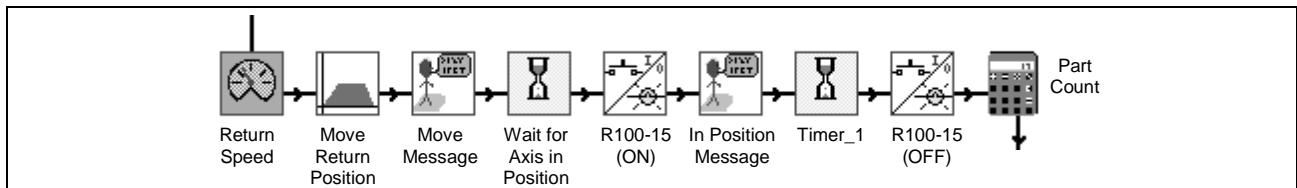


Fig. 3-12: Move Return Position

Setup Subroutine (Counter Reset)

The subroutine in Fig. 3-13 resets the part counter to zero and runs the following Variable Setup subroutine. It moves Axis 1 to its home position. After running the subroutine the program returns back to Task A.

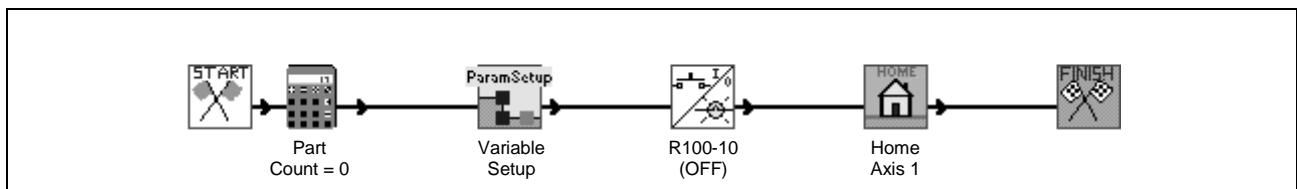


Fig. 3-13: Setup Subroutine (Counter Reset)

Variable Setup Subroutine

The subroutine in Fig. 3-14 sets the variable values for all Floats and Integers. The user can make changes to these variables by selecting ***On-Line Data* ⇒ *Variables***.

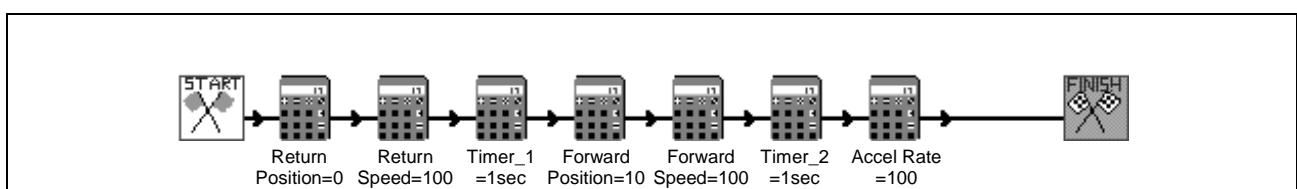


Fig. 3-14: Variable Setup Subroutine

- | | |
|-----------------------|------------------------------|
| F1 - Return_Position | I1 - Timer_1 (Return Dwell) |
| F2 - Forward_Position | I2 - Timer_2 (Forward Dwell) |
| F3 - Return_Speed | I3 - Part_Counter |
| F4 - Forward_Speed | |
| F5 - Accel_Rate | |

VisualMotion Icon Program - Coordinated Mode

Coordinated Mode (CoordinatedMode.str) is a coordinated motion program that is similar to *Position Mode* with the exception of movement type. *Coordinated Mode* uses a series of points within an absolute point table that contains the values required for motion. The starting and ending of *Coordinated Mode* is identical to that of *Position Mode*.

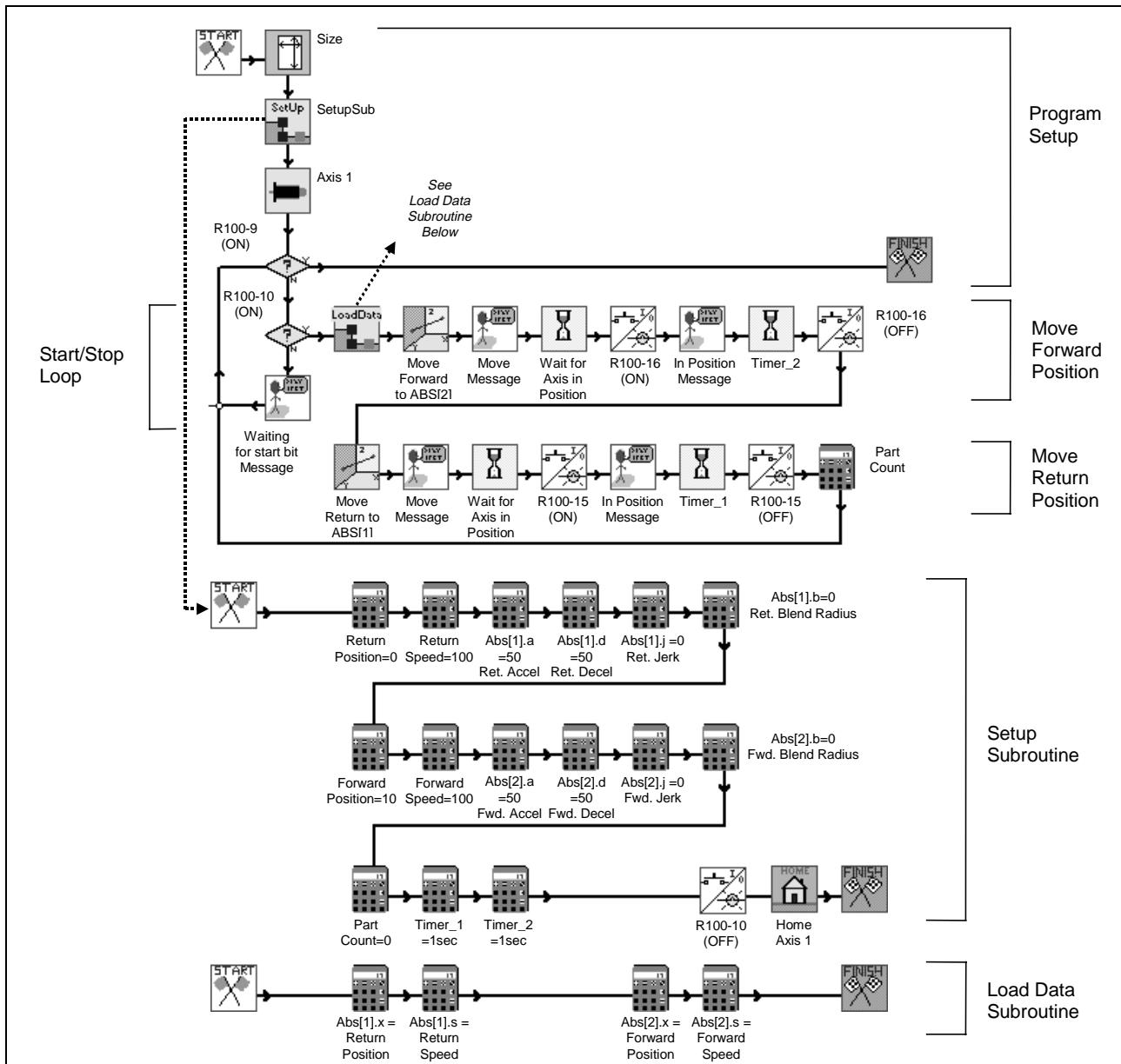


Fig. 3-15: Coordinated Mode Sample Program

Move Forward Position

Coordinated Mode uses coordinated motion. The velocity and move icons used in *Position Mode* have been replaced with a load data subroutine and a path icon. The load data subroutine (refer to Fig. 3-19) assigns values to the absolute point table.

- Path Icon** The path icon uses these values to set up coordinated absolute motion. An absolute move begins from the current position (or the endpoint of the previous path segment) and terminates at the absolute point specified. In this case the forward position would be ABS[2]. Speed, acceleration and deceleration values are also provided with each point move.

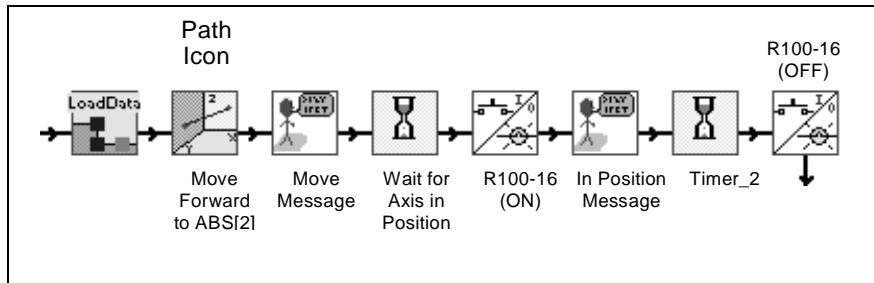


Fig. 3-16: Move Forward Position

The absolute point table can be viewed by selecting **On-Line Data** \Rightarrow **Points**.

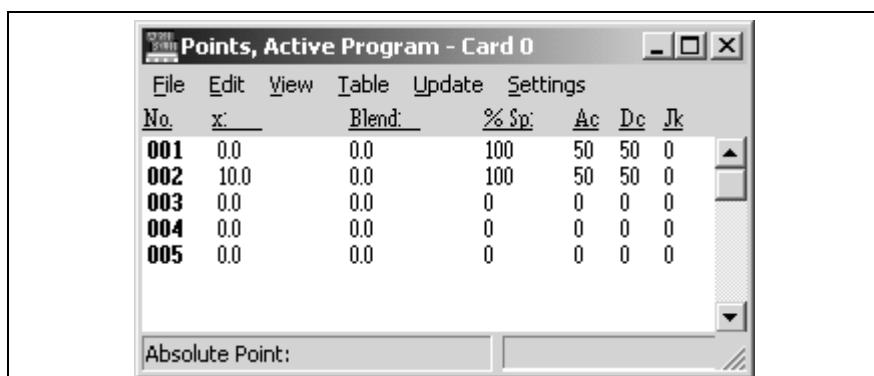


Fig. 3-17: Absolute Point Table

The return position is ABS[1], so the return move will begin at ABS[2] and then end at ABS [1].

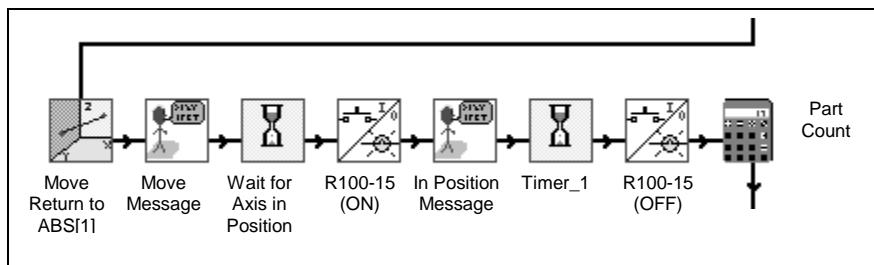


Fig. 3-18: Move Return Position

Load Data Subroutine

The subroutine in Fig. 3-19 associates the variables F1-F4 (Return/Forward Position and Speed) to the absolute point table (Abs[1].x, Abs[2].x, Abs[1].s, Abs[1].x).

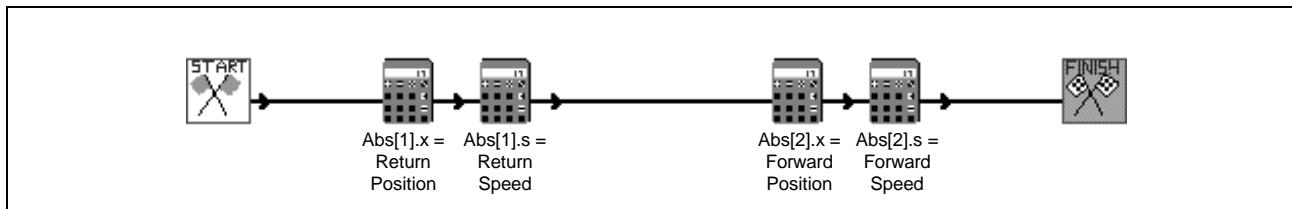


Fig. 3-19: Load Data Subroutine

Variable and Point Table Setup Subroutine

The variable and point table subroutine in Fig. 3-20 assigns values to the associations made within the load data subroutine for Forward/Return Position and Speed. This subroutine also adds values to the absolute point table for axis acceleration ($\text{Abs}[\#].a$), deceleration ($\text{Abs}[\#].d$), jerk($\text{Abs}[\#].j$, and blend radius ($\text{Abs}[\#].b$). In addition, the axis is homed and register 100 bit 10 is turned off.

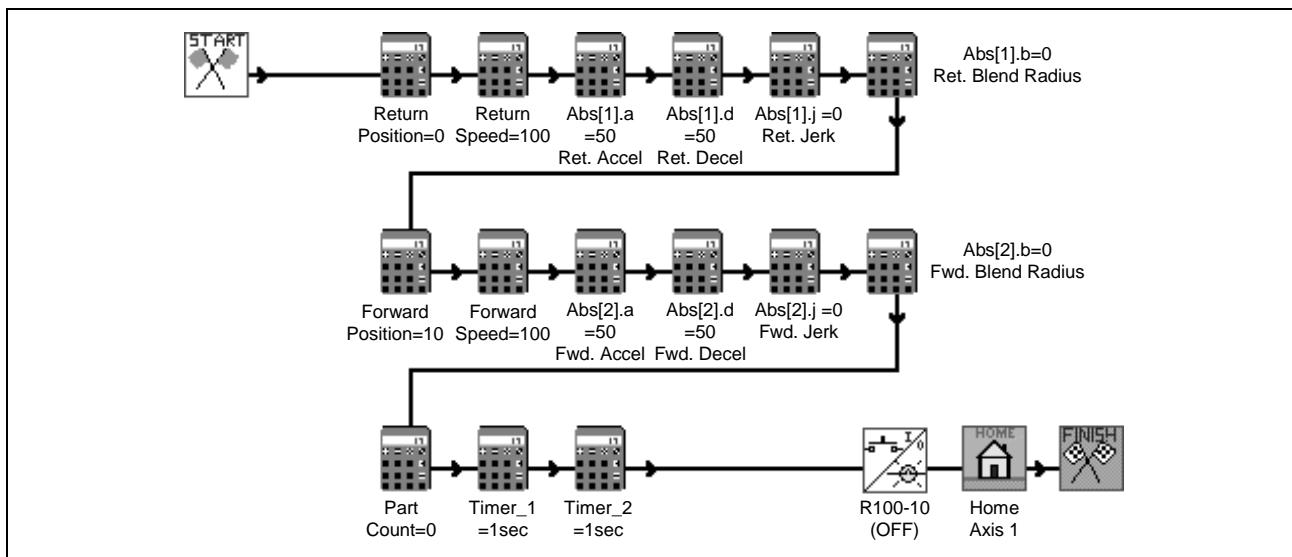


Fig. 3-20: Variable and Point Table Setup Subroutine

4 Multiple Master Overview

4.1 VisualMotion GPP Overview

The Multiple Master functionality of GPP provides the ability of having more than one active master at a time. Electronically synchronized axes can be combined to form Electronic Line Shafting (ELS) Groups. Active masters can control a maximum of eight ELS Groups. Every ELS Group will follow its selected master.

GPP supports six system masters in any combination up to a maximum of the following types:

- 2 Virtual Masters
- 3 Real Masters
- 4 ELS Group Masters

The example configuration in Fig. 4-1 shows 2 Virtual Masters, 2 Real Masters, 1 ELS Group Master (5 masters) and 4 ELS Groups.

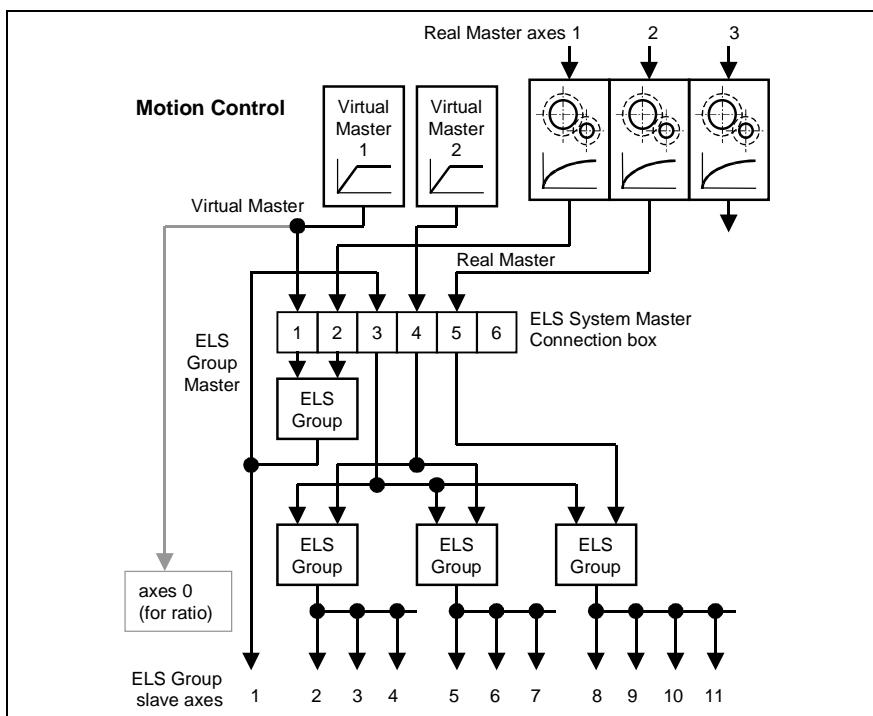


Fig. 4-1: Multiple Master Configuration Example

4.2 Multiple Master Overview

The following overview outlines the main areas of the Multiple Master functionality in GPP firmware.

Virtual Master

Two independent Virtual masters provide positional command signals that are used to drive a group of axes or programmable limit switches. A Virtual Master has two primary modes of operations:

- Velocity Mode
- Position Mode

Refer to chapter 5.2 *Virtual Master* for details.

Real Master

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.

ELS Group Master

An ELS Group Master is the output of an ELS Group used as an input master signal, geared by an M/N ratio, to a different ELS Group.

ELS System Master

Virtual Masters, Real Masters and ELS Group Masters can be combined and assigned to one of 6 ELS System Masters. The master signal from each active system master is conditioned (e.g., geared or filtered) and made available for controlling groups of axes. Refer to 5.3 *Electronic line Shafting Master Assignment* for details.

ELS Group

An ELS Group is defined as a set of slave axes that follow the position command signal from one of the 6 System Masters. By using ELS Groups, slave axes are sectionalized into functional groups that can easily control each machine section as an independent process. During operation, an ELS Group can be switch between master signals. Any changes to ELS Group parameters are immediately available to all axes assigned to a group, keeping the machine section precisely synchronized. Refer to 5.4 *ELS Line Shafting Groups* for details.

5 Multiple Master Functionality in VisualMotion 8 (GPP)

5.1 Initialization of VisualMotion's Multiple Master Functionality

The Multiple Master functionality in VisualMotion is initialized and controlled using the registers and program variables of the associated VisualMotion program.

Assigning Registers in VisualMotion

VisualMotion provides 512 registers for controlling and monitoring of the program. Table 5-1 shows a listing of predefined registers.

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: PPC Predefined Register Structure

During initialization of both Virtual Masters and ELS Groups, the associated registers should be assigned to a free area of the register block. The assigning of registers is defined in the Virtual Master and ELS Group icons, respectively.

Note: When assigning registers, make sure not to use a register that is already used by the PPC or any other device such as the I/O Mapper, CAM indexer or ELS Group.

Virtual Master and ELS Group Default Registers

To avoid using the same registers, the following register numbers in Table 5-2 for Virtual Masters and Table 5-3 for ELS Groups can be used as defaults.

Virtual Master	Control Register	Status Register
1	150	180
2	151	181

Table 5-2: Virtual Master Default Registers

ELS Group	Control Register	Status Register
1	152	182
2	153	183
3	154	184
4	155	185
5	156	186
6	157	187
7	158	188
8	159	189

Table 5-3: ELS Group Default Registers

Note: It is strongly recommended that the programmer use default register assignments. This makes documentation and modifications to user programs an easier task over the scope of a project.

Assigning Program Variables

Values that are used by VisualMotion to run the program, such as Virtual Master velocity or acceleration, are stored as program variables.

Program variables are designated as F# for Floats and I# for Integers. The Size icon in VisualMotion determines the number of program variables allocated to the control. Program variables retain their values during power off.

Float Variables (F#) A variable whose value can be a number containing a decimal point.

Integer Variables (I#) Integers are signed or unsigned whole numbers, such as -3 or 5. Variable start ID blocks are assigned for Virtual Masters, ELS Masters and ELS Groups within their respective icons. Table 5-4 contains the default start ID blocks for each of the mentioned program variables.



Virtual Master
Icon



ELS Master
Assignment
Icon



ELS Group
Icon

Function	Number of Floats	Number of Integers	Float ID Block	Integer ID Block
Virtual Master 1	15	2	F100-F114	I100-I101
Virtual Master 2	15	2	F120-F134	I105-I106
ELS Master Assignment	24	30	F140-F163	I110-I139
ELS Group 1	26	9	F170-F195	I140-I148
ELS Group 2	26	9	F200-F225	I150-I158
ELS Group 3	26	9	F230-F255	I160-I168
ELS Group 4	26	9	F260-F285	I170-I178
ELS Group 5	26	9	F290-F315	I180-I188
ELS Group 6	26	9	F320-F345	I190-I198
ELS Group 7	26	9	F350-F375	I200-I208
ELS Group 8	26	9	F380-F405	I210-I218

Table 5-4: Program Variable Default Start ID Blocks

Assigning Default Labels to Registers and Program Variables

Labels are assigned to program variables in VisualMotion as names to clearly define the value's function. VisualMotion provides default labels and comments for all registers and program variables.

The user can select the default labels and comments by clicking on the **Assign Variable Labels ...** button within the Virtual Master, ELS Master Assignment and ELS Group icons. A Dialog window opens allowing the user to add labels and comments to registers and program variables by clicking on the **Add Default Labels** button for the selected data type. Fig. 5-1 shows an example of adding default labels in the dialog window.

Note: Register and variable labels can be modified by selecting **Edit** ⇒ **Edit Labels** ⇒ **Variable Labels, Register Labels or Bit Labels** in VisualMotion.

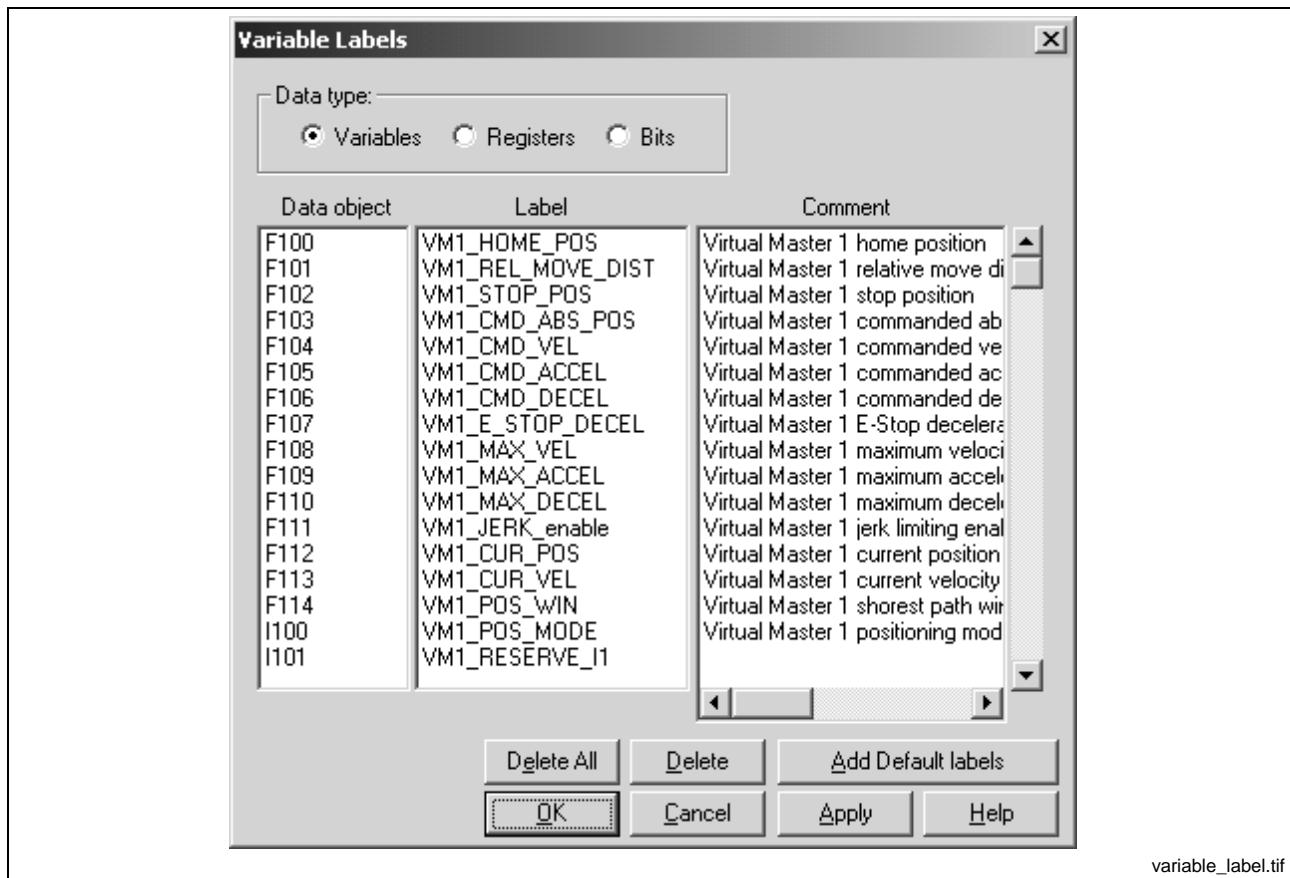


Fig. 5-1: Add Default Labels Example

Virtual Master 1 & 2 Default Register Labels

The default labels for the Virtual Master registers are shown in Table 5-5. The corresponding default bit labels are shown Table 5-6.

Data Object Ttype	Label (20 character limit)	Comment (80 character limit)
Assigned control register number	VM#_CONTROL_REG	Virtual Master # control register
Assigned status register number	VM#_STATUS_REG	Virtual Master # status register

Table 5-5: Virtual Master Default Registers

Default Label Virtual Master 1 & 2 Control Register	Data Object Virtual Master 1 Control Register-Bit	Data Object Virtual Master 2 Control Register-Bit	Comment (80 character limit) Virtual Master 1 & 2 Control Register
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
Default Label Virtual Master 1 & 2 Status Register	Data Object Virtual Master 1 Status Register-Bit	Data Object Virtual Master 2 Status Register-Bit	Comment (80 character limit) Virtual Master 1 & 2 Status Register
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 5-6: Virtual Master 1 & 2 Default Register Bits

Virtual Master 1 & 2 Default Program Variable Labels

Default Label Virtual Master 1 & 2 Program Variable	Data Object Virtual Master 1 & 2	Comment (80 character limit) Virtual Master 1 & 2 Program Variable				Default Value	Units	Update Mode	
VM#_HOME_POS	F100	F120	Virtual Master # home position				0	Degrees	Phase 4
VM#_REL_MOVE_DIST	F101	F121	Virtual Master # relative move distance				1	Degrees	Phase 4
VM#_STOP_POS	F102	F122	Virtual Master # stop position				0	Degrees	Phase 4
VM#_CMD_ABS_POS	F103	F123	Virtual Master # commanded absolute position				0	Degrees	Phase 4
VM#_CMD_VEL	F104	F124	Virtual Master # commanded velocity				20	RPM	Phase 4
VM#_CMD_ACCEL	F105	F125	Virtual Master # commanded acceleration				100	Rad/sec ²	Phase 4
VM#_CMD_DECEL	F106	F126	Virtual Master # commanded deceleration				100	Rad/sec ²	Phase 4
VM#_E_STOP_DECEL	F107	F127	Virtual Master # E-Stop deceleration				500	Rad/sec ²	Phase 2
VM#_MAX_VEL	F108	F128	Virtual Master # maximum velocity				3200	RPM	Phase 2
VM#_MAX_ACCEL	F109	F129	Virtual Master # maximum acceleration				1000	Rad/sec ²	Phase 2
VM#_MAX_DECEL	F110	F130	Virtual Master # maximum deceleration				1000	Rad/sec ²	Phase 2
VM#_JERK_ENABLE	F111	F131	Virtual Master # jerk limiting enable				1		Phase 2
VM#_CUR_POS	F112	F132	Virtual Master # current position				0	Degrees	Phase 4
VM#_CUR_VEL	F113	F133	Virtual Master # current velocity				0	RPM	Phase 4
VM#_POS_WIN	F114	F134	Virtual Master # shortest path window				1	Degrees	Phase 2
VM#_POS_MODE	I100	I105	Virtual Master # positioning mode ^{1.)}				0		Phase 2
VM#_RESERVE_I1	I101	I106							
Each # symbol represents an entry for the number of the Virtual Master Note 1.) Absolute Position Mode, 0=Positive, 1= Negative, 2= Shortest Path									

Table 5-7: Virtual Master 1 & 2 Default Program Variables

ELS Master Assignment Default Program Variable Labels

Default Label ELS Master Assignment Program Variable	Data Object ELS Master Assignment						Comment (80 character limit) ELS Master Assignment Program Variable	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 5-8: ELS Master Assignment Default Program Variables

ELS Master Variable Definition

ELS_MSTR_FREQ#	Only Real Masters use the filter constant. When a filter (<i>ELS_MSTR_FLTR#</i>) is selected for an axis' position feedback, a cutoff frequency for the filter must be entered. The cutoff frequency is the frequency where the signal is reduced by 3dB.
ELS_MSTR_M# and ELS_MSTR_N#	Only Real Masters use the ratio constants (M/N). The output of the master is governed by the equation $y=(M/N)*x$, where x is the feedback value from the real master and y is the master signal used for ELS Groups. All ELS Masters and ELS Groups outputs are modulo 360 degrees.
ELS_MSTR_A#	This variable identifies a valid ID number for a defined master type. For example, when <i>ELS_MSTR_TYPE#</i> is set to 3 (Virtual Master) this number must be a 1 or 2. Valid ID numbers are... <ul style="list-style-type: none"> • Virtual Master: 1 or 2 • ELS Group Master: 1 - 8 • Real Master: 1 - 3
ELS_MSTR_EC#	When using an encoder device as a Real Master, this variable identifies the source. <ul style="list-style-type: none"> • 0 = motor encoder • 1 = external encoder
ELS_MSTR_FLTR#	This variable identifies the type of filtering to use for the axis position feedback. Valid types are... <ul style="list-style-type: none"> • 0 = no filter • 1 = 1st order low pass • 2 = 2nd order low pass • 3 = 3rd order low pass • 4 = 2nd order Butterworth • 5 = 3rd order Butterworth • 6 = 2nd order low pass with velocity feed forward • 7 = 3rd order low pass with velocity and acceleration feed forward
ELS_MSTR_TYPE#	Available master types are... <ul style="list-style-type: none"> • 0 = Real Master • 1 = ELS Group master • 2 = External (future development) • 3 = Virtual Master • 4 = none

ELS Group 1- 8 Default Register Labels

The default labels for the ELS Group registers are shown in Table 5-9. The corresponding default bit labels are shown in Table 5-10.

Data Object Type	Label (20 character limit)	Comment (80 character limit)
Assigned control register number	G#_CONTROL_REG	Group # control register
Assigned status register number	G#_STATUS_REG	Group # status register

Table 5-9: ELS Group 1- 8 Default Registers

Default Label ELS Group 1-8 Control Register	Data Object ELS Group Control Register-Bit								Comment (80 character limit) ELS Group 1-8 Control Register
	1	2	3	4	5	6	7	8	
G#_CT_LOCK_OFF	152-1	153-1	154-1	155-1	156-1	157-1	158-1	159-1	Group # control, 0 → 1 start lock cycle, 1 → 0 start unlock
G#_CT_M_REL_PH	152-2	153-2	154-2	155-2	156-2	157-2	158-2	159-2	Group # control, 0 → 1 triggers master relative phase adjust
G#_CT_S_REL_PH	152-3	153-3	154-3	155-3	156-3	157-3	158-3	159-3	Group # control, 0 → 1 triggers slave relative phase adjust
G#_CT_MSTR_SEL	152-4	153-4	154-4	155-4	156-4	157-4	158-4	159-4	Group # control, 0=master 1, 1=master 2
G#_CT_VAR_CLK	152-5	153-5	154-5	155-5	156-5	157-5	158-5	159-5	Group # control, 0 → 1 forcing
G#_CT_LOCAL	152-6	153-6	154-6	155-6	156-6	157-6	158-6	159-6	Group # control, 0 → 1 local mode, 1 → 0 selected master
G#_CT_JOG_INC	152-7	153-7	154-7	155-7	156-7	157-7	158-7	159-7	Group # control, 0=continuous jog mode, 1=incremental jog mode
G#_CT_JOG_ABS	152-8	153-8	154-8	155-8	156-8	157-8	158-8	159-8	Group # control, 0=absolute incremental mode, 1=relative incremental mode
G#_CT_JOG_PLUS	152-9	153-9	154-9	155-9	156-9	157-9	158-9	159-9	Group # control, 0 → 1 starts jog mode in positive direction
G#_CT_JOG_MINS	152-10	153-10	154-10	155-10	156-10	157-10	158-10	159-10	Group # control, 0 → 1 starts jog mode in negative direction
Default Label ELS Group 1-8 Status Register	Data Object ELS Group Status Register-Bit								Comment (80 character limit) ELS Group 1-8 Status Register
	1	2	3	4	5	6	7	8	
G#_ST_LOCK_ON	243-1	244-1	245-1	246-1	247-1	248-1	249-1	250-1	Group # status, 0=unlocked, 1=locked to master
G#_ST_M_REL_PH	243-2	244-2	245-2	246-2	247-2	248-2	249-2	250-2	Group # status, 1=acknowledges master relative phase adjust
G#_ST_S_REL_PH	243-3	244-3	245-3	246-3	247-3	248-3	249-3	250-3	Group # status, 1=acknowledges slave relative phase adjust
G#_ST_MSTR_SEL	243-4	244-4	245-4	246-4	247-4	248-4	249-4	250-4	Group # status, 0=master 1, 1=master 2
G#_ST_VAR_ACK	243-5	244-5	245-5	246-5	247-5	248-5	249-5	250-5	Group # status, 1=variables updated
G#_ST_LOCAL	243-6	244-6	245-6	246-6	247-6	248-6	249-6	250-6	Group # status, 1=local mode active
G#_ST_RSVD7	243-7	244-7	245-7	246-7	247-7	248-7	249-7	250-7	
G#_ST_RSVD8	243-8	244-8	245-8	246-8	247-8	248-8	249-8	250-8	
G#_ST_MOTION	243-9	244-9	245-9	246-9	247-9	248-9	249-9	250-9	Group # status, 0=no motion, 1=group is in motion
G#_ST_JOG_POS	243-10	244-10	245-10	246-10	247-10	248-10	249-10	250-10	Group # status, 1=jog is at absolute target

Each # symbol represents an entry for the number of the ELS Group

Table 5-10: ELS Group 1- 8 Default Register Bits

ELS Group 1- 8 Default Program Variable Labels

Default Label ELS Group 1-8	Data Object ELS Group 1-8 Program Variable								Comment (80 character limit) ELS Group 1-8	Update Mode
	1	2	3	4	5	6	7	8		
Program Variable								Program Variable		
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

Table 5-11: ELS Group 1- 8 Default Program Variables (part 1 of 2)

ELS Group 1- 8 Default Program Variable Labels (Cont'd)

Default Label ELS Group 1-8 Program Variable	Data Object ELS Group 1-8 Program Variable								Comment (80 character limit) ELS Group 1-8 Program Variable	Update Mode
	1	2	3	4	5	6	7	8		
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
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 5-12: ELS Group 1- 8 Default Program Variables (part 2 of 2)

ELS Group Configuration Word

For every ELS Group, an ELS Group configuration word (**G#_CONFIG**) is used to configure all settings for Switching Synchronization, Phase Control and Initialization. These settings are initially configured within VisualMotion Toolkit's ELS Group icon and active once the program is compiled and downloaded to the control. These settings can also be modified by accessing the appropriate integer number and entering an equivalent hexadecimal value for the first 12 bits of the configuration word. Default integer numbers for G#_CONFIG are found in Table 5-12.

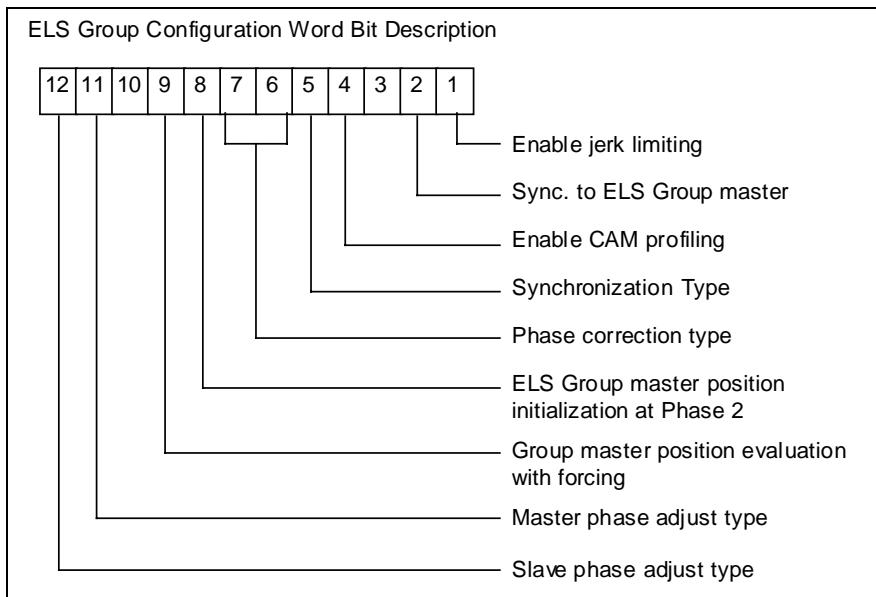


Fig. 5-2: ELS Group Configuration Word Description

Bit 1: Enable Jerk Limiting This bit is used to enable or disable jerk limiting for the ELS Group master command value.

0 = Disable jerk limiting (**default**)

1 = Enable jerk limiting

(*Updated in Phase 2 & Forcing*)

When enabled, step changes in acceleration are converted into 40ms ramps, independent of the SERCOS clock rate, providing a jerk limiting effect.

Bit 2: Sync. to ELS Group Master When the control is switched to manual mode, all ELS Groups are switched to local mode. In local mode, each ELS Group can be jogged independently. When switching back to automatic mode, the user can configure bit 2 using the following two options:

0 = Automatically switch back to the ELS Group master and perform a dynamic synchronization if necessary, see bits 5, 6 and 7 (**default**)

1 = Groups will stay in local mode and must be switched manually

(*Updated in Phase 2*)

Enable CAM Profiling Bit 4: This bit enables the lock on / lock off CAM profile state. For user CAM profiles to function, disable this feature.

0 = state machine enabled (**default**)

1 = state machine disabled

(Updated in Phase 2)

Modifications to the variable G#_H_USER can only be performed when the state machine is disabled. While disabled, the user can select a CAM profile for the ELS Group and modify the G#_H_USER factor. When enabled, the state machine uses as an H factor the values of G#_H_LOCKON, G#_H_LOCKOFF and G#_H_RUN. The G#_H_USER variable displays the current H factor being used for the lock on and lock off cam profiles.

Synchronization Type Bit 5: This bit is used to specify the type of synchronization that will be used when switching between ELS Group input masters.

0 = Dynamic synchronization (**default**)

1 = Immediate (*On the Fly when switching to an unused Virtual Master*)

(Updated in Phase 4)

When bit 5 is set to 1 and an ELS Group's input master is switched to an unused Virtual Master, this Virtual Master will adapt "On the Fly" to the current ELS Group master's position and velocity.

Phase Correction Type Bit 6-7: These bits are used to set the method that will be used for phase corrections during Dynamic Synchronization.

Bit 6	Bit 7	Description
0	0	Shortest path (default)
1	0	Positive direction if phase difference is greater than "G#_LOCK_WIN". Otherwise, use shortest path.
0	1	Negative direction if phase difference is greater than "G#_LOCK_WIN". Otherwise, use shortest path.
1	1	No phase correction (<i>only velocity synchronization is performed</i>)

(Updated in Phase 4)

Bit 8: This bit is used to reinitialize an ELS Groups output master position when the system is switched to Phase 2 (parameter mode) or powered down.

ELS Group Master Position Initialization at Phase 2

When an ELS Group's M/N or H factor has a value other than 1; for example 0.9, and the ELS Group has been moved, then the group's output master position cannot be calculated using the CAM equation.

The reason for this is as follows:

The control monitors and internally stores the ELS Group's current output position. For example, if after two revolutions of the input master (as illustrated in Fig. 5-3), the system is switched to Phase 2 or loses power; the ELS Group's output master position is stored. The user has the option to restart the ELS Group, to an initial position, by setting bit 8 to 0. This will recalculate the ELS Group's output master position using the CAM equation. Setting bit 8 to 1 allows the ELS Group's output master position to start from the stored position (old values) and continue; using the CAM equation, for consecutive revolutions of the ELS Group's input master.

$$[(\text{input master} * \text{M/N}) + \text{master offset}]H + \text{slave offset} = \text{Group output}$$

Group output position with a 0.9 M/N and no offsets

$$[(0^\circ * 0.9) + 0^\circ] * 1 + 0^\circ \Rightarrow 0^\circ \quad ;\text{initial position at start}$$

$$[(0^\circ * 0.9) + 0^\circ] * 1 + 0^\circ \Rightarrow 324^\circ \quad ;\text{after one revolution}$$

$$[(0^\circ * 0.9) + 0^\circ] * 1 + 0^\circ \Rightarrow 288^\circ \quad ;\text{after second revolution}$$

Fig. 5-3: CAM Equation Example

0 = Initialization with calculated value using the cam equation (**default**)

1 = Use old values

(*Updated in Phase 2 & Forcing*)

Bit 9: This bit is used to initialize an ELS Group's output position when switched to local mode (G#_CT_LOCAL).

Group Master Position Evaluation with Forcing

0 = Group master positions will be calculated using cam equation (**default**)

1 = Use old values

(*Updated in Phase 2 & Forcing*)

Bit 11: This bit sets the motion profile type for the active master.

Master Phase Adjust Type

0 = Trapezoidal profile using a velocity profile with dynamic synchronization acceleration/deceleration and additive velocity (**default**)

1 = Immediate – step function

(*Updated in Phase 4*)

Bit 12: This bit sets the motion profile type for the all slave axis associated with the ELS Group.

Slave Phase Adjust Type

0 = Trapezoidal profile using a velocity profile with dynamic synchronization acceleration/deceleration and additive velocity (**default**)

1 = Immediate – step function

(*Updated in Phase 4*)

Bits 13-32: These bits are not defined in GPP firmware.

ELS Runtime Utility

The ELS Runtime Utility in VisualMotion GPP is designed for modifying default program variables. These are variables that were initialized at compile time for the Virtual Masters icon (Assign Initial Values), ELS Master Assignment icon (including assigned ELS Group Masters). As an example, these variables consist of values for moving, stopping and jogging the Multiple Master components. To open this utility, select **On-Line Data** \Rightarrow **ELS** from VisualMotion Toolkit's main menu.

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

The programmer can modify the values initially compiled and downloaded to the control for...

- ELS Masters
- ELS Group Masters
- Virtual Masters

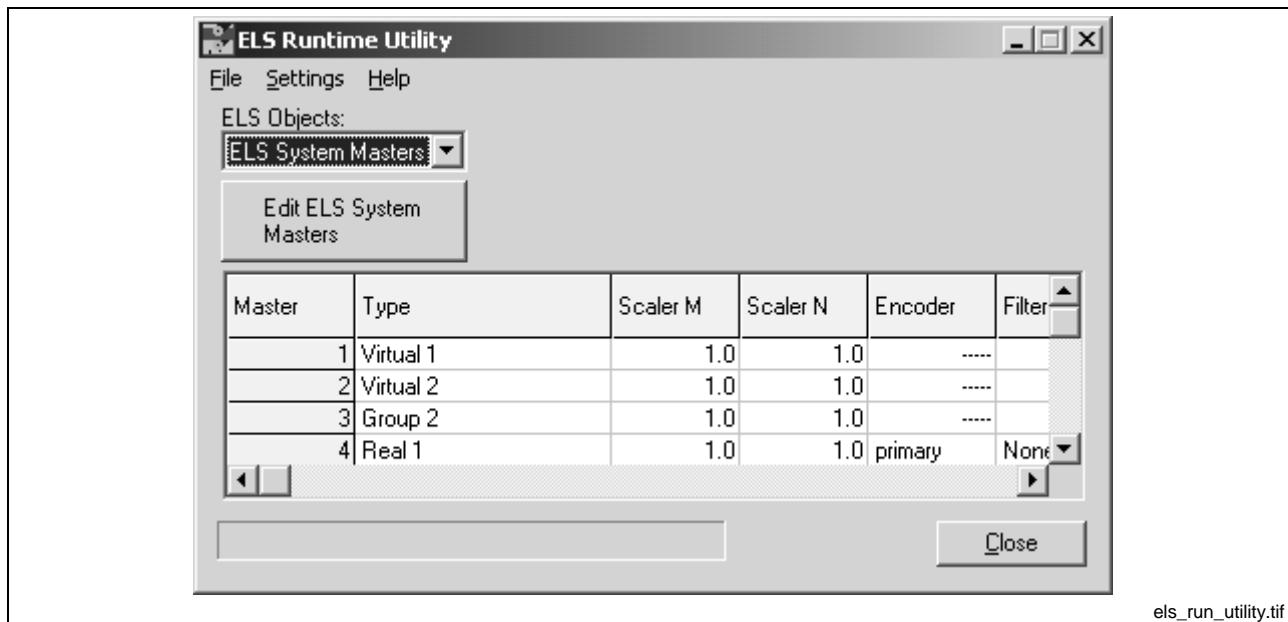


Fig. 5-4: ELS Runtime Utility

Modifying ELS Masters at Runtime

Select *Edit ELS Masters* from the ELS Objects drop-down list and click on the *Edit ELS Masters* button. The Edit ELS Masters window allows the user to modify the types of ELS Masters initially configured at compile time.

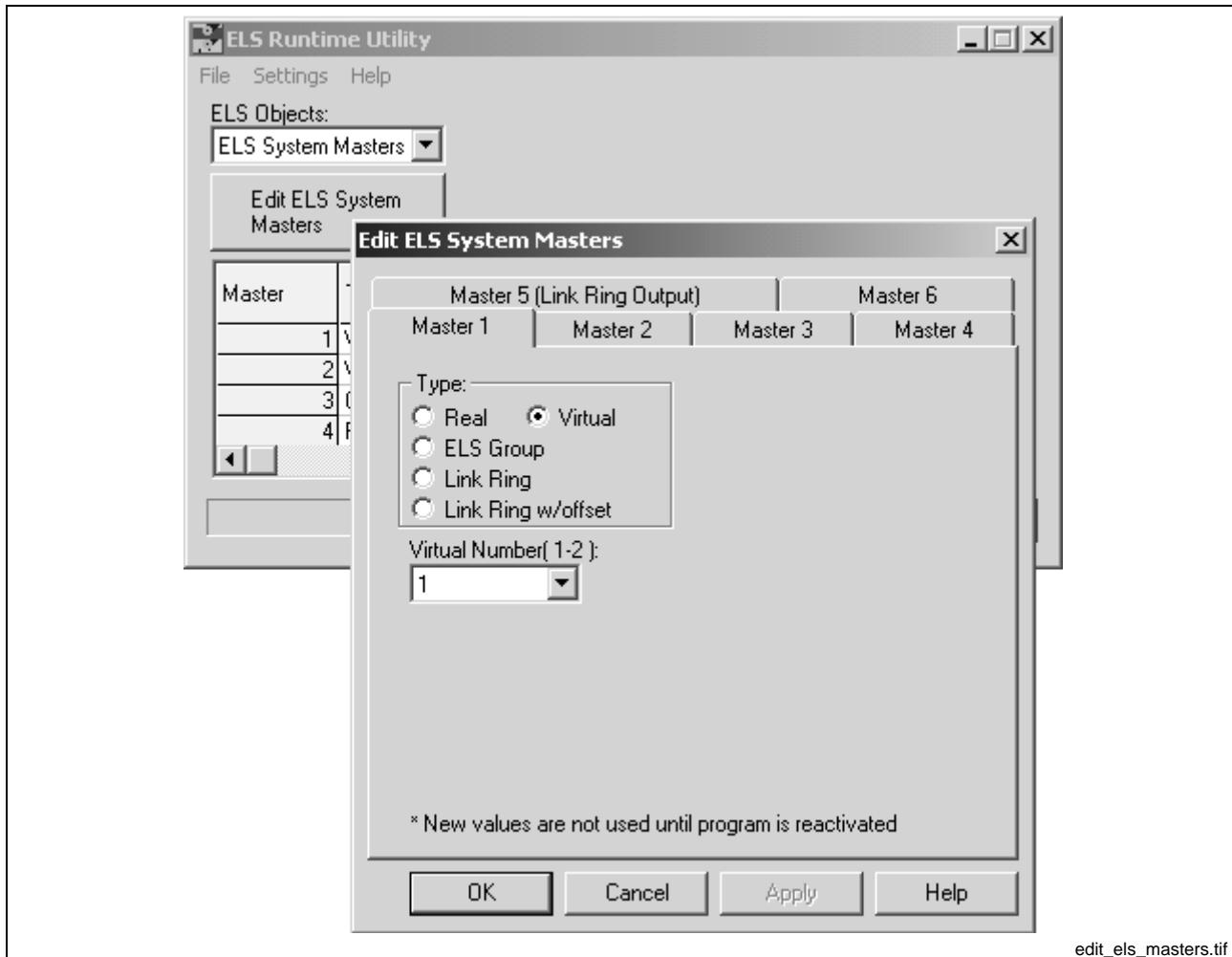


Fig. 5-5: Edit ELS Masters

Changes in the Edit ELS Masters window will modify the types of masters that will be available for configured ELS Groups.

Note: Modifications made to the ELS Masters will be active when the program is restarted. Compile time values in the ELS Master Assignment icon will not be affected. However, variable values can be transferred to another program by using the **Transfer Variables** selection under VisualMotion Toolkit's Commission menu.

Note: Refer to *Table 5-8: ELS Master Assignment Default Program Variables*, to determine what phase the system needs to be in for the variable to be updated.

Modifying ELS Group Masters at Runtime

Selecting ELS Group from the ELS Objects drop-down list and clicking on the Edit ELS Group # Variables button will open the window in Fig. 5-6. From this window, the variables for the ELS Group Master can be modified for functions such as...

- Main (master input, gear ratio, phase adjust, etc.)
- Synchronization
- Initialization controls
- Lock / Unlock Setup
- Stop and Jogging Control and
- Phase types

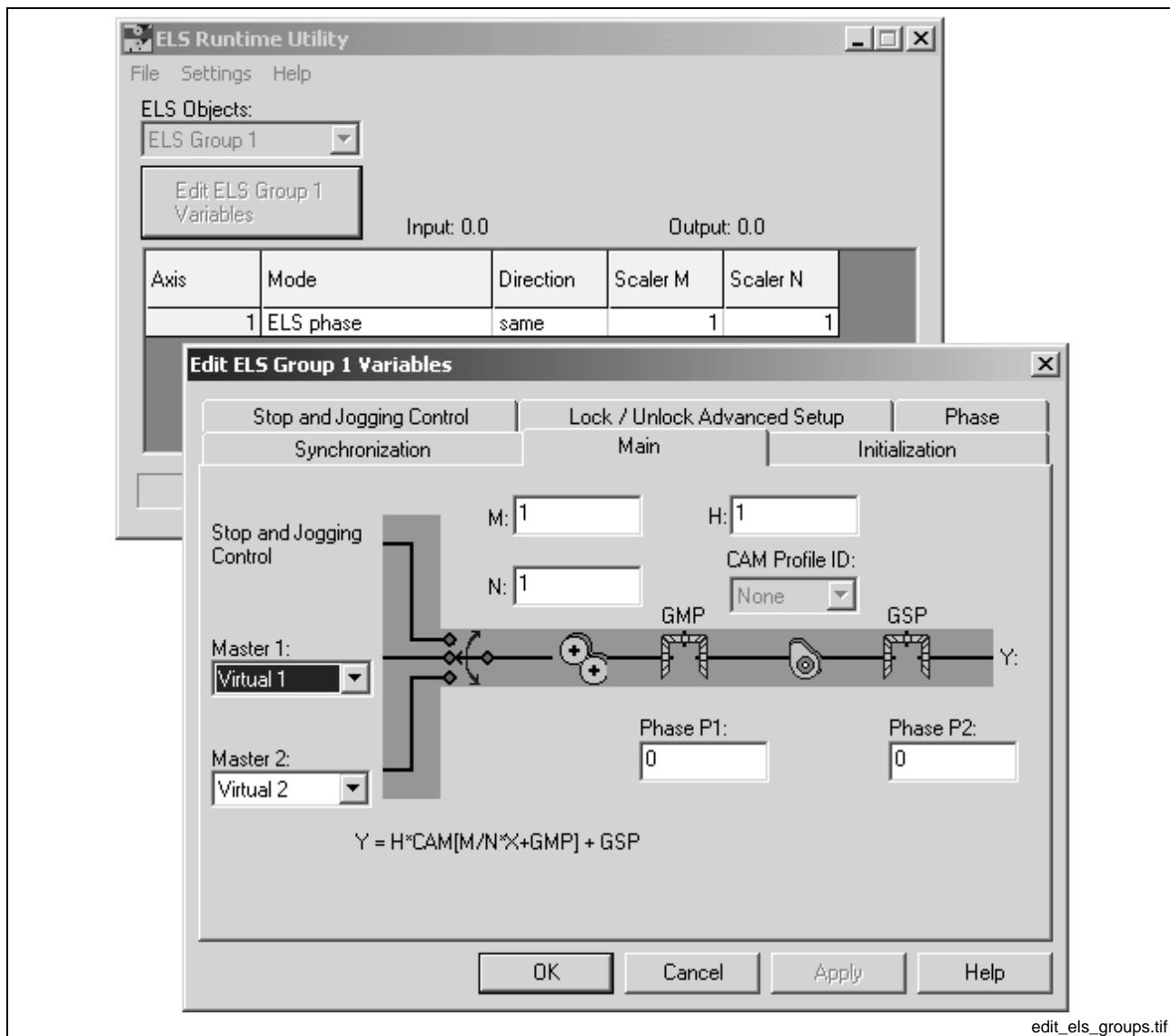


Fig. 5-6: Edit ELS Group # Variables

Note: Refer to *Table 5-12: ELS Group 1- 8 Default Program Variables* to determine what phase the system needs to be in for the variable to be updated.

Modifying Virtual Masters at Runtime

Virtual Master initial values can be modified using the ELS Runtime Utility. For example, changes to the initial Virtual Master velocity value will take effect on the fly. However, changes to the maximum value will be displayed but not take effect until the system is switched in and out of parameter mode (P2).

Note: Changes made to values in the ELS Runtime Utility are saved with the user program. However, if the original icon program is recompiled and downloaded to the control, the values found in the Virtual Master icon will be used. Values must be changed in the Virtual Master icon and downloaded to the control for changes to be available the next time the user program is compiled and downloaded.

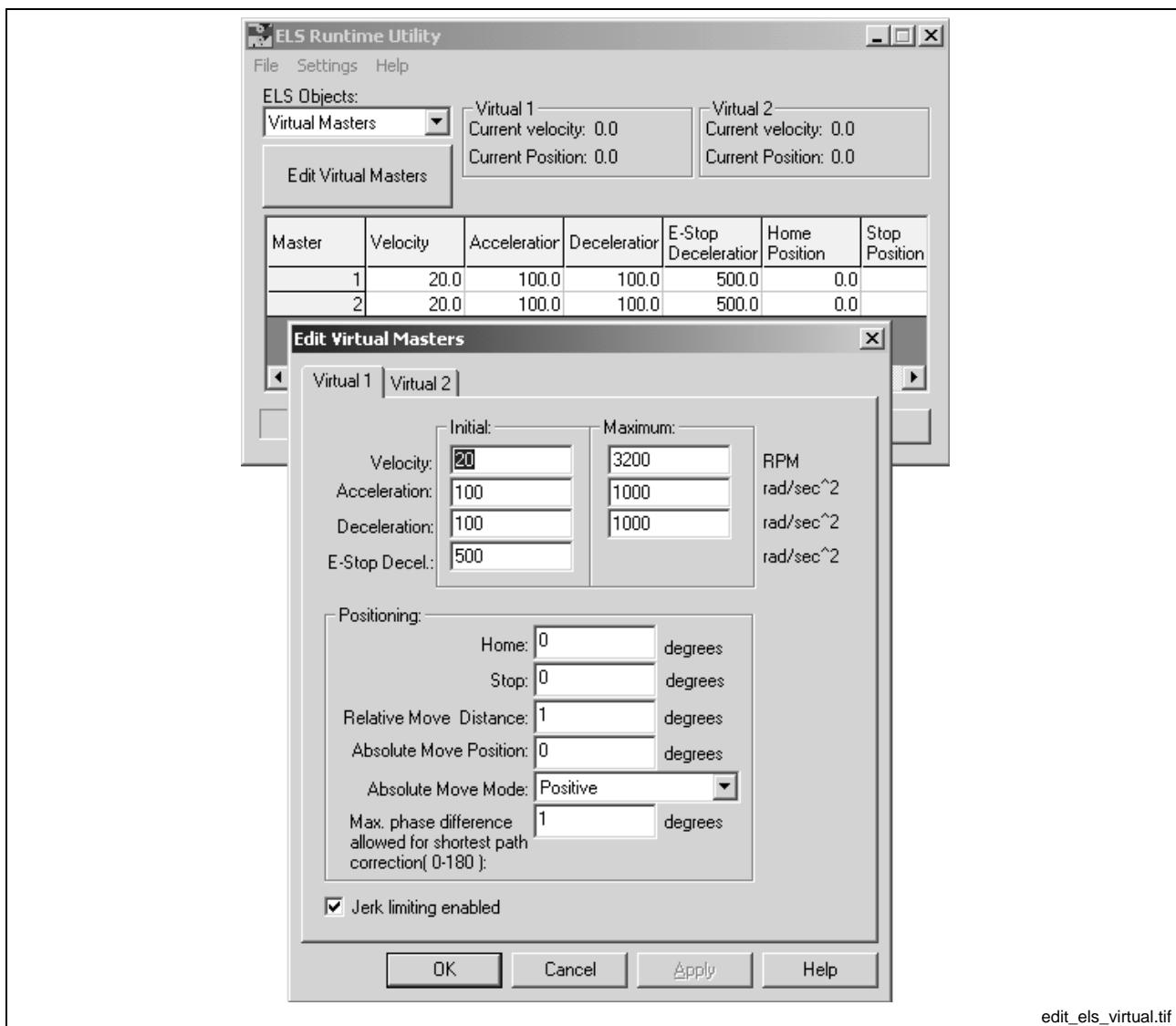


Fig. 5-7: Edit ELS Group # Variables

Note: Refer to *Table 5-7: Virtual Master 1 & 2 Default Program Variables* to determine what phase the system needs to be in for the variable to be updated.

5.2 Virtual Master

GPP 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. Refer to *Initialization of VisualMotion's Multiple Master Functionality* on page 5-1.

Virtual Master Compile Time Initialization

When the *Assign Initial Values ...* button is selected, each Virtual Master contains default initial and maximum values for operating and positioning as shown in Fig. 5-8. These values can be modified before compiling the program or at runtime using the *ELS Runtime Utility* under menu selection **Online-Data ⇒ ELS**.

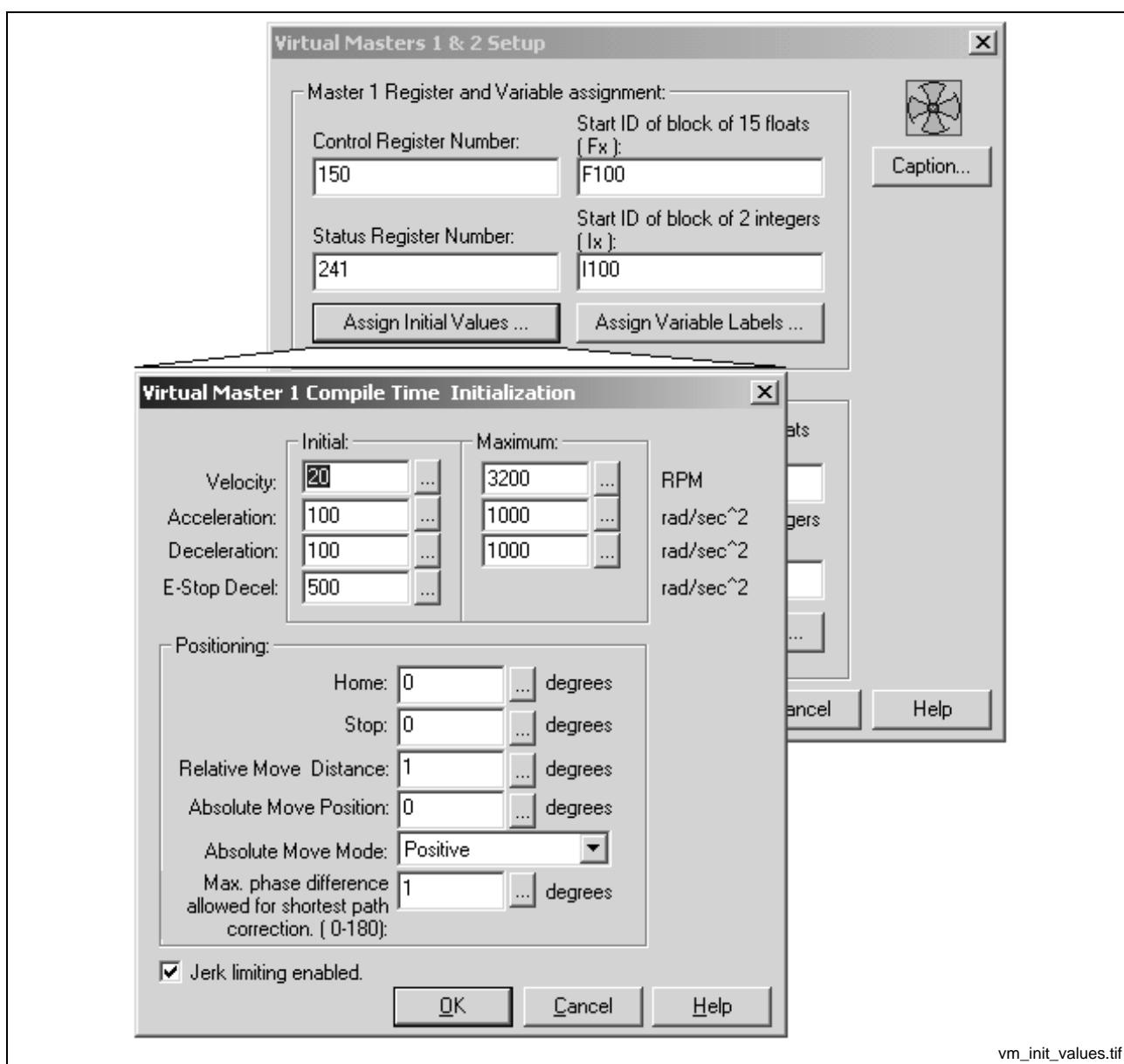


Fig. 5-8: Virtual Master Compile Time Initialization

Initial and Maximum Values

The initial and maximum values set for each Virtual Master in the *Compile Time Initialization* window in Fig. 5-8 are stored as program variables. Refer to Table 5-7: Virtual Master 1 & 2 Default Program Variables for details.

Velocity The *Initial Velocity* value defines a constant velocity that the Virtual Master will accelerate towards when set in motion.

The *Maximum Velocity* value defines the maximum velocity that can be achieved by the Virtual Master during runtime.

Note: The Virtual Master moves in a clockwise (positive) direction when a non-negative velocity value is used. A negative velocity value causes the Virtual Master to move in a counter clockwise (negative) direction.

The velocity that can be achieved by the drive (axes) following a Virtual Master is limited by the drive's Bipolar Velocity Limit. The following conditions exists:

- If the Virtual Master's maximum velocity is less than the drive's Bipolar Velocity Limit Value (S-0-0091), the drive is limited by the Virtual Master.
- If the Virtual Master's maximum velocity is greater than the drive's Bipolar Velocity Limit Value (S-0-0091), the drive will fault when S-0-0091 is exceeded.

Acceleration The *Initial Acceleration* value defines a constant acceleration that the Virtual Master will use to achieve a desired velocity.

The *Maximum Acceleration* value defines the maximum acceleration that can be achieved by the Virtual Master during runtime.

Deceleration The *Initial Deceleration* value defines a constant deceleration that the Virtual Master will use to decelerate the velocity.

The *Maximum Deceleration* value defines the maximum deceleration that can be achieved by the Virtual Master during runtime.

E-Stop Deceleration This value specifies the emergency stop deceleration for each Virtual Master.

Positioning

The positioning values set for each Virtual Master in the Compile Time Initialization window are stored as program variables. Refer to Table 5-7: Virtual Master 1 & 2 Default Program Variables for details.

Home The Virtual Master moves to this home position when the following Virtual Master control register bit is set:

Control Register Bit	State
Bit 2 (VM#_CT_HOME)	0 → 1

This value is written to program variable VM#_HOME_POS when the program is compiled.

Stop The Virtual Master moves to this known safe stop position when the control is switched from velocity mode to absolute positioning mode. This value is written to program variable `VM#_STOP_POS` when the program is compiled. The state of the following bits determine the mode of operation.

Control Register Bit	Velocity Mode	Absolute Position Mode
Bit 3 (<code>VM#_CT_GO</code>)	1	1
Bit 4 (<code>VM#_CT_VMODE</code>)	0 → 1 (velocity mode)	1 → 0 (moves to stop position)

Note: Once the Virtual Master reaches the stop position, the value in `VM#_STOP_POS` is written to program variable `VM#_CMD_ABS_POS`. The control is now operating in absolute positioning mode.

Relative Move Distance

The Virtual Master moves in increments of this value when the Virtual Master's control register bits are set as follows:

Control Register Bit	State
Bit 3 (<code>VM#_CT_GO</code>)	1
Bit 5 (<code>VM#_CT_RELMODE</code>)	1
Bit 6 (<code>VM#_CT_RELTRIG</code>)	0 → 1 (will move with every transition)

This value is written to program variable `VM#_REL_MOVE_DIST` when the program is compiled.

Absolute Move Position

This value is used to move the Virtual Master to an absolute position after the mode of operation is set to absolute position mode. This value is written to program variable `VM#_CMD_ABS_POS` when the program is compiled. The state of the following bits set the mode of operation to absolute position.

Control Register Bit	State
Bit 3 (<code>VM#_CT_GO</code>)	0 → 1
Bit 4 (<code>VM#_CT_VMODE</code>)	0

When bit 3 is set to 1, the Virtual Master moves to the value in program variable `VM#_CMD_ABS_POS`. Any change to this value, while in absolute position mode, will cause the Virtual Master to move to the new position.

Note: If the Virtual Master's mode of operation is switched from velocity to absolute position, the value in `VM#_CMD_ABS_POS` is replaced with the value in program variable `VM#_STOP_POS`.

Only positive values can be used for an absolute position move.

Absolute Move Mode This selection determines the direction that the Virtual Master will use when moving to the *Absolute Move Position* variable. This value is written to program variable *VM#_POS_MODE* when the program is compiled. The following choices are as follows:

- Positive (0 in *VM#_POS_MODE*)
- Negative (1 in *VM#_POS_MODE*)
- Shortest Path (2 in *VM#_POS_MODE*)

Max. phase difference allowed for shortest path correction

This value (0-180°) is used to create a "shortest path" positioning window for the Virtual Master's positive and negative move mode. When the *Absolute Move Mode* is set to positive or negative, the Virtual Master will move in the specified direction unless the new target position is inside the positioning window. If so, then shortest path will be used. Once the Virtual Master has moved to a new absolute position, a new positioning window is created around the new position. This feature is not available when the *Absolute Move Mode* is set to Shortest Path. Fig. 5-9 illustrates the function of this value.

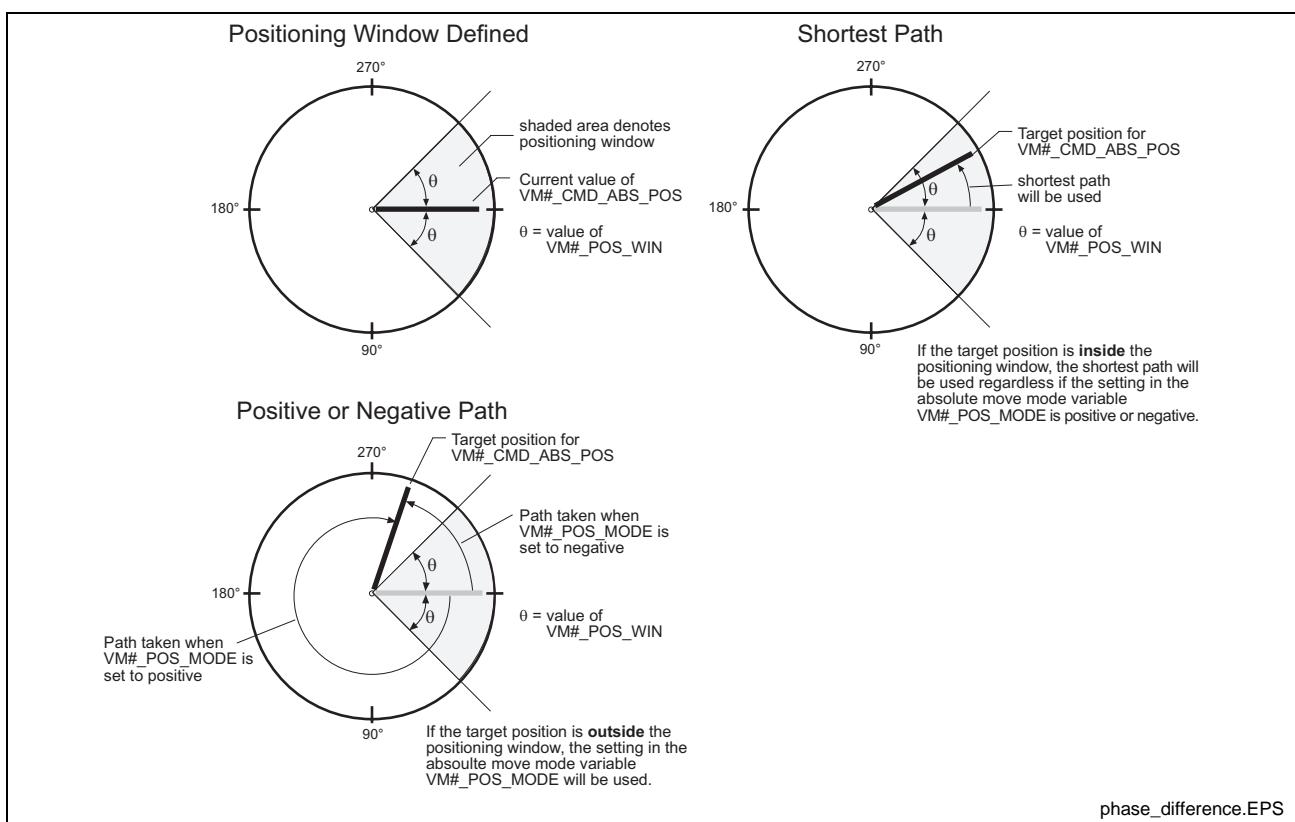


Fig. 5-9: Maximum Phase Difference

Virtual Master Modes of Operation

Velocity Mode

In Velocity mode, the Virtual Master moves at its commanded velocity. The rate of change in the commanded velocity (*VM#_CMD_VEL*) is performed using the defined acceleration/deceleration (*VM#_CMD_ACCEL*, ...*DECEL*) rate. In this mode, the Virtual Master can be either stopped, decelerating immediately, or at a predetermined stop position between 0 and 360 degrees. To stop the master at a desired stop position may take several revolutions (stop ramp) depending on the current velocity and programmed deceleration.

Note: When a master is in velocity mode, an integrator is engaged providing positional output so that all masters have a uniform signal type (position value with modulo of 360 degrees.)

Position Mode

In Position mode, the Virtual Master moves to a programmed relative or absolute position.

Relative Positioning

Travel distances can be greater than the modulo value for relative positioning moves of the Virtual Master.

Absolute Positioning

The maximum travel distance is +/- 180 degrees (shortest path) or 359.99 degrees (positive or negative direction) with absolute positioning.

5.3 Electronic Line Shafting (ELS) Master Assignment

In the ELS Master Assignment icon, six ELS Masters can be assigned. GPP supports the following three ELS Master types:

- Virtual Master
- Real Master
- ELS Group Master

An ELS Master is associated with a number from 1 to 6. This association creates, using software, an ELS Master connection box that uses the master's signal (commanded position) as an input to an ELS Group. Refer to Fig. 4-1: *Multiple Master Configuration Example* for an illustration of the ELS Master connection box.

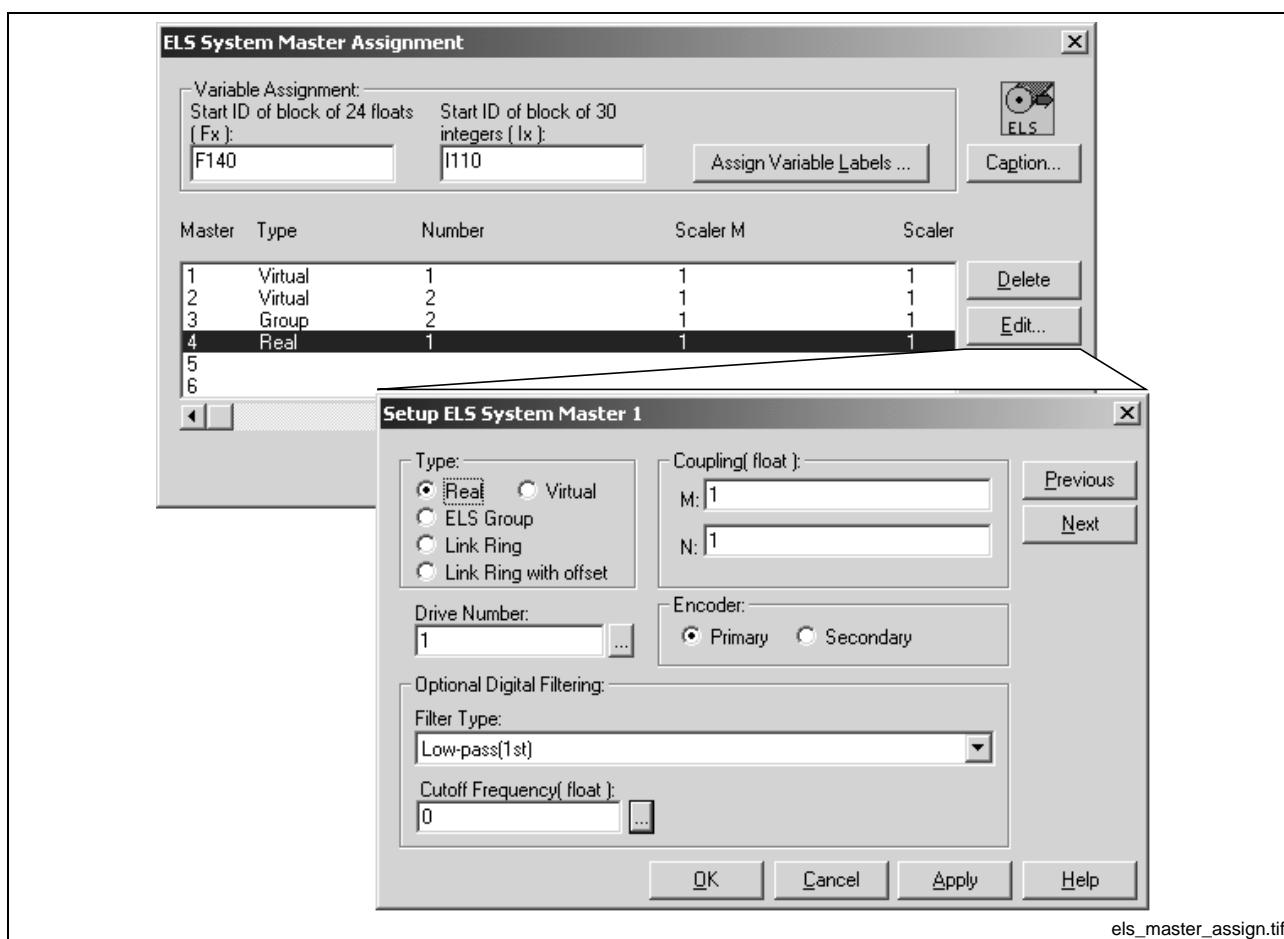


Fig. 5-10: ELS Master Assignment

Cascading ELS Groups

A maximum of four ELS Group Masters can be cascaded to other ELS Groups. When cascading ELS Groups, the lower numbered group's output should become the higher numbered group's input. For example, the output of group 1 is used as the input to group 2 and the output of group 2 is used as the input to group 3. This avoids a delay of one SERCOS cycle between cascading ELS Groups.

Note: An ELS Group Master's output cannot be fed back into the same ELS Group's input.

Note: All motion is associated with task A. Any motion associated with ELS Groups within the VisualMotion program will stop if tasks A stops.

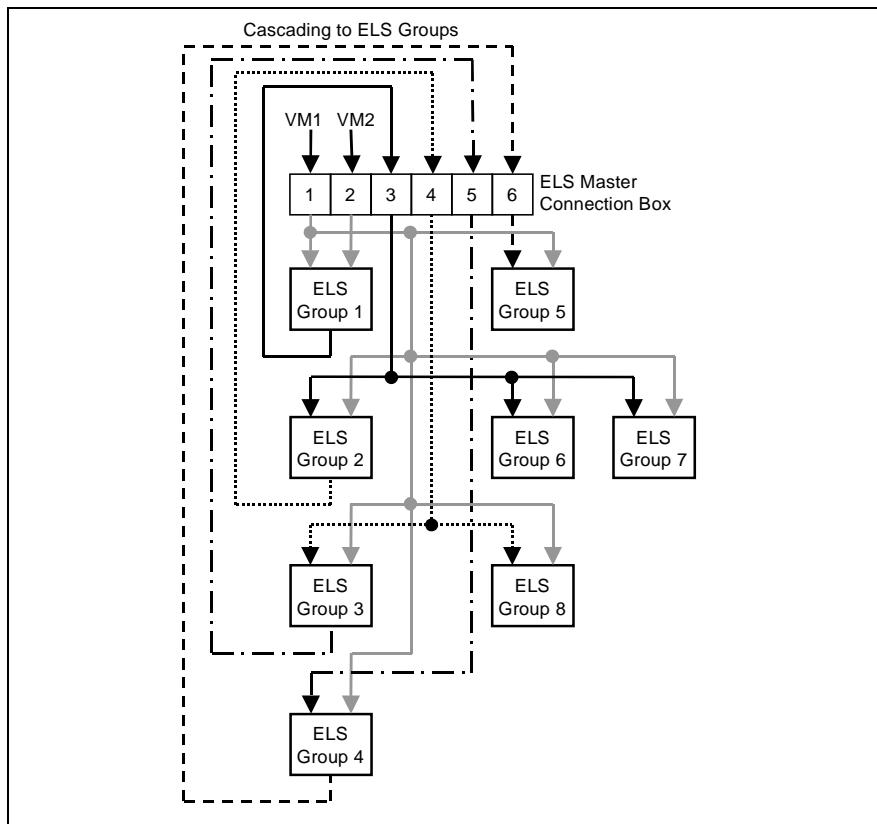


Fig. 5-11: Cascading an ELS Group Master Output

Assigning ELS Master Types

ELS Masters are assigned in the *ELS Master Assignment* icon by selecting a number and clicking on the *Edit...* button

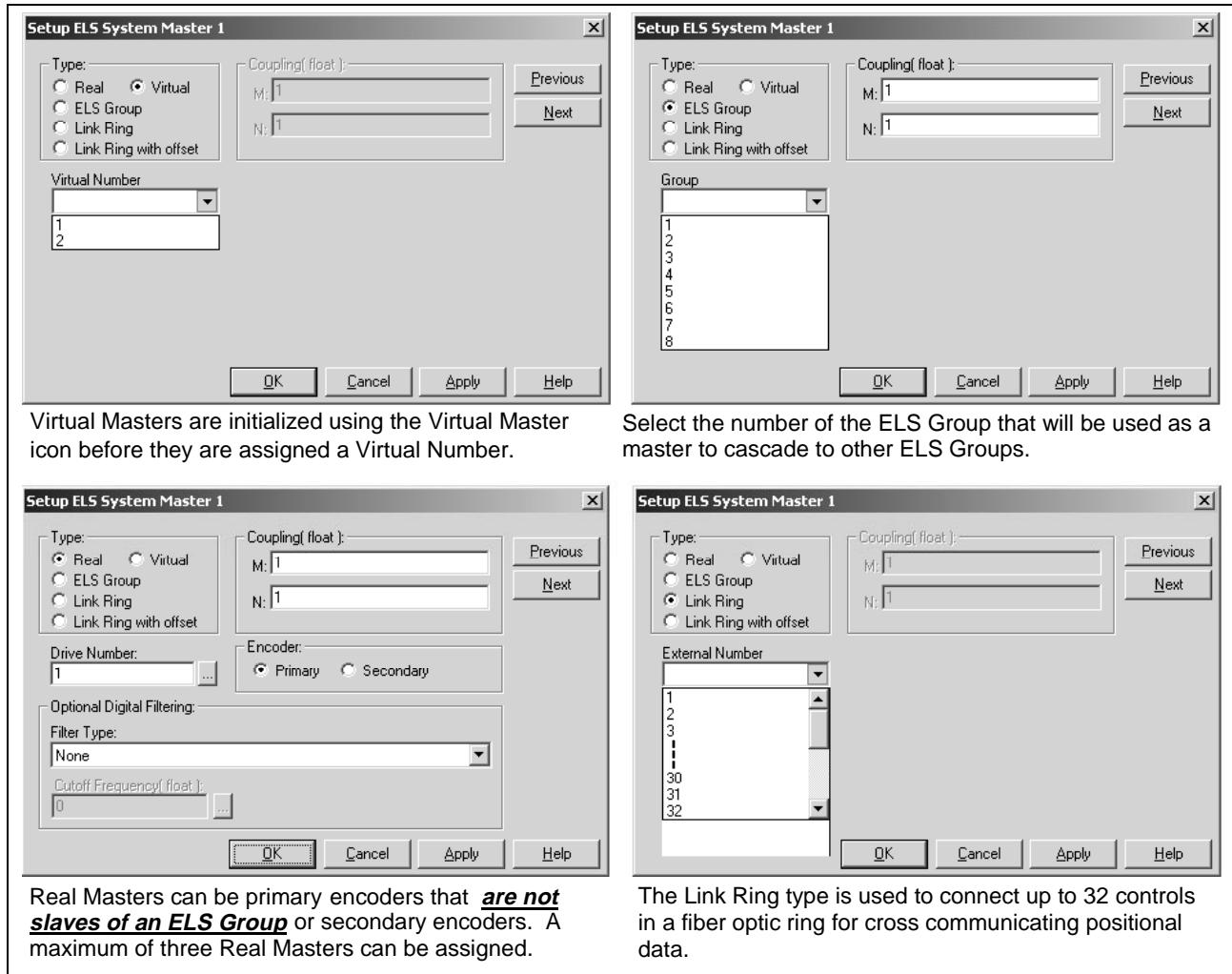


Fig. 5-12: Assigning ELS Master Types

ELS Master Connection Box (System Masters)

ELS Masters are associated to ELS Groups by means of a software connection box. To view the ELS Master connection box, start VisualMotion Toolkit and...

- Select **On-Line Data** ⇒ **ELS**. This will open the *ELS Runtime Utility* window.
- Select ELS Masters as your ELS object and click on the *Edit ELS Masters* button.

A maximum number of six masters are available in VisualMotion GPP firmware.

Note: An ELS Group output can be linked to one of the connection box numbers. ELS Groups are defined in the ELS Master icon. Up to four ELS Groups can cascade in this manner.

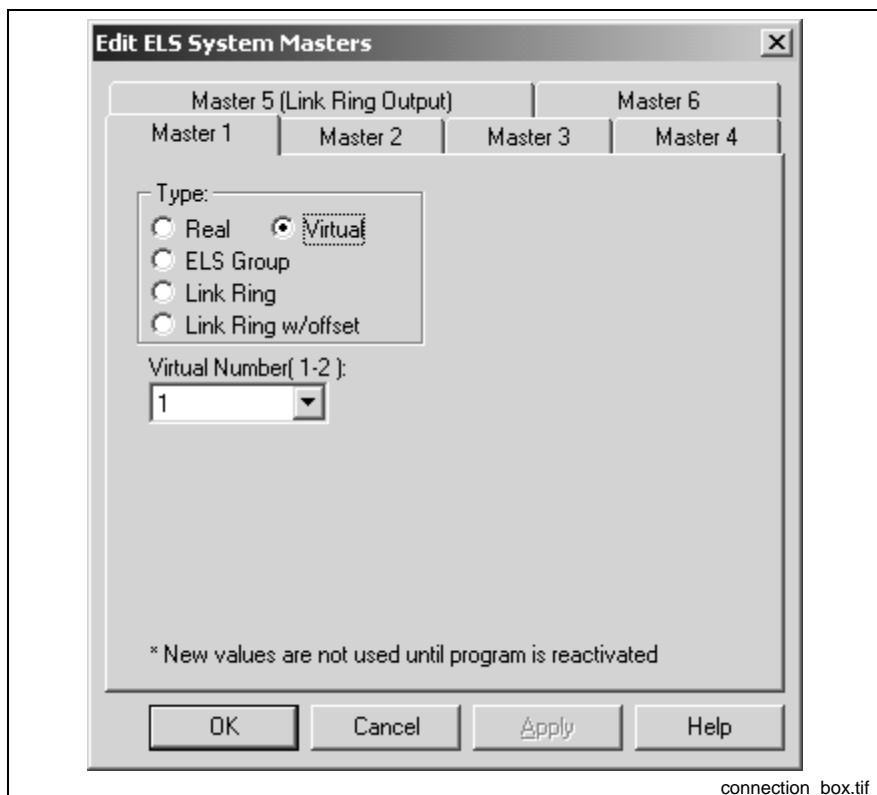


Fig. 5-13: ELS Master Connection Box

5.4 Electronic Line Shafting (ELS) Group



GPP supports a maximum of eight ELS Groups. The initialization of the ELS Group's registers and program variables is defined in the ELS Group Setup icon. Refer to *Initialization of VisualMotion's Multiple Master Functionality* on page 5-1.

An ELS Group's output provides a master position to its assigned ELS Slave axes. ELS Slave axes can only be assigned to an ELS Group at compile time. The ELS Group Master's output position is derived from the currently active group master input. The ELS Group's output signal can be modified using the following features:

- M/N gear ratio
- GMP (Group Master phase offset)
- CAM Profile with/without a Lock On / Lock Off feature using a 3 CAM profile
- GSP (Group Master phase offset)

An ELS Group can only have one active master at any given time determined by the group's control register input bit (G#_CT_MSTR_SEL.) To stop or move a group's master independent from the two input masters, every ELS Group has its own stop ramp and jog engine. To activate the group internal stop ramp, the group has to be switched into local mode (G#_CT_LOCAL.) When the VisualMotion program's task A is in manual mode, the groups are also switched into local mode. In local mode, after completion of the stop ramp, the group can be jogged with the group jog engine.

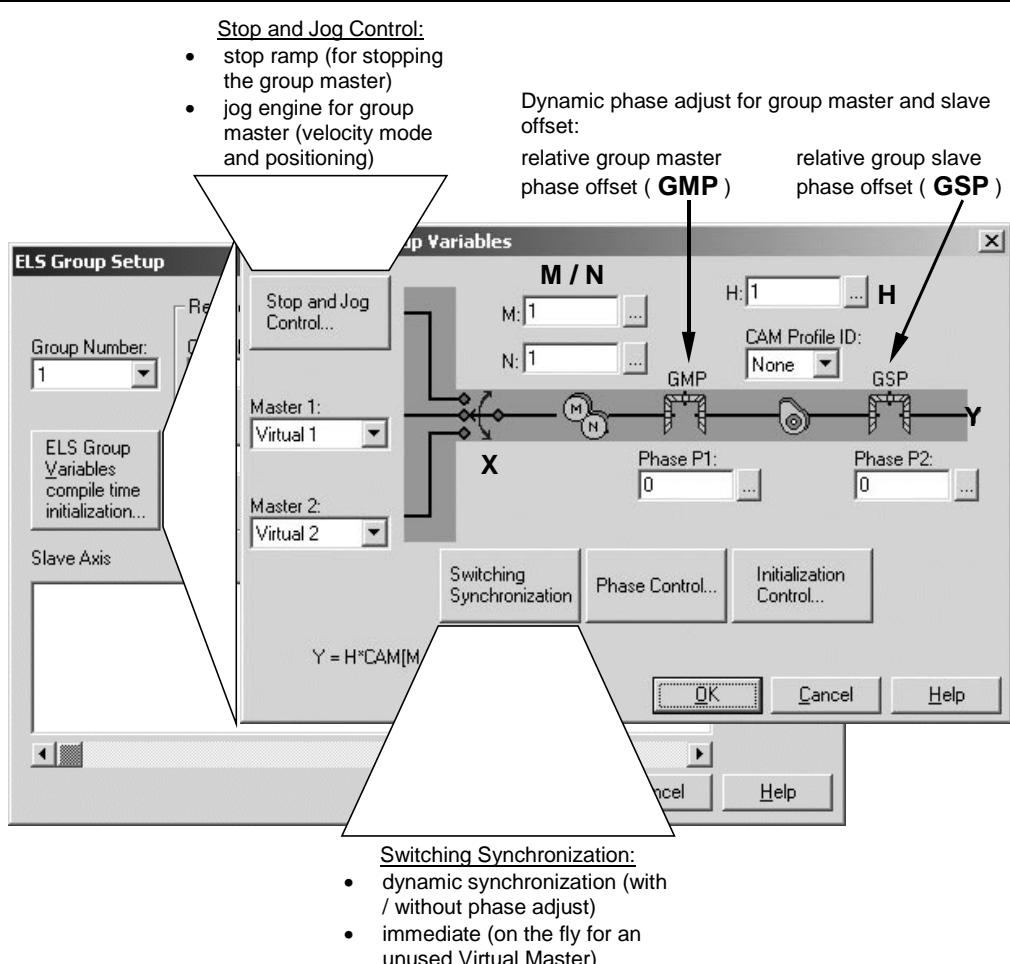


Fig. 5-14: Electronic Line Shafting Group

The ELS Group's active master input signal "X" is a condition of the equation in Fig. 5-15:

$$Y = H * CAM[M / N * X + GMP] + GSP$$

Fig. 5-15: ELS Group Output Equation

Where CAM[] is a control cam profile table or index cam profile, M and N is the current master input / output ratio, H is a cam profile scale factor and GMP and GSP are group master and slave relative phase adjusts. The signal Y drives the group's slave axes (group master position).

ELS Axis Configuration

ELS Slave axes are assigned to an ELS Group by clicking on the Add button and configuring the axis. A maximum of 32 ELS Slave axes (application dependent) can be assigned to an ELS Group. Each ELS axis can be configured with the selections in Fig. 5-16 for...

- Slave Axis number or label
- Synchronization Type
- Slave Direction
- Fine Adjust
- Turns: gear ratio (i.e., When using a gearbox.)

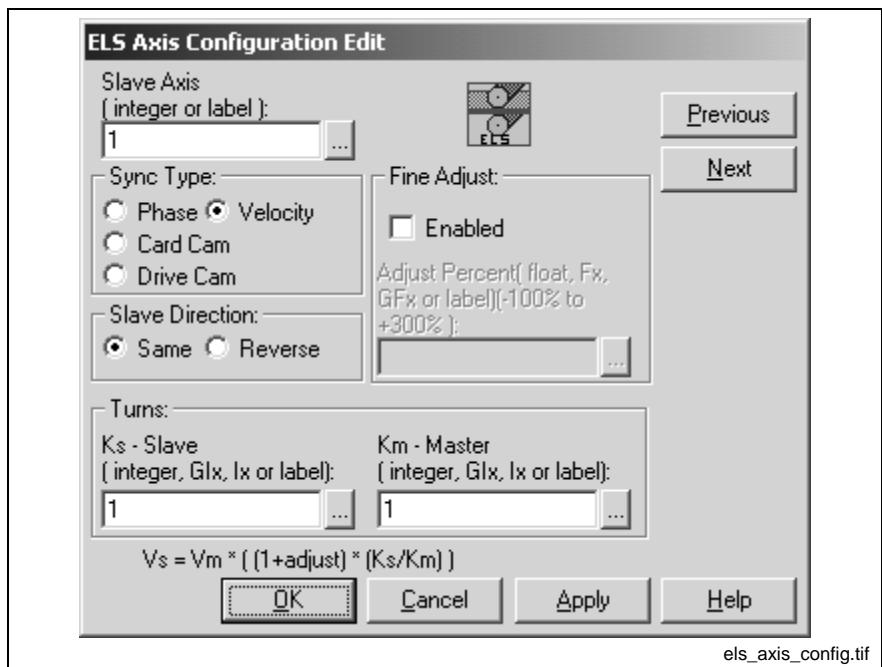


Fig. 5-16: ELS Axis Configuration

Stop and Jog Variables, Compile Time Setup

When an ELS Group is first configured, using the ELS Group icon, default values are supplied for...

- Stop Ramp deceleration
- Jog Controls for continuous, relative or absolute moves

These values are then saved, compiled and downloaded to the control. However, the user can make modifications to any of the values for stop and jogging control using the *ELS Runtime Utility*.

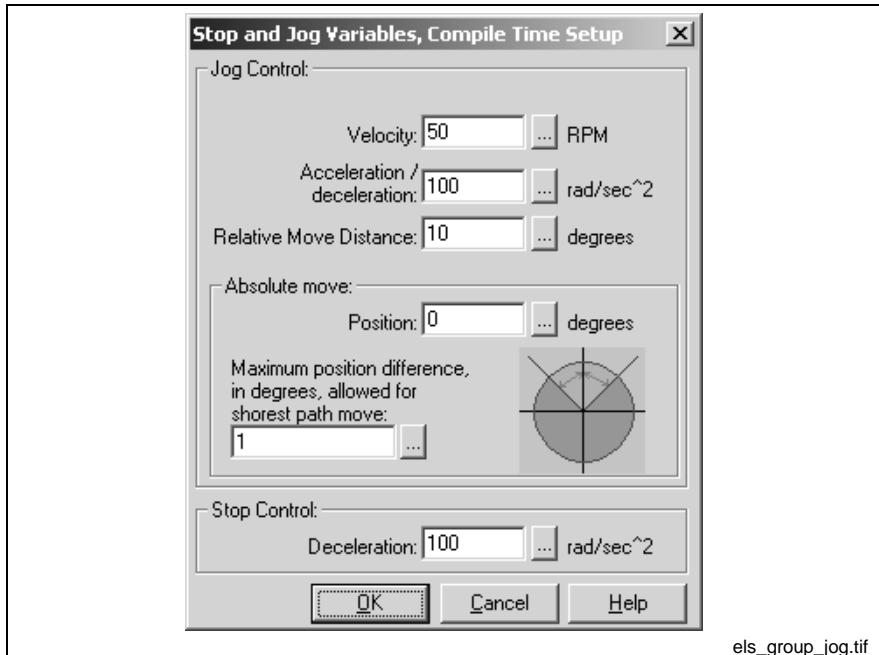


Fig. 5-17: Stop and Jog Variables

Jogging an ELS Group in Local Mode

Before an ELS Group can be jogged, the group must first be switched to local mode by toggling the G#_CT_LOCAL bit. The relative bits for jog controls are shaded within the ELS Group's control register table below.

Bit #	Bit Label	Comment
1	G#_CT_LOCK_OFF	0=lock, 1=starts unlock
2	G#_CT_M_REL_PH	0->1 triggers master relative phase adjust
3	G#_CT_S_REL_PH	0->1 triggers slave relative phase adjust
4	G#_CT_MSTR_SEL	0=master 1, 1=master 2
5	G#_CT_VAR_CLK	0->1 forcing
6	G#_CT_LOCAL	0->1 switch to local mode (stop ramp / jogging), 1->0 switch from local mode to selected group input master
7	G#_CT_JOG_INC	0=set continuous jog mode, 1=set incremental jog mode
8	G#_CT_JOG_ABS	0=set relative incremental mode, 1=set absolute mode (then bit 7 will be ignored)
9	G#_CT_JOG_PLUS	0->1 starts jog motion in positive direction
10	G#_CT_JOG_MINUS	0->1 starts jog motion in negative direction

Table 5-13: ELS Group Control Register

Setting bit 6 high in the group control register will bring the group velocity to zero using the deceleration rate in "G#_STOP_DECEL" variable. After bit 6 (G#_ST_LOCAL) in the group status register receives an acknowledgement that the group has stopped. The group can be jogged.

The following table describes the interaction of the jog bits. In all cases, jog motion ramps to zero when "G#_CT_JOG_PLUS" and "G#_CT_JOG_MINS" are cleared, or when one of these bits are set and then the other is set.

G#_CT_JOG_INC	G#_CT_JOG_ABS	G#_CT_JOG_PLUS = 0 → 1	G#_CT_JOG_MINS = 0 → 1
0	0	Continuous positive velocity.	Continuous negative velocity
1	0	Moves to positive incremental distance of "G#_JOG_INC" variable	Moves to negative incremental distance of "G#_JOG_INC" variable
0	1	Moves to positive absolute distance of "G#_JOG_ABS" variable	Moves to negative absolute distance of "G#_JOG_ABS" variable
1	1	Moves in positive direction * to "G#_JOG_ABS" variable position Status bit "G#_ST_JOG_POS" goes high (1) when in position	Moves in negative direction * to "G#_JOG_ABS" variable position Status bit "G#_ST_JOG_POS" goes high (1) when in position

* shortest path is used if distance is within the "G#_JOG_WIN" float variable

Table 5-14: Group Jogging Bit Status

Switching Synchronization between Group Input Masters

Switching between ELS Group input masters 1 and 2 or local mode can be performed using the following methods:

Immediate

The group's output position and velocity immediately switches, within one SERCOS cycle, between the two masters causing a step or bump in transition. When switching to an inactive Virtual Master, the position and velocity are adjusted "on the fly."

Dynamic Synchronization with or without Phase Adjust

The velocity and position difference will be compensated for by an internal ramp function synchronizing the transition between masters.

Local Mode

Switching to the group's local mode will immediately activate the group's stop ramp. This allows stopping the group's output even if both group master inputs are still moving. When the stop ramp has been completed, signaled by the "group local mode active bit," the group master can be jogged. This allows moving the group master independent from the group master inputs. Deactivation of the local mode will cause dynamic synchronization / immediate switching to the active group input master.

In the following example, ELS Group 1 with master input 1 is designed to synchronize to master input 2 when 2 is selected as the master input of ELS Group 1. In effect, master input 2 acquires the slave axes of ELS Group 1, thus replacing 1.

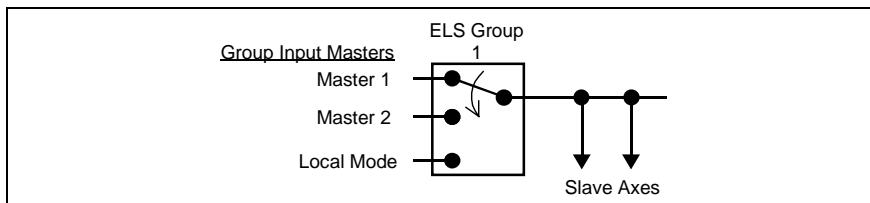


Fig. 5-18: Group Input Master Switching

Synchronization Setup

When the *Switching Synchronization* button is selected, the *Synchronization Setup* window in Fig. 5-19 displays the default values generated by the ELS Group's program variables.

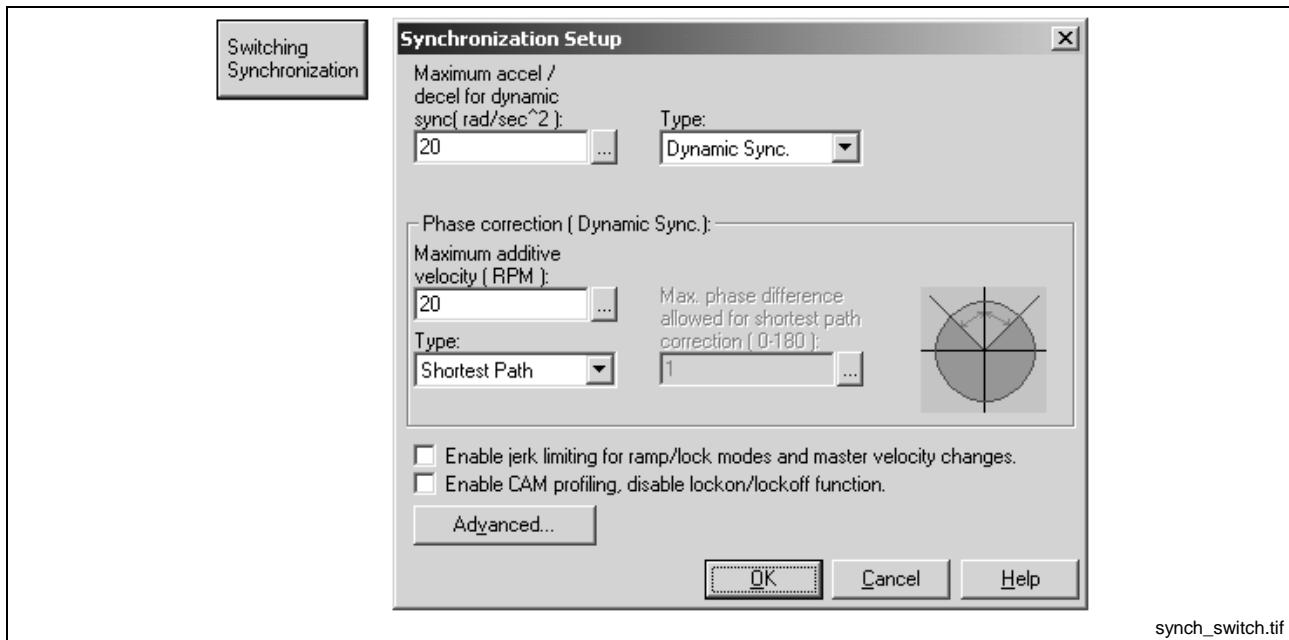


Fig. 5-19: Synchronization Setup

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 (Refer to page 5-33)
- Dynamic Synchronization (Refer to page 5-35)

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 ($\pm 0-180$ 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.

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](#) on page 5-36 for a functional description of this feature.

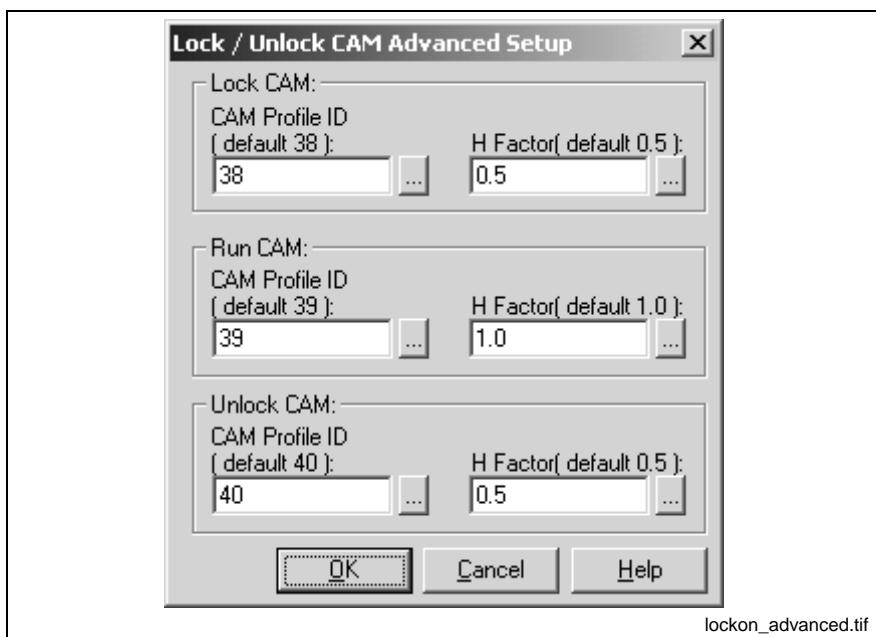


Fig. 5-20: Lock / Unlock CAM Advanced Setup

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.

Immediate Switching

This method allows for an immediate transition to a new input master. Switching takes place within one SERCOS cycle without regard to bumpless transitions. When switching between input masters, the group's velocity and position immediately change to match the target master. The bump is caused by the sudden change in velocity and correction of position difference between input masters. The following graph shows a typical immediate switch with a transitional bump.

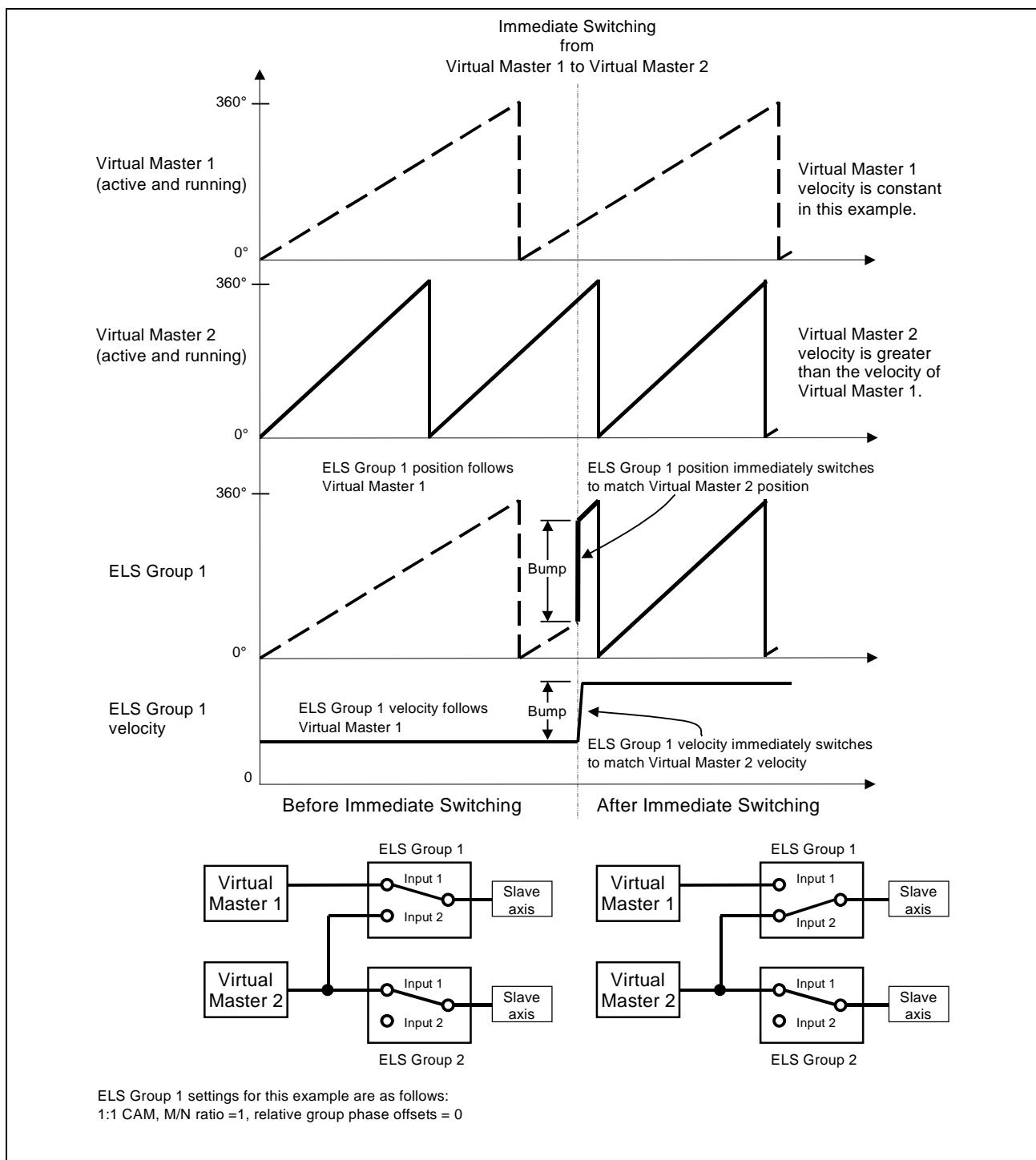


Fig. 5-21: Immediate Switching of ELS Group 1

Immediate Switching to an InActive Virtual Master (also known as “On the Fly” Switching)

A special case is switching to a Virtual Master, which is not the active master for any other ELS Group. The current master is sampled and its position and velocity is measured. These dynamic variables are used to initialize the new Virtual Master's target **on the fly**, allowing for an immediate and bumpless transition.

Note: It is not possible to use this method to synchronize to a Real Master since instantaneous changes in position, velocity, or acceleration would result in a drive fault.

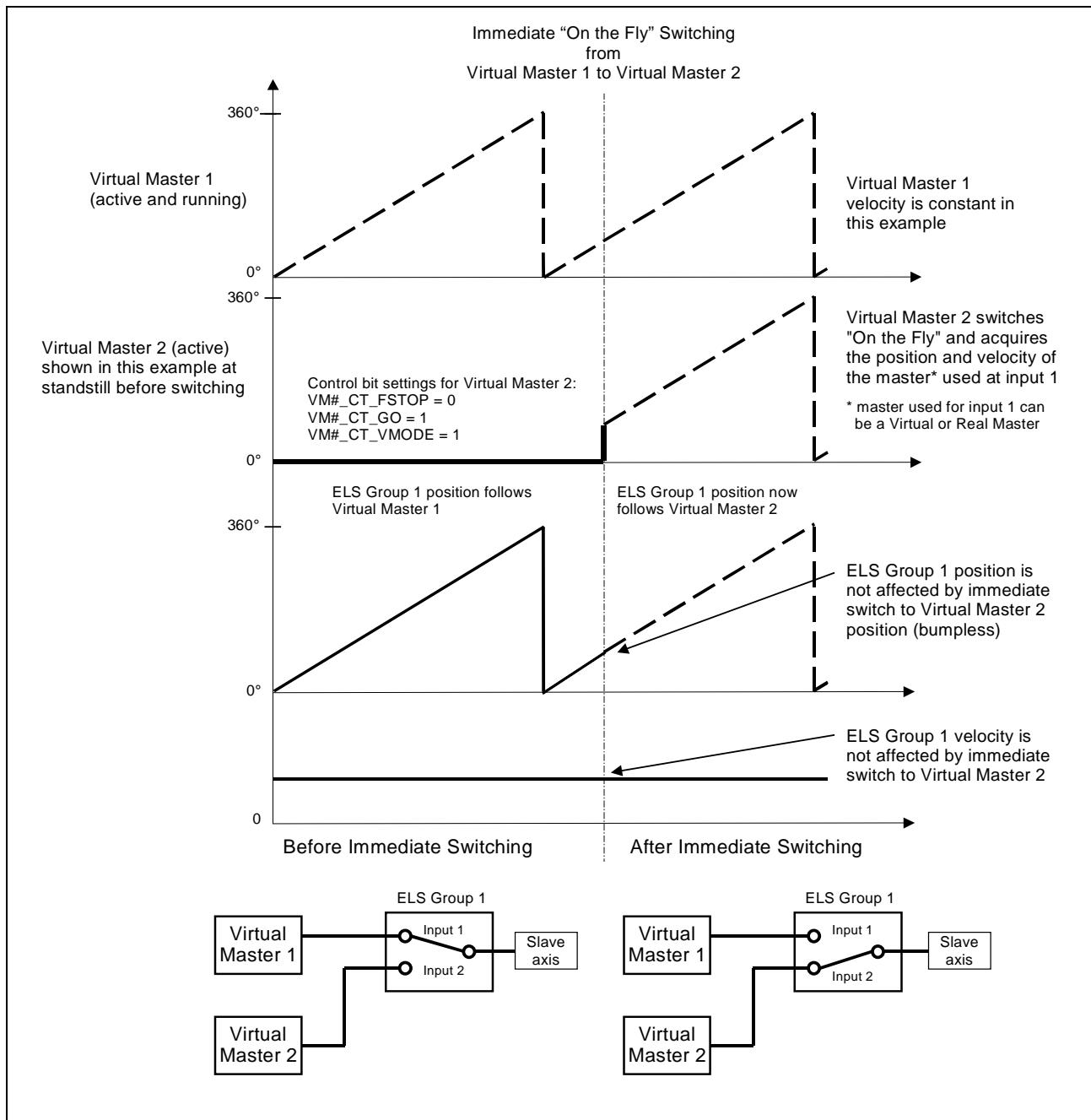


Fig. 5-22: Immediate “On the Fly“ Switching of ELS Group 1

Dynamic Synchronization

Typically, this general-purpose method synchronizes an ELS Group with a real, virtual or ELS Group Master to another real, virtual or ELS Group Master.

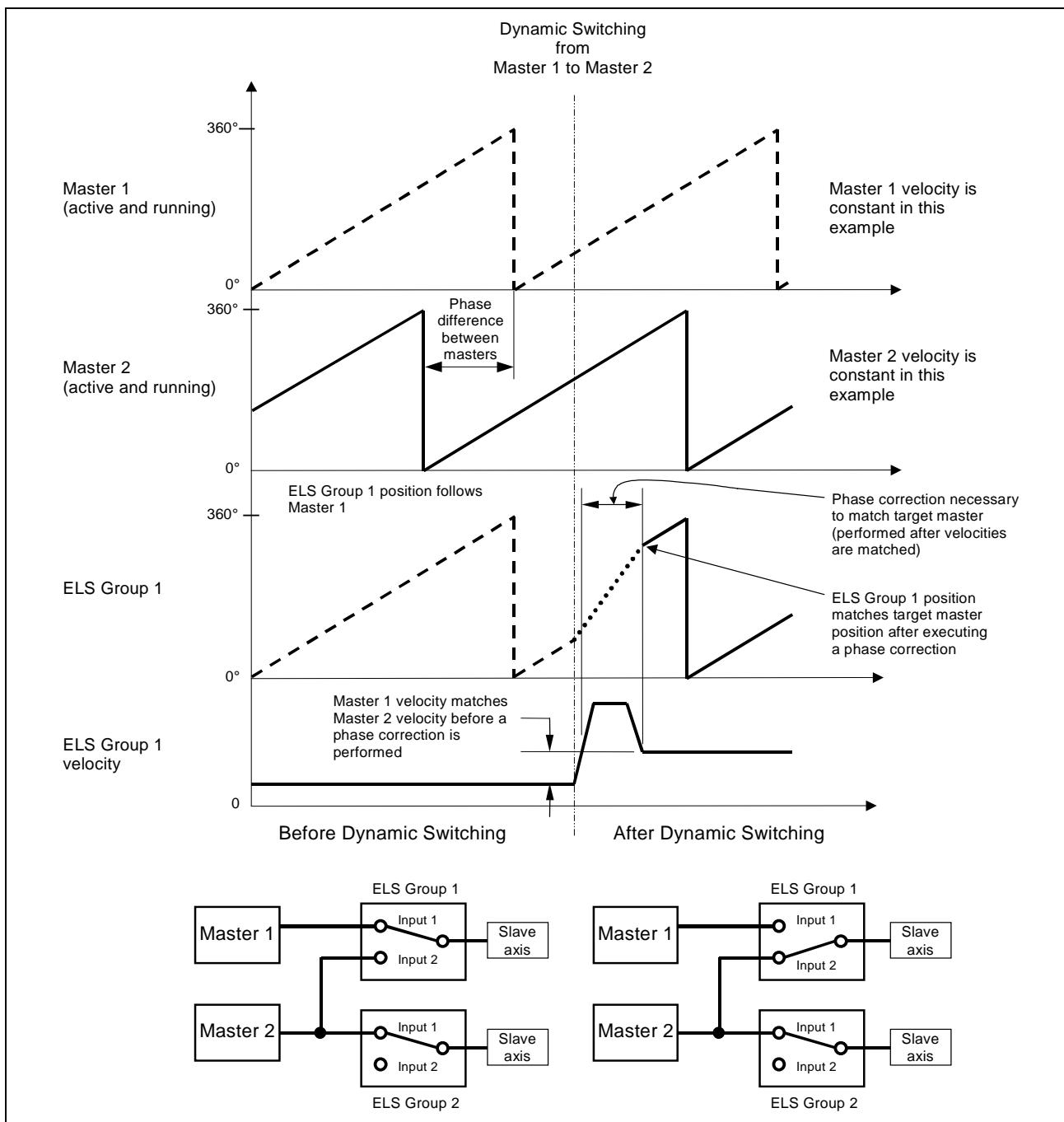


Fig. 5-23: Dynamic Switching of ELS Group 1

Dynamic synchronization allows for a rapid switch to a temporary (internal) velocity mode Virtual Master which then ramps and locks onto the new target master. The temporary master immediately disconnects the group's connection to the first master and allows for a smooth transition to the next master. Ramping is performed using the ELS Group synchronization's acceleration and velocity rates. These rates are also used for dynamic group master and slave phase corrections.

After ramping (velocity synchronization), a phase adjust compensates for any position error. Once the phase adjust is complete, the ELS Group's master is switched to the new input master. The temporary, internal master is dissolved after the transition is complete.

Note: Any attempt to switch masters again during dynamic synchronization is ignored, with the exception of switching to local mode (stop ramp.)

Synchronized “Lock On / Lock Off” of ELS Group Master

VisualMotion using GPP firmware incorporates the ability to stop and restart an ELS Group for one or more cycles of the group's input master. This function is performed using three cam profiles running synchronized with the group's input master. This synchronization between cam profiles and master input eliminates the need of any phase corrections. This function allows the program to stop a specific group's process while maintaining other groups running.

The following is an application example of the Lock On / Lock Off feature in GPP firmware. This example monitors the presence of a gap between products in a horizontal wrapper.

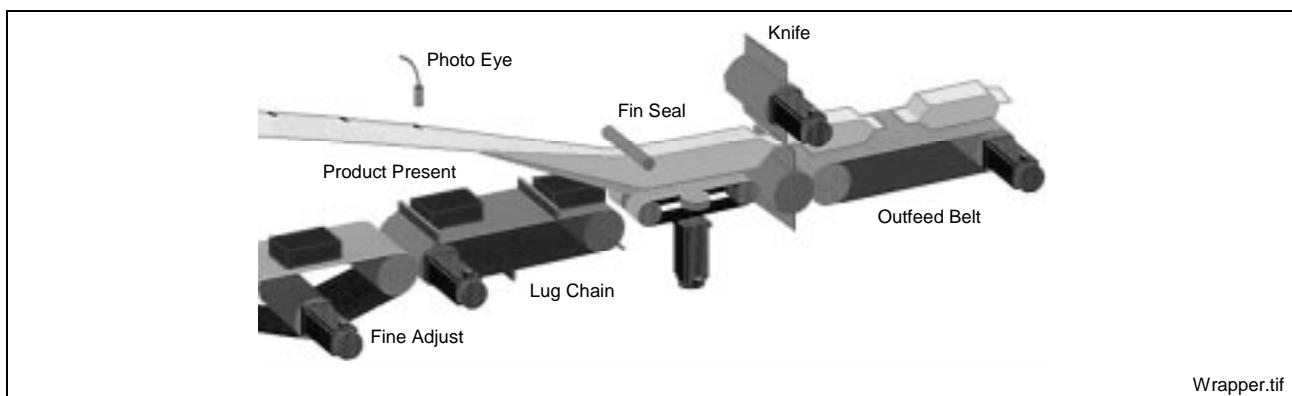


Fig. 5-24: Horizontal Form, Fill and Seal Wrapper

The Lock On / Lock Off feature in GPP is activated by the state condition of bit 1 (G#_CT_LOCK_OFF) in the ELS Group control register.

VisualMotion provides three default cam profiles for the Lock On / Lock Off feature. However, the user can create and download customized cam profiles using the CAM builder function in VisualMotion.

**One-to-One Cam Profile
“Synchronized to Master”**

This cam profile is a one-to-one profile and is normally active and synchronized to the master input unless the Lock On / Lock Off feature is not active. Under normal operating conditions, this cam profile is active and follows the group's active master input.

State of Lock On/ Lock Off bit

G#_CT_LOCK_OFF = 0

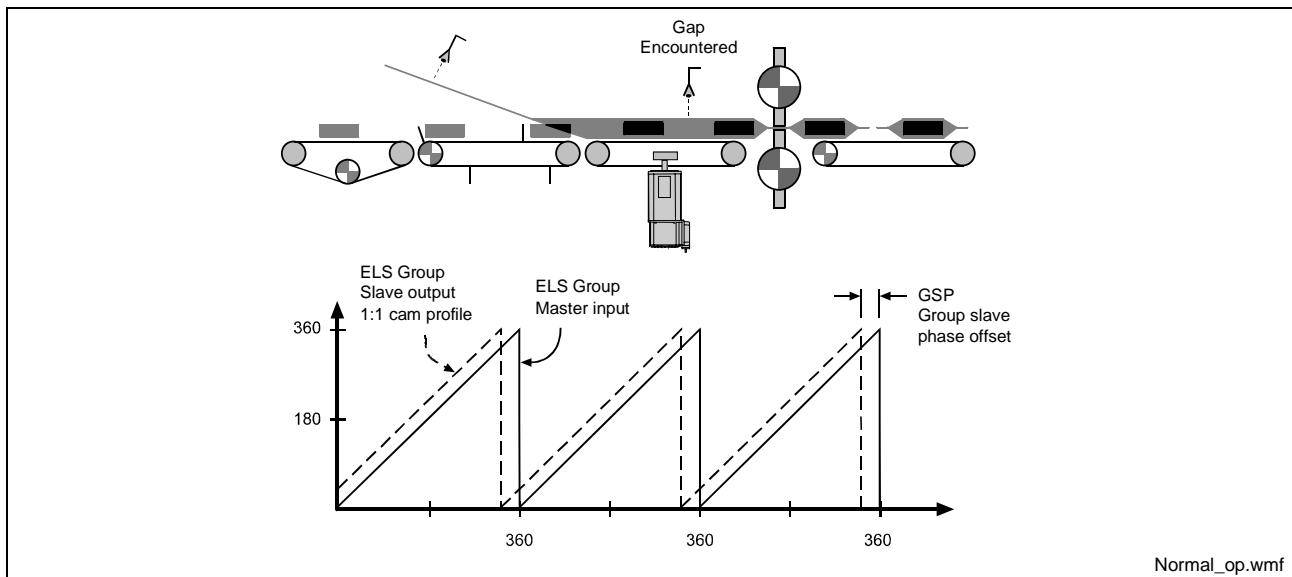


Fig. 5-25: Normal Operation of Wrapper Application

Lock Off The Lock Off cam profile decelerates to a stop over one cycle of the master. After this cycle, the group's velocity is stopped and will not restart unless the LOCK OFF bit is toggled.

Note: All motion to the ELS Group Master, as well as any cascading groups, will stop.

State of Lock On/ Lock Off bit

G#_CT_LOCK_OFF = **0 to 1**

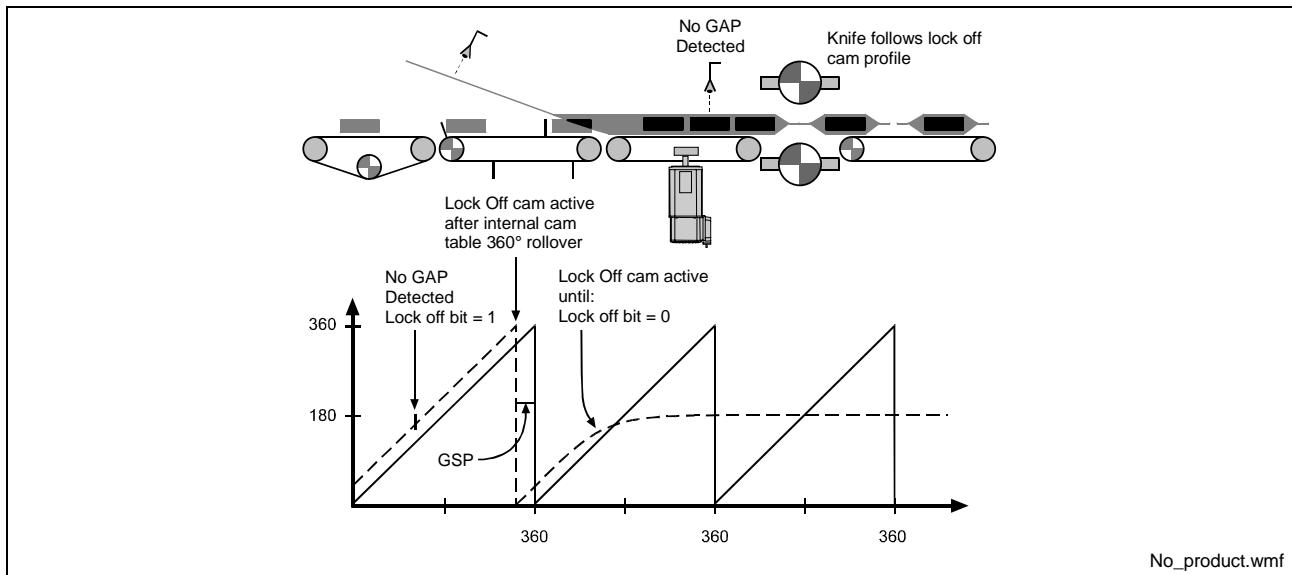


Fig. 5-26: Lock Off Cam Active, No Product - No Seal

Lock On The Lock On cam profile is active and accelerates from a stopped position to match the velocity of the master input over one cycle of the master (360 degrees). After this cycle, the velocity of the group matches that of the master.

State of Lock On/ Lock Off bit

G#_CT_LOCK_OFF = **1 to 0**

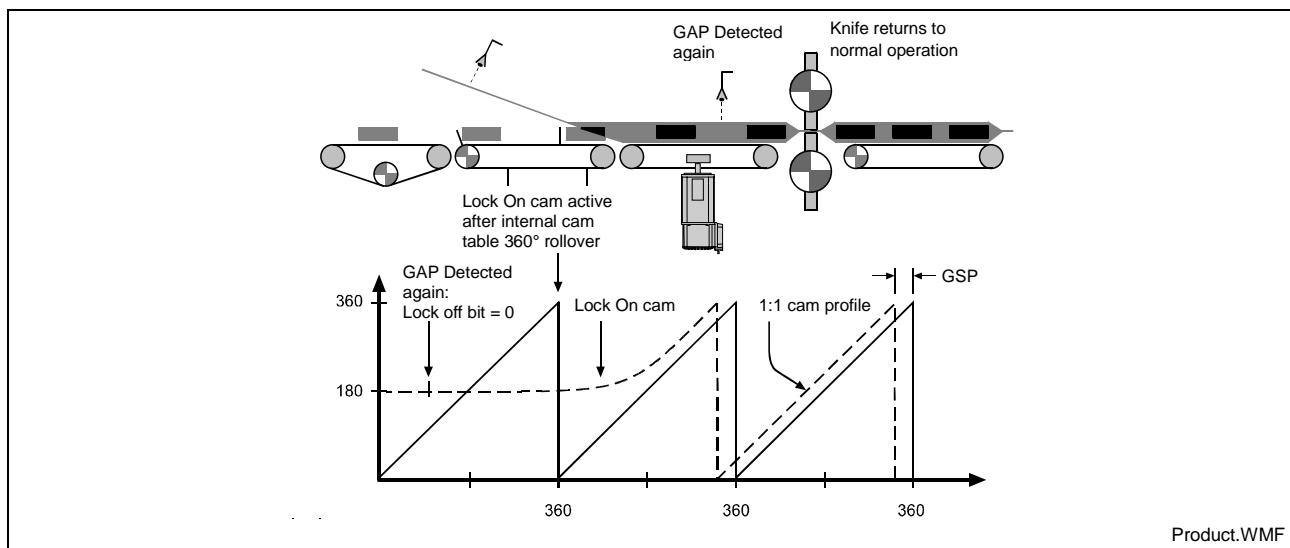


Fig. 5-27: Lock On Cam Active, Product is Present Once Again

6 Profibus Fieldbus Interface

6.1 General Information

Version Note:

Information in this document is based on VisualMotion Toolkit software version 08VRS and PPC-R firmware version GPP08VRS.

Note: For fieldbus hardware information, refer to the *VisualMotion 8 Project Planning Manual*.

PPC-R System Description with a Fieldbus

The PPC-R can operate on a serial fieldbus interface (network) by means of a fieldbus expansion card that communicates with the PPC-R via dual-port RAM. The function of the fieldbus card is similar to that of a network card in a PC: it allows communication with other devices on the network.

In Fig. 6-1, a commonly described fieldbus interface is pictured:

- **Fieldbus Master** - PLC fieldbus interface
- **Fieldbus Slave** - PPC-R fieldbus interface

In this document, we will refer to the PLC as the **fieldbus master** and the PPC-R as the **fieldbus slave**.

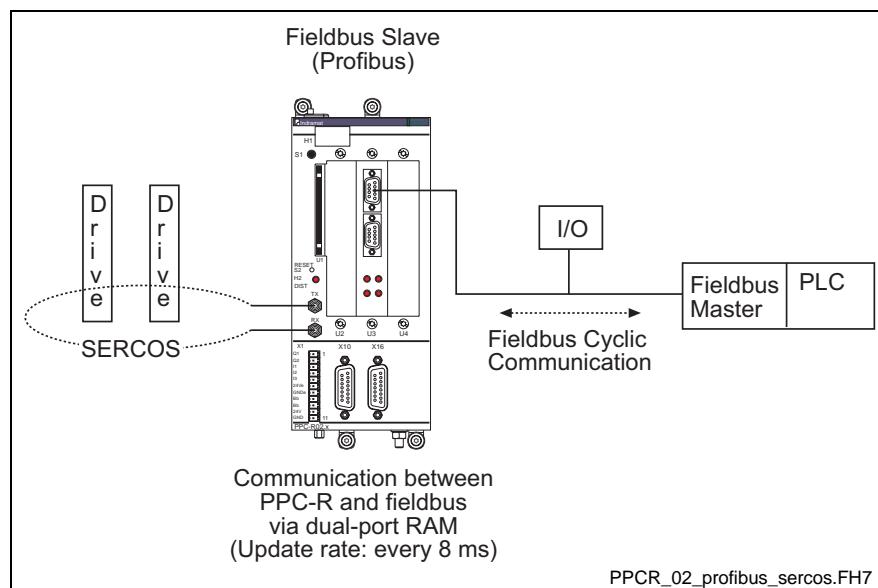


Fig. 6-1: Sample Master/Slave Setup with Fieldbus Card

With the PPC-R, the fieldbus card can be used **only** as a **slave** card in a master/slave setup.

The VisualMotion Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to set up fieldbus configuration and data mapping.

Data Transfer Direction (Output vs. Input)

In the VisualMotion Fieldbus Mapper, output and input are always described with respect to the fieldbus master. The definitions for output and input follow:

output: the communication from the PLC to the PPC-R (i.e. from the fieldbus master to the fieldbus slave).

Synonyms for this type of communication: **send** or **write** data.

input: the communication from the PPC-R to the PLC (i.e. from the fieldbus slave to the fieldbus master).

Synonyms for this type of communication: **receive** or **read** data.

Fieldbus Data Channel Descriptions

The Indramat Profibus fieldbus interface card for the PPC-R supports the cyclic (DP) channel, which is made up of the following two parts:

- **Real-Time Channel** (for **single** and **multiplex** channels)
- **Parameter Channel** (for systems requiring non-cyclic transmissions)

Cyclic (DP) Channel

Cyclic data is user-defined. It is stored in two ordered lists (C-0-2600 for input data, C-0-2601 for output data) and transmitted serially over the bus.

The cyclic data channel is limited to 32 input words and 32 output words. PPC-R data types consume these words in either one-word (or 16-bit) groups for PPC-R registers or two-word (or 32-bit) groups for all other data types.

The PPC-R mapping list is scanned every 8 ms and data is sent and received to/from the fieldbus slave board's dual port RAM.

The cyclic data channel can be made up of any combination of the following data types:

- Real-Time Channel
 - Single Channel
 - Multiplex Channel
- Parameter Channel

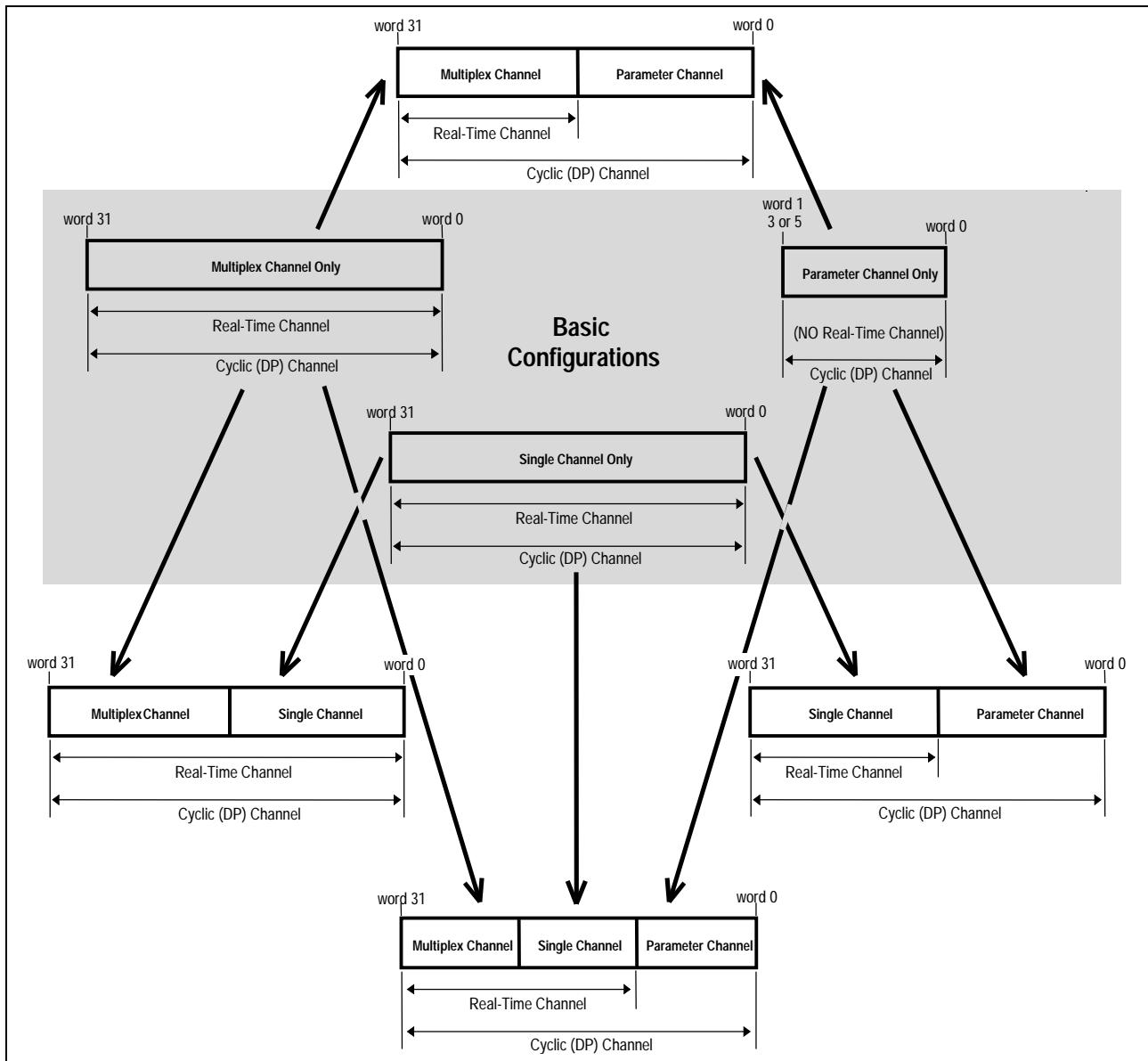


Fig. 6-2: Configuration Options for the Cyclic Data Channel

The Real-Time Channel

In the real-time channel, data is updated cyclically between the fieldbus master and slave. This channel contains two possible data types: **single** and **multiplex**.

Cyclic Data: Types and Sizes

The following table outlines the PPC-R data types that can be transmitted via the cyclic channel and the amount of space (in 16-bit data words) that each data type consumes.

Note:	The cyclic data mapping lists supports only 16- and 32-bit data of the following types for reading and writing:
	- Integer
	- Float
	- Binary (used in control parameters)
	- Hex (used in control parameters)
	For all other data types (e.g. diagnostic messages - "strings"), use the Parameter Channel.

PPC-R Data Type	Data Size (in 16-Bit Words)
Register	1
Program Integer (currently active program ONLY *)	2
Program Float (currently active program ONLY *)	2
Global Integer	2
Global Float	2
Card Parameter	2
Axis Parameter	2
Task Parameter	2
Note: Drive parameters "S" or "P" cannot be transmitted cyclically because of the inherent delay of parameter access over the SERCOS service channel. See " Parameter Channel. " However, if a drive parameter is mapped to an Axis Parameter, that Axis parameter could be used in cyclic data (see description of Axis Parameters 180-196 in the <i>VisualMotion Functional Description</i>).	
* Important Note: Integers and floats are shown only for the currently active program. Each time you activate a new program, the fieldbus reads/writes to the newly-activated program.	

Table 6-1: PPC-R Cyclic Data Types and Sizes

Single Data Types

Single data types are mapped directly in the cyclic mapping ordered lists (C-0-2600, C-0-2601). The data types are updated every 8 ms via dual-port RAM.

Multiplex Data Types (Cyclic Data Channel)

In some multi-axis applications, 32 words of cyclic data transfer are not sufficient to meet the requirement of the application.

When insufficient data transfer space is available, multiplex data can be set up within the cyclic channel. One multiplex container acts as a placeholder for multiple possible PPC-R data types (all of the same word size). The currently transmitted PPC-R data type is based on an index value placed in a multiplex control or status word attached to the end of the cyclic list. Depending on the index specified by the master, the multiplex channel permits a different set of data within the cyclic channel to be transferred as current real-time data. Multiplex containers can be added to the input and output lists separately and the input and output indexes can be designated separately (in the control and status words).

Note: Using the multiplex channel reduces the maximum number of usable words for storing control data to 31. The 32nd word (or last used word, if fewer than 31 words) is used as the multiplex entry control/status word.

Note: When using VisualMotion 8 with GPP 7 firmware, a maximum of 15 multiplex containers and a maximum of 180 mapping items can be transmitted in the input or output list. This limitation of mapping objects means that you cannot multiplex all 15 containers with all 32 available indexes (=480 items). For VisualMotion 8 with GPP 8 firmware, there is no limitation for multiplexing (each of the first 31 words may be multiplexed with up to 32 indexes).

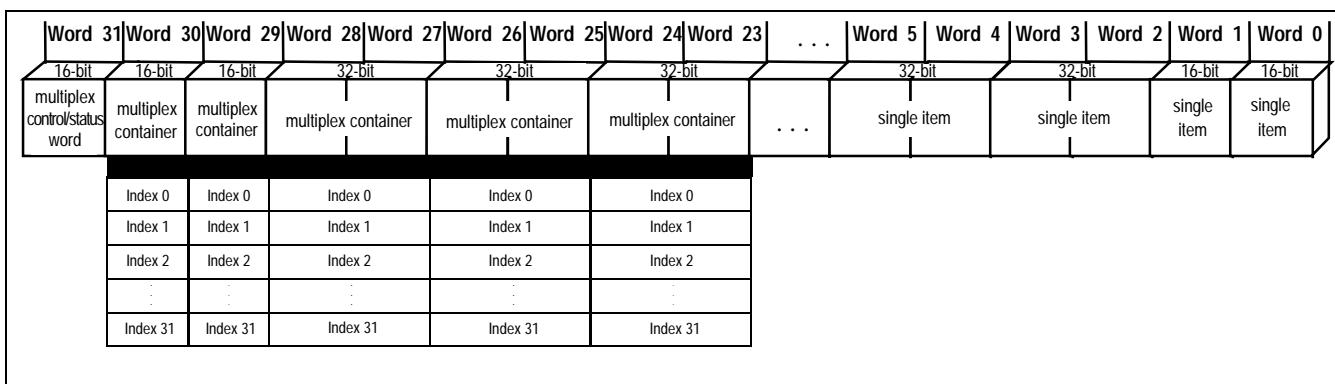


Fig. 6-3: Sample Command (PLC⇒PPC-R) or Response (PPC-R⇒PLC)

The multiplex control and status words serve to command and acknowledge multiplex data transferred between the fieldbus master and the fieldbus slave. The **control** word is associated with **output** communication (PLC⇒PPC-R). The **status** word is associated with **input** communication (PPC-R⇒PLC). Single data items are not affected by the multiplex control and status words.

Note: For specific information about how the fieldbus master uses the multiplex control and status words, refer to Multiplexing on page 6-19.

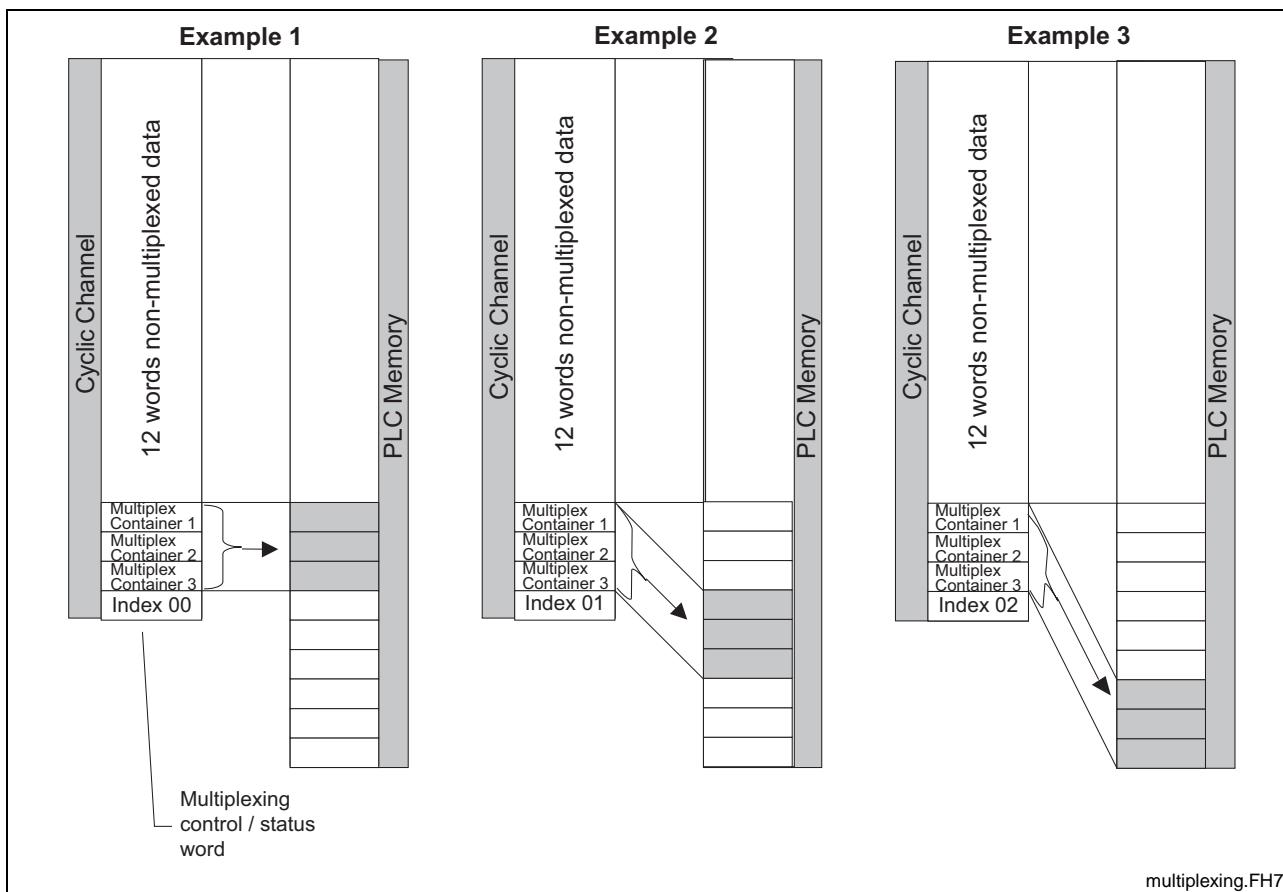


Fig. 6-4: Examples for Reading Data via the Multiplex Channel

Parameter Channel

For Profibus systems using the PPC-R/VisualMotion configuration, a subset of the cyclic (DP) channel can be allocated for non-cyclic communications (e.g. parameterization and extended diagnostic information). This subset of the cyclic channel is called the **Parameter Channel**.

Note: The Parameter Channel is always allocated as the first 3, 4 or 6 words of the Profibus cyclic (DP) channel. The length of the Parameter Channel plus the length of the Real-time Channel used to exchange cyclic data represents the entire length of the DP channel (maximum total length: 16 words). Refer to *Fig. 6-2* for DP channel configuration options.

Two messaging formats are available in the Parameter Channel, to allow for a varying degree of implementation, depending on application requirements:

- **Short Format 3-** messaging format that provides direct access to Indramat mapped objects (Registers, Global Floats, Global Integers, Floats, Integers, as well as Card, Axis, Task, and Drive S and P parameters).

Note: List parameters can be accessed using the Data Exchange Object. For further explanation of how this format functions, refer to *Messaging Formats* on page 6-25.

- **VisualMotion ASCII Format-** provided for backward-compatibility with previous VisualMotion versions. For specific information about this format, refer to the *VisualMotion 6.0 Start-Up Guide* (DOK-VISMOT-VM*-06VRS**-PR02-AE-P, Mat. No. 282762).

6.2 Fieldbus Mapper Functionality

Initializing the Fieldbus Mapper from VisualMotion 8

1. Open an existing program or create a new program. You must be using PPC-R hardware with GPP firmware to use the Fieldbus Mapper described in this document.

Note: Make sure the VisualMotion system is configured for GPP (in the **Settings⇒Configuration** menu item of the main VisualMotion screen).

2. Select **Commission⇒Fieldbus Mapper**. The main Fieldbus Mapper screen appears (refer to *Fig. 6-5*).

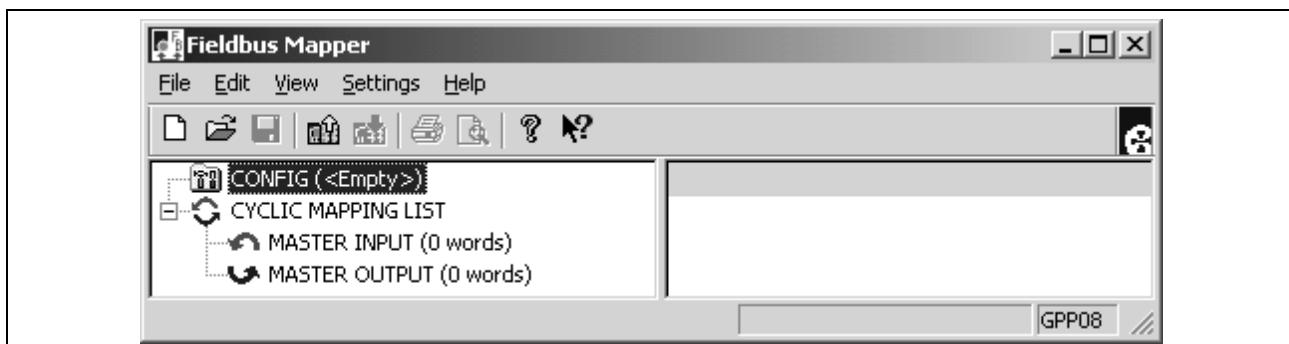


Fig. 6-5: Fieldbus Mapper Main Screen (Blank)

Creating a Fieldbus Mapper File

1. Click or select **File⇒New**. A “setup wizard” goes through three steps:

- Fieldbus Slave Definition
 - Fieldbus Slave Configuration
 - Cyclic Data Configuration
2. Enter the information requested in the setup screens. For more details on each step, refer to *Fieldbus Slave Definition*, *Fieldbus Slave Configuration*, and *Cyclic Data Configuration* for detailed information about each configuration step.
 3. Save the file (automatically has a *.prm extension).

Note: To edit an existing fieldbus mapper file:

1. Click  or select **File**⇒**Open**.
2. Browse to find the desired file (*.prm extension).
3. Click **Open**. The main Fieldbus Mapper screen appears, which lists the configuration information. Refer to Fig. 6-6.

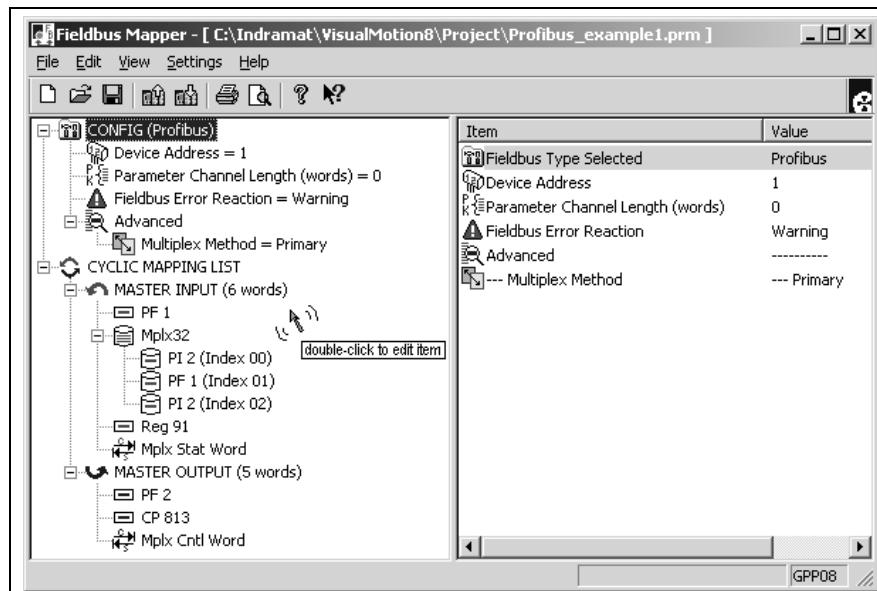


Fig. 6-6: Fieldbus Mapper Main Screen (Complete)

4. From the Fieldbus Mapper main screen, **double-click** on the specific item to be edited. The corresponding setup screen appears.

- Or -

Select the item to edit from the **Edit** menu (refer to Fig. 6-7). For more information about each step, refer to *Fieldbus Slave Definition*, *Fieldbus Slave Configuration*, and *Cyclic Data Configuration* for detailed information about each configuration step.

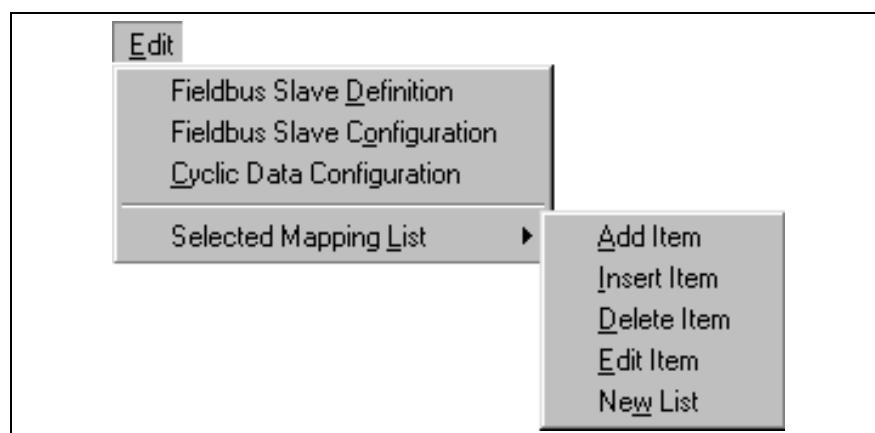


Fig. 6-7: Fieldbus Mapper Edit Menu

Note: You can also directly add, insert, delete, edit an item, or create a new list by:

- clicking on the item to be edited in the main Fieldbus Mapper screen and selecting the desired function under **Edit⇒Selected Mapping List**

OR

- right-clicking on an item to display a menu of functions

Fieldbus Slave Definition

From the Fieldbus Slave Definition window, select the desired Hardware Platform and select **Profibus** as the Fieldbus Type (refer to *Fig. 6-8*).

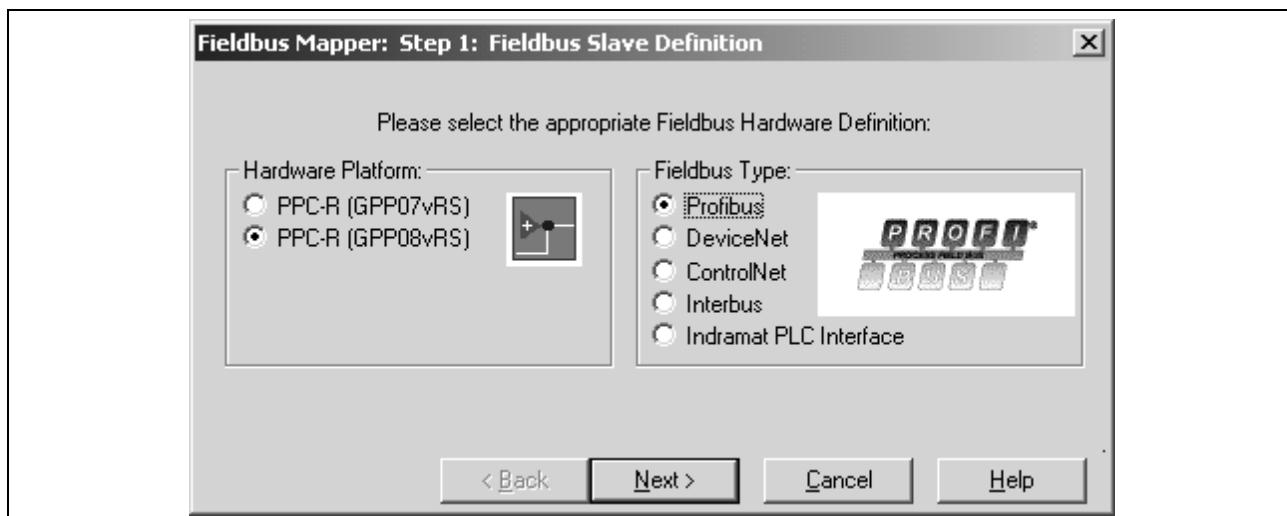


Fig. 6-8: Fieldbus Slave Definition Window

Fieldbus Slave Configuration

The Profibus Fieldbus Slave Configuration screen is shown in *Fig. 6-9* below.

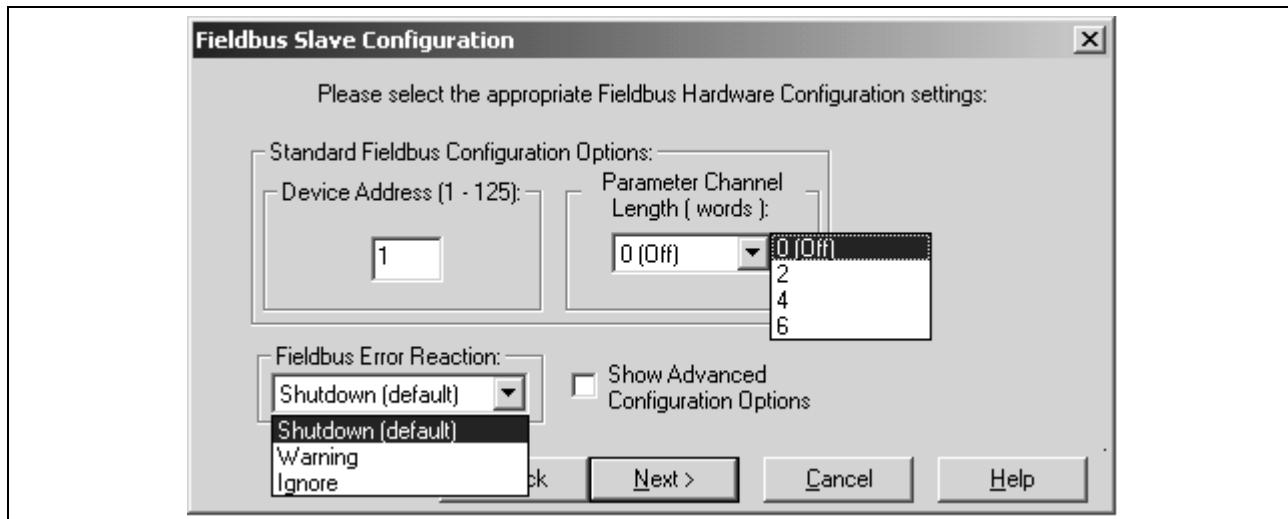


Fig. 6-9: Fieldbus Slave Configuration

Standard Fieldbus Configuration Options:

- Device Address (0-125):** set to a unique number for the devices on the bus
- Parameter Channel Length (words):** set to 0 (Off), 2, 4 or 6 words. If 2, 4 or 6 words are selected, these are automatically allocated for the Parameter Channel in the Cyclic Data Input and Output Lists.

Fieldbus Error Reaction:

Set the Error Reaction to Shutdown (default), Warning or Ignore. Refer to *Fieldbus Error Reaction* on page 6-17 for detailed information about each setting.

Advanced Configuration Options

The “Advanced Options:” are shown only if the checkbox next to **Show Advanced Configuration Options** is checked (refer to *Fig. 6-10* below). In most cases, the default options should apply.

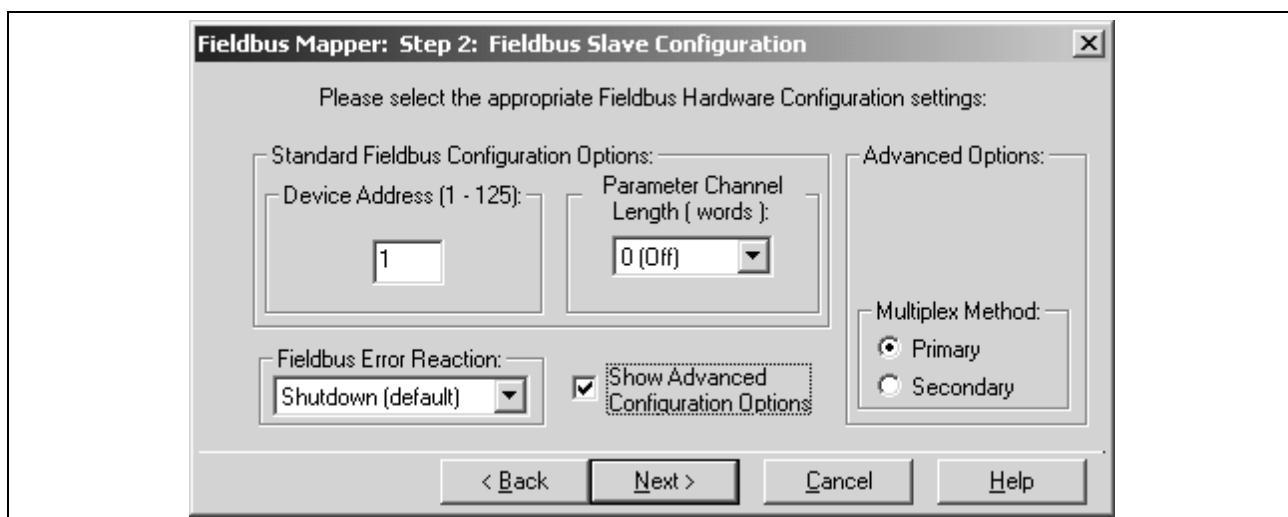


Fig. 6-10: Fieldbus Slave Configuration: Advanced

- Multiplex Method:** select Primary or Secondary (Primary is the default). Select Secondary only if you have an inconsistent fieldbus master. Refer to *Multiplexing* on page 6-19 for detailed information about each method.

Cyclic Data Configuration

An example of the Cyclic Data Configuration screen is shown in *Fig. 6-11* below. In this screen, four words have been allocated for the Parameter Channel (optional for Profibus fieldbuses only). No data has yet been configured. If you are editing an existing Fieldbus Mapper file, the list will probably contain more items.

First, you must select the Cyclic Input List (from PPC-R to PLC) or the Cyclic Output List (from PLC to PPC-R).

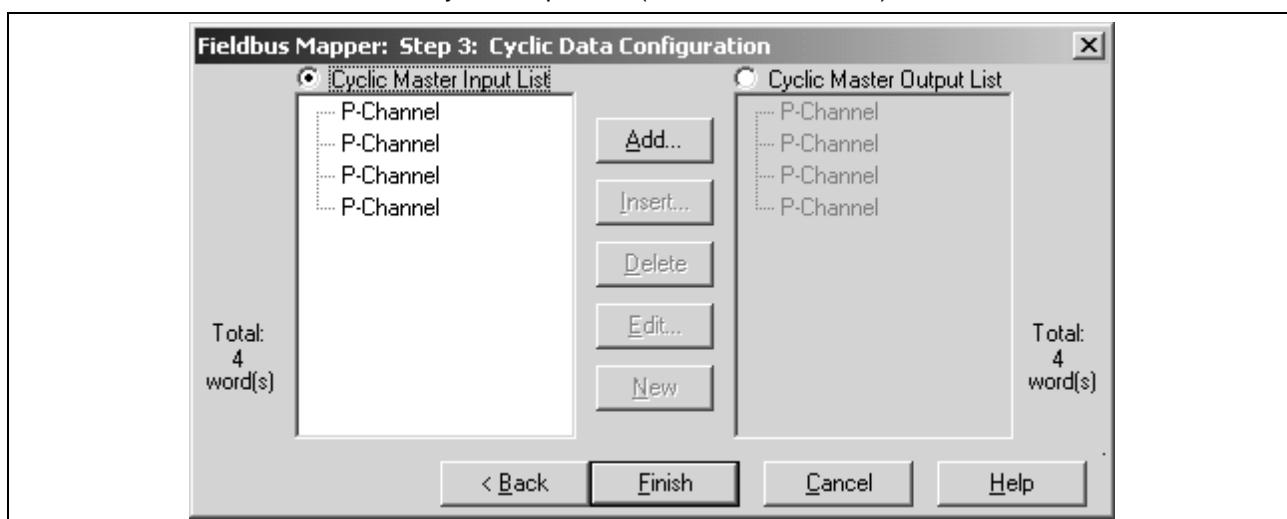


Fig. 6-11: Cyclic Data Configuration—Initial Screen

Adding an Item to the List

1. Select the Cyclic Input List or the Cyclic Output List.
2. Click Add. The screen in *Fig. 6-12* below appears. Select the Data Type (for example, Register).

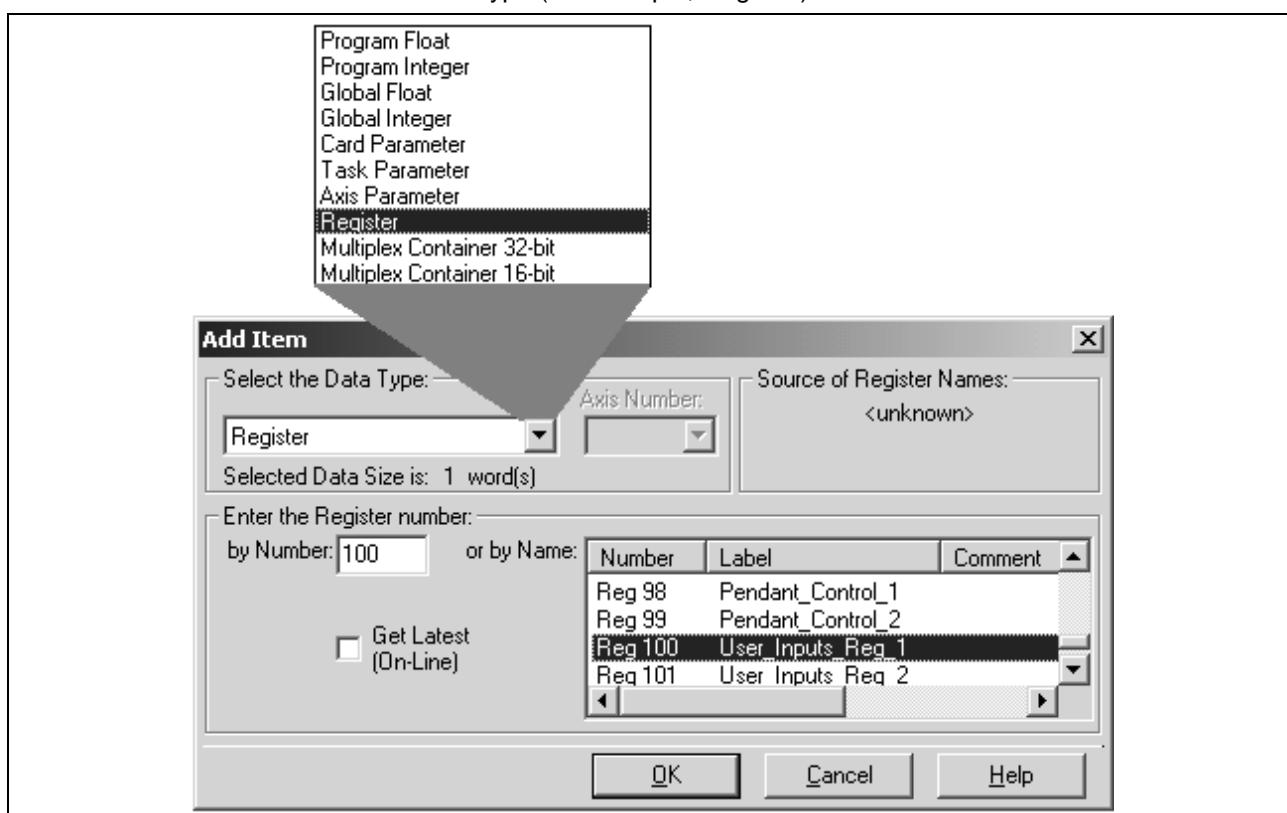


Fig. 6-12: Add Item to Cyclic Data

Note: Registers and 16-bit Multiplex Containers (used only for Registers) require one data word (16 bits), and all other data types require two data words (32 bits) of space.

3. Enter the required information (for example Register Number) or select it from the list below. Only the available data types for your designated VisualMotion hardware setup and fieldbus type are listed.

Note: If you check the box next to "Get Latest (On-Line)," the data type label list is updated based on your firmware version and the currently active program.

4. Click OK to add the selected item to the list.

Adding Multiplex Containers to the List

1. Select the Cyclic Input List or the Cyclic Output List.
2. Click Add.
3. In the "Add Item" screen under "Select the Data Type:" select Multiplex Container 16-bit (for Registers) or Multiplex Container 32-bit (for all other data types).
4. Click OK to add the Multiplex Container to the List. The screen in *Fig. 6-13* below is an example where a 16-bit Multiplex Container and a 32-Bit Multiplex Container have been added.

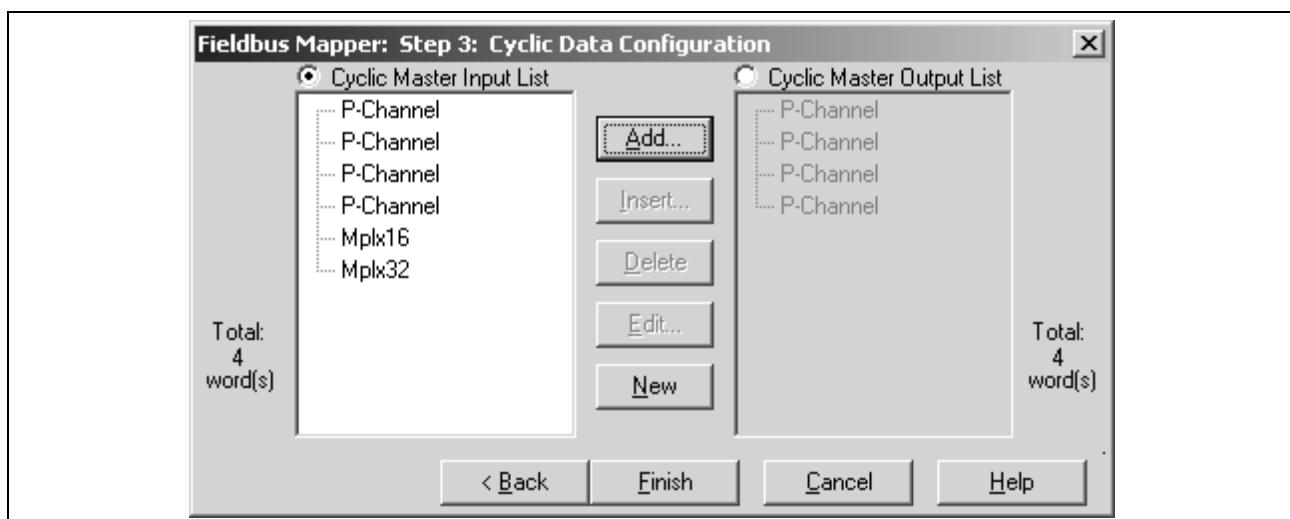


Fig. 6-13: Cyclic Data Configuration, Multiplex Containers

Note: At this point, the Multiplex Containers do not yet contain any items. To add multiplex items, refer to below.

Adding Items to an Empty Multiplex Container

1. In the Cyclic Data Configuration screen, select the multiplex container to which you want to add items.
2. Click Add. The screen in *Fig. 6-14* below appears. Because it is unclear whether you would like to add to the list or to the multiplex container, the Fieldbus Mapper is requesting clarification.

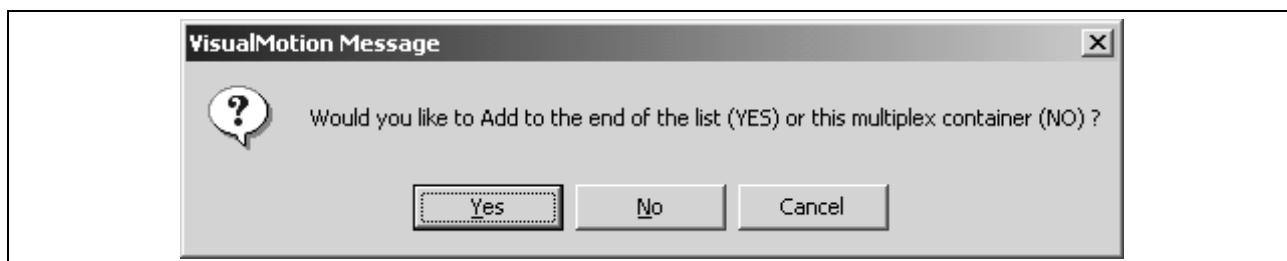


Fig. 6-14: Add Item or Multiplex Item Screen

Note: For subsequent items, highlight any of the indexes within the multiplex container before clicking Add, and the Fieldbus Mapper will know you want to add to that container.

3. To add to the selected multiplex container, click No. The screen in Fig. 6-15 below is an example for adding a 32-bit multiplex item.
4. Select the desired item to be added to the multiplex container.

Note: In addition to the data types that can be added to the multiplex list, an empty item called **Multiplex Empty Item** is available to fill a space within the multiplex container, if nothing is to be mapped to a particular index.

5. Click OK. The item is automatically placed in the multiplex container as the next unassigned index item (e.g. the first item is index 00, the last is index 31).
6. Repeat for as many items as you want to add to the multiplex container, up to 32 items.

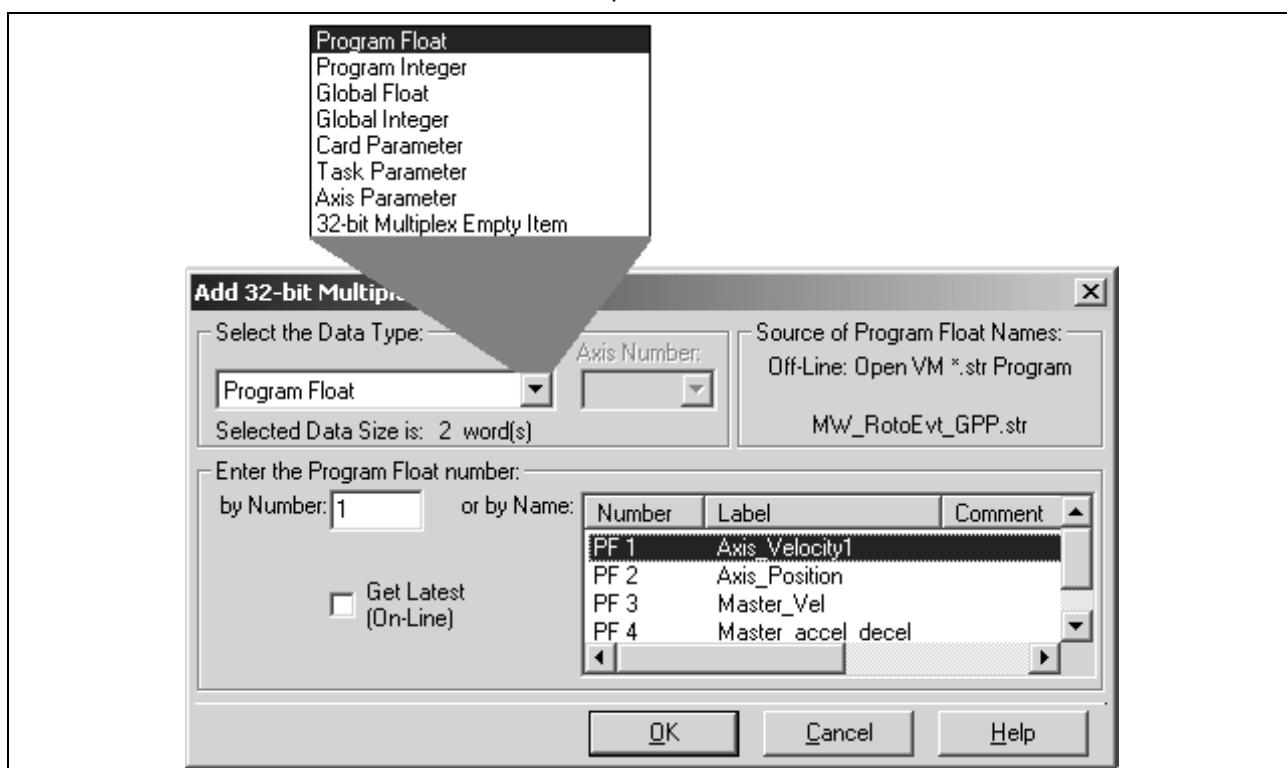


Fig. 6-15: Adding a Multiplex Item to the Container (32-bit example)

Editing the Cyclic Data Lists

To make changes to an existing list, use the following buttons:

Button	Function
Add...	Inserts a new item at the end of the list.
Insert...	Inserts a new item into the list directly before the selected item.
Delete	Removes the selected item from the list.
Edit...	Allows editing of the selected item. (To edit a list item, you may also double-click on it.)
New	Clears up the current list.

Table 6-2: Button Functions in the Cyclic Data Configuration Window

Additional Functions

Several additional functions are now available in the Fieldbus Mapper:

Function	Icon	Menu Selection
Download the current fieldbus configuration from the PPC-R		File⇒Get Fieldbus Configuration from PPC
Upload the current fieldbus configuration in the Fieldbus Mapper to the PPC-R		File⇒Send Fieldbus Configuration to PPC
Print the current fieldbus configuration data		File⇒Print
Preview the printout of the current fieldbus configuration data		File⇒Print Preview

Table 6-3: Additional Functions

Getting the Fieldbus Configuration from the PPC

After getting the fieldbus configuration from the PPC, the following information is detected by the system and appears in the configuration list:

- Fieldbus Type Found
- Fieldbus FW (Firmware) Version
- GPP Control FW (Firmware) Version

An example is shown in *Fig. 6-16* below.

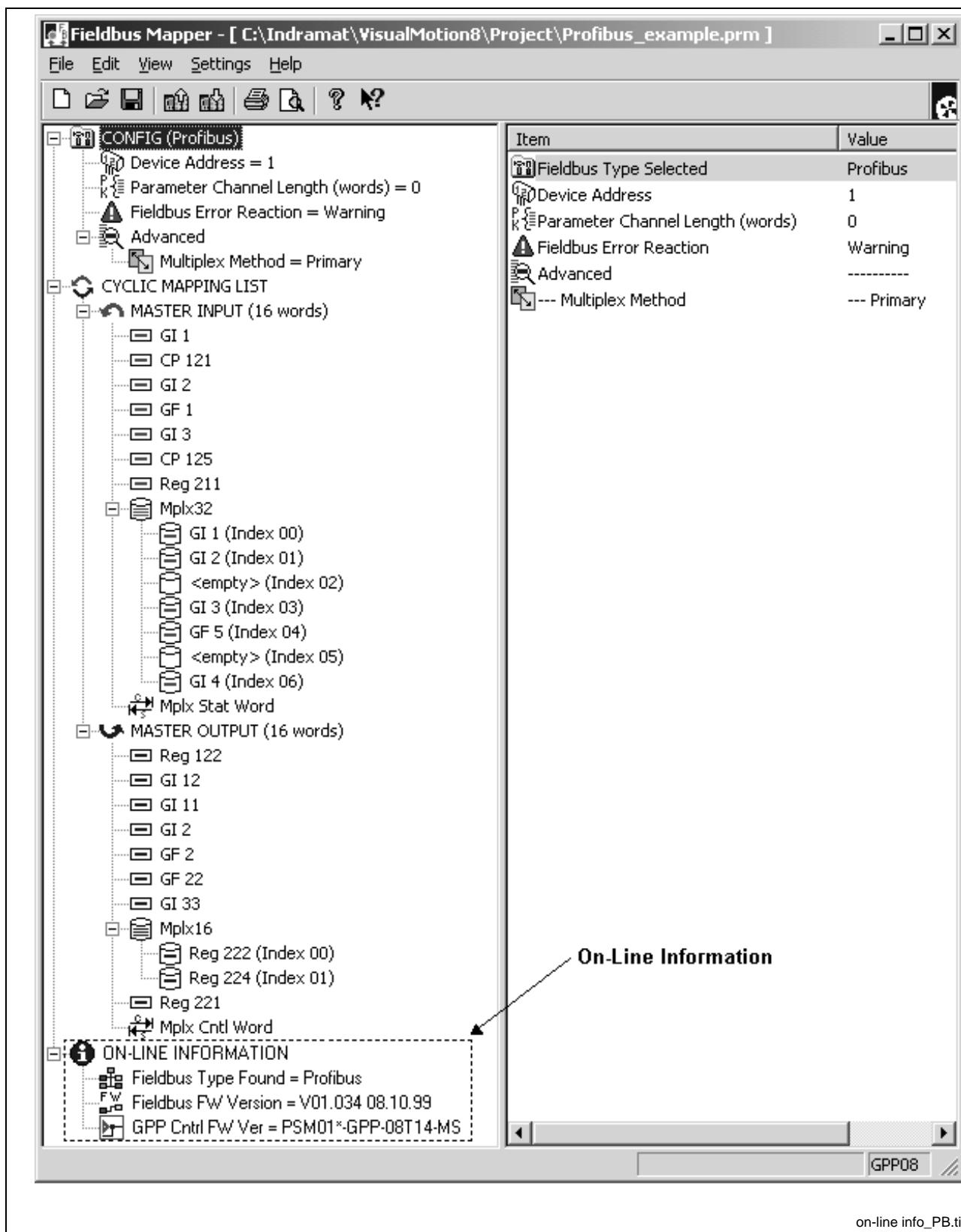


Fig. 6-16: On-Line Fieldbus Configuration Information

6.3 Information for the GPP Programmer

Register 19 Definition (Fieldbus Status)

VisualMotion Register 19 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 6-4 below contains the bit assignment for the diagnostic object 5ff2. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	---	---	---	---	---	---	---	---	---	x5	x4	---	x2	x1

Table 6-4: Bit Assignment for VisualMotion Register 19

Bit Definitions

x1, x2 Status bits for the internal DPR (Dual-Port RAM) communication between the fieldbus slave and the PPC-R:

x1: FB Init. OK , LSB (least significant bit)

x2: FB Init. OK, MSB (most significant bit)

The bit combinations for x1 and x2 are as follows:

Bit 2 (PPC-R)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, or neither the PPC-R nor the fieldbus card have 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 6-5: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

- x4** Status bit for the active bus capabilities of the fieldbus slaves (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. Refer to *Fieldbus Error Reaction* on page 6-17 for an explanation of the Fieldbus Error Reaction setting.
 0--> The fieldbus slave is not (yet) ready for data exchange.
 1--> The fieldbus slave can actively participate on the bus.
- x5** Status bit for the non-cyclic channel (Parameter Channel) (Non-Cyc Ready)
 0--> The non-cyclic channel (Parameter Channel) cannot (yet) be used.
 1--> The non-cyclic channel (Parameter Channel) is ready for use by the fieldbus master.
- x15** Status bit for the cyclic data output (Cyclic Data Valid):
 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 20 Definition (Fieldbus Diagnostics)

VisualMotion Register 20 holds the information for "Fieldbus Diagnostics."

Table 6-6 below contains the bit assignment for the diagnostic object 5ff0. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---".

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
X16	x15	x14	x13	---	---	---	---	---	---	---	---	---	---	---	---

Table 6-6: Bit Assignment for VisualMotion Register 20

Bit Definitions

x13 - x16 Identification of the fieldbus interface card (FB Card Found)

The bit combinations for x13, x14 and x15 are as follows:

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 6-7: Identification of the Fieldbus Interface

Register 26 Definition (Fieldbus Resource Monitor)

The "Fieldbus Resource Monitor" in register 26 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 26 is divided into the following three counter types:

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

Table 6-8 below contains the bit assignment for the fieldbus counters. The assigned bits are labeled with an "x" followed by the bit number.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1

Table 6-8: Bit Assignment for Fieldbus Resource Monitor (Register 26)

Note: View Register 26 in Hexadecimal format to more easily monitor the fieldbus counters.

Hex display format of register 26

0x0000



Fieldbus Time-out Counter (00-FF)

Increments by one every time 10 cyclic mapping list attempts are missed in 10 attempts.

Peak Miss Counter (0-9)

Displays the peak value of the Current Miss Counter.

Current Miss Counter (0-9)

Displays the count of missed cyclic mapping list attempts out of the last 10 attempts

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

Bit Definitions

x13 - x16

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.

x9 - x12

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.

x1 - x8

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. Refer to *Fieldbus Error Reaction* below for an explanation of the Fieldbus Error Reaction setting.

Note: The GPP programmer can monitor the "Current Miss Counter" and define a custom error reaction for missed mapping list update attempts less than 10.

Note: The values in register 26 are read/write and can be reset by the user.

Fieldbus Error Reaction

Note: The Fieldbus Error Reaction setting is active only in SERCOS Phase 4. In all other SERCOS phases, it will be inactive.

You can select how you would like the PPC-R system to react in case of a fieldbus error. This reaction can be set in the "Fieldbus Slave Configuration" 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 stored in Parameter C-0-2635:

Setting	Value in Parameter C-0-2635
Shutdown	0 (default)
Warning Only	1
Ignore	2

Table 6-9: Parameter C-0-2635 Values for Error Reaction Settings

Fieldbus Mapper Timeout

The Fieldbus Mapper continually scans the system for sufficient resources to process the cyclic data mapping lists (2600 and 2601 lists). If 10 out of 10 attempts of the mapping list updates are missed, the system is considered to have insufficient resources and the selected error reaction is evoked, as follows:

If "Shutdown" (0) is set in Parameter C-0-2635, the following error is generated from the PPC-R card: **520 Fieldbus Mapper Timeout**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated: **209 Fieldbus Mapper Timeout**

If "Ignore" (2) is set in Parameter C-0-2635, the system will update as resources become available, but there is no way to monitor whether or not updates actually occur.

Lost Fieldbus Connection

Register 19, bit 4 indicates the status of the fieldbus. Refer to *Register 19 Definition (Fieldbus Status)* on page 6-15 for more specific bit information. The system monitors this bit and evokes the selected error reaction if the bit is low (0), after a fieldbus card is found. A typical situation that will cause this condition is the disconnection of the fieldbus cable from the fieldbus card.

If "Shutdown Control" (0) is set in Parameter C-0-2635, the following error is generated from the PPC-R (active in SERCOS Phase 4 only): **519 Lost Fieldbus Connection**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated (active in SERCOS Phase 4 only): **208 Lost Fieldbus Connection**

If "Ignore" (2) is set in Parameter C-0-2635, there is no noticeable reaction when Register 19 Bit 4 goes low, unless the GPP application program is customized to evoke a special reaction.

Troubleshooting Tip:

If a fieldbus card is not found on the system, the Error Reaction setting will be ignored. If you have a fieldbus card and the Error Reaction is not responding as expected, the system may not "see" your fieldbus card.

6.4 Information for the PLC Programmer

*.gsd File

Indramat supplies a *.gsd file containing supporting information for the PPC-R with a Profibus slave configuration. Contact an Indramat technical representative for the location of this file.

Multiplexing

Primary Multiplex Method (for Consistent Masters only)

Important: You should not use the Primary Multiplex Method for a master that is not consistent over the entire cyclic channel. The Secondary Multiplex Method is available for inconsistent masters. Refer to *Secondary Multiplex Method (for Inconsistent Masters)* on page 6-22.

The advantage of the Primary Method is easier handling of input data for consistent masters.

Control Word and Status Word

Control Word

The control word is transferred in the multiplex channel from master to slave. It tells the slave in which index the data is being transferred from master to slave and in which index the data is requested from slave to master.

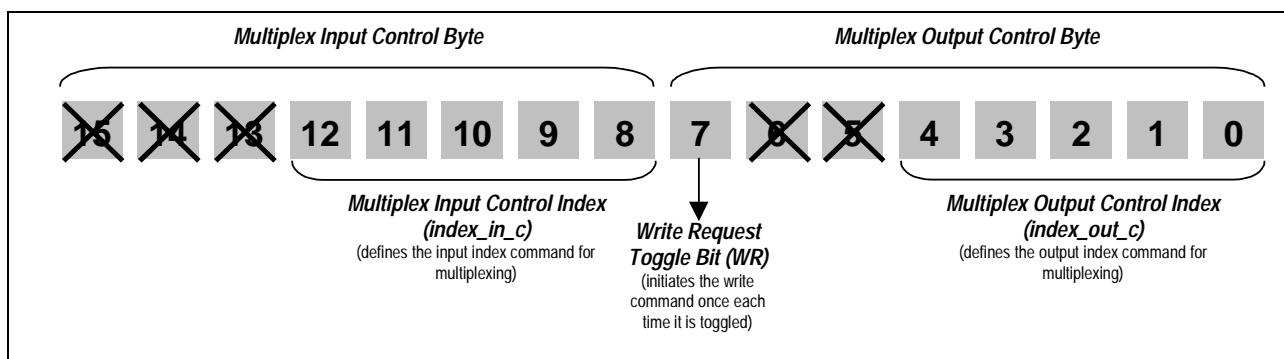


Fig. 6-17: Control Word Definition, Primary Multiplex Method

Index_out_c: tells the slave in which index the data are transferred from master to slave (out = master -> slave, _c = element of control word).

Index_in_c: tells the slave in which index the data is requested from slave to master (in = slave -> master, _c = element of control word).

WR (Write Request): handshake bit (refer to meaning of WR and WA).

Note: Input data via the Multiplex Channel is continually being updated.

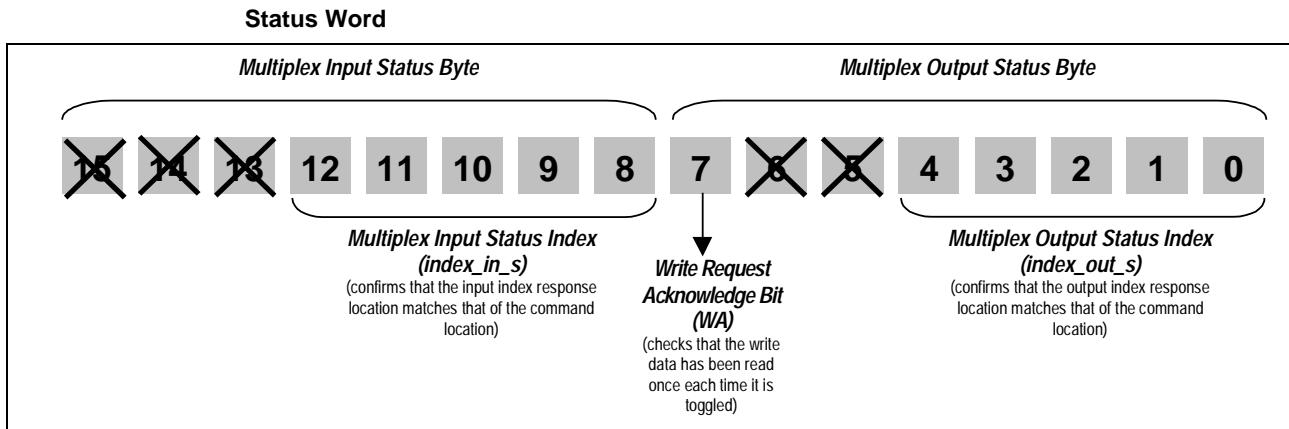


Fig. 6-18: Status Word Definition, Primary Multiplex Method

- **Index_out_s:** acknowledges index written by the master (out = master -> slave, _s = element of status word).
- **Index_in_s:** tells the master which index is transferred from slave to master in the actual process data cycle (in = slave -> master, _s = element of status word).
- **WA (Write Acknowledge):** Handshake bit (refer to meaning of WR and WA).

Handshake Bits WR and WA

WR and WA are handshake bits that allow the controlled writing of data via the multiplex channel. WR and WA control the data transfer for writing data_out (data send from master to slave).

WR == WA:

- tells the master that the slave has received the last multiplex data_out. The master can now send new data_out.
- tells the slave to do nothing, because the master has not yet put new consistent data_out on the bus.

WR! = WA:

- tells the slave to do something, because the master has now put consistent new data_out on bus.
- tells the master to do nothing, because the slave has not yet received the latest multiplex data_out.

Master Communications (Primary Multiplex Method)

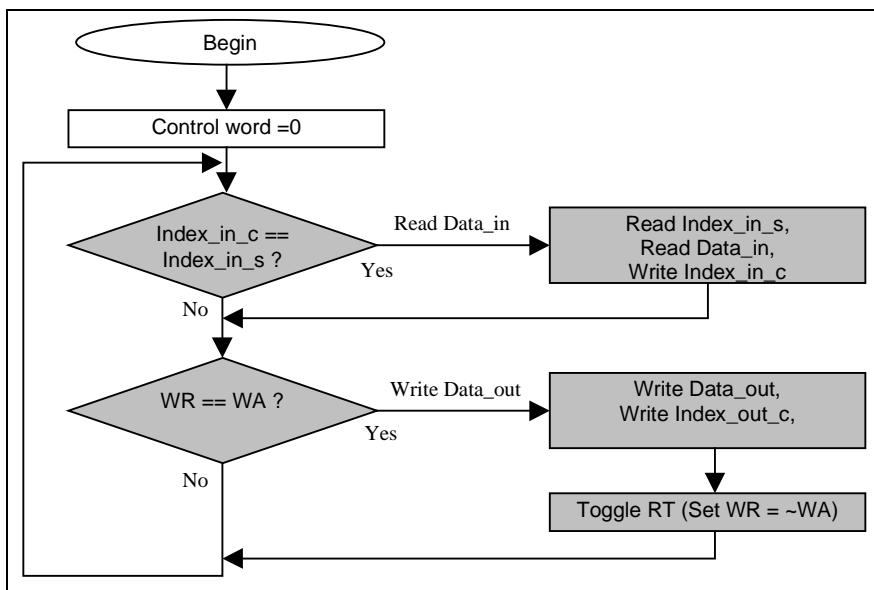


Fig. 6-19: Primary Multiplex Method, Master Communications

Programming Example

To aid in implementing the multiplex function in a PLC program, the following flow chart shows two ways of reading and writing data. Reading and writing can be executed separately, which allows the input data to be updated about 30% faster. The "Read Data" example would be placed at the beginning of a PLC program the "Write Data" example at the end.

Combined reading and writing makes the PLC program simpler, especially when using the same index for both transfer actions.

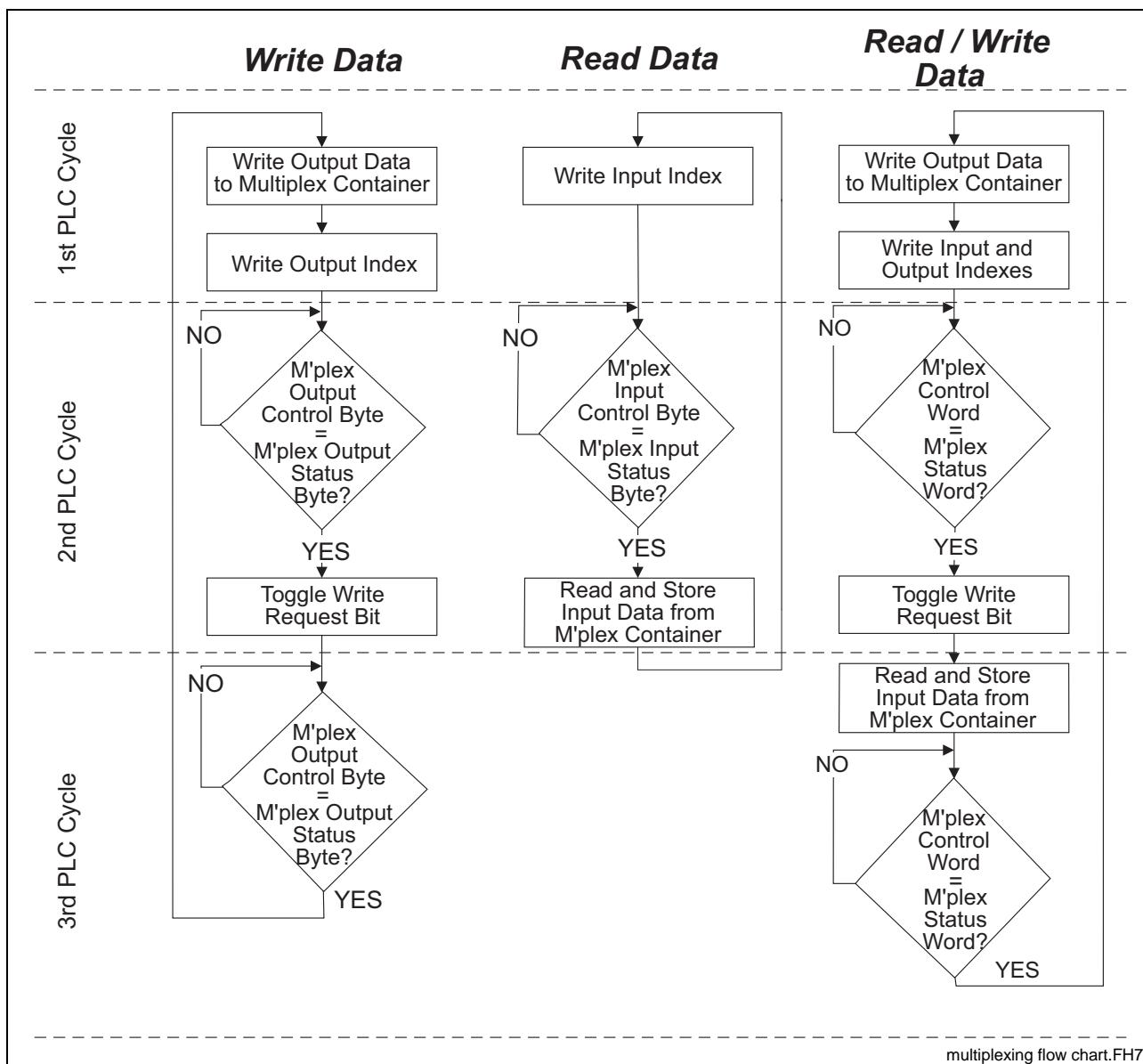


Fig. 6-20: Flow Chart of Multiplex Programming Examples (Primary Method)

Secondary Multiplex Method (for Inconsistent Masters)

Explanation of the Master Consistency Problem

The PPC-R fieldbus slave interfaces can guarantee consistency.

However, some fieldbus masters can only guarantee byte, word or double word consistency. If the master is only word-consistent, it is possible that the master cannot transfer the data and the control word of one multiplex index consistently from the PLC to the fieldbus. Therefore, it is necessary to have a second multiplex method where both input data and output data require the handshake bits to update via the fieldbus.

Note: The meanings of the control and status words are the same as for the Primary Multiplex Method. The only difference is the toggle bits RR and RA, which are used in the Secondary Method.

Fig. 6-21 below illustrates the control word definition for the Secondary Multiplex Method.

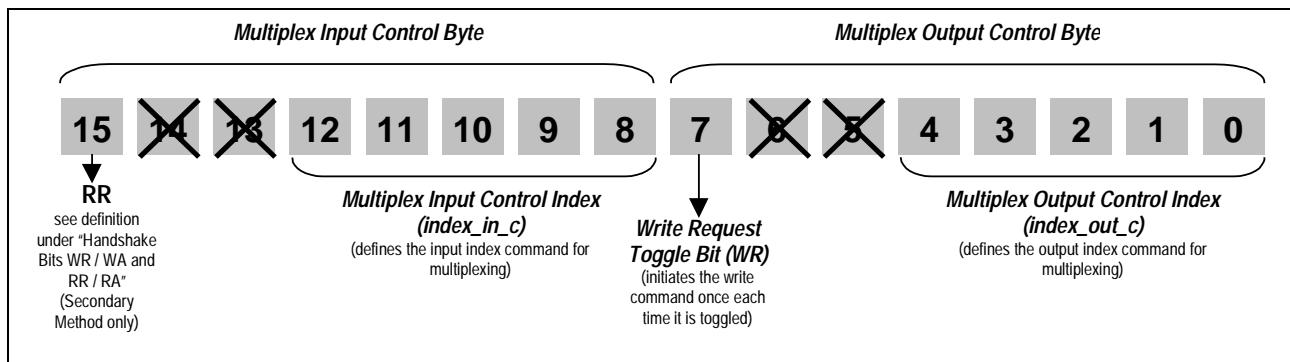


Fig. 6-21: Control Word Definition, Secondary Multiplex Method

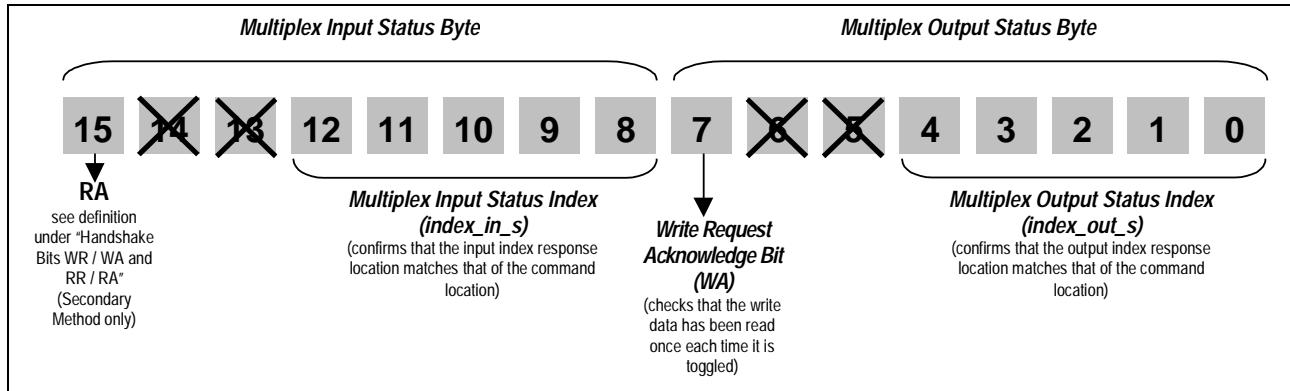


Fig. 6-22: Status Word Definition, Secondary Multiplex Method

The Secondary Multiplex Method has the following features:

- You can transfer a different index from master to slave as from slave to master.
- The handshake bits for both reading and writing of this multiplex channel make the multiplexing possible on inconsistent systems (masters).

Handshake Bits RR and RA

RR (Read Request) and RA (Read Acknowledge) are handshake bits that allow a controlled data transfer and use of the multiplex channel on inconsistent masters. RR and RA control the data transfer for reading data_in (data send from slave to master).

RR == RA:

- tells the master that the slave has sent the requested data_in. The master can now read the data_in and request new data_in.
- tells the slave to do nothing, because the master has not yet put new consistent data on the bus.

RR != RA:

- tells the slave to put new data_in on the bus, because the master requests new data_in.
- tells the master to do nothing, because the slave has not yet put the latest requested multiplex data_in on the bus.

Master Communications (Secondary Multiplex Method)

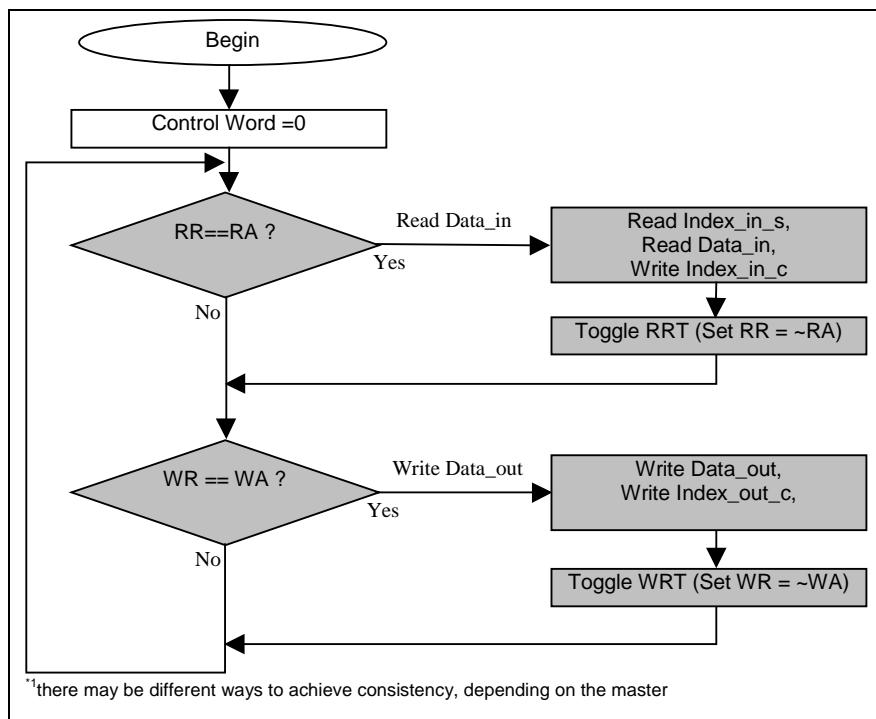


Fig. 6-23: Secondary Multiplex Method, Master Communications

For some masters, it could be enough to first write data and then the control word. For other masters, you may have to implement a delay time (this time could be different from master to master) before writing WR = ~WA.

Non-Cyclic Data Access via the Parameter Channel

Important: The fieldbus master's access of the cyclic channel must be consistent over the entire length of the assigned Parameter Channel in order to establish reliable Parameter Channel communications.

To support the configuration of drives and the access to parameters through the Profibus DP channel, Indramat has established the Parameter Channel.

If the Parameter Channel is used with the PPC-R, the first 2, 4 or 6 data words of the cyclic channel for the slave board must be allocated for non-cyclic transmissions.

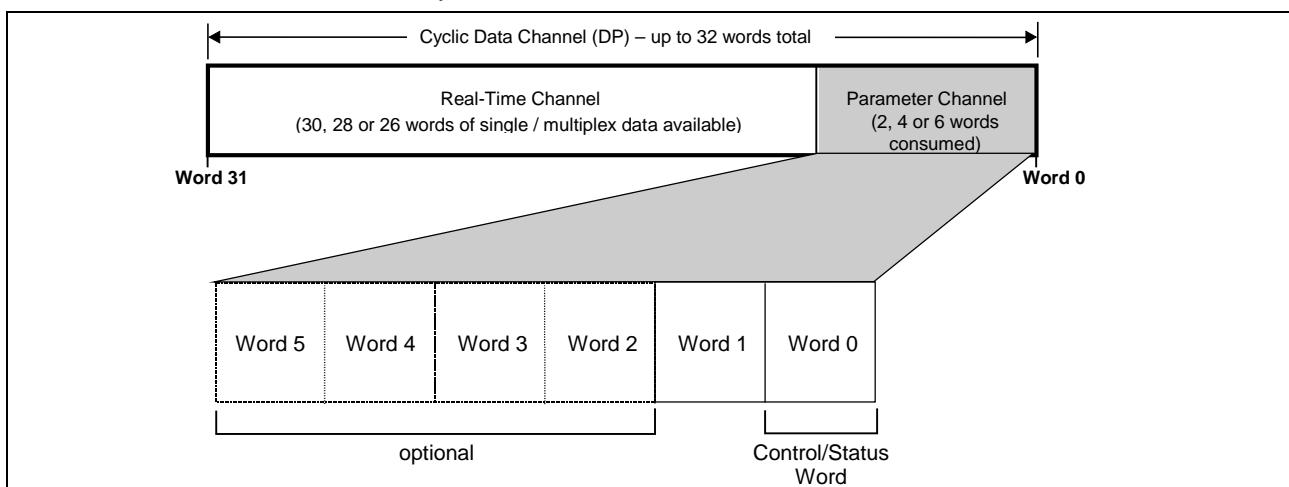


Fig. 6-24: The Parameter Channel inside the Profibus DP Channel

Messaging Formats

Two messaging formats are available in the Parameter Channel:

- **Short Format 3**
- **VisualMotion ASCII Format** - This format is provided for backward-compatibility with VisualMotion 6.0 / GPS firmware. For detailed information, refer to the VisualMotion 6.0 Startup Guide.

Short Format 3: General Explanation

To read or write a VisualMotion data type non-cyclically, a protocol is used inside the Parameter Channel. The protocol requires one word of the Parameter Channel for protocol functions. Thus, depending on the channel length, 3 or 5 data words can be transferred in one cycle. The protocol supports multiple transmissions, but the maximum length of data that can be transferred from or to an object is 128 bytes.

Short Format 3 Data Transfer

The following methods for transferring data are available in Short Format 3:

- Mapped Data
- Data Exchange Objects

Mapped Data

Mapped data is the most powerful feature of the PPC-R non-cyclic fieldbus interface. Through mapped data, the user has access to virtually every PPC-R data type over the fieldbus. It is easy to implement from the PLC side and requires no setup on the PPC-R side.

To access a data type over the fieldbus, it has to be specified by an address that consists of an index and a subindex. The index and subindex for each data type can be calculated by a formula (refer to *Accessing Mapped Data* on page 6-34).

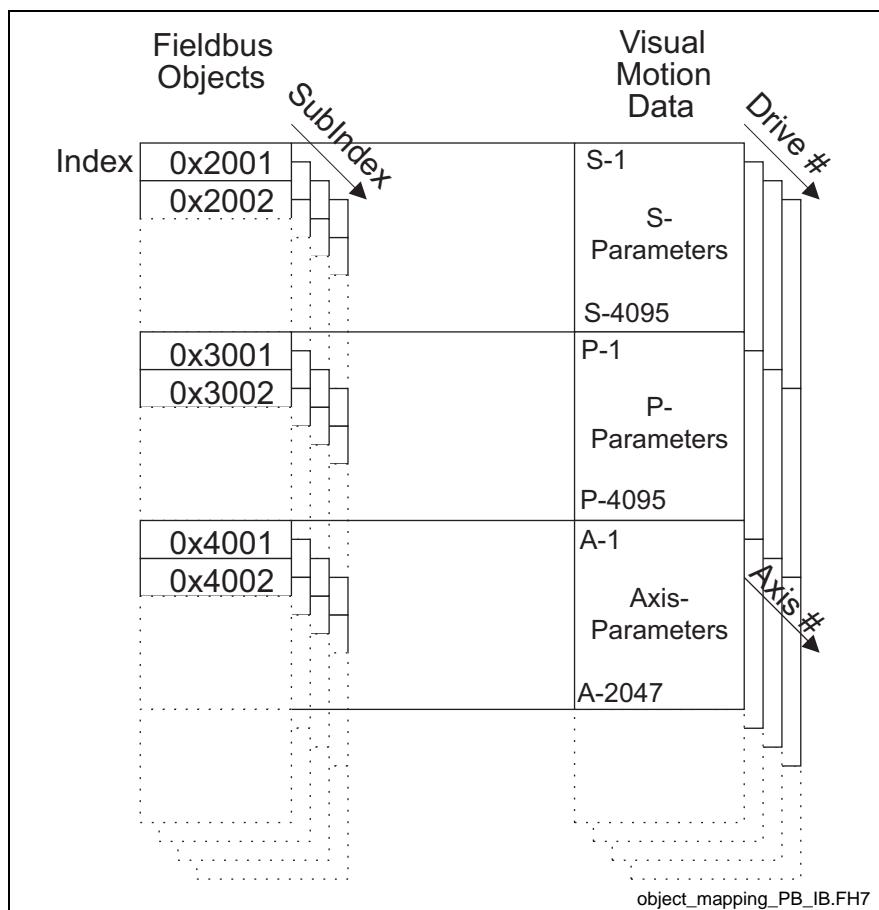


Fig. 6-25: Mapped Data

Mapped data can be used with the following parameters and values:

- S-Parameters (SERCOS Drive S-Parameters)
 - P-Parameters (SERCOS Drive P-Parameters)
 - A-Parameters (PPC Axis Parameters)
 - C-Parameters (PPC C System parameters)
 - T-Parameters (PPC Task parameters)
- size and
format
depend on
parameter *¹

PF-Values (PPC Program Float data, 32 bit – 2 words, IEEE format) *²

GI-Values (PPC Global Integer data, 32 bit – 2 words) *²

GF-Values (PPC Global Float data, 32 bit – 2 words, IEEE format) *²

PI-Values (PPC Program Integer data, 32 bit – 2 words) *²

Reg.-Values (PPC Register data, 16 bit – 1 word) *³

Data Exchange Objects (0x5E70 – 0x5E73) (embedded ASCII Protocol)

*You may notice that parameters accessed via the non-cyclic (Parameter) channel are not always the same size as reported from the attribute field. This is so that the data sizes correspond with the way the different data types are handled in the cyclic channel (Registers are always set to 16-bit size and Parameters are cast to 32-bit size, even if they actually use less space).

1. When **writing** mapped data to a VisualMotion Parameter, you must send the size data corresponding to that of the attribute field within the parameter.
 - a.) For 32-bit parameters, you must send a data size of 32 bits (otherwise, VM error #07 is returned).
 - b.) For 16-bit parameters, you must send a data of size 16-bits. If, for this case, you send data of size 32 bits, one of the following occurs:
 - i.) For parameters of type 16-bit unsigned, only the Low word is stored, and the High word is ignored.
 - ii.) For parameters of type 16-bit signed, bits 0-14 of the low word along with the sign bit #31 are used, and the remaining bits are ignored.
 - c.) For String Parameters (e.g. S-0-0142), you must send the size of the string to write.

- d.) All other Parameter Types (list parameters, command parameters, etc), are not supported for mapped data.
- When reading mapped data from a VisualMotion Parameter, there are 3 possible cases of sizes returned:
- a.) If the parameter type is a string, you receive the number of bytes corresponding to the length of the string.
 - b.) If the parameter is 32-bit or less, you receive a cast 32-bit value for this parameter. This implies that 16-bit parameters are returned as cast in to 32-bit values.
 - c.) All other parameter types (e.g. list parameters, command parameters, etc.), are not supported for mapped data.
2. When writing mapped data to a VisualMotion Program Float, Program Integer, Global Float, or Global Integer, the data size must be 32-bits (2 words). Any other size returns a VM error #07 (Invalid Data Format).
- When reading mapped data from a VisualMotion Program Float, Program Integer, Global Float, or Global Integer, the data size returned is always 32-bit (2 words).
3. When writing mapped data to a VisualMotion Register, the data must be 16-bits (1 word). Any other size returns a VM error #07 (Invalid Data Format).
- When reading mapped data from a VisualMotion Register, the data size returned is always 16-bit (1 word).

User Data Header – Object Index

The index refers to the particular fieldbus slave object that a VisualMotion data type is (automatically) mapped. This object allows for simple, indirect access to VisualMotion data types, and it is combined with the subindex to create a direct relationship to the VisualMotion data types. The available objects can be calculated using the formulas in *Accessing Mapped Data* on page 6-34.

User Data Header – Object SubIndex

The subindex refers to an additional piece of information necessary to obtain direct access to VisualMotion data types. The reference of the subindex depends on the data type in question. For example, the SubIndex refers to the drive number when accessing S and P parameters. However, the subindex refers to the task number when referring to task parameters. The available subindex ranges can be calculated using the formulas in *Accessing Mapped Data* on page 6-34.

Data Exchange Objects

The four data exchange objects 5E70 to 5E73 represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the PPC-R card. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. Both the VisualMotion ASCII message and the fieldbus transfer message must be formulated.

Table 6-10 lists the available data exchange objects and their sizes.

Data Exchange Object	Data Length (in bytes)
5E70	16
5E71	32
5E72	64
5E73	128

Table 6-10: Length of the Data Exchange Objects

Short Format 3 Parameter Channel (PK) Control and Status Words**PK Control Word**

The PK control word is sent from the master to the slave. It is 16 bits wide and the individual bits have the following meanings:

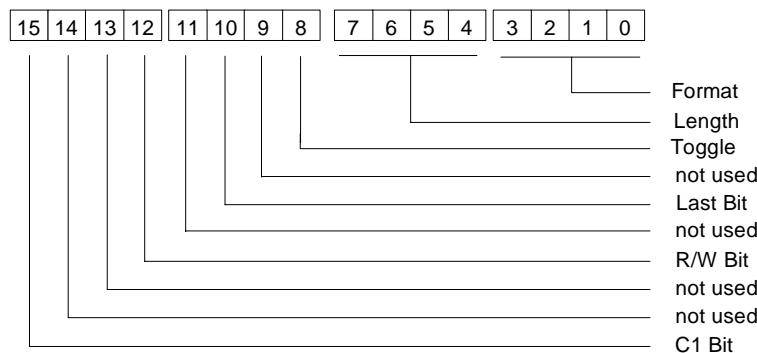


Fig. 6-26: Bits of the PK Control Word

Format: These bits describe the usage and meaning of the following data words in the Parameter Channel. Their value is fixed to 1100_b.

Length: These four bits specify the length of the valid data in bytes, without the control word. The data in the rest of the Parameter Channel is undefined.

Toggle: This bit toggles with every new set of sent data. It is used for a handshake between master and slave. The master is only allowed to toggle this bit when the toggle bit in the status word has the same level as the toggle bit sent in the control word.

L: Last bit. This bit is set when the last fragment of a data block is sent.

R/W: Read/Write; Read = 1, indicates that the master wants to read data.

C1: This bit is used to distinguish between the “old” and “new” handling of the Parameter Channel. For the “new” handling (e.g. Short Format 3), it is fixed to 1

Note: Bits that are not used are set to 0.

PK Status Word

The PK status word is sent as an answer from the slave to the master. The 16 bits have the following meanings:

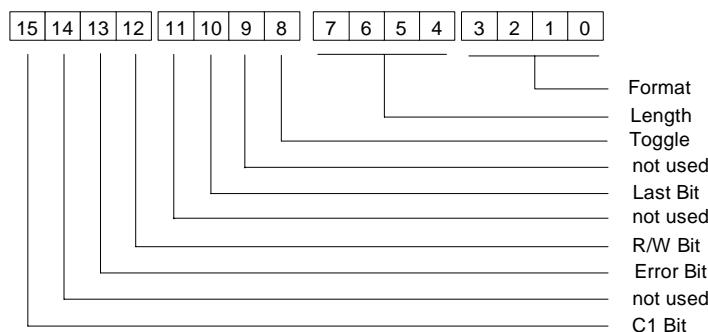


Fig. 6-27: Bits of the PK Status Word

Format: These bits describe the usage and meaning of the following data words in the Parameter Channel. Their value is fixed to 1100_b.

Length: These four bits specify the length of the valid data in bytes, without the status word. The data in the rest of the Parameter Channel is undefined.

Toggle: This bit toggles with every new set of sent data. It is used for a handshake between master and slave. The slave

recognizes new data when the toggle bit it receives (control word) is different from the toggle bit in the status word.

L: Last bit. This bit is set when the last fragment of a data block is sent.

R/W: Read/Write Acknowledgement; Read = 1, indicates that the master wants to read data.

Error Bit: This bit indicates an error that occurred within the slave. The reason for the error is coded in the following data.

C1: This bit is used to distinguish between the "old" and "new" handling of the Parameter Channel. For the "new" handling (Short Format 3), it is fixed to 1

Note: Bits that are not used are set to 0.

Short Format 3: Examples

The following examples show how to write and read an object. They display the read and write access of object index 2001_h, subindex 2_h. The matching Visual Motion data according to the chart at the end of this chapter is S-Parameter 1 of Drive 2.

Notes for the following examples:

Note: These flow charts assume a toggle bit value of 0 when starting. The values of the control and status words can change because of different states of toggle bit and last bit.

Note: The master can detect new data comparing its own toggle bit with the toggle bit received from the slave. If they match, new data was received from the slave.

Note: When writing, only the first telegram from the master contains the index and subindex.

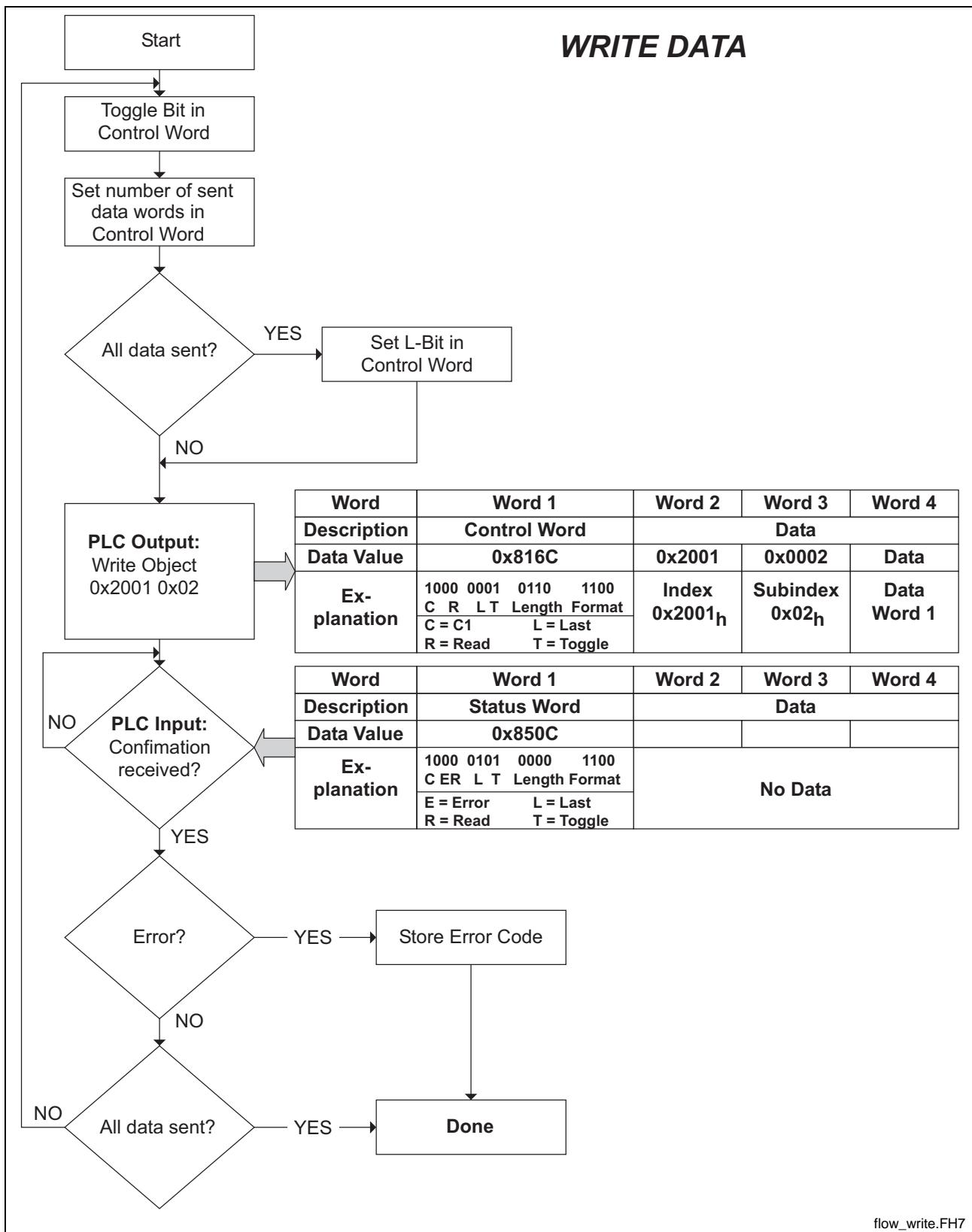


Fig. 6-28: Write Data Object Example

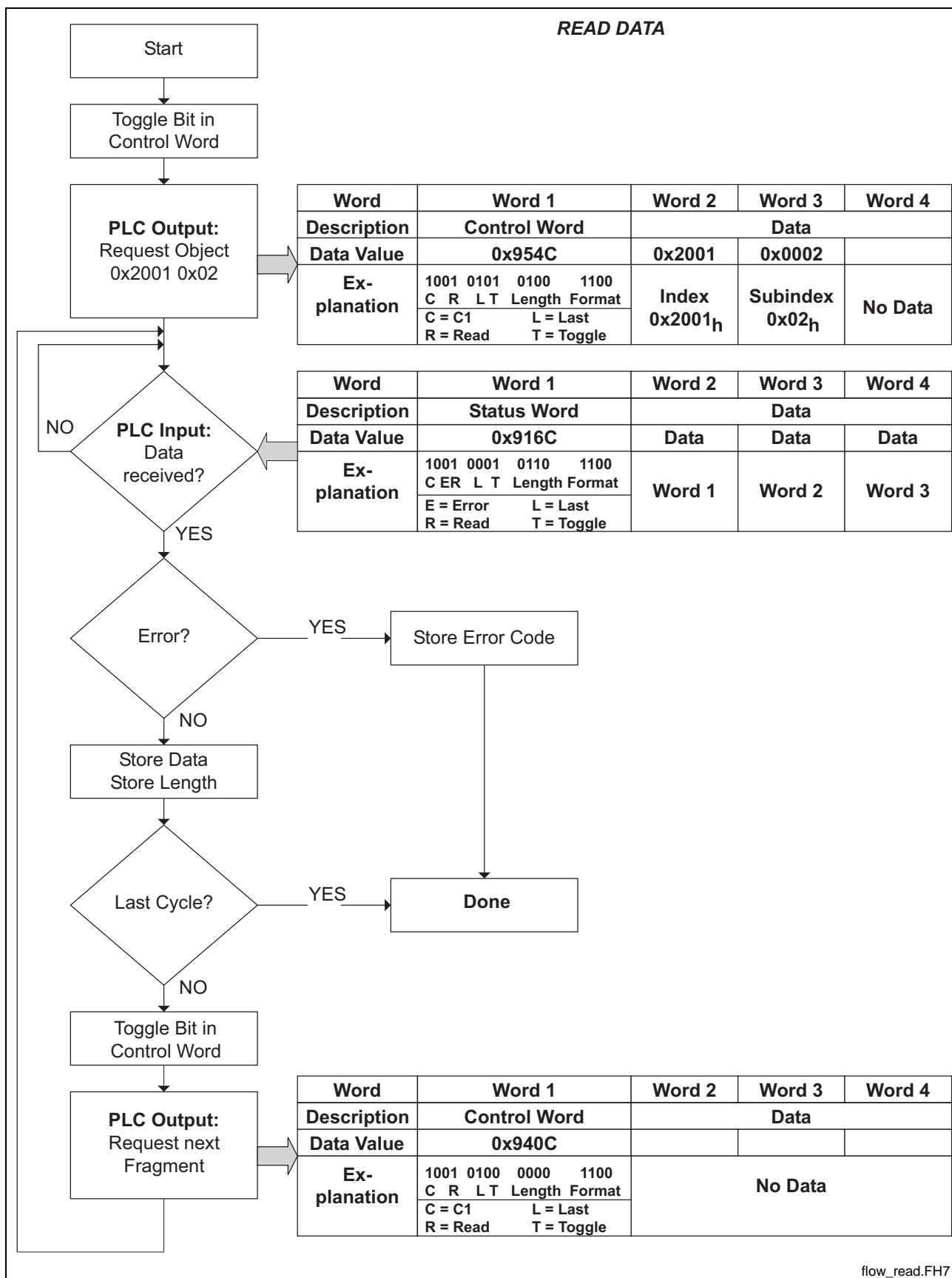


Fig. 6-29: Read Data Object Example

- Canceling Data Transfer** In some cases, it might be necessary to cancel a data transfer. To request a communication reset, the master sends a cancel telegram to the slave.

Word	Word 1	Word 2	Word 3	Word 4
Description	Control Word	Data		
Data Value	0x810F			
Ex- planation	1000 0001 0000 1111 C E R L T Length Format	No Data		
	E = Error L = Last R = Read T = Toggle			

cancel.FH7

Fig. 6-30: Cancel Telegram

The format byte in the command word is set to F_h. The length byte, the L and the R bits are set to 0.

The slave will set its internal state to expect a new command from the master.

Error Messages

If the transmission fails, the slave will respond with an error message as shown below. The status word value can be different for writing. The error bit in the status word is set and the first word contains a 16-bit error code. The toggle bit has the same state as the corresponding request telegram.

Word	Word 1	Word 2	Word 3	Word 4
Description	Status Word	Data		
Data Value	0xA42C	Error code		
Ex- planation	1011 0100 0010 1100 C E R L T Length Format	No Data		
	E = Error L = Last R = Read T = Toggle			

error.FH7

Fig. 6-31: Error Response from Slave

The error code is two bytes long. The high byte specifies the error class and the low byte contains additional information for the application-specific errors (error class 1F_h).

Parameter Channel Error Codes (Hi-Byte)

Error No. (Hex)	Error Description
0x1F	Control-specific error. Refer to Low-Byte for additional error information, which is based on VisualMotion Serial Port Diagnostic Codes (in the VisualMotion Troubleshooting Guide).
0x85	Data length too long (here >128 byte).
0x88	An error occurred during the transmission of data between the PPC-R and the fieldbus slave.
0x8B	Format (bits 0-3 of control word) specified is incorrect.
0x8C	The length set in control byte greater than Parameter Channel.
0x8D	Communication not possible. Parameter Channel too short (<2 bytes).
0x90	The format bits (0-3) of the control word were changed while transmitting several data blocks.
0x95	A read command was issued, but the length field was set to !=0.

Table 6-11: Parameter Channel Error Codes

Handling a Data Exchange Object

When mapped objects are not capable of transferring the desired data, a Data Exchange Object can be used.

The same procedures for writing and reading mapped objects via Short Format 3 apply to the Data Exchange Object.

Selecting a Data Exchange Object

Depending on the length of a VisualMotion ASCII message, any of these data exchange objects can be selected.

Note: The entire data length of the data exchange object must always be transmitted even if the VisualMotion ASCII message is shorter.

For example, if you want to transmit an ASCII message of 42 bytes, you must use object 5E72. To avoid a response error from the fieldbus slave, you must append 22 "Null" characters to the end of the ASCII message to complete a data size of 64 bytes.

Note: The checksum for the VisualMotion ASCII protocol is NOT used with the data exchange object. If the checksum is sent as part of the string, it will be ignored, and no checksum will be sent in the VisualMotion ASCII response messages. To ensure data integrity, the fieldbus protocols support a low-level checksum.

Transmission Sequence via a Data Exchange Object

Note: For the data exchange object, two transmission sequences (and two response sequences) are required, to send the read or write message to and then receive the response message from the PPC-R card.

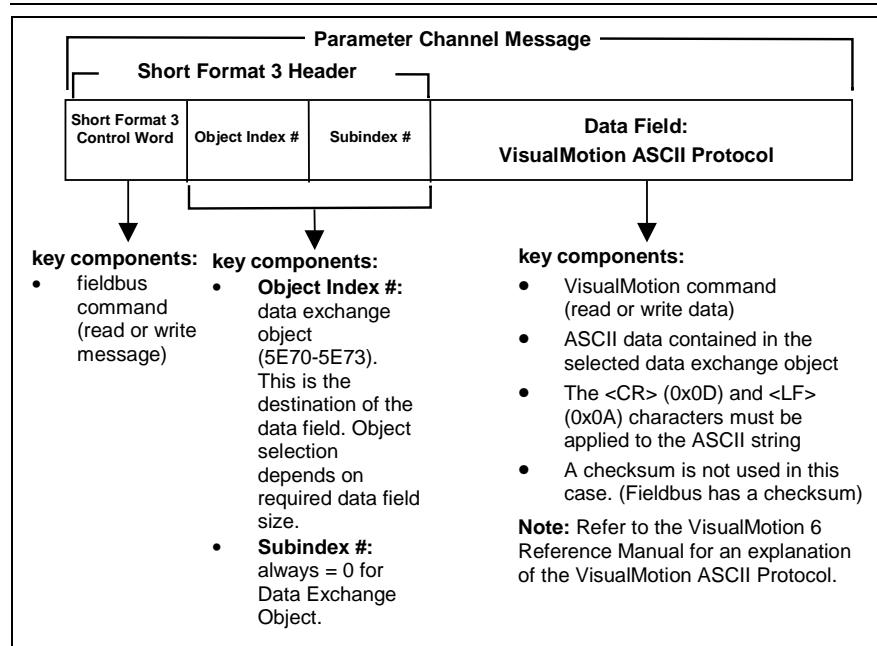


Fig. 6-32: Format of a PK Short Format 3 Message using a Data Exchange Object

The following sequence describes the communication between the fieldbus master (PLC) and the fieldbus slave (PPC-R). For details on reading and writing data in Short Format 3, refer to *Messaging Formats* on page 6-25.

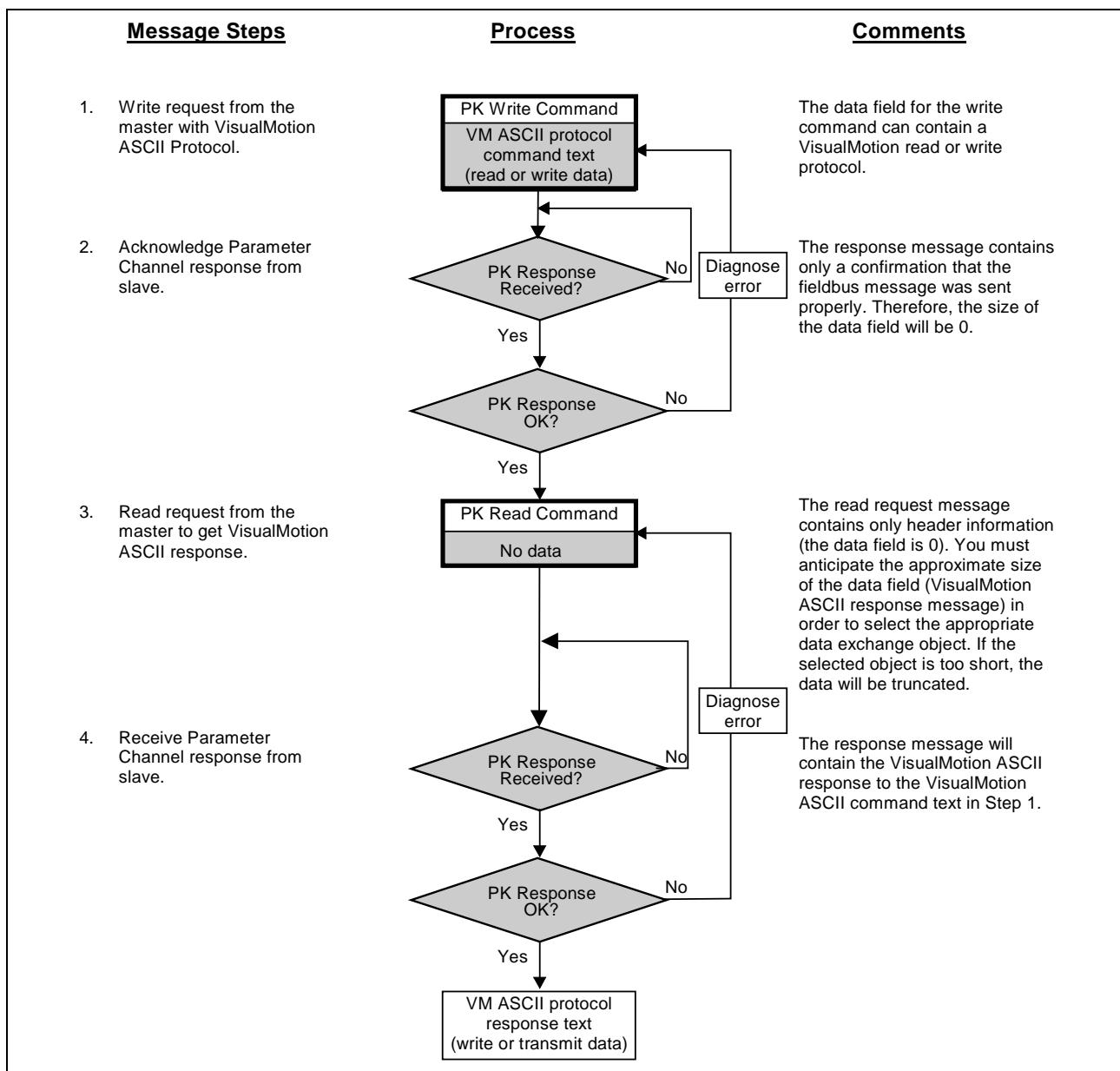


Fig. 6-33: Parameter Channel Short Format 3 Procedure, Using Data Exchange Object

Accessing Mapped Data

Indramat has pre-configured a number of VisualMotion data types to Profibus indexes and subindexes. We call this concept **mapped data**. These data types can be accessed via the Profibus Parameter Channel. The index and subindex for each of these data types can be calculated using the formulas in *Table 6-12* below.

	Object Index #	SubIndex #	Formula
Data Exchange Object	0x5E73	0x00	
	----	----	
	0x5E70	0x00	
<FREE> (349 objects available)	0x5E65	0xFF	
	----	----	(with SubIndex)
	0x5D14	0x01	
Program Integers (Int 1 – Int 5100)	0x5D13	0xFF	Index = 0x5D00 + [(Program Integer – 1) \ 255]
	----	----	
	0x5D00	0x01	SubIndex = Program Integer – [(Index – 0x5D00) * 255]
Program Floats (Float 1 – Float 5100)	0x5CFF	0xFF	Index = 0x5CEC + [(Program Float – 1) \ 255]
	----	----	
	0x5CEC	0x01	SubIndex = Program Float – [(Index – 0x5CEC) * 255]
<FREE> (235 objects available)	0x5CEB	0xFF	
	----	----	(with SubIndex)
	0x5C01	0x01	
Global Integers (GInt 1 – GInt 2550*)	0x5C00	0xFF	Index = 0x5BF7 + [(Global Integer – 1) \ 512]
	----	----	
	0x5BF7	0x01	SubIndex = Global Integer – [(Index – 0x5BF7) * 512]
Global Floats (GFloat 1 – Gfloat 2550*)	0x5BF6	0xFF	Index = 0x5BED + [(Global Float – 1) \ 512]
	----	----	
	0x5BED	0x01	SubIndex = Global Float – [(Index – 0x5BED) * 512]
<FREE> (245 objects available)	0x5BEC	0xFF	
	----	----	(with SubIndex)
	0x5AF8	0x01	
Registers (Reg. 1 – Reg. 2550**)	0x5AF7	0xFF	Index = 0x5AEE + [(Register – 1) \ 255]
	----	----	
	0x5AEE	0x01	SubIndex = Register – [(Index – 0x5AEE) * 255]
T-Parameters (T-0-0001 – T-0-1020)	0x5AED	0x04	Index = 0x56F1 + T-Parameter
	----	----	
	0x56F1	0x01	SubIndex = Task Number
<FREE> (241 objects available)	0x56F0	0xFF	
	----	----	(with SubIndex)
	0x5600	0x01	
C-Parameters (C-0-0001 - C-0-3583)	0x55FF	0x01	Index = 0x4800 + C-Parameter
	----	----	
	0x4801	0x01	SubIndex = 1
A-Parameters (A-0-0001 - A-0-2047)	0x47FF	0x63	Index = 0x4000 + A-Parameter
	----	----	
	0x4001	0x01	SubIndex = Axis Number
P-Parameters (P-0-0001 - P-0-4095)	0x3FFF	0x63	Index = 0x3000 + P-Parameter
	----	----	
	0x3001	0x01	SubIndex = Drive Number

	Object Index #	SubIndex #	Formula
S-Parameters (S-0-0001 - S-0-4095)	0x2FFF	0x63	Index = 0x2000 + S-Parameter
	----	----	
	0x2001	0x01	SubIndex = Drive Number
<Reserved>	0x1FFF	----	
	----	----	
	0x0000	----	

* current limitation: first 512 global integers/floats.

**current limitation: first 512 registers.

Table 6-12: Formulas for Determining Mapped Objects

Example Lookup Tables for Mapped Objects

Card (C) Parameters

The following is an example lookup table for C-Parameters, when using mapped objects.

Example Look-up Chart for:	C-Parameters	CP 0.Y	==>	CP = Card Parameter
				Y = Parameter Number
Index				
SubIndex =				
	0x4801	0x4802	0x4803
	0x48FF	0x4900	0x55FE
	0x55FF			
0x01	CP 0.1	CP 0.2	CP 0.3	
				CP 0.255
				CP 0.256
				CP 0.3582
				CP 0.3583

Table 6-13: Mapped Object Lookup Table for C-Parameters

Axis(A) Parameters The following is an example lookup table for A-Parameters, when using mapped objects. The same formula also applies to SERCOS (S) and Task (T) Parameters.

Example Look-up Chart for:			A-Parameters AP X.Y		==>		AP = Axis Parameter X = Axis Number Y = Parameter Number							
Index														
SubIndex =														
	0x4001	0x4002	0x4003	0x40FF	0x4100	0x47FE 0x47FF						
0x01	AP 1.1	AP 1.2	AP 1.3		AP 1.255	AP 1.256		AP 1.2046 AP 1.2047						
0x02	AP 2.1	AP 2.2	AP 2.3		AP 2.255	AP 2.256		AP 2.2046 AP 2.2047						
0x03	AP 3.1	AP 3.2	AP 3.3		AP 3.255	AP 3.256		AP 3.2046 AP 3.2047						
:	:	:	:	:	:	:	:	:						
:	:	:	:	:	:	:	:	:						
0x63	AP 99.1	AP 99.2	AP 99.3		AP 99.255	AP 99.256		AP 99.2046 AP 99.2047						

Table 6-14: Mapped Object Lookup Table for A-Parameters

Product-Specific (P) Parameters The following is an example lookup table for P-Parameters, when using mapped objects.

Example Look-up Chart for:			P-Parameters PP X.Y		==>		PP = SERCOS P-Parameter (set 0 only) X = Drive Number Y = Parameter Number							
Index = (Class ID && Instance ID for DeviceNet)														
SubIndex = (Attribute ID for DNet)														
	C118, In1	C118, In2	C118, In3	C118, In255	C119, In1	C134, In14 C134, In15						
0x3001	0x3002	0x3003			0x30FF	0x3100		0x3FFE 0x3FFF						
0x01	PP 1.1	PP 1.2	PP 1.3		PP 1.255	PP 1.256		PP 1.4094 PP 1.4095						
0x02	PP 2.1	PP 2.2	PP 2.3		PP 2.255	PP 2.256		PP 2.4094 PP 2.4095						
0x03	PP 3.1	PP 3.2	PP 3.3		PP 3.255	PP 3.256		PP 3.4094 PP 3.4095						
:	:	:	:	:	:	:	:	:						
:	:	:	:	:	:	:	:	:						
0x63	PP 99.1	PP 99.2	PP 99.3		PP 99.255	PP 99.256		PP 99.4094 PP 99.4095						

Table 6-15: Mapped Object Lookup Table for P-Parameters

Integers The following is an example lookup table for Integers, when using mapped objects. The same formula also applies to Floats, Global Integers, Global Floats and Registers.

Example Look-up Chart for:		VM Program Integers	PI 0.Y	==>	PI = Program Integer Y = Program Integer Number
Index					
SubIndex =	0x5D00	0x5D01	0x5D02	0x5D13
	0x01	PI 1	PI 256	PI 511	PI 4846
	0x02	PI 2	PI 257	PI 512	PI 4847
	0x03	PI 3	PI 258	PI 513	PI 4848
	:	:	:	:	:
	:	:	:	:	:
	0xFF	PI 255	PI 510	PI 765	PI 5100

Table 6-16: Mapped Object Lookup Table for Integers

7 DeviceNet and ControlNet Fieldbus Interfaces

7.1 General Information

Version Note:

Information in this document is based on VisualMotion Toolkit software version 08VRS and PPC-R firmware version GPP08VRS (for DeviceNet) or GPP08V16 (for ControlNet).

Note: For fieldbus hardware information, refer to the *VisualMotion 8 Project Planning Manual*.

PPC-R System Description with a Fieldbus

The PPC-R can operate on a serial fieldbus interface (network) by means of a fieldbus expansion card that communicates with the PPC-R via dual-port RAM. The function of the fieldbus card is similar to that of a network card in a PC: it allows communication with other devices on the network.

In Fig. 7-1: Sample Master/Slave Setup with Fieldbus Card below, a commonly described fieldbus interface is pictured:

- **Fieldbus Master** - PLC fieldbus interface
- **Fieldbus Slave** - PPC-R fieldbus interface

In this document, we will refer to the PLC as the **fieldbus master** and the PPC-R as the **fieldbus slave**.

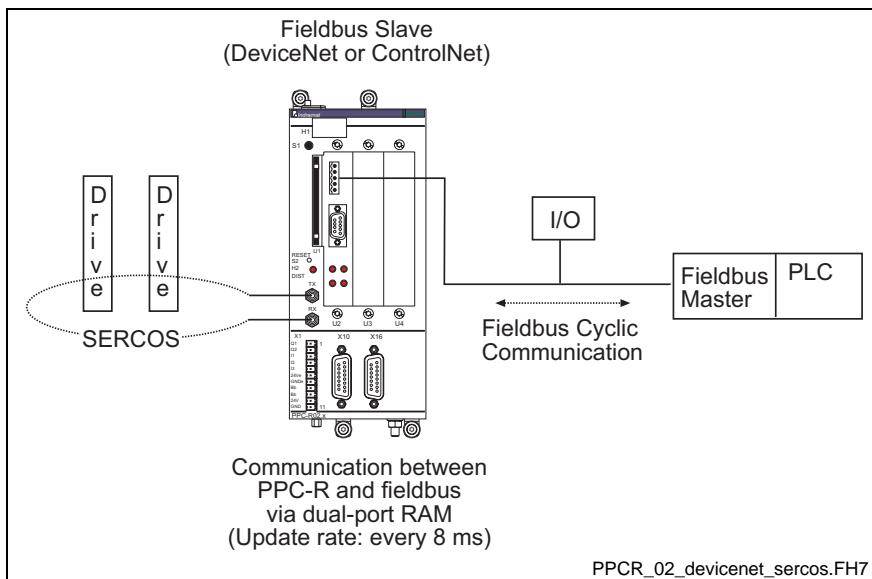


Fig. 7-1: Sample Master/Slave Setup with Fieldbus Card

With the PPC-R, the fieldbus card can be used **only** as a **slave** card in a master/slave setup.

The VisualMotion Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to set up fieldbus configuration and data mapping.

Data Transfer Direction (Output vs. Input)

In the VisualMotion Fieldbus Mapper, output and input are always described with respect to the fieldbus master. The definitions for output and input follow:

output: the communication from the PLC to the PPC-R (i.e. from the fieldbus master to the fieldbus slave).

Synonyms for this type of communication: **send** or **write** data.

input: the communication from the PPC-R to the PLC (i.e. from the fieldbus slave to the fieldbus master).

Synonyms for this type of communication: **receive** or **read** data.

Fieldbus Data Channel Descriptions

The Indramat DeviceNet and ControlNet fieldbus interface cards for the PPC-R support the following communication channels:

- **Cyclic Channel:** Polled I/O (for single and multiplex channels)
- **Non-Cyclic Channel:** Explicit Messaging

Cyclic (Polled I/O) Channel

Cyclic data is user-defined. It is stored in two ordered lists (C-0-2600 for input data, C-0-2601 for output data) and transmitted serially over the bus. In the cyclic channel, data is updated cyclically between the fieldbus master and slave.

The cyclic data channel is limited to 32 input words and 32 output words. PPC-R data types consume these words in either one-word (or 16-bit) groups for PPC-R registers or two-word (or 32-bit) groups for all other data types.

The PPC-R mapping list is scanned every 8 ms and data is sent and received to/from the fieldbus slave board's dual port RAM.

The cyclic data channel can be made up of any combination of the following data types:

- single
- multiplex

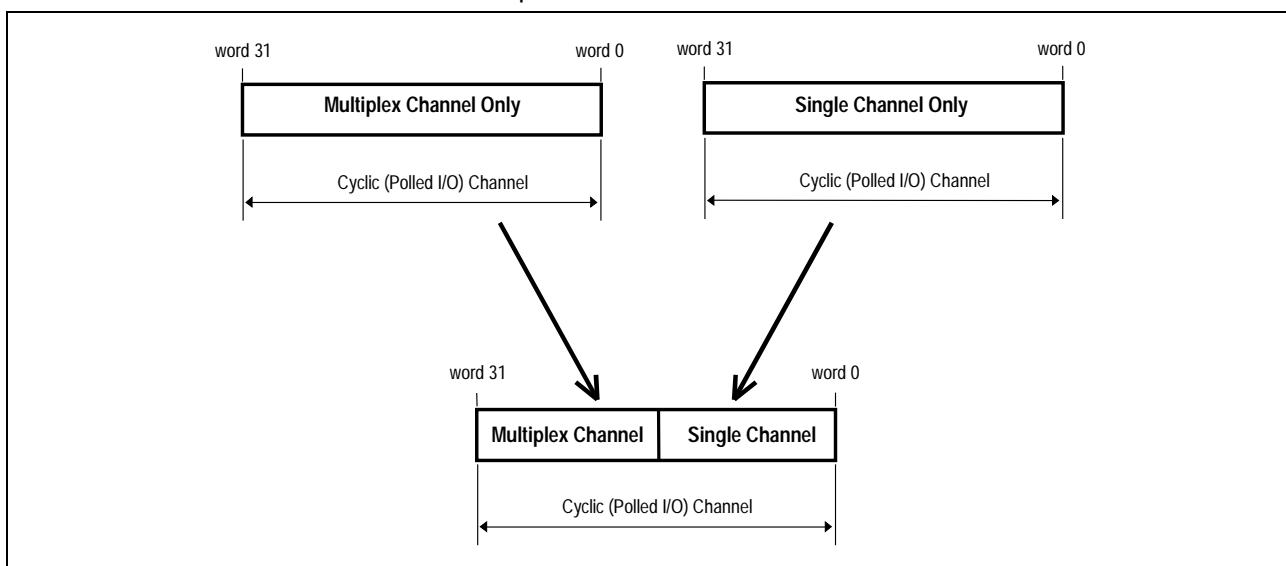


Fig. 7-2: Configuration Options for the Cyclic Data Channel

Cyclic Data: Types and Sizes

The following table outlines the PPC-R data types that can be transmitted via the cyclic channel and the amount of space (in 16-bit data words) that each data type consumes.

- Note:** The cyclic data mapping lists supports only 16- and 32-bit data of the following types for reading and writing:
- Integer
 - Float
 - Binary (used in PPC-R parameters)
 - Hex (used in PPC-R parameters)
- For all other data types (e.g. diagnostic messages - "strings"), Explicit Messaging.

PPC-R Data Type	Data Size (in 16-Bit Words)
Register	1
Program Integer (currently active program ONLY *)	2
Program Float (currently active program ONLY *)	2
Global Integer	2
Global Float	2
Card Parameter	2
Axis Parameter	2
Task Parameter	2
Note: Drive parameters "S" or "P" cannot be transmitted cyclically because of the inherent delay of parameter access over the SERCOS service channel. However, if a drive parameter is mapped to an Axis Parameter, that Axis parameter could be used in cyclic data (see description of Axis Parameters 180-196 in the <i>VisualMotion Functional Description</i>).	
* Important Note: Integers and floats are shown only for the currently active program. Each time you activate a new program, the fieldbus reads/writes to the newly-activated program.	

Table 7-1: PPC-R Cyclic Data Types and Sizes

Single Data Types

Single data types are mapped directly in the cyclic mapping ordered lists (C-0-2600, C-0-2601).

Multiplex Data Types (Cyclic Data Channel)

In some multi-axis applications, 32 words of cyclic data transfer are not sufficient to meet the requirement of the application.

When insufficient data transfer space is available, multiplex data can be set up within the cyclic channel. One multiplex container acts as a placeholder for multiple possible PPC-R data types (all of the same word size). The currently transmitted PPC-R data type is based on an index value placed in a multiplex control or status word attached to the end of the cyclic list. Depending on the index specified by the master, the multiplex channel permits a different set of data within the cyclic channel to be transferred as current real-time data. Multiplex containers can be added to the input and output lists separately and the input and output indexes can be designated separately (in the control and status words).

- Note:** Using the multiplex channel reduces the maximum number of usable words for storing PPC-R data to 31. The 32nd word (or last used word, if fewer than 31 words) is used as the multiplex entry control/status word.

Note: When using VisualMotion 8 with GPP 7 firmware, a maximum of 15 multiplex containers and a maximum of 180 mapping items can be transmitted in the input or output list. This limitation of mapping objects means that you cannot multiplex all 15 containers with all 32 available indexes (=480 items). For VisualMotion 8 with GPP 8 firmware, there is no limitation for multiplexing (each of the first 31 words may be multiplexed with up to 32 indexes).

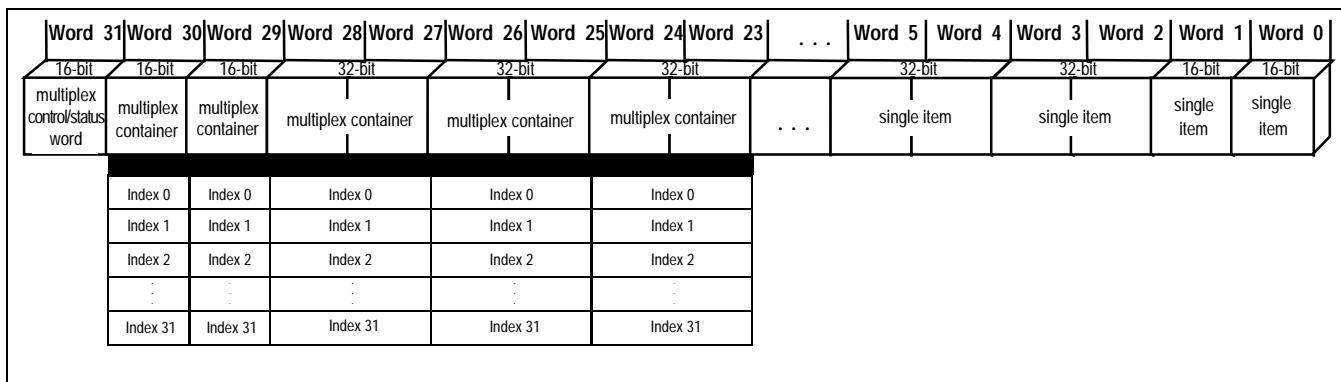


Fig. 7-3: Sample Command (PLC⇒PPC-R) or Response (PPC-R ⇒PLC)

Multiplex Control and Status Words

The multiplex control and status words serve to command and acknowledge multiplex data transferred between the fieldbus master and the fieldbus slave. The **control** word is associated with **output** communication (PLC⇒PPC-R). The **status** word is associated with **input** communication (PPC-R⇒PLC). Single data items are not affected by the multiplex control and status words.

Note: For specific information about how the fieldbus master uses the multiplex control and status words, refer to *Multiplexing* on page 7-21.

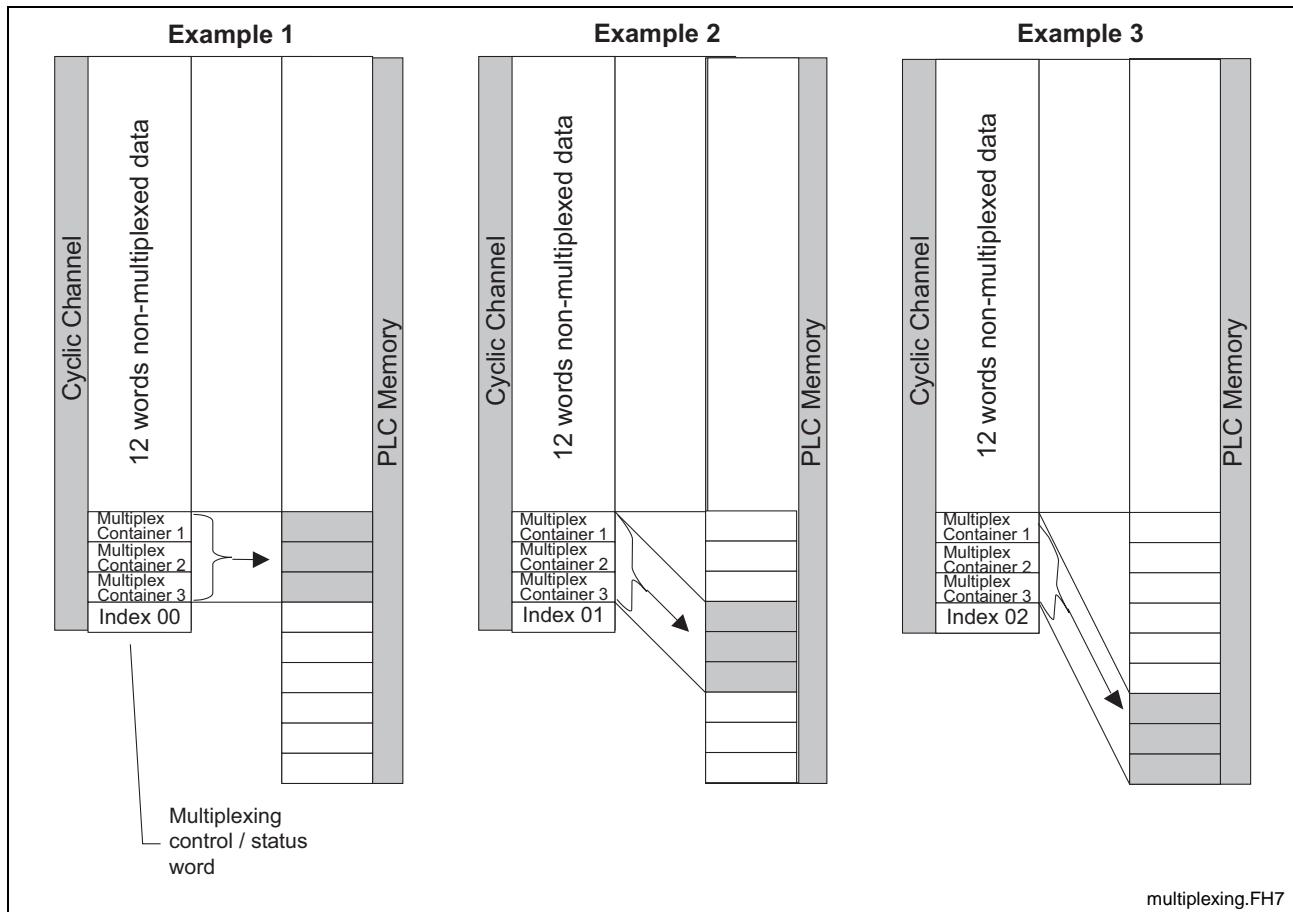


Fig. 7-4: Examples for Reading Data via the Multiplex Channel

Non-Cyclic Channel (Explicit Messaging)

The non-cyclic channel is used for data that needs to be transferred only once or sporadically, such as:

- the transmission of lists
- parametrization of axes or programs
- any non-cyclically mapped data

Instead of being updated during each cycle, non-cyclic data is transferred using a command initiated by the master. Though any data type can be transferred non-cyclically, diagnostic messages and drive parameters (S and P) **must** be transferred non-cyclically because of the non-cyclic retrieval for drive parameters through SERCOS and the length of the diagnostic messages.

There are two types of non-cyclic data transmissions for the PPC-R/VisualMotion system:

- mapped data (directly to PPC-R data types)
- data exchange object

Non-cyclic data can be accessed via Explicit Messaging support of the Fieldbus master.

Mapped Data

Mapped data is the most powerful feature of the PPC-R non-cyclic fieldbus interface. Through mapped data, the user has access to virtually every PPC-R parameter over the fieldbus. It is easy to implement from the PLC side and requires no setup on the PPC-R side.

Data Exchange Objects

Four data exchange objects Class 100, Instance 1-4, Attribute 100 are available for the transfer of non-cyclic data. These objects represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the PPC-R card, in the same way that data is transferred using the VisualMotion ASCII Format via an Explicit Message. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. For more specific information about these objects, refer to *Data Exchange Objects* on page 7-30.

7.2 Fieldbus Mapper Functionality

Initializing the Fieldbus Mapper from VisualMotion 8

1. Open an existing program or create a new program. You must be using PPC-R hardware with GPP firmware to use the Fieldbus Mapper described in this document.

Note: Make sure the VisualMotion system is configured for GPP (in the **Settings** ⇒ **Configuration** menu item of the main VisualMotion screen).

2. Select **Commission** ⇒ **Fieldbus Mapper**. The main Fieldbus Mapper screen appears (refer to *Fig. 7-5* below).

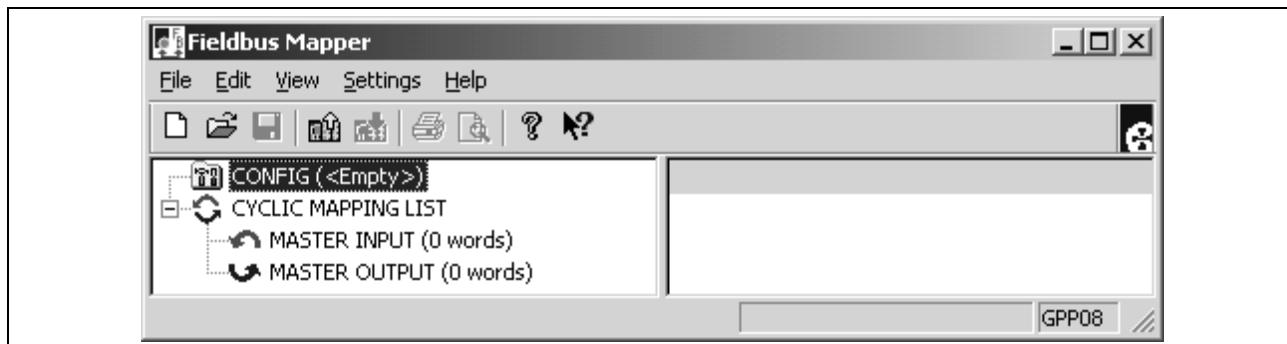


Fig. 7-5: Fieldbus Mapper Main Screen (Blank)

Creating a New Fieldbus Mapper File

1. Click or select **File** ⇒ **New**. A “setup wizard” goes through three steps:
 - Fieldbus Slave Definition
 - Fieldbus Slave Configuration
 - Cyclic Data Configuration
2. Enter the information requested in the setup screens. For more details on each step, refer to *Fieldbus Slave Definition*, *Fieldbus Slave Configuration*, and *Cyclic Data Configuration* for detailed information about each configuration step.
3. Save the file (automatically has a .prm extension).

Editing an Existing Fieldbus Mapper File

1. Click or select **File** ⇒ **Open**.
2. Browse to find the desired file (*.prm extension).
3. Click **Open**. The main Fieldbus Mapper screen appears, which lists the configuration information. Refer to *Fig. 7-6* below.

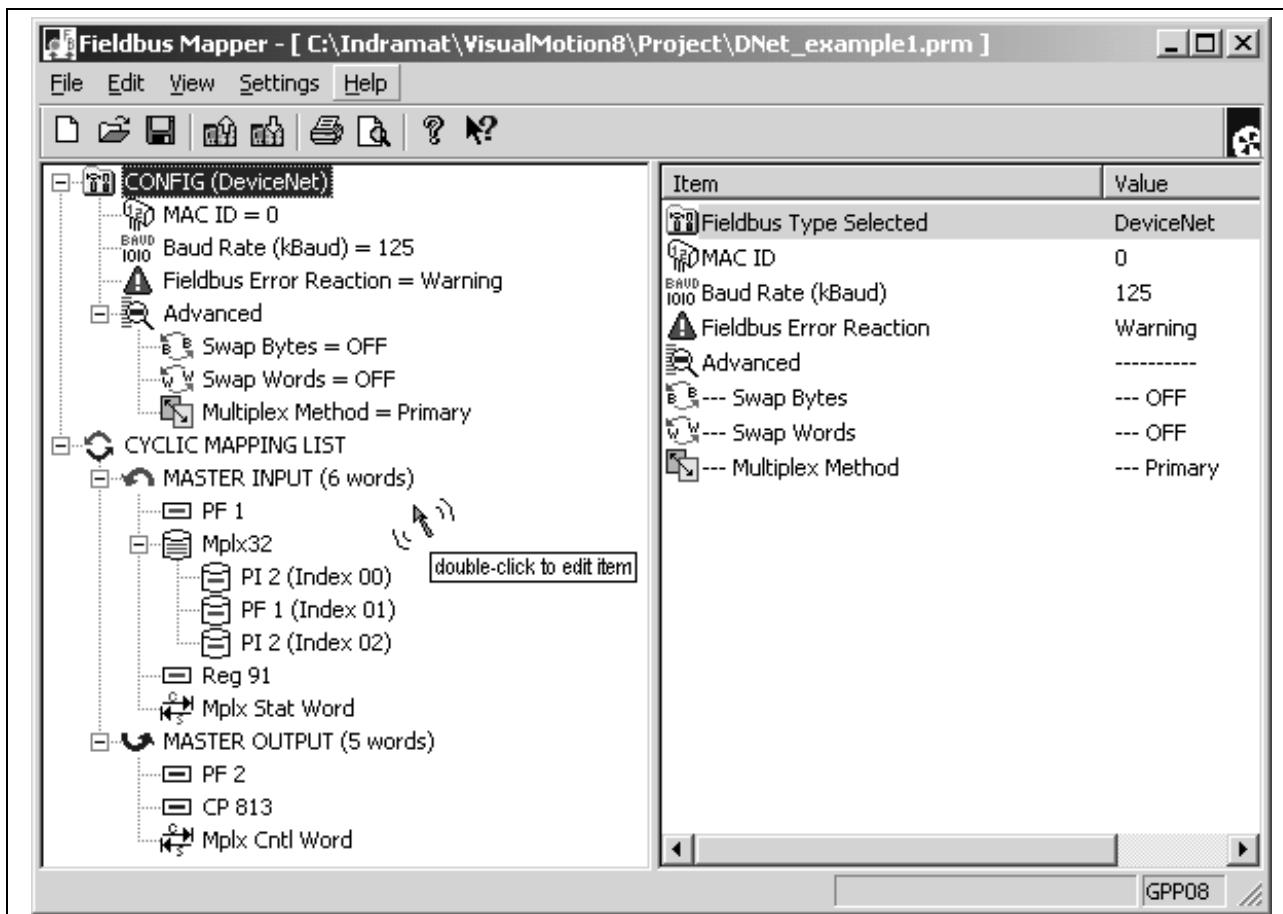


Fig. 7-6: Fieldbus Mapper Main Screen (Complete)

- From the Fieldbus Mapper main screen, **double-click** on the specific item to be edited. The corresponding setup screen appears.

- Or -

Select the item to edit from the Edit menu (refer to Fig. 7-7 below). For more information about each step, refer to *Fieldbus Slave Definition*, *Fieldbus Slave Configuration*, and *Cyclic Data Configuration* for detailed information about each configuration step.

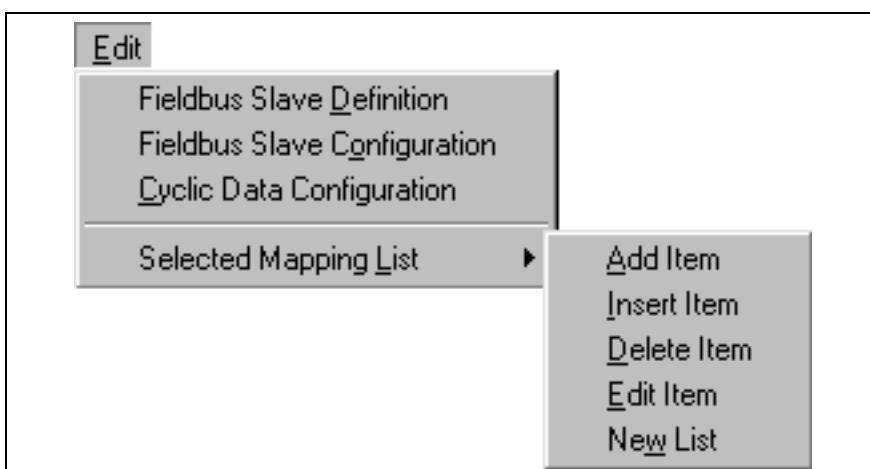


Fig. 7-7: Fieldbus Mapper Edit Menu

Note: You can also directly add, insert, delete, edit an item, or create a new list by:

- clicking on the item to be edited in the main Fieldbus Mapper screen and selecting the desired function under **Edit ⇒ Selected Mapping List**
- right-clicking on an item to display a menu of functions

Fieldbus Slave Definition

Note: The **ControlNet** Fieldbus Type is supported only by Hardware Platform **PPC-R (GPP08vRS)**.

From the Fieldbus Slave Definition window, select the desired Hardware Platform and **DeviceNet** or **ControlNet** as the Fieldbus Type (refer to *Fig. 7-8* below).

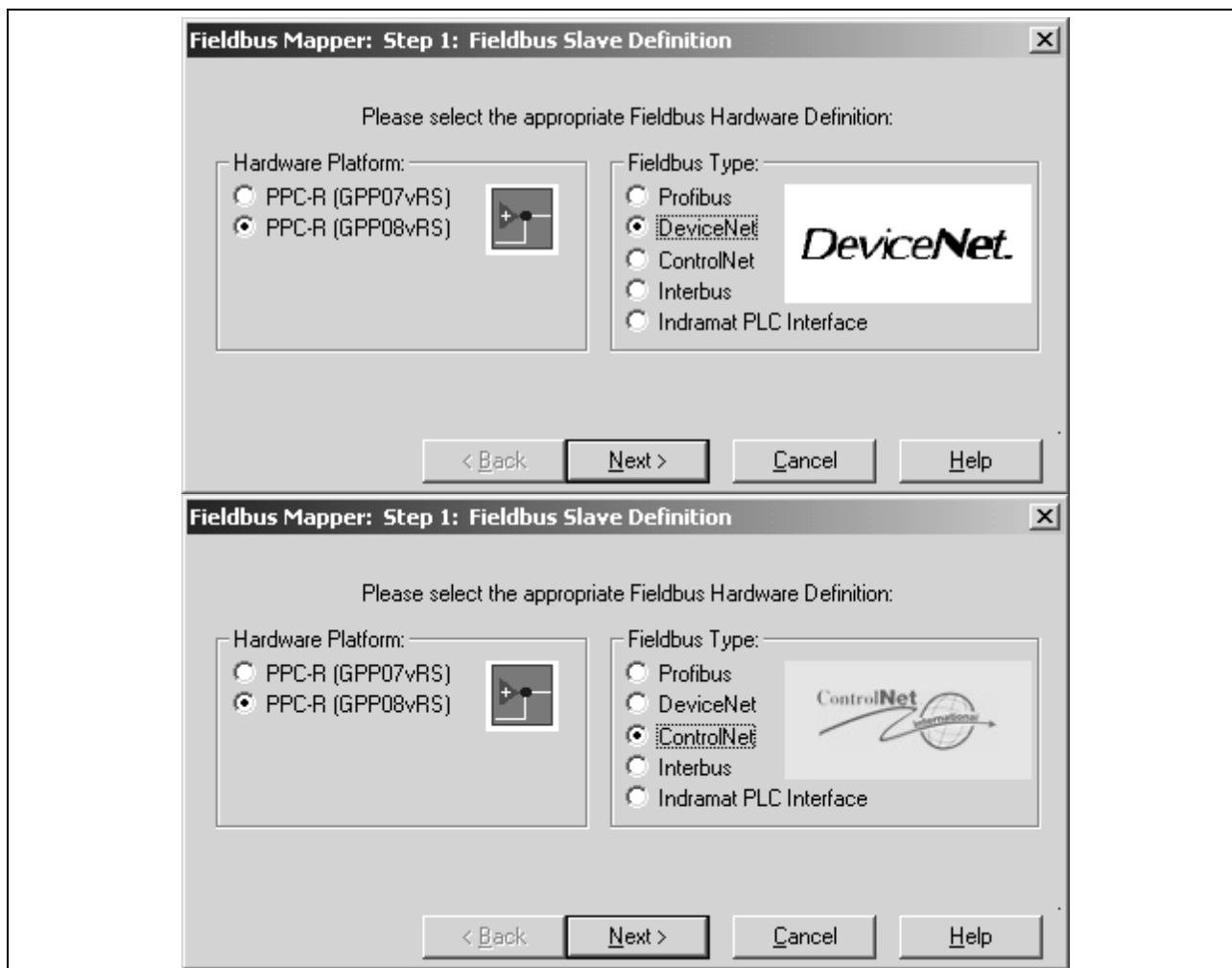


Fig. 7-8: Fieldbus Slave Definition Window

Fieldbus Slave Configuration

The Fieldbus Slave Configuration screens for DeviceNet and ControlNet are shown in figure *Fig. 7-9* below.

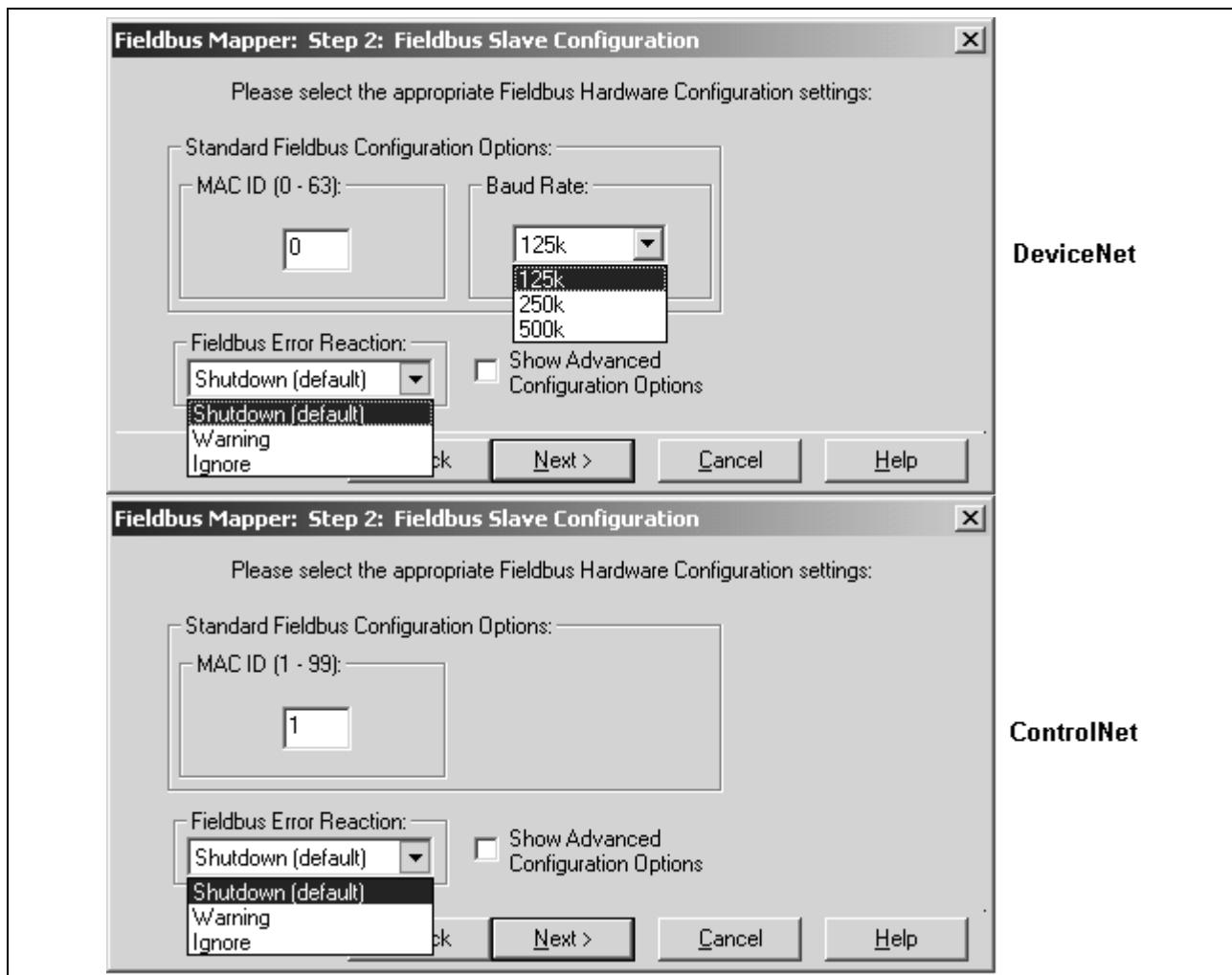


Fig. 7-9: Fieldbus Slave Configuration

Standard Fieldbus Configuration Options

- **MAC ID (0-63 for DeviceNet, 1-99 for ControlNet):** set to a unique number for this device on the bus.

- **Baud Rate (DeviceNet only):** set to match that of the master.

Fieldbus Error Reaction

Set the Error Reaction to Shutdown (default), Warning or Ignore. Refer to **Fieldbus Error Reaction** on page 7-19 for detailed information about each setting.

Advanced Configuration Options

The **Advanced Options:** are shown only if the checkbox next to *Show Advanced Configuration Options* is checked (refer to *Fig. 7-10* below). In most cases, the default options should apply.

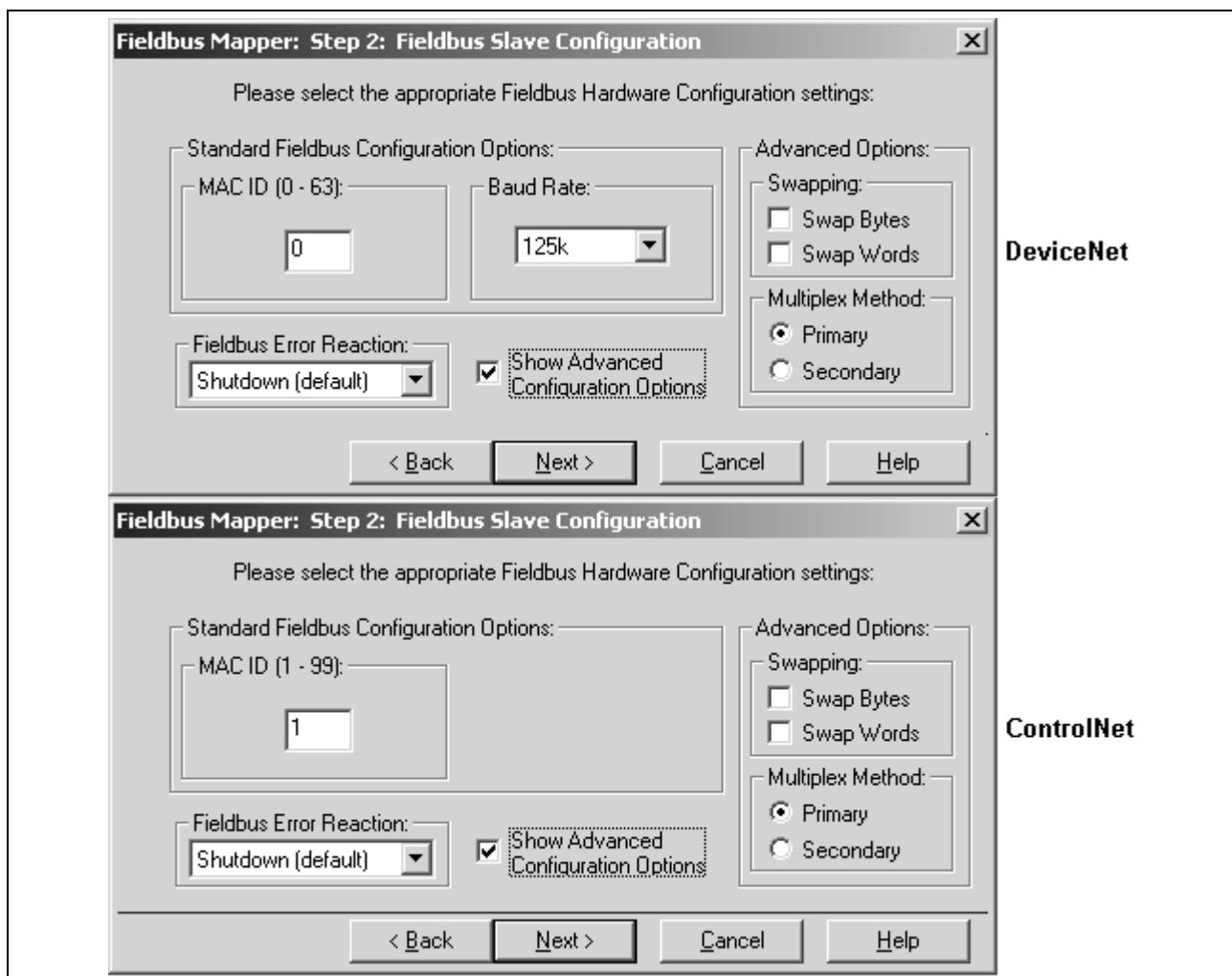


Fig. 7-10: Fieldbus Slave Configuration: Advanced

- **Swapping:** If word and byte swapping is required by your PLC, select the checkboxes next to "Swap Bytes" and "Swap Words." Bytes and words are not swapped if the boxes are left unchecked. Refer to *Word and Byte Swapping* on page 7-20.

Note: When the Allen-Bradley 1747-SDN (DeviceNet Scanner) Module for the SLC-Series PLC is used, both **Swap Bytes** and **Swap Words** can be checked, so the order of resulting data appears correctly.

- **Multiplex Method:** select Primary or Secondary (Primary is the default). Select Secondary only if you have an inconsistent fieldbus master. Refer to *Multiplexing* on page 7-21 for detailed information about each method.

Cyclic Data Configuration

An example of the Cyclic Data Configuration screen is shown in *Fig. 7-11* below. No data has yet been configured. If you are editing an existing Fieldbus Mapper file, the list will probably contain more items.

First, you must select the Cyclic Input List (from PPC-R to PLC) or the Cyclic Output List (from PLC to PPC-R).

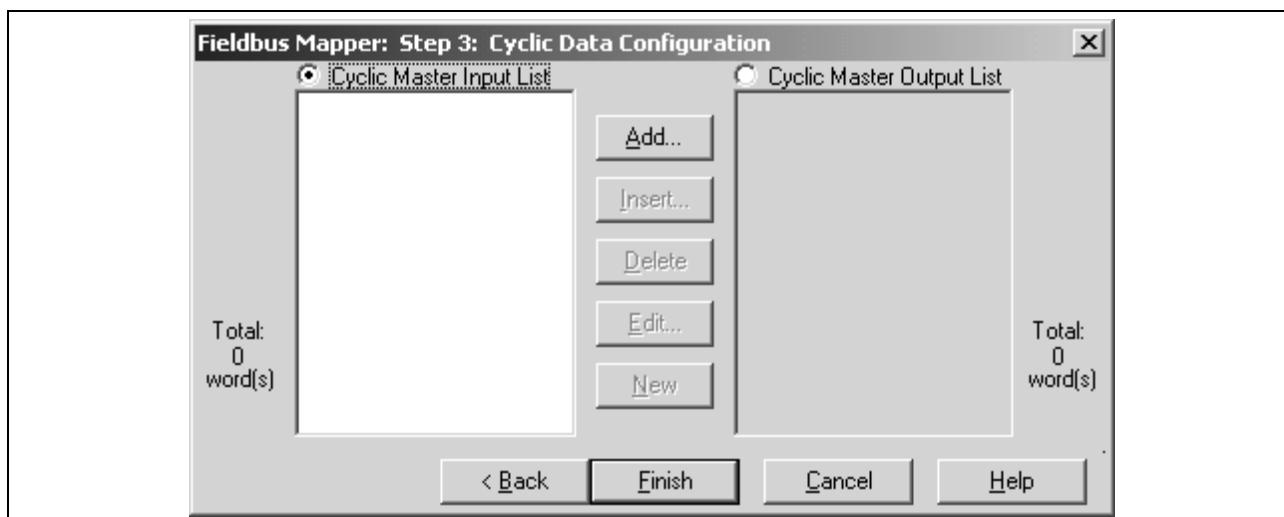


Fig. 7-11: Cyclic Data Configuration—Initial Screen

Adding an Item to the List

1. Select the Cyclic Input List or the Cyclic Output List.
2. Click Add. The screen in *Fig. 7-12* below appears. Select the Data Type (for example, Register).

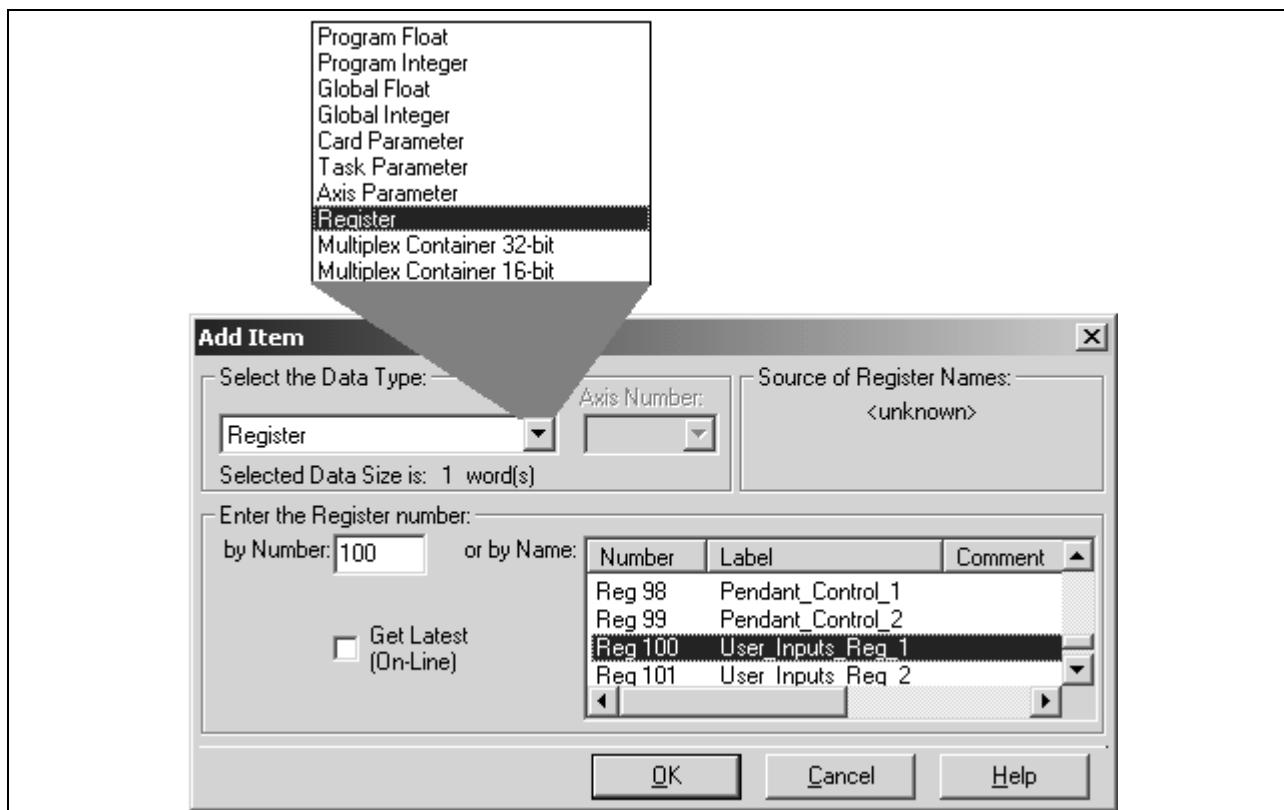


Fig. 7-12: Add Item to Cyclic Data

Note: Registers and 16-bit Multiplex Containers (used only for Registers) require one data word (16 bits), and all other data types require two data words (32 bits) of space.

3. Enter the required information (for example Register Number) or select it from the list below. Only the available data types for your designated VisualMotion hardware setup and fieldbus type are listed.

Note: If you check the box next to "Get Latest (On-Line)," the data type label list is updated based on your firmware version and the currently active program.

4. Click OK to add the selected item to the list.

Adding Multiplex Containers to the List

1. Select the Cyclic Input List or the Cyclic Output List.
2. Click Add.
3. In the *Add Item* screen under *Select the Data Type*: select Multiplex Container 16-bit (for Registers) or Multiplex Container 32-bit (for all other data types).
4. Click OK to add the Multiplex Container to the List. The screen in *Fig. 7-13* below is an example where a 16-bit Multiplex Container and a 32-Bit Multiplex Container have been added.

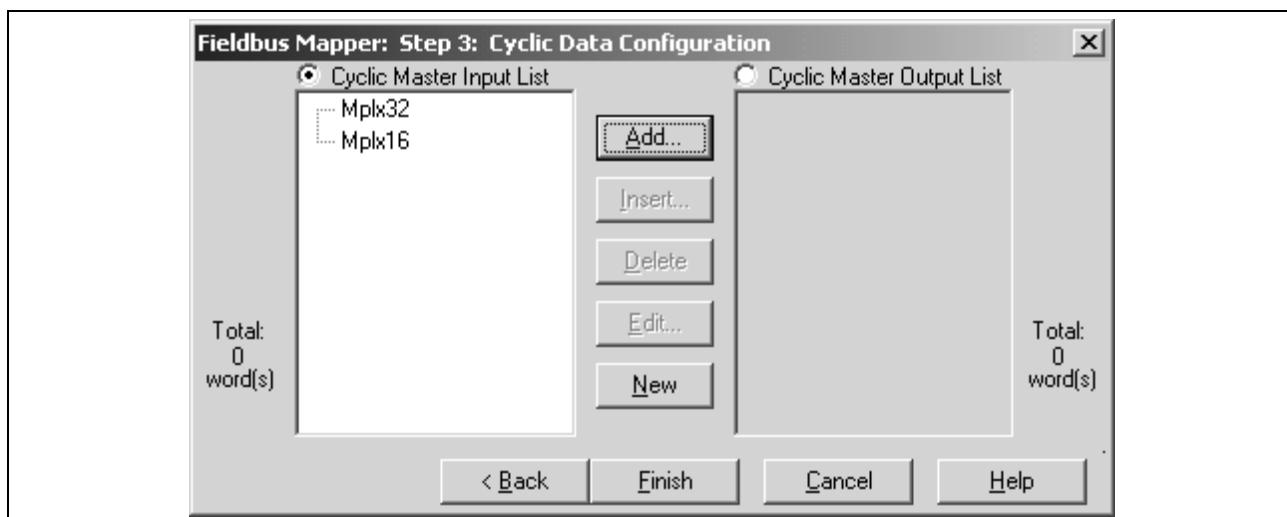


Fig. 7-13: Cyclic Data Configuration, Multiplex Containers

Note: At this point, the Multiplex Containers do not yet contain any items. To add multiplex items refer to *Adding Items to an Empty Multiplex Container* below.

Adding Items to an Empty Multiplex Container

1. In the *Cyclic Data Configuration* screen, select the multiplex container to which you want to add items.
2. Click Add. The screen in *Fig. 7-14* below appears. Because it is unclear whether you would like to add to the list or to the multiplex container, the Fieldbus Mapper is requesting clarification.

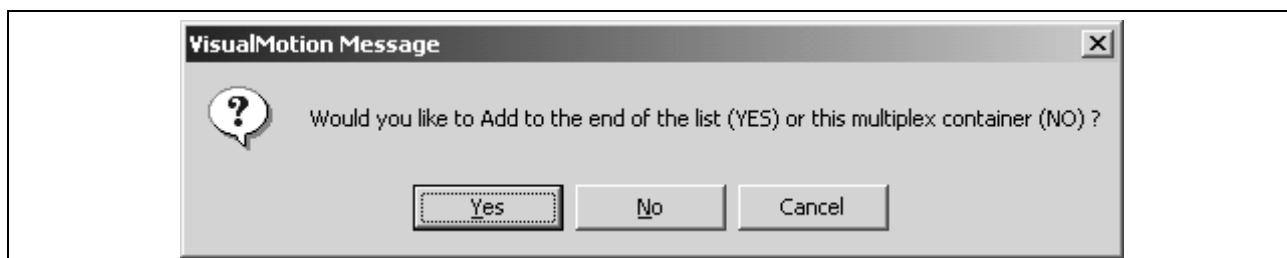


Fig. 7-14: Add Item or Multiplex Item Screen

Note: For subsequent items, highlight any of the indexes within the multiplex container before clicking Add, and the Fieldbus Mapper will know you want to add to that container.

3. To add to the selected multiplex container, click No. The screen in *Fig. 7-15* below is an example for adding a 32-bit multiplex item.
4. Select the desired item to be added to the multiplex container.

Note: In addition to the data types that can be added to the multiplex list, an empty item called *Multiplex Empty Item* is available to fill a space within the multiplex container, if nothing is to be mapped to a particular index.

5. Click OK. The item is automatically placed in the multiplex container as the next unassigned index item (e.g. the first item is index 00, the last is index 31).
6. Repeat for as many items as you want to add to the multiplex container, up to 32 items.

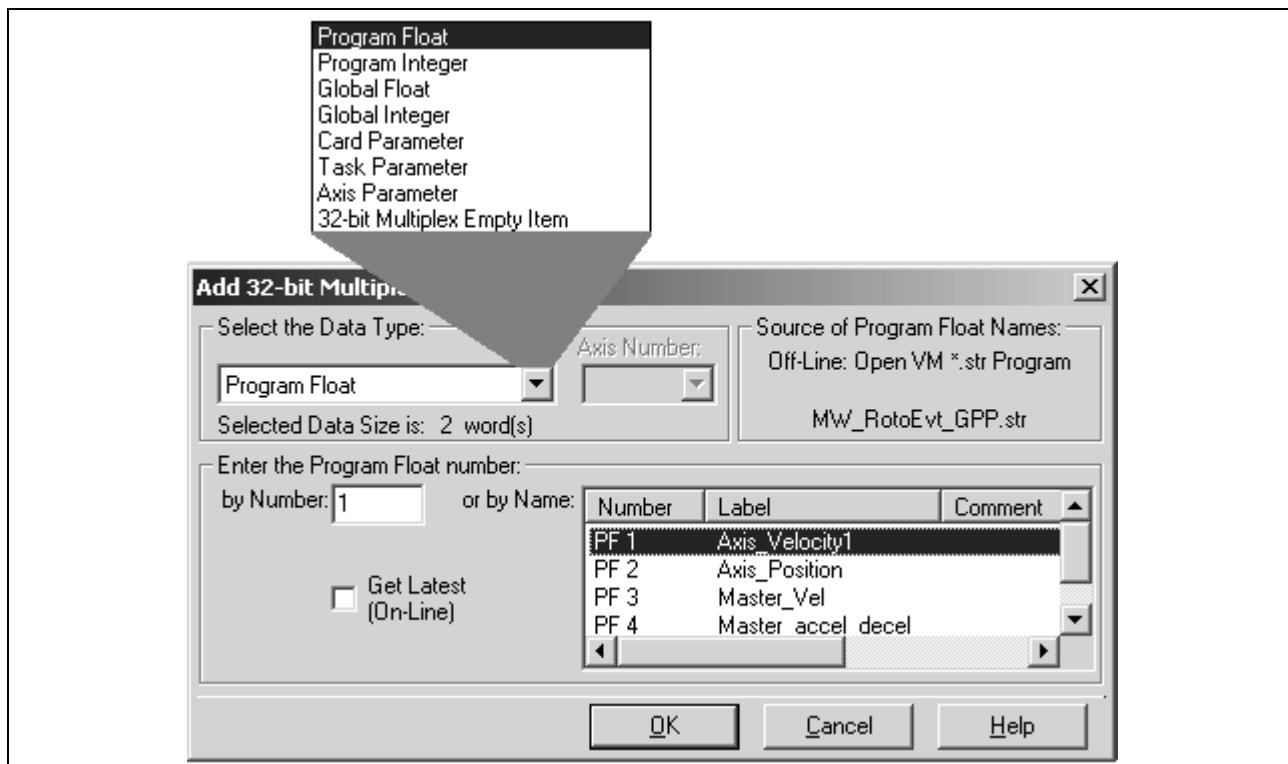


Fig. 7-15: Adding a Multiplex Item to the Container (32-bit example)

Editing the Cyclic Data Lists

To make changes to an existing list, use the following buttons:

Button	Function
 Add...	Inserts a new item at the end of the list.
 Insert...	Inserts a new item into the list directly before the selected item.
 Delete	Removes the selected item from the list.
 Edit...	Allows editing of the selected item. (To edit a list item, you may also double-click on it.)
 New	Clears up the current list.

Table 7-2: Button Functions in the Cyclic Data Configuration Window

Additional Functions

Several additional functions are now available in the Fieldbus Mapper:

Function	Icon	Menu Selection
Download the current fieldbus configuration from the PPC-R		<i>File</i> ⇒Get Fieldbus Configuration from PPC
Upload the current fieldbus configuration in the Fieldbus Mapper to the PPC-R		<i>File</i> ⇒Send Fieldbus Configuration to PPC
Print the current fieldbus configuration data		<i>File</i> ⇒Print
Preview the printout of the current fieldbus configuration data		<i>File</i> ⇒Print Preview

Table 7-3: Additional Functions

Getting the Fieldbus Configuration from the PPC

After getting the fieldbus configuration from the PPC, the following information is detected by the system and appears in the configuration list:

- Fieldbus Type Found
- Fieldbus FW (Firmware) Version
- GPP Control FW (Firmware) Version

An example for DeviceNet is shown in *Fig. 7-16* below. For ControlNet fieldbuses, the Fieldbus Type would be changed and the Baud Rate would not appear.

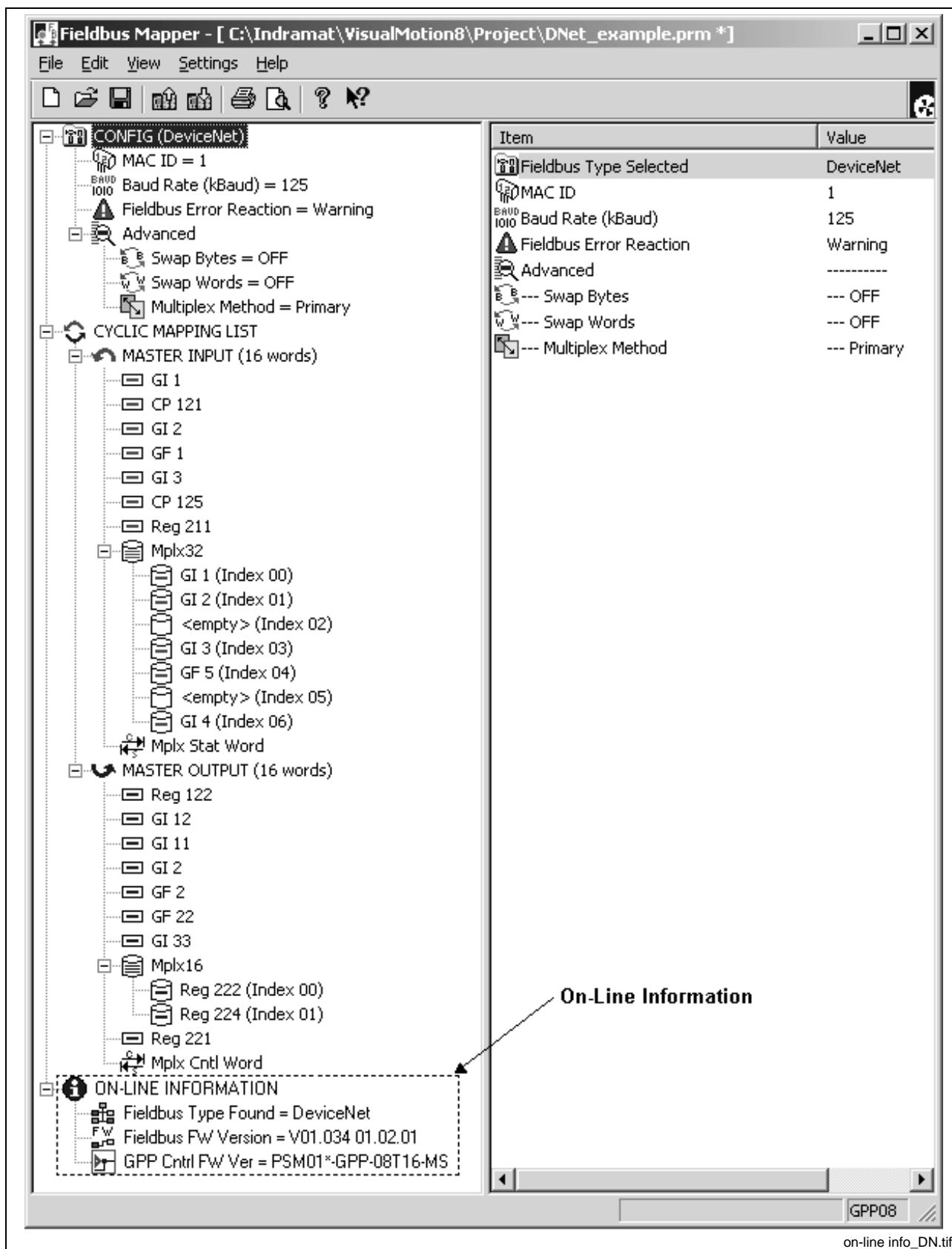


Fig. 7-16: On-Line Fieldbus Configuration Information (DeviceNet Example)

7.3 Information for the GPP Programmer

Register 19 Definition (Fieldbus Status)

VisualMotion Register 19 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 7-4 below contains the bit assignment for the diagnostic object 5ff2. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	---	---	---	---	---	---	---	---	---	x5	x4	---	x2	x1

Table 7-4: Bit Assignment for VisualMotion Register 19

Bit Definitions

x1, x2 Status bits for the internal DPR (Dual-Port RAM) communication between the fieldbus slave and the PPC-R:

x1: FB Init OK , LSB (least significant bit)

x2: FB Init OK, MSB (most significant bit)

The bit combinations for x1 and x2 are as follows:

Bit 2 (PPC-R)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, or neither the PPC-R nor the fieldbus card have 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 7-5: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

x4 Status bit for the active bus capabilities of the fieldbus slaves (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. Refer to *Fieldbus Error Reaction* on page 7-19 for an explanation of the Fieldbus Error Reaction setting.

0--> The fieldbus slave is not (yet) ready for data exchange.

1--> The fieldbus slave can actively participate on the bus.

x5 Status bit for the non-cyclic channel (Explicit Messaging) (Non-Cyclic Ready)

0--> The non-cyclic channel (Explicit Messaging) cannot (yet) be used.

1--> The non-cyclic channel (Explicit Messaging) is ready for use by the fieldbus master.

x15 Status bit for the cyclic data output (Cyclic Data Valid):

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 20 Definition (Fieldbus Diagnostics)

VisualMotion Register 20 holds the information for "Fieldbus Diagnostics."

Table 7-6 below contains the bit assignment for the diagnostic object 5ff0. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---".

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	---	---	---	---	---	---	---	---	---	---	---	---

Table 7-6: Bit Assignment for VisualMotion Register 20

Bit Definitions

x13 - x16 Identification of the fieldbus interface card (FB Card Found)

The bit combinations for x13, x14 and x15 are as follows:

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 7-7: Identification of the Fieldbus Interface

Register 26 Definition (Fieldbus Resource Monitor)

The "Fieldbus Resource Monitor" in register 26 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 26 is divided into the following three counter types:

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

Table 7-8 below contains the bit assignment for the fieldbus counters. The assigned bits are labeled with an "x" followed by the bit number.

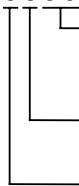
Current Miss Counter				Peak Miss Counter				Fieldbus Time-out Counter							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1

Table 7-8: Bit Assignment for Fieldbus Resource Monitor (Register 26)

Note: View register 26 in Hexadecimal format too more easily monitor the fieldbus counters.

Hex display format of register 26

0x0000



Fieldbus Time-out Counter (00-FF)
Increments by one every time 10 cyclic mapping list attempts are missed in 10 attempts.

Peak Miss Counter (0-9)
Displays the peak value of the Current Miss Counter.

Current Miss Counter (0-9)
Displays the count of missed cyclic mapping list attempts out of the last 10 attempts

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

Bit Definitions

x13 - x16 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.

x9 - x12 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.

x1 - x8 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. Refer to *Fieldbus Error Reaction* below for an explanation of the Fieldbus Error Reaction setting.

Note: The GPP programmer can monitor the "Current Miss Counter" and define a custom error reaction for missed mapping list update attempts less than 10.

Note: The values in register 26 are read/write and can be reset by the user.

Fieldbus Error Reaction

Note: The Fieldbus Error Reaction setting is active only in SERCOS Phase 4. In all other SERCOS phases, it will be inactive.

You can select how you would like the PPC-R system to react in case of a fieldbus error. This reaction can be set in the "Fieldbus Slave Configuration" 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 stored in Parameter C-0-2635:

Setting	Value in Parameter C-0-2635
Shutdown	0 (default)
Warning Only	1
Ignore	2

Table 7-9: Parameter C-0-2635 Values for Error Reaction Settings

Fieldbus Mapper Timeout

The Fieldbus Mapper continually scans the system for sufficient resources to process the cyclic data mapping lists (2600 and 2601 lists). If 10 out of 10 attempts of the mapping list update are missed, the system is considered to have insufficient resources and the selected error reaction is evoked, as follows:

If "Shutdown" (0) is set in Parameter C-0-2635, the following error is generated from the PPC-R card: **520 Fieldbus Mapper Timeout**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated: **209 Fieldbus Mapper Timeout**

If "Ignore" (2) is set in Parameter C-0-2635, the system will update as resources become available, but there is no way to monitor whether or not updates actually occur.

Lost Fieldbus Connection

Register 19, bit 4 indicates the status of the fieldbus. Refer to *Register 19 Definition (Fieldbus Status)* for more specific bit information. The system monitors this bit and evokes the selected error reaction if the bit is low (0), after a fieldbus card is found. A typical situation that will cause this condition is the disconnection of the fieldbus cable from the fieldbus card.

If "Shutdown" (0) is set in Parameter C-0-2635, the following error is generated from the PPC-R (active in SERCOS Phase 4 only):

519 Lost Fieldbus Connection

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated (active in SERCOS Phase 4 only):

208 Lost Fieldbus Connection

If "Ignore" (2) is set in Parameter C-0-2635, there is no noticeable reaction when Register 19 Bit 4 goes low, unless the GPP application program is customized to evoke a special reaction.

Troubleshooting Tip:

If a fieldbus card is not found on the system, the Error Reaction setting will be ignored. If you have a fieldbus card and the Error Reaction is not responding as expected, the system may not "refer to" your fieldbus card.

7.4 Information for the PLC Programmer

*.eds File

Indramat supplies an *.eds file containing supporting information for the PPC-R with a DeviceNet or ControlNet slave configuration. This file is provided on the VisualMotion 8 installation CD.

Word and Byte Swapping

In the Fieldbus Mapper, it is possible to enable automatic word and byte swapping for DeviceNet and ControlNet fieldbuses (for both input and output), depending on the type of PLC used.

- **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.

Example: Allen-Bradley 1747-SDN Module for the SLC-Series PLC

When the Allen-Bradley 1747-SDN (DeviceNet Scanner) Module for the SLC-Series PLC is used, both **Swap Words** and **Swap Bytes** can be checked in the Fieldbus Mapper, so the order of resulting data appears correctly.

Multiplexing

Primary Multiplex Method (for Consistent Masters only)

Important: You should use the Primary Multiplex Method only for a master that is consistent over the entire cyclic channel. The Secondary Multiplex Method is available for inconsistent masters. Refer to *Explanation of the Master Consistency Problem* on page 7-24.

The advantage of the Primary Method is easier handling of input data for consistent masters.

Control Word and Status Word

Control Word

The control word is transferred in the multiplex channel from master to slave. It tells the slave in which index the data is being transferred from master to slave and in which index the data is requested from slave to master.

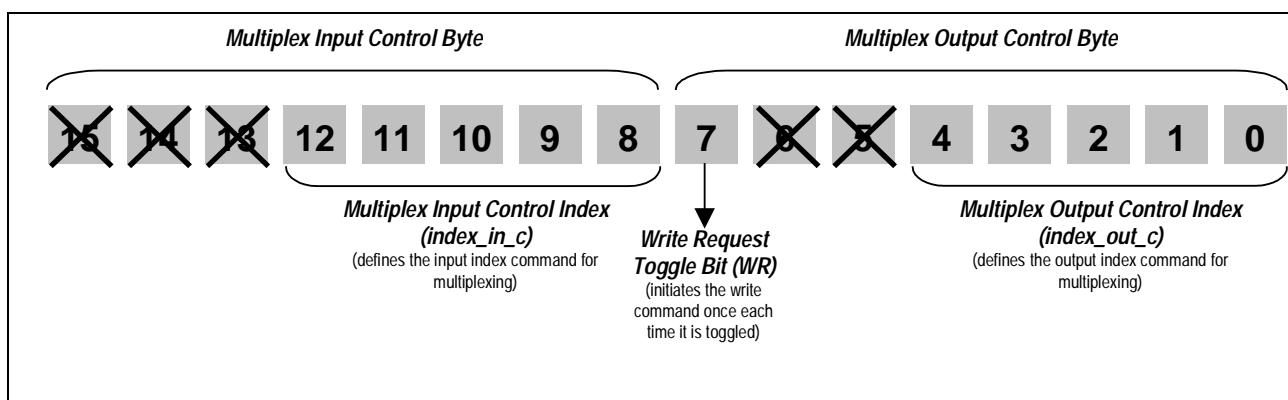


Fig. 7-17: Control Word Definition, Primary Multiplex Method

- **Index_out_c:** tells the slave in which index the data are transferred from master to slave (out = master -> slave, _c = element of control word).
- **Index_in_c:** tells the slave in which index the data is requested from slave to master (in = slave -> master, _c = element of control word).
- **WR (Write Request):** handshake bit (refer to meaning of WR and WA).

Note: Input data via the Multiplex Channel is continually being updated.

Status Word The status word is transferred in the multiplex channel from slave to master. It acknowledges the written index and the requested index.

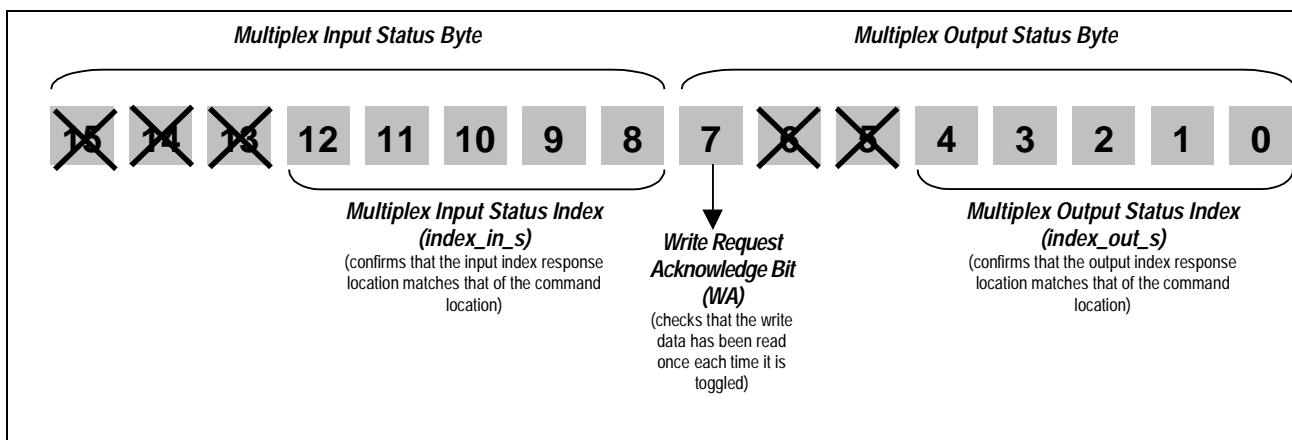


Fig. 7-18: Status Word Definition, Primary Multiplex Method

- **Index_out_s:** acknowledges index written by the master (out = master -> slave, _s = element of status word).
- **Index_in_s:** tells the master which index is transferred from slave to master in the actual process data cycle (in = slave -> master, _s = element of status word).
- **WA (Write Acknowledge):** Handshake bit (refer to meaning of WR and WA).

Handshake Bits WR and WA

WR and WA are handshake bits that allow the controlled writing of data via the multiplex channel. WR and WA control the data transfer for writing data_out (data send from master to slave).

WR == WA:

- tells the master that the slave has received the last multiplex data_out. The master can now send new data_out.
- tells the slave to do nothing, because the master has not yet put new consistent data_out on the bus.

WR! = WA:

- tells the slave to do something, because the master has now put consistent new data_out on bus.
- tells the master to do nothing, because the slave has not yet received the latest multiplex data_out.

Master Communications (Primary Multiplex Method)

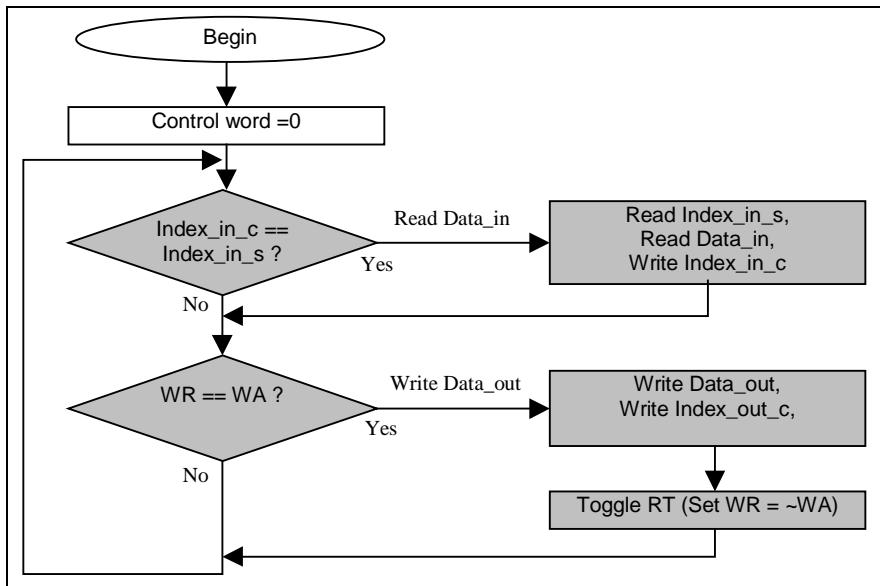


Fig. 7-19: Primary Multiplex Method, Master Communications

Programming Example

To aid in implementing the multiplex function in a PLC program, the following flow chart shows two ways of reading and writing data. Reading and writing can be executed separately, which allows the input data to be updated about 30% faster. The “Read Data” example would be placed at the beginning of a PLC program the “Write Data” example at the end.

Combined reading and writing makes the PLC program simpler, especially when using the same index for both transfer actions.

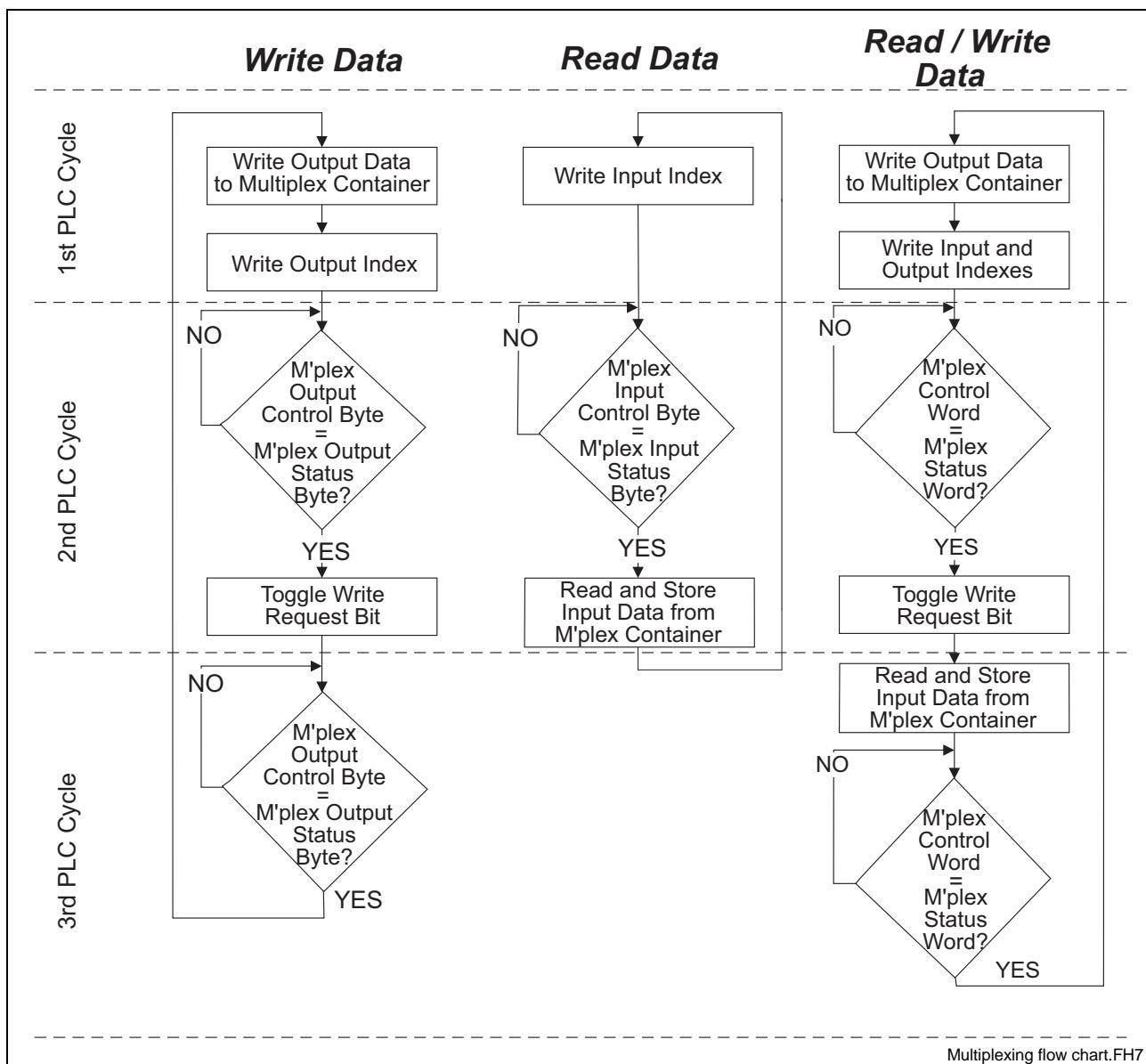


Fig. 7-20: Flow Chart of Multiplex Programming Examples (Primary Method)

Secondary Multiplex Method (for Inconsistent Masters)

Explanation of the Master Consistency Problem

The PPC-R fieldbus slave interfaces can guarantee consistency.

However, some fieldbus masters can only guarantee byte, word or double word consistency. If the master is only word-consistent, it is possible that the master cannot transfer the data and the control word of one multiplex index consistently from the PLC to the fieldbus. Therefore, it is necessary to have a second multiplex method where both input data and output data require the handshake bits to update via the fieldbus.

Note: The meanings of the control and status words are the same as for the Primary Multiplex Method. The only difference is that toggle bits RR and RA are used in the Secondary Method.

Fig. 7-21 and Fig. 7-22 below illustrate the control and status word definitions for the Secondary Multiplex Method.

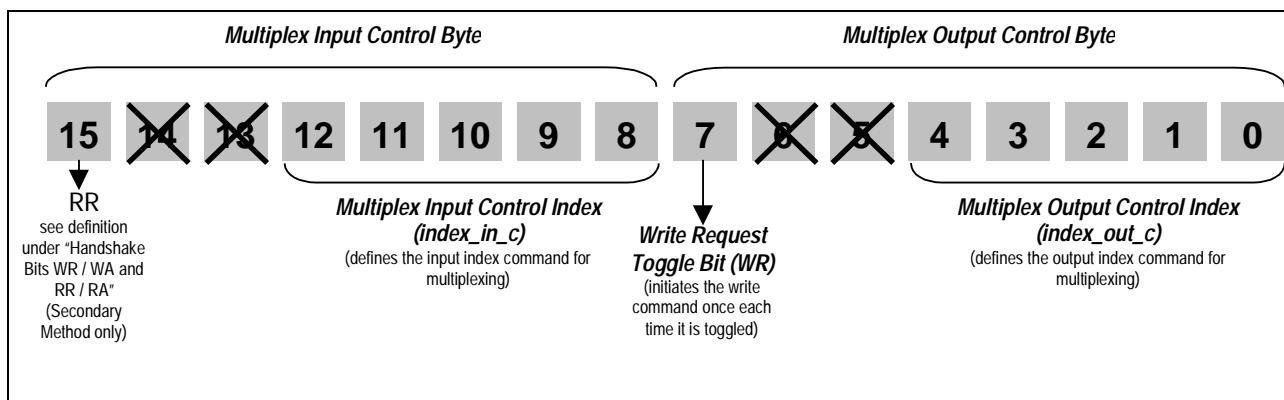


Fig. 7-21: Control Word Definition, Secondary Multiplex Method

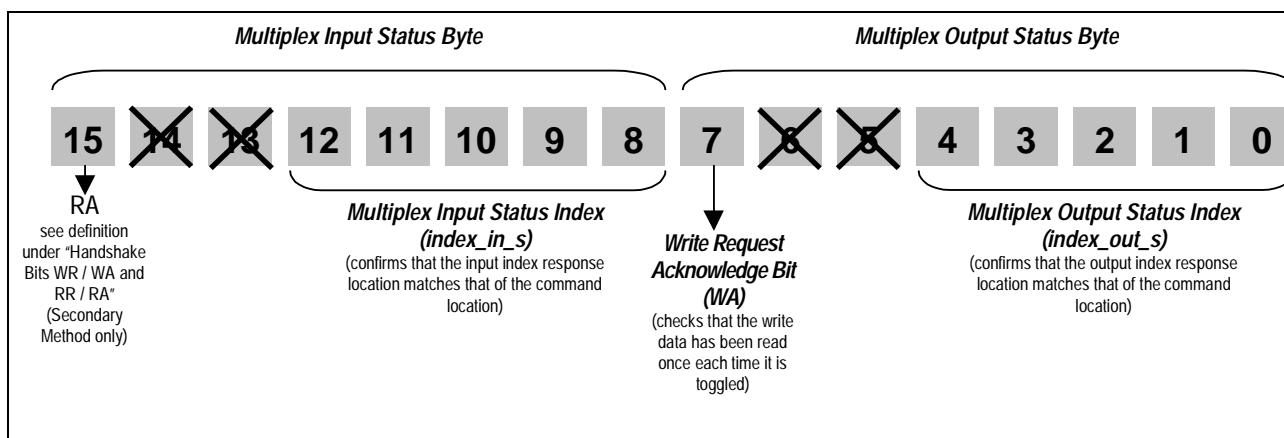


Fig. 7-22: Status Word Definition, Secondary Multiplex Method

The Secondary Multiplex Method has the following features:

- You can transfer a different index from master to slave as from slave to master.
- The handshake bits for both reading and writing of this multiplex channel make the multiplexing possible on inconsistent systems (masters).

Handshake Bits RR and RA

RR (Read Request) and RA (Read Acknowledge) are handshake bits that allow a controlled data transfer and use of the multiplex channel on inconsistent masters. RR and RA control the data transfer for reading data_in (data send from slave to master).

RR == RA:

- tells the master that the slave has sent the requested data_in. The master can now read the data_in and request new data_in.
- tells the slave to do nothing, because the master has not yet put new consistent data on the bus.

RR != RA:

- tells the slave to put new data_in on the bus, because the master requests new data_in.
- tells the master to do nothing, because the slave has not yet put the latest requested multiplex data_in on the bus.

Master Communications (Secondary Multiplex Method)

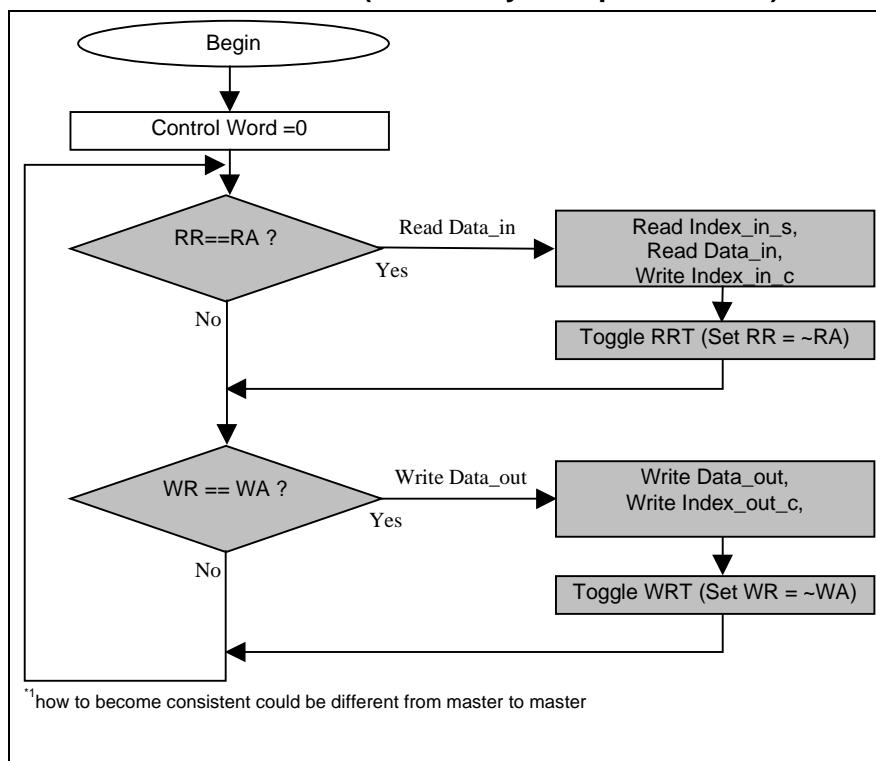


Fig. 7-23: Secondary Multiplex Method, Master Communications

For some masters, it could be enough to first write data and then the control word. For other masters, you may have to implement a delay time (this time could be different from master to master) before writing $WR = \sim WA$.

Non-Cyclic Data (Explicit Messaging)

The following methods for transferring data are available via DeviceNet/ControlNet Explicit Messaging:

- Mapped Data
- Data Exchange Objects

Mapped Data

Mapped data is the most powerful feature of the PPC-R non-cyclic fieldbus interface. Through mapped data, the user has access to virtually every PPC-R parameter over the fieldbus. It is easy to implement from the PLC side and requires no setup on the PPC-R side.

To access a VisualMotion data type over the fieldbus, it has to be specified by an address that consists of a Class, Instance and Attribute. Every data type has a related Class, Instance and Attribute. The Class, Instance and Attribute for each data type can be calculated by a formula (refer to *Example Lookup Tables for Mapped Data* on page 7-36).

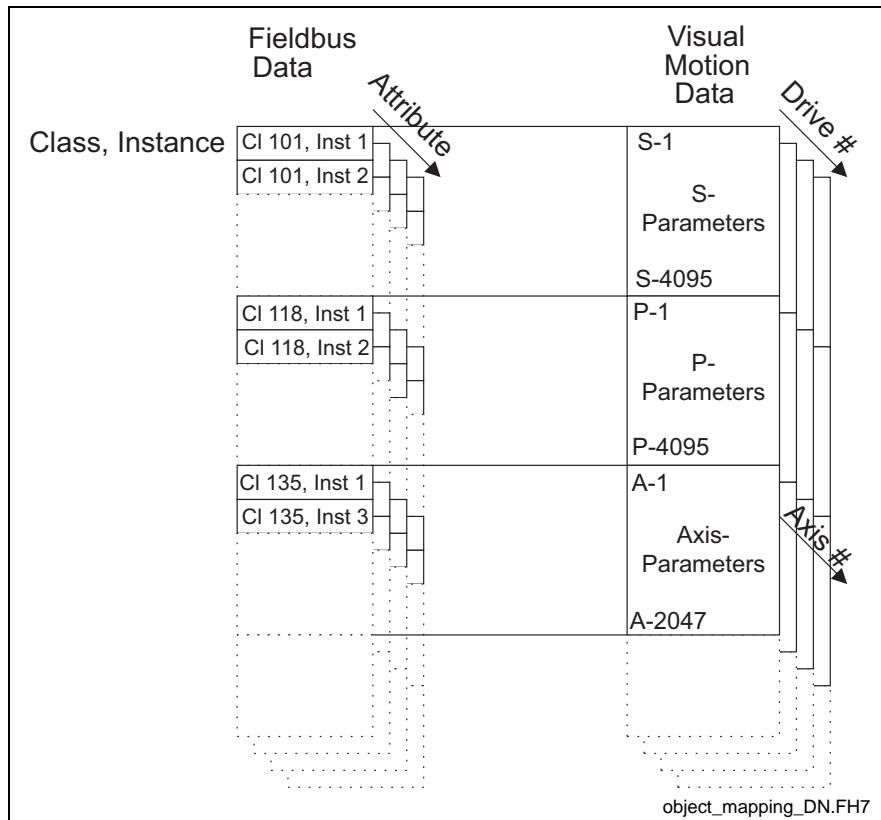


Fig. 7-24: Mapped Data

Mapped data can be used with the following parameters and values:

- S-Parameters (SERCOS Drive S-Parameters)
 - P-Parameters (SERCOS Drive P-Parameters)
 - A-Parameters (PPC Axis Parameters)
 - C-Parameters (PPC C System parameters)
 - T-Parameters (PPC Task parameters)
- size and
format
depend on
parameter *¹
- PF-Values (PPC Program Float data, 32 bit – 2 words, IEEE format) *²
- GI-Values (PPC Global Integer data, 32 bit – 2 words) *²
- GF-Values (PPC Global Float data, 32 bit – 2 words, IEEE format) *²
- PI-Values (PPC Program Integer data, 32 bit – 2 words) *²
- Reg.-Values (PPC Register data, 16 bit – 1 word) *³
- Data Exchange Objects (0x5E70 – 0x5E73) (embedded ASCII Protocol)

*You may notice that parameters accessed via the non-cyclic (Parameter) channel are not always the same size as reported from the attribute field. This is so that the data sizes correspond with the way the different data types are handled in the cyclic channel (Registers are always set to 16-bit size and Parameters are cast to 32-bit size, even if they actually use less space).

1. When **writing** mapped data to a VisualMotion Parameter, you must send the size data corresponding to that of the attribute field within the parameter.
 - a.) For 32-bit parameters, you must send a data size of 32 bits (otherwise, VM error #07 is returned).
 - b.) For 16-bit parameters, you must send a data of size 16-bits. If, for this case, you send data of size 32 bits, one of the following occurs:
 - i.) For parameters of type 16-bit unsigned, only the Low word is stored, and the High word is ignored.
 - ii.) For parameters of type 16-bit signed, bits 0-14 of the low word along with the sign bit #31 are used, and the remaining bits are ignored.
 - c.) For String Parameters (e.g. S-0-0142), you must send the size of the string to write.

- d.) All other Parameter Types (list parameters, command parameters, etc), are not supported for mapped data.

When reading mapped data from a VisualMotion Parameter, there are 3 possible cases of sizes returned:

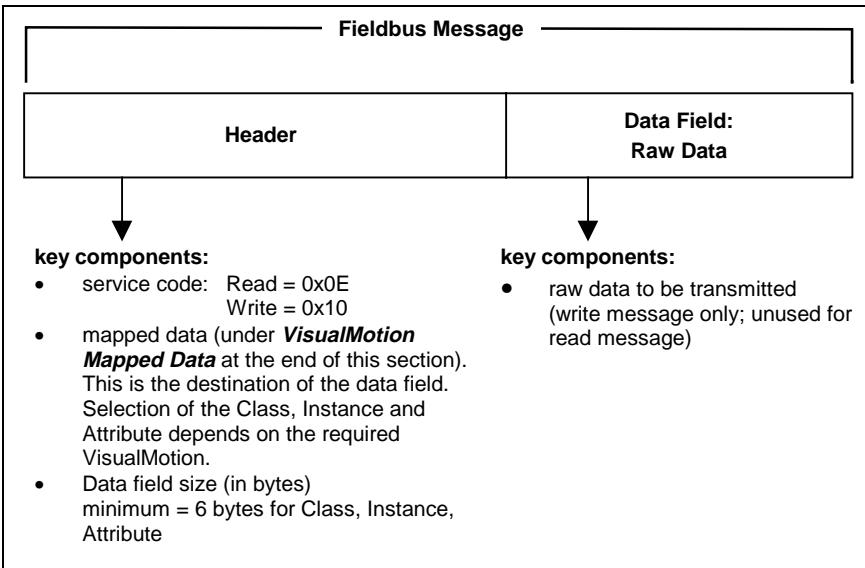
- a.) If the parameter type is a string, you receive the number of bytes corresponding to the length of the string.
 - b.) If the parameter is 32-bit or less, you receive a cast 32-bit value for this parameter. This implies that 16-bit parameters are returned as cast in to 32-bit values.
 - c.) All other parameter types (e.g. list parameters, command parameters, etc.), are not supported for mapped data.
2. When writing mapped data to a VisualMotion Program Float, Program Integer, Global Float, or Global Integer, the data size must be 32-bits (2 words). Any other size returns a VM error #07 (Invalid Data Format).
- When reading mapped data from a VisualMotion Program Float, Program Integer, Global Float, or Global Integer, the data size returned is always 32-bit (2 words).
3. When writing mapped data to a VisualMotion Register, the data must be 16-bits (1 word). Any other size returns a VM error #07 (Invalid Data Format).
- When reading mapped data from a VisualMotion Register, the data size returned is always 16-bit (1 word).

Selecting Mapped Data

To access a data type over the fieldbus, it has to be specified by an address that consists of a Class, Instance and Attribute. Class, Instance and Attribute for each data type can be calculated by a formula (refer to *Accessing Mapped Data* on page 7-34).

Transmission Sequence for Mapped Data

Note: For mapped data, only one transmission (and one response) is required, to send a read or write message to and receive a response from the PPC-R.



Important: The format of the Fieldbus message header and the method of implementation are dependent on the Fieldbus type and the master (PLC) being used. Refer to your Fieldbus master/PLC documentation for proper transport and formatting of the message header.

Non-Cyclic Mapped Data Write

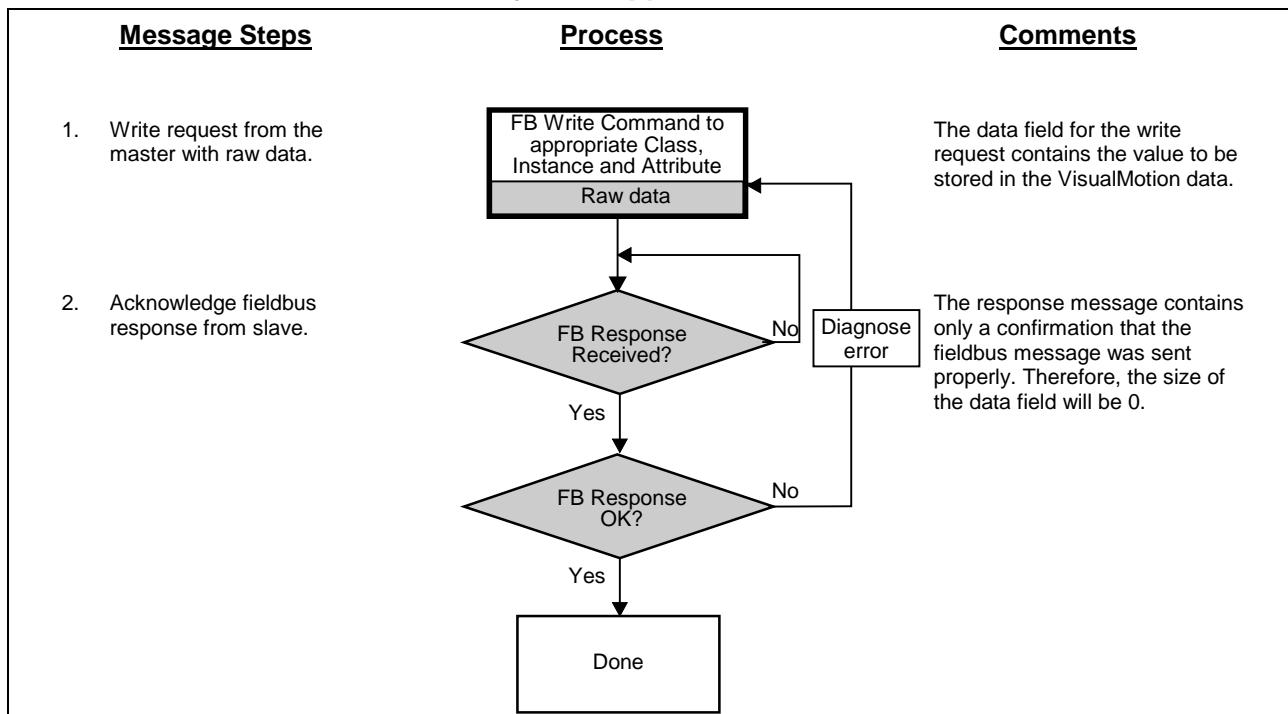
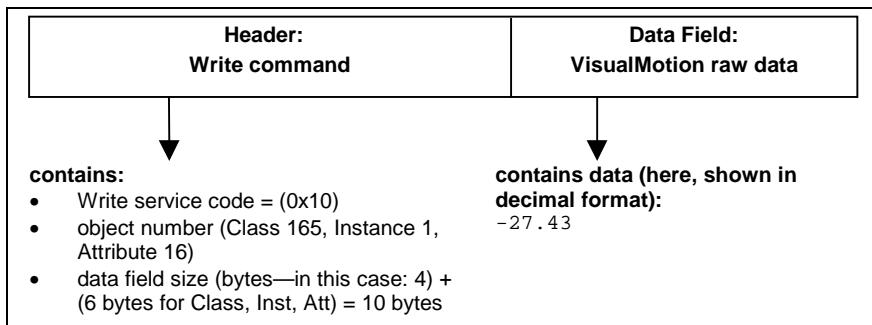


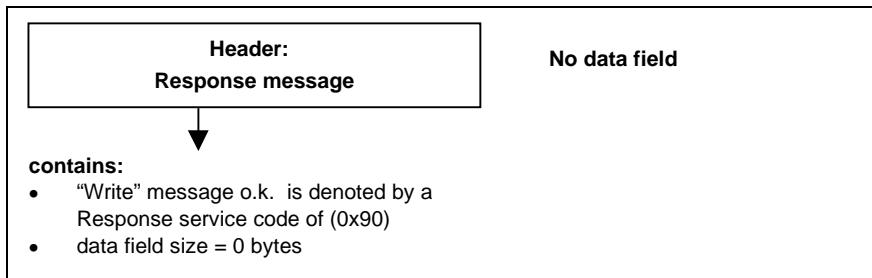
Fig. 7-25: Non-Cyclic Mapped Data Write Process

Example: Write the value -27.43 to Program Float 16 (This is a 32-bit data type, which is mapped to Class 165, Instance 1, and Attribute 16. The Class, Instance and Attribute can be calculated using the formulas under *Accessing Mapped Data* at the end of this chapter.)

1. Write request from the master with raw data.



2. After the write request from the master, the PPC-R sends a response message.



3. If the message response (code in message header) shows o.k., the transaction is complete.

Non-Cyclic Mapped Data Read

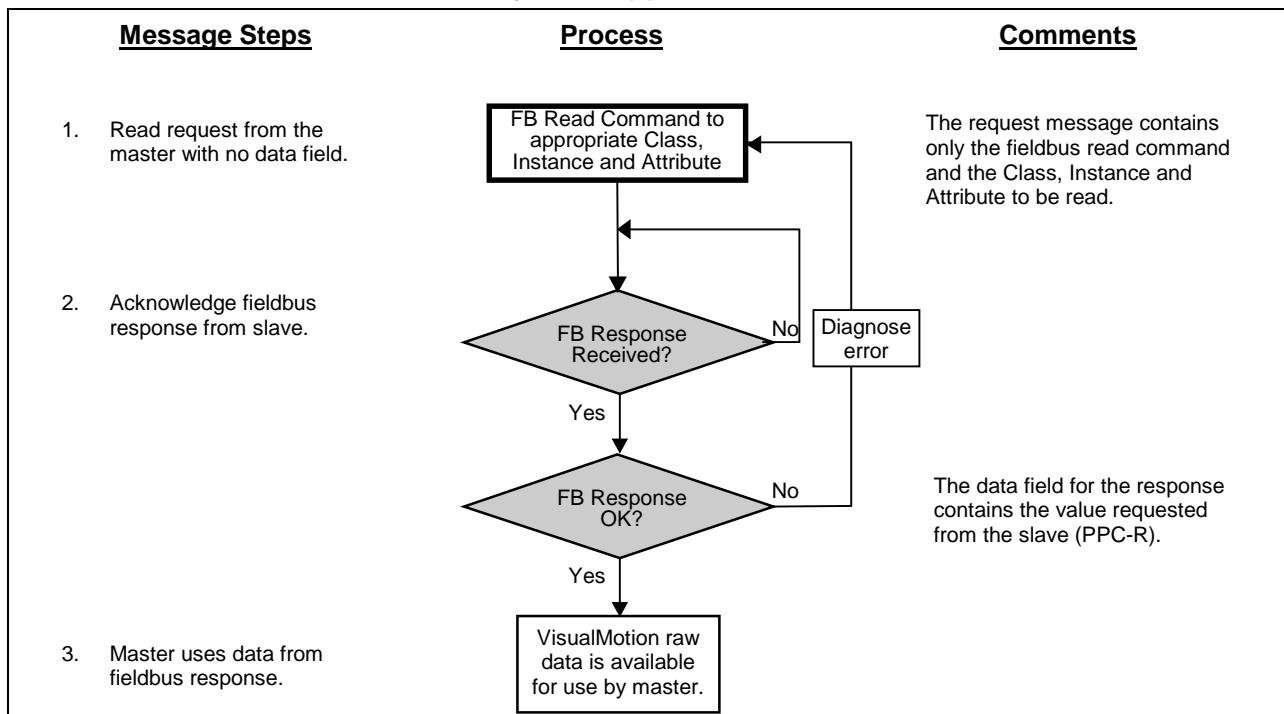
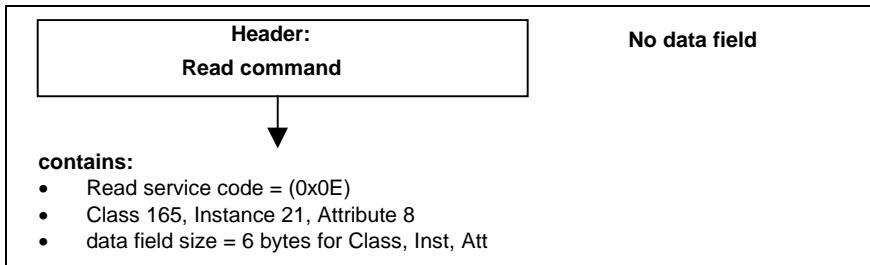


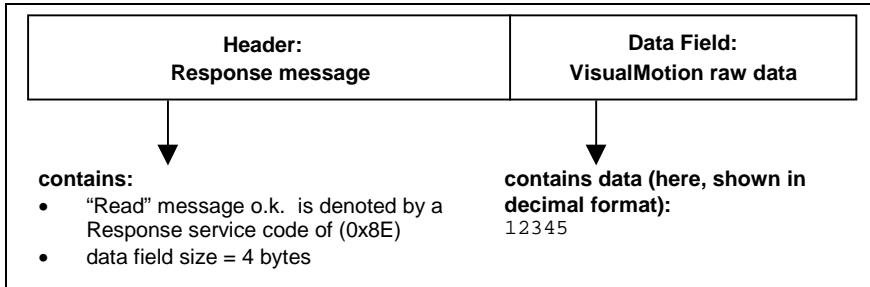
Fig. 7-26: Non-Cyclic Mapped Data Read Process

Example: Read the value contained in Program Integer 8. (This is a 32-bit data type, which is mapped to Class 165, Instance 21, and Attribute 8. The Class, Instance and Attribute can be calculated using the formulas under *Accessing Mapped Data* at the end of this chapter.)

1. Read request from the master.



2. After the read request from the master, the PPC-R sends a response message.



If the message response (code in message header) shows o.k., the requested value is attached to the message in the data field. This value is now available for use by the master (PLC).

Data Exchange Objects

The four data exchange objects Class 100, Instance 1-4, Attribute 100 represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the PPC-R. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic

text, etc.), but more work is required in the master to perform a transmission of this type. Both the VisualMotion ASCII message and the fieldbus transfer message must be formulated.

Table 7-10 below lists the available data exchange objects and their sizes.

Data Exchange Object	Data Length (in bytes)
Class 100, Instance 1, Attribute 100	16
Class 100, Instance 2, Attribute 100	32
Class 100, Instance 3, Attribute 100	64
Class 100, Instance 4, Attribute 100	128

Table 7-10: Length of the Data Exchange Objects

Selecting a Data Exchange Object

Depending on the length of a VisualMotion ASCII message, any of these data exchange objects can be selected.

Note: The entire data length of the data exchange object must always be transmitted even if the VisualMotion ASCII message is shorter.

For example, if you want to transmit an ASCII message of 42 bytes, you must use Class 100, Instance 3. To avoid a response error from the Fieldbus slave, you must append 22 "Null" characters to the end of the ASCII message to complete a data size of 64 bytes.

Note: The checksum for the VisualMotion ASCII protocol is NOT used with the data exchange object. If the checksum is sent as part of the string, it will be ignored, and no checksum will be sent in the VisualMotion ASCII response messages. To ensure data integrity, the Fieldbus protocols support a low-level checksum.

Transmission Sequence via a Data Exchange Object

Note: For the data exchange object, two transmissions (and two responses) are required, to send the read or write message to and then receive the response message from the PPC-R.

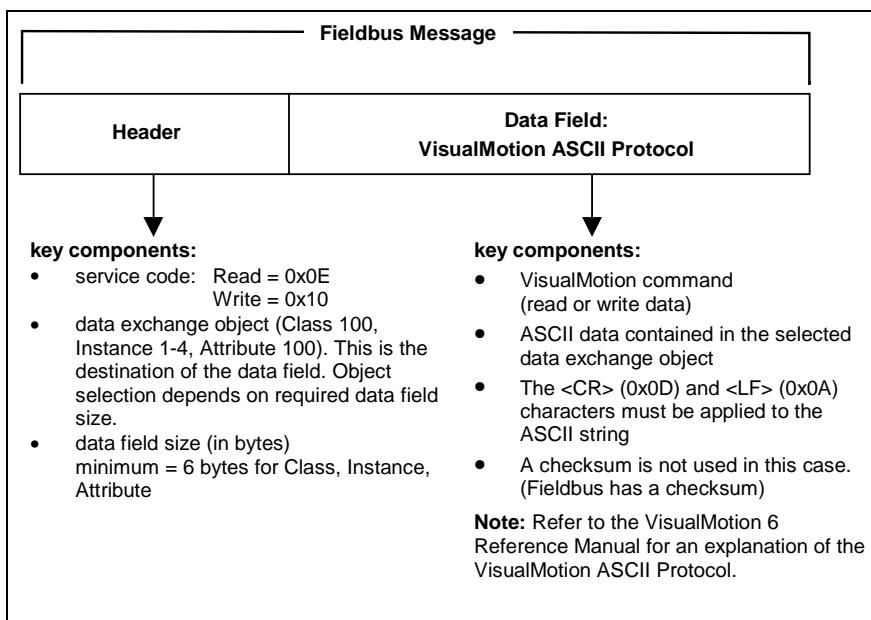


Fig. 7-27: Format of a Non-Cyclic Fieldbus Message using a Data Exchange Object

Important: The format of the fieldbus message header is dependent on the type of master (PLC) being used. Refer to your PLC manufacturer's manual for specific information on this topic.

The following sequence describes the communication between the Fieldbus master (PLC) and the Fieldbus slave (PPC-R):

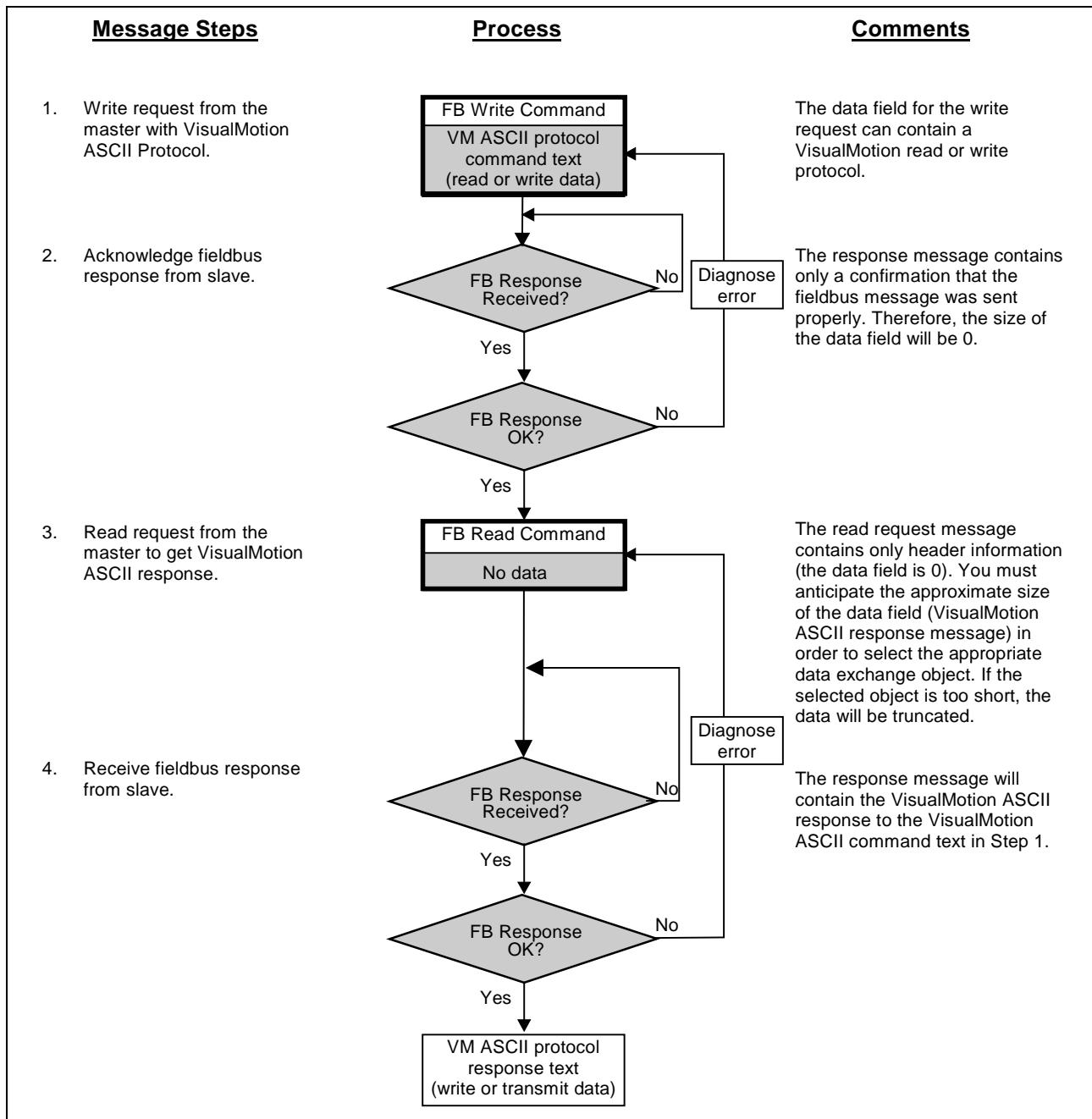
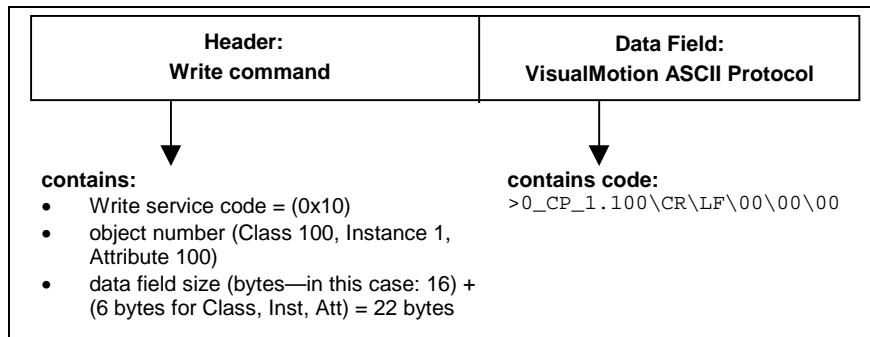


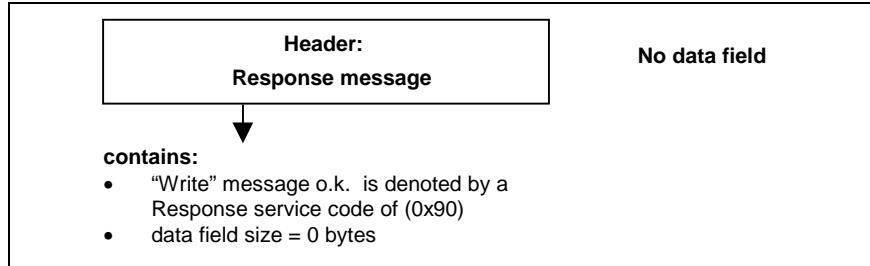
Fig. 7-28: Non-Cyclic (Explicit Messaging) VisualMotion ASCII Communication Process

Example: Read Card Parameter 100 (PPC-R firmware version)

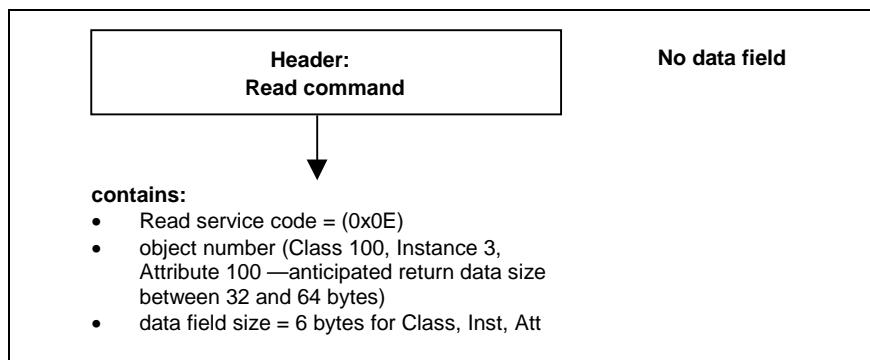
1. Write request from the master with VisualMotion ASCII Protocol.



2. After the first read request from the master, the PPC-R sends a response message.



3. Read request from the master for the VisualMotion ASCII response message.

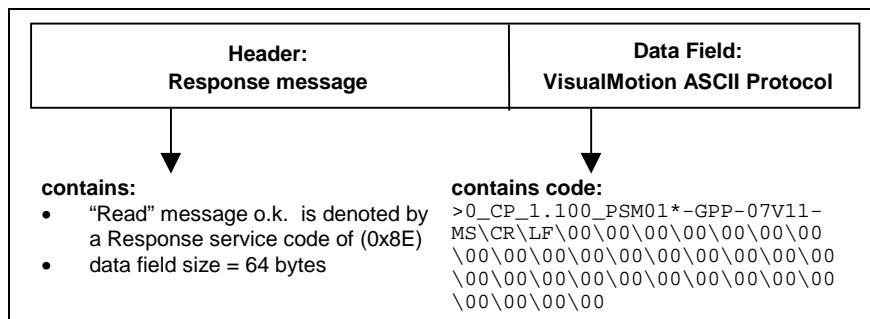


Note: To ensure that all of the data requested in this step is received in step 4 below, a data exchange object of the appropriate size must be selected.

If the selected data exchange object is too small, the data will be truncated.

If the selected data exchange object is too large, efficiency of transmission will be compromised.

4. The PPC-R sends the final response message.



DeviceNet General Error Codes (not valid for ControlNet fieldbuses)

Error No. (Hex)	Error Name	Error Description
0x09	Invalid attribute value	Invalid attribute data detected.
0x0E	Attribute not settable	A request to modify a non-modifiable attribute was received.
0x13	Not enough data	The service did not supply enough data to perform the specified operation.
0x14	Attribute not supported	The attribute specified does not exist in the device.
0x15	Too much data	The service supplied more data than was expected.
0x16	Object does not exist	The object specified does not exist in the device.
0x1F	Vendor-specific error	A vendor-specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflects the error. Refer to extended error code in the VisualMotion ASCII Error Codes (VisualMotion Reference Manual).

Table 7-11: DeviceNet Error Codes

Accessing Mapped Data

Indramat has pre-configured a number of VisualMotion data types to DeviceNet or ControlNet Classes, Instances and Attributes. We call this **concept-mapped data**. These data types can be accessed via DeviceNet/ControlNet Explicit Messaging. The Class, Instance and Attribute for each of these data types can be calculated using the formulas in *Table 7-12* below.

	Class, Instance	Attribute	Formula
Data Exchange Object	Class 166 Instance 137	0	Note: for backwards compatibility, also listed as
	----	----	Class 100, Instance 1 - 4, Attribute 100
	Class 166 Instance 134	0	
<FREE> (349 objects available)	Class 166, Instance 133	255	
	----	----	
	Class 165 Instance 41	1	
Program Integers (Int 1 – Int 5100)	Class 165 Instance 40	255	Class = 165
	----	----	Instance = 21 + [(Program Integer - 1) \ 255]
	Class 165 Instance 21	1	Attribute = Program Integer - [(Instance - 21) * 255]]
Program Floats (Float 1 – Float 5100)	Class 165 Instance 20	255	Class = 165
	----	----	Instance = 1 + [(Program Float - 1) \ 255]
	Class 165, Instance 1	1	Attribute = Program Float - [(Instance - 1) * 255]]

	Class, Instance	Attribute	Formula
<FREE> (235 objects available)	Class 164, Instance 255	255	
	----	----	
	Class 164, Instance 21	1	
Global Integers (GInt 1 – GInt 2550*)	Class 164, Instance 20	255	Class = 164
	----	----	Instance = 11 + [(Global Integer - 1) \ 255]
	Class 164, Instance 11	1	Attribute = Global Integer - [(Instance - 11) * 255]]
Global Floats (GFloat 1 – Gfloat 2550*)	Class 164, Instance 10	255	Class = 164
	----	----	Instance = 1 + [(Global Float - 1) \ 255]
	Class 164, Instance 1	1	Attribute = Global Float - [(Instance - 1) * 255]]
<FREE> (245 objects available)	Class 163, Instance 255	255	
	----	----	
	Class 163, Instance 11	1	
Registers (Reg. 1 – Reg. 2550**)	Class 163, Instance 10	255	Class = 163
	----	----	Instance = 1 + [(Register - 1) \ 255]
	Class 163, Instance 1	1	Attribute = Register - [(Instance - 1) * 255]]
T-Parameters (T-0-0001 – T-0-1020)	Class 162, Instance 255	4	Class = 159 + [(T-Parameter - 1) \ 255]
	----	----	Instance = T-Parameter - [(Class - 159) * 255]
	Class 159, Instance 1	1	Attribute = Task Number
<FREE> (GFloat 1 – Gfloat 2550)	Class 158, Instance 255	255	
	----	----	
	Class 158, Instance 14	1	
C-Parameters (C-0-0001 - C-0-3583)	Class 158, Instance 13	1	Class = 144 + [(C-Parameter - 1) \ 255]
	----	----	Instance = C-Parameter - [(Class - 144) * 255]
	Class 144, Instance 1	1	Attribute = 1 (data)
A-Parameters (A-0-0001 - A-0-2047)	Class 143, Instance 7	99	Class = 135 + [(A-Parameter - 1) \ 255]
	----	----	Instance = A-Parameter - [(Class - 135) * 255]
	Class 135, Instance 1	1	Attribute = Axis Number
P-Parameters (P-0-0001 - P-0-4095)	Class 134, Instance 15	99	Class = 118 + [(P-Parameter - 1) \ 255]
	----	----	Instance = P-Parameter - [(Class - 118) * 255]
	Class 118, Instance 1	1	Attribute = Drive Number
S-Parameters (S-0-0001 - S-0-4095)	Class 117, Instance 15	99	Class = 101 + [(S-Parameter - 1) \ 255]
	----	----	Instance = S-Parameter - [(Class - 101) * 255]
	Class 101, Instance 1	1	Attribute = Drive Number

* current limitation: first 256 global integers/floating-point numbers.

**current limitation: first 512 registers.

Table 7-12: Formulas for Determining Mapped Objects

Example Lookup Tables for Mapped Data

Card (C) Parameters

The following is an example lookup table for C-Parameters, when using mapped objects.

Example Look-up Chart for: C-Parameters CP 0.Y ==> CP = Card Parameter Y = Parameter Number									
Attribute ID									
	Class 144	Class 144	Class 144	Class 144	Class 145	Class 158	Class 158
	Instance 1	Instance 2	Instance 3	Instance 255	Instance 1	Instance 12	Instance 13
1	CP 0.1	CP 0.2	CP 0.3		CP 0.255	CP 0.256		CP 0.3582	CP 0.3583

Table 7-13: C-Parameters Lookup Table for Mapped Data Types

Axis(A) Parameters

The following is an example lookup table for A-Parameters, when using mapped objects. The same formula also applies to SERCOS (S) and Task (T) Parameters.

Example Look-up Chart for: A-Parameters AP X.Y ==> AP = Axis Parameter X = Axis Number Y = Parameter Number									
Attribute ID									
	Class 135	Class 135	Class 135	Class 135	Class 136	Class 143	Class 143
	Instance 1	Instance 2	Instance 3	Instance 255	Instance 1	Instance 6	Instance 7
1	AP 1.1	AP 1.2	AP 1.3		AP 1.255	AP 1.256		AP 1.2046	AP 1.2047
2	AP 2.1	AP 2.2	AP 2.3		AP 2.255	AP 2.256		AP 2.2046	AP 2.2047
3	AP 3.1	AP 3.2	AP 3.3		AP 3.255	AP 3.256		AP 3.2046	AP 3.2047
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
99	AP 99.1	AP 99.2	AP 99.3		AP 99.255	AP 99.256		AP 99.2046	AP 99.2047

Table 7-14: A-Parameters Lookup Table for Mapped Data Types

Product-Specific (P) Parameters The following is an example lookup table for P-Parameters, when using mapped objects.

Example Look-up Chart for:			P-Parameters		PP X.Y		==>		PP = SERCOS P-Parameter (set 0 only)	
Attribute ID	Class 118	Class 118	Class 118	Class 118	Class 119	Class 134	Class 134	
	Instance 1	Instance 2	Instance 3	Instance 255	Instance 1	Instance 14	Instance 15	
	1	PP 1.1	PP 1.2	PP 1.3		PP 1.255	PP 1.256		PP 1.4094 PP 1.4095	
	2	PP 2.1	PP 2.2	PP 2.3		PP 2.255	PP 2.256		PP 2.4094 PP 2.4095	
	3	PP 3.1	PP 3.2	PP 3.3		PP 3.255	PP 3.256		PP 3.4094 PP 3.4095	
	:	:	:	:	:	:	:	:	:	
	:	:	:	:	:	:	:	:	:	
	99	PP 99.1	PP 99.2	PP 99.3		PP 99.255	PP 99.256		PP 99.4094 PP 99.4095	

Table 7-15: P-Parameters Lookup Table for Mapped Data Types

Integers The following is an example lookup table for Integers, when using mapped objects. The same formula also applies to Floats, Global Integers, Global Floats and Registers.

Example Look-up Chart for:			VM Program Integers		PI 0.Y		==>		PI = Program Integer	
Attribute ID =	Class 165	Class 165	Class 165	Class 165	Class 165	Y = Program Integer Number	
	Instance 21	Instance 22	Instance 23		Instance 40		Instance 40			
	1	PI 1	PI 256	PI 511		PI 4846		PI 4846		
	2	PI 2	PI 257	PI 512		PI 4847		PI 4847		
	3	PI 3	PI 258	PI 513		PI 4848		PI 4848		
	:	:	:	:	:	:	:	:	:	
	:	:	:	:	:	:	:	:	:	
	255	PI 255	PI 510	PI 765		PI 5100		PI 5100		

Table 7-16: Program Integers Lookup Table for Mapped Data Types

8 Interbus Fieldbus Interface

8.1 General Information

Version Note:

Information in this document is based on VisualMotion Toolkit software version 08V06 and PPC-R firmware version GPP08V16.

Note: For fieldbus hardware information, refer to the *VisualMotion 8 Project Planning Manual*.

PPC-R System Description with a Fieldbus

The PPC-R can operate on a serial fieldbus interface (network) by means of a fieldbus expansion card that communicates with the PPC-R via dual-port RAM. The function of the fieldbus card is similar to that of a network card in a PC: it allows communication with other devices on the network.

In Fig. 8-1, a commonly described fieldbus interface is pictured:

- **Fieldbus Master** - PLC fieldbus interface
- **Fieldbus Slave** - PPC-R fieldbus interface

In this document, we will refer to the PLC as the **fieldbus master** and the PPC-R as the **fieldbus slave**.

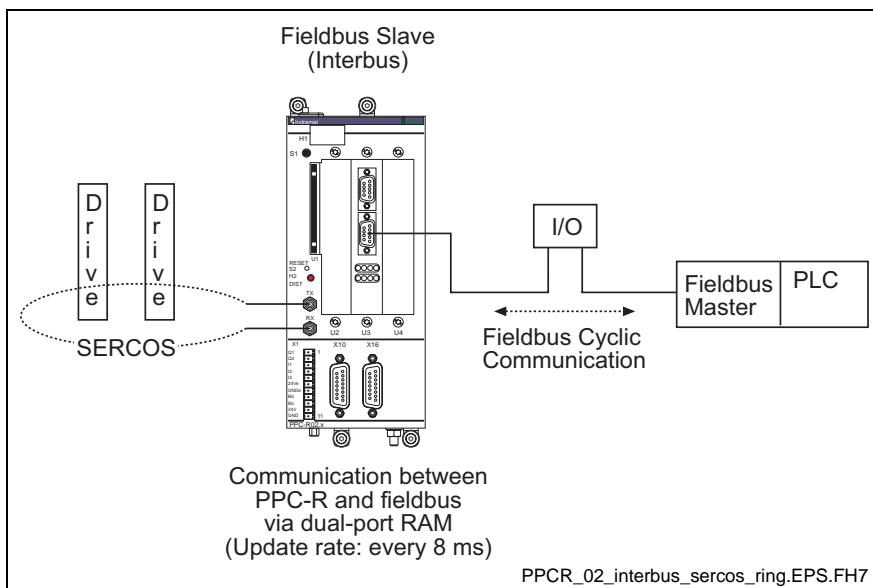


Fig. 8-1: Sample Master/Slave Setup with Fieldbus Card

With the PPC-R, the fieldbus card can be used **only** as a **slave** card in a master/slave setup.

The VisualMotion Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to set up fieldbus configuration and data mapping.

Data Transfer Direction (Output vs. Input)

In the VisualMotion Fieldbus Mapper, output and input are always described with respect to the fieldbus master. The definitions for output and input follow:

output: the communication from the PLC to the PPC-R (i.e. from the fieldbus master to the fieldbus slave).

Synonyms for this type of communication: **send** or **write** data.

input: the communication from the PPC-R to the PLC (i.e. from the fieldbus slave to the fieldbus master).

Synonyms for this type of communication: **receive** or **read** data.

Fieldbus Data Channel Descriptions

The Indramat Interbus fieldbus interface card for the PPC-R supports the following communication channels:

- Cyclic (PD) Channel
- Non-Cyclic (PCP) Channel

Fig. 8-2 shows the possible channel configurations.

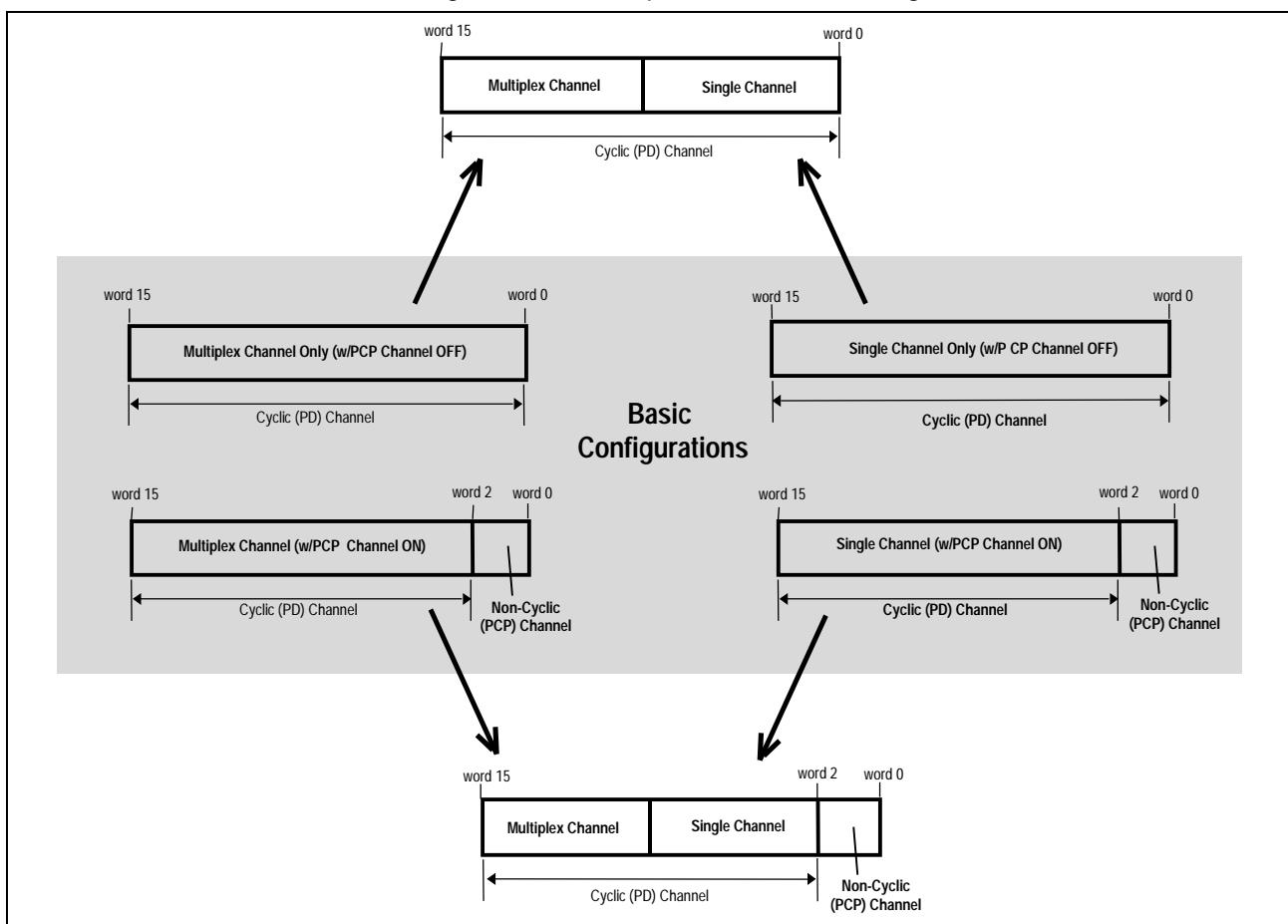


Fig. 8-2: Interbus Channel Configuration Options

Cyclic (PD) Channel

The cyclic (PD) channel, sometimes called the real-time channel, contains user-defined data. This data is stored in two ordered lists (C-0-2600 for input data, C-0-2601 for output data) and transmitted serially over the bus. This data is updated cyclically between the fieldbus master and slave.

The cyclic data channel is limited to 16 input words and 16 output words (provided the non-cyclic channel is turned off). If the non-cyclic (PCP) channel is turned on, it consumes 2 words, thus limiting the cyclic channel to 14 input words and 14 output words. PPC-R data types consume these words in either one-word (16-bit) groups for PPC-R registers or two-word (32-bit) groups for all other data types.

The PPC-R mapping list is scanned every 8 ms and data is sent and received to/from the fieldbus slave board's dual port RAM.

The cyclic data channel can be made up of any combination of the following data types:

- Single Channel
- Multiplex Channel

Cyclic Data: Types and Sizes

The following table outlines the PPC-R data types that can be transmitted via the cyclic channel and the amount of space (in 16-bit data words) that each data type consumes.

Note: The cyclic data mapping lists supports only 16- and 32-bit data of the following types for reading and writing:

- Integer
- Float
- Binary (used in control parameters)
- Hex (used in control parameters)

For all other data types (e.g. diagnostic messages - "strings"), use the non-cyclic Channel.

PPC-R Data Type	Data Size (in 16-Bit Words)
Register	1
Program Integer (currently active program ONLY *)	2
Program Float (currently active program ONLY *)	2
Global Integer	2
Global Float	2
Card Parameter	2
Axis Parameter	2
Task Parameter	2
Note: Drive parameters "S" or "P" cannot be transmitted cyclically because of the inherent delay of parameter access over the SERCOS service channel. See " Non-Cyclic (PCP) Channel ." However, if a drive parameter is mapped to an Axis Parameter, that Axis parameter could be used in cyclic data (see description of Axis Parameters 180-196 in the VisualMotion Reference Manual).	
* Important Note: Integers and floats are shown only for the currently active program. Each time you activate a new program, the fieldbus reads/writes to the newly-activated program.	

Table 8-1: PPC-R Cyclic Data Types and Sizes

Single Data Types

Single data types are mapped directly in the cyclic mapping ordered lists (C-0-2600, C-0-2601).

Multiplex Data Types (Cyclic Data Channel)

Important: You should use multiplexing **only** if your Interbus master is consistent over the entire cyclic channel!

In some multi-axis applications, 14 or 16 words of cyclic data transfer are not sufficient to meet the requirement of the application.

When insufficient data transfer space is available, multiplex data can be set up within the cyclic channel. One multiplex container acts as a placeholder for multiple possible PPC-R data types (all of the same word size). The currently transmitted PPC-R data type is based on an index value placed in a multiplex control or status word attached to the end of the cyclic list. Depending on the index specified by the master, the multiplex channel permits a different set of data within the cyclic channel to be transferred as current real-time data. Multiplex containers can be added to the input and output lists separately and the input and output indexes can be designated separately (in the control and status words).

Note: Using the multiplex channel reduces the maximum number of usable words for storing control data to 15. The 16th word (or last used word, if fewer than 15 words) is used as the multiplex entry control/status word.

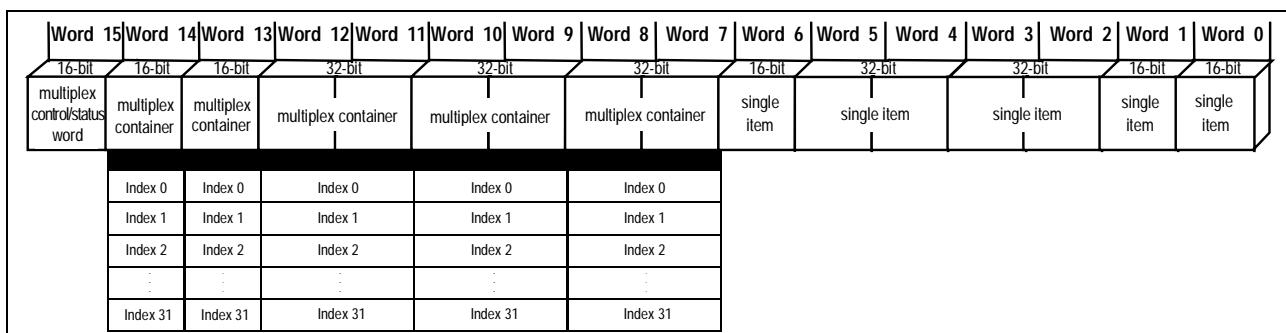


Fig. 8-3: Sample Command (PLC⇒PPC-R) or Response (PPC-R⇒PLC)

The multiplex control and status words serve to command and acknowledge multiplex data transferred between the fieldbus master and the fieldbus slave. The **control** word is associated with **output** communication (PLC⇒PPC-R). The **status** word is associated with **input** communication (PPC-R⇒PLC). Single data items are not affected by the multiplex control and status words.

Note: For specific information about how the fieldbus master uses the multiplex control and status words, refer to Multiplexing on page 8-18.

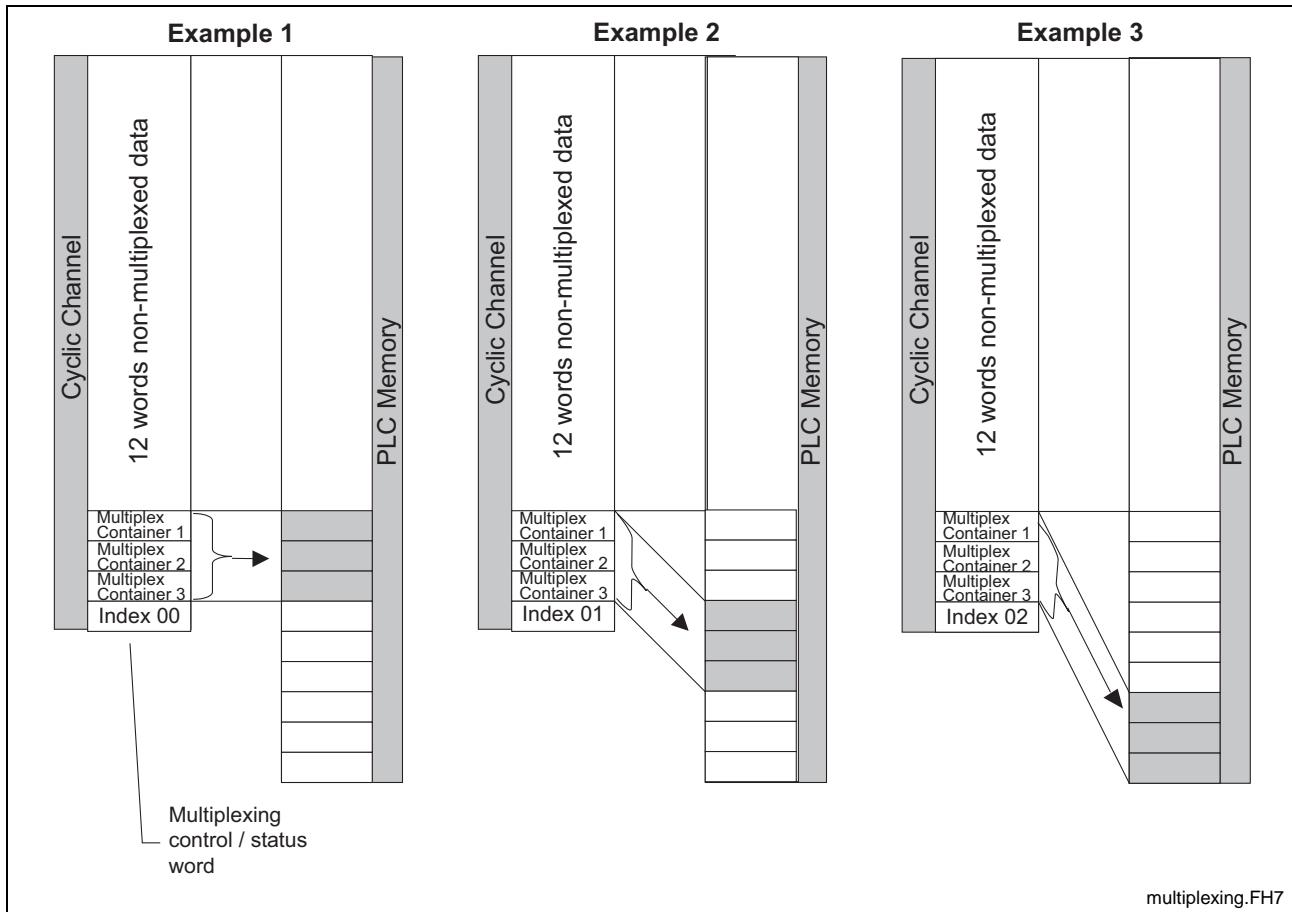


Fig. 8-4: Examples for Reading Data via the Multiplex Channel

Non-Cyclic (PCP) Channel

For Interbus systems using the PPC-R/VisualMotion hardware configuration, the non-cyclic (PCP) channel can be used for parameterization, extended diagnostic information and other “non-urgent” communication.

When enabled, the PCP channel is always fixed at a length of 2 words. If it is not needed, the PCP channel can be disabled, allowing use of those two words for the cyclic channel.

Note: For further explanation of the features supported in the PCP channel, refer to ***Non-Cyclic Data Access via the Non-Cyclic (PCP) Channel*** on page 8-22.

8.2 Fieldbus Mapper Functionality

Initializing the Fieldbus Mapper from VisualMotion 8

1. Open an existing program or create a new program. You must be using PPC-R hardware with GPP firmware to use the Fieldbus Mapper described in this document.

Note: Make sure the VisualMotion system is configured for GPP (in the **Settings⇒Configuration** menu item of the main VisualMotion screen).

2. Select **Commission⇒Fieldbus Mapper**. The main Fieldbus Mapper screen appears (refer to Fig. 8-5).

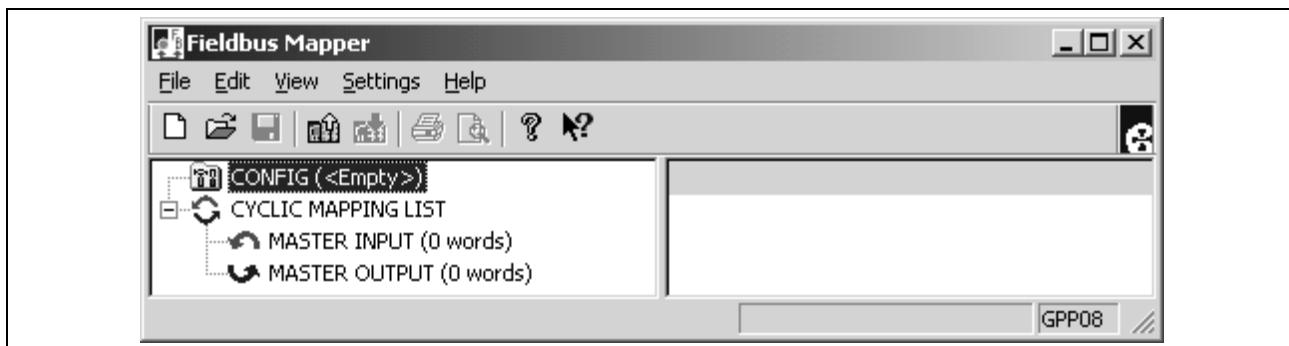


Fig. 8-5: Fieldbus Mapper Main Screen (Blank)

Creating a New Fieldbus Mapper File

1. Click or select **File⇒New**. A “setup wizard” goes through three steps:
 - Fieldbus Slave Definition
 - Fieldbus Slave Configuration
 - Cyclic Data Configuration
2. Enter the information requested in the setup screens. For more details on each step, refer to *Fieldbus Slave Definition*, *Fieldbus Slave Configuration*, and *Cyclic Data Configuration* for detailed information about each configuration step.
3. Save the file (automatically has a *.prm extension).

Editing an Existing Fieldbus Mapper File

1. Click or select **File⇒Open**.
2. Browse to find the desired file (*.prm extension).
3. Click **Open**. The main Fieldbus Mapper screen appears, which lists the configuration information. Refer to Fig. 8-6.

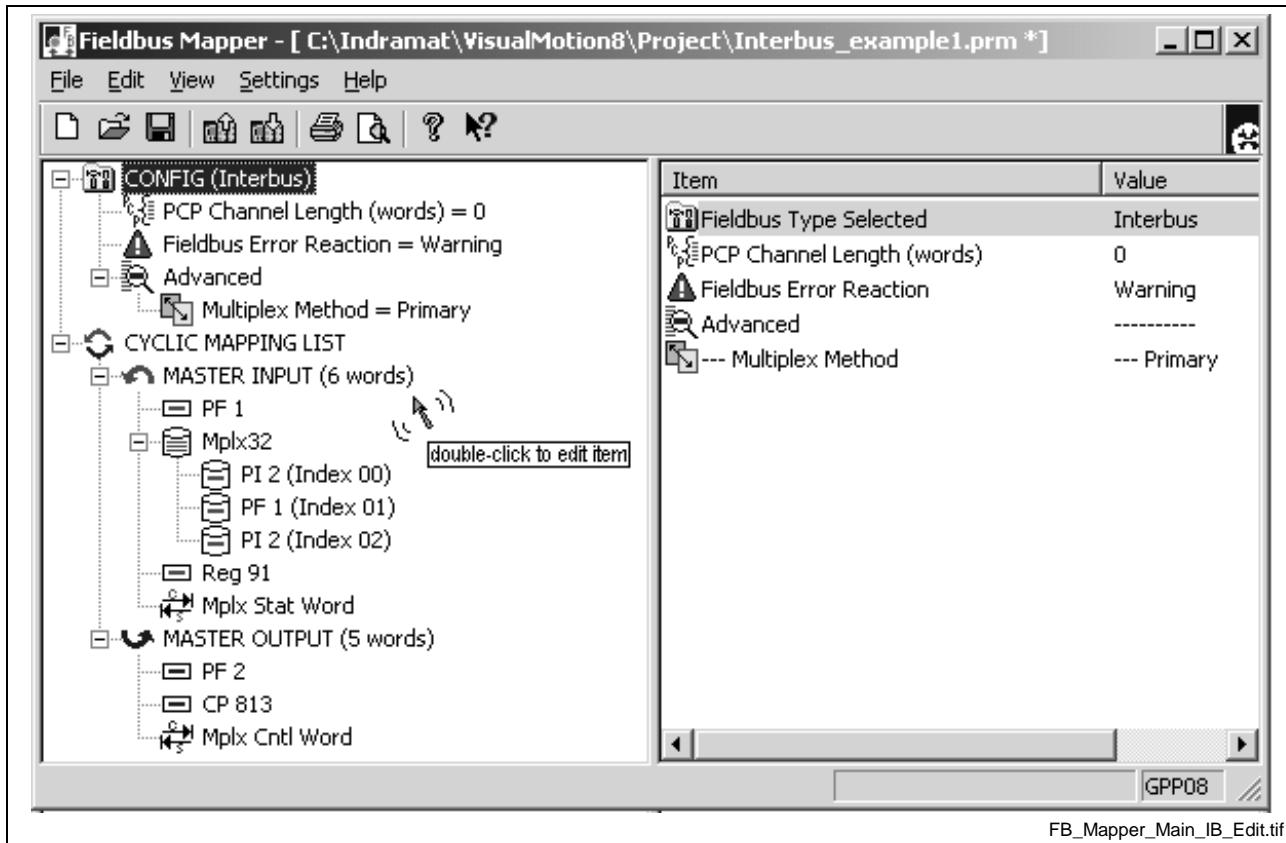


Fig. 8-6: Fieldbus Mapper Main Screen (Complete)

4. From the Fieldbus Mapper main screen, **double-click** on the specific item to be edited. The corresponding setup screen appears.

- Or -

Select the item to edit from the **Edit** menu (refer to Fig. 8-7). For more information about each step, refer to *Fieldbus Slave Definition*, *Fieldbus Slave Configuration*, and *Cyclic Data Configuration* for detailed information about each configuration step.

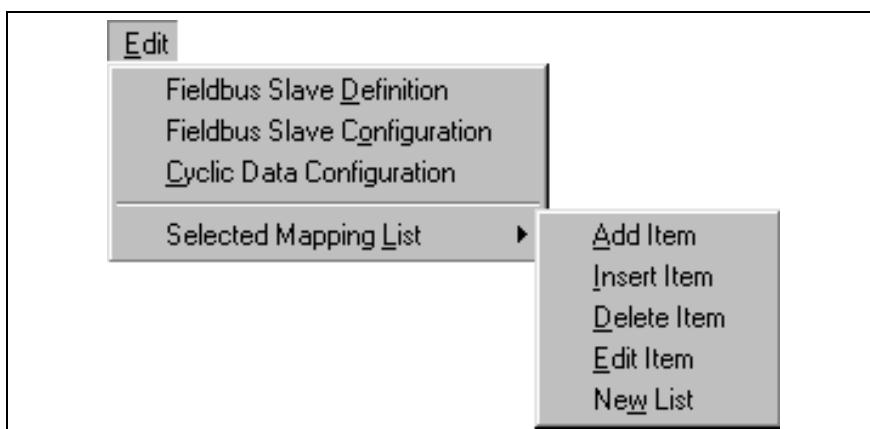


Fig. 8-7: Fieldbus Mapper Edit Menu

Note: You can also directly add, insert, delete, edit an item, or create a new list by:

- clicking on the item to be edited in the main Fieldbus Mapper screen and selecting the desired function under **Edit⇒Selected Mapping List**

OR

- right-clicking on an item to display a menu of functions

Fieldbus Slave Definition

Note: The **Interbus** Fieldbus Type is supported only by Hardware Platform **PPC-R (GPP08vRS)**.

From the Fieldbus Slave Definition window, select **PPC-R (GPP08vRS)** as the Hardware Platform and **Interbus** as the Fieldbus Type (refer to *Fig. 8-8*).

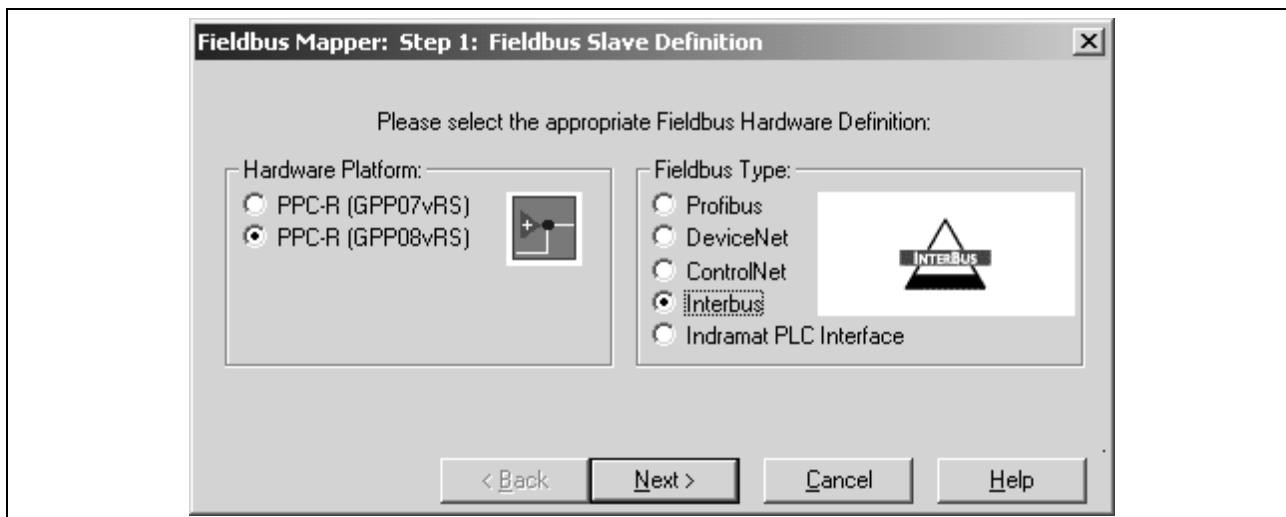


Fig. 8-8: Fieldbus Slave Definition Window

Fieldbus Slave Configuration

The Interbus Fieldbus Slave Configuration screen is shown in *Fig. 8-9* below.

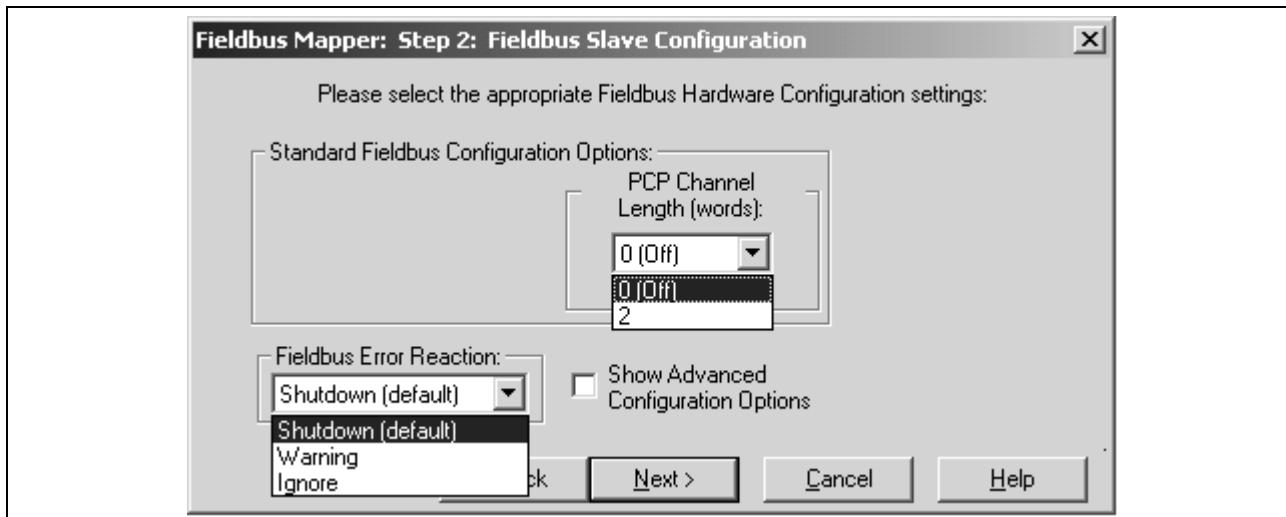


Fig. 8-9: Fieldbus Slave Configuration

Fieldbus Error Reaction

Set the Error Reaction to Shutdown (default), Warning or Ignore. Refer to *Fieldbus Error Reaction* on page 8-16 for detailed information about each setting.

PCP Channel Length

The PCP (non-cyclic) channel can be set to 0 words (Off) or 2 words (On).

Advanced Configuration Options

The “Advanced Options:” are shown only if the checkbox next to **Show Advanced Configuration Options** is checked (refer to *Fig. 8-10* below). In most cases, the default options should apply.

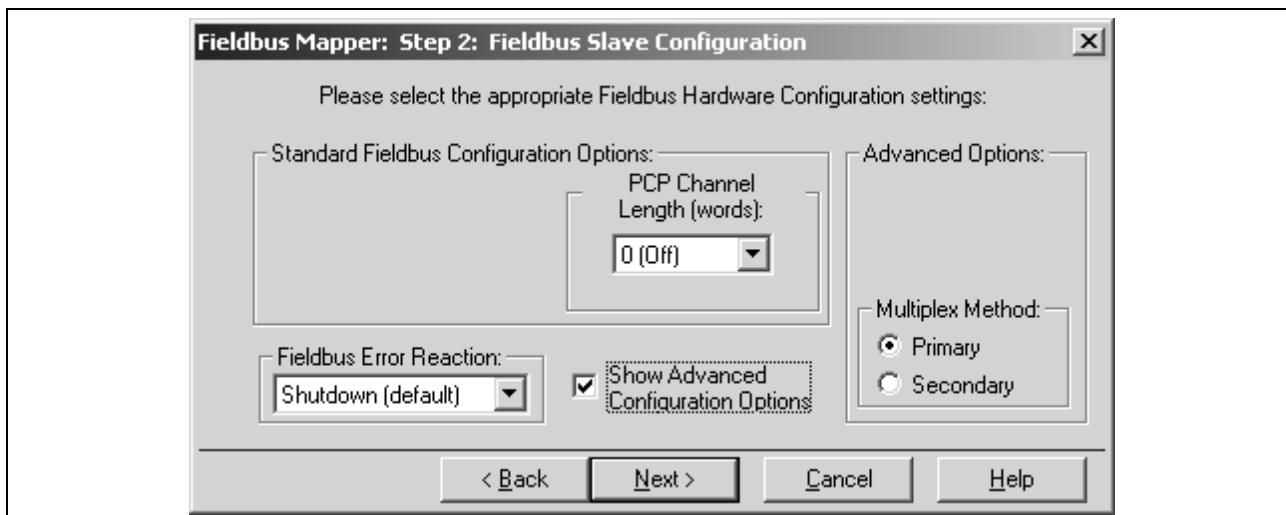


Fig. 8-10: Fieldbus Slave Configuration: Advanced

- **Multiplex Method:** select Primary or Secondary (Primary is the default). Select Secondary only if you have a consistent fieldbus master. Refer to *Multiplexing* on page 8-18 for detailed information about each method.

Cyclic Data Configuration

An example of the Cyclic Data Configuration screen is shown in *Fig. 8-11* below. No data has yet been configured. If you are editing an existing Fieldbus Mapper file, the list will probably contain more items.

First, you must select the Cyclic Input List (from PPC-R to PLC) or the Cyclic Output List (from PLC to PPC-R).

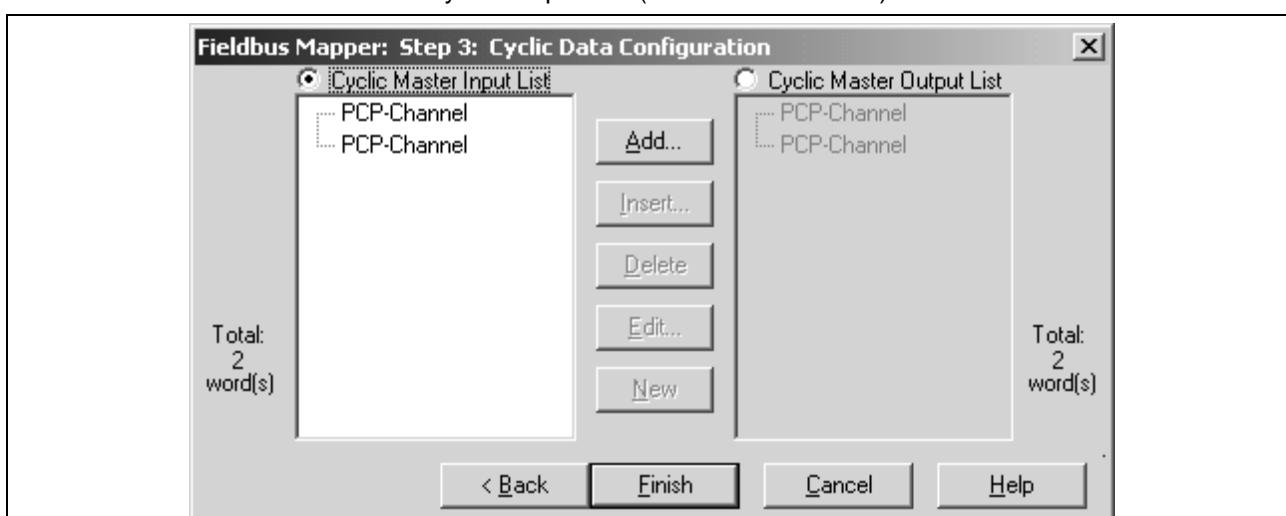


Fig. 8-11: Cyclic Data Configuration—Initial Screen

Adding an Item to the List

1. Select the Cyclic Input List or the Cyclic Output List.
2. Click Add. The screen in *Fig. 8-12* below appears. Select the Data Type (for example, Register).

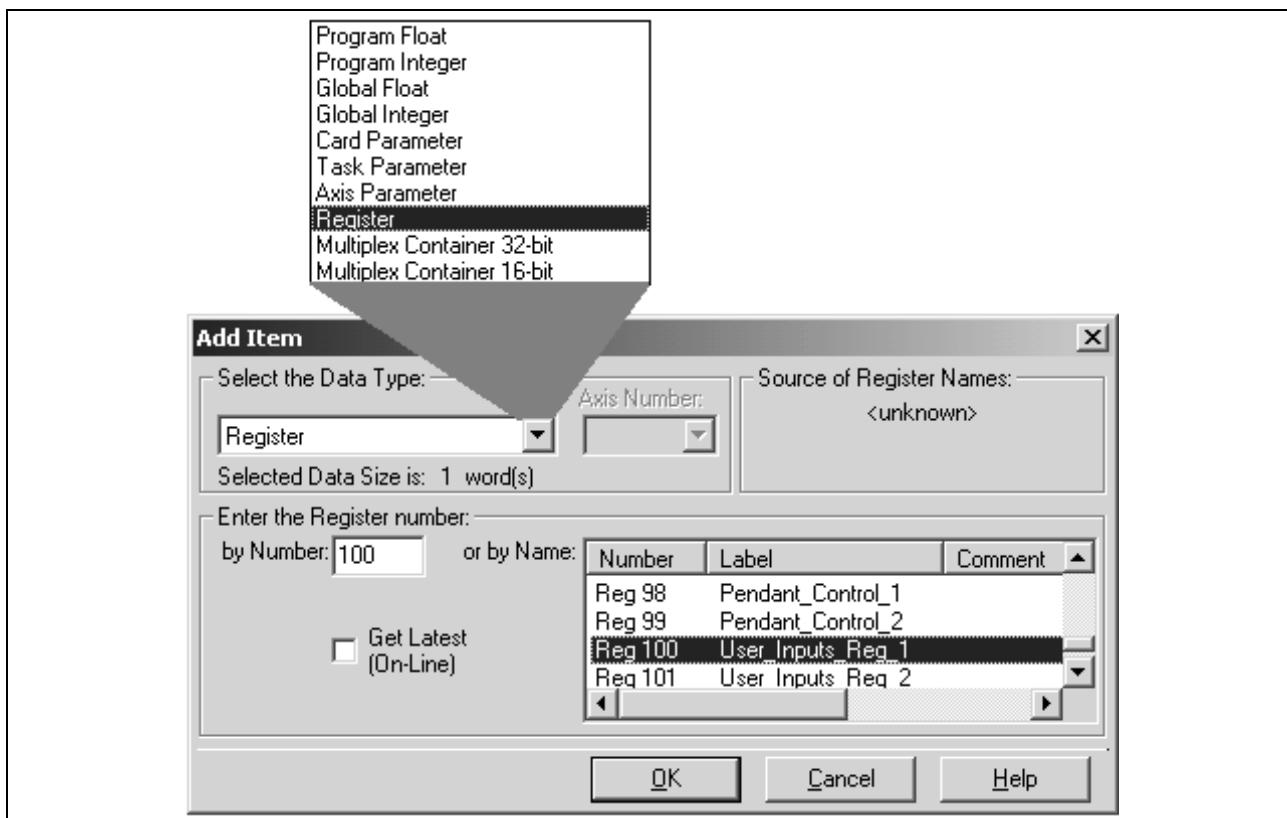


Fig. 8-12: Add Item to Cyclic Data

Note: Registers and 16-bit Multiplex Containers (used only for Registers) require one data word (16 bits), and all other data types require two data words (32 bits) of space.

3. Enter the required information (for example Register Number) or select it from the list below. Only the available data types for your designated VisualMotion hardware setup and fieldbus type are listed.

Note: If you check the box next to "Get Latest (On-Line)," the data type label list is updated based on your firmware version and the currently active program.

4. Click OK to add the selected item to the list.

Adding Multiplex Containers to the List

1. Select the Cyclic Input List or the Cyclic Output List.
2. Click Add.
3. In the "Add Item" screen under "Select the Data Type:" select Multiplex Container 16-bit (for Registers) or Multiplex Container 32-bit (for all other data types).
4. Click OK to add the Multiplex Container to the List. The screen in *Fig. 8-13* below is an example where a 16-bit Multiplex Container and a 32-Bit Multiplex Container have been added.

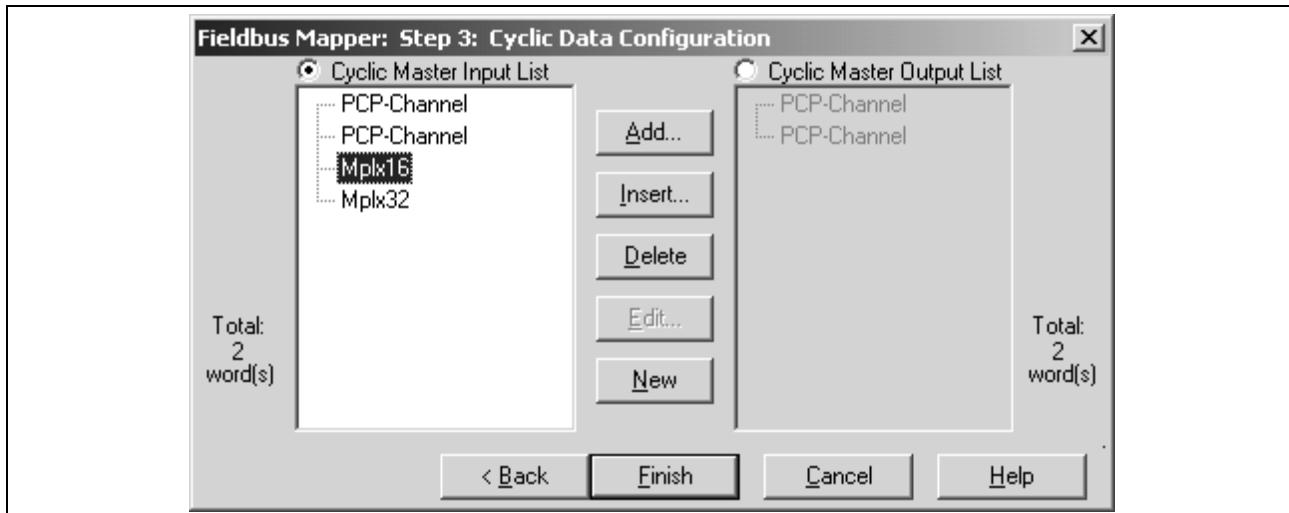


Fig. 8-13: Cyclic Data Configuration, Multiplex Containers

Note: At this point, the Multiplex Containers do not yet contain any items. To add multiplex items, refer to below.

Adding Items to an Empty Multiplex Container

1. In the Cyclic Data Configuration screen, select the multiplex container to which you want to add items.
2. Click Add. The screen in *Fig. 8-14* below appears. Because it is unclear whether you would like to add to the list or to the multiplex container, the Fieldbus Mapper is requesting clarification.

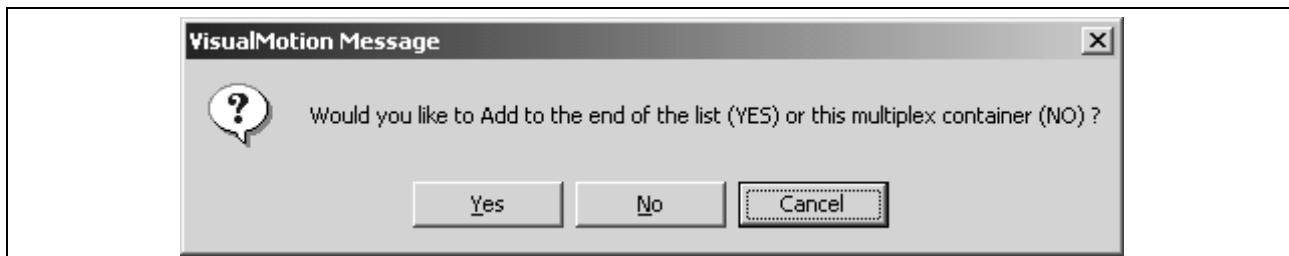


Fig. 8-14: Add Item or Multiplex Item Screen

Note: For subsequent items, highlight any of the indexes within the multiplex container before clicking Add, and the Fieldbus Mapper will know you want to add to that container.

3. To add to the selected multiplex container, click No. The screen in *Fig. 8-15* below is an example for adding a 32-bit multiplex item.
4. Select the desired item to be added to the multiplex container.

Note: In addition to the data types that can be added to the multiplex list, an empty item called **Multiplex Empty Item** is available to fill a space within the multiplex container, if nothing is to be mapped to a particular index.

5. Click OK. The item is automatically placed in the multiplex container as the next unassigned index item (e.g. the first item is index 00, the last is index 31).
6. Repeat for as many items as you want to add to the multiplex container, up to 32 items.

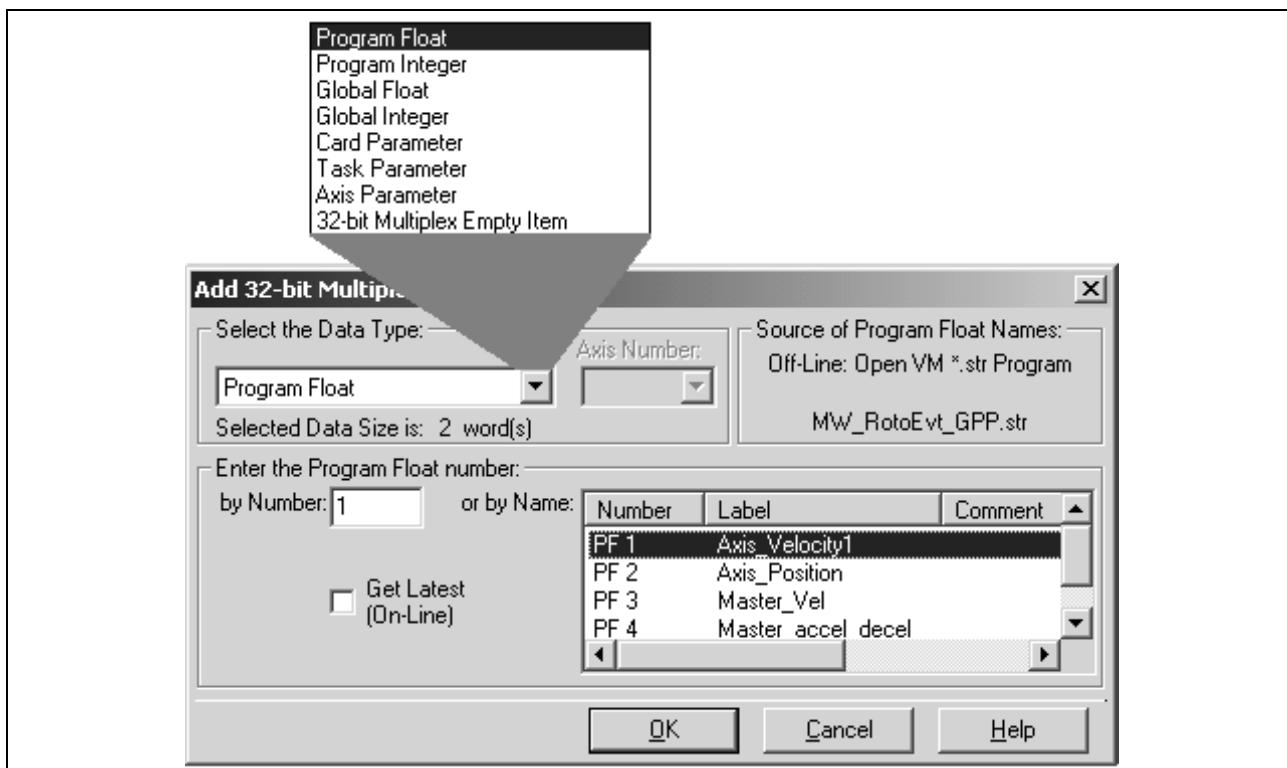


Fig. 8-15: Adding a Multiplex Item to the Container (32-bit example)

Editing the Cyclic Data Lists

To make changes to an existing list, use the following buttons:

Button	Function
Add...	Inserts a new item at the end of the list.
Insert...	Inserts a new item into the list directly before the selected item.
Delete	Removes the selected item from the list.
Edit...	Allows editing of the selected item. (To edit a list item, you may also double-click on it.)
New	Clears up the current list.

Table 8-2: Button Functions in the Cyclic Data Configuration Window

Additional Functions

Several additional functions are available in the Fieldbus Mapper:

Function	Icon	Menu Selection
Download the current fieldbus configuration from the PPC-R		File⇒Get Fieldbus Configuration from PPC
Upload the current fieldbus configuration in the Fieldbus Mapper to the PPC-R		File⇒Send Fieldbus Configuration to PPC
Print the current fieldbus configuration data		File⇒Print
Preview the printout of the current fieldbus configuration data		File⇒Print Preview

Table 8-3: Additional Functions

Getting the Fieldbus Configuration from the PPC

After getting the fieldbus configuration from the PPC, the following information is detected by the system and appears in the configuration list:

- Fieldbus Type Found
- Fieldbus FW (Firmware) Version
- GPP Control FW (Firmware) Version

An example is shown in *Fig. 8-16* below.

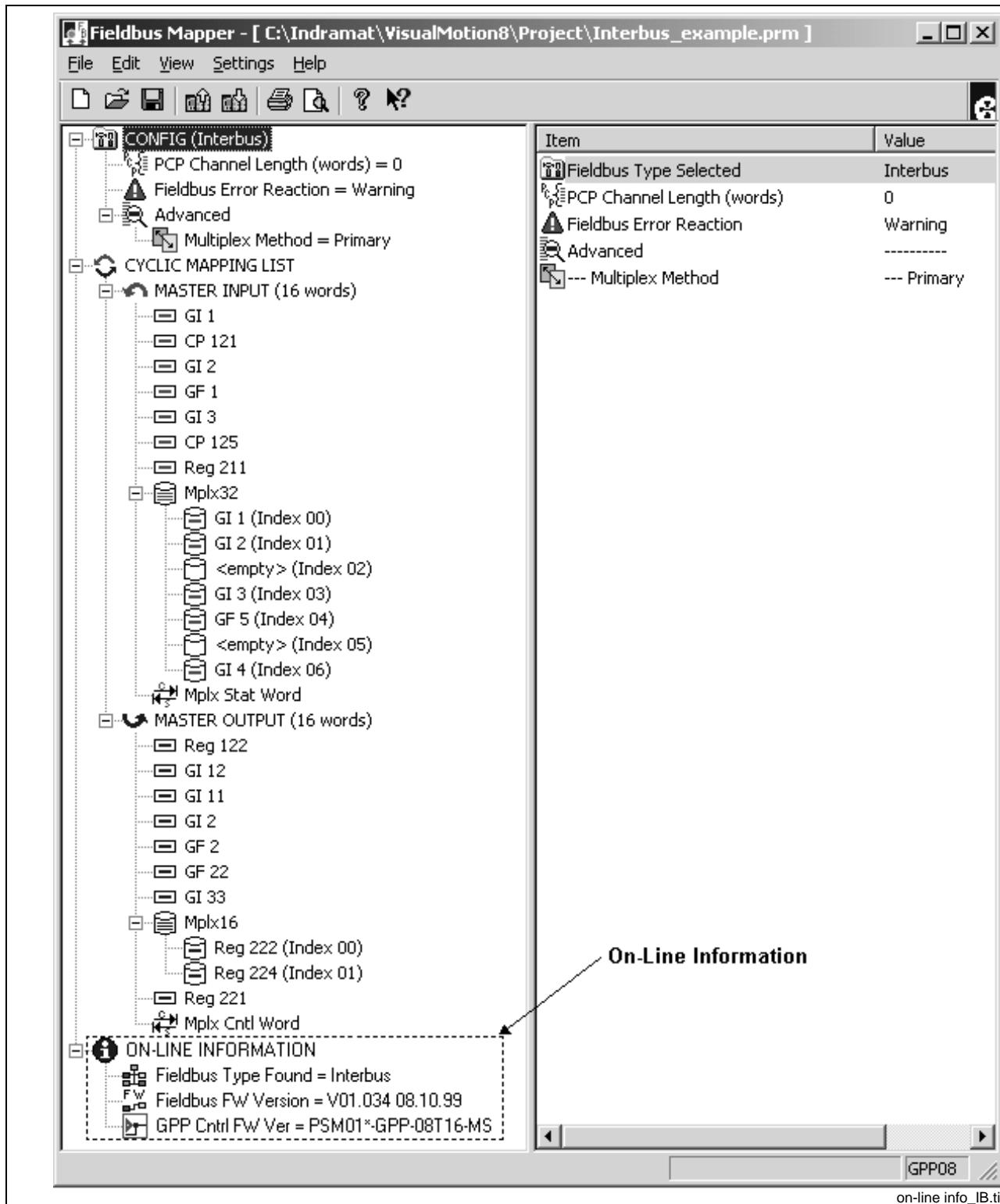


Fig. 8-16: On-Line Fieldbus Configuration Information

8.3 Information for the GPP Programmer

Register 19 Definition (Fieldbus Status)

VisualMotion Register 19 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 8-4 below contains the bit assignment for the diagnostic object 5ff2. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	---	---	---	---	---	---	---	---	---	x5	x4	---	x2	x1

Table 8-4: Bit Assignment for VisualMotion Register 19

Bit Definitions

x1, x2 Status bits for the internal DPR (Dual-Port RAM) communication between the fieldbus slave and the PPC-R:

x1: FB Init. OK , LSB (least significant bit)

x2: FB Init. OK, MSB (most significant bit)

The bit combinations for x1 and x2 are as follows:

Bit 2 (PPC-R)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, or neither the PPC-R nor the fieldbus card have 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 8-5: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

x4 Status bit for the active bus capabilities of the fieldbus slaves (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. Refer to *Fieldbus Error Reaction* on page 8-16 for an explanation of the Fieldbus Error Reaction setting.

0--> The fieldbus slave is not (yet) ready for data exchange.

1--> The fieldbus slave can actively participate on the bus.

x5 Status bit for the non-cyclic (PCP) channel (Non-Cyc Ready)

0--> The non-cyclic channel cannot (yet) be used.

1--> The non-cyclic channel is ready for use by the fieldbus master.

x15 Status bit for the cyclic data output (Cyclic Data Valid):

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 20 Definition (Fieldbus Diagnostics)

VisualMotion Register 20 holds the information for "Fieldbus Diagnostics."

Table 8-6 below contains the bit assignment for the diagnostic object 5ff0. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---".

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
X16	x15	x14	x13	---	---	---	---	---	---	---	---	---	---	---	---

Table 8-6: Bit Assignment for VisualMotion Register 20

Bit Definitions

x13 - x16 Identification of the fieldbus interface card (FB Card Found)

The bit combinations for x13, x14 and x15 are as follows:

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 8-7: Identification of the Fieldbus Interface

Register 26 Definition (Fieldbus Resource Monitor)

The "Fieldbus Resource Monitor" in register 26 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 26 is divided into the following three counter types:

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

Table 8-8 below contains the bit assignment for the fieldbus counters. The assigned bits are labeled with an "x" followed by the bit number.

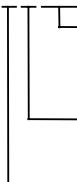
Current Miss Counter				Peak Miss Counter				Fieldbus Time-out Counter							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1

Table 8-8: Bit Assignment for Fieldbus Resource Monitor (Register 26)

Note: View Register 26 in Hexadecimal format too more easily monitor the fieldbus counters.

Hex display format of register 26

0x0000



Fieldbus Time-out Counter (00-FF)

Increments by one every time 10 cyclic mapping list attempts are missed in 10 attempts.

Peak Miss Counter (0-9)

Displays the peak value of the Current Miss Counter.

Current Miss Counter (0-9)

Displays the count of missed cyclic mapping list attempts out of the last 10 attempts

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

Bit Definitions

x13 - x16 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.

x9 - x12 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.

x1 - x8 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. Refer to *Fieldbus Error Reaction* below for an explanation of the Fieldbus Error Reaction setting.

Note: The GPP programmer can monitor the "Current Miss Counter" and define a custom error reaction for missed mapping list update attempts less than 10.

Note: The values in register 26 are read/write and can be reset by the user.

Fieldbus Error Reaction

Note: The Fieldbus Error Reaction setting is active only in SERCOS Phase 4. In all other SERCOS phases, it will be inactive.

You can select how you would like the PPC-R system to react in case of a fieldbus error. This reaction can be set in the "Fieldbus Slave Configuration" 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 stored in Parameter C-0-2635:

Setting	Value in Parameter C-0-2635
Shutdown	0 (default)
Warning Only	1
Ignore	2

Table 8-9: Parameter C-0-2635 Values for Error Reaction Settings

Fieldbus Mapper Timeout

The Fieldbus Mapper continually scans the system for sufficient resources to process the cyclic data mapping lists (2600 and 2601 lists). If 10 out of 10 attempts of the mapping list updates are missed, the system is considered to have insufficient resources and the selected error reaction is evoked, as follows:

If "Shutdown" (0) is set in Parameter C-0-2635, the following error is generated from the PPC-R card: **520 Fieldbus Mapper Timeout**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated: **209 Fieldbus Mapper Timeout**

If "Ignore" (2) is set in Parameter C-0-2635, the system will update as resources become available, but there is no way to monitor whether or not updates actually occur.

Lost Fieldbus Connection

Register 19, bit 4 indicates the status of the fieldbus. Refer to *Register 19 Definition (Fieldbus Status)* on page 8-14 for more specific bit information. The system monitors this bit and evokes the selected error reaction if the bit is low (0), after a fieldbus card is found. A typical situation that will cause this condition is the disconnection of the fieldbus cable from the fieldbus card.

If "Shutdown Control" (0) is set in Parameter C-0-2635, the following error is generated from the PPC-R (active in SERCOS Phase 4 only): **519 Lost Fieldbus Connection**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated (active in SERCOS Phase 4 only): **208 Lost Fieldbus Connection**

If "Ignore" (2) is set in Parameter C-0-2635, there is no noticeable reaction when Register 19 Bit 4 goes low, unless the GPP application program is customized to evoke a special reaction.

Troubleshooting Tip:

If a fieldbus card is not found on the system, the Error Reaction setting will be ignored. If you have a fieldbus card and the Error Reaction is not responding as expected, the system may not "see" your fieldbus card.

8.4 Information for the PLC Programmer

Important: The fieldbus master's access of the cyclic channel must be consistent over the entire channel length in order to establish reliable multiplexing communications.

Multiplexing

Primary Multiplex Method (for Inconsistent Masters)

Explanation of the Master Consistency Problem

The PPC-R fieldbus slave interfaces can guarantee consistency.

However, some fieldbus masters can only guarantee byte, word or double word consistency. If the master is only word-consistent, it is possible that the master cannot transfer the data and the control word of one multiplex index consistently from the PLC to the fieldbus. Therefore, it is necessary to have a second multiplex method where both input data and output data require the handshake bits to update via the fieldbus.

Fig. 8-17 below illustrates the control word definition for the Primary Multiplex Method.

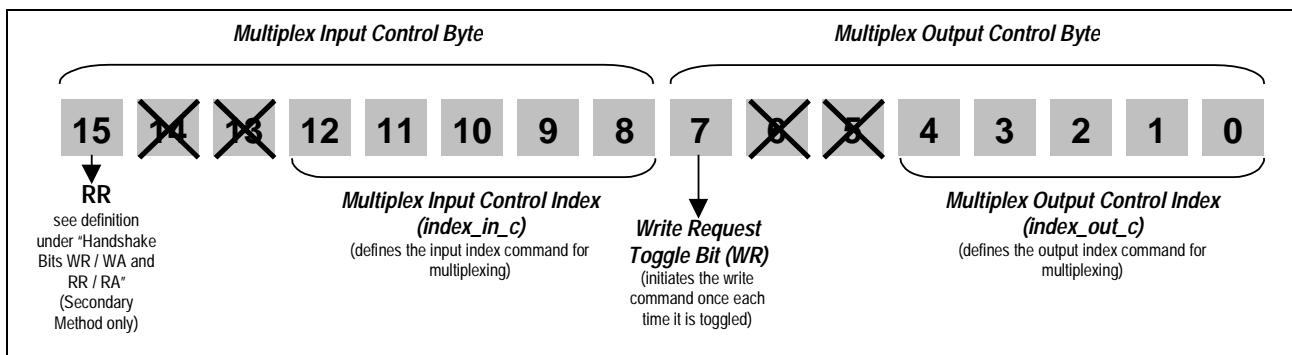


Fig. 8-17: Control Word Definition, Primary Multiplex Method

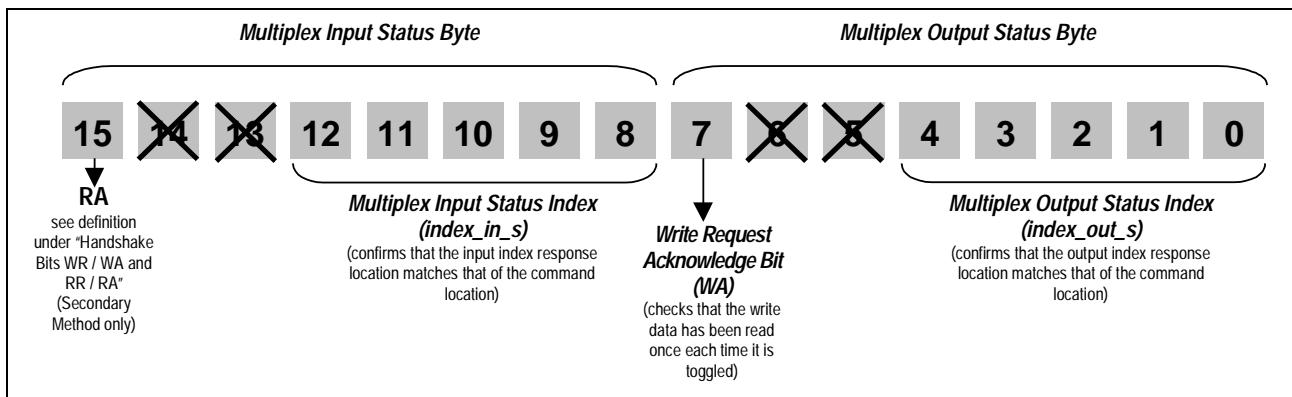


Fig. 8-18: Status Word Definition, Primary Multiplex Method

The Primary Multiplex Method has the following features:

- You can transfer a different index from master to slave as from slave to master.
- The handshake bits for both reading and writing of this multiplex channel make the multiplexing possible on inconsistent systems (masters).

Handshake Bits WR and WA

WR and WA are handshake bits that allow the controlled writing of data via the multiplex channel. WR and WA control the data transfer for writing data_out (data send from master to slave).

WR == WA:

- tells the master that the slave has received the last multiplex data_out. The master can now send new data_out.
- tells the slave to do nothing, because the master has not yet put new consistent data_out on the bus.

WR != WA:

- tells the slave to do something, because the master has now put consistent new data_out on bus.
- tells the master to do nothing, because the slave has not yet received the latest multiplex data_out.

Handshake Bits RR and RA

RR (Read Request) and RA (Read Acknowledge) are handshake bits that allow a controlled data transfer and use of the multiplex channel on inconsistent masters. RR and RA control the data transfer for reading data_in (data send from slave to master).

RR == RA:

- tells the master that the slave has sent the requested data_in. The master can now read the data_in and request new data_in.
- tells the slave to do nothing, because the master has not yet put new consistent data on the bus.

RR != RA:

- tells the slave to put new data_in on the bus, because the master requests new data_in.
- tells the master to do nothing, because the slave has not yet put the latest requested multiplex data_in on the bus.

Master Communications (Primary Multiplex Method)

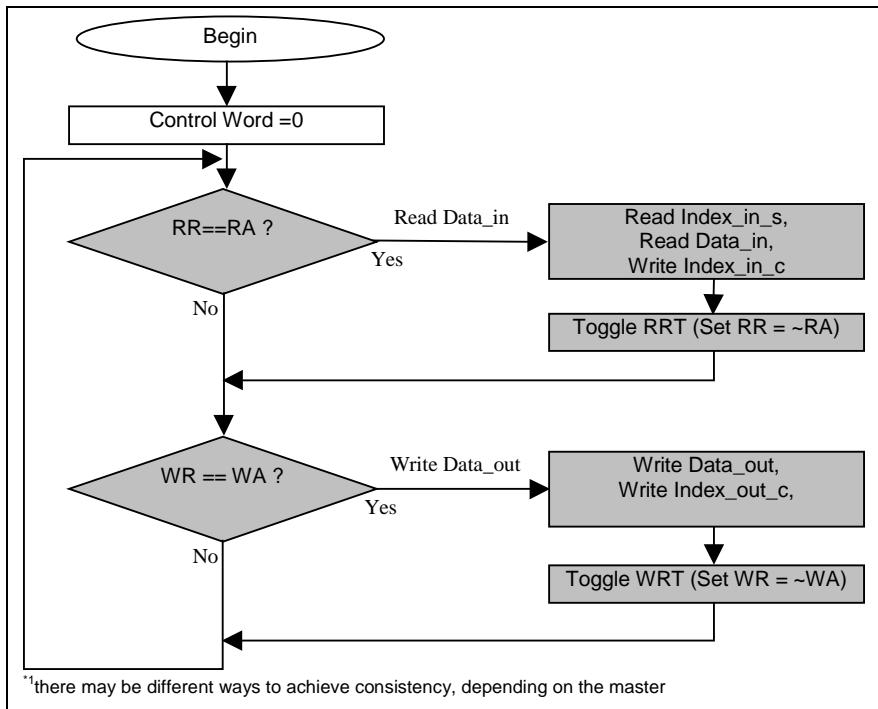


Fig. 8-19: Primary Multiplex Method, Master Communications

For some masters, it could be enough to first write data and then the control word. For other masters, you may have to implement a delay time (this time could be different from master to master) before writing WR = ~WA.

Secondary Multiplex Method (for Consistent Masters only)

Important: You should use the Secondary Multiplex Method **only** for a master that is consistent over the entire cyclic channel. The Primary Multiplex Method is available for inconsistent masters. Refer to *Primary Multiplex Method (for Inconsistent Masters)* on page 8-18.

The advantage of the Secondary Method is easier handling of input data for consistent masters.

Note: The meanings of the control and status words are the same as for the Primary Multiplex Method. The only difference is the toggle bits RR and RA, which are used only in the Primary Method.

Control Word and Status Word

Control Word

The control word is transferred in the multiplex channel from master to slave. It tells the slave in which index the data is being transferred from master to slave and in which index the data is requested from slave to master.

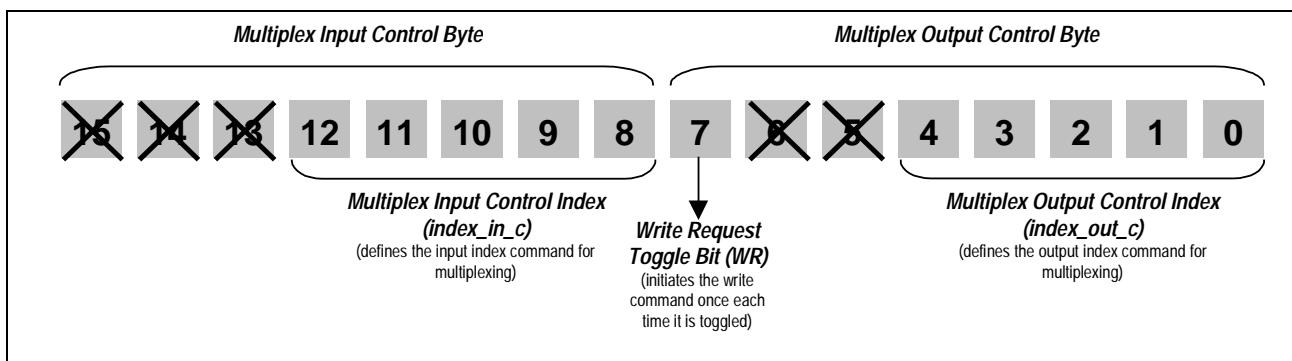


Fig. 8-20: Control Word Definition, Secondary Multiplex Method

Index_out_c: tells the slave in which index the data are transferred from master to slave (out = master -> slave, _c = element of control word).

Index_in_c: tells the slave in which index the data is requested from slave to master (in = slave -> master, _c = element of control word).

WR (Write Request): handshake bit (refer to meaning of WR and WA).

Note: Input data via the Multiplex Channel is continually being updated.

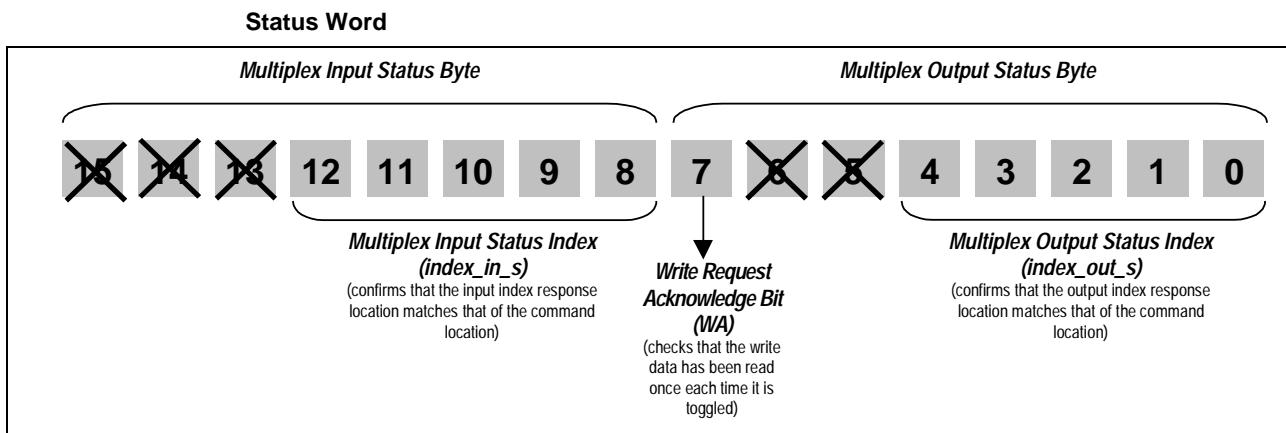


Fig. 8-21: Status Word Definition, Secondary Multiplex Method

- **Index_out_s:** acknowledges index written by the master (out = master -> slave, _s = element of status word).
- **Index_in_s:** tells the master which index is transferred from slave to master in the actual process data cycle (in = slave -> master, _s = element of status word).
- **WA (Write Acknowledge):** Handshake bit (refer to meaning of WR and WA under the Primary Multiplex Method).

Master Communications (Secondary Multiplex Method)

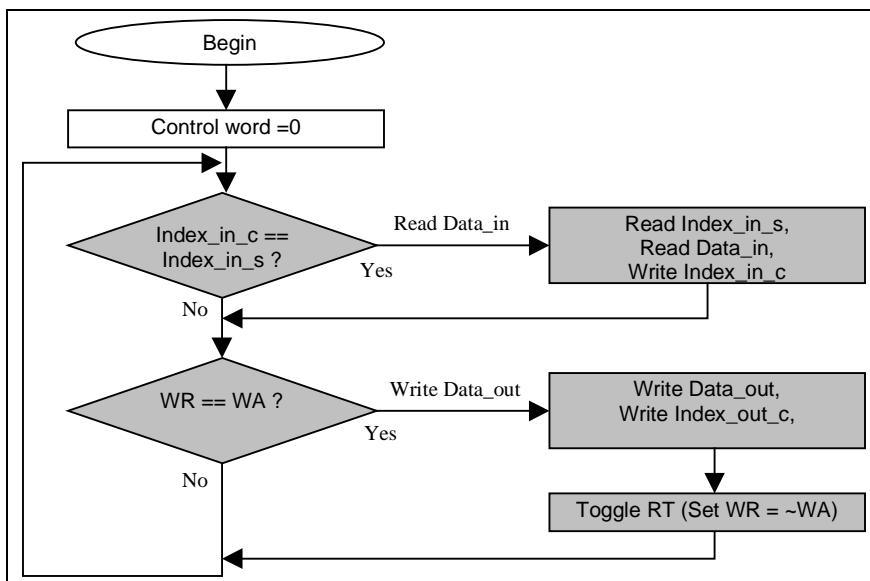


Fig. 8-22: Secondary Multiplex Method, Master Communications

Programming Example

To aid in implementing the multiplex function in a PLC program, the following flow chart shows two ways of reading and writing data. Reading and writing can be executed separately, which allows the input data to be updated about 30% faster. The “Read Data” example would be placed at the beginning of a PLC program the “Write Data” example at the end.

Combined reading and writing makes the PLC program simpler, especially when using the same index for both transfer actions.

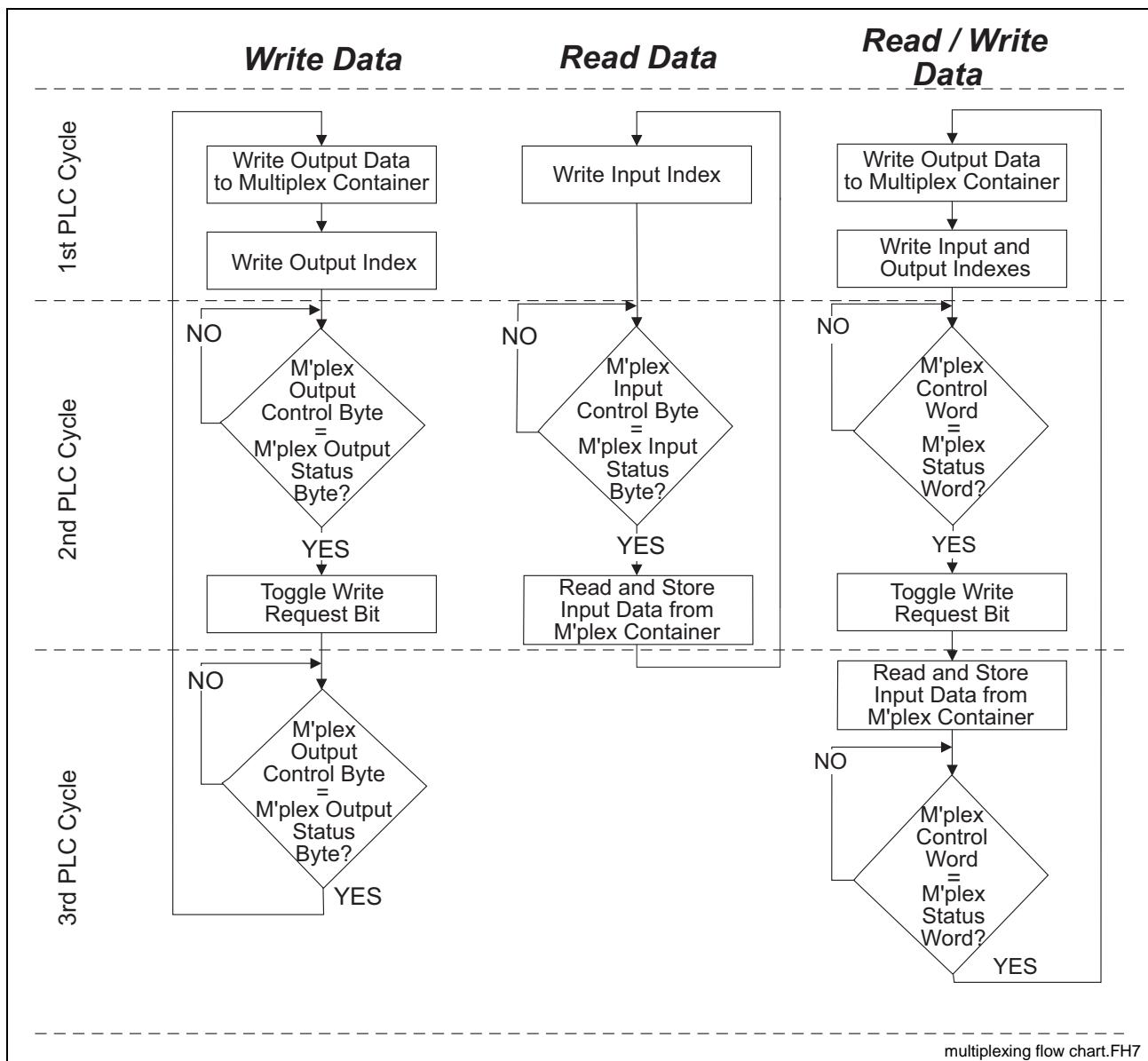


Fig. 8-23: Flow Chart of Multiplex Programming Examples (Secondary Method)

Non-Cyclic Data Access via the Non-Cyclic (PCP) Channel

To support the configuration of drives and the access to parameters via Interbus, Indramat supports the PCP channel.

Note: The PCP Channel is fixed at a length of 2 words when enabled.

To read or write a VisualMotion data type non-cyclically, a special set of pre-defined objects is used in the PCP channel. Refer to the documentation provided by the fieldbus manufacturer for access and support of the PCP channel.

The following methods for transferring data are available in the PCP channel:

- Mapped Data
- Data Exchange Objects

Mapped Data

Mapped data is the most powerful feature of the PPC-R non-cyclic fieldbus interface. Through mapped data, the user has access to virtually every PPC-R data type over the fieldbus. It is easy to implement from the PLC side and requires no setup on the PPC-R side.

To access a data type over the fieldbus, it has to be specified by an address that consists of an index and a subindex. The index and subindex for each data type can be calculated by a formula (refer to *Accessing Mapped Data* on page 8-26).

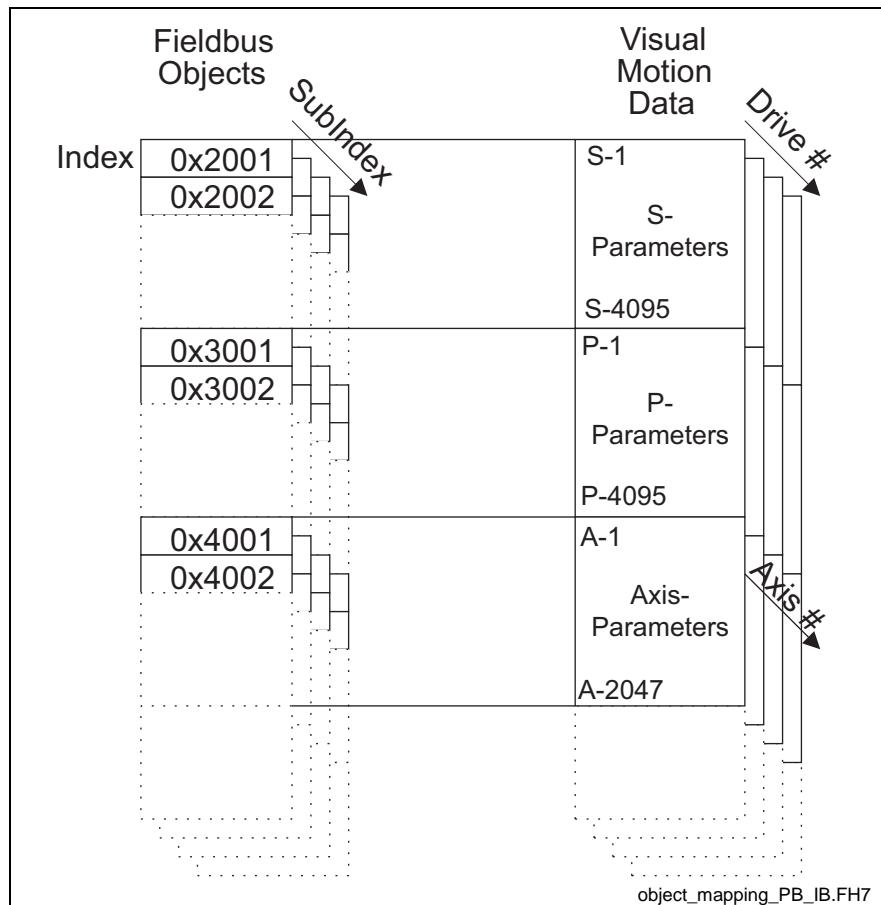


Fig. 8-24: Mapped Data

Mapped data can be used with the following parameters and values:

- S-Parameters (SERCOS Drive S-Parameters)
 - P-Parameters (SERCOS Drive P-Parameters)
 - A-Parameters (PPC Axis Parameters)
 - C-Parameters (PPC C System parameters)
 - T-Parameters (PPC Task parameters)
- size and
format
depend on
parameter *¹

PF-Values (PPC Program Float data, 32 bit – 2 words, IEEE format) *²

GI-Values (PPC Global Integer data, 32 bit – 2 words) *²

GF-Values (PPC Global Float data, 32 bit – 2 words, IEEE format) *²

PI-Values (PPC Program Integer data, 32 bit – 2 words) *²

Reg.-Values (PPC Register data, 16 bit – 1 word) *³

Data Exchange Objects (0x5E70 – 0x5E73) (embedded ASCII Protocol)

*You may notice that parameters accessed via the non-cyclic (Parameter) channel are not always the same size as reported from the attribute field. This is so that the data sizes correspond with the way the different data types are handled in the cyclic channel (Registers are always set to 16-bit size and Parameters are cast to 32-bit size, even if they actually use less space).

1. When **writing** mapped data to a VisualMotion Parameter, you must send the size data corresponding to that of the attribute field within the parameter.
 - a.) For 32-bit parameters, you must send a data size of 32 bits (otherwise, VM error #07 is returned).
 - b.) For 16-bit parameters, you must send a data of size 16-bits. If, for this case, you send data of size 32 bits, one of the following occurs:
 - i.) For parameters of type 16-bit unsigned, only the Low word is stored, and the High word is ignored.
 - ii.) For parameters of type 16-bit signed, bits 0-14 of the low word along with the sign bit #31 are used, and the remaining bits are ignored.
 - c.) For String Parameters (e.g. S-0-0142), you must send the size of the string to write.
 - d.) All other Parameter Types (list parameters, command parameters, etc), are not supported for mapped data.

When reading mapped data from a VisualMotion Parameter, there are 3 possible cases of sizes returned:

- a.) If the parameter type is a string, you receive the number of bytes corresponding to the length of the string.
 - b.) If the parameter is 32-bit or less, you receive a cast 32-bit value for this parameter. This implies that 16-bit parameters are returned as cast in to 32-bit values.
 - c.) All other parameter types (e.g. list parameters, command parameters, etc.), are not supported for mapped data.
 2. When writing mapped data to a VisualMotion Program Float, Program Integer, Global Float, or Global Integer, the data size must be 32-bits (2 words). Any other size returns a VM error #07 (Invalid Data Format).
- When reading mapped data from a VisualMotion Program Float, Program Integer, Global Float, or Global Integer, the data size returned is always 32-bit (2 words).
3. When writing mapped data to a VisualMotion Register, the data must be 16-bits (1 word). Any other size returns a VM error #07 (Invalid Data Format).

When reading mapped data from a VisualMotion Register, the data size returned is always 16-bit (1 word).

Object Index

The index refers to the particular fieldbus slave object that a VisualMotion data type is (automatically) mapped. This object allows for simple, indirect access to VisualMotion data types, and it is combined with the subindex to create a direct relationship to the VisualMotion data types. The available objects can be calculated using the formulas in *Accessing Mapped Data* on page 8-26.

Object SubIndex

The subindex refers to an additional piece of information necessary to obtain direct access to VisualMotion data types. The reference of the subindex depends on the data type in question. For example, the SubIndex refers to the drive number when accessing S and P parameters. However, the subindex refers to the task number when referring to task parameters. The available subindex ranges can be calculated using the formulas in *Accessing Mapped Data* on page 8-26.

Data Exchange Objects

The four data exchange objects 5E70 to 5E73 represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the PPC-R card. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. Both the VisualMotion ASCII message and the fieldbus transfer message must be formulated.

Table 8-10 lists the available data exchange objects and their sizes.

Data Exchange Object	Data Length (in bytes)
5E70	16
5E71	32
5E72	64
5E73	128

Table 8-10: Length of the Data Exchange Objects

Handling a Data Exchange Object

When mapped objects are not capable of transferring the desired data, a Data Exchange Object can be used.

The same procedures for writing and reading data apply to the Data Exchange Object.

Selecting a Data Exchange Object

Depending on the length of a VisualMotion ASCII message, any of these data exchange objects can be selected.

Note: The entire data length of the data exchange object must always be transmitted even if the VisualMotion ASCII message is shorter.

For example, if you want to transmit an ASCII message of 42 bytes, you must use object 5E72. To avoid a response error from the fieldbus slave, you must append 22 "Null" characters to the end of the ASCII message to complete a data size of 64 bytes.

Note: The checksum for the VisualMotion ASCII protocol is NOT used with the data exchange object. If the checksum is sent as part of the string, it will be ignored, and no checksum will be sent in the VisualMotion ASCII response messages. To ensure data integrity, the fieldbus protocols support a low-level checksum.

Transmission Sequence via a Data Exchange Object

Note: For the data exchange object, two transmission sequences (and two response sequences) are required, to send the read or write message to and then receive the response message from the PPC-R card.

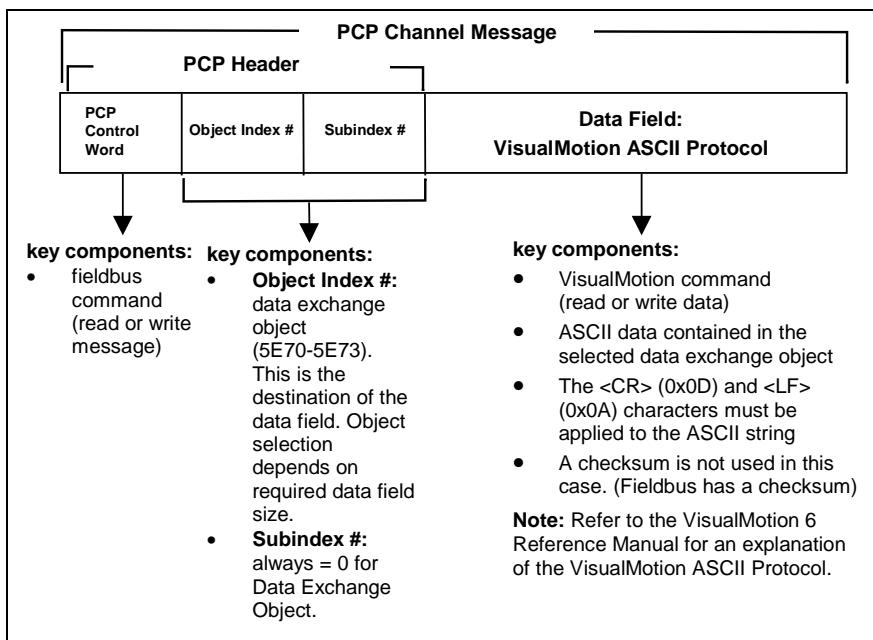


Fig. 8-25: Format of a PCP Channel Message using a Data Exchange Object

For information about reading and writing data using the PCP channel, consult the documentation provided by the fieldbus manufacturer.

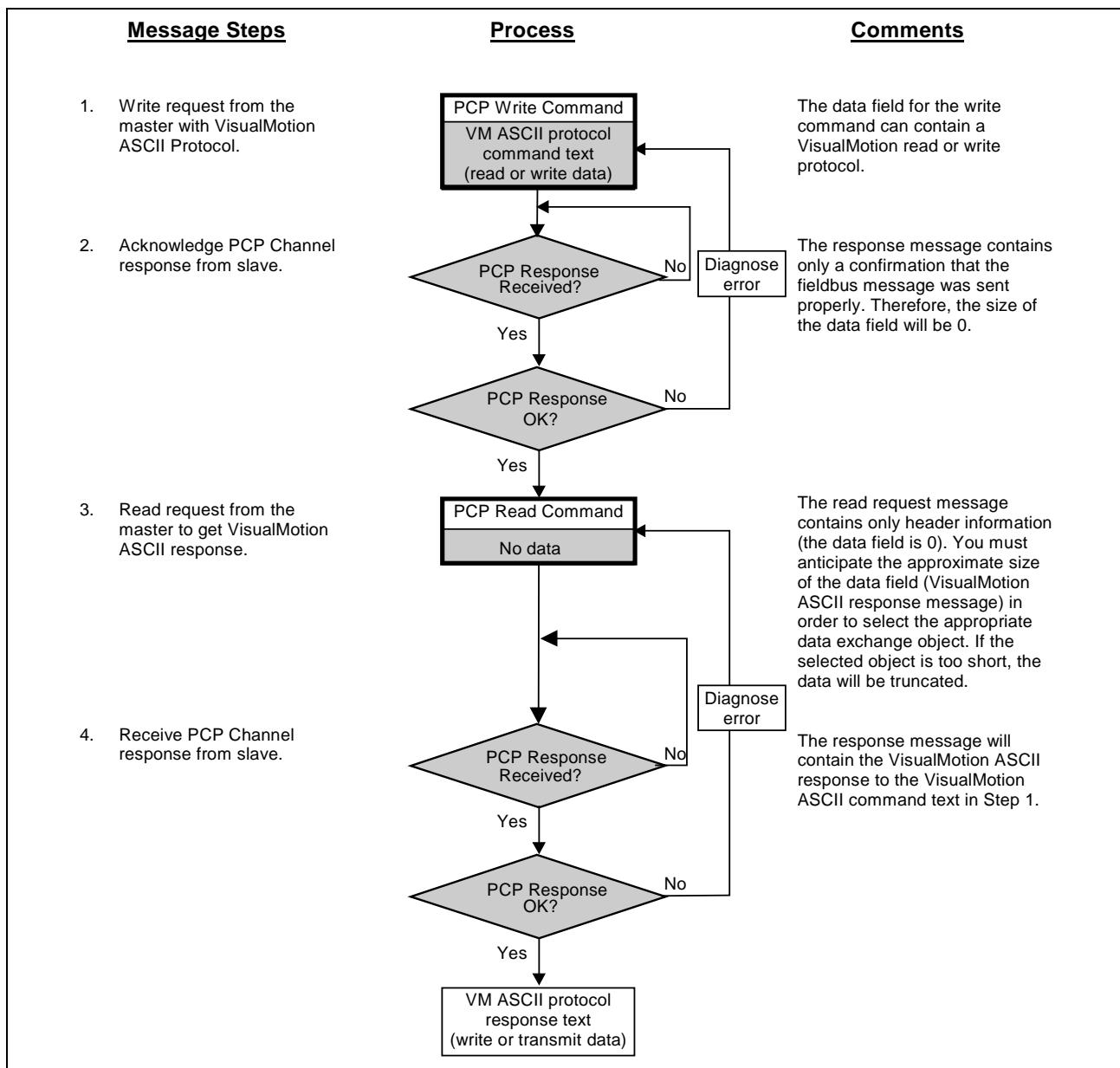


Fig. 8-26: PCP Channel Procedure, Using Data Exchange Object

Accessing Mapped Data (via the PCP Channel)

Indramat has pre-configured a number of VisualMotion data types to Interbus indexes and subindexes. We call this concept **mapped data**. These data types can be accessed via the Interbus PCP Channel. The index and subindex for each of these data types can be calculated using the formulas in *Table 8-11* below.

	Object Index #	SubIndex #	Formula
Data Exchange Object	0x5E73	0x00	
	----	----	
	0x5E70	0x00	
<FREE> (349 objects available)	0x5E65	0xFF	
	----	----	(with SubIndex)
	0x5D14	0x01	
Program Integers (Int 1 – Int 5100)	0x5D13	0xFF	Index = 0x5D00 + [(Program Integer – 1) \ 255]
	----	----	
	0x5D00	0x01	SubIndex = Program Integer – [(Index – 0x5D00) * 255]
Program Floats (Float 1 – Float 5100)	0x5CFF	0xFF	Index = 0x5CEC + [(Program Float – 1) \ 255]
	----	----	
	0x5CEC	0x01	SubIndex = Program Float – [(Index – 0x5CEC) * 255]
<FREE> (235 objects available)	0x5CEB	0xFF	
	----	----	(with SubIndex)
	0x5C01	0x01	
Global Integers (GInt 1 – GInt 2550*)	0x5C00	0xFF	Index = 0x5BF7 + [(Global Integer – 1) \ 255]
	----	----	
	0x5BF7	0x01	SubIndex = Global Integer – [(Index – 0x5BF7) * 255]
Global Floats (GFloat 1 – Gfloat 2550*)	0x5BF6	0xFF	Index = 0x5BED + [(Global Float – 1) \ 255]
	----	----	
	0x5BED	0x01	SubIndex = Global Float – [(Index – 0x5BED) * 255]
<FREE> (245 objects available)	0x5BEC	0xFF	
	----	----	(with SubIndex)
	0x5AF8	0x01	
Registers (Reg. 1 – Reg. 2550**)	0x5AF7	0xFF	Index = 0x5AEE + [(Register – 1) \ 255]
	----	----	
	0x5AEE	0x01	SubIndex = Register – [(Index – 0x5AEE) * 255]
T-Parameters (T-0-0001 – T-0-1020)	0x5AED	0x04	Index = 0x56F1 + T-Parameter
	----	----	
	0x56F1	0x01	SubIndex = Task Number
<FREE> (241 objects available)	0x56F0	0xFF	
	----	----	(with SubIndex)
	0x5600	0x01	
C-Parameters (C-0-0001 - C-0-3583)	0x55FF	0x01	Index = 0x4800 + C-Parameter
	----	----	
	0x4801	0x01	SubIndex = 1
A-Parameters (A-0-0001 - A-0-2047)	0x47FF	0x63	Index = 0x4000 + A-Parameter
	----	----	
	0x4001	0x01	SubIndex = Axis Number
P-Parameters (P-0-0001 - P-0-4095)	0x3FFF	0x63	Index = 0x3000 + P-Parameter
	----	----	
	0x3001	0x01	SubIndex = Drive Number

	Object Index #	SubIndex #	Formula
S-Parameters (S-0-0001 - S-0-4095)	0x2FFF	0x63	Index = 0x2000 + S-Parameter
	----	----	
	0x2001	0x01	SubIndex = Drive Number
<Reserved>	0x1FFF	----	
	----	----	
	0x0000	----	

* current limitation: first 256 global integers/floating-point numbers.

**current limitation: first 512 registers.

Table 8-11: Formulas for Determining Mapped Objects

Example Lookup Tables for Mapped Objects

Card (C) Parameters

The following is an example lookup table for C-Parameters, when using mapped objects.

Example Look-up Chart for: C-Parameters CP 0.Y ==> CP = Card Parameter Y = Parameter Number									
Index									
SubIndex =									
	0x4801	0x4802	0x4803	0x48FF	0x4900	0x55FE	0x55FF
0x01	CP 0.1	CP 0.2	CP 0.3		CP 0.255	CP 0.256		CP 0.3582	CP 0.3583

Table 8-12: Mapped Object Lookup Table for C-Parameters

Axis(A) Parameters The following is an example lookup table for A-Parameters, when using mapped objects. The same formula also applies to SERCOS (S) and Task (T) Parameters.

Example Look-up Chart for:		A-Parameters AP X.Y		==>		AP = Axis Parameter	
				X = Axis Number			
				Y = Parameter Number			
Index							
SubIndex =	0x4001	0x4002	0x4003	0x40FF	0x4100
	0x01	AP 1.1	AP 1.2	AP 1.3		AP 1.255	AP 1.256
	0x02	AP 2.1	AP 2.2	AP 2.3		AP 2.255	AP 2.256
	0x03	AP 3.1	AP 3.2	AP 3.3		AP 3.255	AP 3.256
	:	:	:	:	:	:	:
	:	:	:	:	:	:	:
	0x63	AP 99.1	AP 99.2	AP 99.3		AP 99.255	AP 99.256

Table 8-13: Mapped Object Lookup Table for A-Parameters

Product-Specific (P) Parameters The following is an example lookup table for P-Parameters, when using mapped objects.

Example Look-up Chart for:		P-Parameters PP X.Y		==>		PP = SERCOS P-Parameter (set 0 only)	
				X = Drive Number			
				Y = Parameter Number			
Index = (Class ID && Instance ID for DeviceNet)							
SubIndex = (Attribute ID for DNet)	C118, In1	C118, In2	C118, In3	C118, In255	C119, In1
	0x3001	0x3002	0x3003	0x30FF	0x3100
	0x01	PP 1.1	PP 1.2	PP 1.3		PP 1.255	PP 1.256
	0x02	PP 2.1	PP 2.2	PP 2.3		PP 2.255	PP 2.256
	0x03	PP 3.1	PP 3.2	PP 3.3		PP 3.255	PP 3.256
	:	:	:	:	:	:	:
	0x63	PP 99.1	PP 99.2	PP 99.3		PP 99.255	PP 99.256

Table 8-14: Mapped Object Lookup Table for P-Parameters

Integers The following is an example lookup table for Integers, when using mapped objects. The same formula also applies to Floats, Global Integers, Global Floats and Registers.

Example Look-up Chart for:		VM Program Integers			PI 0.Y	==>	PI = Program Integer
							Y = Program Integer Number
		Index					
SubIndex =	0x5D00	0x5D01	0x5D02	0x5D13		
	0x01	PI 1	PI 256	PI 511		PI 4846	
	0x02	PI 2	PI 257	PI 512		PI 4847	
	0x03	PI 3	PI 258	PI 513		PI 4848	
	:	:	:	:	:	:	
	:	:	:	:	:	:	
	0xFF	PI 255	PI 510	PI 765		PI 5100	

Table 8-15: Mapped Object Lookup Table for Integers

9 VisualMotion Networking Functionality

9.1 Ethernet Interface

The Ethernet option card resides in the PPC-R control and contains its own TCP/IP (*Transmission Control Protocol/Internet Protocol*) stack. The TCP/IP stack enables the Ethernet interface to transmit data over the network or Internet and communicate with VisualMotion Toolkit via the DDE Server. The PPC-R control can be ordered preconfigured with an Ethernet card.

Note: Refer to the available PPC-R hardware configurations in chapter 11 of the *VisualMotion 8 Project Planning Manual*.

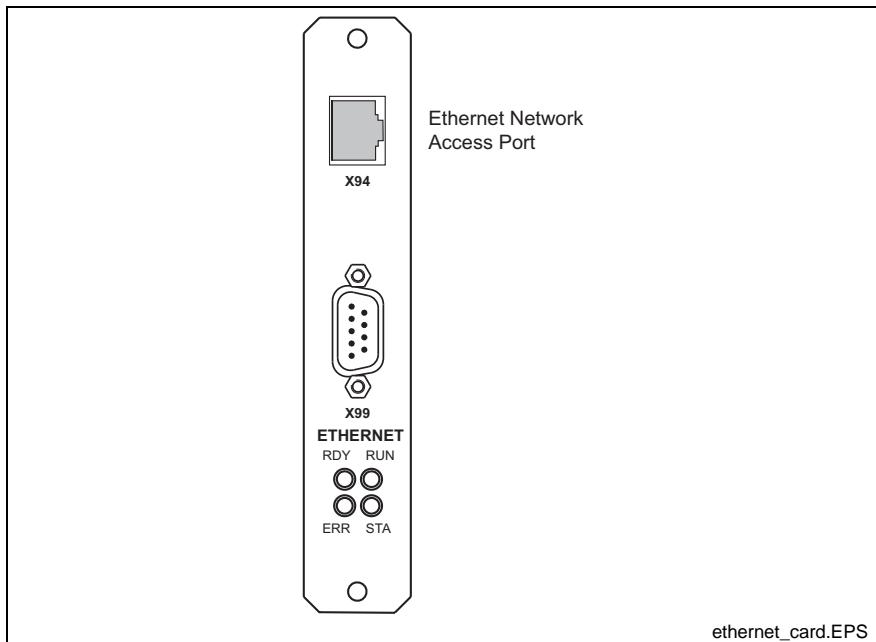


Fig. 9-1: Ethernet Interface Card

Hardware Requirements

A Pentium2-based PC running either Windows 2000/NT with all available network features installed is preferred. A PC running Windows 98/95 can be used but will provide a reduced level of network functionality.

Technical Data

The Ethernet card provides the following technical data:

- *Transfer rate:* up to 20 Mbps (**Megabits per seconds**)
- *Transmission modes:* half and full duplex
- *Communication Protocols:* TCP/IP
- *Cable type:* RJ-45

Local Area Network (LAN) Switch

The Ethernet interface card can be connected through a LAN Switch (recommended) receiving messages unique to that particular control. A switch can also provide parallel communication (full duplex). A switch can increase data transfer from 10 Mbps up to 20 Mbps.

The following LAN Switch device can be used:

- 3com 3C16734B (dual-speed)

A HUB (common connection point for devices in a network) can also be used to connect the Ethernet card to the network. Only half-duplex is possible when using a HUB.

DDE Server Communication

The DDE Server supports a transport mechanism for communicating to the control (PPC-R control with GPP Firmware) using TCP/IP. Using VisualMotion Toolkit, the user selects the “Network” connection method and specifies a “Card Number”. The DDE Server uses this information to initiate a connection-oriented path with the motion control (GPP). Fig. 9-2 conceptually illustrates the communication between a host PC and the control via Ethernet.

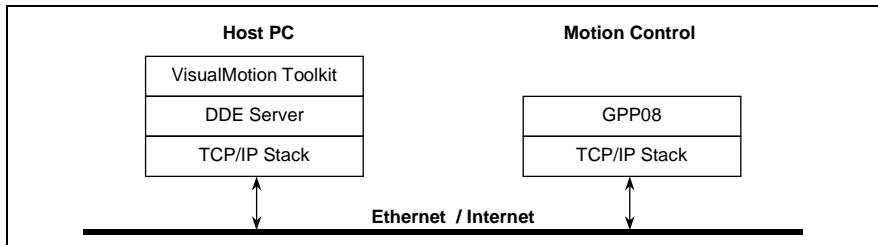


Fig. 9-2: PC to Control Communication via Ethernet

VisualMotion Ethernet Registers

The following VisualMotion registers provide status and monitor network communications for diagnostic purposes:

- Register 50 – Ethernet status. The bits in this register indicate a status of the interface and the current message being processed.
 - Bit 1 – Indicates the interface is present (0 = not present, 1 = present)
 - Bit 9 – A request has been received and the Ethernet callback function is executing.
 - Bit 10 – Response is pending. The message is being processed by the motion control (GPP).
 - Bit 11 – The message processing is complete.
 - Bit 12 – The response message has been sent to the DDE Server.
 - Bit 13 – Sets when a failure to communicate request to GPP Ethernet handler exist.
 - Bit 16 – An invalid protocol has been received (standard or encrypted ASCII).
- Register 51 – Standard ASCII message count. This register indicates the number of messages that have been received in standard ASCII protocol. It is a 16-bit integer and will rollover when the maximum is reached.
- Register 52 – Encrypted ASCII message count. This register indicates the number of messages that have been received in encrypted ASCII protocol. It is a 16-bit integer and will rollover when the maximum is reached.
- Register 53 – Invalid protocol message count. This register indicates the number of messages that have been received in which the protocol cannot be determined. It is a 16-bit integer and will rollover when the maximum is reached.

Ethernet Card Setup

Before an Ethernet card can be accessed, the following control parameters must be configured using VisualMotion Toolkit's *Overview Parameter* tool.

- C-0-0400 – Card IP Address
- C-0-0401 – Card Subnet Mask
- C-0-0402 – Card Gateway IP Address
- C-0-0403 – Half / Full Duplex Mode
- C-0-0405 – Card Network Password

In addition to the setup parameters, the following read-only parameters are supported:

- C-0-0404 – Card Network Access Control (read-only via Ethernet)

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-0-0406 – CIF Ethernet Card Hardware ID
 - C-0-0407 – CIF Ethernet Card Firmware Version
 - C-0-0408 – CIF Driver version string

The following steps are used to configure an Ethernet card via a serial connection (IKB0005) to VisualMotion Toolkit.

1. Power up the control, with a connected Ethernet card, and start VisualMotion Toolkit.
2. Select **On-line Data** ⇒ **Parameters** to open the *Overview Parameter* tool.
3. Modify control parameter C-0-0400 and enter the Ethernet card's IP Address in dot notation, for example "172.18.11.205".
4. Modify control parameter C-0-0401 and enter the Ethernet card's Subnet Mask in dot notation, for example "255.255.0.0".
5. Modify control parameter C-0-0402 and enter the Ethernet card's Gateway IP Address in dot notation, for example "172.16.1.1".

Note: The Ethernet card's IP Address, Subnet Mask and Gateway IP Address is provided to the user by their respective IT department. Every Ethernet card must have a unique IP Address assigned.

6. Modify control parameter C-0-0403 and set the transmission mode to either half duplex or full duplex. Typing "HALF" or "FULL" in uppercase letters modify this parameter.

Note: Full-duplex (20 Mbps) can only be achieved if connecting to the Ethernet card via a LAN switch.

7. Modify control parameter C-0-0405 and enter an alphanumeric network password, up to 20 characters, that will be used to modify the access level to the control.

Note: When connected to the control via an Ethernet connection, the password in control parameter C-0-0405 is displayed as asterisks. Only the user with serial access to the control can view the actual text password and modify it if desired.

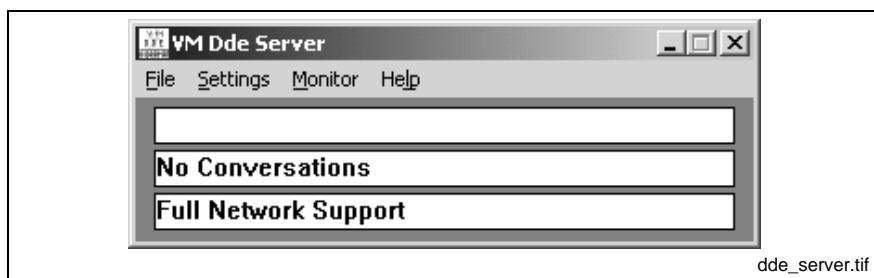
8. Close VisualMotion Toolkit and cycle power to the control in order for these changes to take affect.

Note: After the control is powered up, the **RDY** LED should be on and the **RUN** and **STA** LED's should flash continuously.

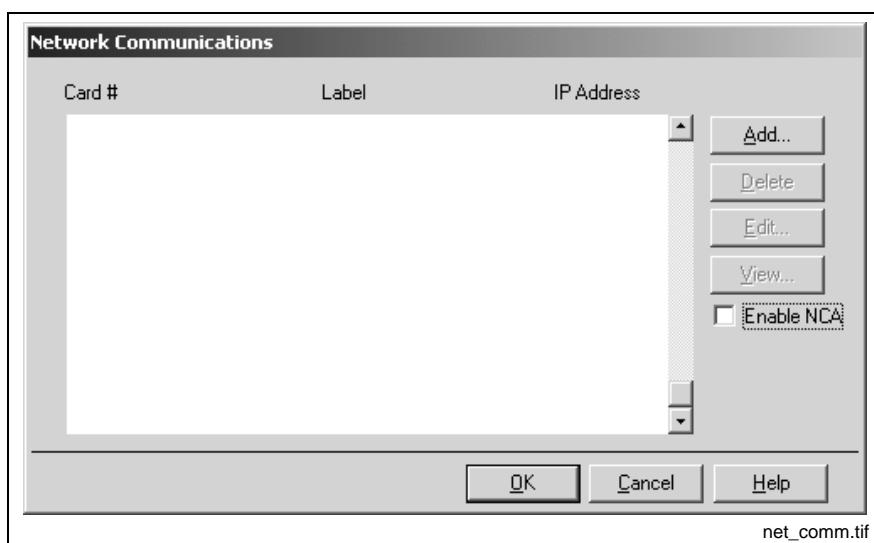
VisualMotion DDE Server Settings

Communication between VisualMotion Toolkit and an Ethernet-ready control is performed via the DDE Server. The following steps outline the DDE Server setup procedure.

1. Start VisualMotion Toolkit and launch the DDE Server shown in the figure below by selecting **VM Tools** \Rightarrow **CLC_DDE -> Release8**.



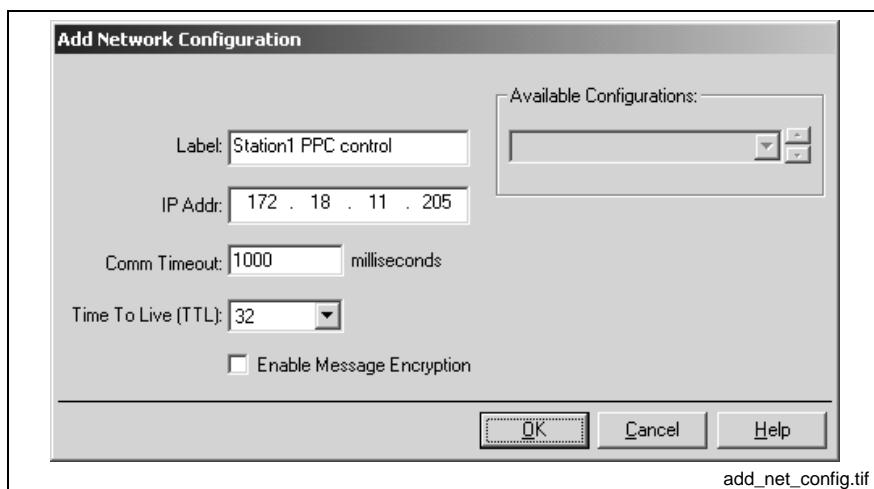
2. Select **Settings** \Rightarrow **Network Communications** from the DDE Server to open the *Network Communications* window below.



3. Click on the **Add** button to add a network configuration.

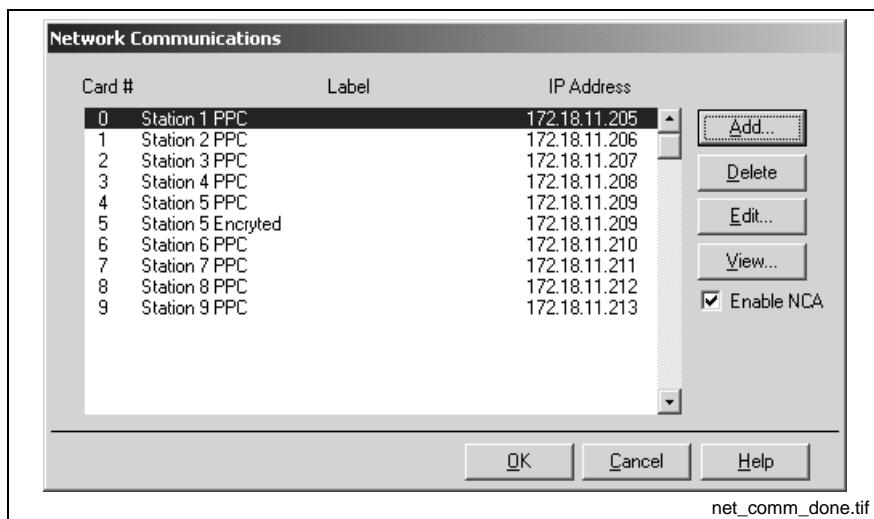
4. In the *Add Network Configuration* window below, enter the following information:
- A *label* (up to 20 characters) that identifies the control.
 - The control's Ethernet *IP Address* in dot notation, for example "172.18.11.205".

Note: The "Communication Timeout" (1000 ms) and "Time to Live" (32) default values can be modified if desired.
The "Enable Message Encryption" feature encrypts messages before transporting them to the control, providing another layer of security.



5. Click on the **OK** button to return to the *Network Communication* window where the new configuration will be displayed.

Note: The order in which configurations are added to the *Network Communication* window determine the control's card number.



Note: A maximum of 10 network connections are allowed in the *Network Communication* window. A DDE Server message will be issued if the **Add** button is pressed for an 11 connection.

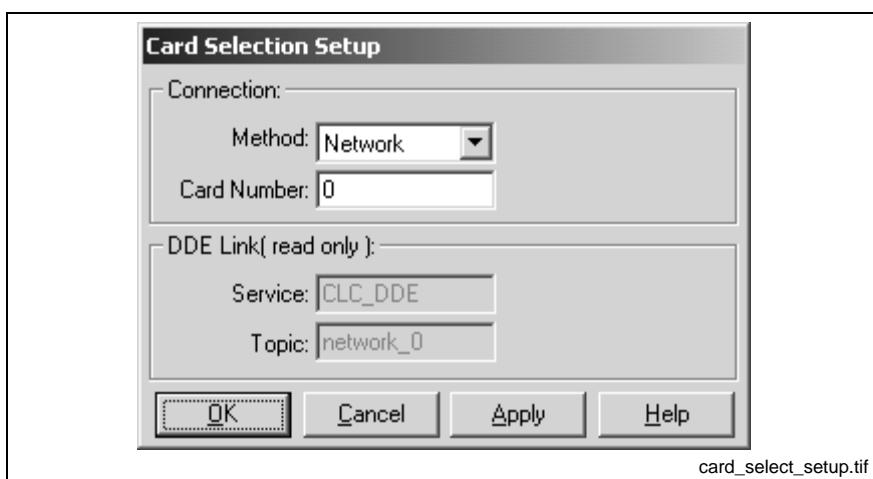
6. Checking the "Enable NCA" option from the *Network Communication* window enables the Network Communication Accelerator. This feature accelerates communication to the control by creating an open message loop between VisualMotion Toolkit.
7. Press the **OK** button to return the DDE Server main window.

Establishing Communication using VisualMotion Toolkit

Before communication can be established using VisualMotion Toolkit, the DDE Server configuration in the previous section must have been performed for the desired control.

The following steps initiate communication to an Ethernet-ready control via VisualMotion Toolkit.

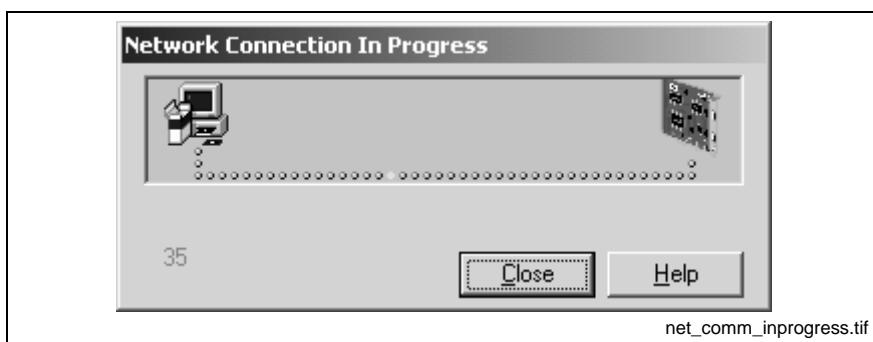
1. Start VisualMotion Toolkit and select **Settings** ⇒ **Card Selection Setup** to open the *Card Selection Setup* window below.



2. Select **Network** as the Connection Method from the drop down list.
3. Enter the control's card number and click on the **OK** button.

Note: The control's card number is determined in the order, in which they are added to the *Network Communications* window, starting with 0.

The *Network Connection In Progress* window below is displayed during communication initialization.



Once communication is established, VisualMotion Toolkit can access the control based on the access level set in control parameter C-0-0404, Card Access Network Control.

No Access Password Protection

When accessing a control with a "No Access" security level, the DDE Server will display the *Network Access Password Protected* window in Fig. 9-3. When the *Enter Password* button is pressed, VisualMotion will issue a "97 Requested operation prohibited from network" error. In order to gain access to the control, the user must enter the password assigned in control parameter C-0-0405.

Note: When connected to the control via an Ethernet connection, the password in control parameter C-0-0405 is displayed as asterisks. Only the user with serial access to the control can view the actual text password and modify it if desired.

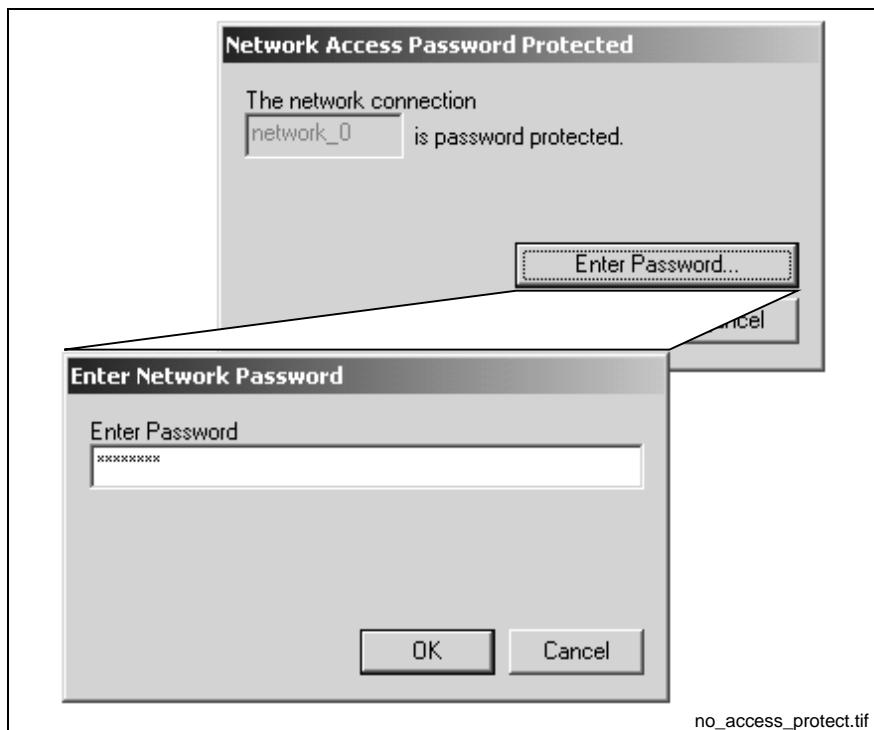


Fig. 9-3: Network Access Password Protect

9.2 Dynamic Data Exchange

The Microsoft Windows® operating system specifies a method for transferring data between applications called Dynamic Data Exchange (DDE). DDE is a message protocol that developers can use for exchanging data between Windows-based applications. The communication server uses the Dynamic Data Exchange Management library (DDEML) built on top of the DDE protocol. The DDEML provides services that the message-based DDE protocol does not support. Under the DDEML, a client application requests information from a server application, or it sends unsolicited data to the server. The client does this by passing predefined ASCII strings to the server through the DDEML.

Client and server must first establish a common conversation before data can be exchanged. A service name and a topic name define conversation. The DDE server application uses this information for establishing communication. After having established a conversation, the client application can now exchange data by specifying an item name. The item name identifies the specific data to be exchanged.

There are three basic types of data transactions that can be initiated by the client application. A **request** transaction is used to obtain data from the server. The server application knows how to obtain the requested information. The second type of transaction is an **advise link**. After a client application establishes an advise link with a server, it is up to the server to poll the data for changes. If the server finds that the data has changed, it will notify the client application. The third type of transaction is a **poke**. A poke transaction is used to send data for a specific item to the server.

The Dynamic Data Exchange Server

CLC_DDE is a Windows-based Dynamic Data Exchange (DDE) Server application that is used to communicate with Rexroth Indramat's GPP motion control. It has been implemented using Windows Dynamic Data Exchange Management Library (DDEML).

Key Features

- Serial connection to the control with support for an RS485 auto switching adapter
- Support for a modem connection (AT protocol) to a control
- Network support via an Ethernet connection
- Connection for editing a control compiled program file offline (Requires *CLC_FILE.DLL*)
- Demonstration connection for testing client applications offline (Requires *DEMO.INI*)
- Access to server parameters and status through DDE
- Supports *Request*, *Advise* and *Poke* transactions

Dynamic Data Exchange Interface

A Windows™ application, known as a *client*, can pass information between other applications known as *servers* using Dynamic Data Exchange (*DDE*). A client establishes a conversation with a server specifying a *Service* and a *Topic*. Once a conversation has been started, a client may request or send information by specifying an *item*.

Service Name

The control communication server supports two DDE service names. The standard service name is **CLC_DDE**. This should be used for all connections except when connecting to a control compiled program file. In this case, use **CLC_FILE**.

Topic Name

When the standard service name is used to exchange control data, the topic name identifies the method of connection to the control and the unit number. Valid strings consist of a communication device name and a unit number. Valid device names are **SERIAL_**, **ISA_**, **AT_MODEM_**, **DEMO_**, ***.exe**, **PC104_ or network_** and valid card unit numbers are '0' to 'F'. Connections that use the CLC_FILE service should specify the VisualMotion program file as the topic name. If the file is not located in the same directory as clc_dde.exe then the complete path should be included. To exchange server data, the service name should be **CLC_DDE** and the topic name should be **SERVER**. This is the only topic that will not support an advise link. See section SERVER Topic Name on page 9-30.

- Examples:**
- "**SERIAL_0**" Serial connection to a GPP control designated as unit '0'
 - "**ISA_1**" PC talking over the ISA bus to a CLC-P01 card designated as unit 1
 - "**PC104_0**" PC talking over the PC104 bus to a CLC-P02 card designated as unit 0
 - "**SERVER**" Exchange CLC_DDE server information

Item Name

The item name identifies the specific data to exchange. When exchanging control data, the item name consists of a string that contains the class, subclass and data identifiers of the information for the GPS/GPP controller. The strings follow the ASCII serial protocol. Refer to [Direct ASCII Communications](#) for an explanation of these codes. When exchanging server data the item name should consist of the section and entry name from the INI file (clc_dde.ini). The two names must be divided by a pipe ('|') character. Not all server data has read/write capabilities.

- Example:**
- "**RX 0.10**" Specifies register 10 in hexadecimal format
 - "**TP 2.20**" Specifies task B parameter 20
 - "**CP 1.122**" Specifies card parameter 122

9.3 The VM DDE Server

The DDE Server is first launched when a request is made to the control. The DDE Server can be setup to display the control's unit number and current status. In this mode, the DDE Server can act as a diagnostic window for the control system.

The DDE Server is displayed as an icon on the Start toolbar when the window in Fig. 9-4 is minimized.

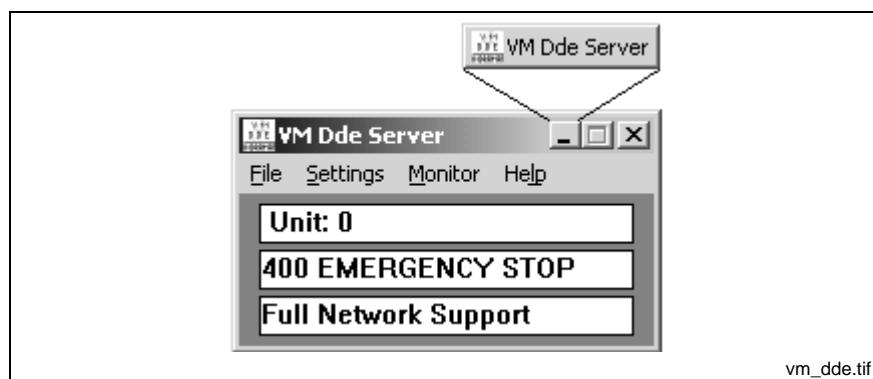


Fig. 9-4: DDE Server

All settings for the DDE Server are performed from the menu selections in Fig. 9-5.

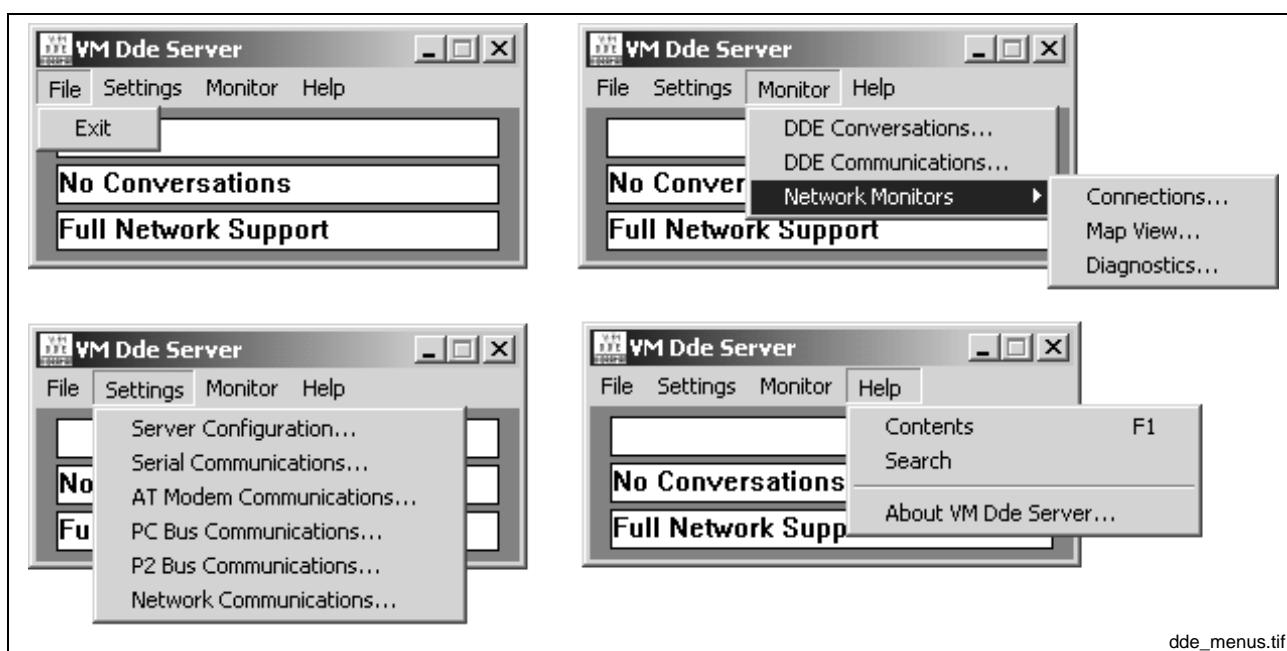


Fig. 9-5: DDE Server Menu Selections

The Settings Menu

The Settings menu is used to configure the DDE Server and communications for the types displayed in Fig. 9-6.



Fig. 9-6: The Settings Menu

Server Configuration

The Server Configuration window allows setting of various system parameters as well as providing performance status information.

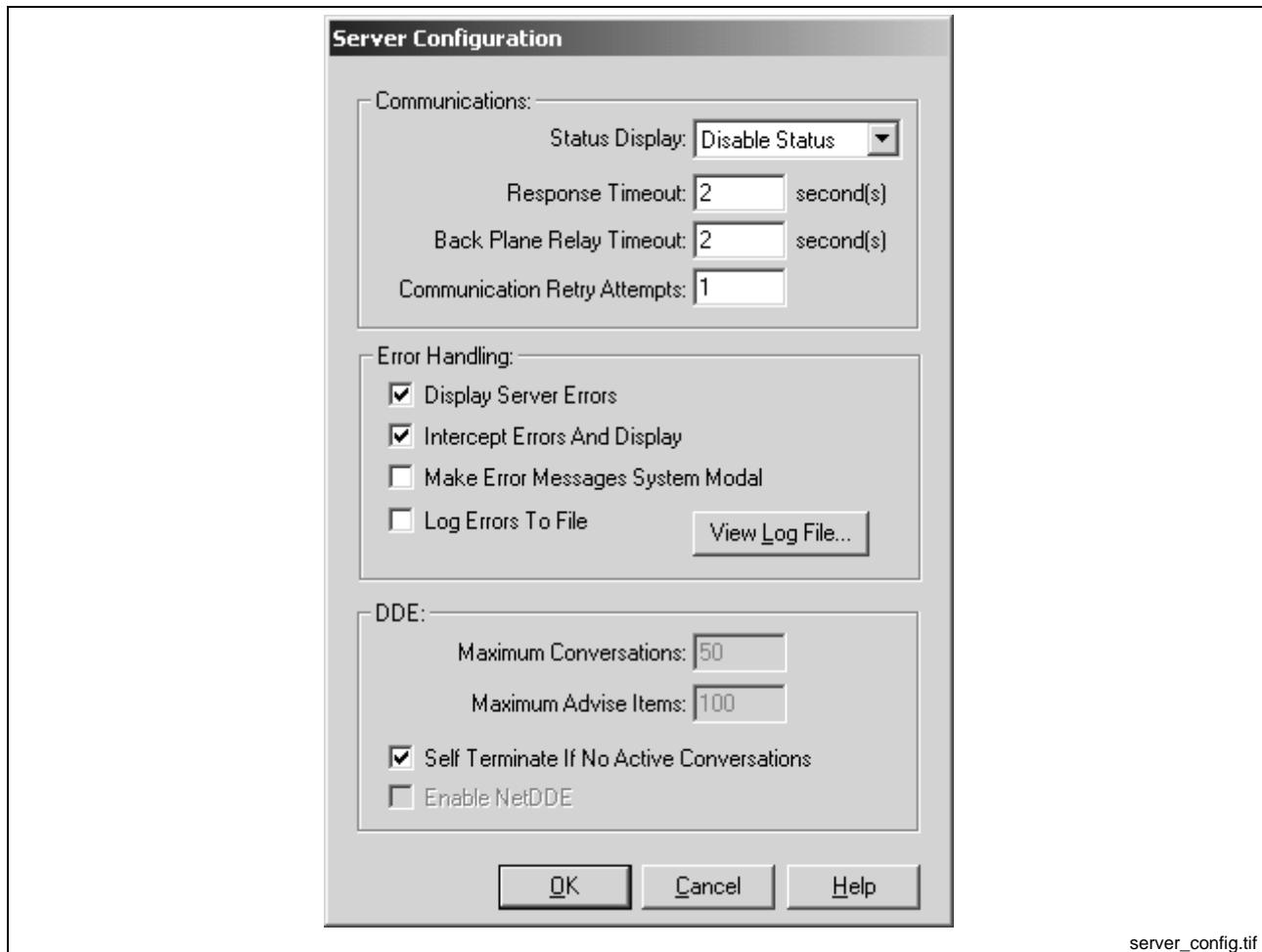


Fig. 9-7: Server Configuration

Communications

Status Display	Selects the device/unit (i.e. serial_0) combination to be displayed in the status window of the server. The request will be inserted into the standard client advise loop queue. Selecting "Disable Status" can turn off this feature. Disable Status displays " <i>Status Display is Disabled</i> " Serial_0 displays current status of system; for example, " <i>007 Program Running: A</i> "
Response Time-out	The amount of time in seconds that the server will wait for a completed response from the control before diagnosing a disconnect. The valid range of values is 1-900 seconds.
Back Plane Relay Time-out	This feature is not supported in GPP 8.
Communication Retry Attempts	The number of times the server will re-send a message before it issues an error. The valid range of values is 0-255 seconds.

Error Handling

Displays Server Errors	Checking this box will cause the server to display error responses in a message box.
Intercept Errors And Display	Checking this box will cause the server to intercept control error responses and displayed them in a message box. Request and poke transactions will return a failure to the client application. Advise links will remain active; however, they will return nothing until the error is resolved. The error response will be written to the error log file if that feature is enabled. If this box is not checked the error string will be returned to the client.
Make Error Messages System Modal	Checking this box will cause all server generated message boxes to have system modal attributes. This means that all applications will be suspended until the user responds to the message box. The window can not be forced to the background.
Log Errors To File	Checking this box will cause the server to log all server errors to a file. The current system date and time will be associated with each log entry. As a default this feature is not enabled.
<input type="button" value="View Log File..."/>	Pressing this button will cause the current error log file to be displayed in notepad.

DDE

Maximum Conversations	This is a static display of the maximum number of allowed DDE conversations as specified in the INI file. The server will refuse any DDE connection request in excess of this value.
Maximum Advise Items	This is a static display of the maximum number of allowed DDE advise links as specified in the INI file. The server will refuse any request for advise links in excess of this value.
Self Terminate If No Active Conversations	Checking this box will cause the server to close when the last DDE conversation has terminated. This is the default state.

Serial Communications

The Serial Communications window allows the user to select the serial communication parameters the server will use. When this window is open, all communications are suspended. If changes are made to the configuration, they will take affect when the Apply or OK button is pressed.

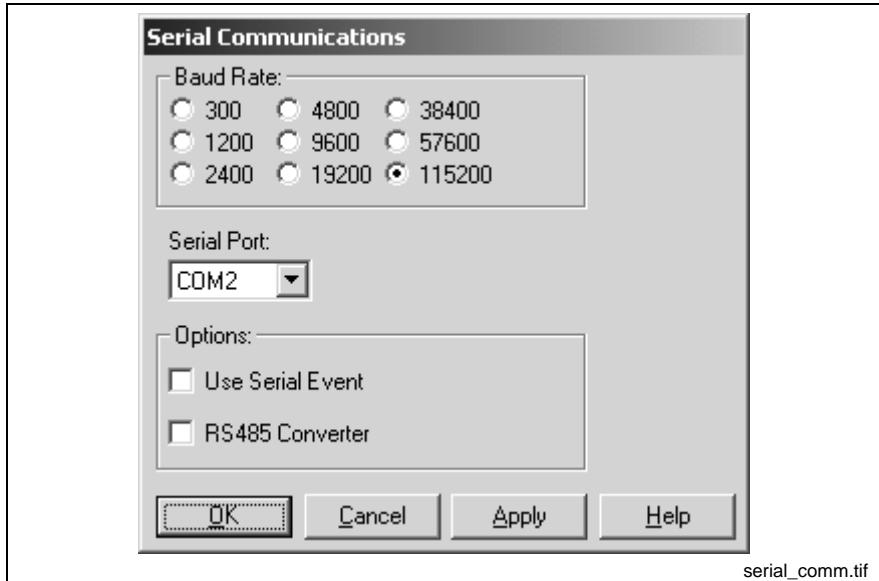


Fig. 9-8: Serial Communications

Baud Rate	Set the baud rate to match the control's port as displayed in VisualMotion under menu selection Settings ⇒ Control Serial Ports .
Serial Port	Select the serial communications COM port to use on the PC.
Use Serial Event	Checking this box causes Windows to notify the server when a completed message is in the receive queue. This will increase the number of serial messages sent over polling for a response. Slower computers may not be able to utilize this feature.
RS485 Converter	This option should be used when an RS232 to RS485 converter is present. A delay will be inserted between messages, which is equal to at least one character transmission at the selected baud rate. This is necessary to ensure that the control has had sufficient time in which to turn the RS485 transmitter off and enable the receiver. Please note that the converter must toggle the transmitter and receiver automatically, and disable echo back.

AT Modem Communications

The AT Modem Communications selection initially launches a Windows wizard to configure the modem to its current world location. Next, the Modem Configuration window opens. Refer to Modem Communication using TAPI Interface on page 9-26.

PC Bus Communications

Note: GPP 8 does not support the PC Bus Communication selection from the Settings menu. However, VisualMotion Toolkit 8 supports downward compatibility for CLC hardware using GPS firmware version 6 or greater.

The PC Communications window allows the user to view control status indicators and set communication parameters. When this window is open, all communications are suspended. If changes are made to the configuration, they will take affect when the "Save" button is pressed. The dynamic link library "CLC_P.DLL" must be in the control directory or the Windows path.

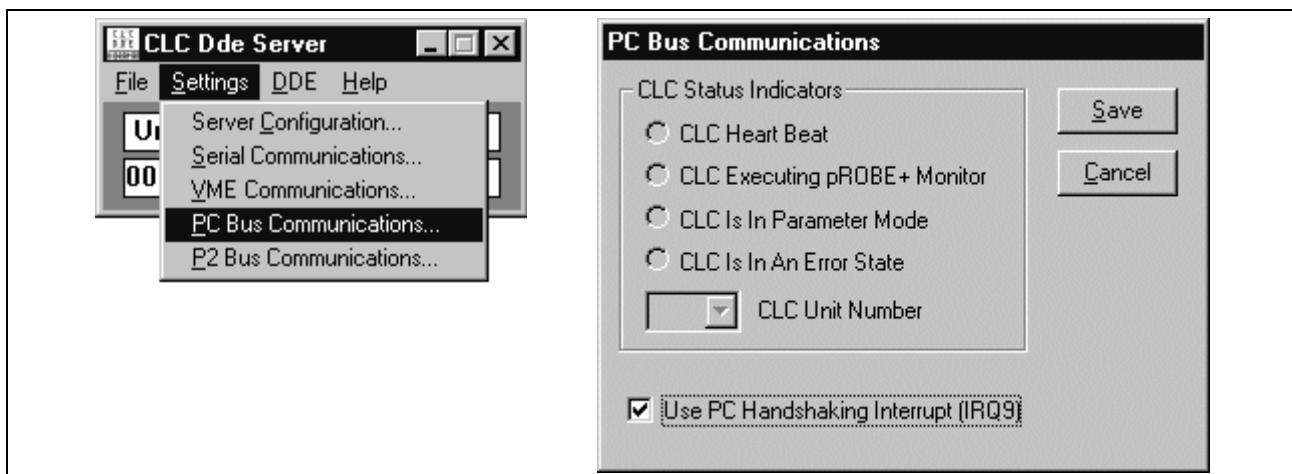


Fig. 9-9: PC Bus Communications

Status Indicators

Heart Beat	This indicator will blink indicating that the selected control card is running.
Executing pROBE+Monitor	This indicator will be marked if the selected control card has faulted and is running the pROBE+ monitor.
Is In Parameter Mode	This indicator will be marked when the selected control card is in parameter mode.
Is In An Error State	This indicator will be marked when the selected control card is in an error state. Card parameter 122 will contain the specific error message.
Unit Number	<p>This option selects the card number. This card number must match the settings of hardware switches S8 through S11. To locate a card without knowing the switch settings, use the following procedure:</p> <ul style="list-style-type: none"> ⇒ Start with card number 0. ⇒ If sounds are active on your PC and 0 is not the correct card number, a sound will be heard. ⇒ Continue this procedure until a sound is not heard. This is an indication that the correct card number has been selected. <p>Once this is established, use the same card number under Settings ⇒ Card Selection Setup in VisualMotion Toolkit. Connection method must also be set to PC ISA Bus.</p>
Use PC Handshaking Interrupt (IRQ 9)	<p>When selected, this option will force all CLC-P control cards to terminate communication responses with a PC interrupt (IRQ 9). Hardware jumper S5 must be inserted on the CLC-P card for this option to work properly. If this option is not used, the server will poll for a communication response every 55 milliseconds.</p> <p>Note: When using the interrupt option on the CLC-P control card, no other hardware devices may use IRQ 9.</p>

P2 Bus Communications

Note: GPP 8 does not support the P2 Bus Communication selection from the Settings menu. However, VisualMotion Toolkit 8 supports downward compatibility to GPS firmware version 6 or greater.

The P2 Communications window allows the user to view control status indicators and set communication parameters. When this window is open, all communications are suspended. If changes are made to the configuration, they will take affect when the "OK" button is pressed.

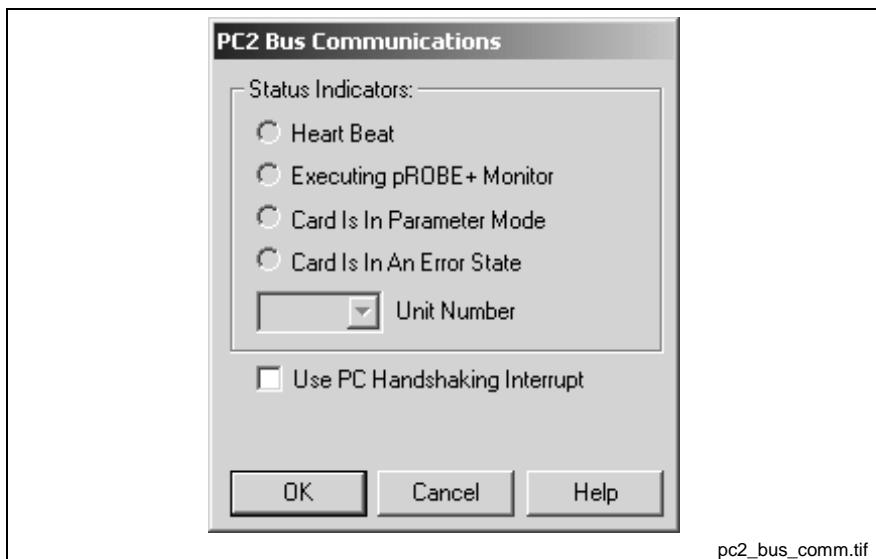


Fig. 9-10: PC2 Bus Communication

Status Indicators

Heart Beat	This indicator will blink indicating that the selected control card is running.
Executing pROBE+Monitor	This indicator will be marked if the selected control card has faulted and is running the pROBE+ monitor.
Card Is In Parameter Mode	This indicator will be marked when the selected control card is in parameter mode.
Card Is In An Error State	This indicator will be marked when the selected control card is in an error state. Card parameter 122 will contain the specific error message.
Unit Number	<p>This option selects the card number. This card number must match the settings of hardware S1 DIP switches 1 through 4. To locate a card without knowing the DIP switch settings use the following procedure:</p> <ul style="list-style-type: none"> ⇒ Start with card number 0. ⇒ If sounds are active on your PC and 0 is not the correct card number, a sound will be heard. ⇒ Continue this procedure until a sound is not heard. This is an indication that the correct card number has been selected. <p>Once this is established, use the same card number under Settings ⇒ Card Selection Setup in VisualMotion Toolkit. Connection method must also be set to PC-104 Bus.</p>
Use PC Handshaking Interrupt	<p>When selected, this option will force all CLC-P02 control cards to establish communication responses with a PC interrupt selected. Hardware S1 DIP switches 5 through 8 settings must match the interrupt selected for this option to work properly. If this option is not used, the server will poll for a communication response every 55 milliseconds. When selected, this option will increase the handshake response time.</p> <p>Note: The default interrupt setting is INT10.</p>

Network Communications

The Network Communications window is used to configure the IP Addresses of Ethernet-ready controls. An Ethernet-ready control is a PPC-R motion control unit ordered preconfigured with an optional Ethernet card. After the pertinent Ethernet control parameters are configured, the user can add the card's IP Address to the DDE Server. Selecting **Settings** ⇒ **Network Communications** from the DDE Server's main window opens the window in Fig. 9-11.

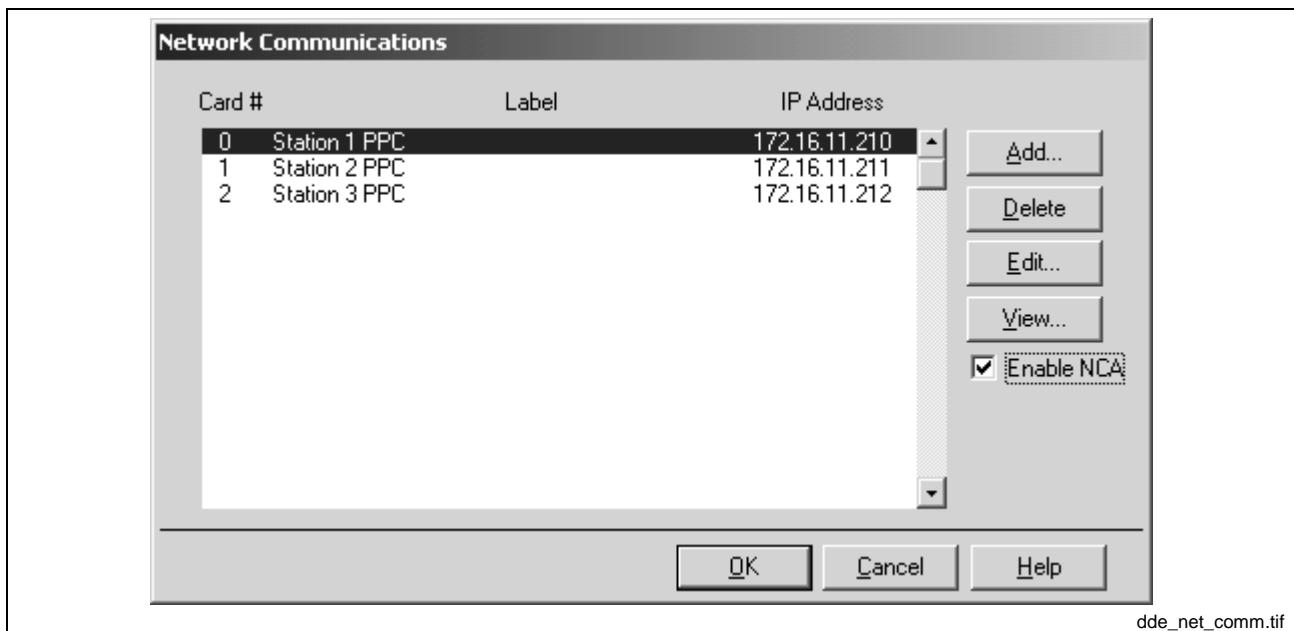


Fig. 9-11: Network Communications

Note: A maximum of 10 network connections are allowed in the *Network Communication* window. A DDE Server message will be issued if the **Add** button is pressed for an 11 connection.

When opened, the *Network Communications* window will display a list of existing configurations. The *Card #* is assigned by the DDE Server and is used to associate a communication path between VisualMotion Toolkit and the motion control. The *Label* identifies the motion control with a text string. The *IP Address* is displayed in dot notation and is unique to the Ethernet card of the desired control (as entered in card parameter C-0-0400). The following list describes the available buttons:

- **Add:** Adds a network configuration to the list
- **Delete:** Removes the selected configuration from the list
- **Edit:** Modifies the selected configuration in the list
- **View:** Displays detailed information of the selected configuration from the list

The **Enable NCA** checkbox is used to enable the **Network Communication Accelerator**. This feature accelerates communication between the control and VisualMotion Toolkit by creating an open message loop.

Add Network Configuration

When the **Add...** button in the *Network Communication* window is pressed, the Add Network Configuration window in Fig. 9-12 opens. The user populates the fields in this window with specific information that will configure a communication path to the desired control via the DDE Server.

Note: The first configuration that is entered will be assigned a 0 as the *Card #* in the network. Consecutive card numbers will be assigned for additional controls added to the network configuration.

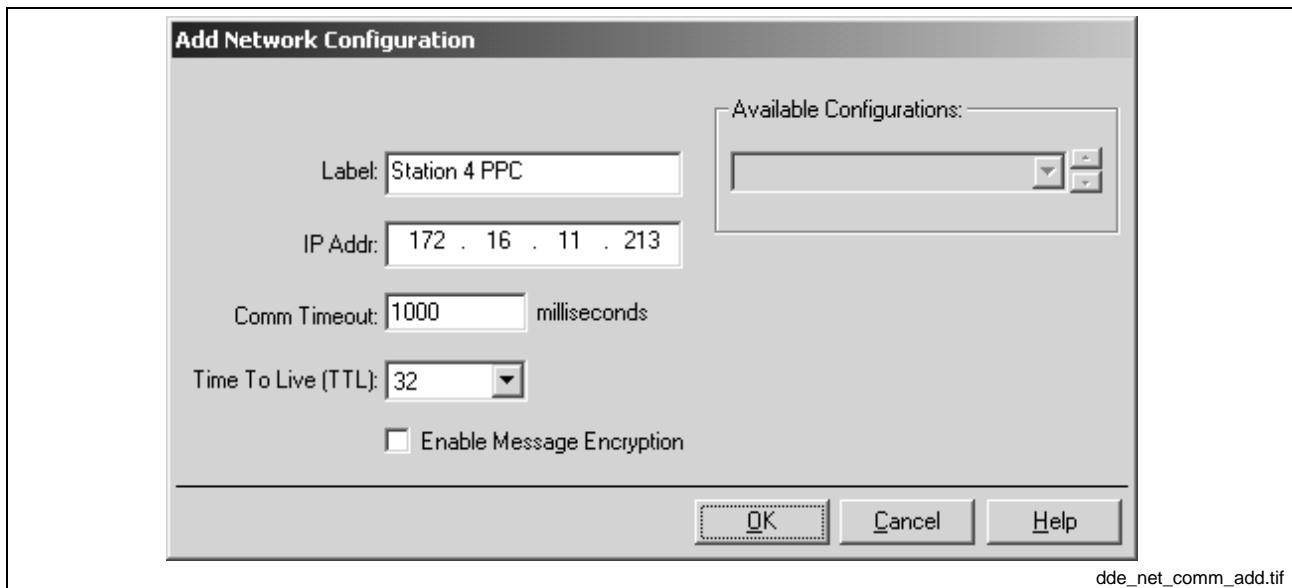


Fig. 9-12: Add Network Configuration

Label:

The user enters a text string (up to 20 characters) that identifies the control.

IP Address:

The user enters a unique IP Address (provided by the IT department), in dot notation, that was used to configure control parameter C-0-0400, Card IP Address.

Comm Timeout:

The communication timeout field displays a time in milliseconds that represents the amount of time that the network will wait for a read or write operation. The default is 1000 ms.

Note: The Communication Timeout value should not exceed to the DDE Server *Response Timeout* in the Server Configuration window of Fig. 9-7.

Time to Live (TTL):

This field displays the maximum number of hops (router connections) that the control will allow. If the number of actual linking connection hops exceeds the number in the field, the network connection will not occur. Clicking on the drop down list and selecting 32, 64 or 100 can change this field. A default value of 32 is sufficient.

Note: This feature is not available for Windows 98/95.

Enable Message Encryption

This feature encrypts messages before transporting them to the control. This provides another layer of security.

Note: The *Available Configurations* field is only accessible when the Edit or View buttons in the *Network Communications* window are used.

Delete a Network Configuration

A configuration is deleted by selecting the desired item in the list and clicking on the Delete button. Next, a VisualMotion DDE Server Message will warn the user before deleting the selected entry, as illustrated in Fig. 9-13.

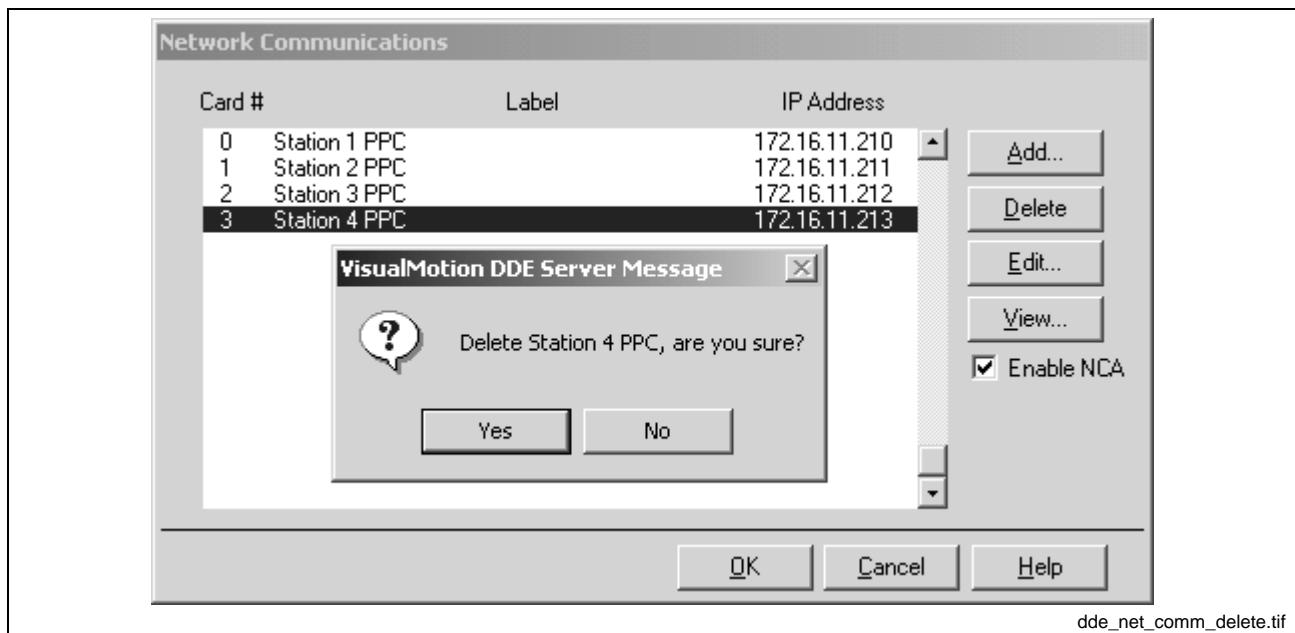


Fig. 9-13: Deleting a Configuration Entry

Edit a Network Configuration

Selecting the desired item from the Network Communication window listing and clicking on the **Edit...** button modifies a configuration entry. This function can also be performed by double clicking in the item in the *Network Communication* window. The user can select a different configuration from the *Available Configurations* drop down list. After the necessary modifications are made, click on the **OK** button to save the changes and return to the *Network Communication* window.

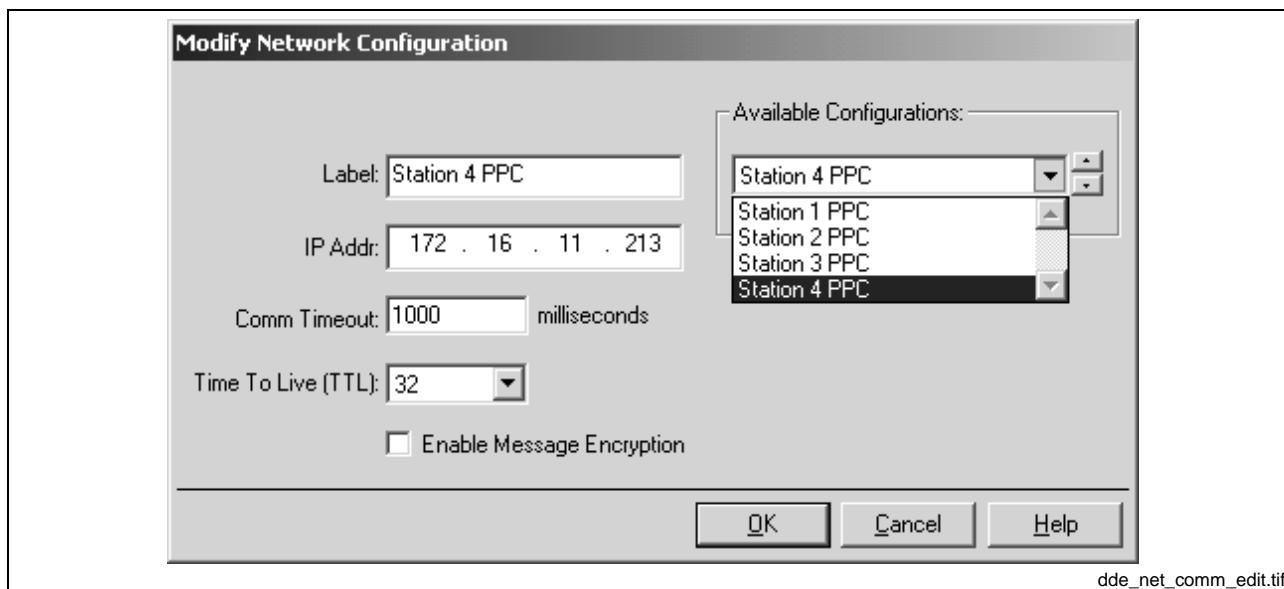


Fig. 9-14: Modify Network Configuration

View Network Configuration

A configuration entry is viewed by selecting the desired item from the *Network Communication* window listing and clicking on the **View...** button. The user can view a different configuration from the *Available Configurations* drop down list. The configuration fields are grayed out in this view and can not be modified.

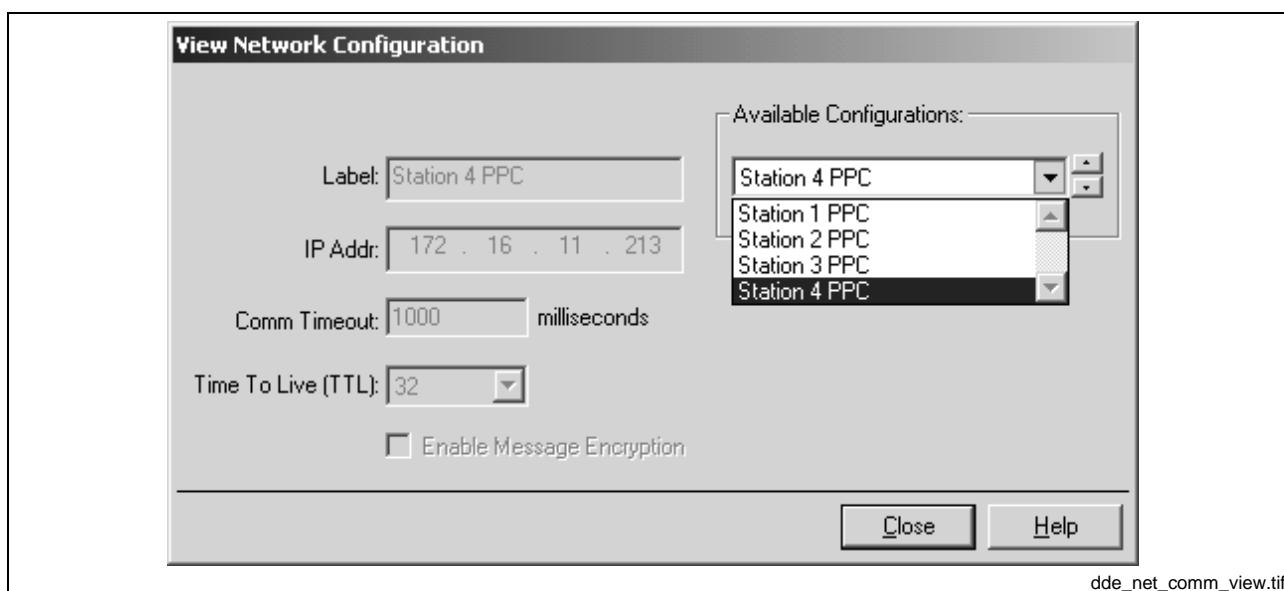


Fig. 9-15: View Network Configuration

The Monitor Menu

The Monitor menu is used to monitor DDE conversation, communication and diagnose network communications.

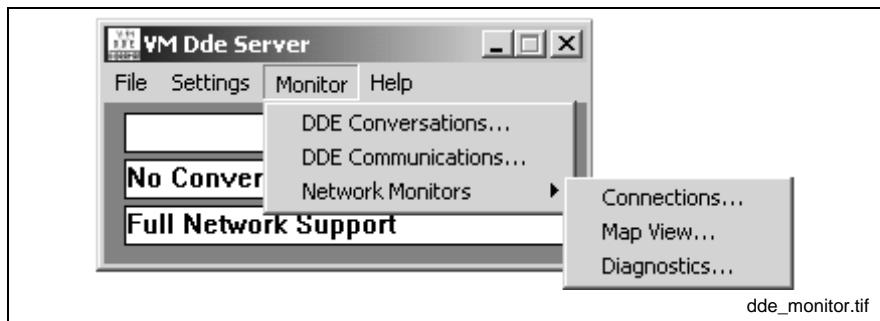


Fig. 9-16: The Monitor Menu

DDE Conversations

The DDE Conversations window displays the **Conversation**, **Service and Topic Handles** for all of the current DDE conversations. The **Item Count** column shows the total number of active advise links, request transactions and poke transactions. Double click on a specific conversation entry in order to view the item transaction list. A second method is to select the conversation and then use the **Properties** button. This window is useful when creating client applications that talk to the control communications server.

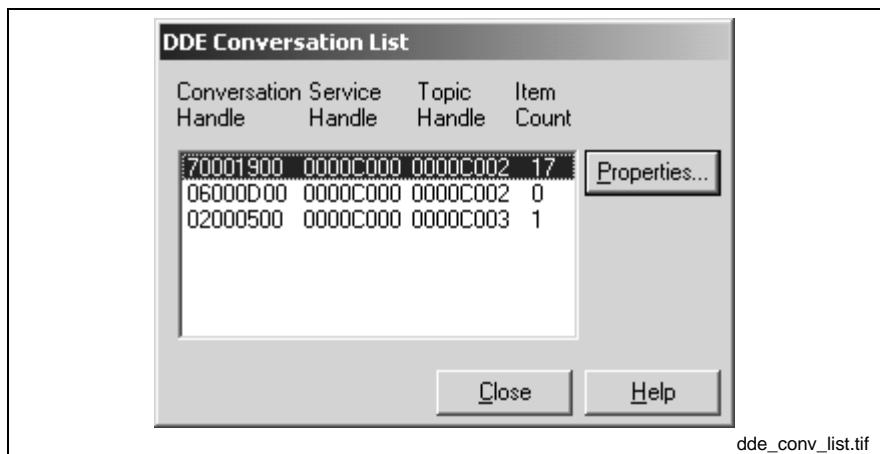


Fig. 9-17: DDE Conversations

DDE Conversation Item

The DDE Conversation Item window can be used to view the item transaction list for a conversation. The Service name, Topic string, Item, Format and Transaction Type are displayed in text format. Use the “Next” and “Previous” buttons to cycle through the Item field. This feature only cycles through when more than one item is in the conversation.

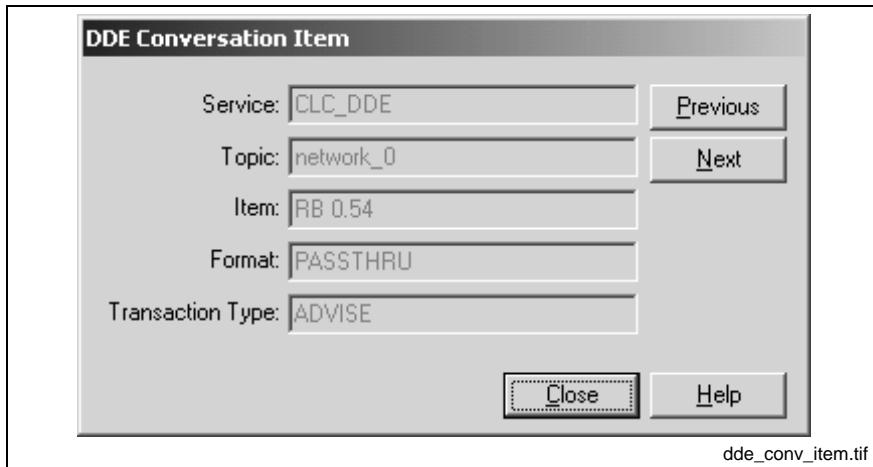


Fig. 9-18: DDE Conversation Item

DDE Communications

The DDE Communication Monitor displays all of the current DDE conversations. The monitor can display DDE requests and/ or responses depending on the selection made under the *Settings* menu.

The active window builds a communications log of all DDE conversations that occur while the monitor is running. Selecting *Clear* will empty the log. Selecting *Stop* will stop the conversation monitoring and allow users to scroll through the log. The Monitor window can be resized to enlarge the active viewing area.

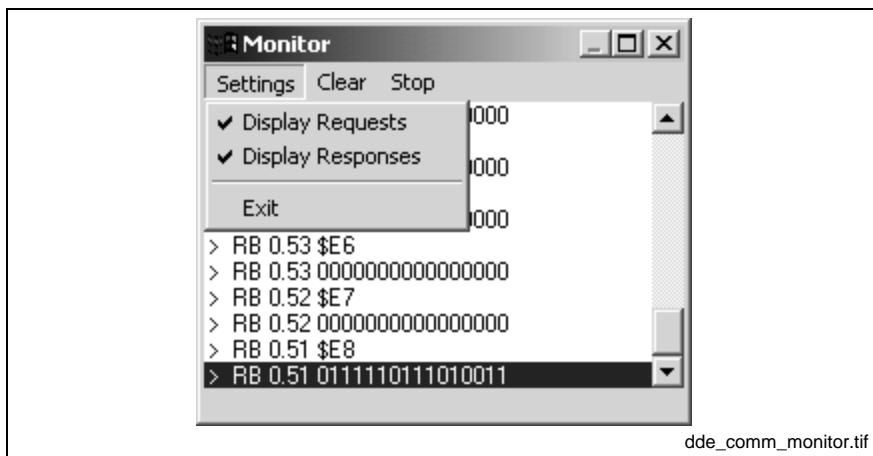


Fig. 9-19: DDE Communication Monitor

Network Monitors

Network monitors are used to monitor data transfer of connected configurations, determine accessible of IP Addresses on the network and test network communication of a specific IP Address.



Fig. 9-20: Network Monitors

Connections

Selecting **Monitor** \Rightarrow **Network Monitors** \Rightarrow **Connections** for the VM DDE Server opens the Connections window in Fig. 9-21. From this window, the user can monitor data transfer statistics, performance statistics and error counts.

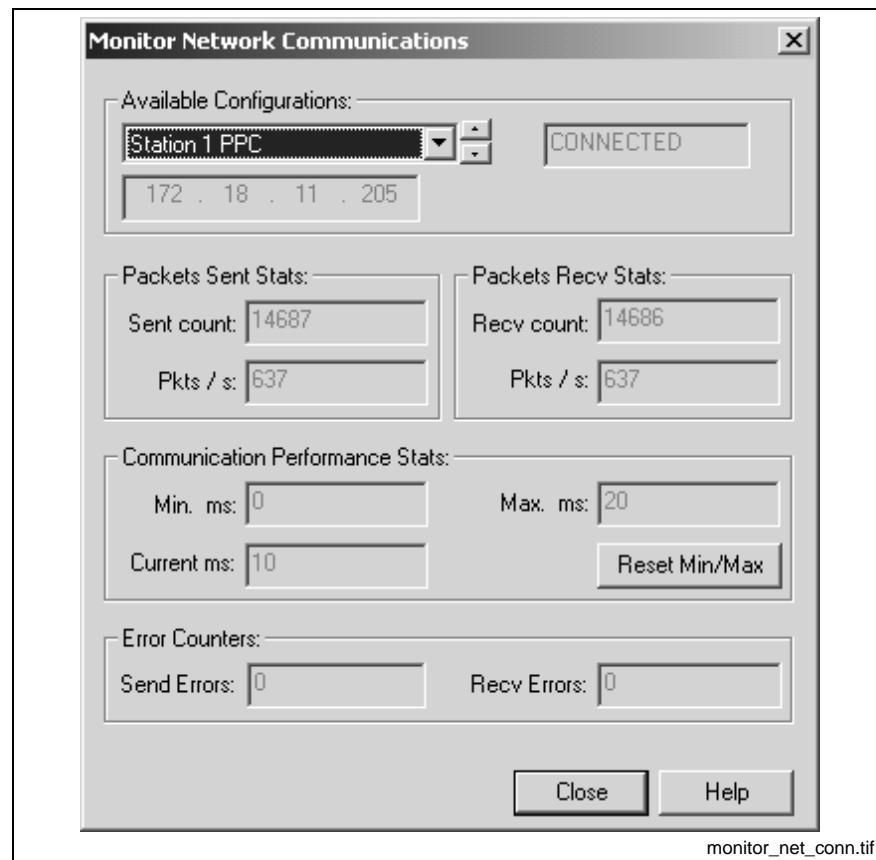


Fig. 9-21: Monitor Network Communications

Map View

Selecting **Monitor** ⇒ **Network Monitors** ⇒ **Map View** from the VM DDE Server main menu opens the *Control Network Map* window in Fig. 9-22. This window is used to locate active controls within a network Subnet IP Address.

Note: The Map View feature is not available for Windows 98/95

Entering a valid Subnet IP Address (provided by the IT department) and clicking on the **Start** button will find all active Ethernet-ready control and display them as a green **G**. The *Timeout* field represents the maximum time needed to determine whether a connection has been established. If all known connections are displayed, click the **Stop** button to terminate the search. The yellow **B** represents the broadcast address for the given Subnet.

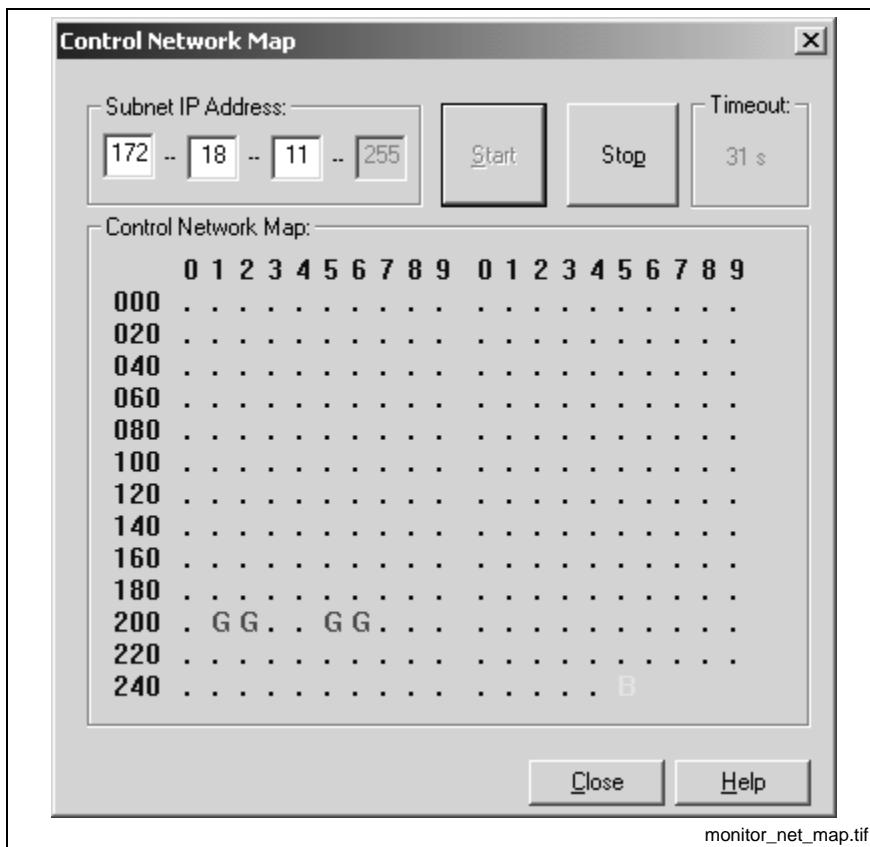


Fig. 9-22: Control Network Map

- Card Query** Double clicking on a green **G**, opens the *Control Card Query* window in Fig. 9-23. From this window, the user can view the control's system configuration as well as the control's network information.

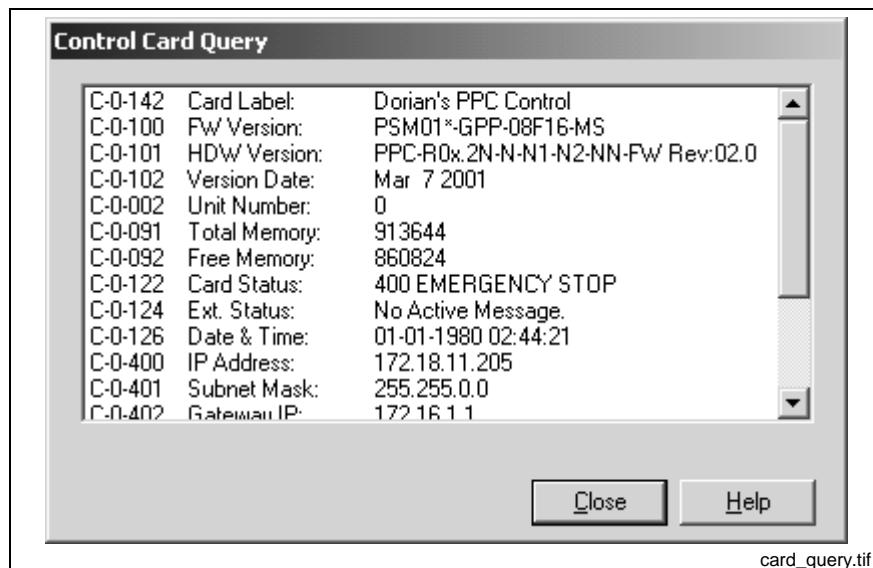


Fig. 9-23: Control Card Query

- Add Configuration** Right clicking on a green **G** opens a small popup window, where the user can add the selected control to the *Network Communication* section on page 9-16. Selecting **Add Configuration** opens the *Modify Network Configuration* window in Fig. 9-24. From this window, the user can modify the *Label* that will appear in the *Network Communication* window located under the VM DDE Server's menu selection, **Settings** ⇒ **Network Communications**.

Note: This feature is equivalent to selecting the **Add** button on the *Network Communication* window when adding a new configuration to the VM DDE Server.

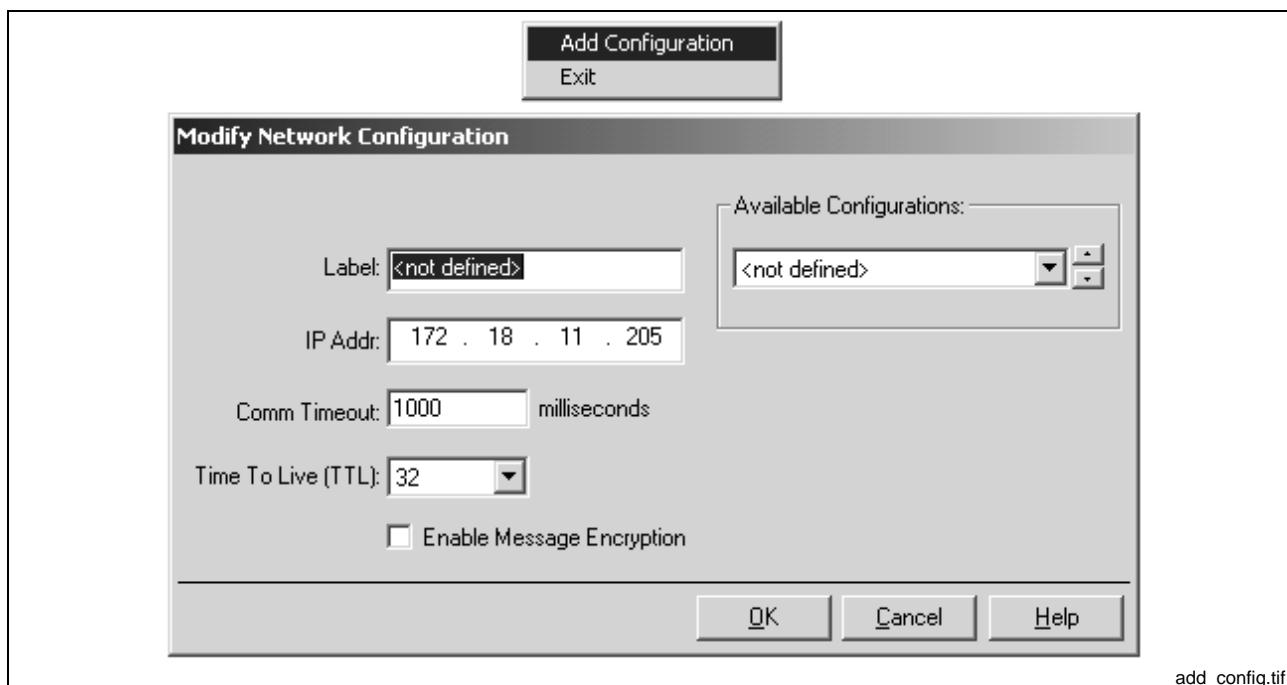


Fig. 9-24: Add Configuration

Diagnostics

Selecting **Monitor** ⇒ **Network Monitors** ⇒ **Diagnostics** from the VM DDE Server main menu opens the *Connection Diagnostics* window in Fig. 9-25. From this window the user can test the network communication for a control listed in the *Available Configurations* drop down list. Each test returns a message indicating if the test passed or failed. In addition to network testing, the user can also trace the route of the network connection for the selected control.

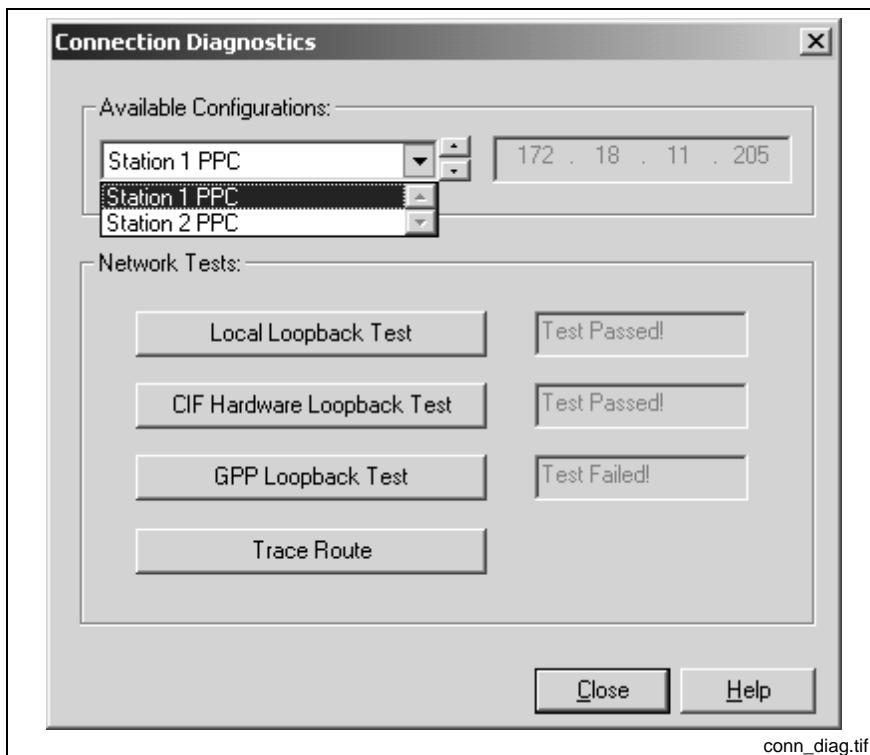


Fig. 9-25: Connection Diagnostics

Local Loopback Test

This test verifies that the PC's network interface is operational.

Note: This test is not available for Windows 98/95.

CIF Hardware Loopback Test

This test verifies that the DDE Server can communicate with the Ethernet card in the control.

Note: This test is not available for Windows 98/95.

GPP Loopback Test

This test verifies that the DDE Server can communicate with the GPP firmware in the control via the control's Ethernet card.

Trace Route

This is not a test. This button opens the *Trace Route* window, which maps the route of the message through routers from the DDE server to the control.

Note: This test is not available for Windows 98/95.

9.4 Modem Communication using TAPI Interface

VisualMotion 8 and the DDE Server can communicate with a control installed in a remote location using a Windows® standard **Telephony Application Programming Interface (TAPI)**. Any modem supporting auto-answer, no flow control, and baud rates from 9600 to 115200 can be used for communications.

Before VisualMotion can communicate with a remote GPP control connected to a modem, the TAPI interface must be setup in VisualMotion Toolkit and the DDE Server. Select **Settings** ⇒ **Card Selection Setup** from VisualMotion Toolkit's main menu and specifying the connection method as *AT Modem* from the Card Selection Setup window in Fig. 9-26.

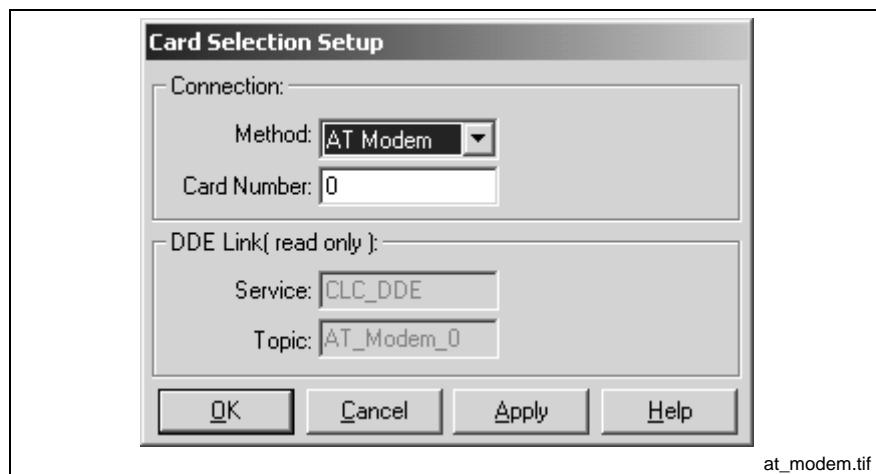


Fig. 9-26: Card Selection Setup

Technical Requirements

The remote modem communicates to the control through a serial connection and a null modem adapter. The remote modem dictates the baud rate of the connection. Thus, when the remote modem is set to 9600 baud, the local modem will assume that rate of transmission. The remote modem also must be configured to auto-answer calls with no flow control. The user must determine how to set up the specific remote modem with these parameters. An example setup that utilizes Hayes Accura 336/56K modems and an Indramat IKB0005 serial cable is shown in Fig. 9-27.

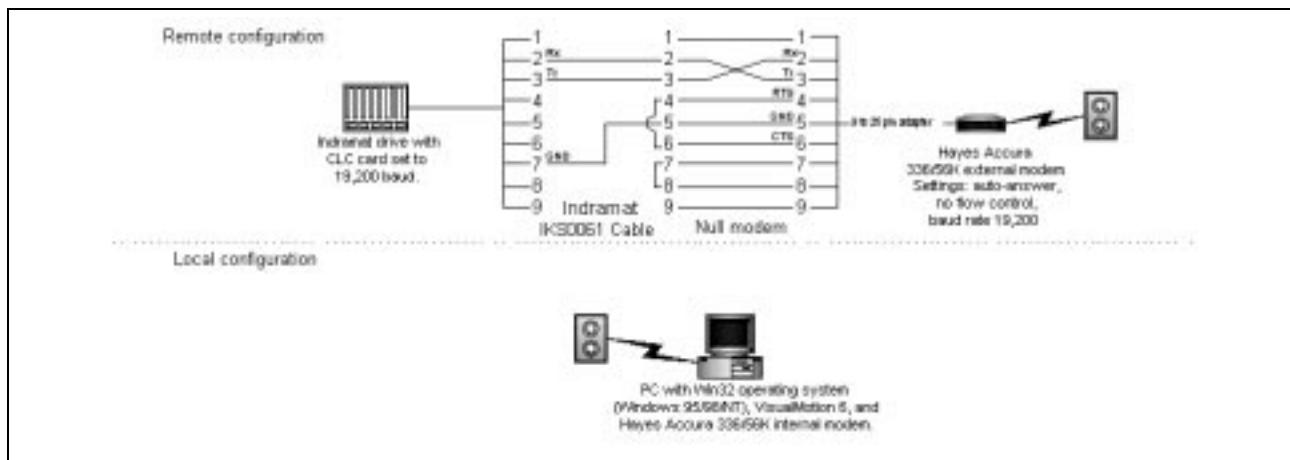


Fig. 9-27: Example Modem Setup

Modem Configuration

Once the AT Modem connection method is setup and saved in VisualMotion Toolkit, the first attempt at communicating with the control, such as **Diagnostics** ⇒ **System**, will open the *Modem Configuration* window in Fig. 9-28. If a configured modem is detected, the *TAPI Line* field will display the modem. For a complete listing of modem errors, refer to Table 9-2 and on page 9-29.

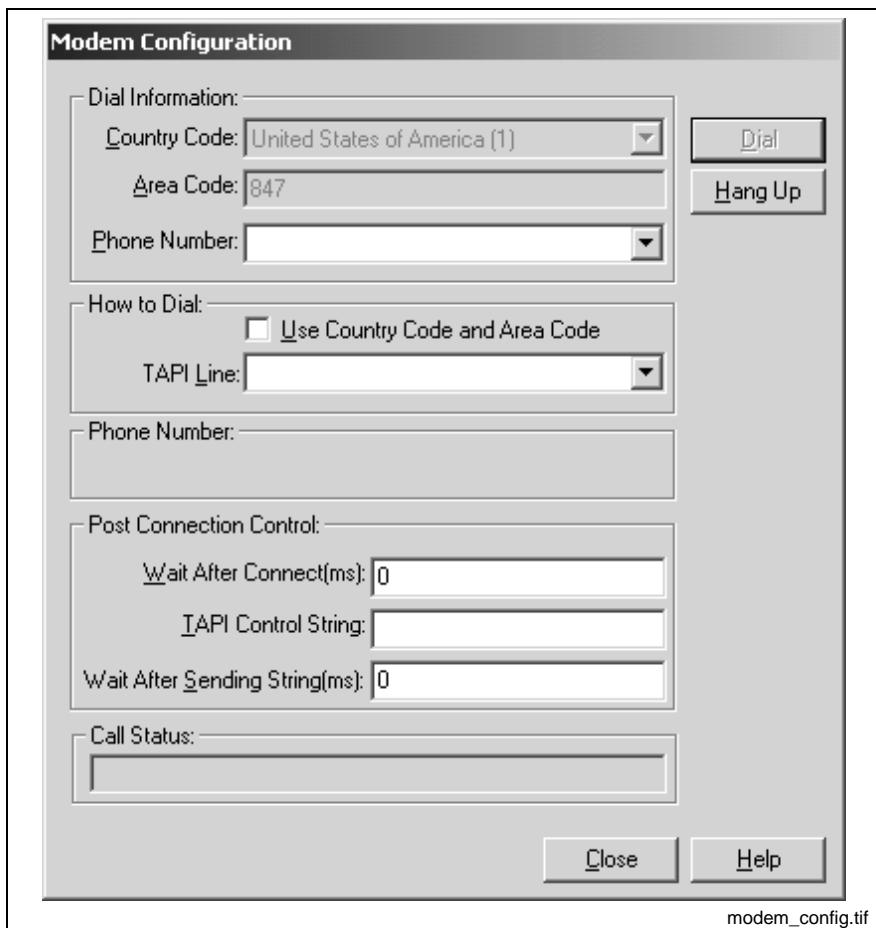


Fig. 9-28: Modem Configuration

The fields in the Modem Configuration window must be setup before actual communications can be established with the remote modem. Select the country and area code of the remote location where the control is setup. Enter the telephone number and *How to Dial* options and click on the **Dial** button. Optional Post Connection Control fields are available for entering a timeout value (up to 10,000 ms or 10 seconds) and a control string to be written to the modem once the connection is established. The format of acceptable input control strings for the TAPI Control String fields are listed in Table 9-1. When the modem establishes communications, the Modem Configuration window disappears and VisualMotion 8 is now linked with the remote control.

TAPI Control String

The TAPI Control String field allows the user to send commands directly to the modem after a connection has been established. The string can be a mixture of control and ASCII characters. To enter control characters, use the decimal values listed in Table 9-1 surrounded by <> brackets. For example, to send <CTRL Q> followed by a 2, the user would enter <17>2 in the control string field.

ASCII Command	Decimal Equivalent
<CTRL A>	1
<CTRL B>	2
<CTRL C>	3
<CTRL D>	4
<CTRL E>	5
<CTRL F>	6
<CTRL G>	7
<CTRL H>	8
<CTRL I>	9
<CTRL J>	10
<CTRL K>	11
<CTRL L>	12
<CTRL M>	13
<CTRL N>	14
<CTRL O>	15
<CTRL P>	16
<CTRL Q>	17
<CTRL R>	18
<CTRL S>	19
<CTRL T>	20
<CTRL U>	21
<CTRL V>	22
<CTRL W>	23
<CTRL X>	24
<CTRL Y>	25
<CTRL Z>	26
<ESC>	27

Table 9-1: TAPI Control String ASCII Codes

VisualMotion Modem Error Messages

If modem connections cannot be established, VisualMotion will issue various error messages. The following table contains a list of such messages.

Error	Solution
TAPI does not support this line version.	User's version of operating system does not support TAPI 1.4.
TAPI line error.	Select a line that can be used for telephony.
TAPI line does not support data modem.	Select another line that supports data modem use.
TAPI line cannot make outbound calls.	Select a line that supports outbound dialing.
TAPI line is already in use.	Select a new line or disconnect the current connection.
TAPI line does not support partial dialing.	Select a line that supports partial dialing.
TAPI line unable to translate phone number.	Re-enter a valid phone number.
TAPI line does not support voice call.	Select a line that supports voice calling.
TAPI line does not support Comm/Data modem class.	Select a line that supports communication and data modem.
TAPI line is not usable.	Selected line is in use or does not support data modem use. Select a line that does not have these restrictions.

Table 9-2: VisualMotion Modem Error Messages

9.5 SERVER Topic Name

The “SERVER” topic name allows a DDE client application access to CLC_DDE’s parameter set and status. The server will accept request and poke transactions. When accessing a parameter the client application should specify the section and entry names from the INI file. The two names must be separated by a pipe character (‘|’). When requesting status information the client should use “STATUS” as the section name (i.e. “STATUS|ErrorState”). RW = Read/Write RO = Read Only

Section: GENERAL	Response_Timeout	R W	1-900 Seconds	Message response time out.
	Relay_Timeout	R W	1-900 Seconds	Message time out when using VME pass-through.
	Communication_Retry	R W	0-255	Number of times to re-send a message.
	Suspend_Polling	RO	0 or 1	If 1 CLC_DDE status polling will be disabled.
	Display_control_Errors	R W	0 or 1	If 1 CLC_DDE will intercept & display control Errors.
	Log_Errors	R W	0 or 1	If 1 all server errors will be logged to the error file.
	Modal_Errors	R W	0 or 1	Displayed errors with the system modal attribute.
	Self_Terminate	R W	0 or 1	Close CLC_DDE when last conversation terminates.
	Monitor_List_Size	R W	1-500	# of entries in communication monitor window.
	Editor	R W	256 Characters	Name & path of text editor to use to display error log.
Section: SERIAL	Baud rate	RO	115200..300	Baud rate for serial connection to control card.
	Port	RO	1-4	COM port number to use for serial connection.
	Serial_Event	RW	0 or 1	Use serial event option to increase performance.
	RS485_Converter	RW	0 or 1	Activate RS485 adapter code.
Section: VME	Sustain_Bus	RW	0 or 1	Release every cycle option for XYCOM PC.
	A32_Addressing	RW	0 or 1	Use A32 addressing for XYCOM PC.
	VME_IRQ	RO	0-7	Number of VME IRQ to use (0 = disabled).

Section: AT_MODEM	Baud rate	RO	56000..300	Baud rate to use to communicate with the modem.
	Port	RO	1-4	COM port number the modem is on.
	Auto_Connect	RW	0 or 1	Initialize & connect on conversation connection.
	Phone	RW	50 Characters	Phone number to dial.
	Initialize_Script	RW	100 Characters	Script to initialize modem.
	Disconnect_Script	RW	100 Characters	Script to disconnect modem.
	Dial_Prefix	RW	50 Characters	Script to send to modem before telephone number.
	Escape_Sequence	RW	50 Characters	Script to send modem to return to command mode.
Section: PC	PC_IRQ	RO	0 or 1	if 1 use PC interrupt for communications.
Section: P2	PC_IRQ	RO	0 or 1	if 1 use PC interrupt for communications.
Section: DDE	Status	RO	200 Characters	CLC_DDE's status request item.
	Max_Conversations	RO	1-3274	Maximum allowed conversations.
	Max_Advise_Items	RO	1-3500	Maximum allowed advise items.
Section: STATUS	ErrorState	RO	0 or 1	If 1 CLC_DDE is issuing an error.
	ErrorText	RO	256 Characters	Error text message CLC_DDE is displaying.
	RequestState	RO	0 or 1	If 1 CLC_DDE is actively communicating.

9.6 DDE Client Interfaces

The following examples illustrate how to create custom DDE client interfaces for the Control. DDE communication allows certain software applications to read from (*request*), and write to (*poke*) the Control card.

Creating and Customizing a DDE Client Interface in Microsoft® Excel

The following example illustrates how to create a custom DDE client interface for the Control card using Microsoft® Excel (*Version 5.0 and up*). Other programs that support DDE communication can also be used in a similar fashion. Requested information can be read directly by a spreadsheet, chart or database, while poke transactions allow users to control program execution from within this custom interface.

DDE Worksheet Functions

A DDE request can be made directly from a *cell* within an Excel™ Worksheet using a formula outlined in Fig. 9-29. The VM DDE Server should be running before a request is made. Each request is queued in the server and then handled using round robin arbitration. The Excel Worksheet will automatically update the cell, as the information becomes available. The response time varies according to how many other applications are running and how many DDE conversations are occurring at the same time. Limit the number of active DDE requests within a worksheet in order to get a faster response time.

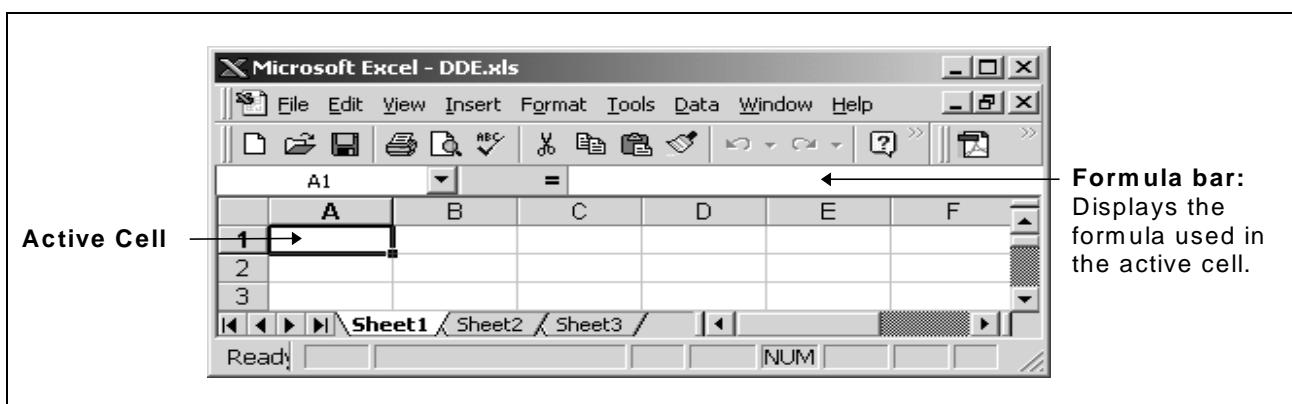


Fig. 9-29: Excel Worksheet

Select a cell and then enter the **DDE Service name**, **Topic name** and **Item name** within the formula bar using the following syntax as an example:

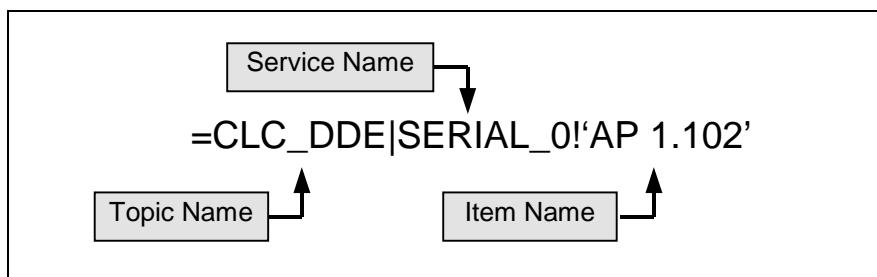


Fig. 9-30: Syntax Example

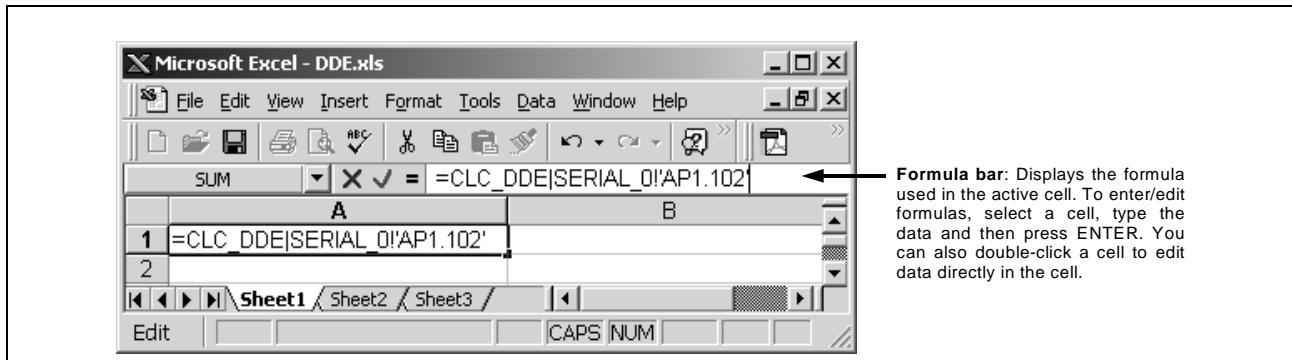


Fig. 9-31: Example DDE Formula

The **Item name** 'AP1.102' will read the value of the axis parameter A-0-0102 that is the current axis feedback position.

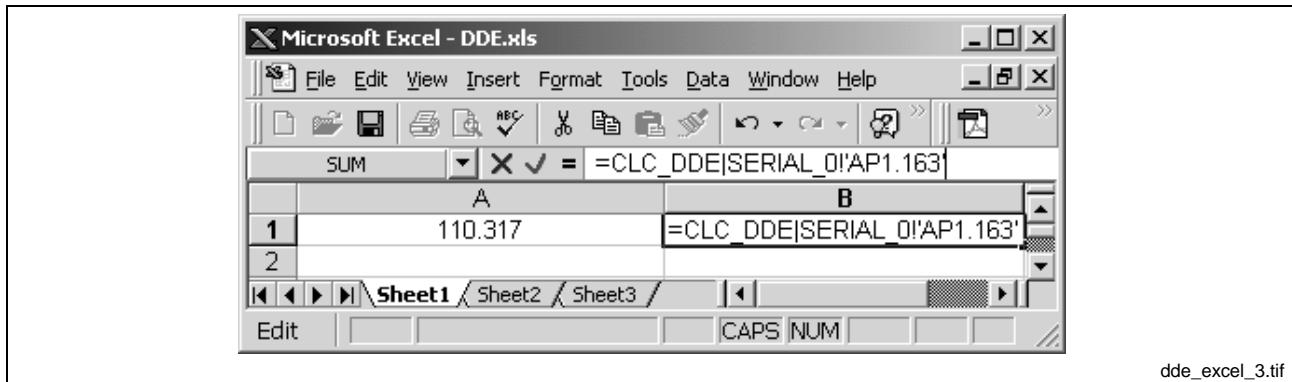


Fig. 9-32: Example DDE Formula Result

The **Item name** 'AP1.163' will read the value of the Axis Parameter A-0-0163 that is the Control Cam Output Position. Refer to **Direct ASCII Communications** for the syntax used with other item names.

A pie chart can be used to graphically illustrate the actual feedback and slave axis positions between 0° and 360°. Enter two formulas in the second row which subtract the feedback position and slave positions from 360° ($=360-A1$, $=360-B1$). Select each column and create a pie chart using the Excel *Chart Wizard*. If the VM DDE Server is active and a program is running the pie charts will rotate to reflect the current ELS positions as they change.

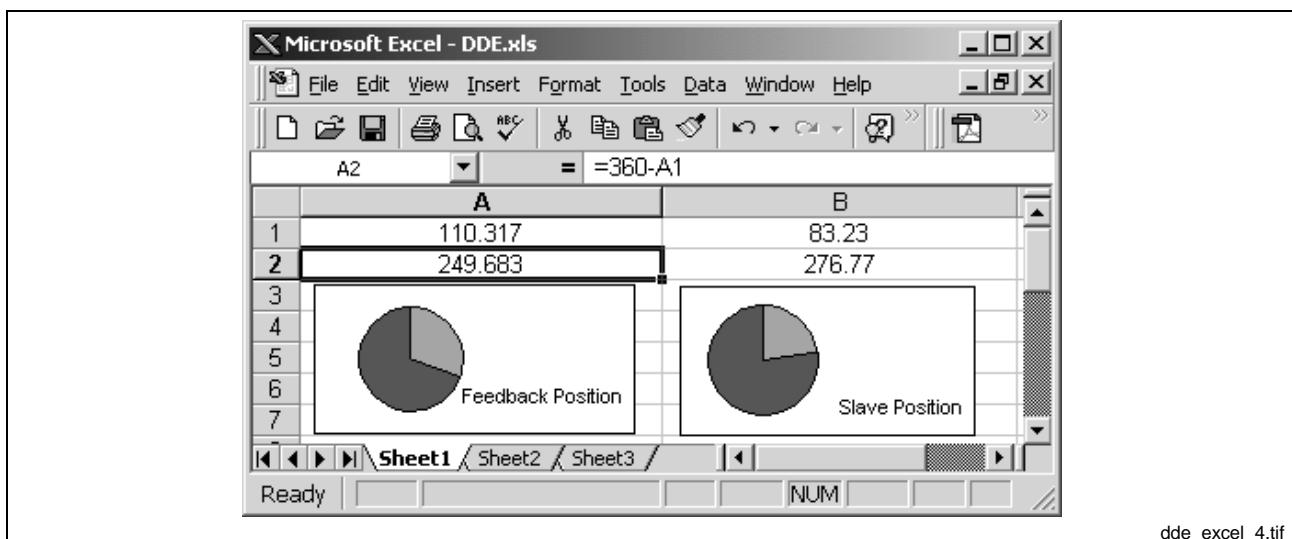


Fig. 9-33: Pie Chart

DDE Functions using Visual Basic® for Excel™

Visual Basic® for Excel™ has its own DDE Functions that can be used in a spreadsheet macro or module. The following Visual Basic® macros illustrate how to use the *DDERequest* and *DDEPoke* functions. The *DDERequest* function is used to read the values of the CAM coefficient and phase offset parameters. The *DDEPoke* function is used to write values to predefined program variables that were added in the spreadsheet. Each variable also has a corresponding **Item name** needed for DDE communication.

The variables listed in column A were predefined in the CAM program to store the CAM coefficients and Phase Adjust values. **Macro 1** requests the current coefficient values from the corresponding Control System (card) and Axis parameters.

Note: When making a *DDERequest* from a macro the Service name and Topic name are included in the *DDEInitiate* function and assigned to a variable (MYControl).

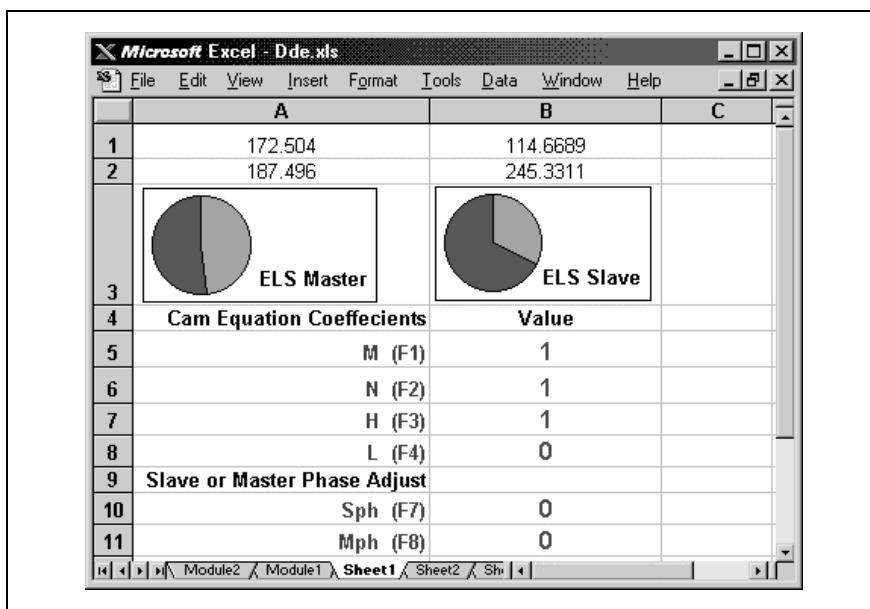


Fig. 9-34: Macro 1

Function Request()	DDEInitiate ("Service Name", "Topic Name")
MYControl = DDEInitiate("CLC_DDE", "SERIAL_0")	
m = Application.DDERequest(MYControl, "AP 1.032")	A-0-0032 Cam Slave Factor (M)
Worksheets(1).Cells(5, 2).Value = m	A-0-0031 Cam Master Factor (N)
n = Application.DDERequest(MYControl, "AP 1.031")	A-0-0033 Cam Stretch Factor (H)
Worksheets(1).Cells(6, 2).Value = n	A-0-0035 Cam Master Position (L)
h = Application.DDERequest(MYControl, "AP 1.033")	
Worksheets(1).Cells(7, 2).Value = h	A-0-0162 Cam Slave Phase Adjust
l = Application.DDERequest(MYControl, "AP 1.035")	
Worksheets(1).Cells(8, 2).Value = l	A-0-0151 ELS Phase Offset
sph = Application.DDERequest(MYControl, "AP 1.162")	
Worksheets(1).Cells(10, 2).Value = sph	
mph = Application.DDERequest(MYControl, "AP 1.151")	
Worksheets(1).Cells(11, 2).Value = mph	
End Function	

Additional variables were added to column A to adjust the ELS Master velocity (F5) and the active CAM number (I1). Column C contains the **Item name** that corresponds with each program variable (F1-F8 and I1). The DDEPoke command in **Macro 2** references the worksheet for the DDE **Item name** and value for each variable.

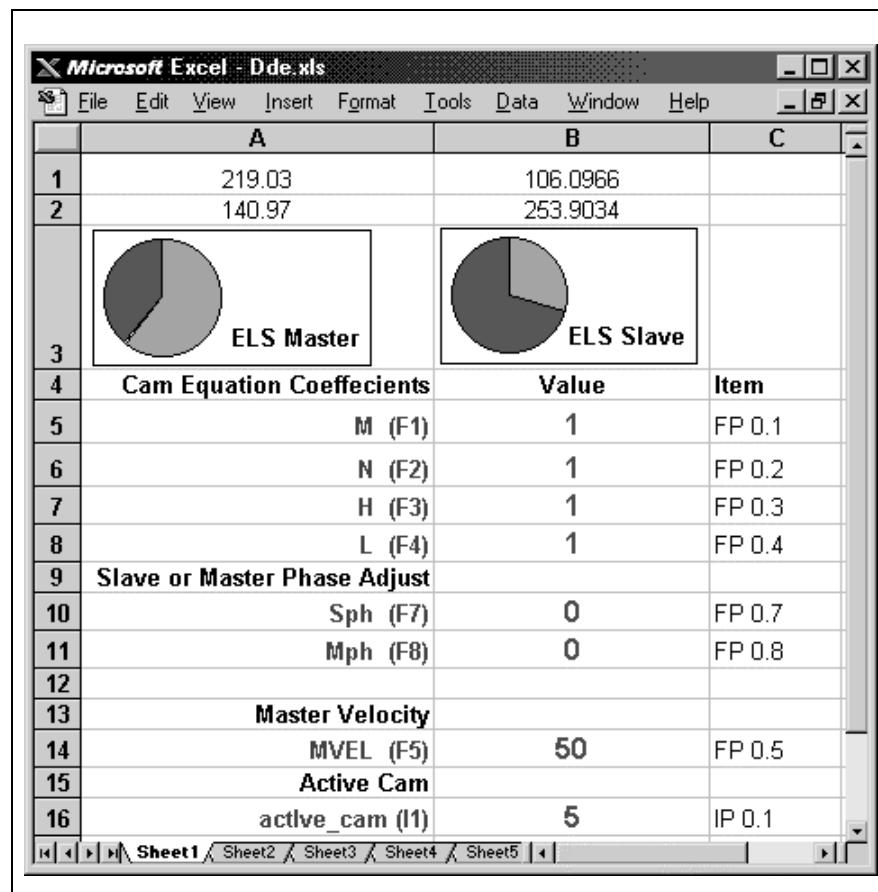


Fig. 9-35: Macro 2

When **Macro 2** is executed the data in the value column will be written or “poked” to the variables defined by the corresponding **Item names**. This allows the user to see how different values will alter the performance of the slave axis with respect to the master. The DDEPoke command uses the following syntax:

Application.DDEPoke *MYControl*, *Item Name*, *Value*

MYControl is defined in the *DDEInitiate* command and includes the DDE **Service name** and **Topic name**.

```
Function Poke()

MYControl = DDEInitiate("CLC_DDE", "SERIAL_0")

Application.DDEPoke MYControl, Worksheets(1).Cells(5, 3).Value, Worksheets(1).Cells(5, 2)
Application.DDEPoke MYControl, Worksheets(1).Cells(6, 3).Value, Worksheets(1).Cells(6, 2)
Application.DDEPoke MYControl, Worksheets(1).Cells(7, 3).Value, Worksheets(1).Cells(7, 2)
Application.DDEPoke MYControl, Worksheets(1).Cells(8, 3).Value, Worksheets(1).Cells(8, 2)
Application.DDEPoke MYControl, Worksheets(1).Cells(10, 3).Value, Worksheets(1).Cells(10, 2)
Application.DDEPoke MYControl, Worksheets(1).Cells(11, 3).Value, Worksheets(1).Cells(11, 2)

Application.DDEPoke MYControl, Worksheets(1).Cells(14, 3).Value, Worksheets(1).Cells(14, 2)

Application.DDEPoke MYControl, Worksheets(1).Cells(16, 3).Value, Worksheets(1).Cells(16, 2)

End Function
```

Wonderware

In order for Wonderware to communicate with the Control, a DDE link between the two must be created. The link, or DDE Access, tells Wonderware what Windows application to use (clc_dde.exe server) in order to communicate with the Control. This application must be running in order for Wonderware to communicate with the Control.

To establish a DDE link, choose DDE Access Names under the Special Menu in Intouch Development. The DDE Access Name Definition window will open

Press the Add button:

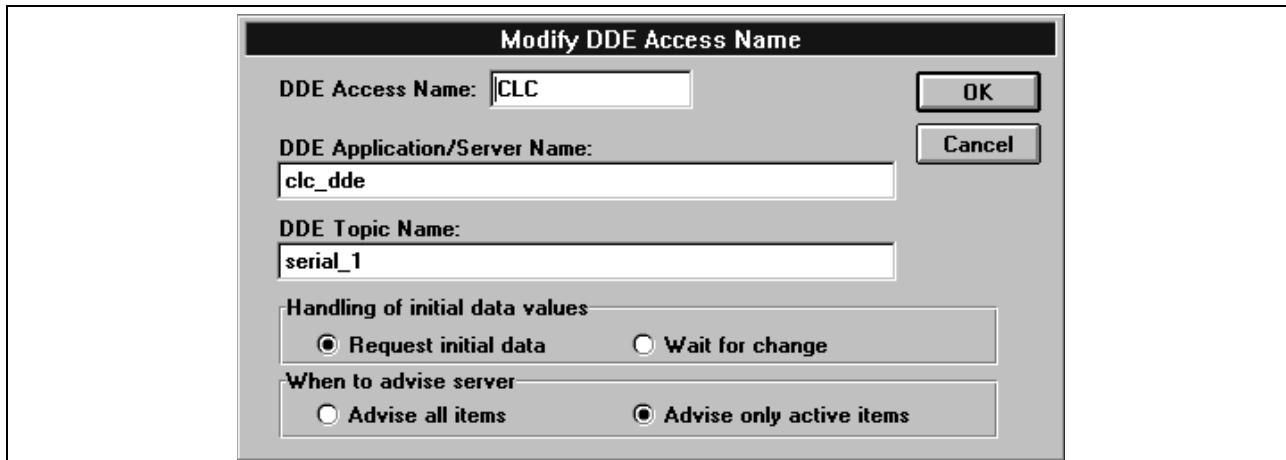


Fig. 9-36: Modify DDE Access Name

DDE Access Name

The DDE Access Name can be any name you choose.

DDE Application/Server Name

For the Control, this is the clc_dde server (clc_dde.exe) provided in the Visual Motion toolkit. It is not necessary to use the .exe extension. It is good practice, however, to include the path for the clc_dde server in the DOS Path statement and to configure Wonderware to launch the clc_dde server. If Wonderware is in your Windows Start-up group, then the clc_dde server application should also be in that group. Since Windows launches applications in the startup group from left to right, the clc_dde server icon should be to the left of the Intouch icon.

Topic Name

The topic name will depend upon the method of communication between the computer and the Control. The following items describe the different methods of communication.

Serial Communications

If you are communicating with the control via the computer's serial port, the topic name will be "serial_x" where x is the Control device number, card parameter C-0-0002. The default for a control is device #0.

If you are still not sure of the topic name, it can be found in the Control server application after VisualMotion has established communications with the Control. To find the active topic name, open "Server Configuration" under the Settings menu in the clc_dde server application. The topic name is found in the "Control Status Display" box.

Tagnames

To display a Control parameter, variable, etc. in Wonderware, the following type of tagname can be used:

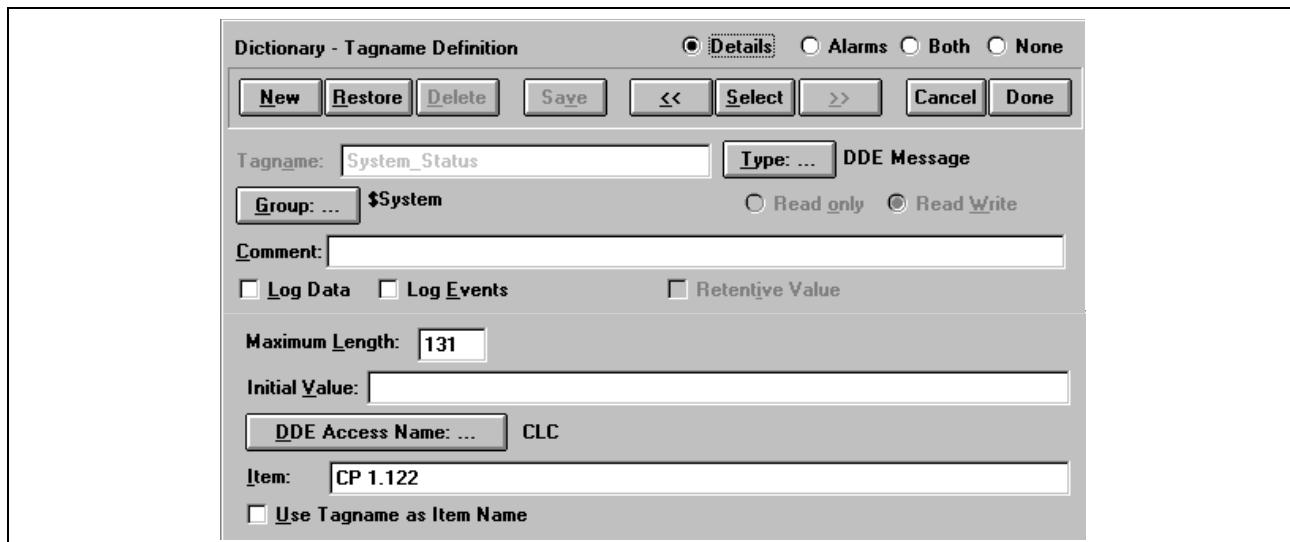


Fig. 9-37: Tagname for Displaying Parameters

The tagname in above is labeled *System_Status*. It is type DDE Message, since it is only displaying the parameter. DDE Access name is "Control".

The item field requests the parameter or variable to display; for example, the item request C-0-0122, system status is entered in the figure above. Other examples:

Parameters	Example	
Card	CP 1.122	C-0-0122 System Status Message
Task	TP 1.123	Task A param. 123, Task Status Message
Drive	DP 2.95	Drive 2, param 95 Drive Status Message
Axis	AP 3.4	Axis 3, param. 4 Axis Options
Variables	Example	
Floating Points	FP 0.12	Active Program, float # 12
Integers	IP 1.5	Program #1, integer #5
Global Floats	GF 0.1	Active Program, global float #1
Global Integers	GI 2.2	Program #2, global integer #2

To write to a Control variable, parameter, etc, the following type of tagname can be used:

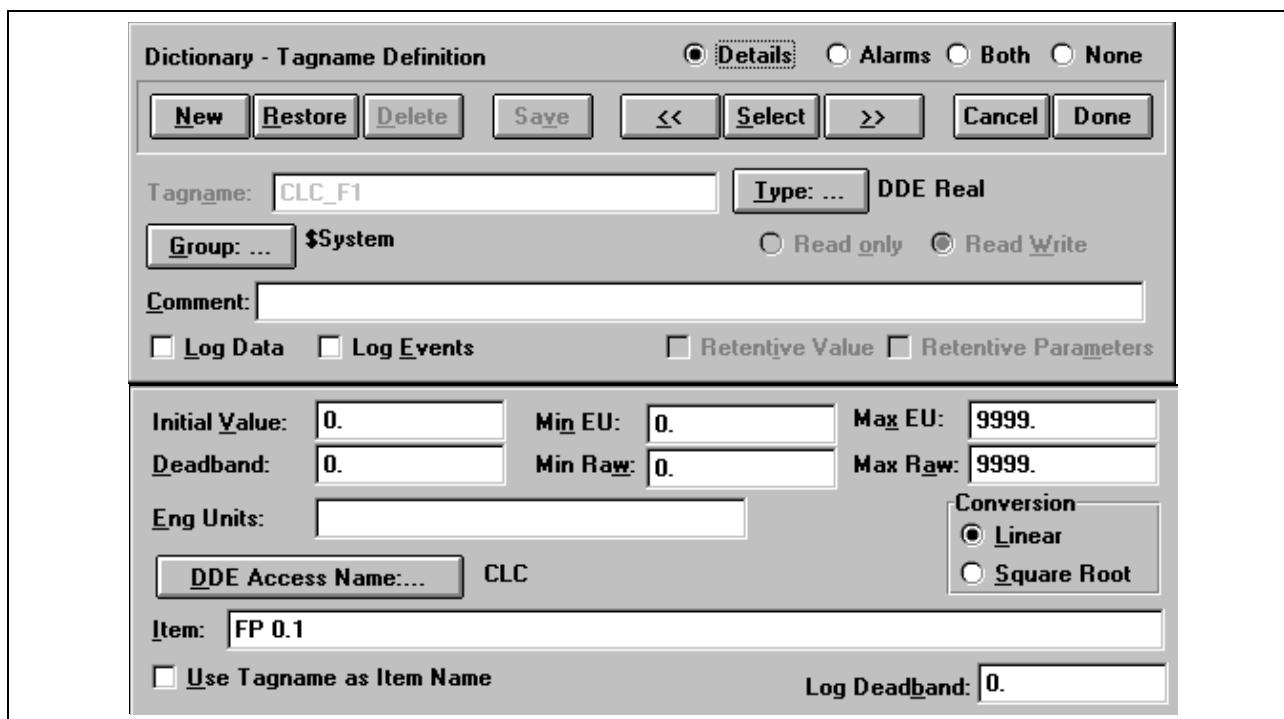


Fig. 9-38: Tagname Definition

The tagname in above makes floating point variable #1 of the active Control program a DDE real variable. The value of FP 0.1 can be changed in Wonderware by changing the value of the tagname, Control_F1.

To change a bit in a Control register:

Configure a tagname as a Type DDE Integer, min EU -99999, min Raw - 99999

max EU 99999, max Raw 99999

Tagname Reg_100

Item RD 0.100

The above tagname will write to register 100. To change a specific bit in that register, configure a pushbutton as shown below.

Object type: Button

Button Type: User Input, Discrete

Tagname: Reg_100.00

The above button has two states, on and off. When in the on state, register 100 bit 1 will be set to 1. When in the off state, that bit will be set to 0. To write to bit 2, change the tagname to Reg_100.01, bit 3 = Reg_100.02, bit 4 = Reg_100.03, bit 16 = Reg_100.15

10 Drive Parameter Editor

10.1 Overview

The Drive Parameter Editor shown in Fig. 10-1 can be launch by selecting **Commission** ⇒ **Drives** or **Diagnostics** ⇒ **Drives** from VisualMotion Toolkit's main menu.

When opened, the Drive Parameter Editor uploads the status information for the first drive found. A graphical representation of the selected drive and motor are automatically displayed and provides information as to the type of drive, drive firmware version and connected motor. The programmed position, velocity and acceleration values from the control are displayed along with the feedback status for the selected drive.

The Drive # box allows selection of another drive by entering a drive number or scrolling with the up/down list buttons. When a different drive is selected, the information on this screen is automatically updated. Since the drive internally generates rate profiles for single axis motion, the programmed acceleration is also displayed. Acceleration is not shown for coordinated motion since the control path planner manages acceleration for coordinated motion.

If applicable within inner windows of the Drive Parameter Editor window, the **Next** and **Previous** buttons are used to select from multiple drives. These buttons eliminate the need to close the current window displaying drive information and selecting a different drive from the Drive Parameter Editor window.

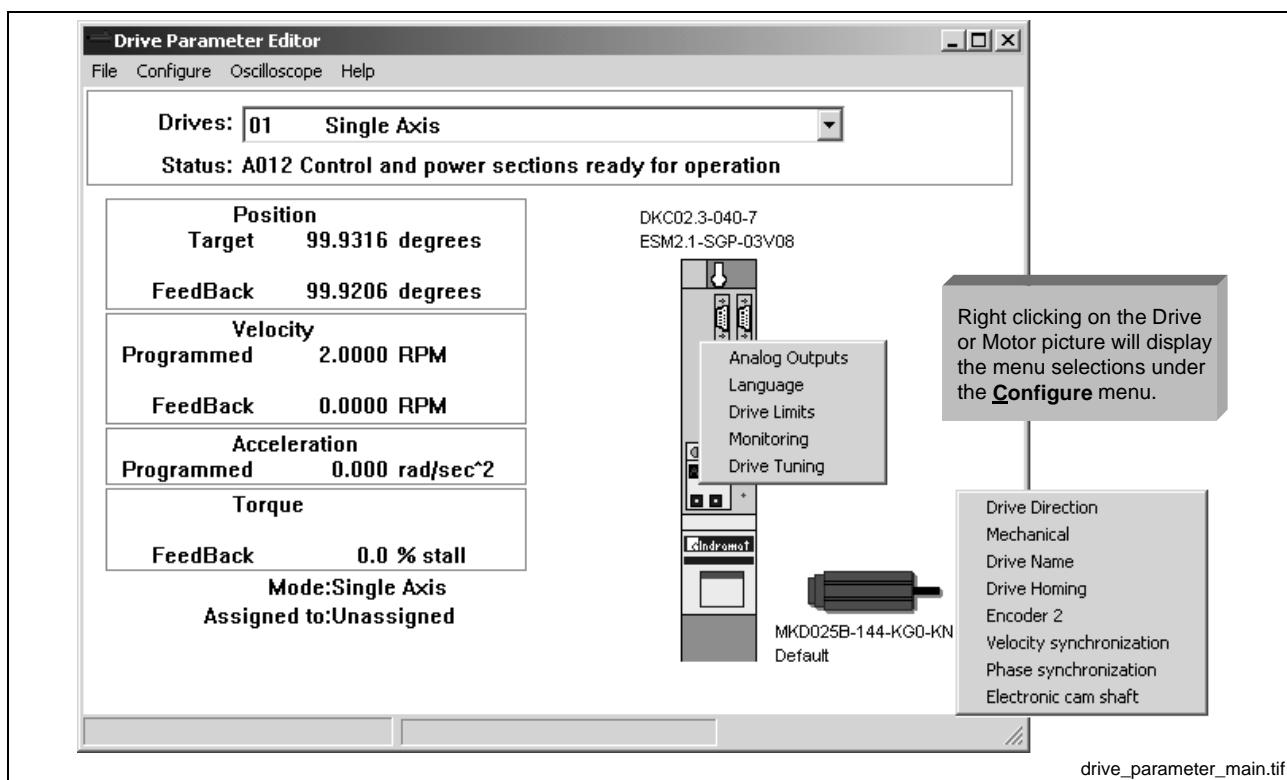


Fig. 10-1: Drive Parameter Editor

10.2 File Menu

Load Default Parameters

The Load Default Parameter selection initiates a command in the drive (P-0-4094, C800 Command Base-parameter load) to load a default set of parameters preconfigured by Indramat for the connected motor/drive combination. All drive parameters are cleared and preset with default (initial) values listed in S-0-0192, IDN-list of backup operation data. These default parameters define the basic state of the drive that permits the drive to be switched to a “ready for operation (bb)” mode. This selection enables the user to return an unstable motor/drive combination to its default settings.

Note: The system must be in parameter mode (P2) before selecting Load Default Parameters.

When selected, VisualMotion will warn as to the overwriting of drive parameters as shown in Fig. 10-2. Selecting “Yes” will issue the command the drive to load its default parameters.

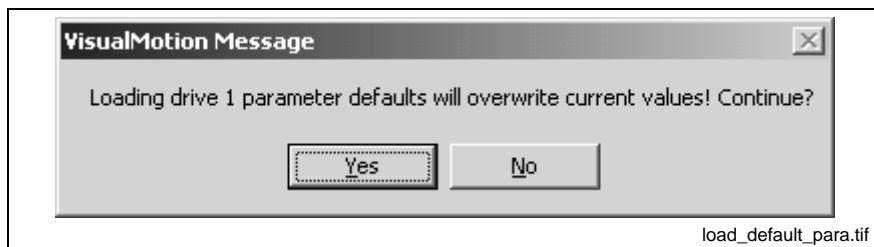


Fig. 10-2: Load Default Parameter Warning

Minimize Parameter Lists

This function can be used to minimize the data size of certain parameter lists used for configuring positional data, before uploading drive parameters for viewing or archiving. The minimized parameter lists are not used for any VisualMotion functionality. This function simply reduces the amount of data that will be uploaded for the selected drive. When finished, VisualMotion will issue the message indicating the completion of the function, as shown in Fig. 10-3.

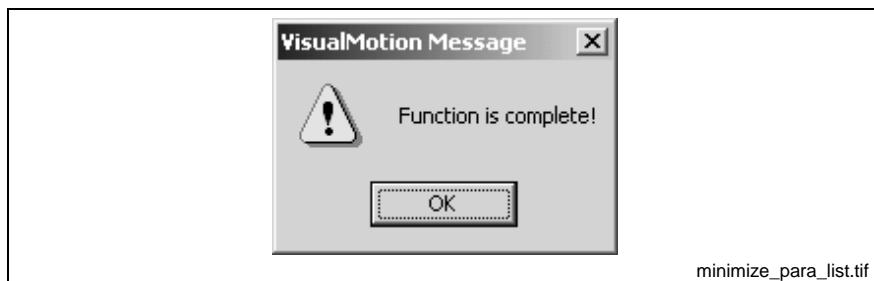


Fig. 10-3: Minimize Parameter List Completion Message

Transfer Parameters

The Drive Parameter Transfer window is used to upload drive parameters to a file for archiving or viewing. Archived drive parameters can be downloaded from a file to the control by selecting "Download to drive archived file".

Note: Downloading of drive parameters requires that the system be switched to parameter mode (P2).

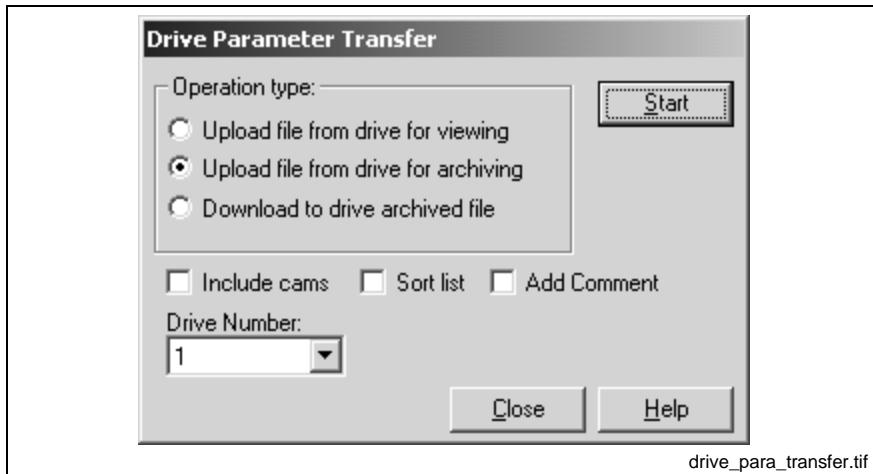


Fig. 10-4: Drive Parameter Transfer

Each drive initially contains a set of default parameters. Once the drive's parameters are modified for a system, they should be archived and saved to a file for future reference. Select a drive number for the desired operation type. The parameter set for the selected drive connected to the control may be transferred across the SERCOS ring.

Parameters may be uploaded in one of the following two methods:

- Upload file from drive for viewing
- Upload file from drive for archiving

Uploading 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 text file parameter set uploaded for viewing cannot be downloaded to the control.

Uploading for archiving saves the file with a ".prm" file extension in the default directory (*Indramat\VisualMotion8\Project*), with proper format for downloading to the control.

The check boxes are used for including cams, sorting the list of parameters and to include a comment for the archived drive parameter file.



If the "Add Comment" check box is selected, the user can enter a comment, up to a maximum of 256 characters, in the window shown in Fig. 10-5.

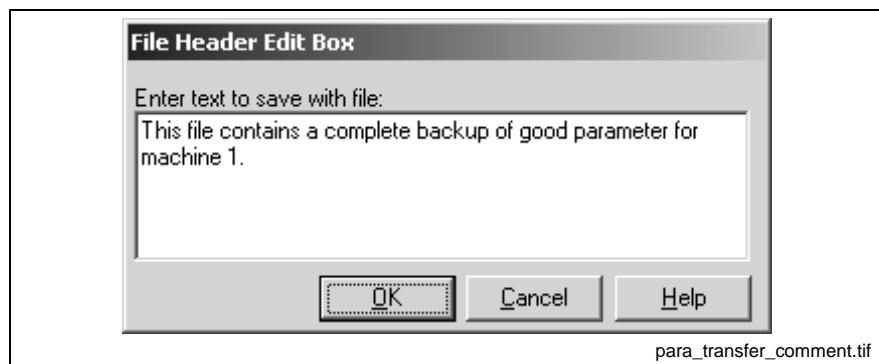


Fig. 10-5: Add Comment Window

Scan for SERCOS Drives

This function scans the SERCOS fiber optic ring for all drives connected in the system.

10.3 Configure Menu

The selections under the Configure menu can be selected by clicking on the Configure menu and choosing one of the sub-menu items or right clicking on either the drive or motor graphical representation on the main Drive Parameter Editor window.

Analog Outputs

The *Drive n Analog Outputs* window is used to define the signal selection and scaling of the drive's AK1 and AK2 analog outputs. ECODRIVE03, by default, displays the Preset Signals.

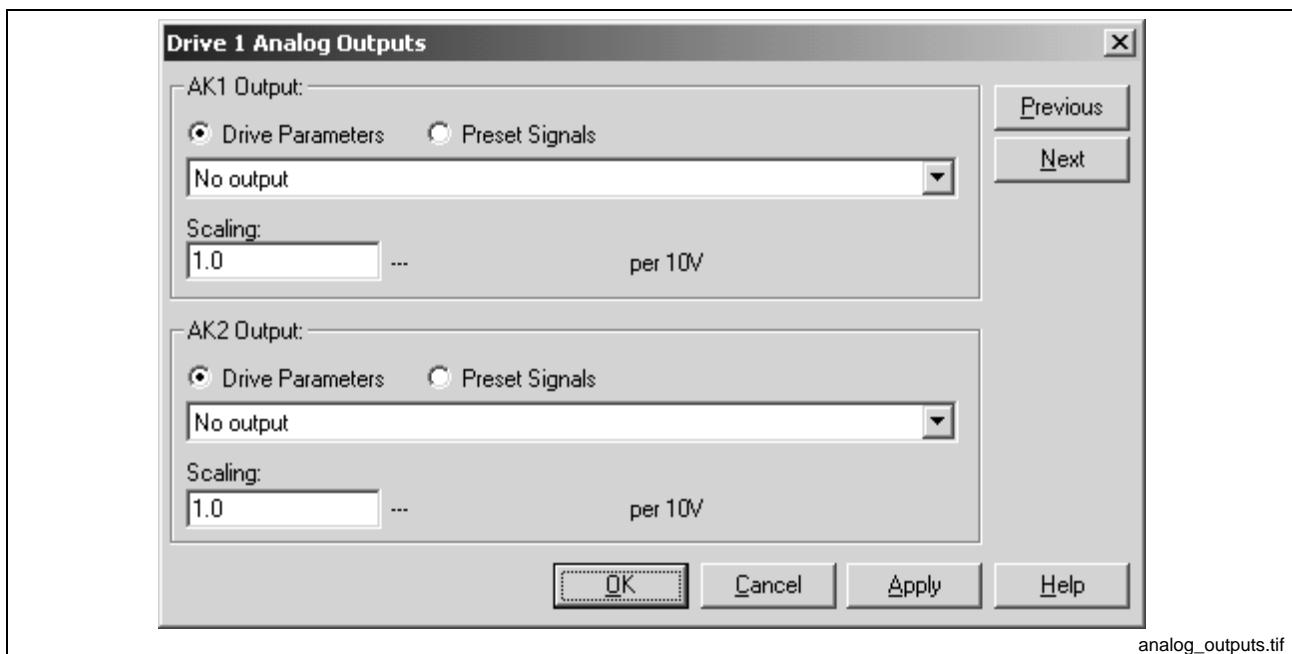


Fig. 10-6: Drive n Analog Output

The selectable AK1 and AK2 analog output signals for DIAX04 and ECODRIVE03 are listed in Table 10-1 and Table 10-2 respectively. Individual data boxes allow independent scaling of each output (maximum 10-volt output) to meet the requirements of an external indicator (analog or digital panel meter, etc.).

Preset Signals for AK1 and AK2 Analog Output Signals	DIAX04	ECODRV3
No output	X	X
Sine signal from motor encoder	X	X
Cosine signal from motor encoder	X	X
Sine signal from external encoder	X	X
Cosine signal from external encoder	X	X
Position command value difference between each SERCOS cycle	X	X
DC bus power	X	X
Rectified DC bus power	X	X
Effective current	X	X
Relative current	X	X
Thermal loading of power stage (P-0-0141)	X	X
Motor temperature	X	X
Magnetism current (Asynchronous Motors)	X	X
Velocity command at velocity controller	X	X
Synchronous position command value	X	
Synchronous velocity	X	
Master axis position fine interpolation	X	
Master axis speed in the NC cycle	X	

Table 10-1: Preset Signals for AK1 and AK2

Drive Parameters for AK1 and AK2 Analog Output Signals	DIAx04	ECODRV3
No output	X	X
S-0-0036 – Velocity command value	X	X
S-0-0037 – Additive velocity command value	X	
S-0-0040 – Velocity feedback value	X	X
S-0-0047 – Position command value	X	X
S-0-0048 – Position command value additional	X	X
S-0-0051 – Position feedback value 1 (Ext. feedback)	X	X
S-0-0053 – position feedback value 2 (Ext. feedback)	X	X
S-0-0080 – Torque / Force Command	X	X
S-0-0084 – Torque / Force feedback value	X	X
S-0-0134 – Master control word	X	X
S-0-0135 – Drive status word	X	X
S-0-0182 – Manufacturer Class 3 Diagnostics	X	X
S-0-0189 – Following error	X	X
S-0-0258 – Target position	X	
S-0-0259 – Positioning velocity	X	
S-0-0347 – Speed deviation	X	X
S-0-0383 – Motor temperature	X	X
S-0-0403 – Position feedback value status	X	X
P-0-0052 – Position value feedback 3		X
P-0-0053 – Master drive position		X
P-0-0083 – Gear ratio fine adjust	X	
P-0-0098 – Max. model deviation	X	X
P-0-0141 – Thermal drive load	X	X
P-0-4044 – Braking resistor load		X

Table 10-2: Drive Parameters for AK1 and AK2

After AK1 and AK2 are selected, click on the Apply or OK buttons to download the signal selections and scaling factors to the selected drive through the GPP control and across the SERCOS ring. For an explanation of each parameter listed in the tables above, refer to the respective drive manuals.

Drive Direction

The Drive Direction window is used for viewing and setting a drive's directional parameters. These parameters invert the direction of the commands to the drive. For example, a 5-inch move will move 5 inches negative if the Position Command is set to the reverse direction.

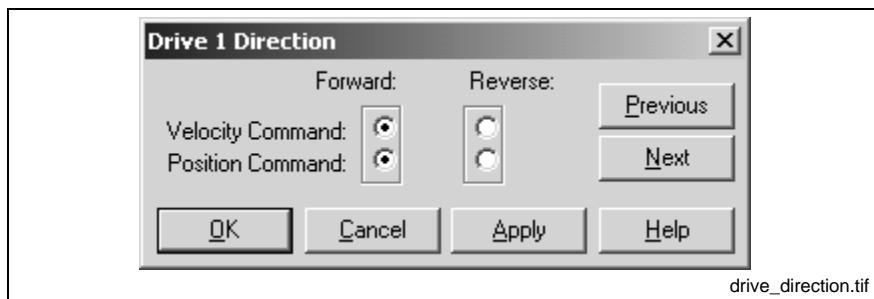


Fig. 10-7: Drive Direction

The drive currently being displayed is displayed at the top of the window. To change a drive's settings, switch to parameter mode, then select the direction buttons and press the Apply or OK button.

Drive Language Selection

The Drive Language Selection window in Fig. 10-8 is used to change the language for the selected drive. All drive parameters for the selected drive are changed. Any window within the Drive Parameter Editor that displays drive parameter text, will also be in the selected language.

Any text that is part of the User Interface will remain in the configured programming language set in **Settings** ⇒ **Configuration** from VisualMotion Toolkit's main window.



Fig. 10-8: Drive Language Selection

Drive Name

The Drive Name window is used for naming a drive. The default name is the axis number; however, a custom name can be assigned and saved to any of the drives on the SERCOS ring.

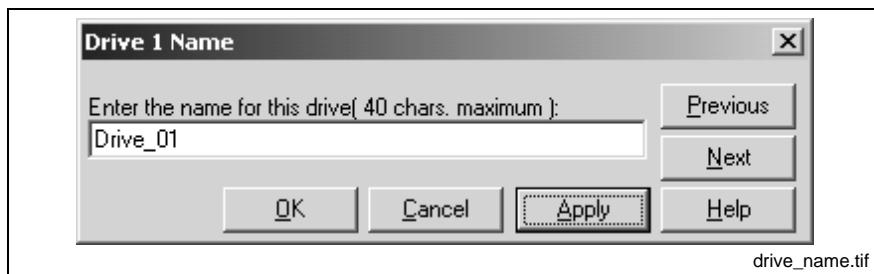


Fig. 10-9: Drive name

Drive Monitoring

Selecting Drive Monitoring from the Configure menu opens the Drive Monitoring window and updates the information in Fig. 10-10 for the current drive selected. Double clicking on any of the white value areas opens the *Edit Parameters* window where values can be changed and saved.

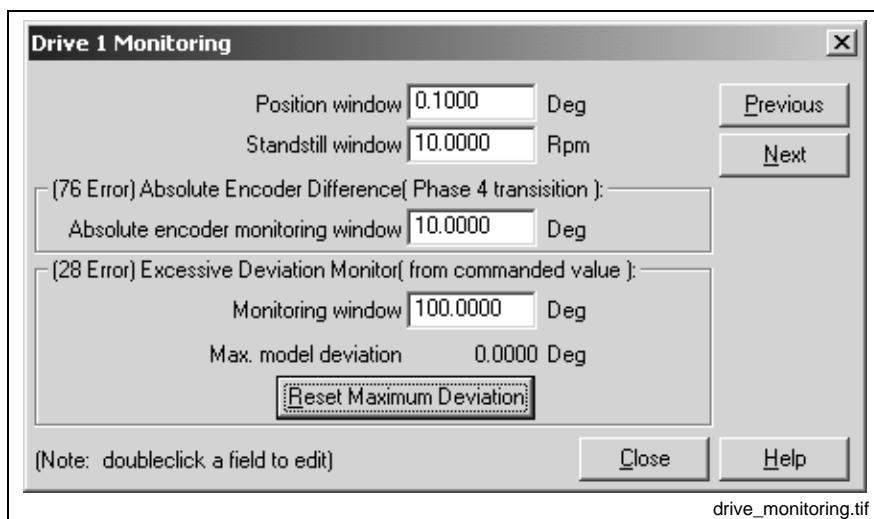


Fig. 10-10: Drive Monitoring

Position window (S-0-0057) sets the tolerance distance used to determine if it's in position.

Standstill window (S-0-0124) sets the velocity to determine if it's at a standstill.

Absolute Encoder Difference (P-0-0097) sets the maximum distance the motor can move when powered down, without causing an error 76 on power up to phase 4.

Monitoring window (S-0-0159) sets the maximum position unit (or percent for some drives) from the command value before the drive issues an error 28.

The current **Maximum Model Deviation** from the command value is displayed in position units (or a percent for some drives). This value can be reset using the *Reset Maximum Deviation* button.

Drive Tuning

Selecting Drive Tuning from the Configure menu opens the Drive Tuning window for the currently active drive. The adjustments within the Velocity loop are related to a machine's performance. The Current Loop adjustments are set according to the respective motor/drive combination and should not be altered from their initial default setting. If modifications are required, double clicking on any of the boxes in Fig. 10-11 will open the *Edit Parameters* window where values can be changed and saved.

Clicking on the *Load Defaults* button will load default parameter values for all drive loops. The control must be in Parameter Mode. The default values assume a 1:1 ratio of load inertia to motor inertia.

For further information on tuning Indramat drives and tuning parameters, refer to the respective drive manual.

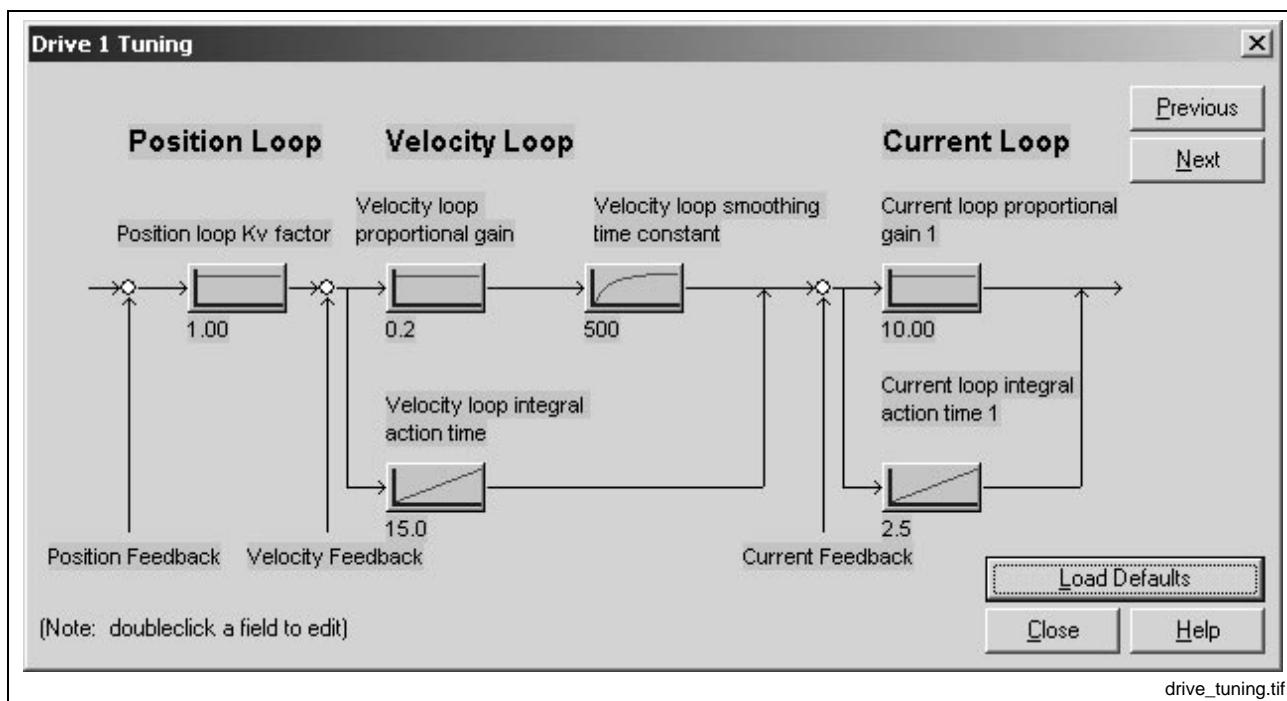


Fig. 10-11: Drive Tuning

Position Loop

The Position Loop **K_v Factor** (SERCOS parameter S-0-0104) data entry box adjusts the proportional gain of the Position Loop. This parameter value may be set from 0 to 655.35.

Velocity Loop

The Velocity Loop **Proportional Gain** data entry box adjusts the gain of the loop feedback path (SERCOS parameter S-0-0100). The gain is initially adjusted by the drive/motor combination for a 1:1 load/motor inertia ratio. This parameter value may be set from 0 to 6553.5.

The **Smoothing Time Constant** data box sets a low-pass filter that limits the bandwidth of the feedback loop and reduces digital quantization effects (Indramat parameter P-0-0004). The time constant is set in microseconds, any entry under 250 μ s switches off filtering.

The Velocity Loop **Integral Action Time** data box also sets a low pass filter time constant integrating the velocity loop feedback signal (SERCOS parameter S-0-0101). This parameter is typically used to adjust the loop response time, matching the load to motor and reducing

overshoot that may result from a rapid (step) change. This filter has a lower frequency breakpoint than the Smoothing/Roll-off value.

Current Loop

The Current Regulator Proportional Gain (S-0-0106) and Integral Action Time (S-0-0107) adjustments are used for the initial tuning of respective motor/drive combinations. This adjustment should not be changed once it is set for a specific system.

Drive Limits

The Drive Limitations window is used for enabling and setting hardware and software position limits. Bipolar limit values for each drive can also be modified and saved. Positive and Negative Software **Travel Limits** set float values for the drive's Positive (S-0-0049) and Negative (S-0-0050) parameters. The Travel Limits are not active unless Enabled is checked.

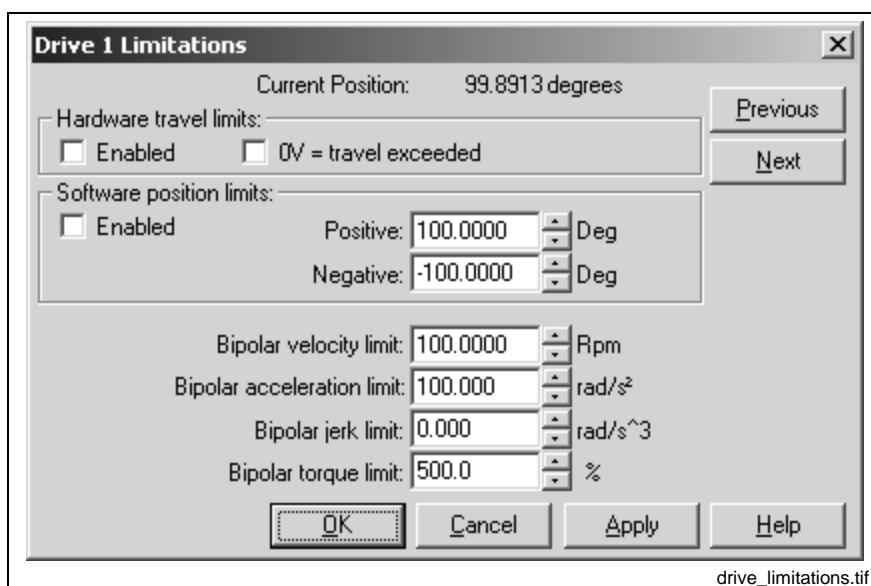


Fig. 10-12: Drive Limits

The **bipolar velocity limit** value (S-0-0091) determines the maximum allowable speed in either direction. If the velocity limit value is exceeded, the drive responds by setting the message "ncommand > nlimit" in Class 3 Diagnostics (IDN S-0-0013).

The **bipolar acceleration** parameter (S-0-0138) reduces the maximum acceleration ability of the drive symmetrically around 0, to the programmed value in both directions.

The **bipolar jerk limit** value (S-0-0349) determines the maximum allowable jerk in either direction.

The **bipolar torque limit** value (S-0-0092) determines the maximum allowable torque in either direction. If the torque limit value is exceeded, the drive sets the message "T > Tlimit" in Class 3 Diagnostics (S-0-0013).

Drive Reference

When the Drive Reference window is opened the drive's feedback type parameter is read and either the single or the multi-turn encoder window is displayed.

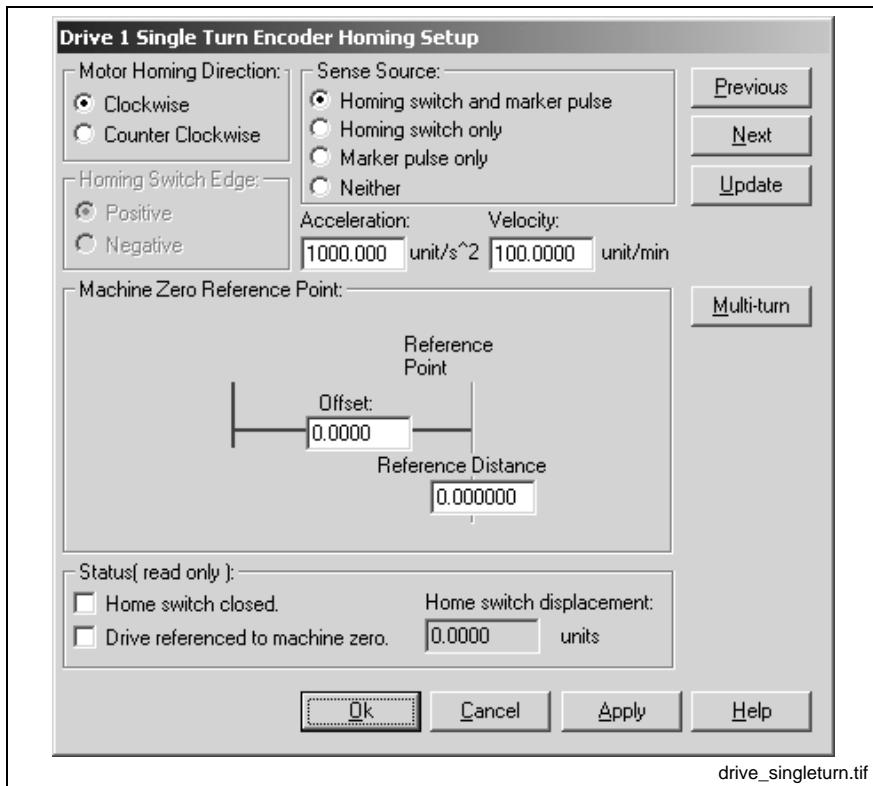


Fig. 10-13: Drive Reference

Homing Direction

The homing direction of the motor is selected in either clockwise or counter-clockwise direction (facing the motor shaft).

Sense Source

The motor's encoder is considered to be at the home position if the selected option is satisfied. The Neither selection disables the homing reference source. An initial acceleration and velocity for homing can also be specified.

Machine Zero Reference Point

Offset: A position that the motor moves to after the selected Sense Source is reached. The motor first moves to the selected sense source; for example, marker pulse, homes to offset position value (S-0-0150) and then copies the Reference Distance value (S-0-0052) to axis parameter A-0-0102.

Reference Distance: Position value that is written to axis parameter A-0-0102 after a homing procedure command. All commanded position move values are referenced from this position.

Status (read only)

Home switch closed: This box is checked after the motor reaches its set Sense Source.

Drive reference to machine zero: This box is checked after a *Machine Zero Reference Point* is established.

Multi-turn Encoder

When a drive is selected containing a motor with an absolute encoder, the *Multi-turn Encoder Homing Reference* window in Fig. 10-14 is automatically launched.

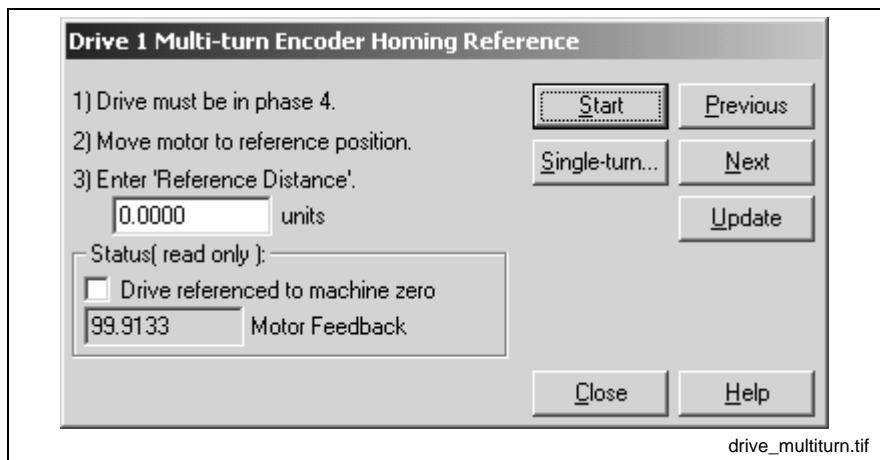


Fig. 10-14: Multi-turn Encoder Homing Reference

This window is used to enter a *Reference Distance* for machine zero. Once this distance is entered, click on the Start button and a message window will open informing you that "*All motion for this axis will be referenced to this new position*". Press *OK* to accept, or *Cancel* to Abort.

After successful completion of this procedure, the checkbox "Drive referenced to machine zero" will be checked.

Encoder 2

The Encoder 2 setup displays three consecutive windows that are used for configuring an external encoder for each drive in a system.

Note: The system must be switched to parameter mode (P2) before an external encoder can be setup.

Each drive having an external encoder requiring setup must first be selected from the drop-down list in the Drive Parameter Editor main window.

Select one the following **Function** types in Fig. 10-15 and click on the *Apply* button before proceeding to the next window. The available function types are as follows:

- Additional load side feedback
- ELS Master
- Single load side feedback
- Measuring wheel
- Spindle



Fig. 10-15: Drive n Encoder 2 Setup

Drive Encoder Setup - 2nd Window

Clicking on the *Next Step* button will display the second setup window where the Encoder Direction, Application Type and Encoder Type are selected.

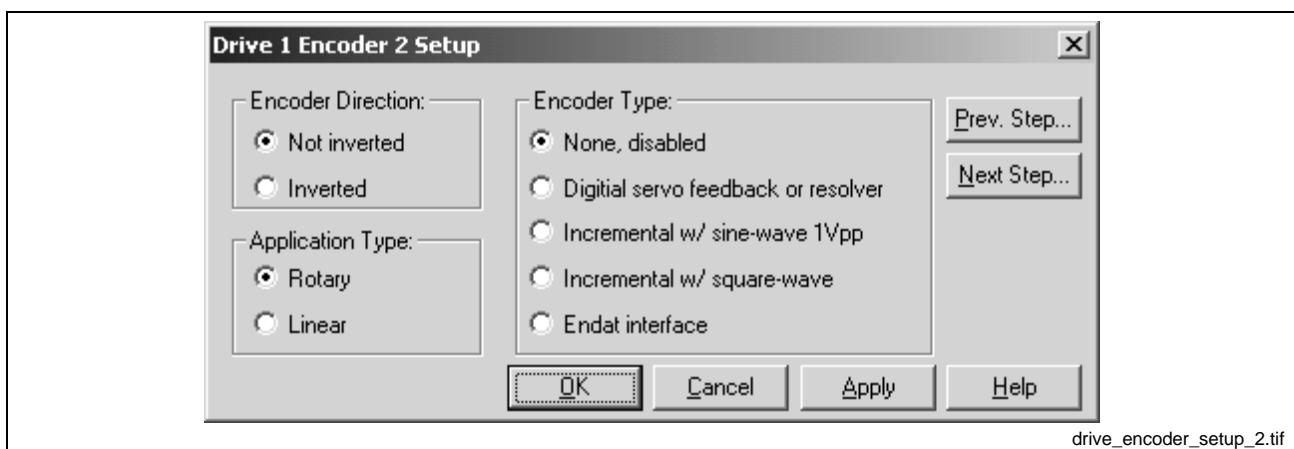


Fig. 10-16: Drive n Encoder 2 Setup (second window)

Encoder Direction sets the rotational direction of the external encoder with no relationship to the motor that is attached to the same drive. The rotational direction is viewed while facing the encoder's shaft.

- **Not inverted** is a clockwise rotation.
- **Inverted** is a counter clockwise rotation.

Application Type identifies the application as either rotary or linear.

Encoder Type identifies the style of the external encoder being used.

After the selections are made, click on the Apply button before continuing to the Next Step window. Clicking on the Prev. Step button will backup one window.

Drive Encoder Setup – 3rd Window

The last setup window in Fig. 10-17 is used for setting feedback resolution, filter time constant and monitoring window.



Fig. 10-17: Drive n Encoder 2 Setup (last window)

- **Feed Constant** is grayed out
- **Feedback Resolution** is used to account for diameter difference when coupling to a larger measuring contact wheel. The value entered in this field is the quotient of the equation in the figure below.

$$\frac{\text{Circumference of measuring wheel } (2\pi r)}{\text{Line counts per revolution of encoder}} = \text{Feedback Resolution}$$

Example:
An encoder with a line counts per rev. of 256 is coupled to a measuring wheel with a radius of 70mm

$$\frac{2 * \pi * 70 \text{ mm}}{256 \text{ counts/rev}} = 1.7180 \text{ mm/rev}$$

Fig. 10-18: Feedback Resolution value

- **Filter Time Constant** is used to filter out negative effects resulting from poor coupling of encoder 2.

- **Monitoring Window Feedback 2** defines the maximum allowable deviation between motor encoder and secondary (measuring wheel) encoder.

Note: When Encoder 2 is used as a measuring wheel, the monitoring window value must be set to 0.0, otherwise an excessive deviation error will be issued.

Encoder 2 Measuring Wheel Example

The following drive families and firmware versions support the measuring wheel command

- ECODRIVE03 using SGP-01V13 or later firmware.
- DIAX04 using ELS-05V11 or later firmware.

Measuring wheel example

Roll feed applications use a drive's primary motor to move sheet material (through the use of motor-driven rollers) off of a larger roll for processing further down the line. Due to slip or lag from the motor-driven rollers, the encoder on the motor is not suitable for measuring material lengths. A secondary (external) encoder can be configured as a measuring wheel to provide a more precise position value from a remote location on the machine. The torque to turn the measuring wheel is minimal, so the slip is negligible.

Note: The measuring wheel must remain in constant contact with the material or the position control loop will not be properly closed. When contact with the material is no longer possible, the position feedback must be switched back to the motor encoder.

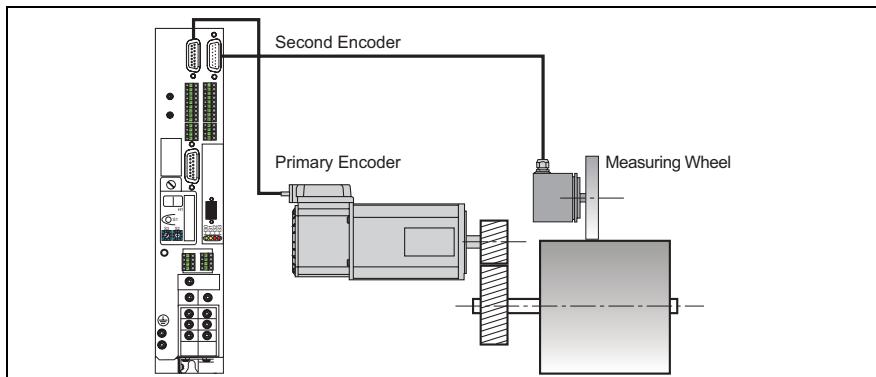


Fig. 10-19: Roll feed using a measuring wheel

Encoder 2 Setup

Using the Encoder 2 setup windows, select Measuring Wheel as a **Function** type in Fig. 10-18 and click on the Apply button before proceeding to the next window. When Measuring Wheel is selected and saved, a "3" is written to parameter **P-0-0185, Function of encoder 2**.

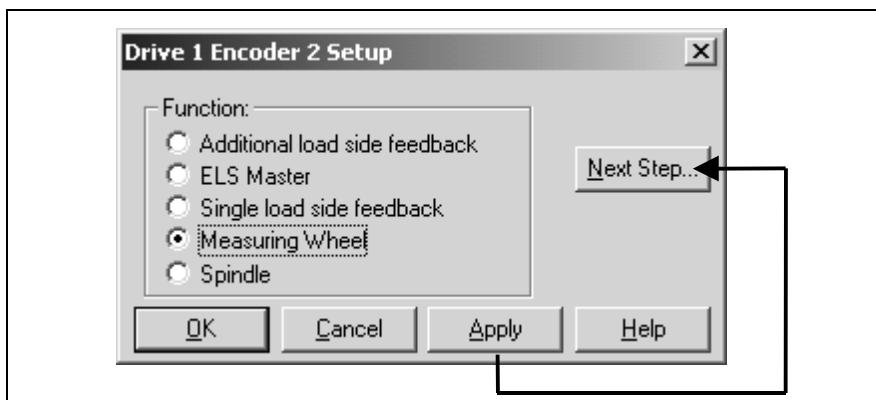


Fig. 10-20: Drive n Encoder 2 Setup

Click on the Next Step button. Select the encoder's direction and type.

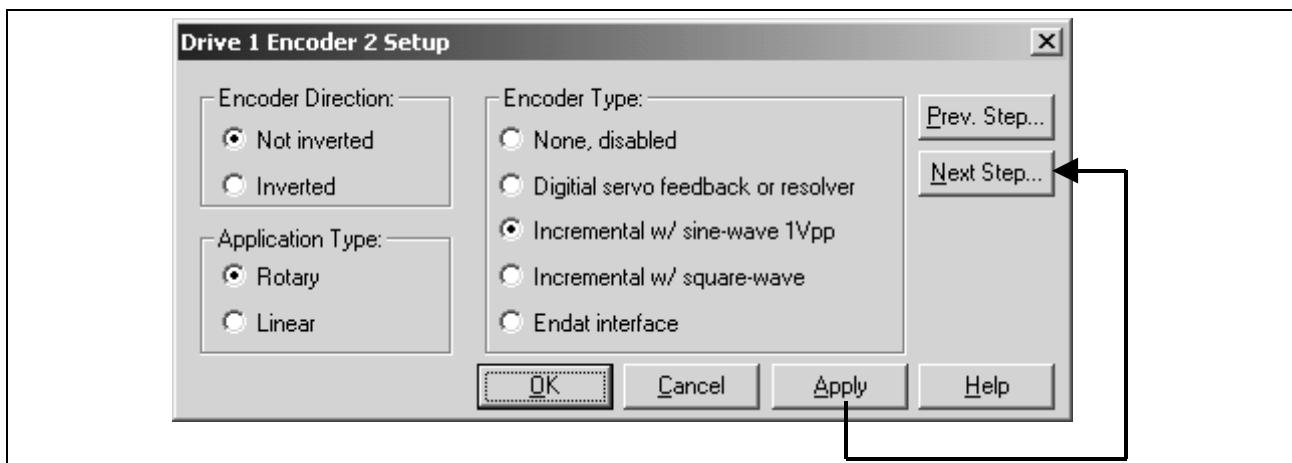


Fig. 10-21: Drive n Encoder 2 Setup (second window)

- **Encoder Direction** sets the rotational direction of the external encoder with no relationship to the motor attached to the same drive. The rotational direction is viewed while facing the encoder's shaft.
 - **Not inverted** is a clockwise rotation.
 - **Inverted** is a counter clockwise rotation.
- **Application Type** identifies the application as either rotary or linear.

Note: Application type should be set to rotary. Linear scales have a defined range of movement. For this reason, they are not typically applied to applications that require an infinite range of motion, such as roll feed.

- **Encoder Type** (P-0-0075, Feedback 2 type) identifies the style of the secondary encoder being used.

After the selections are made, click on the Apply button before continuing to the next window.

The last window in the Encoder 2 setup is used to fine-tune the accuracy of the measuring wheel.



Fig. 10-22: Drive n Encoder 2 Setup (last window)

Activating the Measuring Wheel

Before the measuring wheel can be activated, the following step must be performed:

1. The encoder must be configured as a measuring wheel using VisualMotion's Encoder 2 Setup.
2. The material must be in the feed roll and in contact with the material to ensure stability.

Positioning control is switched from the motor encoder to the measuring wheel by changing the value of parameter **P-0-0220, D800 Command Measuring wheel Operation Mode**, as stated in Table 10-3.

Value of Parameter P-0-0220	Description
0	Switches position control from measuring wheel to motor encoder.
3	Switches position control from motor encoder to measuring wheel.

Table 10-3: Parameter P-0-0220 settings

When a value of "3" is entered into P-0-0220, the positioning control is switched to the measuring wheel. When a value of "0" is entered into P-0-0220, the motor's position feedback is set to the same value as the measuring wheel and positioning control is switched back to the motor encoder.

Using VisualMotion to Activate Measuring Wheel

Using a VisualMotion program, a parameter transfer icon can be used to change the value of P-0-0220. Add the parameter transfer icon prior to motion in the program flow and setup the Parameter Transfer window in Fig. 10-23 as shown.

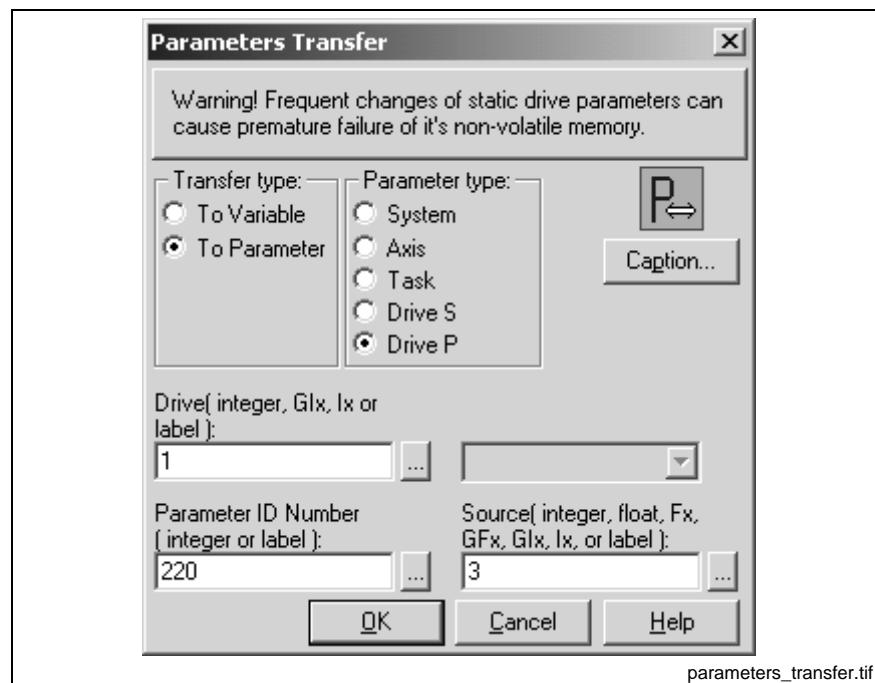


Fig. 10-23: Parameter Transfer

Mechanical

Selecting Mechanical from the Configure menu opens the *Drive n Mechanicals* window and uploads the current values from the drive. This dialog window allows easy access to several important parameters, which must be set before running any motion programs.

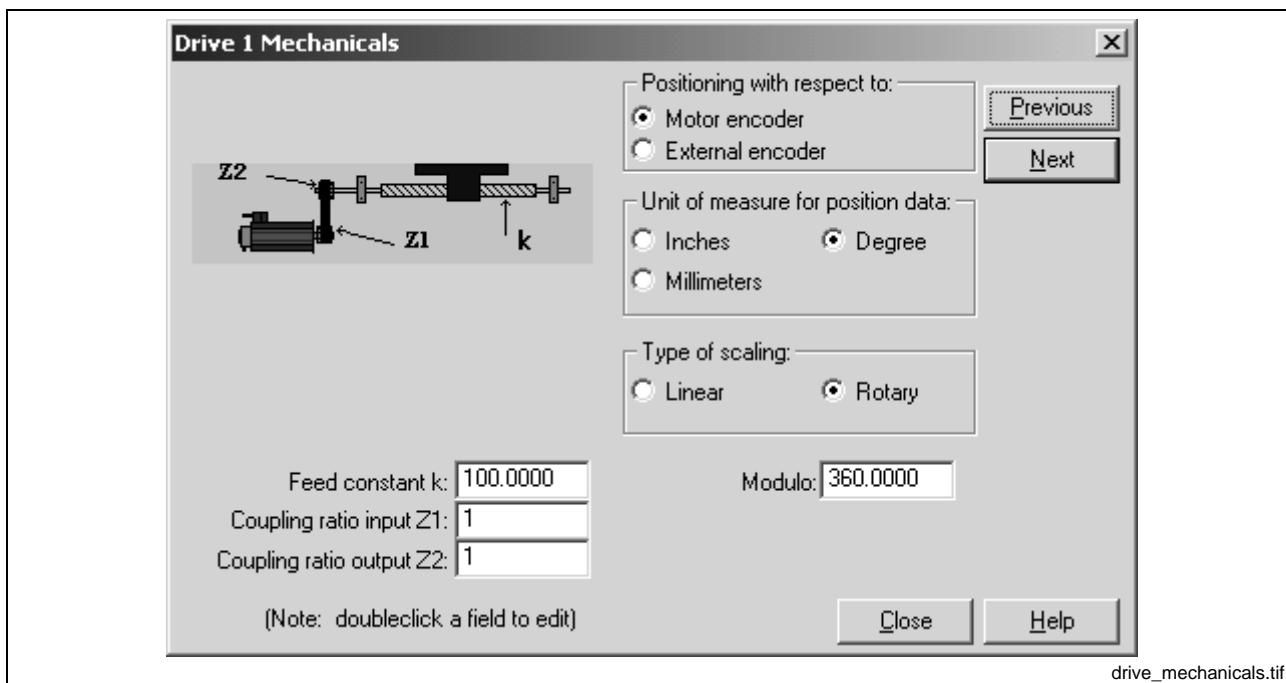


Fig. 10-24: Drive n Mechanicals

Positioning with respect to:

- Motor encoder** configures the drive to use the motor's encoder to close the position loop and provide cyclic feedback from drive parameter S-0-0051.

- **External encoder** configures the drive to use the connected external encoder to close the position loop and provide cyclic feedback from drive parameter S-0-0053.

Units of measure for position data:

This selection allows for inches, millimeters or radians to be used for the system-wide unit of measurement (A-0-0005).

Type of scaling:

Type of scaling can be linear or rotary (A-0-0004 bit 2). When linear is selected, absolute positioning is enabled in the drive. When rotary is selected, position is in degrees, velocity in RPM, and acceleration in radians/sec².

Feed constant k:

Feed Constant allows setting the ratio of movement in system units resulting from each revolution of the driven shaft (S-0-00123). For example, a five Threads per Inch ball screw provides 0.200 inch movement per revolution.

Coupling ratio

Output (Z2) and Input (Z1) Revolutions permit setting the ratio between the motor shaft and driven shaft. Integer values permit preservation of maximum system accuracy with ratios that result in repeating decimals (i.e. 1:3 = 0.333333). These values are set in the drive's Input Revolutions of Load Gear (S-0-0121) and Output Revolutions of Load Gear (S-0-0122) parameters.

Modulo

The **Modulo** value (S-0-0103) is indicated as a maximum rotational value in which the motor will turn before resetting the position to zero. The default value when operating in modulo mode is 360. (Modulo mode is set in the drive by setting bit 7 of IDN S-0-0076, Scaling Options for Position Data).

Parameters

Selecting **Parameters** from the Configure menu opens the *Parameter Overview* tool in Fig. 10-25. 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 Ring 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.

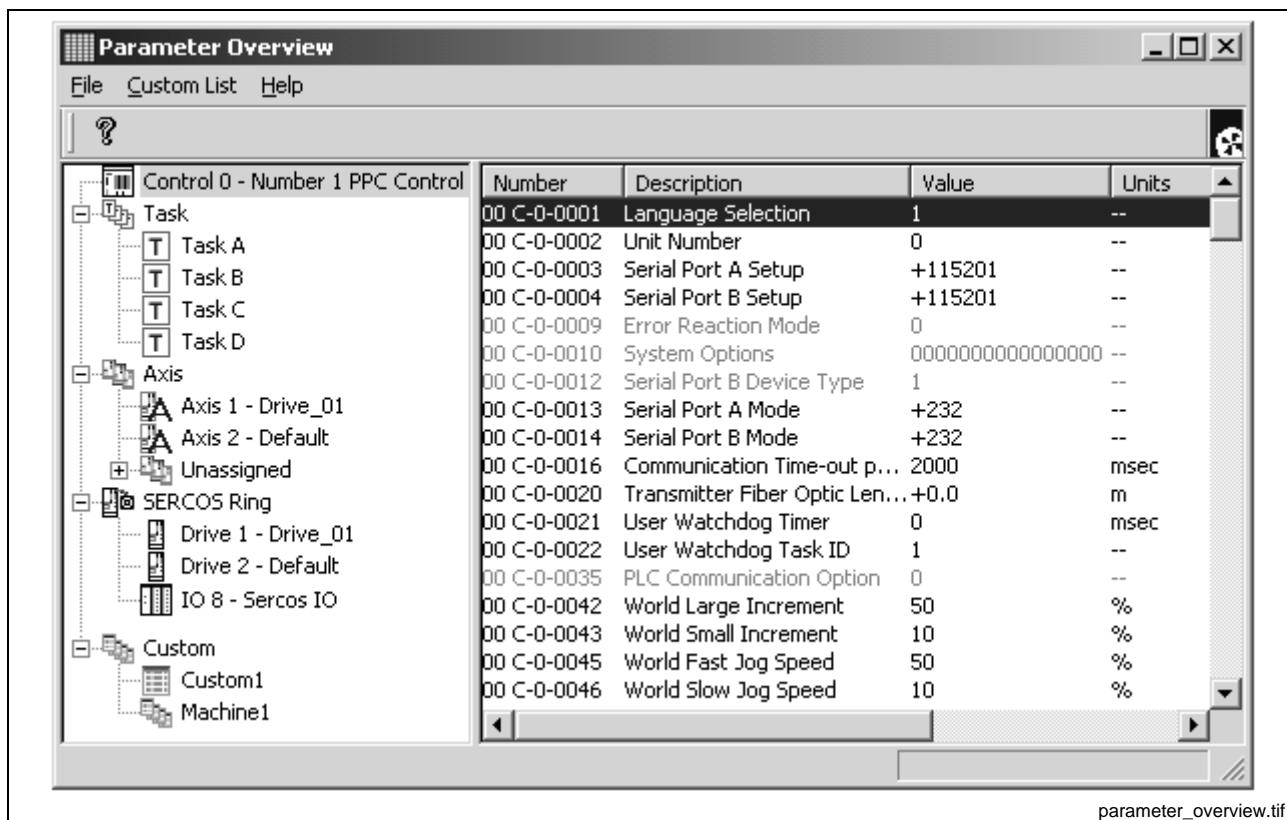


Fig. 10-25: Parameter Overview Window

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.

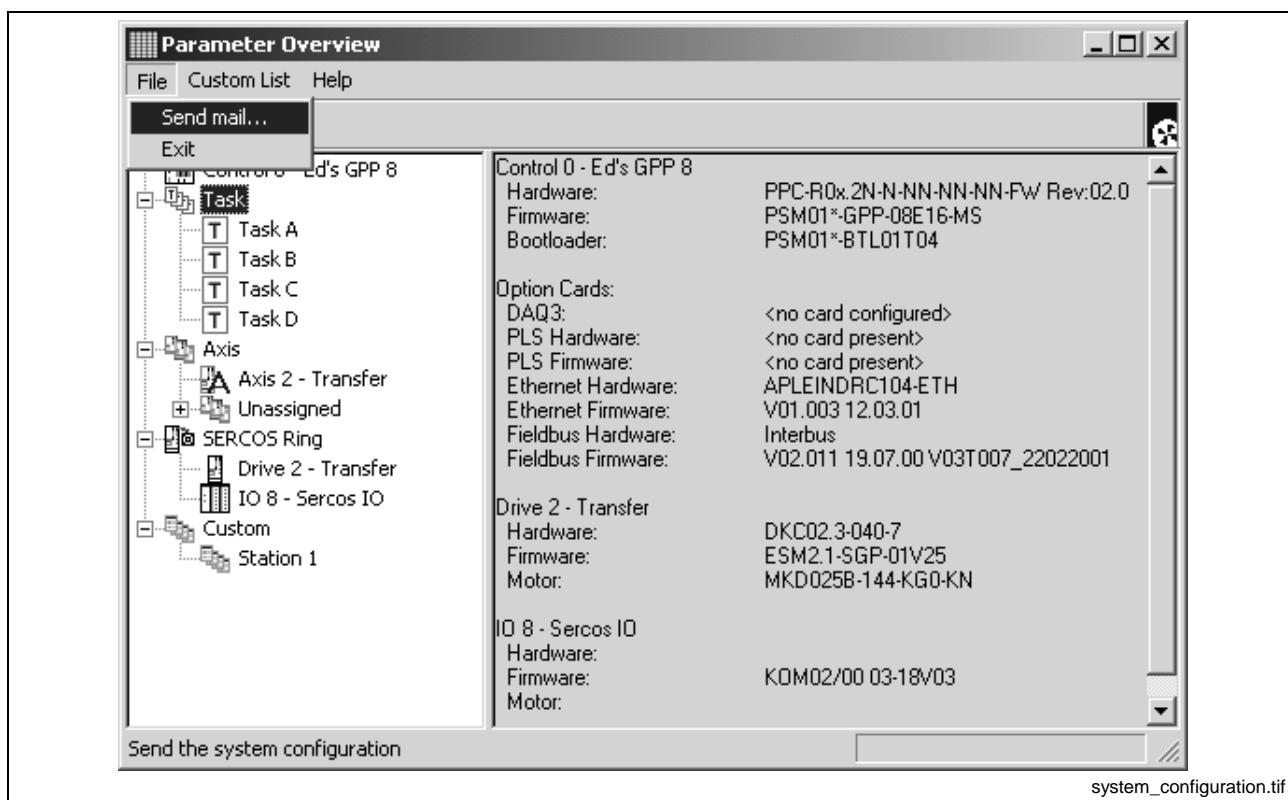


Fig. 10-26: System Configuration

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 Fig. 10-26.

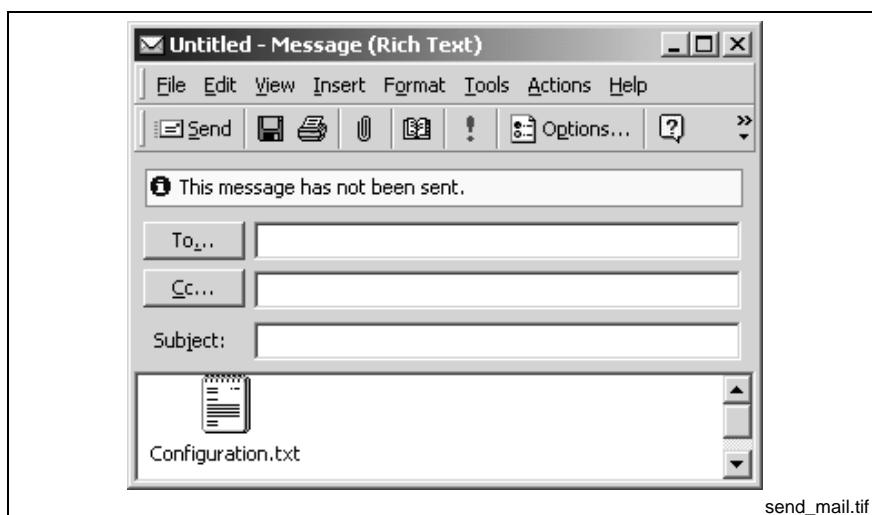


Fig. 10-27: Send Mail

Note: The *Send mail...* option will not be visible if the connected PC does not have a configured e-mail service.

Parameter Access

Access to parameters is controlled by the Parameter Overview tool and is dependent on the current Phase. Parameters are displayed in different colors to provide a visual representation of their access level. The following table explains the color code / access combination.

Color code	Access Level
gray text	read only
black text	read/write
blue text (parameter list)	entries and number of entries in the list can be modified
red text (Phase switch error)	read/write

Table 10-4: Access to Parameters

Edit a Parameter

To edit a parameter, double click on the desired parameter from the *Parameter Overview* window or select the parameter, right click and select "**Edit Selection**". The parameter's data limits are displayed above the input field from minimum to maximum values. A gray field in the parameter value box is an indication that the parameter is read-only. Pressing the **Help** button or F1 key can access context sensitive help for the selected parameter. Fig. 10-28 shows the three basic parameter edit window that can be displayed.

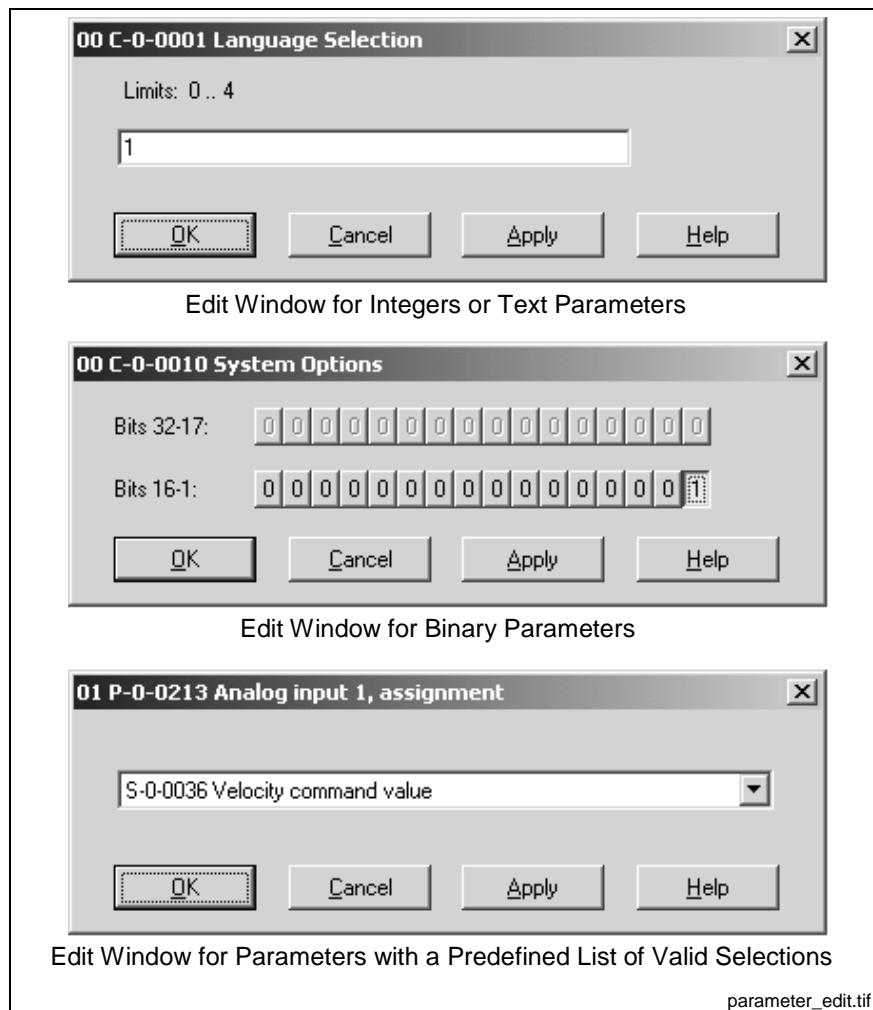


Fig. 10-28: Parameter Edit Window

Edit a Binary Parameter

The binary parameter edit window in Fig. 10-29 is modified by clicking on the desired bit(s). Holding the mouse cursor over a bit, without clicking, 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.

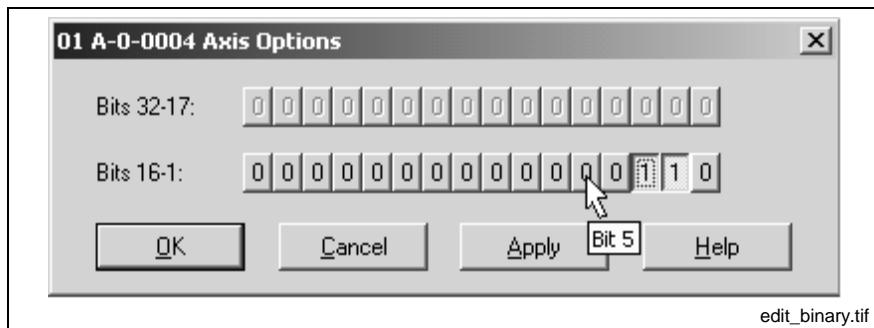


Fig. 10-29: Edit a Binary Parameter

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 **Refresh (F5)** updates all the parameters visibly displayed in the main window. The **Find (F3)** option allows the user to location a parameter by using a partial description.

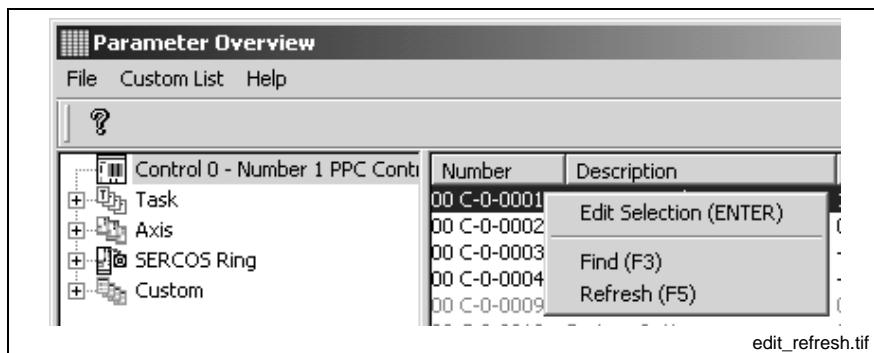


Fig. 10-30: Edit, Refresh or Find

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

Display a Parameter List

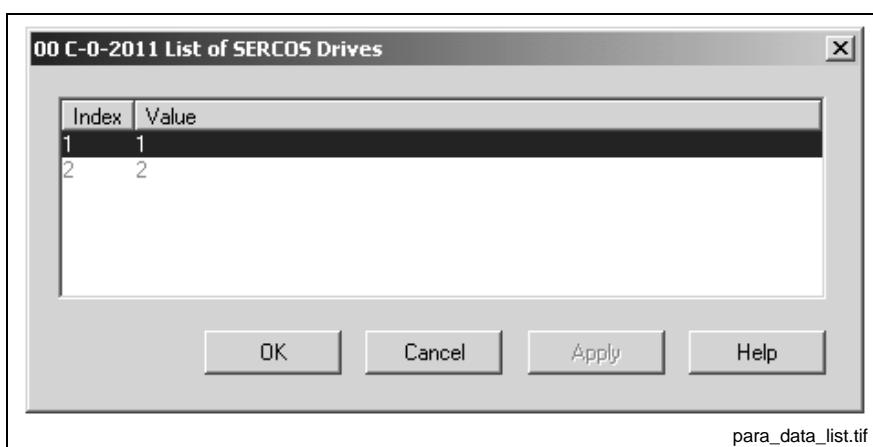
The information contained in Parameter lists can be displayed in one of the following two formats:

- 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 Fig. 10-31.



The screenshot shows a Windows-style dialog box titled "00 C-0-2011 List of SERCOS Drives". It contains a table with two columns: "Index" and "Value". The data is as follows:

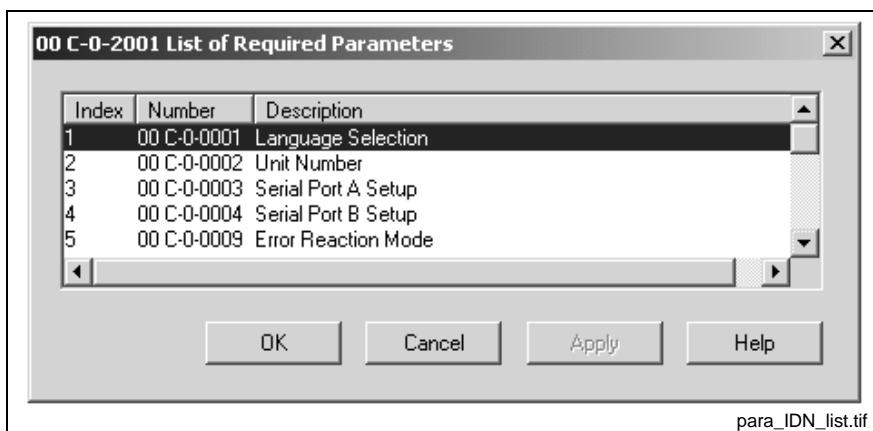
Index	Value
1	1
2	2

At the bottom of the dialog are four buttons: OK, Cancel, Apply, and Help. The file name "para_data_list.tif" is visible at the bottom right of the dialog.

Fig. 10-31: 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 Fig. 10-32.



The screenshot shows a Windows-style dialog box titled "00 C-0-2001 List of Required Parameters". It contains a table with three columns: "Index", "Number", and "Description". The data is as follows:

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

At the bottom of the dialog are four buttons: OK, Cancel, Apply, and Help. The file name "para_IDN_list.tif" is visible at the bottom right of the dialog.

Fig. 10-32: 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.

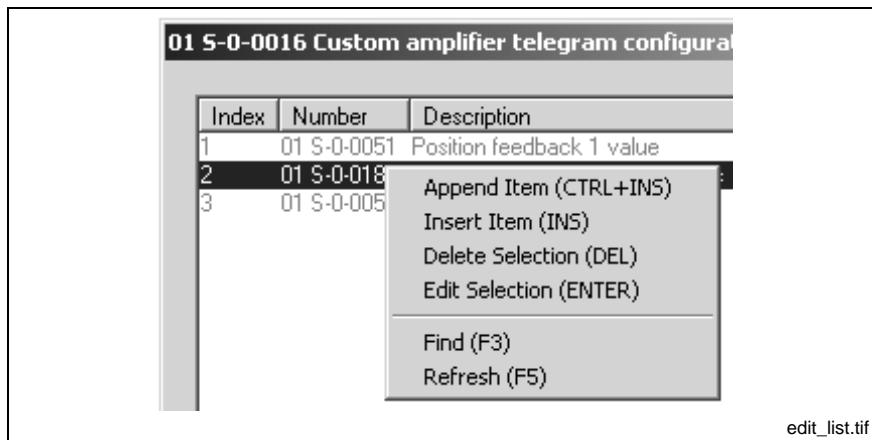


Fig. 10-33: Edit a List

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

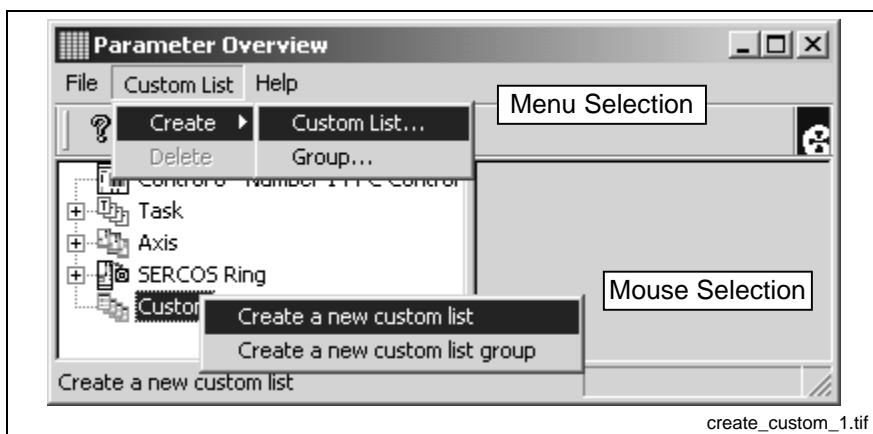
- **Append Item (CTRL+INS)** – adds a new entry to the end of the current list.
- **Insert Item (INS)** – inserts an entry above the selected value or parameter.
- **Delete Selection (DEL)** – deletes the selected data or parameter from the list.
- **Edit Selection (ENTER)** – opens an edit window where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

Create a Custom List

A custom list is a user defined grouping of parameters that can be specific to an application or machine. 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. Right click on **Custom** to open a popup window and select "Create a new custom list". The menu selection option in the figure below can also be used after Custom is selected.



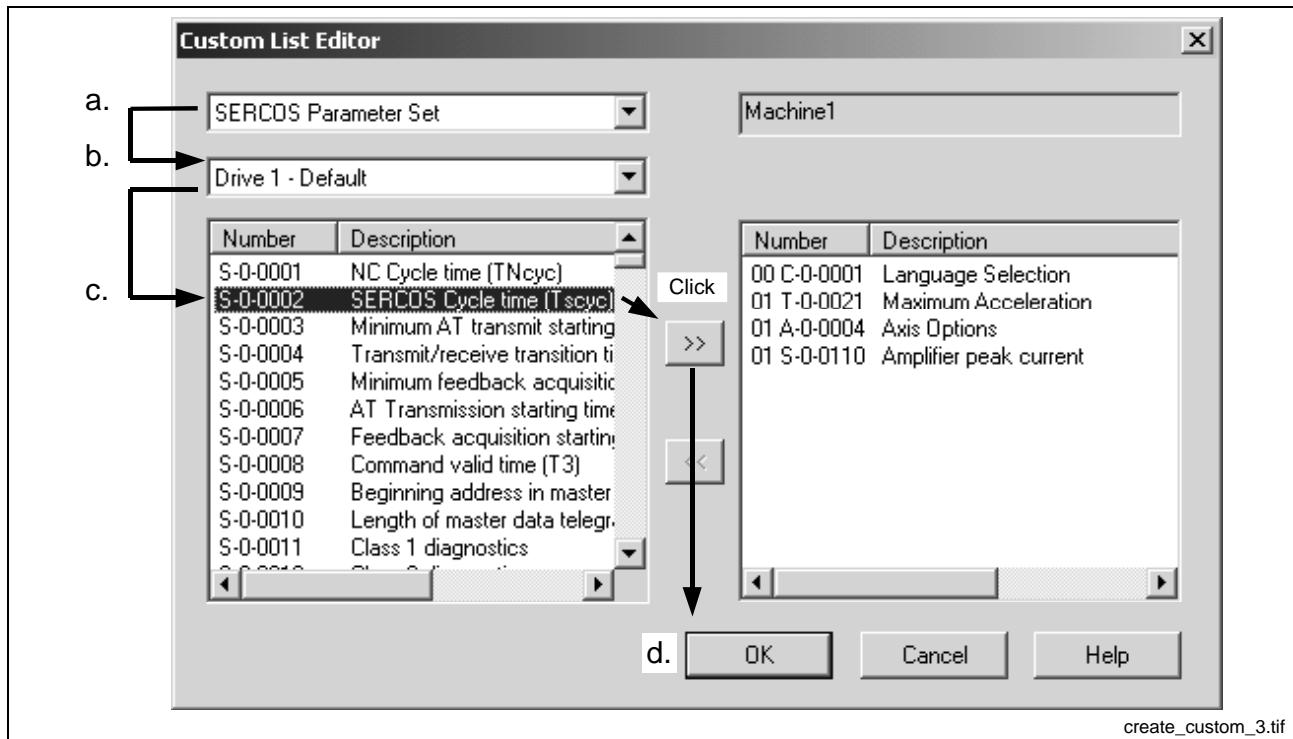
2. Enter a name for the custom list, up to a maximum of 80 characters, that identifies the list and press the **OK** button.



3. Follow the sequence below to add parameters to the custom list.

- a. Select one of the following parameter types from the drop-down list:
 - Control Parameter Set
 - Task Parameter Set
 - Axis Parameter Set
 - SERCOS Parameter Set (Drive or I/O)
- b. If applicable, select a task (A, B, C or D) for Task Parameter Set and an address number for Axis or SERCOS Parameter Sets.
- c. Select the parameter in the left window and press the insert button to add the parameter to the list. Repeat the process for the same or different parameter types until all the desired parameters are added to the list.
- d. Press the **OK** button to complete the process.

Note: To remove a parameter from the custom list, select the parameter and press the  remove button.



The newly created custom list name will appear below the Custom tree selection, as shown in Fig. 10-34.

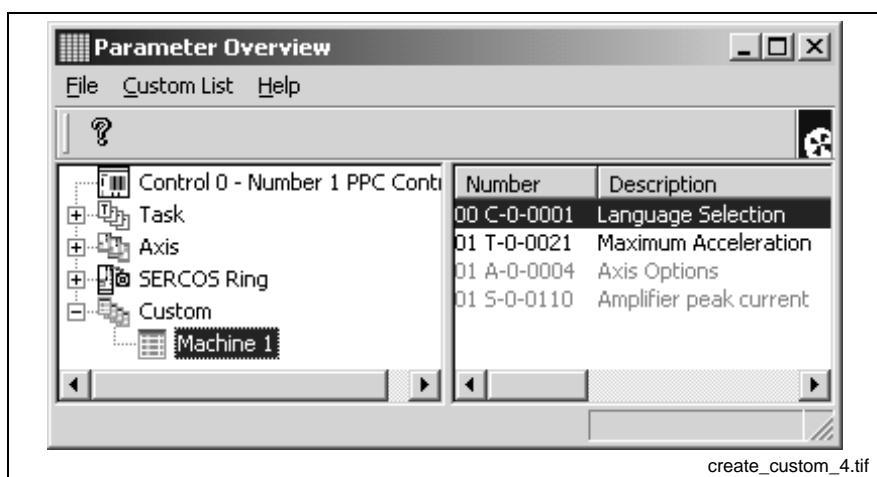


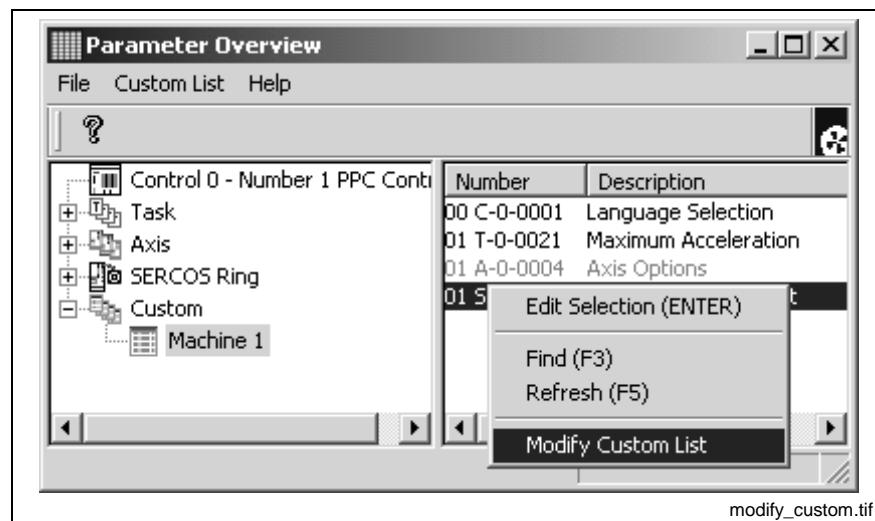
Fig. 10-34: Created Custom List

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 name 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".

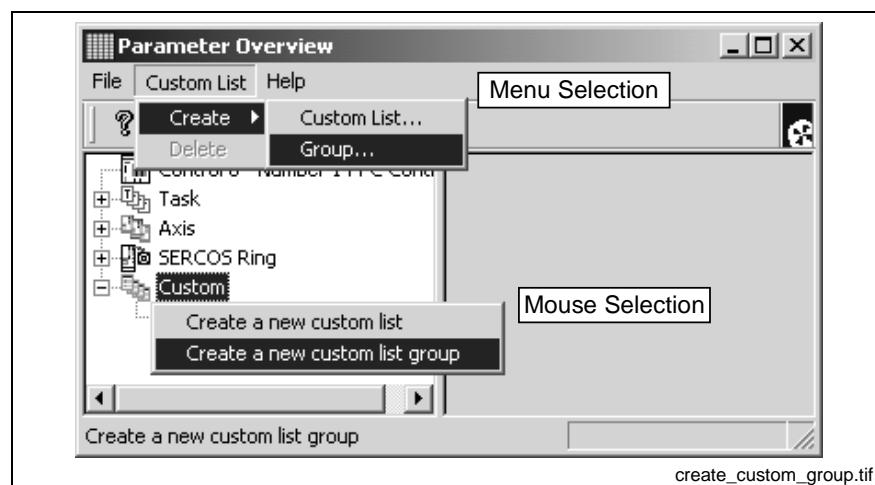
The addition and removal of parameters is described in section, "Create a Custom List" on 10-26.

Create a Custom List Group

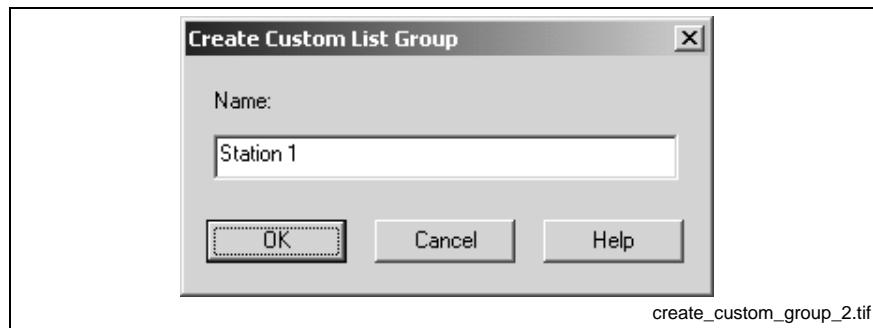
A custom list group is a folder created under the **Custom** tree selection where multiple custom list files can be grouped. This option allows the user to group or categorize different custom lists to a particular project or machine. Multiple custom list groups can be created following the previous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

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

1. Right click on **Custom** to open the popup window and select "Create a new custom list group". The "Group" menu selection option in the figure below can also be used after Custom is selected.



2. Enter a name for the custom list group, up to a maximum of 80 characters, that identifies the group and press the **OK** button.



The newly created custom list group name will appear below the Custom tree selection, as shown in Fig. 10-35.

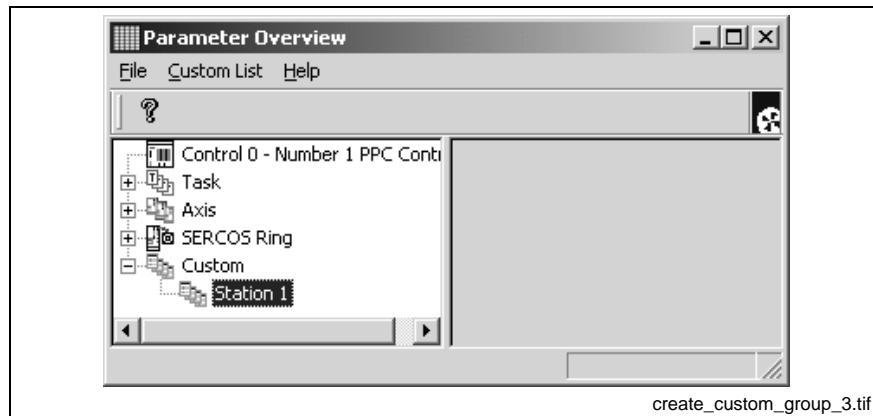


Fig. 10-35: Created Custom List Group

To add a custom list under the group name, select the group name and follow the steps in section Create a Custom List on page 10-26.

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

Synchronization

Velocity Synchronization

Velocity synchronization is used in printing machines for simple transport feeds. The drive runs with velocity synchronous to the master axis. An electronic gear presets the track speed at the circumference of the transport feed or the winder. A gear ratio fine adjust is available for use as a tension control. The gear ratio fine adjustment can be configured as cyclical data, permitting velocity changes at the slave axis while the master axis speed is constant. Modifying the master axis gear parameter can also change velocity.

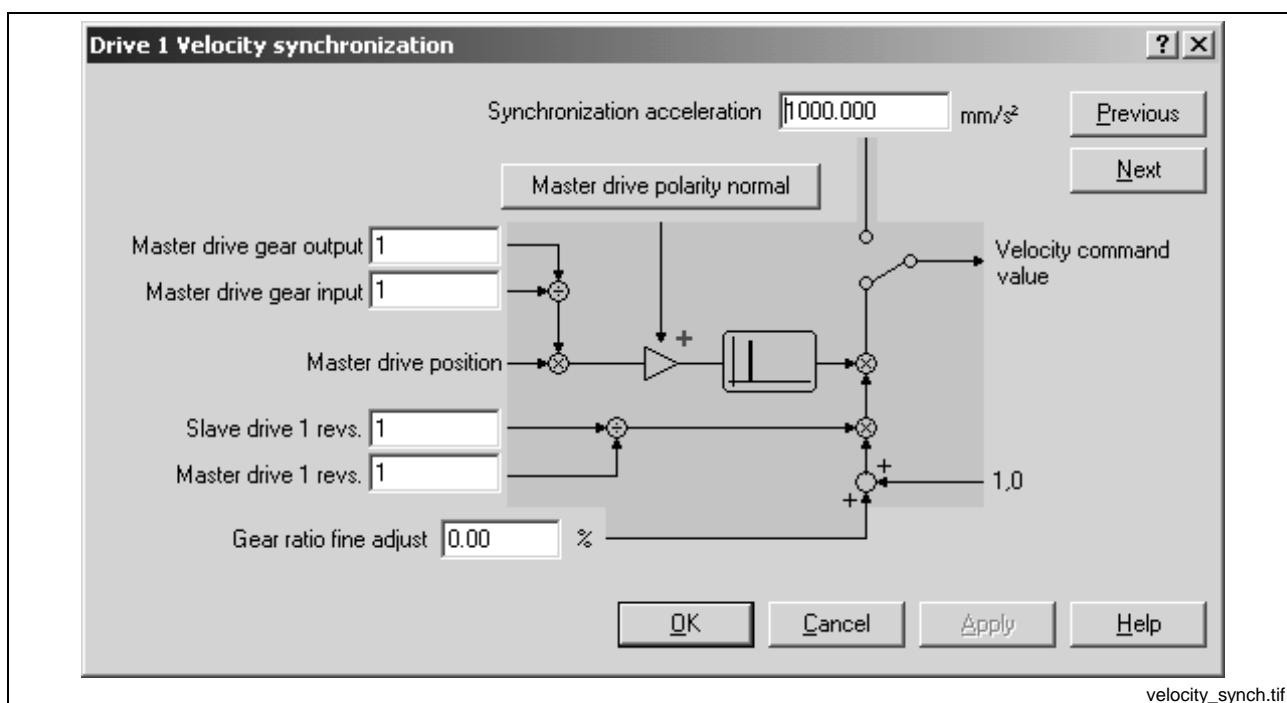


Fig. 10-36: Drive n Velocity Synchronization

Synchronization Acceleration (P-0-0142)

Maximum acceleration or deceleration that is used to dynamically synchronize the slave's output position to that of the master input position (ramp up and lock on). Acceleration and delay is performed with the synchronization acceleration in the second step of dynamic synchronization (ramp up and lock on). This affects device-operating modes with underlying position control. When running an angle offset, the slave drive is accelerated or decelerated with the synchronization acceleration.

Master Drive Polarity (P-0-0108)

This parameter inverts master drive position polarity. This means that an inverted, electronic gearbox can be implemented. Click on the button to change between Master drive polarity normal (positive) and Master drive polarity reversed (negative).

Master Drive Gear Output (P-0-0157)

This parameter together with **Master Drive Gear Input (P-0-0156)**, determines the master drive gear ratio. The output ratio of these two parameters is multiplied with the master drive position before it is sent to the drive. This 16-bit word can be modified in phase 4.

Slave Drive n Revs (S-0-0237)
Master Drive n Revs. (S-0-0236)

The output ratio of these two parameters is multiplied with the master drive position before it is sent to the drive. This is a 32 bit word that can only be modified in phase 2.

Gear Ratio Fine Adjust

The output ratio of the electronic gearbox is changed by this percentage value. This parameter is only active in velocity synchronization mode and is typically used as a tension control.

Phase Synchronization

Phase synchronization is used for machining processes that require an absolute phase synchronization of the drive's output position to the master axis input position (e.g., printing, punching or perforating in printing machines). When this mode is activated, the drive will either accelerate or decelerate to match the master's velocity. Then, dynamic synchronization is performed to match the position of the slave drive to that of the master's position set in the control.

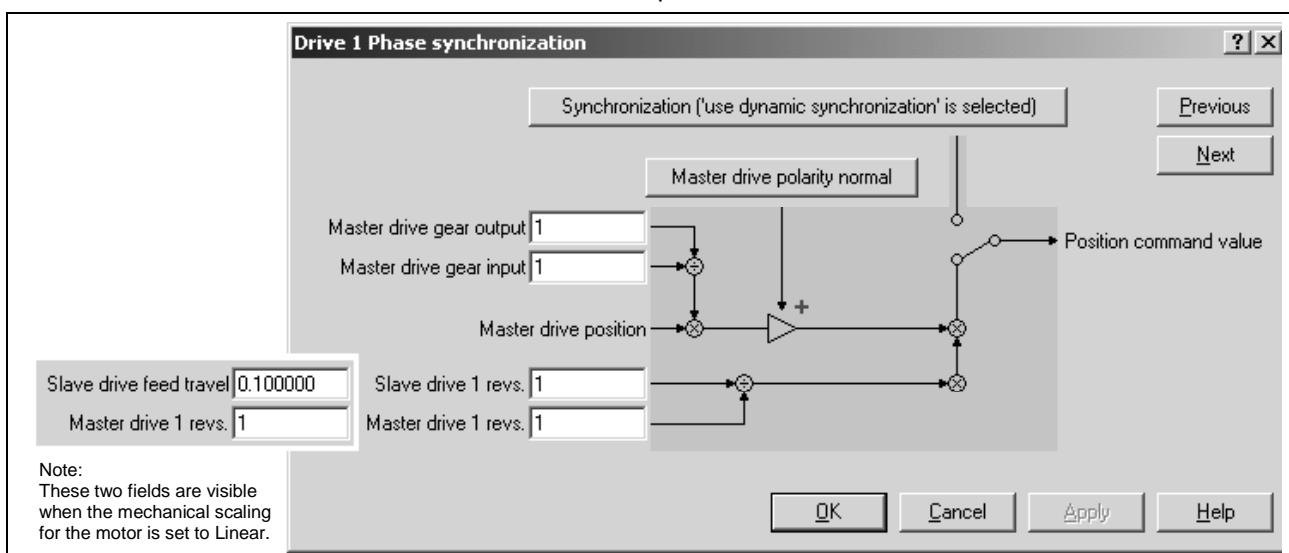


Fig. 10-37: Drive n Phase Synchronization

**Master Drive Polarity
(P-0-0108)**

This parameter inverts master drive position polarity. This means that an inverted, electronic gearbox can be implemented. Click on the button to change between Master drive polarity normal (positive) and Master drive polarity reversed (negative).

**Master Drive Gear Output
(P-0-0157)**

This parameter together with **Master Drive Gear Input (P-0-0156)** determines the master drive gear ratio. The output ratio of these two parameters is multiplied with the master drive position before it is sent to the drive. This 16-bit word can be modified in phase 4.

Slave Drive n Revs (S-0-0237)
Master Drive n Revs. (S-0-0236)

The output ratio of these two parameters is multiplied with the master drive position before it is sent to the drive. This 32-bit word can only be modified in phase 2.

**Slave Drive Feed Travel
(P-0-0159)**

During linear angle synchronization, the slave axis performs one feed per revolution of the master axis. Parameter P-0-0159, Slave drive feed travel, together with the parameter S-0-0236, Master drive 1 revs., determines the distance to go per revolution of the master axis. This parameter is visible when the mechanical scaling for the motor is set to linear under **Configure** \Rightarrow **Mechanical**.

The following parameter values are found under the Synchronization window in Fig. 10-38: Synchronization Window and are used to fine adjust the synchronization between the position command value (slave drive output position) to the master drive axis input position.

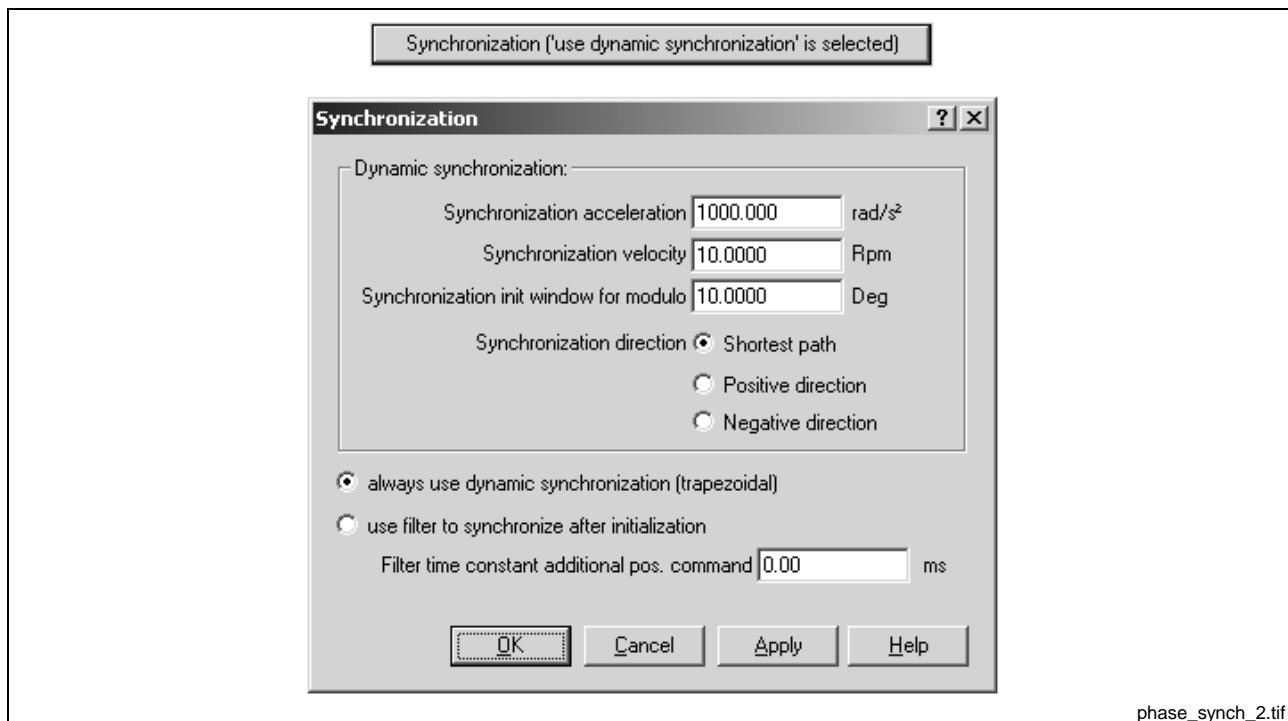


Fig. 10-38: Synchronization Window

Synchronization Acceleration (P-0-0142)	Maximum acceleration or deceleration that is used to dynamically synchronize the slave's output position to that of the master input position (ramp up and lock on). Acceleration and delay is performed with the synchronization acceleration in the second step of dynamic synchronization (ramp up and lock on). This affects device-operating modes with underlying position control. When running an angle offset, the slave drive is accelerated or decelerated with the synchronization acceleration.
Synchronization Velocity (P-0-0143)	The maximum velocity used for dynamic synchronization.
Synchronization Init Window for Modulo (P-0-0151)	The second step in dynamic synchronization (ramp up and lock on) establishes a path that must be crossed to reach absolute synchronization. If the position difference between slave and master exceeds "synchronization window in modulo format P-0-0151", then the synchronization direction is determined by parameter "synchronization direction" (P-0-0154). If the position difference is greater than that set in P-0-0151, then the selection in P-0-0154 "synchronization direction" is used.
Synchronization Direction (P-0-0154)	Synchronization will be perform in the direction specified by P-0-0154 when the synchronization position difference is greater than P-0-0151. The following direction paths are available: <ul style="list-style-type: none"> • Shortest path • Positive direction • Negative direction

Synchronization Mode (P-0-0155) The drive will start the dynamic synchronization automatically after one of the following operating modes is activated:

- phase synchronization
- cam shaft
- pattern transmission

The S-0-0047 Position Command Values, will be generated by the drive until the absolute synchronization ($S-0-0047 = XSynch + S-0-0048$) is reached. The P-0-0142, Synchronization Acceleration, and P-0-0143, Synchronization Velocity, will be taken into consideration.

The following synchronization modes will then be examined:

Always use dynamic synchronization (trapezoidal)

If synchronization mode 0 is set, a path will be created after every change of the position command value based on the following (phase adjusting) equation:

$$\text{Path} = XSynch + S-0-0048 - S-0-0047$$

In addition, the path will be taken with regard to the synchronization acceleration and velocity.

Use filter to synchronize after initialization

If synchronization mode 1 is set, parameters P-0-0142 and P-0-0143 will be inoperative after absolute synchronization is reached. The following changes of the S-0-0048, Position Command Value Additional, will then be smoothed through a filter of the first order. The time constant for the filter will be set by parameter P-0-0060, Filter Time Constant Additional Position Command.

Filter Time Constant Additional Position (P-0-0060) If P-0-0155 is set to "Always use dynamic synchronization (trapezoidal)", then dynamic synchronization will be switched off after absolute synchronization is reached for the first time. Changes to the S-0-0048, additional position command value will be smoothed with a filter of the first order. The time constant of the filter can be set with this parameter.

Electronic Cam Shaft

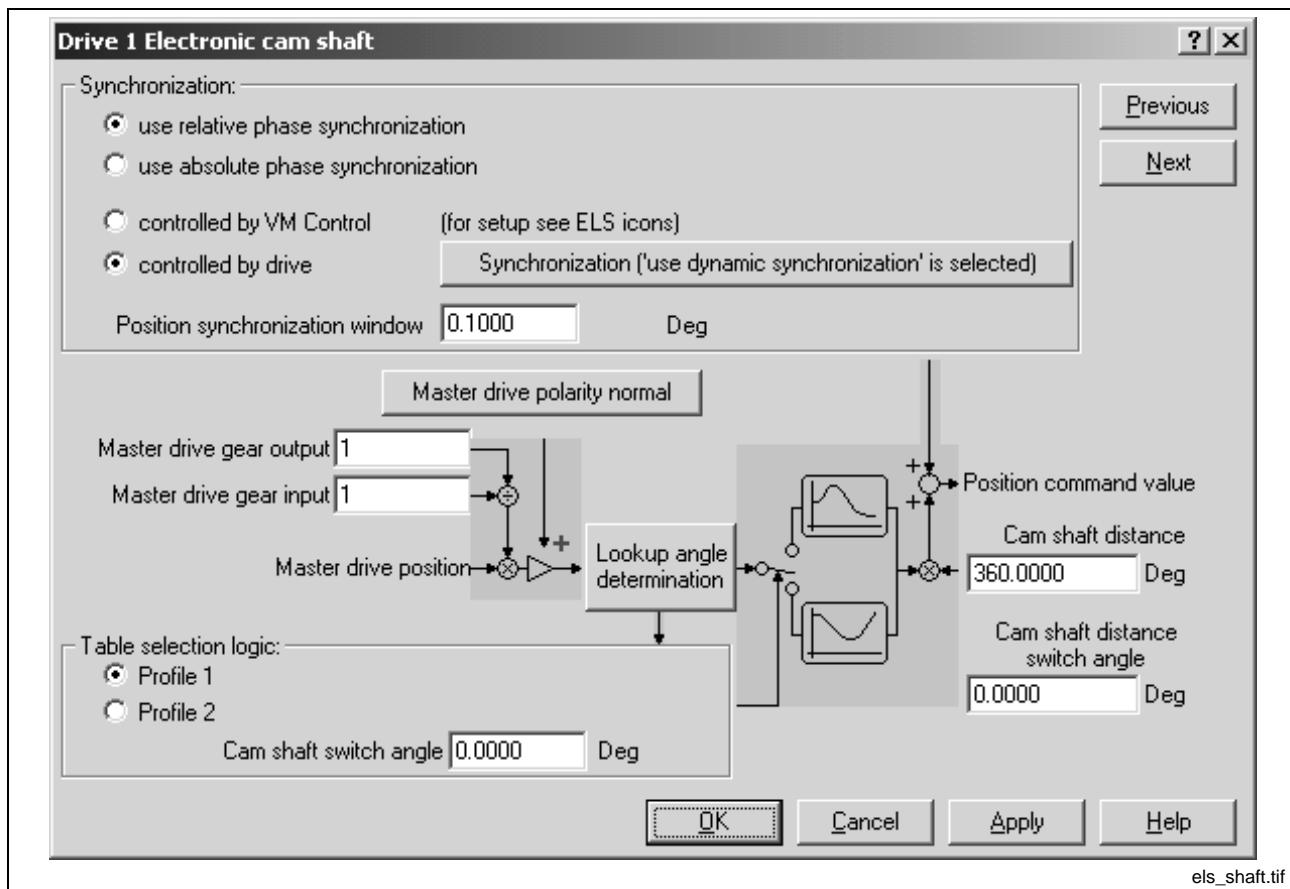


Fig. 10-39: Drive n Electronic Cam Shaft

The selections in the Synchronization area determine the type of phase offset at synchronization, the device that will generate the phase offset and the settings for dynamic synchronization.

**Use Relative Phase Synchronization
(A-0-0164, Bit 6=0)**

When relative phase synchronization is enabled, the master and slave positions are synchronized and the relative phase offset difference is written to parameter P-0-0151.

**Use Absolute Phase Synchronization
(A-0-0164, Bit 6=1)**

When absolute phase synchronization is enabled, the phase offset value in parameter A-0-0151 is used to set a phase offset between the master and slave positions.

**Generated by the Control
(A-0-0164, Bit 1=0)**

When selected, the phase offset is generated by the control.

**Generated by the Drive
(A-0-0164, Bit 1=1)**

When selected, the phase offset is generated by the drive.

Position Synchronization Window (S-0-0228)

If the difference between the position command value and the feedback value is smaller than the synchronization window during the parameterized synchronization operating mode with underlying position control, then bit 8 in the S-0-0182, Manufacturer Class 3 Status will be set.

Refer to Fig. 10-38 for details on dynamic synchronization settings.

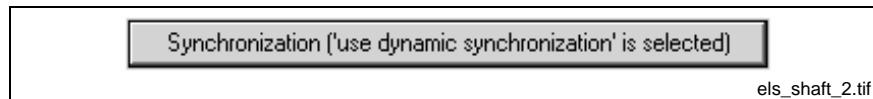


Fig. 10-40: Synchronization for Electronic Cam Shaft

**Master Drive Polarity
(P-0-0108)**

This parameter inverts master drive position polarity. This means that an inverted, electronic gearbox can be implemented. Click on the button to change between Master drive polarity normal (positive) and Master drive polarity reversed (negative).

**Master Drive Gear Output
(P-0-0157)**

This parameter together with **Master Drive Gear Input (P-0-0156)** determines the master drive gear ratio. The output ratio of these two parameters is multiplied with the master drive position before it is sent to the drive. This 16-bit word can be modified in phase 4.

**Slave Drive n Revs (S-0-0237)
Master Drive n Revs. (S-0-0236)**

The output ratio of these two parameters is multiplied with the master drive position before it is sent to the drive. This 32-bit word can only be modified in phase 2.

**Slave Drive Feed Travel
(P-0-0159)**

During linear angle synchronization, the slave axis performs one feed per revolution of the master axis. Parameter P-0-0159, Slave drive feed travel, together with the parameter S-0-0236, Master drive 1 revs., determines the distance to go per revolution of the master axis. This parameter is visible when the mechanical scaling for the motor is set to linear under **Configure ⇒ Mechanical**.

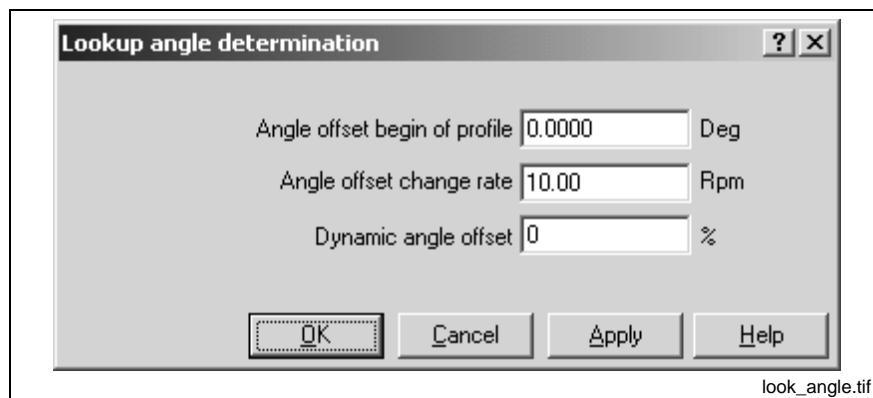


Fig. 10-41: Lookup angle Determination

**Angle Offset Begin of Profile
(P-0-0061)**

The profile (table) will be shifted by this angle in relation to the Master drive position.

The offset is used in the cam shaft or pattern transmission operating modes (Master Phase Adjust).

**Angle Offset Change Rate
(P-0-0158)**

In Cam shaft or pattern transmission operating modes, P-0-0061 affects the table access angle. To avoid jumps of the table access angle; a new value for P-0-0061 does not immediately become effective. Starting with the current value, a ramp-like approximation of the new value is performed. The approximation is performed along the shortest path. The ramp is set in P-0-0158.

**Dynamical Angle Offset
(P-0-0085)**

This parameter is used to compensate a lag error if the position controller has not set to lag free control. The table access angle is set prematurely and is velocity dependent.

Table Selection Logic Profiles (P-0-0088)	Each drive controller is capable of containing two internal cam tables. This parameter selects which cam table profile to use when the cam shaft switch angle is encountered. Profile 1 relates to the cam table profile in parameter P-0-0072 and profile 2 relates to the cam table profile in parameter P-0-0092.
Cam Shaft Switch Angle (P-0-0094)	If the Master drive position passes this angle in a positive or negative direction, then a switch will be made to the cam-profile table that was pre-selected by parameter P-0-0088, Cam Shaft Control. Parameter P-0-0089, Cam Shaft Status will be set to the activated cam profile table. When the drive is first initialized, the cam profile set in P-0-0088 will be activated. Parameter P-0-0089 also will be set.
Cam Shaft Distance (P-0-0093)	This parameter determines the factor with which the cam profile will be multiplied.
Cam Shaft Distance Switch Angle (P-0-0144)	A new value for the P-0-0093, Cam Shaft Distance will become active only when the table access angle passes the cam shaft switch angle. The angle for the table access is derived out of the following parameters: <ul style="list-style-type: none">• P-0-0053, Master drive position• P-0-0061, Angle offset begin of profile• P-0-0085, Dynamical angle offset• P-0-0108, Master drive polarity• P-0-0156, Master drive gear input revolutions• P-0-0157, Master drive gear output revolutions• P-0-0158, Angle offset change rate This works only in a curve pattern disk (Cam profile) operating mode.

10.4 Oscilloscope

The oscilloscope utility is used to capture and display run-time data. The capture can be of the control or on a drive that supports this feature. Selected data is acquired on the drive or control, passed to VisualMotion Toolkit, and displayed on the graphical format. The graphical display and supporting data can be printed, or the data can be saved to a file for later review.

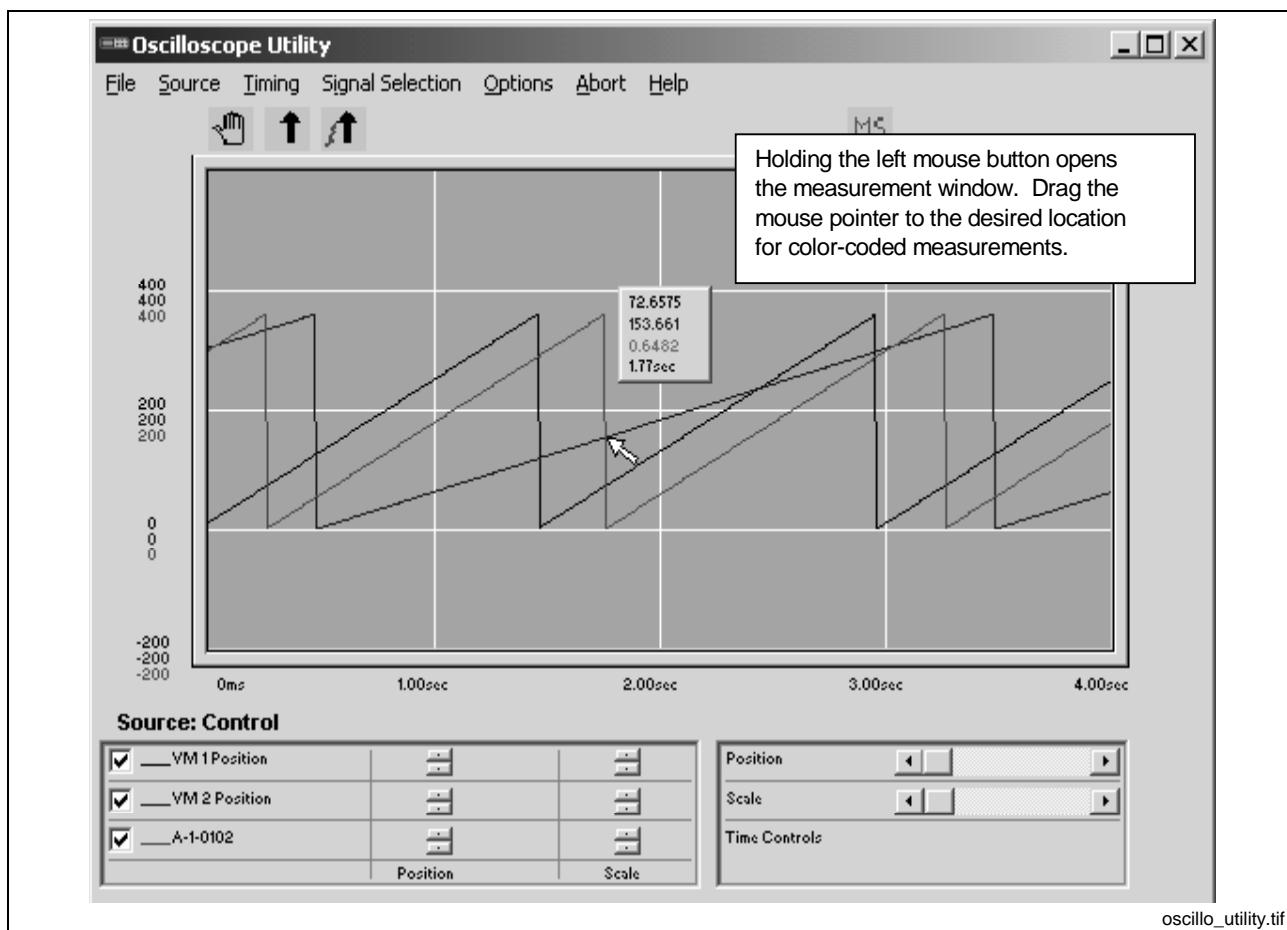


Fig. 10-42: Oscilloscope Utility

File Menu

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

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

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

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

Exit - terminates this utility

Source Menu

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

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

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

Timing

The Oscilloscope timing options in Fig. 10-43 are used for setting the **Sampling Rate** (How often a trace is captured) and **Sample Count** (How many sampling rates are captured). The **Capture Duration** field displays the total capture duration that is calculated by multiplying the Sample Count and Sampling Rate. A **Pretrigger** can be added and is a percentage of the capture interval.

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

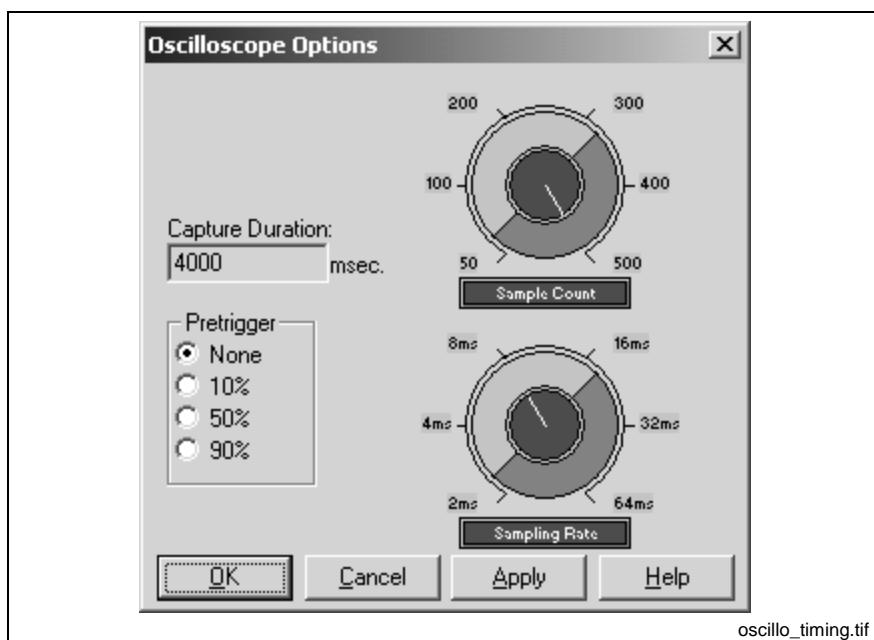


Fig. 10-43: Oscilloscope Options

Signal Selection

The Drive Signal Setup in Fig. 10-44 is available when a drive is selected under the Source menu.

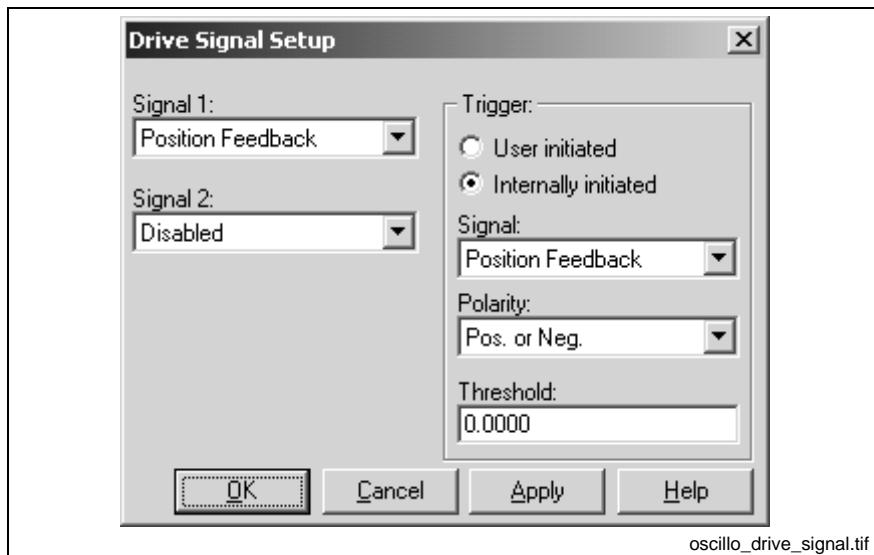


Fig. 10-44: Drive Signal Selection

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

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

The Control Signal Setup in Fig. 10-45 is available when a Control is selected under the Source menu.

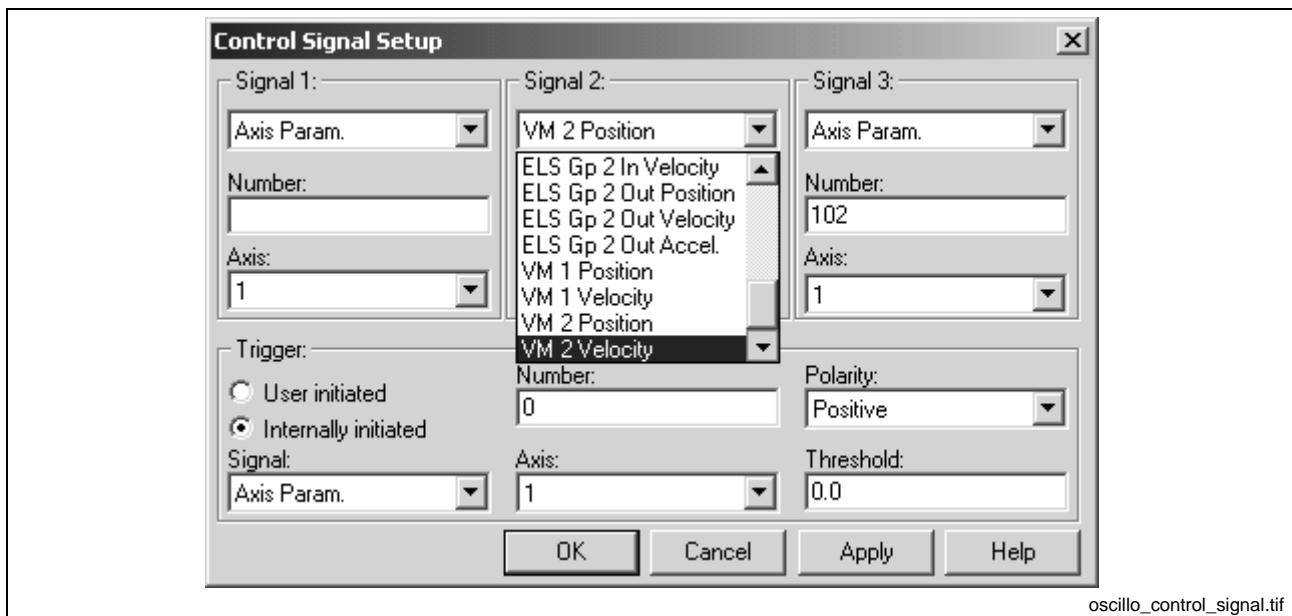


Fig. 10-45: Control Signal Selection

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

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

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

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

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

For **Internally initiated** captures, the available signals are the same as the signals for Signals 1 – 3. The heading in the trigger fields will change based on the signal selected. The trigger polarity options are on positive edge, negative edge, or both. Signal threshold is the signal level to trigger.

Options Menu

From the options menu, a trace's appearance can be changed from Lines to Dots. When combined with the Time Controls feature on page 10-44, the user can scale (zoom) in to reveal the individual dots that makeup the trace.

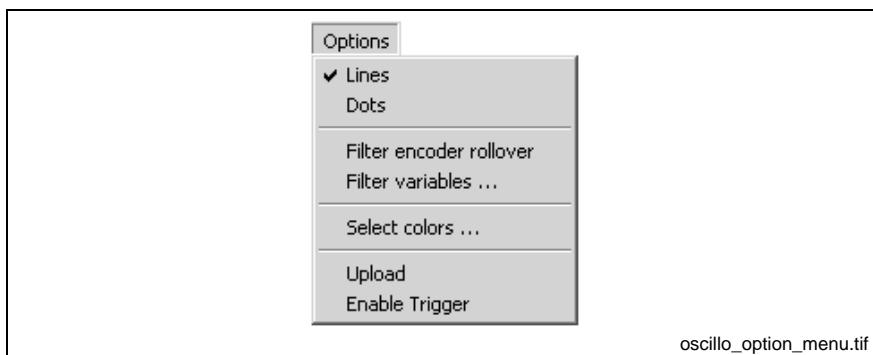


Fig. 10-46: Options Menu

Filter encoder rollover

The position control loop in servo systems is continuously correcting the position of an encoder when at standstill. This continuous correction in position can cause dithering that will be captured by the oscilloscope. When selected, the Filter encoder rollover will eliminate any dithering based on the settings of the Filter variables ... window in Fig. 10-47.

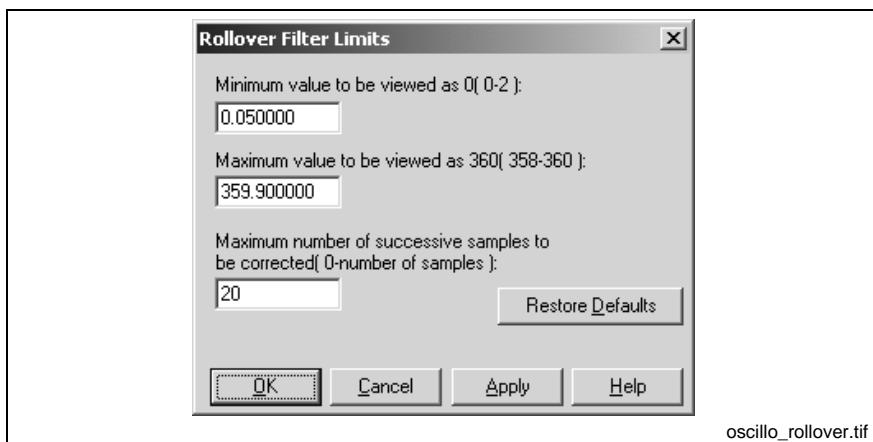


Fig. 10-47: Filter variables

Minimum value to be viewed as 0

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

Maximum value to be viewed as 360

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

The oscilloscope will immediately capture any position value outside the minimum or maximum filter windows. An example of the filter encoder rollover is shown in Fig. 10-48.

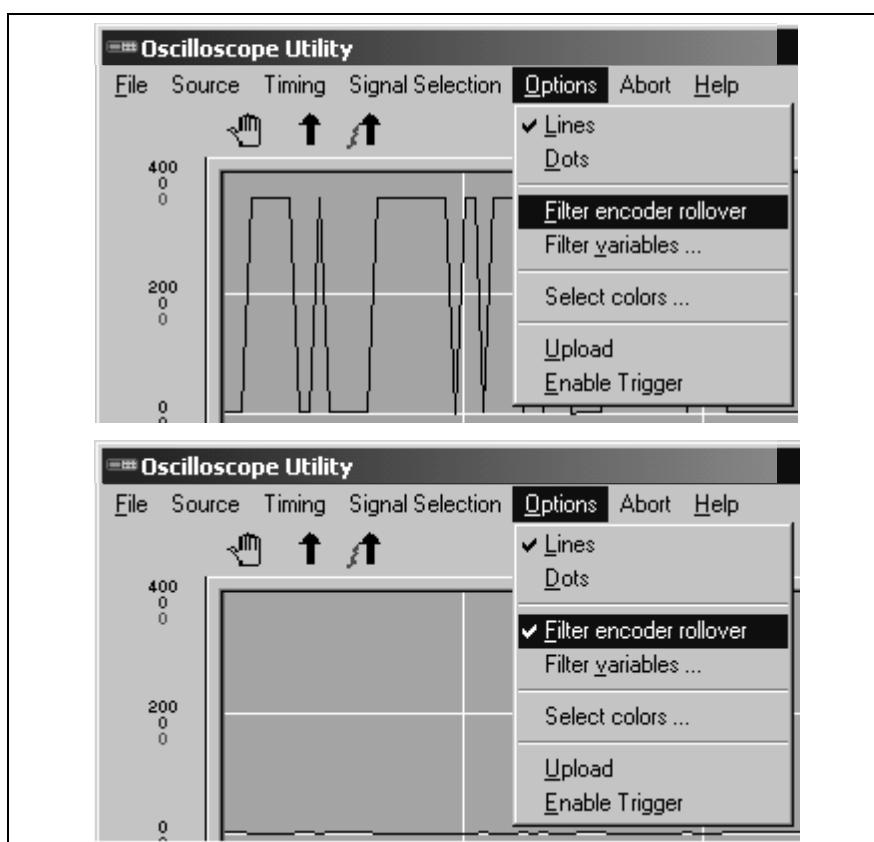


Fig. 10-48: Filter encoder rollover

Select colors...

The three available signal traces are color coded and can be changed for both run time and memory by selecting a color for each trace in the Signal color selection window in Fig. 10-49.

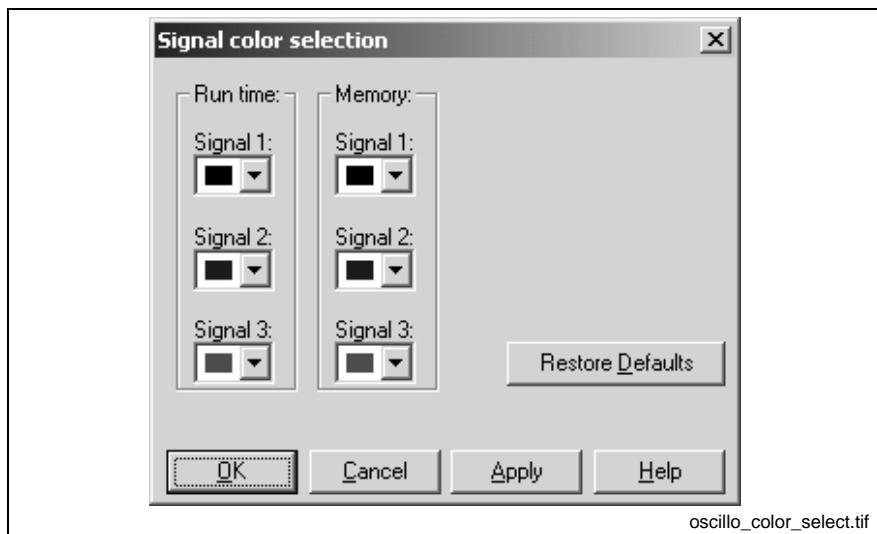


Fig. 10-49: Signal color selection

Abort, Upload and Enable Trigger

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

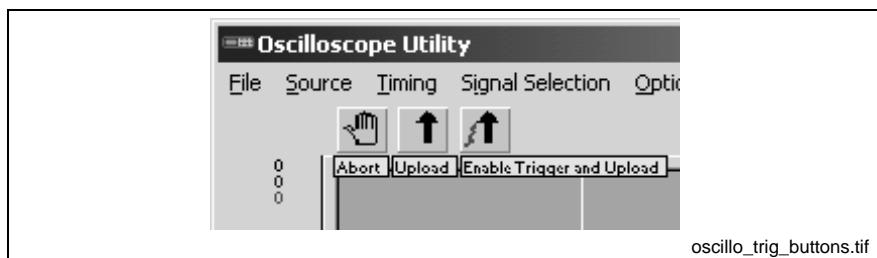


Fig. 10-50: Signal triggering buttons

Oscilloscope memory buttons

Using memory buttons, traces can be stored into memory for viewing and comparing.

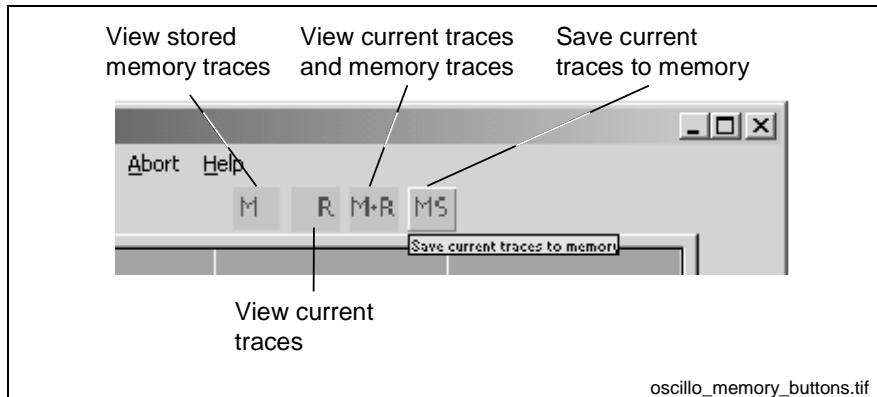


Fig. 10-51: Oscilloscope memory buttons

Manipulating trace signals

When multiple traces are captured at one time, they can be positioned and scaled independently of each other by using the up and down arrows for each trace as shown in Fig. 10-52. The traces can also be turned on or off by clicking on the check boxes to the right of the signal description.

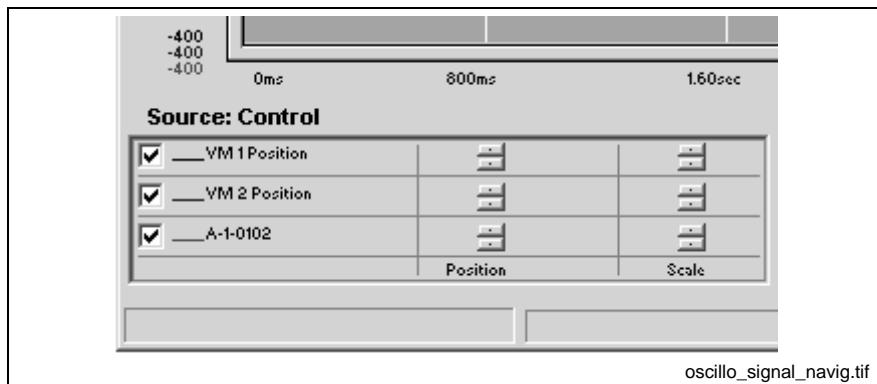


Fig. 10-52: Manipulating trace signal

By positioning the traces above and below each other, the user can more easily distinguish between the signal. Fig. 10-53 shows an example of three traces repositioned for clarity.

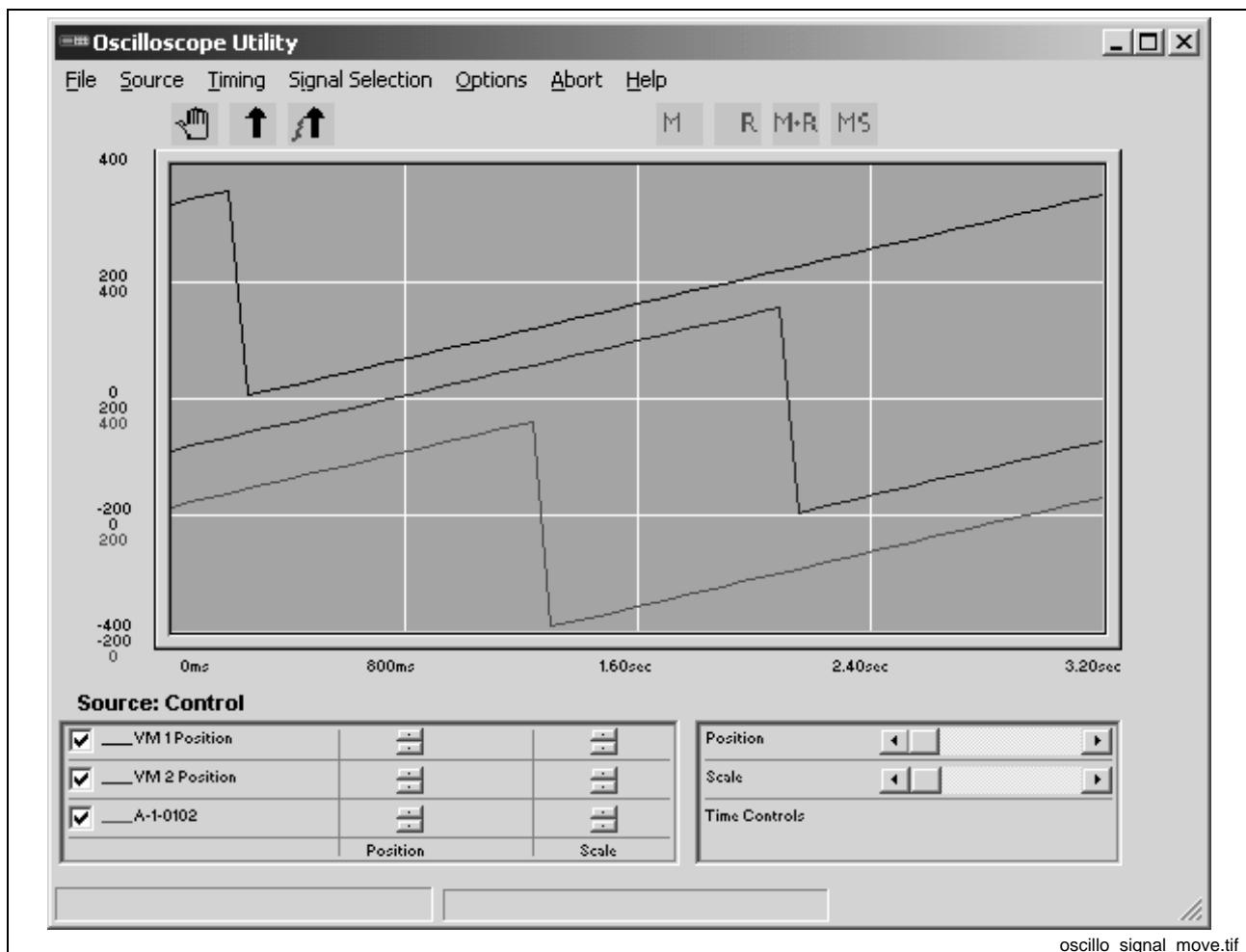


Fig. 10-53: Separated oscilloscope traces

Time Controls

The position and scale functions in Fig. 10-54 are used to more closely analyze a specific area of a captured trace. The scale function acts as a zoom, allowing the user to view smaller sections of a trace, while the position function controls the horizontal scrolling.

Note: The position function only works after using scale.

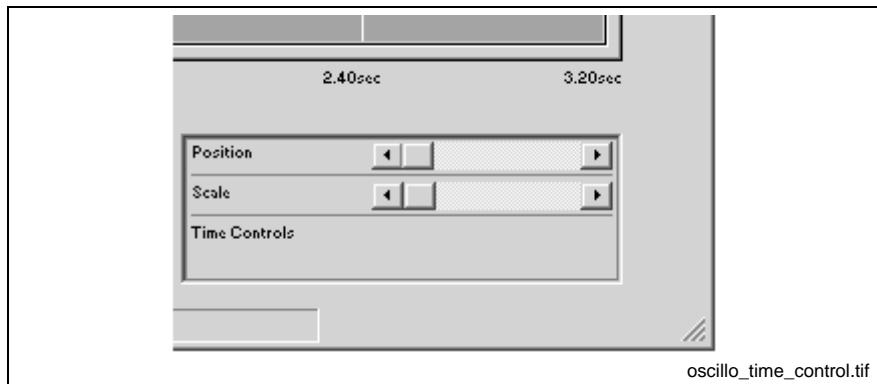


Fig. 10-54: Time Controls

11 Programming Concepts

11.1 Overview

VisualMotion provides two methods for creating user programs. The first is an icon-programming environment that allows the user to place predefined icon instructions within a workspace. The second is a text-based programming language using a command syntax which combines Basic and Assembly languages. Refer to the *Textual Language Programming* chapter in the *VisualMotion 8 Functional Description* for details.

This chapter will describe the following major programming functions available in VisualMotion:

- Events

Programming Environment

VisualMotion provides up to four user-defined multi-tasking motion control tasks for up to 32 axes. The programming environment provides the user with different workspaces for main motion task and sub-functions, such as subroutines and events. VisualMotion's motion tasks are named Task A through Task D. All four motion tasks are executed at the same time and are multi-tasked, starting with Task A.

Instruction Execution

Normal multi-tasking is achieved by consecutively executing one icon instruction from each task. VisualMotion's multi-tasking path planner and the use of Indramat intelligent digital drives permit simultaneous execution of multiple motion commands.

11.2 Events

VisualMotion events are privileged forms of subroutines. When an event is running, all program motion will continue but the user programming in the tasks is suspended until it the event has finished. Events provide an efficient method for executing certain functions within an application without constantly monitoring for conditions in a user program.

An event consists of two parts:

- Event Type – the condition that triggers an event
- Event Function – the subroutine that runs when the event Type is triggered

Important Note:

Because an event preempts all user tasks, they can adversely effect task response time and execution. Events should be used only when a real-time response is needed and should take minimal time to finish. For example, involved calculations or waiting for other I/O will block the execution of all tasks that are associated with that event. If possible, use the event routine to set a flag value in an integer variable to accelerate a return to the task. Then, during an idle period, test the flag and perform required operations. Reset the flag after testing and operations are complete.

Events in VisualMotion will temporarily interrupt task execution. Events occur during the execution of their assigned task and preempt lower

priority tasks and events. VisualMotion's priorities associated with user programs are listed below.

Highest	Priority 1 - Path Planner and single axis events Priority 2 - Event from Task A Priority 3 - Event from Task B Priority 4 - Event from Task C Priority 5 - Event from Task D Priority 6 - Timer Task (repeating events)
Lowest	Priority 7 - User Motion Tasks A, B, C, D and BTC06

General Information

When creating a VisualMotion program with events, the following conditions should be considered:

- Events interrupt user tasks
- Events (except the repeating timer event) are assigned to a user task
- Each user task has a separate event queue, which can store up to 25 events
- The repeating timer event queue is separate from user task queues and can store up to 16 events

Note: An event queue (stack) is a storage area where events are placed as they are triggered. The events are accumulated in the queue and are activated in the order in which they are received. Once an event is done, it is cleared from the queue and the remaining queues move up. If the maximum number of event queues is exceeded, a Stack Overflow error will be issued by VisualMotion.

- A higher priority event interrupts a lower priority event. For example, events associated with Task A will interrupt execution of events associated with Tasks B, C, & D. Events associated with Task B will interrupt events associated with Tasks C & D, and so on.
- The queue handles events with the same priority in a **First In First Out (FIFO)** order.
- The PPC-R X1 input events are considered “fast input events” and will always be placed at the top of the FIFO queue.

Event Setup

Before an event is created in a VisualMotion program, it must first be declared in the Size icon. Enter the number of events that will be used in the VisualMotion program within the Events field of the Size icon.

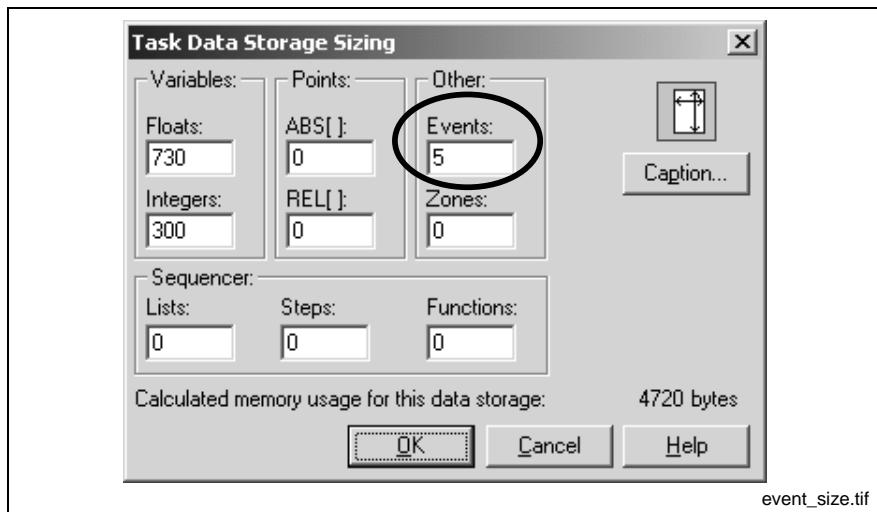


Fig. 11-1: Size Icon

Event functions are added by selecting **Edit** \Rightarrow **Add Event Function** from the VisualMotion main menu.

Note: The number of events typed in the Size icon can not be changed after the user program has been downloaded. A program must be offline to change the number of events and then recompiled.

From the *Event Function Control Block* window, enter a name for the event and click the OK button.

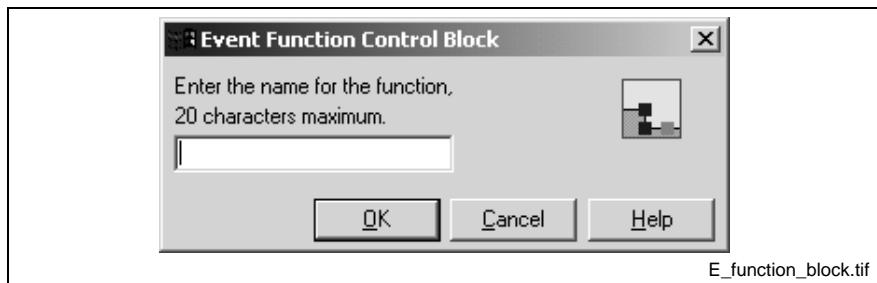


Fig. 11-2: Event Function Control Block

After the event name is entered, a new icon programming workspace opens containing a *Start* and *Finish* icon. Create the event subroutine that will run when the condition in the main program is true. Any VisualMotion icon that executes during run-time can be used in the event function subroutine.

Note: Icon functions that require a long time to process, such as the Wait, Branch, and Parameter Transfer icon functions, should be avoided in event programs.

Creating New Events Off-Line

The Calc icon can be used to declare the event. In the Calc icon, events are setup and initialized with operational values as part of the program flow. When the running program reaches the Calc icon, the values in the icon are written to the Events table.

Events are configured in the Calc icon with the following equation:

EVT[event number] = {Status, Type, Direction, Argument, Function, "Message"}

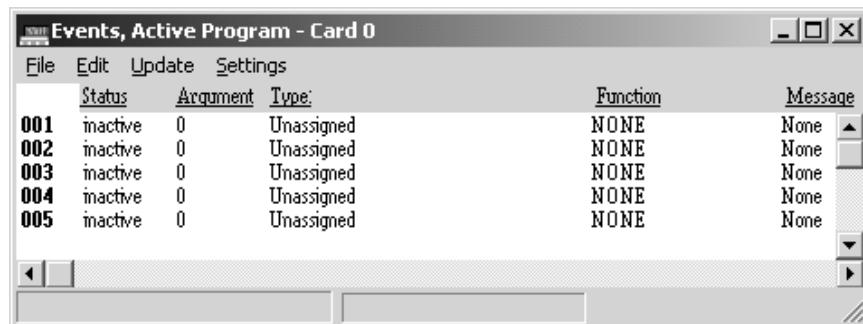
The equation can be defined with the data listed in the following table.

Subclass	Selections	Type	Access
A - Argument	A numeric value: (milliseconds if timed event) (% of the segment distance if coordinated motion) (degrees 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

Table 11-1: Event Table Data

Refer to the **Calc** icon description in the *VisualMotion 8 Functional Description* for additional information.

Note: An on-line method of event setup uses the Events table in place of the Calc icon. This method can only be used if the program has been compiled and downloaded and the program is on-line. After the user program is compiled and downloaded, the event type, event function, and any additional information is specified in the event table. The event table is launched by selecting **On-Line Data** ⇒ **Events** from VisualMotion's main menu. The number of available events in the table is generated from the number entered in the **Size** icon. Double-clicking an event number opens the *Edit Event # Values* window.



When event types are setup and initialized using the Events table, the steps must be repeated before activation every time the program is compiled and downloaded to the control.

Transferring Existing Events

An event can be transferred from an existing program into a new user program by using three different methods:

- Parameter Transfer Tool
- Data Transfer function
- Restore archive program function

Note: Only Events table information is transferred during an event transfer. The VisualMotion icons associated with the event are not transferred and must be created in the target program.

It is important to note that transferring an event into a user program that already contains events will result in the original event or events being overwritten. Events are overwritten in the order in which they appear in the events table. When a greater number of events are transferred into a program than are declared, only the matching number of transferred events will be accepted in the program. The remainder of the events will be lost. When transferring events into a program with events, be sure that you want to replace the existing event or events and that enough events have been declared to accommodate incoming events.

Parameter Transfer Tool

The Parameter Transfer tool allows you to select individual events in a user program to transfer. The event is saved as a file that you select to download into your program.

Note: Only events in a program that have been previously downloaded to the control are available for transfer. An event must be listed in the Events table to be available for selection in the parameter transfer tool.

To transfer an event from another program into your current program:

1. Select Transfer⇒Events from the Commission menu in VisualMotion.

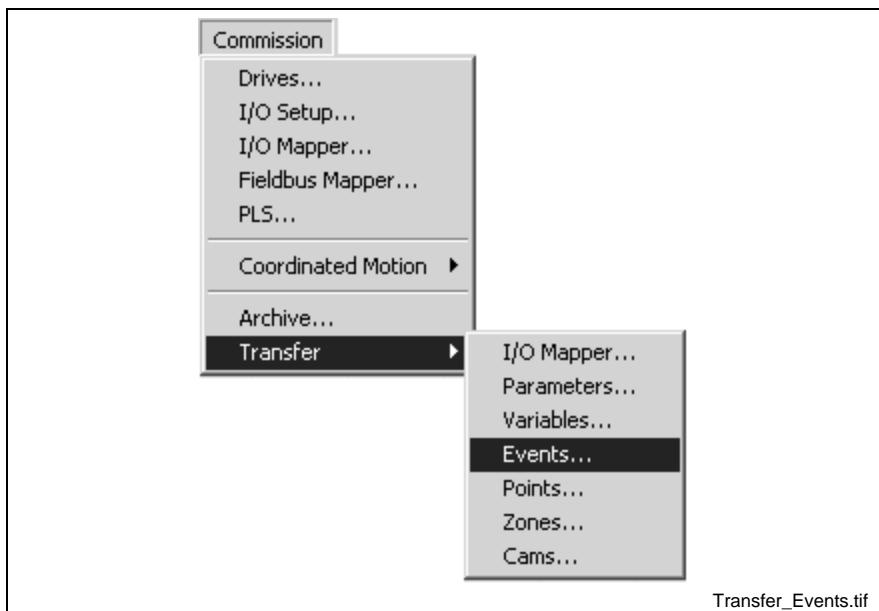


Fig. 11-3: Opening Events Transfer Tool

2. In the Transfer Event Table window, select upload and the program containing the event you want to transfer.

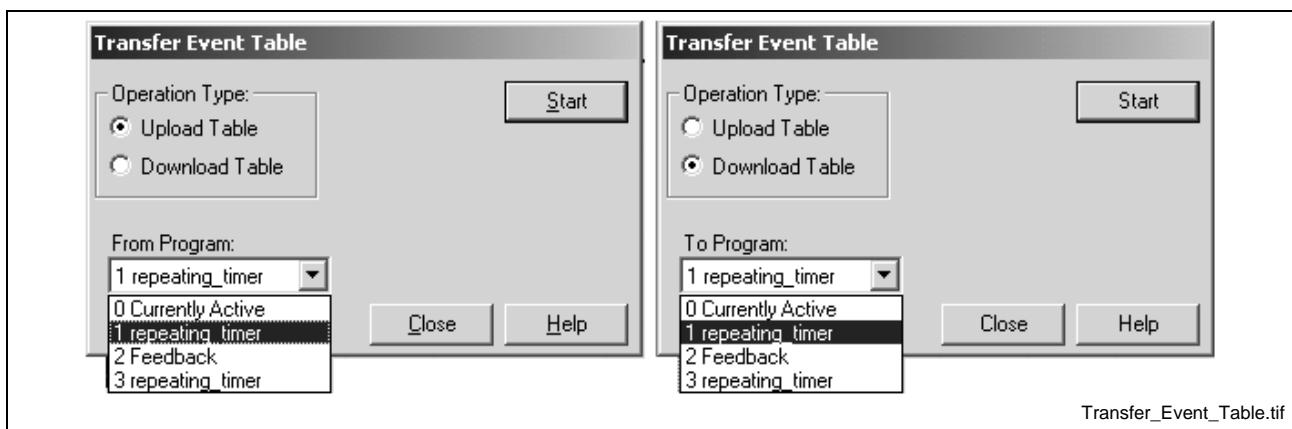


Fig. 11-4: Transfer Event Table Window

3. Select the folder where you want to save the event file and click Save.
4. Select Download in the Transfer Event Table and the destination program.
5. Select the event you are transferring from the folder where it was saved and click Open.

The transferred event will now be listed as the first event in the Events table of your current program.

Data Transfer

Data Transfer is available in the Program Management window. The data transfer process is designed for transferring all the data from one program to another. To transfer the events of a program, it is necessary to use the parameter transfer tool. To transfer program data:

1. Select Program Management from the Build menu in VisualMotion.

Note: Data transfer can only take place between two programs that have been previously downloaded to the control.

2. Select your destination program and click Data Transfer.
3. In the Data Transfer window, click Selective Transfer.

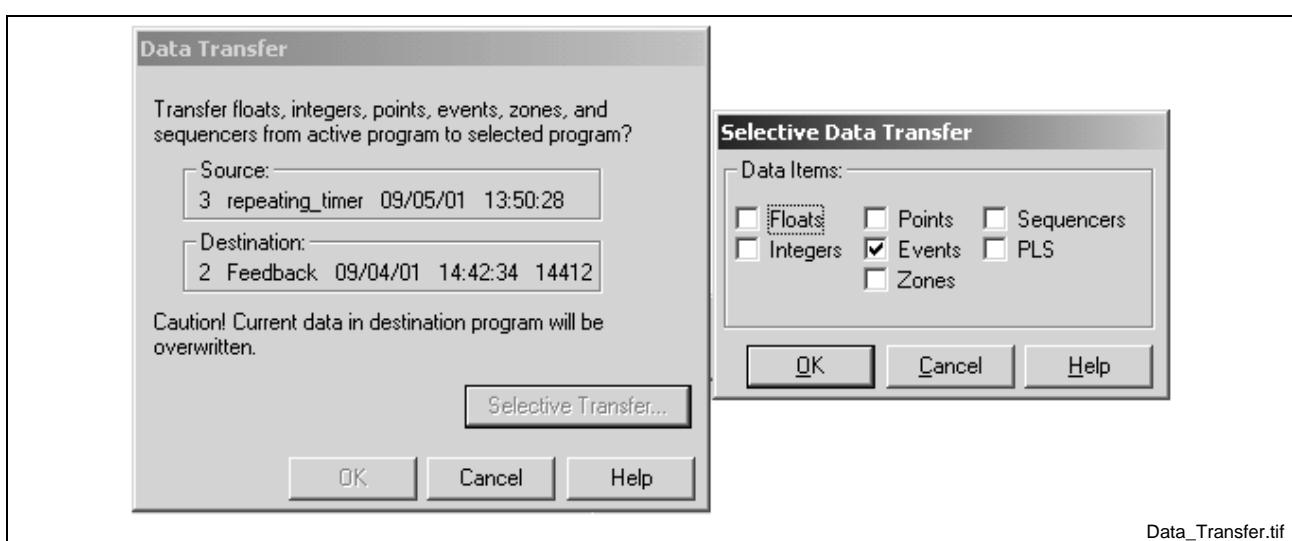


Fig. 11-5: Data Transfer of Events

4. Select Events in the Selective Data Transfer window.

Restoring Archived Program

Events can be imported into a new program from an archived program. Events can not be restored separately from their associated program. The program must first be restored and then events can be transferred to your current program using the Parameter Transfer tool. To restore an archived program:

1. Select Archive from the Commission menu in VisualMotion.
2. In the Archive window, select Restore, Selective, Programs, and the file where the archived program is located. Click the Advanced button.

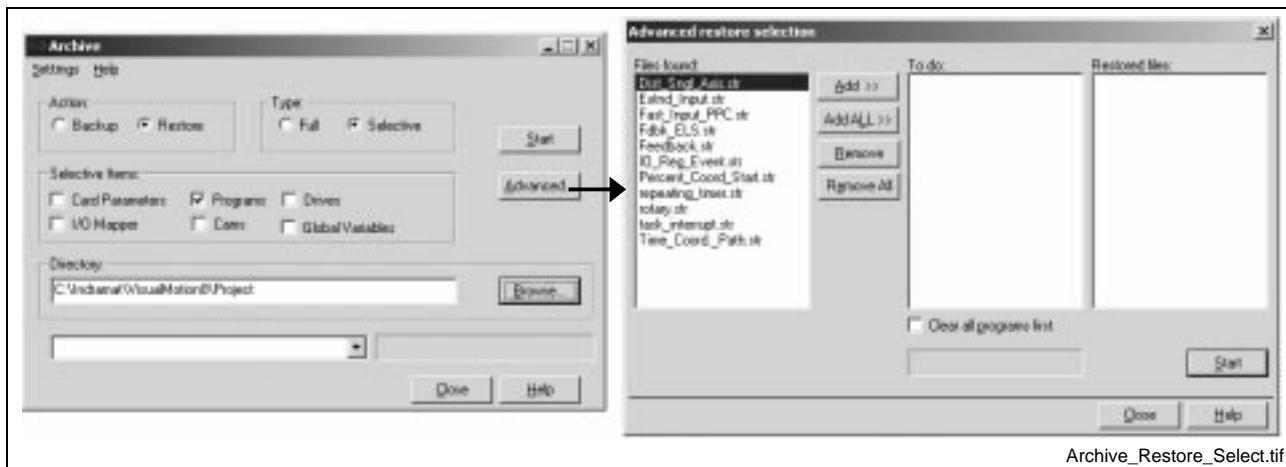


Fig. 11-6: Restore Archived Program

3. Select the archived file and click Add to add it to the To do field. Click the Start button.

Once the archived program has been restored, you can transfer events from that program using the Parameter Transfer tool.

Note: If an archived program has never been active, it will need to be downloaded to the control after it has been restored.

Event Arming/Disarming

Events must be armed to enable them. Depending on the event type, the event may be armed with the Event icon or within the icon used to configure the program motion, such as the Path and Move icons.

With the exception of the repeating rotary and repeating timer events, which have an autorearm mechanism, an event must be rearmed in the program to execute again. Events that autorearm execute every time the trigger condition is met in the program. To disable an autorearm event, it must be disarmed in the Event icon.

Note: Disarming is only required for events that autorearm.

Event Processing

Before an event can be triggered, it must first be entered in the events table. This occurs automatically if the event has been configured in the Calc icon, otherwise it must be configured manually in the events table each time the program is downloaded and activated. An event is

identified by its assigned index number, which is listed in the event table. The order in which events are processed is dependent on their sequential location in the event table.

When an event is armed, its status in the event table changes from inactive to queued. The queued event is monitored until the its associated trigger is reached in the VisualMotion program. The event is then sent to the execution stack of its designated Task. Depending on the type of event, it is either sent to the end or the beginning of the execution stack queue.

In the execution stack, the event's state changes from queued to pending. Pending tasks are executed in a FIFO sequence from their execution stack during which their state changes from pending to executing. The event status returns to inactive when the event has finished executing.

Changed data in the event table while a program is running will not be processed until the event has been rearmed, even though the change is sent to the queue. To allow changes in the event table to be acknowledged by the user program, add a branch icon that monitors for the designated value and then directs the program to rearm the event.

11.3 Event Types

VisualMotion provides the following event types for activation in a user program. Descriptions of the events include a sample program to illustrate the recommended approach to programming.

Percentage of Coordinated Path from Start / before End Event

Multi-axis coordinated motion requires the trigger distance specified as a percentage of the total length of the segment. To create this event in the user program, the following information is required:

- Event type (Percent of coordinated path from start / before end)
- Optional "Event Function" to run
- Percent of path (trigger distance)

When specifying a distance-based event trigger with coordinated motion, an event occurring within a blend segment may not trigger as anticipated. The range of potential paths that could be generated by the path planner through the blend segment must be considered. When one segment is blended into another, with one event set to trigger near the end of the first segment and another event near the beginning of the next segment, and the blend radius specified is sufficiently large, the second segment may blend into the first segment far enough that the second segment event triggers before the first segment event can occur. The blend segment can be controlled by ensuring that the programmed blend radius is smaller than the specified trigger distance of the second segment.

Note: This event is triggered after the specified distance has been traversed on the axis path. It can be triggered from both coordinated and non-coordinated (single-axis) moves. The response time for coordinated motion events is dependent upon the number of coordinated axes in the program and the priority of the task containing the event (task priority A - D). Tasks assigned to coordinated axes require one additional SERCOS cycle. Therefore, the response time may vary from 2 ms to 6 ms SERCOS cycles.

Sample Program

The following is an example of a multi-axis coordinated motion program with an event. The event is configured to run at a designated percent of the path traveled from the start of the path segment. In this example, Event 1 is triggered when the axis has traveled 10% of the distance of path segment 1. A maximum of four events could be triggered in each path segment.

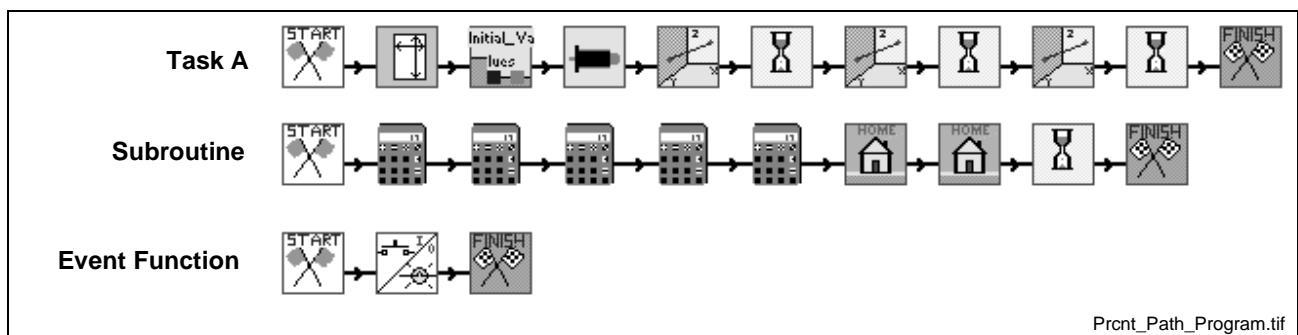


Fig. 11-7: Sample Program with Percent of Coordinated Path Event

The icons used to configure this type of event include:

Declare Event



Calc Icon – The event is defined with the equation:

EVT[1]={0,3,0,10,Event_1, "Distance (10%) from Start Coord path"} .

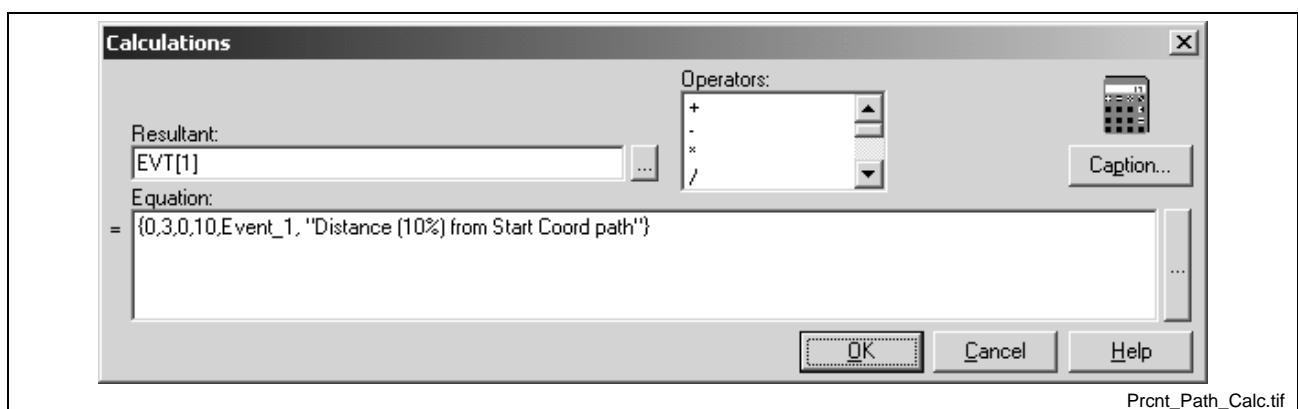


Fig. 11-8: Calc Icon Setup for Percent of Coordinated Path Event

Trigger Event



Calc Icon – Define the point in the path that will trigger the event. Point data entered in the Calc icon is displayed in the Points table after the VisualMotion program is running. Point information can also be entered directly in the table.

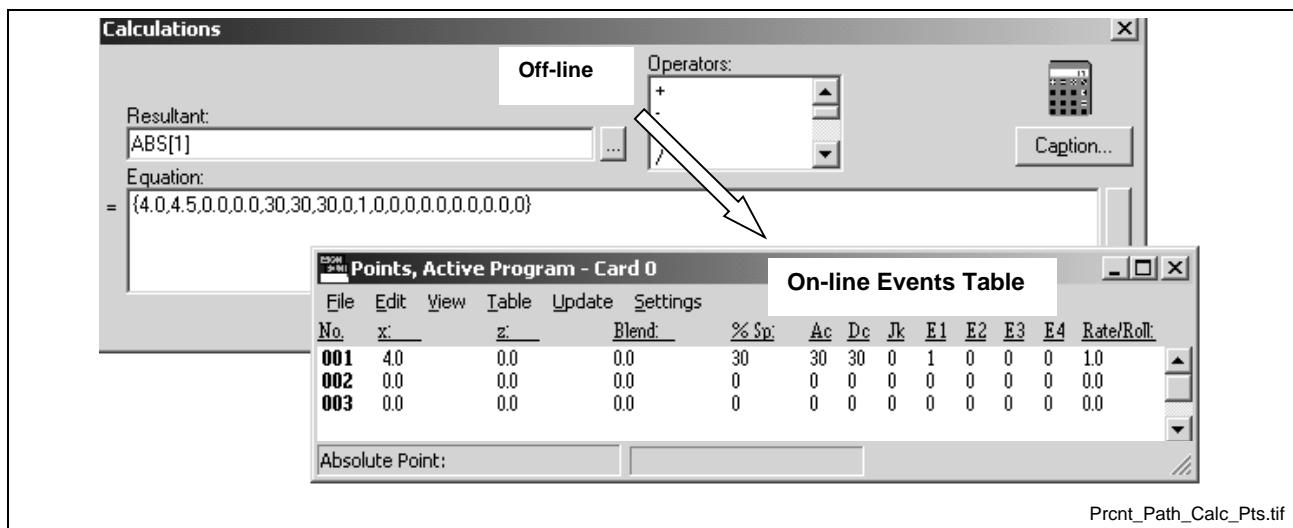


Fig. 11-9: Calc Icon Setup for Percentage of Coordinated Path Event Trigger

Arm/Rearm Event

Path Icon – Arm the event by selecting the motion type and the index of the path endpoint. To rearm the event, place a second Path icon in the VisualMotion program.



Note:



 The circle icon can also be used to arm an event. Select motion type and the index of the path midpoint and endpoint. Rearm the event with a second Circle icon at the end of the VisualMotion program.

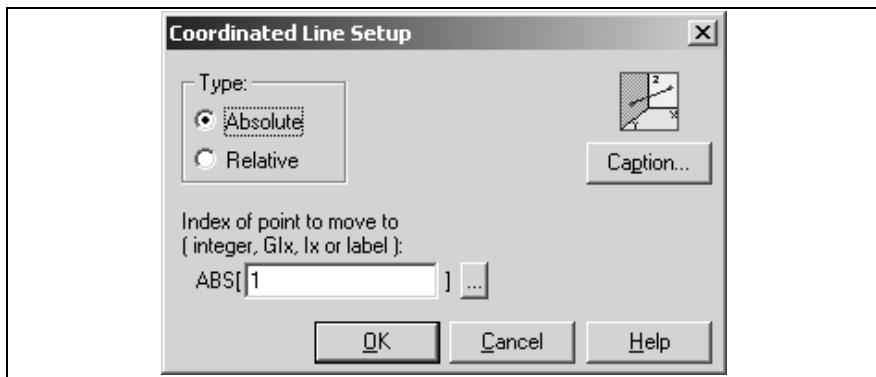


Fig. 11-10: Path Icon Setup for Percentage of Coordinated Path Event

Time In Coordinated Path from Start / before End Event

Coordinated motion can provide time-based events that are related to travel time along a specified geometry segment and are initiated by the path planner. These events execute at a fixed time after motion starts or before motion ends in the specified segment. The following information is required to configure this type of event:

- Event Type (Time in Coordinated Path from start / before end)
 - Optional "Event Function" to run
 - Time (20 – 604,000,000 ms)

Sample Program

The following is an example of a multi-axis coordinated motion program with two events. The events are triggered at designated times (in ms) from the start of the path segment they are assigned to. Each path segment can accommodate up to four events.

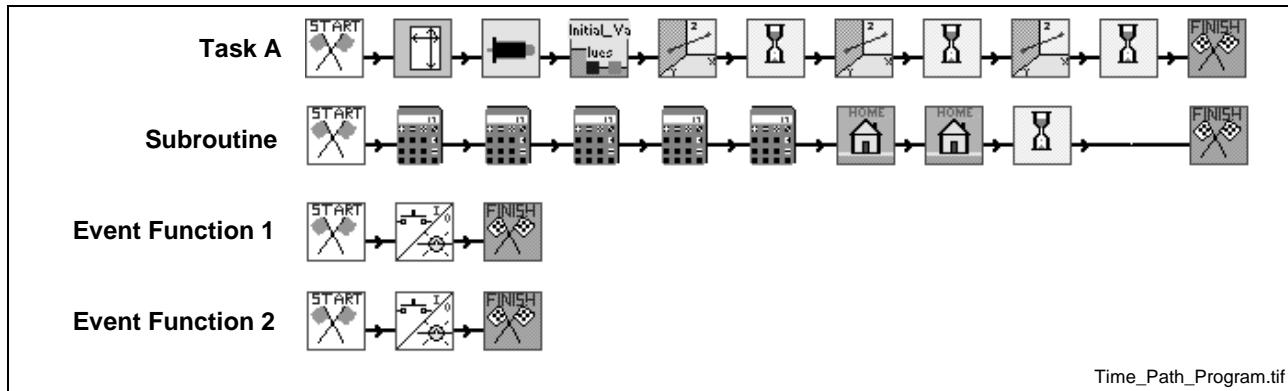


Fig. 11-11: Sample Program Time in Coordinated Path

The icons used to configure this type of event include:

Declare Events



Calc Icon – The events are defined with the equations:

$EVT[1]=\{0,2,0,50,Event_1, "Time (50ms) from Start Coord path"\}$

$EVT[2]=\{0,2,0,30,Event_2, "Time (30ms) from Start Coord Path"\}.$

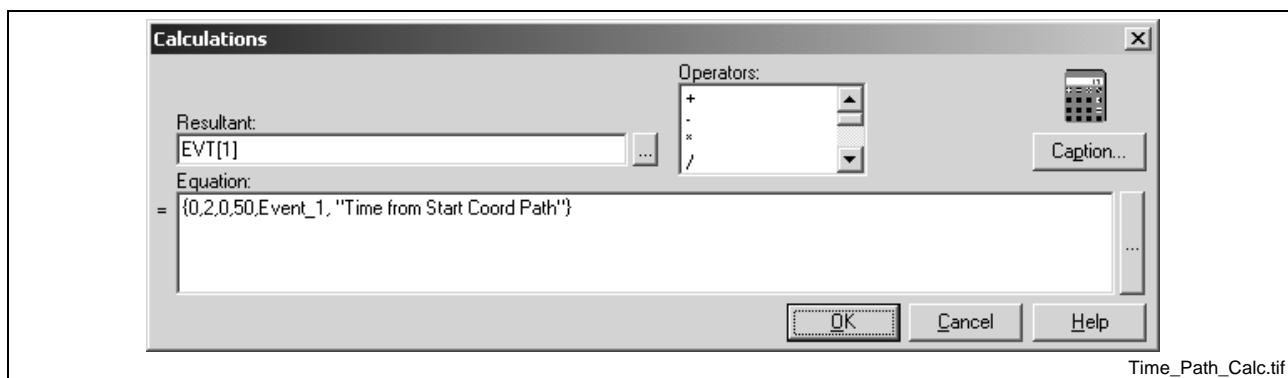


Fig. 11-12: Calc Icon Setup for Time in Coordinated Path Event

Trigger Event



Calc Icon – Define the point in the path that will trigger the event. Point data entered in the Calc icon is displayed in the Points table after the VisualMotion program is running. The table can also be used as an alternative method for setting point information.

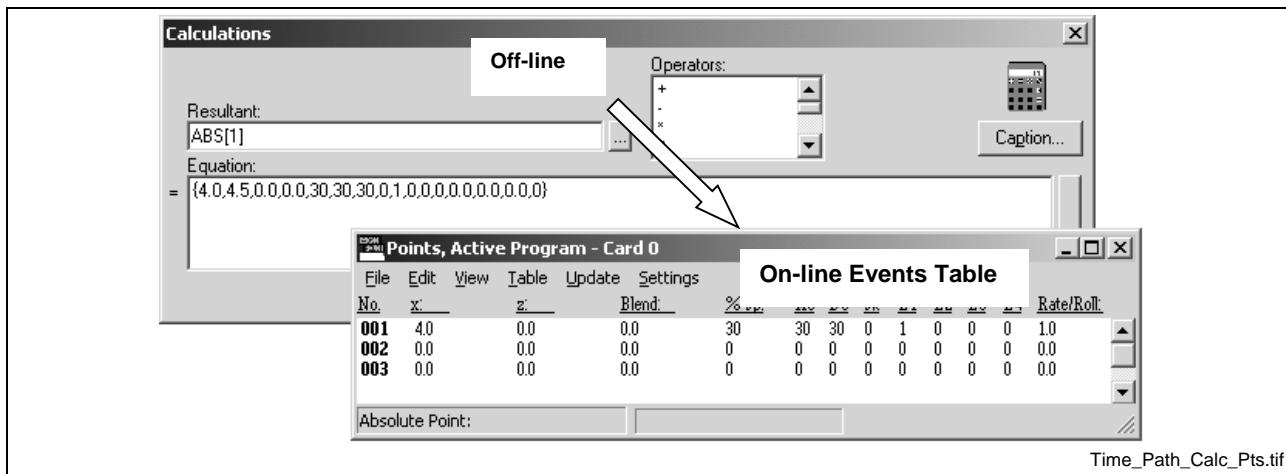


Fig. 11-13: Calc Icon Setup for Time in Coordinated Path Event Trigger

Arm/Rearm Event

Path Icon – Arm the event by selecting the motion type and the index of the path endpoint. Place a second Path icon in the VisualMotion program to rearm the event.

Note:

The circle icon can also be used to arm an event. Select motion type and the index of the path midpoint and endpoint. Rerarm the event with a second Circle icon in the VisualMotion program.

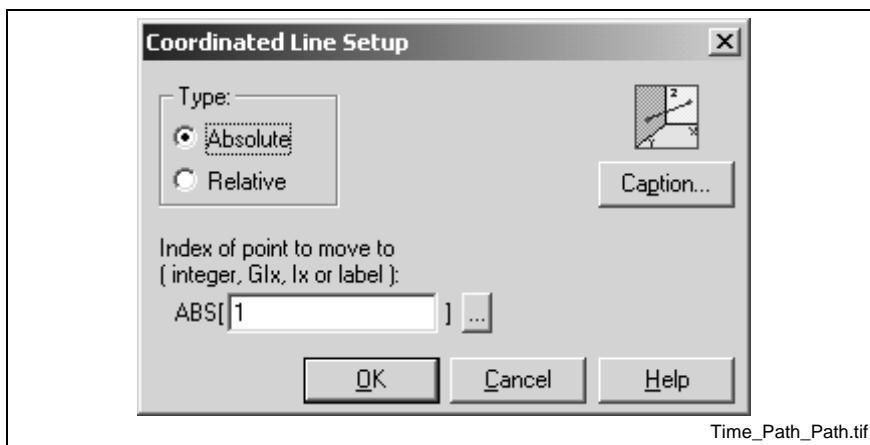


Fig. 11-14: Path Icon Setup for Time in Coordinated Path Event

Single Axis Distance from Start / before End Event

In a Single axis non-coordinated motion program, an event can be configured to occur at a set distance from the start or before the end of the program. By assigning a value to the motion path, the event can be set up to trigger when the value has been reached. The event number to be executed is specified in the "Optional Events" field of the Move icon. The following event information must be entered in the program:

- Event type (Single Axis distance from the start / before end)
- Optional "Event Function" to run
- Axis position (trigger distance)

A maximum of four events can be assigned to each single axis move.

Note: Single axis non-coordinated motion icons use the internal positioning intelligence of Indramat's digital drives. Because the rate profile for single axis motion is developed within the drive, the time method of triggering an event related to motion is not supported.

Sample Program

The following is a sample configuration of a single axis program with two events that are triggered at designated distances (mm or degrees) from the start of the program.

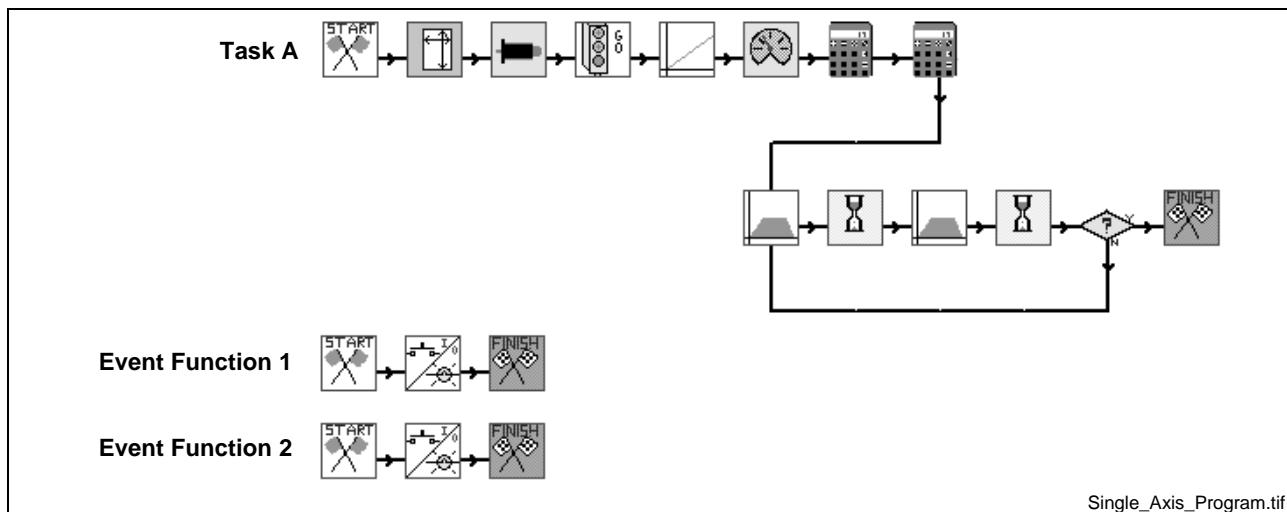


Fig. 11-15: Sample Program with Single Axis Distance Event

The icons used to configure this type of event include:

Declare Events



Calc Icon – The events are defined with the equations:

$EVT[1]=\{0,4,0,2.0,Event_1, "Distance (2.0) from Start of Move"\}$

$EVT[2]=\{0,4,1,25.0,Event_2,"Distance (25.0) from Start end move"\}.$

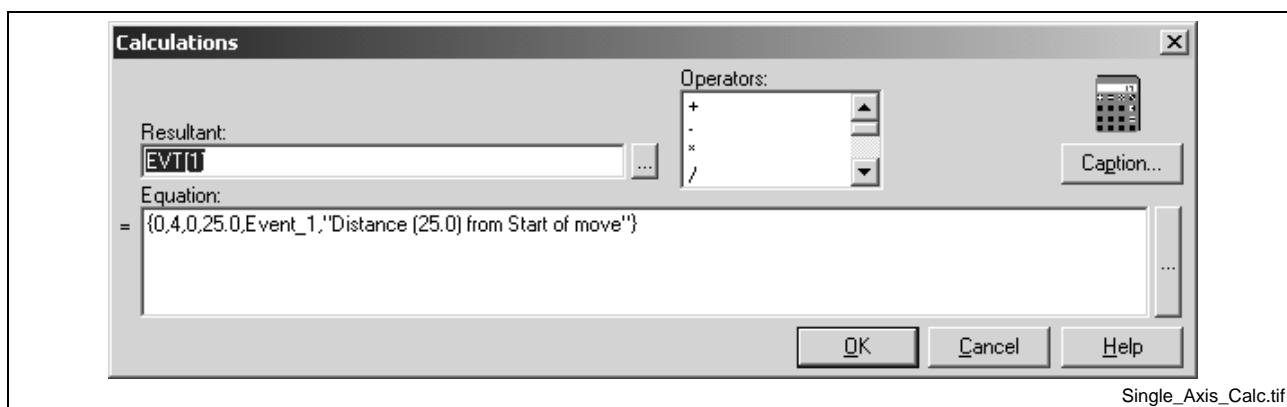


Fig. 11-16: Calc Icon Setup for Distance from Single Axis Event

Arm/Rearm Events Move Icon – Assign the events to an axis move.

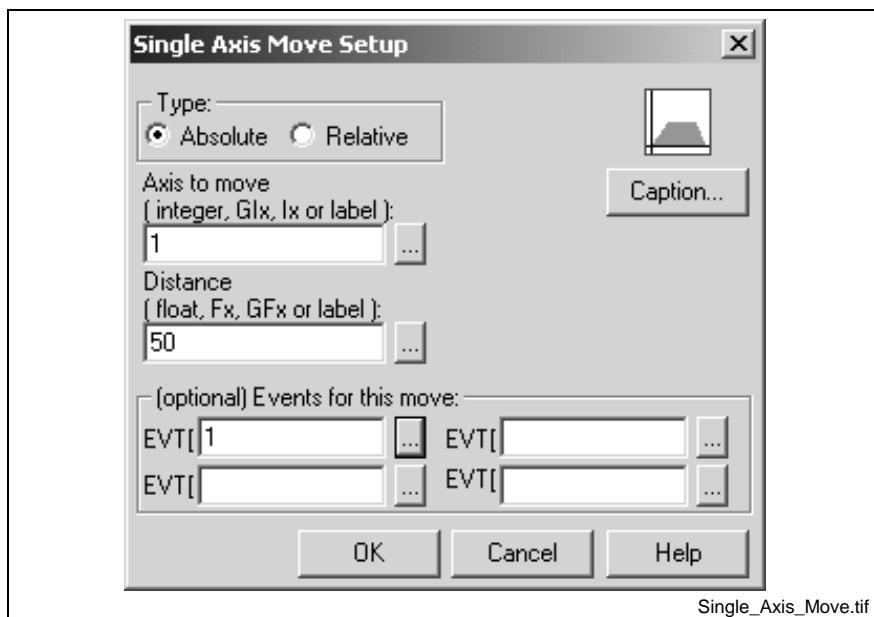


Fig. 11-17: Move Icon Setup for Single Axis Distance from Start Event

Repeating Timer Event

The repeating timer event is a time-based event that executes an operation, such as switching on a pump or calculating statistics, at programmed intervals. Once a repeating timer event is armed, it executes every interval until it is disarmed. If the event is not disarmed, it will continue to execute, even when its associated task has finished. A maximum of 16 repeating timer events can be active at any given time in the user program.

Note: An event must be declared before it can be armed by placing the Calc icon before the arm icon in the VisualMotion program sequence.

The time interval argument for the event specified in the Calc Icon or the Events table has a resolution of 20 milliseconds. The following information must be specified in the Calc icon or Events table to create the event:

- Event Type (Repeating Timer)
- Optional "Event Function" to run
- Time (20 – 604,000,000 ms)

When using a repeating timer event in your program, it is important to restrict the type of icons you place in the event function as the event preempts all user tasks. Task response time and execution can be adversely affected if the event function requires a long time to process. Icons to avoid or use with caution in an event function include:

- Parameter transfer
- Wait
- Branch (looping)

Sample Program

The following sample program contains a repeating timer event that is triggered by a condition in the extended input event, event function 1. Task A arms event function 1 with a manual bit change to provide maximum control over the potentially lengthy repeating timer event. Event function 1 triggers the repeating timer event, event function 2, if the default bit condition does not change. If the bit condition changes, the subroutine is activated to end the event function. Event Function 2 runs the repeating timer event if the same default bit condition in event function 1 remains unchanged. If the bit condition changes, event function 2 runs the subroutine to deactivate the event. After both event functions have finished, Task A resumes.

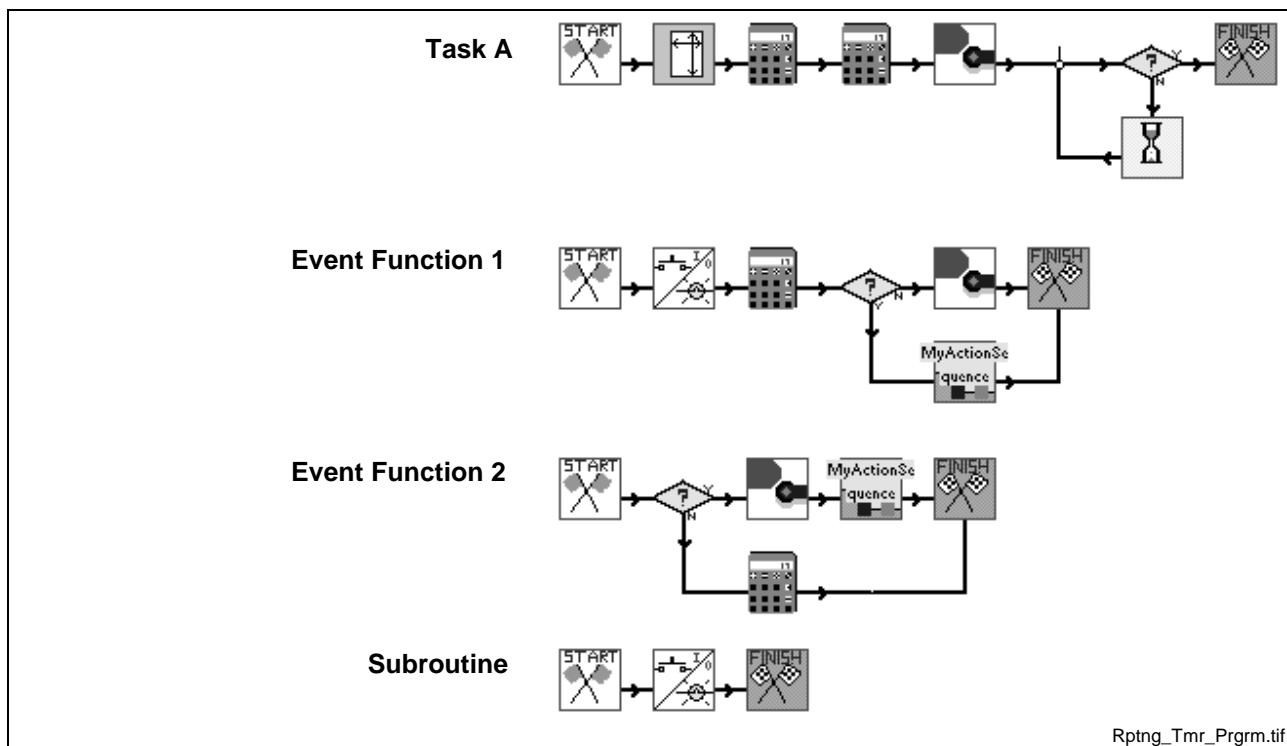


Fig. 11-18: Sample Program with Repeating Timer Event

The icons used to configure this type of event include:

Declare Events



Calc Icon – Events are declared with the equations:

```
EVT[1]={0,10,0,1,Input1_ON,"Extended input evts (reg88_bit1)"};
```

```
EVT[2]={0,1,0,500,PeriodicCheck,"Repeating timer @ 500 ms"}.
```

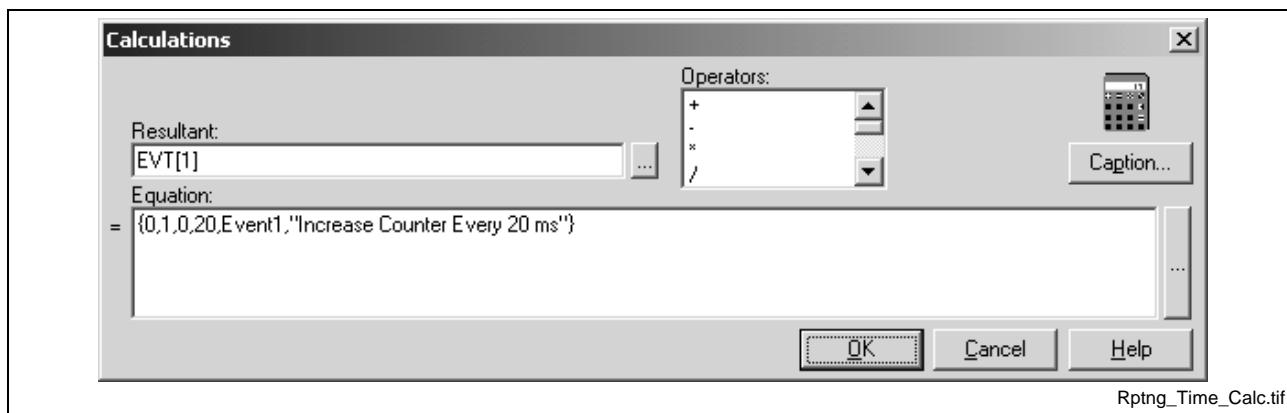


Fig. 11-19: Calc Icon Setup for Repeating Timer Event

Arm/Disarm Event



Event Icon - Select Arm Event and the Event index to arm the event. To disarm the event, place a second Event icon at the end of the event function or the end of the Task and select Disarm Event. In the sample program, event function 1 arms event 2, the repeating timer event. Event 2 is disarmed when the I/O register 120 bit 9 in the event goes low or the disarm event icon is encountered after the condition set in the branch icon of Task A is met.

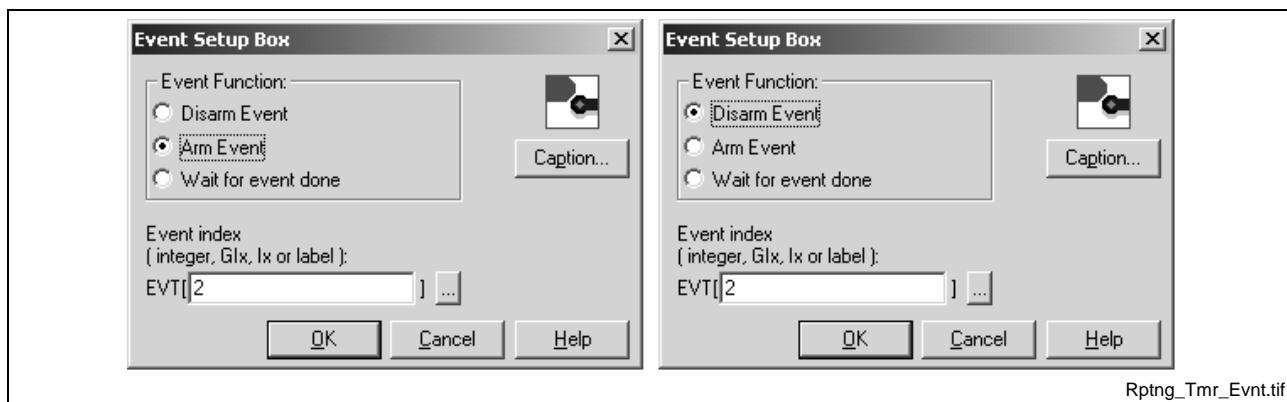


Fig. 11-20: Event Icon Setup for Arming/Disarming Repeating Timer Event

Rotary (Repeating Axis Position) Event

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

Each time the event position is passed in either direction, the function specified in the event table is executed. Because rotary motion uses the shortest path to reach the next specified position, verify that the axis will travel through the position specified to trigger the rotary event.

Once a repeating rotary event is armed, it executes every programmed degree interval until it is disarmed. If the event is not disarmed, it will continue to run even when its associated task has ended.

Note: An event must be declared before it can be armed by placing the Calc icon before the arm icon in the VisualMotion program sequence.

The following information is required to configure this type of event:

- Event type (Rotary – Repeating Axis Position)
- Optional "Event Function" to run
- Degrees (0 - 359)

Important:

If changes are made to the arguments of rotary events in the event table, then the "Rotary Events" icon must be executed again in the user program before the changes will take effect.

Note: Rotary events cannot be attached to an axis with single axis distance based events.

Sample Program

The following is a sample single axis motion program with a rotary event. A maximum of four events can be configured for each axis, group, or master in a program.

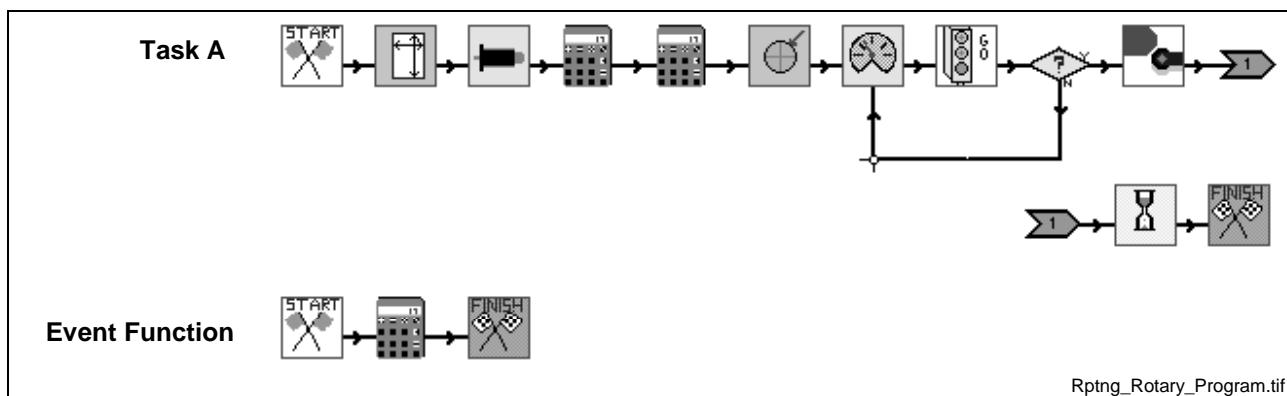


Fig. 11-21: Sample Program with Repeating Rotary Event

The icons used to configure this type of event include:

Declare Event Calc Icon – The event is defined with the equation:



EVT[1]= {0,5,0,180,Event1,"Increase counter"}.

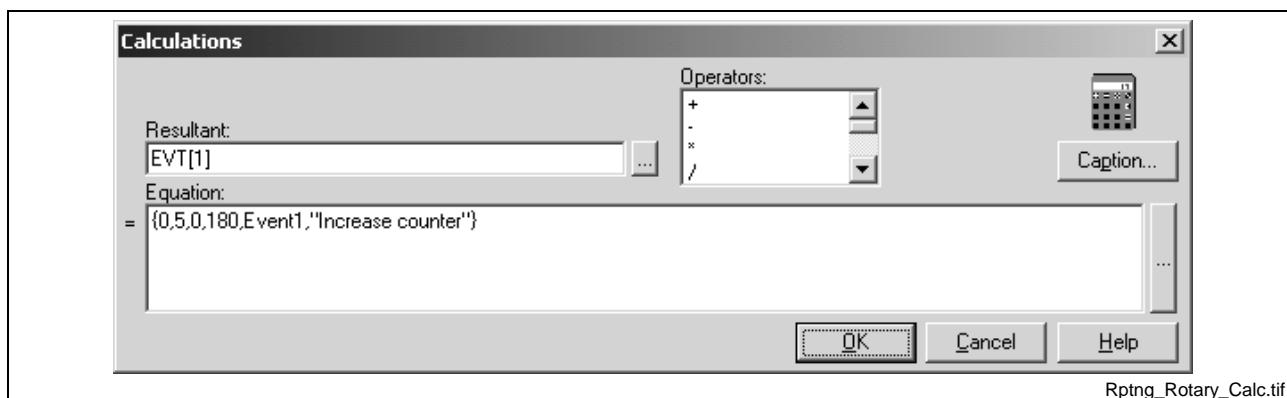
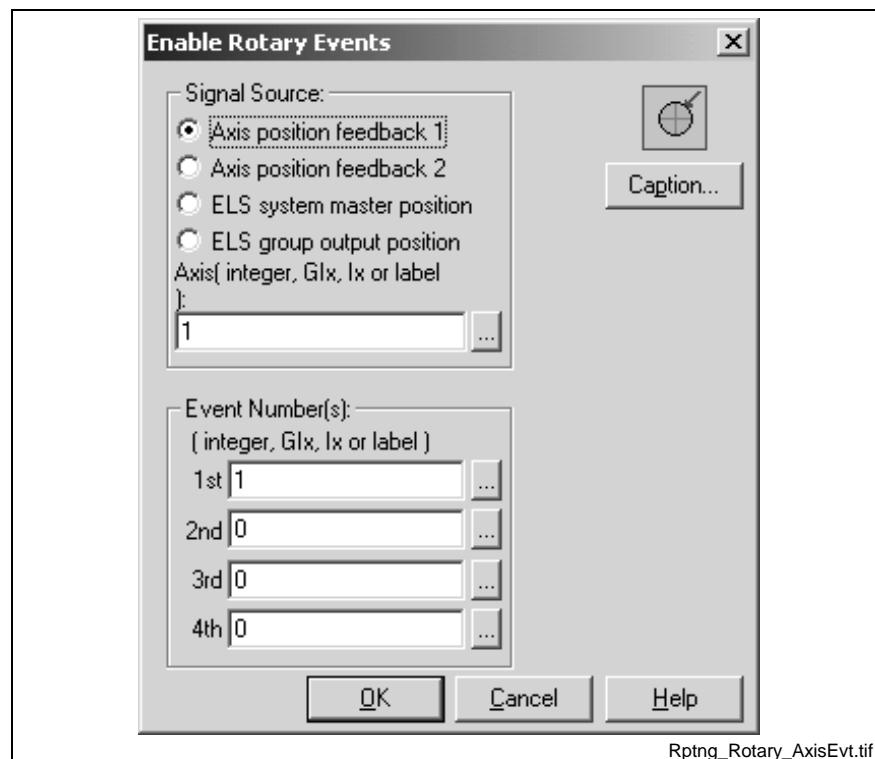


Fig. 11-22: Calc Icon Setup for Repeating Rotary Event

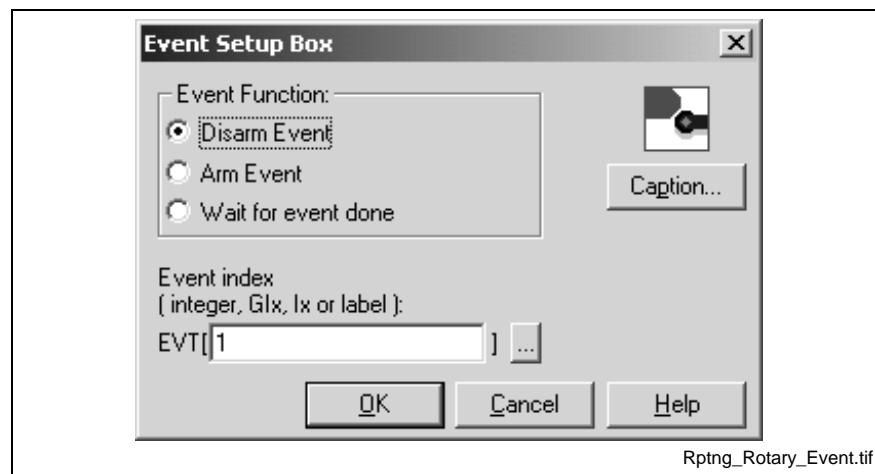
Arm Event

AxisEv1 Icon - Arm the event by selecting the signal source and event number. In this example, the signal source selected is the axis feedback position for axis number 1.



Rptng_Rotary_AxisEvt.tif

Fig. 11-23: AxisEv1 Icon Setup to Enable Repeating Rotary Event

Disarm Event Event Icon – Select Disarm to disarm the event.

Rptng_Rotary_Event.tif

Fig. 11-24: Event Icon Setup to Disarm Repeating Rotary Event

Task Input Transition Event

Bit 9 of each Task's Control Register (2 - 5) is reserved as an event interrupt input for the task. Every low-to-high transition of this input can trigger an event in the task. An interrupt input type event permits triggering an event function from an external input. The following information is required to create a Task Interrupt event:

- Event type (Task Input Transition)
- Optional "Event Function" to run

The control scans the input every 2ms and queues an event upon a low-to-high transition. Once triggered, 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 invert the logic of the interrupt input, or to direct other external inputs to the Task Control Register's Event Interrupt bit. Logic in the event function can then scan the multiple inputs to determine the source of the interrupt.

Sample Program

The following is a sample single axis non-coordinated motion program with a task input transition event. A maximum of one event can be configured for each program task.

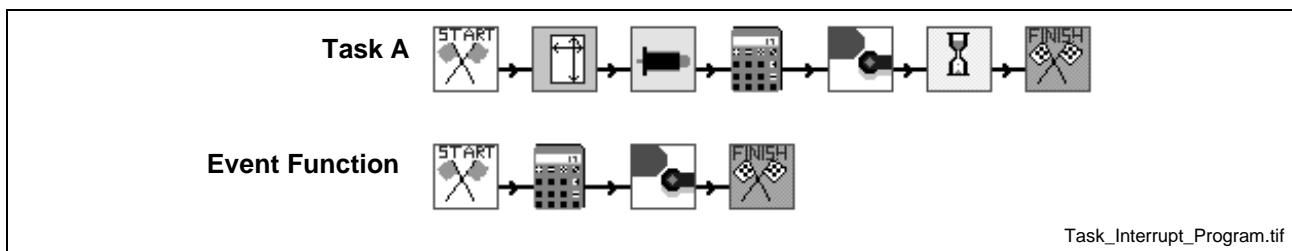


Fig. 11-25: Sample Program with Task Input Transition Event

Declare Event Calc Icon – The event is defined with the equation:
 $EVT[1] = \{0,6,0,0,Event_1,"Increase counter"\}.$

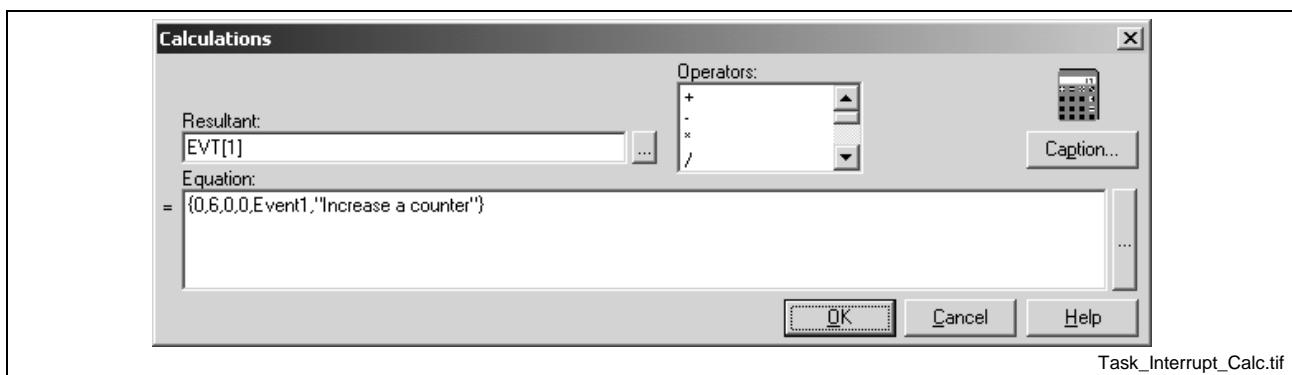


Fig. 11-26: Calc Icon Setup for Task Input Transition Event

Arm/Rearm Event Event Icon - Select Arm for the event. In the event, use this icon with Arm selected to rearm the event.

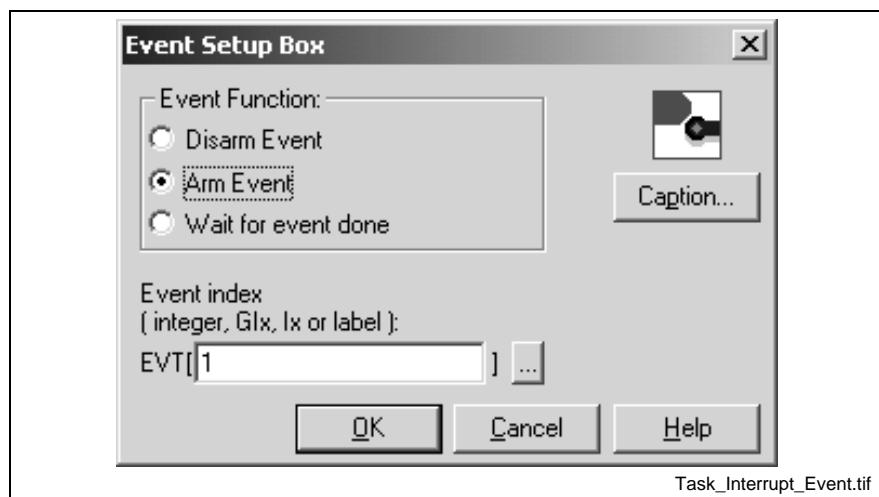


Fig. 11-27: Event Icon Setup to Arm Task Input Transition Event

Feedback Capture

VisualMotion uses the SERCOS probe functionality and real-time bits along with the event system to allow user programs to perform registration functions. The DIAX04 and ECODRIVE03 digital drives provide two probe inputs that can be used for capturing the feedback position.

Note: Only the DIAX04 digital drive is capable of transmitting probe feedback from two probes simultaneously through the SERCOS cyclic data telegram. The ECODRIVE03 is limited by its smaller bit capability to a single probe. To send a second probe signal with the ECODRIVE03, it is necessary to re-initiate the program.

The inputs are physically wired to each drive according to Fig. 11-28.

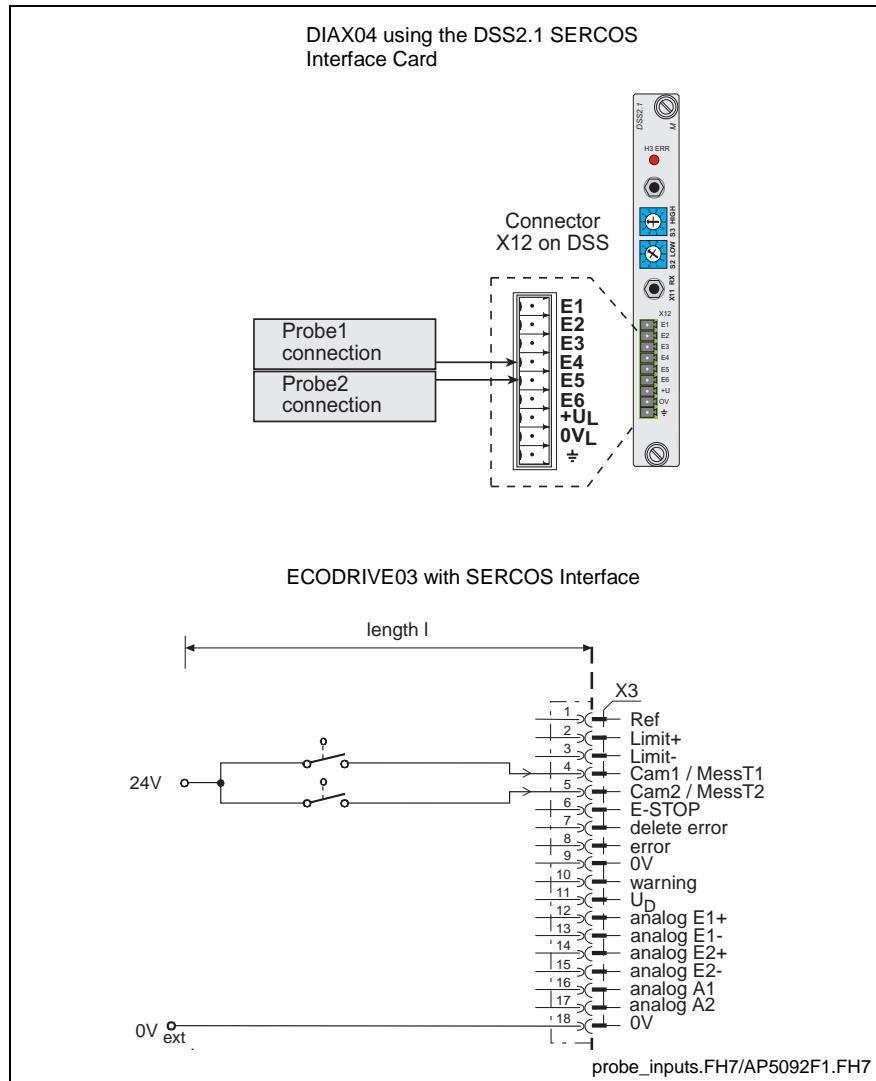


Fig. 11-28: Connecting the Probe Inputs

The probe inputs are scanned every 1 µs. Upon either a positive or negative transition of a probe input, the drive captures and places the position into the cyclic data telegram.

Typically, probes are used to detect registration marks on material. By controlling when the probe is armed, other printing on the material can be filtered out. When the position is captured, the drive signals the control with a real time bit in the SERCOS cyclic data telegram. When the control detects a change in the real time bit, it can execute an optional event function.

Note: It is also important that the value being read in the feedback event matches the parameter in the amplifier telegram (AT). If feedback is requested from a probe and its associated parameter is not in the AT, the service channel is used to transmit the data.

The following table lists the four probe triggers and their associated SERCOS and Control parameters:

Probe Trigger	SERCOS Parameter	Control Parameter
Probe 1, 0->1	S-0-0130	A-0-171
Probe 1, 1->0	S-0-0131	A-0-172
Probe 2, 0->1	S-0-0132	A-0-173
Probe 2, 1->0	S-0-0133	A-0-174

Table 11-2: Capture Trigger and Associated SERCOS and Control Parameters

The probe input is stored in the drive as an S (SERCOS) parameter. The probe parameter is transferred to the PPC through the SERCOS telegram and stored in the PPC as an A (Axis) parameter, as illustrated in the figure below. The user program in the PPC can reference the A parameter through the A argument of a Probe Event (F10 = EVT[1].A) configured in the Calc icon. The program can also reference the parameter through the A argument configured in the Parameter Transfer icon.

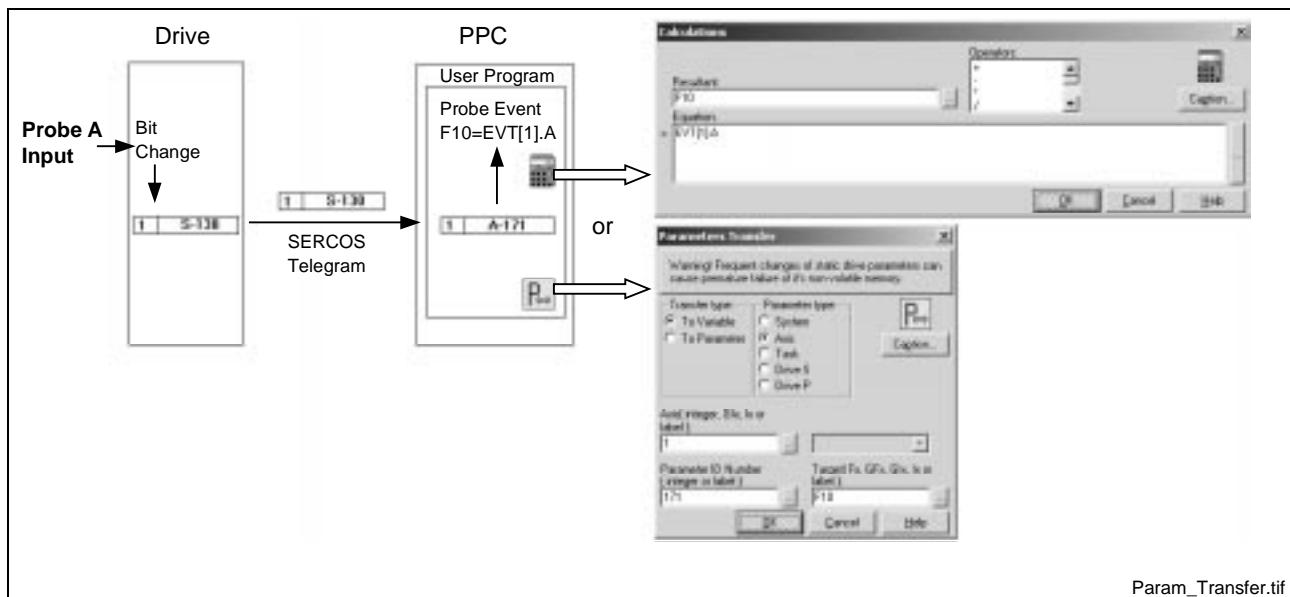


Fig. 11-29: Parameter Transfer Sequence

Feedback Capture for Single Axis, Velocity Mode and Ratioed Axis

For Single Axis and Velocity mode applications, the Feedback Capture events are configured through the **Axis** icon and armed through the **ProbeEvt** icon. Both icons should be configured with the same trigger for the SERCOS probe.

The following information is required to configure a feedback capture event:

- Event type (Feedback Capture)
- Optional "Event Function" to run

Sample Program

The following is an example of a single axis program with a feedback capture event. A maximum of two events for each drive can be configured in the user program.

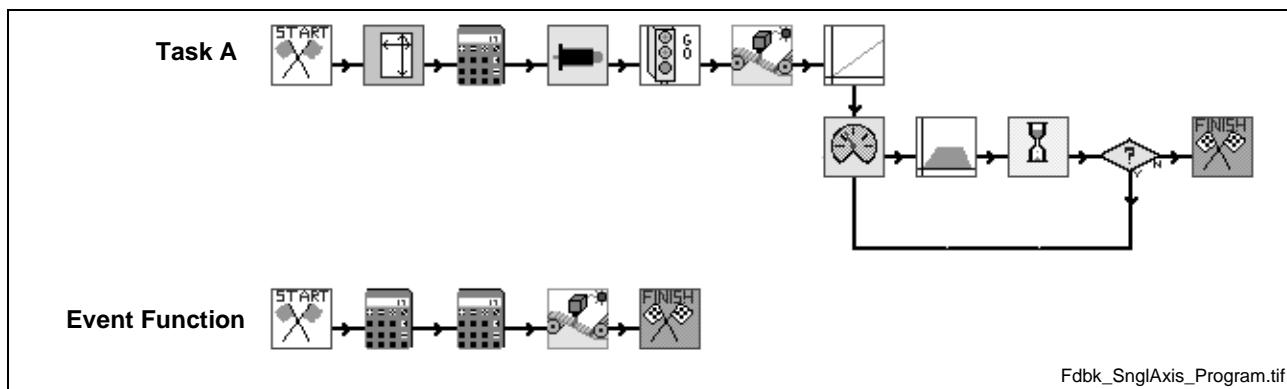


Fig. 11-30: Feedback Capture Event for Single Axis

This example contains the following icons for creating a program with this type of event:

Declare Event Calc icon - Set the trigger for the position capture event with the equation:
EVT[1]={0,9,0,0,Event_1,"Axis Probe Position Capture".}

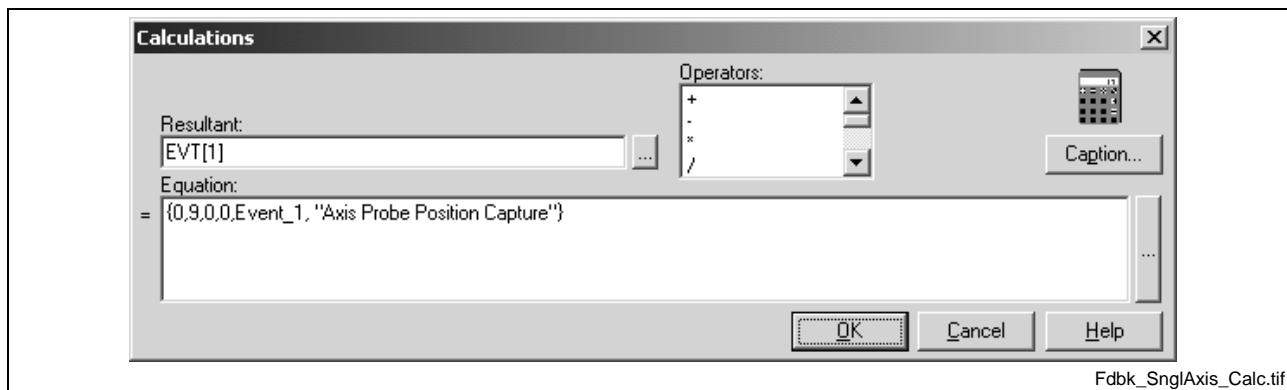


Fig. 11-31: Calc Icon Setup for Single Axis Feedback Probe Event

Arm/Rearm Event ProbeEvt icon - Select trigger to enable feedback capture event and rearm the event

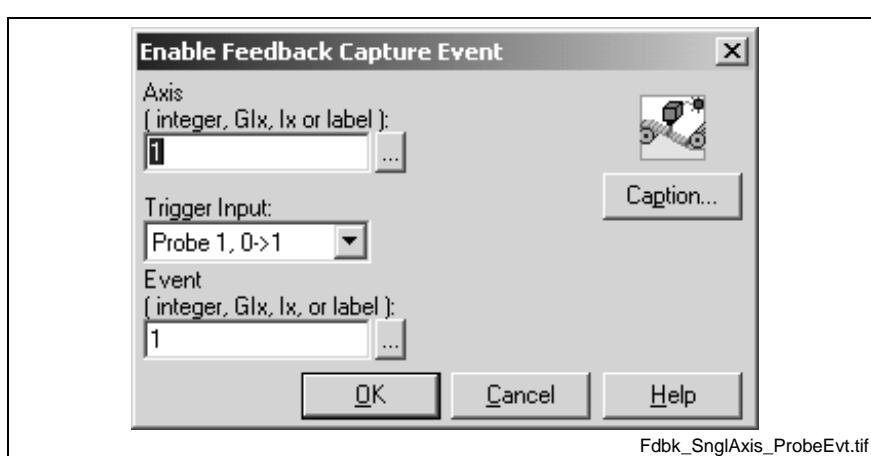


Fig. 11-32: ProbeEvt Icon Setup for Single Axis Feedback Probe Event



Calc icon - A VisualMotion program with a probe must contain a Calc icon in the event with the argument, F10=EVT[1].A. This allows data collected from the probe to be read by the controller.

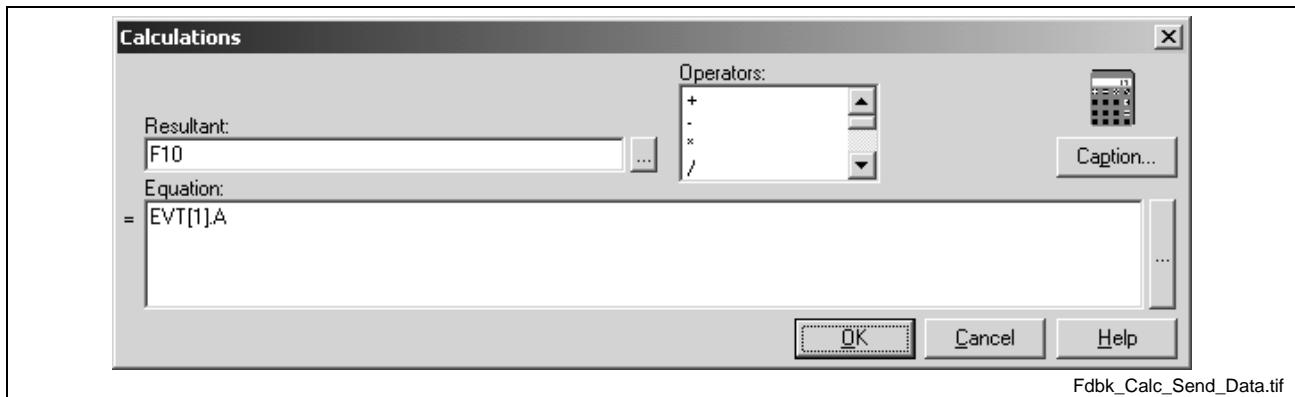


Fig. 11-33: Calc Icon Setup for Probe Feedback

Feedback Capture for ELS, Coordinated and Torque Modes

When an axis is configured for ELS, Coordinated or Torque modes, the triggers for a feedback event cannot be configured within the Axis icon. For this reason, a Probe icon must be used to setup the SERCOS probe functionality in the drive.

The following information is required to set up a feedback capture event:

- Event type (Feedback Capture)
- Optional "Event Function" to run

Sample Program

The following is a sample ELS program with a position capture event. A maximum of two events for each drive can be configured in the user program.

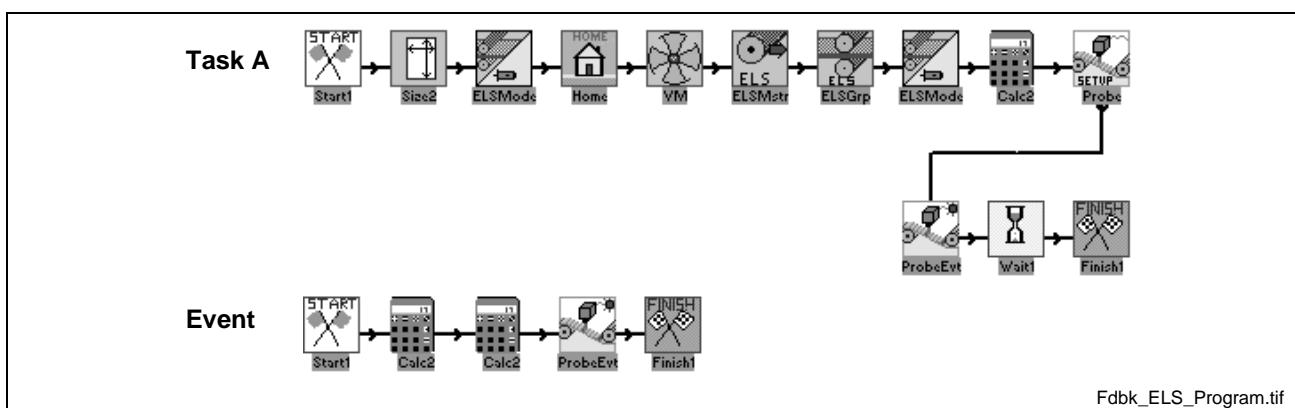


Fig. 11-34: Sample Program with Feedback Capture Event for ELS

The icons for creating an ELS, coordinated, or torque mode program with a feedback capture event include:

Declare Event Calc icon - Define the event trigger information with the equation:

 $EVT[1]=\{0,9,0,0,Event_1,"Axis probe position capture"\}.$

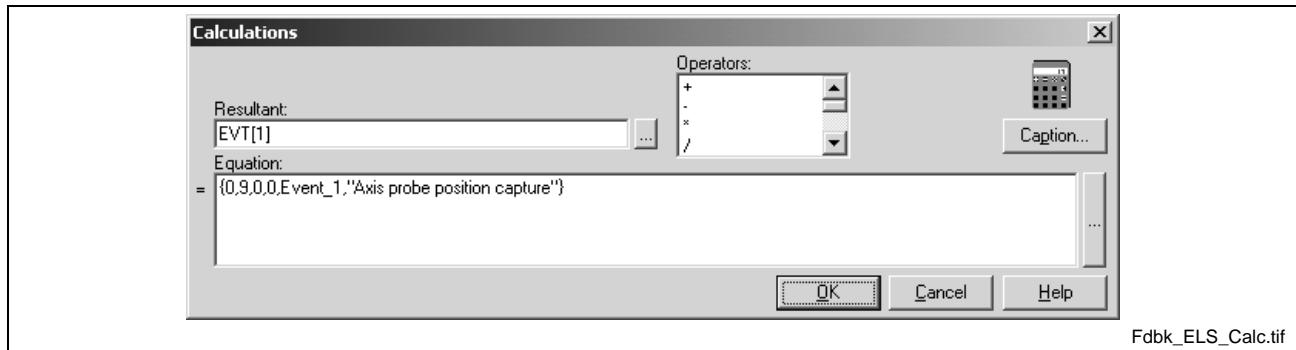


Fig. 11-35: Calc Icon Setup for ELS Feedback Event

Event Trigger Probe icon – Select the capture trigger for the event.

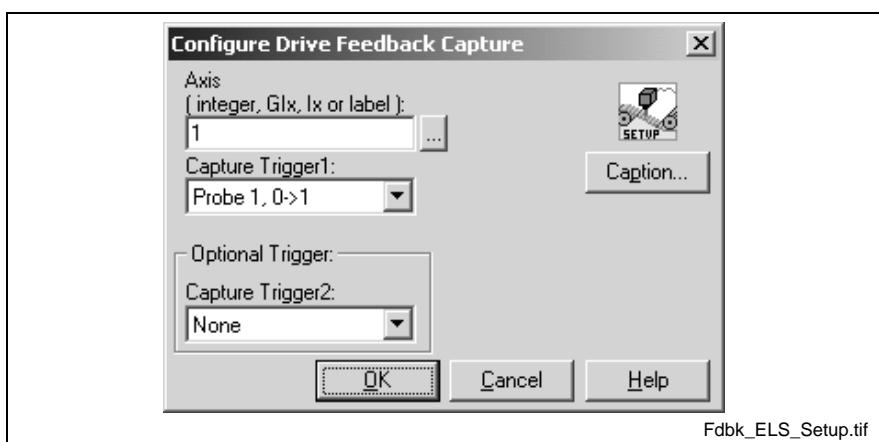



Fig. 11-36: Probe Icon Setup for ELS Feedback Event

Arm/Rearm Event ProbeEvt icon - Select the axis number, event number, and trigger input to enable the feedback capture event. This icon is also used to rearm the event.

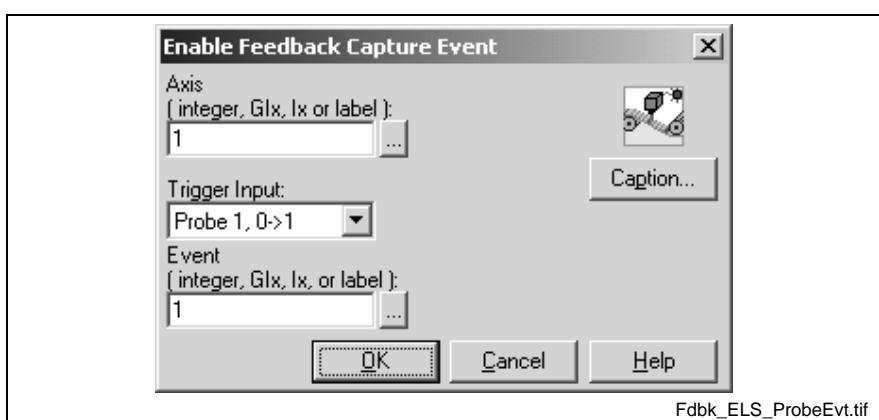



Fig. 11-37: ProbeEvt Icon Setup for ELS Feedback Event



Calc icon - A VisualMotion program with a probe must contain a Calc icon in the event with the argument, F10=EVT[1].A. This allows data collected from the probe to be read by the controller.

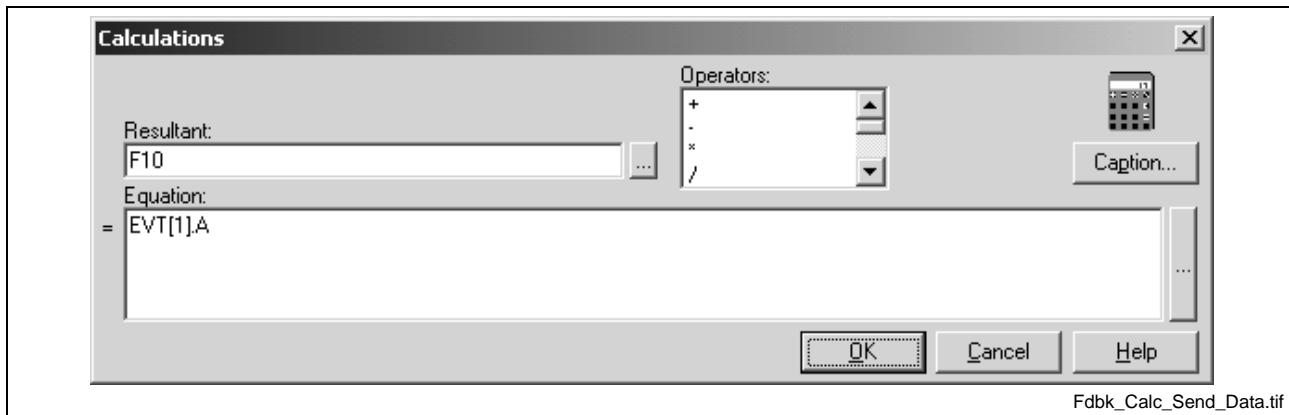


Fig. 11-38: \$\$Graphic caption\$\$

I/O Register Event

The I/O Register event uses register 88 (USER_XI_REG) to trigger up to 16 events in Task A. While, register 89 (USER_XO_REG) is used to monitor the status of events triggered by Register 88. Each low-to-high transition of a bit triggers an event in the corresponding task. Together they provide a time critical way to control motion without 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). The I/O Register event uses input register 88 (USER_XI_REG) to trigger an Extended User Input (EUI) on a positive edge bit transition. The event function assigned to the bit is executed at that time. Only positive rising edges in register 88 can trigger an EUI.

When an EUI is armed through the Event Setup Box Icon, its bit is set in register 89. This provides an external output that indicates the EUI is armed and ready for operation. It remains reset until a high-to-low (1-0) transition on the corresponding input bit in register 88 is detected. When an input is detected, the output bit in register 89 is reset; indicating the event is armed and ready for another positive edge. The output bit is also reset if the event is disarmed through the Event icon.

The following information is required for configuring an I/O Register event:

- Event type (I/O register event)
- Optional "Event Function" to run
- Bit number (of Register 88)

Sample Program

The following is an example of a single axis motion control program with an I/O Register event. A maximum of 16 absolute events can be configured for a task.

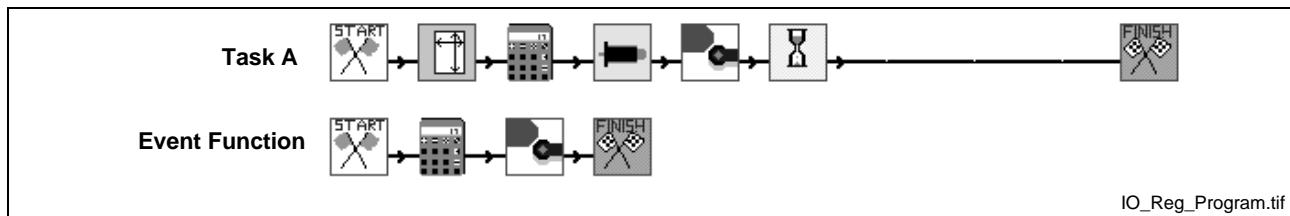


Fig. 11-39: Sample Program with I/O Register Event

Declare Event Calc icon - Declare the event trigger information with the equation:



EVT[1]={0,10,0,16,Event_1,"IO Register Event"}.

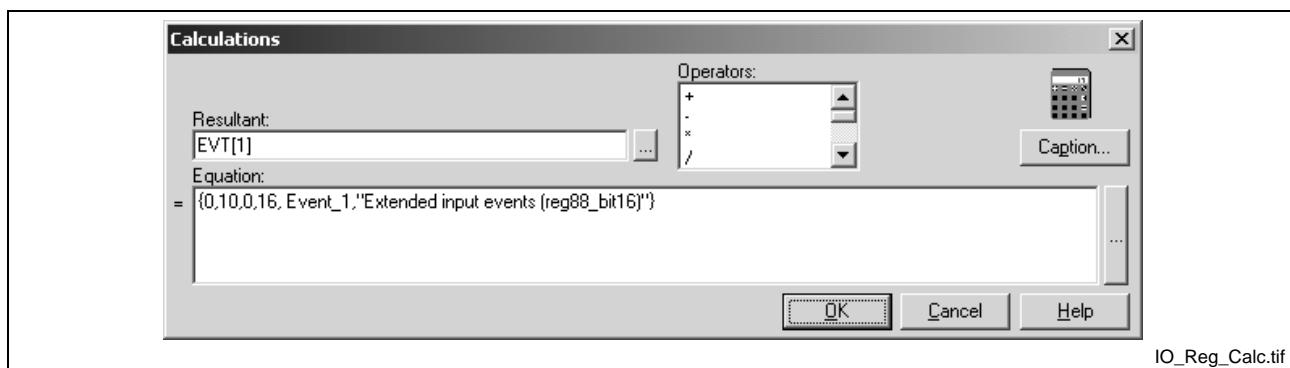


Fig. 11-40: Calc Icon Setup for I/O Register Event

Arm/Rearm Event Event Icon - Select Arm Event to arm the event. Place a second Event icon with Arm Event selected after the first Event icon in the user program to rearm the event.

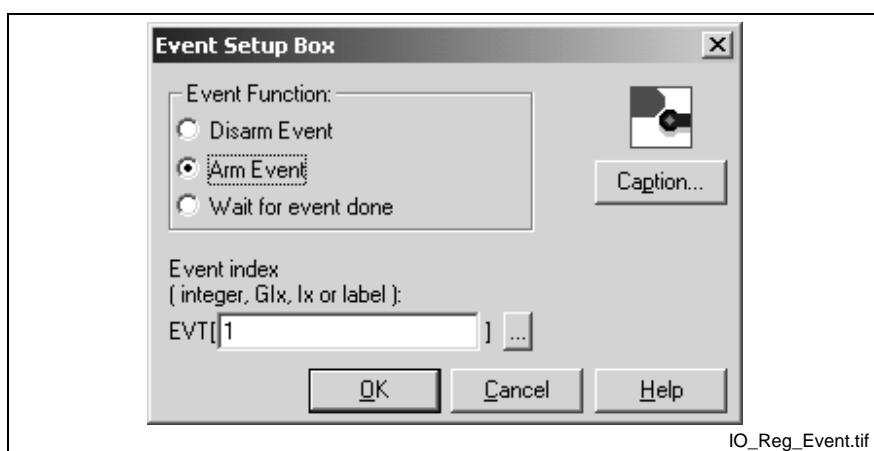


Fig. 11-41: Event Icon Setup for I/O Register Event

PPC-R X1 Input Event

The PPC-R X1 High Speed Inputs can be used for high priority events. When a positive or negative rise is detected by pins 3, 4, and 5 on Connector X1 of the PPC-R, the associated event function is triggered. If another event is currently running and a PPC-R X1 input event is triggered, it will run immediately after the current event has finished and before the next event in the queue.

The PPC-R X1 input event is triggered by a physical switch that is connected to pin 3, 4, or 5 of connector X1. The event is triggered by latching the switch to provide 24V to the input. The event could be triggered by either a latch or unlatch of the switch.

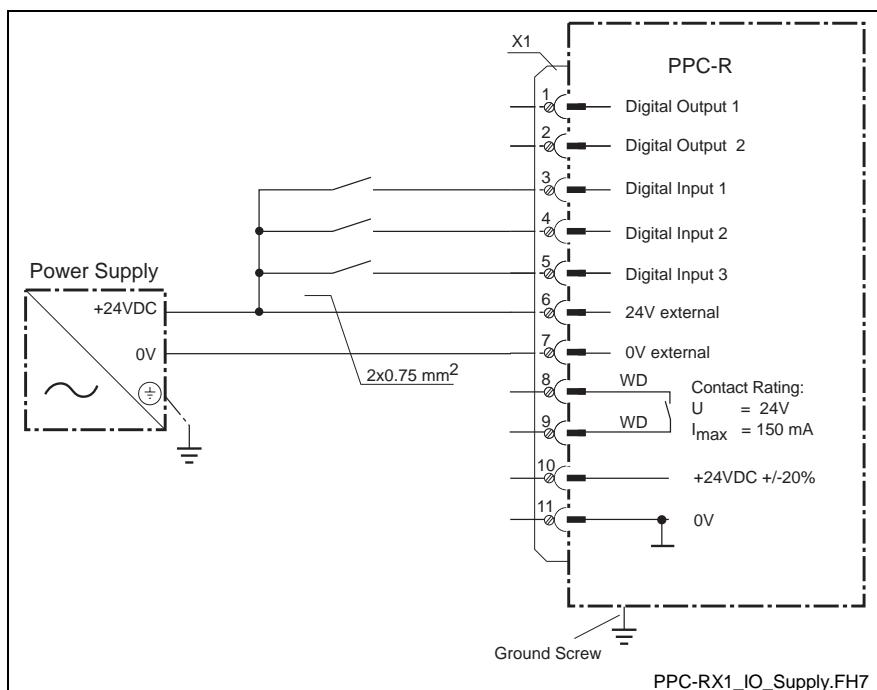


Fig. 11-42: Wiring Diagram for Digital Input/Output Supply Voltage for PPC-R X1 input event

Note: The digital inputs on connector X1 are not functional unless 24V are supplied to pins 6 and 7.

Sample Program

The following is an example of a single axis motion control program with a PPC-R X1 Input event. A maximum of three absolute PPC-R X1 input events can be configured for each task.

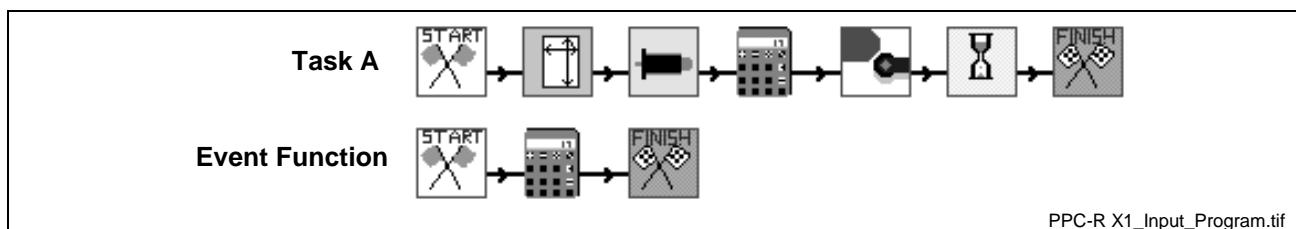


Fig. 11-43: Sample Program with PPC-R X1 Input Event

The icons used to configure this type of event include:

Declare Event



Calc Icon - Define the event with the equation:

EVT[1]={0,11,0,2,Event_1, "Fast PPC input event (PPC 12 0→1")} for a positive rise or latch trigger or

EVT[1]={0,11,1,3,Event_1, "Fast PPC input event (PPC 13 1→0")} for a negative rise or unlatch trigger.

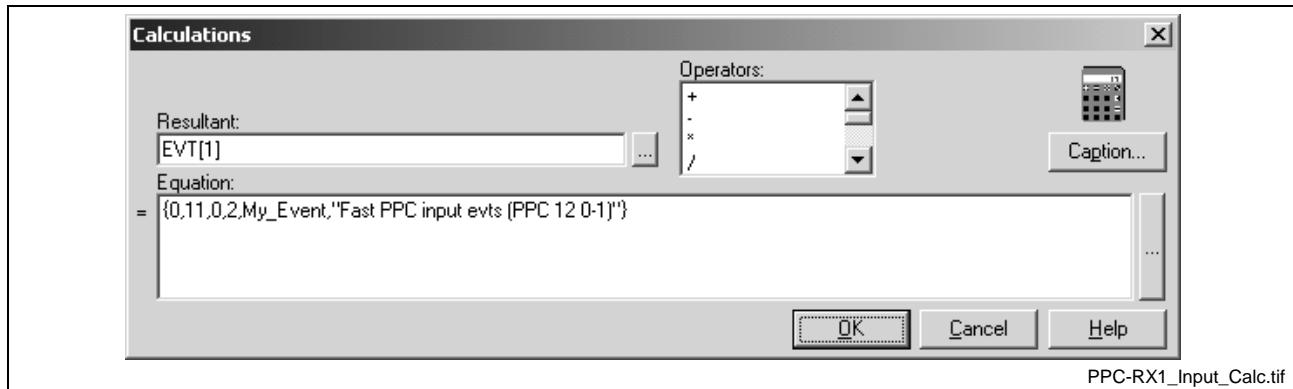


Fig. 11-44: Calc Icon Setup for PPC-R X1 Input Event

Arm/Rearm Event



Event Icon - Select Arm Event to arm the event. Place a second Event icon with Arm Event selected after the first Event icon in the user program to rearm the event.

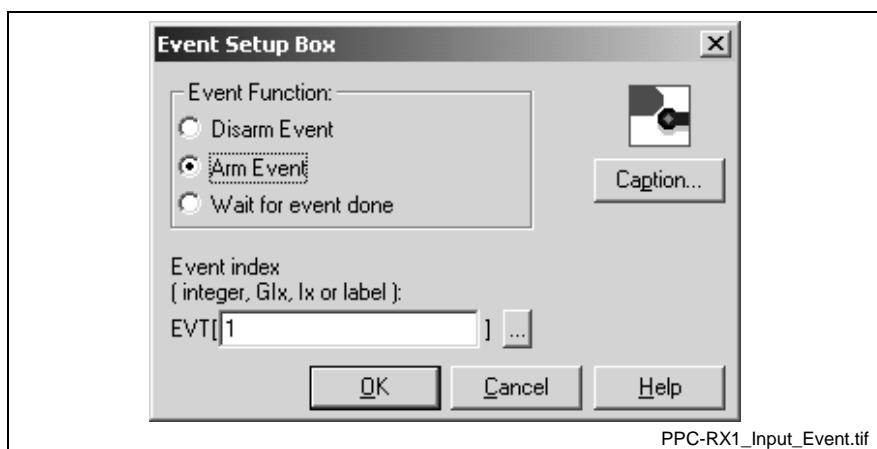


Fig. 11-45: Event Icon Setup for PPC-R X1 Input Event

11.4 Summary of Event Types

The following table contains a summary of the parameters for each event type described in detail in the previous section.

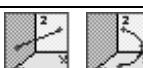
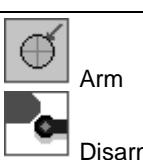
Event Type	Arm Mechanism	Auto Rearm	Maximum Number	Priority*
Percentage of Coordinated Path from Start/End		No	4 events per move segment	1 to 4, based on which task the axis is assigned
Time in Coordinated Path from Start/End		No	4 events per move segment	1 to 4, based on which task the axis is assigned
Single Axis Distance from Start/before End		No	4 events per single axis move	1 to 4 based on which task the axis is assigned
Repeating Timer		Yes	16 absolute	5
Rotary Repeating Axis		Yes	4 for each axis, group, or master	1 to 4 based on which task the single axis is assigned
Task Input Transition		No	1 per task	1 for Task A, 2 for Task B, 3 for Task C, and 4 for Task D
Feedback Capture for Single Axis, Velocity Mode, and Ratioed Axis		No	2 per drive	1 to 4 based on which task the axis is assigned
Feedback Capture for ELS, Coordinated, and Torque Modes		No	2 per drive	1 to 4 based on which task the axis is assigned
I/O Register		No	16 absolute	1 for Task A
PPC-R X1 Input		No	3 absolute	1 for Task A

Table 11-3: Summary of Event Parameters

* Priority Levels: Task A = 1 (Highest)

Task B = 2

Task C = 3

Task D = 4

Repeating Timer = 5 (Lowest)

12 Program Debugging and Monitoring

12.1 Finding Program Problems

Identifying errors in complex programs can be a difficult and frustrating process. Therefore, understand the application and design a program structure around the application.

Start with simple, basic program blocks. Test the blocks, independently if possible, even if testing requires writing a bit more program just for test purposes. A tested and dependable section of a program allows you to focus on just the potential problem areas. If the program compiles correctly, make sure that the problem lies with the program, not the hardware. If necessary, write short test programs to test individual hardware functions.

Use a program branch and the VisualMotion's message capabilities to insert a message into your program. Shortly stopping the program and checking critical values can tell you where things are going wrong.

Think through the implications of using triggered events. Remember that events and the execution of event functions typically occur asynchronously to program tasks. You cannot always depend on the timing of triggered events. It may be necessary to add additional program code to provide synchronization.

The following Task parameters can also be used to help with program debugging:

- T-0-0130 Current Instruction Pointer
- T-0-0131 Current Instruction
- T-0-0132 Instruction Pointer at Error
- T-0-0133 Composite Instruction Pointer
- T-0-0135 Current Subroutine
- T-0-0136 Stack Variable Data
- T-0-0137 Task Subroutine Breakpoint
- T-0-0138 Sequencer Information
- T-0-0200 Last Active Event Number

If the program does not compile, or compiles with errors, use VisualMotion's "Display Code" selection from the Build menu to check that the compiler is generating the instructions you intend. Remember that the compiler doesn't check your program's logic, the compiler can only check for proper syntax and use. VisualMotion's compilers typically provide error or warning dialog windows that refer to line numbers in the displayed code. The following section provides the syntax of the displayed code.

Test Code

A typical example of additional code for testing program functionality is the use of counters. One way to implement a counter would be to change the state of an I/O bit after a distance event has occurred. After each move that is suppose to trigger the event, increment an integer variable called "move_count." Then, use a branch statement to test whether the I/O bit did in fact change state. If it did, then increment an integer variable called "event_count." The final value of "event_count" can be compared to "move_count" to see if, in fact, the event occurred once for every move.

12.2 Control Compiler Base Code

Compiling an Icon or Text Language program produces a text file output listing in Control Base Code, using mnemonics and syntax similar to assembly language. The Base Code resulting from the compilation of a program may be viewed using Window's Notepad by selecting "Display Code" from the VisualMotion Build menu. Base Code may also be viewed using a compatible ASCII-only text editor.

Base Code is typically used as an aid to debugging when checking a program for logical errors. Base Code files are view-only program listing files. Editing a Base Code file has no effect on a subsequent re-compilation of the program.

The labels in a Base Code listing result from both user-defined labels and the labels that are generated internally by the VisualMotion compiler.

Base Code instruction mnemonics and valid arguments

The following lists the Base Code instruction mnemonics and valid arguments. Instructions requiring more than one argument show the arguments separated by commas. Alternative forms for arguments are shown by enclosing a general form for each argument in square brackets, separated by a vertical bar "|".

ABORT_PATH

ABORT_PATH [task]

Halts coordinated motion in the specified task.

ACCEL

ACCEL [axis | label | Ix | Glx],[rate | label | Fx | Gfx]

Sets the acceleration rate for the specified axis.

AXES

AXES task mode, axis,

Specifies the axes to be assigned to this task and how they will be used.
(All axes used in a task must be declared for that task.)

Task mode:

1. for single axis non-coordinated motion.
2. for coordinated axis for multi-axis coordinated motion.
3. for velocity mode, rotation only - no axis positioning.
4. for ratioed slave axis.
5. for ELS mode.
6. for Torque mode.

Axis = a valid identifier for an axis, from 1 to the maximum number of axes

AXES_GROUP

AXES_GROUP task mode, axis, axis, axis, axis, axis, axis

AXIS_EVENT

AXIS_EVENT [axis | label | Ix | Glx], event #1, event #2, event #3, event #4

Enables up to four Repeating Position events for a single-axis, ELS, ratio, or velocity mode axis.

Event # = [an integer Event # | label | Ix | Glx]

AXIS_ATPOSITION**AXIS_ATPOSITION** *axis, position*

where

Axis = drive number**Position** = position to wait for**AXIS_WAIT****AXIS_WAIT** *[axis / label / Ix / Gix]*

If the argument is a positive integer representing a valid axis, program execution waits for the axis to be within its preset drive position window. If the argument is -1, program execution waits until all axes in the task are within their position windows. The position window is defined by the drive parameters: Position Window and Zero Velocity. **AXIS_WAIT** will wait indefinitely if used with velocity mode (axis task mode 3) since positioning is not used.

BNE**BNE** *label (subroutine or event)*

Branches to label if the task's status word is set to "not equal."

BEQ**BEQ** *label (subroutine or event)*

Branches to label if the task's status word is set to "equal."

BGT**BGT** *label (subroutine or event)*

Branches to label if the task's status word is set to "greater than."

BLT**BLT** *label (subroutine or event)*

Branches to label if the task's status word is set to "less than."

BGE**BGE** *label (subroutine or event)*

Branches to label if the task's status word is set to "greater than or equal."

BLE**BLE** *label (subroutine or event)*

Branches to label if the task's status word is set to "less than or equal."

BRA**BRA** *label (subroutine or event)*

Branch to label, always (no matter what)

CALL_FUNC**CALL_FUNC** *func_offset, ret_pointer, arg_count, arg1_ptr, ... argn_ptr*

Calls the function at func_offset with a return pointer and a variable number of arguments.

func_offset	offset in bytes from current program counter to start of function
ret_pointer	pointer to int or float return variable if (0), there is no return value

arg_count	number of arguments passed to the function if (0), there are no arguments there can be between 1 and 5 arguments
arg1_ptr... argn_ptr	pointer to argument passed to function can be int, float, global int, global float, constant int, constant float, local int, local float, absolute or relative point label used as initial value of local variable

CALC*CALC evaluates the equation***CAP_ENABLE***CAP_ENABLE axis, probe, event#*

Enables the event on the axis for the probe transition. When the transition occurs, the event triggers.

Axis = from 1 to the maximum number of axes**Probe:**

- 1 = probe 1, 0 --> 1
- 2 = probe 1, 1 --> 0
- 3 = probe 2, 0 --> 1
- 4 = probe 2, 1 --> 0

Event = [event # | label | Ix | GIx]**CAP_SETUP***CAP_SETUP axis, probe*

At program activation, the drive is configured to capture feedback position on its probe transition and to include position data in its cyclic telegram data.

Axis = from 1 to the maximum number of axes**Probe:**

- 1 = probe 1, 0 --> 1
- 2 = probe 1, 1 --> 0
- 3 = probe 2, 0 --> 1
- 4 = probe 2, 1 --> 0

CLEAR*CLEAR [Ix / GIx / Fx / GFx / label]*

Sets integer or float variable to zero

COMP*COMP [Ix / GIx / Fx / GFx / label], [Ix / GIx / Fx / GFx / label]*

Set the task's status word to the logical result of 1st argument minus 2nd argument.

DATA_SIZE*DATA_SIZE I, F, ABS, REL, EVT, ZONE*

Sets the amount of memory allocated for each type of data in one of the four program tasks. (The total program requirement is the sum of the DATA_SIZE allocations for each task in the program.)

I = the number of integer variables allocated for this task
F = the number of floating point variables allocated for this task
ABS = the number of absolute point table entries allocated for this task
REL = the number of relative point table entries allocated for this task
EVT = the number of event table entries allocated for this task
ZONE = the number of zone table entries allocated for this task

DEC

DEC [Ix | GIx | label]

Subtracts 1 from the specified integer variable

DECEL

DECEL axis, rate

Sets the deceleration rate for the axis

Axis = [integer constant | label | Ix | GIx]

Rate = [floating point constant | label | Fx | GFx]

ELS_ADJUST

ELS_ADJUST axis, offset

Sets the phase or velocity offset for the ELS axis.

Axis = [integer constant | label | Ix | GIx]

Offset = [floating point constant | label | Fx | GFx]

ELS_ADJUST1

ELS_ADJUST1 axis, offset, type

Axis = [integer constant | label | Ix | GIx]

Offset = [floating point constant | label | Fx | GFx]

Type:

1 = absolute

2 = incremental

3 = continuos +

4 = continuos -

ELS_GROUPM

ELS_GROUPM group number, control register, status register, float block, integer block

where

Group number = 1 to 8

ELS_GROUPS

ELS_GROUPS group number, axis number, motion type

where

Group number = 1 to 8

Axis number = 1 to 32

Motion type:

1 = phase

2 = velocity

3 = card cam

4 = drive cam

ELS_INIT

ELS_INIT *ELS type, slave axis, master axis, encoder, sync type*

Initializes the relationship between master and slave axes.

ELS type:

1 = Virtual Master

2 = Real Master (daisy-chained)

3 = Real Master (SERCOS)

4 = follow axis feedback

Slave axis = [integer constant | label]

Master axis = [integer constant | label]

Encoder:

1 = primary encoder

2 = secondary encoder

Sync type:

1 = velocity

2 = phase

ELS_MASTER

ELS_MASTER *float block, integer block*

ELS_MODE

ELS_MODE *axis, mode*

Sets the mode for the specified ELS axis.

Axis = [integer constant | label | Ix | GIx]

Mode:

1 = single axis

2 = ELS synchronization

ELS_STOP

ELS_STOP

END

Defines the end of the program for the task.

EVENT_DONE

EVENT_DONE *event*

Marks the specified event status as complete.

Event = [integer constant | Ix | GIx | label]

EVENT_ENABLE

EVENT_ENABLE *event*

Activates the specified repeating timer event.

Event = [integer constant | Ix | GIx | label]

EVENT_END

Defines the end of an event routine program code.

EVENT_START

Marks the beginning of an event routine program code.

EVENT_WAIT*EVENT_WAIT event*

Pauses task execution until the specified active event completes.

Event = [integer constant | Ix | GIx | label]**FUNC_ARG**

func_label: FUNC_ARG label, type, <min value>, <max value>

Declares local variables.

func_label	text label of function
Label	text string identifier of local variable
Type	'F'=float, 'I'=integer, "ABS"= ABS point index, "REL"= REL point index
min value	optional minimum value of argument
max value	optional maximum value of argument

FUNC_END

func_label: FUNC_END return value

Indicates the end of a function and optional return value.

Func_label	text label of function
Return value	return argument

FUNC_START

func_label: FUNC_START

Indicates the start of the function named by 'func_label'.

Func_label	text label of function
------------	------------------------

GET_PARAM*GET_PARAM type, set, ID number, destination*

Copies the specified parameter data to the specified integer or floating point variable (the variable type must match the parameter type).

Type:

A = axis

C = system

D = drive

T = task

Set = axis or drive ([integer constant | Ix | label]), or task ID letter**ID number** = identifying parameter number (range 1 to 65535)**Destination** = destination variable, [Ix | GIx | Fx | GFx | label]**GO**

GO axis

Starts continuous motion on the axis. The axis must be configured as non-coordinated or velocity mode.

Axis = [integer constant | Ix | GIx | label]

HOME*HOME axis*

Enables motion homing the specified axis. (The homing parameters must have been set in the DDS drive.)

Axis =[integer constant | Ix | GIx | label]**INC***INC [Ix | label]*

Adds 1 to the specified integer variable.

LOCAL/VAR

func_label: LOCAL/VAR label, type

Declares local variables.

Func_label	text label of function
Label	text string identifier of local variable
Type	'F'=float, 'I'=integer

MESSAGE*MESSAGE type, message, variable*

where

Type:

1 = status

2 = diagnostic

Message = Up to 80 characters**Variable** = Fx, Ix, GFx, Gix**MESSAGE_PORT**

func_label MESSAGE_PORT target, string, <pointer>

Outputs formatted string to designated port.

Func_label	text label of function
Target	1 = diagnostic message. 2 = status message. 3 = serial host port(Port A). 4 = serial teach pendant port(Port B).
String	Formatted text string to display, formatting types are %d, %f, %x
pointer	Optional single argument - Rx, Fx, Ix, GFx , or Gix

MOVE_JOINT*MOVE_JOINT ABS point*

Moves the joint based on an absolute point (six-axis CLC only.)

ABS point = [integer constant | Ix | GIx | label], an entry in the absolute point table

KINEMATIC*KINEMATIC kinematics library number*

Selects the set of equations specified by the library number from an optional kinematics library. Used to translate Cartesian coordinates for custom coordinated motion applications such as robotics.

MOVEA_AXIS

MOVEA_AXIS axis, distance, event, event, event, event

Starts single axis absolute motion for the specified axis, and activates the specified events.

Axis = [integer constant | Ix | Glx | label]

Distance = [floating point constant | Fx | GFx | label]

Event = [integer constant | Ix | Glx | label]

MOVER_AXIS

MOVER_AXIS axis, distance, event, event, event, event

Starts single axis relative motion for the specified axis, and activates the specified events. An event is specified by an integer number index into the event table or a label for an integer variable containing the index.

Axis = [integer constant | Ix | Glx | label]

Distance = [floating point constant | Fx | GFx | label]

Event = [integer constant | Ix | Glx | label]

MOVEA_PATH

MOVEA_PATH ABS point

Starts coordinated motion from the current position to the point specified in the absolute point table.

ABS point = [integer constant | Ix | Glx | label]

MOVER_PATH

MOVER_PATH ABS point, REL point

Starts coordinated straight line motion from the current position to the point specified by the vector sum of the absolute and relative points.

ABS point = [integer constant | Ix | Glx | label]

REL point = [integer | Ix | Glx | label]

MOVEA_CIRCLE

MOVEA_CIRCLE ABS point, ABS point

Starts coordinated motion from the current position, through the first specified point, ending at the second specified point.

ABS point = [integer constant | Ix | Glx | label]

MOVER_CIRCLE

MOVER_CIRCLE REL point, REL point, ABS point

Starts coordinated motion from the current position, through the point specified by the vector sum of the ABS point and the first REL point, ending at the point specified by the vector sum of the ABS point and the second REL point.

ABS point = [integer constant | Ix | label]

REL point = [integer constant | Ix | label]

MSG_DIAG

MSG_DIAG ASCII text string

Sets the current diagnostic message to the specified ASCII text string.

MSG_STATUS*MSG_STATUS ASCII text string*

Sets the current status message to the specified ASCII text string.

PARAM_BIT*PARAM_BIT type, set, ID number, source, I/O mask*

Sets the parameter bit specified by the type, set, ID number and I/O mask to the value in the specified source variable at initialization.

Type:

A = axis

C = system

D = drive

T = [A | B | C | D] (task ID letter)

Set = [integer constant | Ix | GIx | label] for axis or drive; or [A | B | C | D] for task**ID number** = [integer constant] for a parameter number in the range 1 to 65535**Source** = [integer constant | floating point constant | Ix | GIx | Fx | GFx | label]**I/O mask** = specifies 1 to 16 bits in an I/O register**PARAM_INIT***PARAM_INIT type, set, ID number, source*

Sets the specified parameter to the value in the specified variable at initialization.

Type:

A = axis

C = system

D = drive T = [A | B | C | D] (task ID letter)

Set = [integer constant | Ix | GIx | label] for axis or drive; or [A | B | C | D] for task**ID number** = [integer constant] for a parameter number in the range 1 to 65535**Source** = [integer constant | floating point constant | Ix | GIx | Fx | GFx | label]**PID_CONFIG***PID_CONFIG #, type, control_register, status_register, loop_time, set_point_type, set_point, set_point_axis, feedback_type, feedback, feedback_axis, output_type, output, output_axis, control_block***#:** PID loop number, range 1-10.**type:** PID loop type, currently only 1 is valid.**control_register:** label or number of register used for control of this loop.**status_register:** label or number of register used for status of this loop**loop_time:** update time of this loop, multiples of 8 millisecond.**set_point_type:** Type of set point, 1=variable, 3=unsigned register, 4=signed register**set_point:** Axis parameter, register, variable, or equivalent label to be used as the set point of this loop.**set_point_axis:** For axis parameters, axis number; else 0.

feedback_type: Type of feedback, 1=variable, 2=axis parameter, 3=unsigned register, 4=signed register

feedback: Axis parameter, register, variable, or equivalent label to be used as the feedback of this loop.

feedback_axis: For axis parameters, axis number; else 0.

output_type: Type of output, 1=variable, 2=axis parameter, 3=unsigned register, 4=signed register

output: Axis parameter, register, variable, or equivalent label to be used as the output of this loop.

output_axis: For axis parameters, axis number; else 0.

control_block: First variable in a block of 20 float variables(Fx) to be used for this loop.

See also **VAR_INIT**.

PLS_INIT

PLS_INIT switch number, 0, output register, master type, axis/number, offset

PLS_INIT switch number, element, on position, off position

where

Switch number = 1

Element = 1 to 16

On position = 0 to 360

Off position = 0 to 360

Offset = 0 to 360

Axis/number = drive number if drive based / 1 or 2 if real master

Master type:

1 = ELS

2 = Virtual

3 = Real(1 or 2)

4 = drive based

PLS1_INIT

PLS1_INIT switch number, 0, output register, master type, number, offset, mask register

PLS1_INIT switch number, element, on position, off position, lead time

where

Switch number = 1

Element = 1 to 16

On position = 0 to 360

Off position = 0 to 360

Lead time = 0 to cycle time

Number = ELS Master or ELS Group number

Master type =

5 = ELS Master

6 = ELS Group

POSITION

POSITION task, ABS point

Copies the current position coordinates of the specified task to the specified ABS point table entry. The contents of the point table entry are overwritten.

Task = [A | B | C | D]

ABS point = [integer constant | Ix | GIx | label]

RATIO

RATIO master axis, slave axis, master ratio, slave ratio

Sets the ratio between the specified master and slave axes.

Master axis = [integer constant | Ix | GIx]

Slave axis = [integer constant | Ix | GIx]

Master ratio = [floating point constant | Fx | GFx]

Slave ratio = [floating point constant | Fx | GFx]

READ

READ register, count, target variable

Copies the contents of the specified I/O register(s) to the lower 16 bits of the specified integer variable(s). The upper word of the variable(s) is zero-filled. Only a contiguous block of registers can be moved.

Register = an integer constant specifying the number of the starting source I/O register.

Count = a positive integer constant for the number of register to copy
target variable = the starting integer variable table entry for the destination of the data.

RESUME_PATH

RESUME_PATH task

Restarts previously halted coordinated motion in the specified task.

Task = [A | B | C | D]

RETURN

Marks the end of a subroutine's program code, and returns program execution to the calling program.

ROBOT_ORGIN

ROBOT_ORIGIN point

where

Point = relative point index to be used as origin

ROBOT_TOOL

ROBOT_TOOL point

where

Point = relative point index to be used as tool offset

ROTARY_EVENT

ROTARY_EVENT type, axis, event1, event2, event3, event4 (GPP)

Event1 = index of event to trigger

Event2 = index of event to trigger

Event3 = index of event to trigger

Event4 = index of event to trigger

Type = drive number if drive based / ELS Master or ELS Group number

0 = Drive

1 = ELS Master

2 = ELS Group

Axis = drive number if drive based / ELS Master or ELS Group number

SET

SET I/O state, register, I/O mask

Sets the specified register's bits, that are enabled by the I/O mask, to the state specified by I/O state.

I/O state = 16 bit binary word of bits to set in the specified register. 0 = off, 1 = on.

Register = an integer number specifying an I/O register

I/O mask = 16 bit binary word specifying the bits that may be changed. 1 = enabled

SET_PARAM

SET_PARAM type, set, ID number, source

Copies the specified parameter's value to the specified integer or floating point variable. The source variable data type must match the destination parameter data type.

Type:

A = axis

C = system

D = driveT = task

Set = [Ix | GIx | label] for axis or drive; or [A | B | C | D] for task ID letter]

ID number = identifying parameter number, within the range: 1 to 65535

Source = [integer constant | floating point constant | Ix | GIx | Fx | GFx | label]

START

Marks the beginning of a task or subroutine.

STOP

STOP axis

Signals the drive to halt single-axis or velocity mode motion on the specified axis if the argument is a positive integer (1 - 8). If the argument is -1, motion is halted for all single-axis and velocity mode axes in the task. Signaling the drive to halt motion decelerates the axis to zero velocity using the deceleration rate programmed in the appropriate drive parameter.

Axis = [integer constant | Ix | GIx | label]

STOP_PATH

STOP_PATH task

Stops coordinated motion in the specified task.

Task = [A | B | C | D]

TEST

TEST register, I/O mask

Sets the task's status word to the result of a logical AND of the specified register and the I/O mask.

Register = a positive integer constant for a modifiable CONTROL register.

I/O mask = 16 bit binary word.

V_MASTER

V_MASTER number, control register, status register, float block, integer block

where

Number = 1 to 6

VAR_INIT

VAR_INIT ar_start, arg1, arg2, arg3... arg20

var_start: First variable in a block of program variables(Fx, Ix) to be initialized.

arg1- arg20: initializing values.

VEL

VEL axis, rate

Sets the velocity specified by rate in the specified task axis.

Axis = [integer constant | Ix | GIx | label]

Rate = [floating point constant | Fx | GFx | label]

WAIT

WAIT delay

where

Delay = 1 to 32767 msec

WAIT_IO

WAIT_IO register, I/O mask, I/O state

Suspends task execution until the specified I/O conditions are met.

Register = an integer constant for a CONTROL register

I/O mask = identifies 1 to 16 bits in an I/O register

I/O state = 0 --> off; non-zero --> on

WAIT_PATH

WAIT_PATH task, ABS or REL point, condition

Suspends task execution until the specified path planner conditions are met.

Task = [A | B | C | D]

ABS point = [integer constant | Ix | GIx | label], reference to an absolute table entry

REL point = [integer constant | Ix | GIx | label], reference to a relative table entry

Condition:

0 = Ready

1 = Accel

- 2 = Slew
- 3 = Blending
- 4 = Target decel
- 5 = Controlled stop
- 6 = Stopped
- 7 = At target
- 8 = Done

WRITE

WRITE *register, count, source*

Copies the data in the specified integer variable(s) to the specified I/O register.

Register = an integer number for the starting destination I/O register.

Count = a positive integer constant for the number of registers to copy.

Source = an integer number for the starting source integer variable table entry.

12.3 Icon Language Warnings and Error Messages

VisualMotion Icon Compiler generates the following warning messages. After receiving a warning message you may continue or exit the compilation.

- Data missing in one or more fields, do you still wish to continue?
- Caution! Changing Modes may halt motion. Continue?
- Caution! Changing Modes may start motion. Continue?
- File does not contain source program!
- Icon workspace at end is not empty. Program parts will be lost, continue anyway?
- Transfer failed!

VisualMotion's Icon Compiler displays the following error messages:

- More than one connect icon with number %d!
- Function variables must be defined first!
- Data Size icon objects exceed size of non-volatile ram!
- Change to default registers and variables for this number?
- Only one ELS System Master icon allowed per program!
- Only one Virtual Master icon allowed per program!
- Only eight ELS Groups allowed!
- Only ten PIDs allowed!
- Only four CAM Indexers allowed!
- Warning! Frequent changes of static drive parameters can cause premature failure of it's non-volatile memory.
- Valid Entries are '0' or '1'.
- Invalid name!
- Cannot change task or open dialog window while dialog window is open
- Axis undefined or not unique.
- Valid event numbers are 1 to 100.

- Valid axis numbers are 1 to 8.
- Valid number range is 1 - 32767.
- Valid percents are 1 - 100.
- Labels must start with an alpha character!
- Label name already exists!
- Number missing or out of range.
- Selected Icon is not a subroutine or no icon selected!
- Data Field Empty!
- Label type must be defined!
- Task name undefined.
- No filename specified.
- Non-Branch icons have only two output connections.
- Branch icons have only two output connections.
- Point out of range.
- Connection could not be made, try connecting adjacent ---?
- Only connections between icons or adjacent blocks can be ---?
- Finish icon not found or open path!
- Start icon not found or multiple Start icons found!
- Icon program not found!
- Cannot open code file!
- Unknown icon term _____.
- Missing axis selection.
- Open in program flow, at or near highlighted icon, ---?
- Branch Icon has missing connection or one in wrong dire ---?
- No axis selected!
- Time Delay out of range!
- Could not initialize update timer!
- Operation type not selected!
- Drive numbers doesn't match.
- Should drive number be c ---?
- Can't open file _____!
- Source or target not selected!
- Valid range ____ - ____
- Valid range ____ - ____
- CONTROL card parameters cannot be changed!
- File syntax other than parameters!
- File of different type parameters!
- CONTROL card is not communicating!
- No selection made!

12.4 Text Language Error Messages

The following are error messages produced by the Visual Motion text compiler. Line numbers refer to code displayed by selecting "Display Code" from the Visual Motion Build menu. For further information on the format of the code displayed see Control Compiler Base Code.

First Pass Errors

- CONTROL code converter error log file!
- Unable to open source file!
- Line [nnn], Maximum number of terms reached!
- Line [nnn], unknown mnemonic operator - [xxx]
- Line [nnn], unknown, missing or wrong argument - [xxx]
- Line [nnn], missing point argument!
- Line [nnn], missing closing bracket "]"!
- Line [nnn], additional arguments - [xxx...!]!
- Line [nnn], point number '??' out of range (1-nn)!
- Line [nnn], missing arguments!
- Line [nnn], unknown IF conditional terms - [?] [?]!
- Line [nnn], ELSE or ENDIF without IF term!
- Line [nnn], maximum number of nested IFs exceeded!
- Line [nnn], sequencing error, IF, ELSE, or ENDIF imbalanced
- Line [nnn], missing message text!
- Line [nnn], incompatible circle arguments - [xxx]
- Line [nnn], variable out of range - [variable name]
- Line [nnn], right side of EQU must be a number - [____]!
- Line [nnn], label [label name] not found
- Line [nnn], arguments must be integer or constant!
- Line [nnn], bit number [nn] out of range (1-nn)!
- Line [nnn], register number [nn] out of range (1-nn)!
- Line [nnn], integer variable number [nn] out of range (1-nn)!
- Line [nnn], register number + count exceeds range (1-nn)!
- Line [nnn], axis number [nn] out of range (1-n)!
- Line [nnn], mode number [nn] out of range (0-n)!
- Line [nnn], mark "____" also defined on line [nn]!
- Mark [____] on line [nnn] was not referenced in program!
- Mark [____] used on line [nnn] is not declared!
- Line [nnn], event number [nn] out of range (1-nn)!
- Line [nnn], delay value [n...] out of range (1-n...)!
- Line [nnn], too many arguments!

Second Pass Compiler Errors

Line xx, more than one equal operator.

On line xx, more than one “=” character was found.

Line xx, colon used for other than mark!

A colon was found beyond the first word on line xx.

Line xx, function start found inside of subroutine!

A Start icon was found inside a subroutine on line xx.

Line xx, function end found without function start!

A Finish icon was found without first finding a Start icon on line xx.

Duplicate local argument 'xx' found in subroutine 'yy'!

Two local arguments with the same name xx were found in subroutine yy.

Subroutine 'xx' has more than 5 user accessible arguments!

A subroutine can only have 5 arguments passed to it. Subroutine xx has more than 5.

Subroutine 'xx' has more than 16 local variables/arguments!

A subroutine can only have 16 local or stack variables. Subroutine xx has more than 16.

Line xx, invalid sequencer list index 'xx'!

An error was made while defining a sequencer on base code line xx. One of the sequencer "list_Numbers" is greater than 30 or has been entered out of sequence (0,1,2,3,4,5).

Line xx, invalid sequencer step index 'yy'!

An error was made while defining a sequencer step on base code line xx. One of the sequencer steps "step_Numbers", yy, is greater than sequencer functions defined in the "DATA/SIZE instruction of the program or has been entered out of sequence (0, 1,2,3,4,5).

Number of sequencer step names exceed sequencer step size!

The number of sequencer steps found is greater than sequencer Steps defined in the "DATA/SIZE instruction of the program.

Number of sequencer names exceed sequencer list size!

The number of sequencer names found is greater than sequencer Lists defined in the "DATA/SIZE instruction of the program.

Line xx, invalid axis number - yy!

An error was found in the PLS/INIT instruction on base code line xx. The "axis" number yy is not valid for the type selected. Valid ranges are:

Type	Range
1 or 2	1-2
3 or 4	1-32
5	1-6
6	1-8

Line xx, invalid PLS master type - yy, range 1 – 4.

An error was found in the PLS/INIT instruction on base code line xx. The "type" yy is not a value from 1 to 4.

Line xx, duplicate label 'yy' or multiple definition of variable!

An error was found in the FUNCTION/ARG instruction or START icon on base code line xx. The label yy was used already.

Line xx, error in number of function arguments - yy!

An error was found in the CALL instruction or SUB icon on base code line xx. The number of arguments yy, passed to the subroutine is different than defined in the function called.

Line xx, index 'yy' is a float!

An error was found on base code line xx. The index used to a variable is a float (i.e. F[F5]).

Line xx, two names assigned to a sequencer 'yy'!

An error was made while defining a sequencer on base code line xx. A sequencer index 'yy' is given two different names.

Line xx, same name assigned to two sequencers 'yy'!

An error was made while defining a sequencer on base code line xx. The same name 'yy' is given to two sequencers.

Line xx, invalid cam option type 'yy'!

An error was made while defining the CAM/BUILD instruction on base code line xx. The cam option or type 'yy' is outside the range 1-4.

Line xx, end point 'yy' is less than start point!

An error was made while defining the CAM/BUILD instruction on base code line xx. The point defined as the end_point 'yy' is less than the start point.

Line xx, invalid cam wait option - yy, range 0 - 1

An error was made while defining the CAM/BUILD instruction on base code line xx. The wait option 'yy' is outside the range 0-1.

Maximum number of messages reached!

The number of messages, status and diagnostic, allowed per program is 500. An attempt to exceed this was found.

Multiple PLS initializations found!

More than one instruction was found to define the same PLS data .

Line xx, invalid message type, range 1 - 2

An error was made in the MESSAGE instruction on base code line xx. The valid range of values are 1-2.

Line xx, invalid cam type 'yy'! 0=CLC, 1=Drive.

An error was made in the CAM/ENGAGE instruction on base code line xx. The value 'yy' is invalid, valid range of values is 0-1.

Line xx, ELS slave 'yy' same as master!

An error was made in the ELS/INIT instruction on base code line xx. The slave axis 'yy' is the same as the master axis.

Line xx, invalid PID number 'yy', range 1 - 10

An error was made in the PID/CONFIGURE instruction on base code line xx. The loop number 'yy' is invalid, range is 1-10.

Line xx, invalid PID type 'yy', range 1 - 1

An error was made in the PID/CONFIGURE instruction on base code line xx. The type 'yy' is invalid, the only type available is 1.

Line xx, same PID status and control registers 'yy'

An error was made in the PID/CONFIGURE instruction on base code line xx. The same register number 'yy' was used for the control and status, they must be different.

Line xx, invalid PID loop time 'yy', range 8 - 152**Line xx, PID loop time 'yy', not multiple of 8**

An error was made in the PID/CONFIGURE instruction on base code line xx. Loop times are multiples of 8 ms, from 8 to 152. The value 'yy' is not valid.

Line xx, Data initialization 'yy', exceeds data range**Line xx, variable block 'yy' exceeds variable allocation!**

An error was made in the VAR/INIT instruction on base code line xx. An attempt was made to initial variables beyond their range with 'yy'. Increase size of variables in DATA/SIZE instruction.

Line xx, Multiple configurations For PID loop yy

An error was made in the PID/CONFIGURE instruction on base code line xx. More than one initialization found for PID 'yy'.

Line xx, PID control blocks overlapping 'yy'

An error was made in the PID/CONFIGURE instruction on base code line xx. A float variables control block overlaps another.

Line xx, invalid PID argument 'yy'

An error was made in the PID/CONFIGURE instruction on base code line xx. Invalid set_point_type, feedback_type, or output_type found 'yy', valid range 1-4.

Line xx, zone element 'yy' missing or entered with spaces!**Line xx, zone element 'yy' unknown!**

An error was defining a zone element instruction on base code line xx. Invalid text found was 'yy'.

Line xx, Missing open parenthesis!

An error was found in a mathematical expression on base code line xx. A closed parenthesis found without matching open.

Line xx, invalid ELS Group number 'yy', range 1 – 8

An error was made in the ELS_GROUP instruction on base code line xx. Invalid group number found 'yy', valid range 1-8.

Line xx, multiple ELS Master instructions found!

A second ELS_MASTER instruction was found on base code line xx. Only one ELS_MASTER instruction is allowed per program.

Line xx, multiple ELS Group 'yy' instructions found!

A second ELS_GROUP instruction for group 'yy' was found on base code line xx. Only one ELS_GROUP instruction per group is allowed in a program.

Line xx, axis 'yy' found in multiple ELS Group instructions!

A second ELS_GROUP instruction for axis 'yy' was found on base code line xx. An axis can only be assigned to one ELS_GROUP.

Line xx, invalid ELS Master number 'yy', range 1 - 6

An error was made in the ELS_MASTER instruction on base code line xx. Invalid master number found 'yy', valid range 1-6.

Line xx, Valid modes are 0=axis, 1=ELS Master, 2=ELS Group !

An error was made in the ROTARY/EVENT instruction on base code line xx. Valid modes are 0=axis, 1=ELS Master, 2=ELS Group

Line xx, invalid Virtual Master number 'yy', range 1 - 2

An error was made in the V_MASTER instruction on base code line xx. Master number 'yy' is not in the range 1-2.

Line xx, Illegal syntax : syntax 'yy' is not allow at the moment.

An error was made in the mathematical equation instruction or Calc icon on base code line xx. Syntax 'yy' is not allowed in this sequence of terms.

'xxxx' - unresolved mark reference.

The mark 'xxxx' was used as a destination in a branch or subroutine call, but was not found in the code. Check for possible spelling error or missing subroutine.

Line xx, all probe types zero or not unique!

The probe arguments are both zero or are the same.

Line xx, argument 'yyyy' out of range!

The argument 'yyyy' is out of range, check syntax in manual.

Line xx, axes missing or not unique!

In a AXES_GROUP command for ratioed axis, the slave axis argument is zero or is the same as the master axis.

Line xx, axis number 'yyyy' out of range (www, xxxx, 1-zzzz).

The axis number or label 'yyyy' has not been resolved to a valid number. The numbers 'www', 'xxxx', and range 1 to 'zzzz' are valid axis numbers.

Line xx, bit number 'yyyy' out of range (1-16)!

On line 'xx', the string 'yyyy' is evaluated to number outside of the valid range for register bits.

Line xx, 'compare' arguments must be floats, integers, or constants!

Compare arguments must be Fx, GFx, Glx, Ix or equivalent labels or constants. Comparisons are derived from "IF" statements in textual language programs or "BRANCH" icons in GUI programs.

Line xx, event element 'yyyy' missing or entered with spaces!

On line 'xx', the compiler has not found a "]" in the event string 'yyyy'. It uses this to position to the start of the event element. The event element { s, t, d, a, f, m }must follow immediately.

Line xx, event element 'yyyy' unknown!

The event element 'yyyy' was not found in the event element table, check manual for exact syntax.

Line xx, event EVT[].yy data is not changeable in program**Line xx, event function 'yyyy' not found in program!**

The event function 'yyyy' was not found in the program. Check spelling and capitalization.

Line xx, event message 'yyyy' must start with quotes!

The compiler is expecting a quote to start the ASCII string for the event message, but did not find it.

Line xx, event number 'yyyy' out of range!

On line 'xx', the string 'yyyy' was evaluated to be out of the range for events defined for this program. Events and other variables are declared in the "DATA/SIZE" command in a textual language program or by the "SIZE" icon in GUI programs.

Line xx, float number 'yyyy' conversion error!

The string 'yyyy' for conversion to a float was determined to contain one of the following errors:

No numeric characters.

More than one exponent symbol 'E' ('e').

More than two sign symbols'.

More than one decimal point.

Alpha characters other than 'E' ('e').

Line xx, hex number 'yyyy' conversion error!

On line 'xx', the string 'yyyy' is greater than 10 characters long or contains non-hexadecimal characters. Valid strings start with 0x and contain ASCII characters 0-9, A-F or a-f (0x1BF8).

Line xx, integer number 'yyyy' conversion error!

The string 'yyyy' for conversion to an integer was determined to contain one of the following errors:

No numeric characters.

Number of numeric characters exceed 10.

The converted number exceeds 0xFFFFFFFF.

Line xx, Invalid argument 'yyyy'!**Line xx, Invalid cam number 'yyyy'! Range 1 to 8.**

The CAM number 'yyyy' was evaluated to be less than one or greater than 8.

Line xx, Invalid count or count plus register exceeds range!

The count of registers to be transferred was evaluated to be less than one or when added to the starting register exceeds the maximum register range (512 registers for GPP).

Line xx, Invalid Encoder type 'yyyy', 1=primary, 2=secondary!

The ELS master encoder type 'yyyy' was evaluated to be less than one or greater than 2.

Line xx, Invalid ELS type 'yyyy', range 1 to 4!

The ELS type 'yyyy' was evaluated to be less than one or greater than 4.

Line xx, Invalid sync type 'yyyy', 1=velocity, 2=phase, 3=cam!

The ELS sync type 'yyyy' was evaluated to be less than one or greater than 3.

Line xx, Invalid VME Address 'yyyy'!

The VME address 'yyyy' was evaluated to be less than one or greater than 0xFFEFFFFF.

Line xx, Invalid VME address width 'yyyy'!

The address width 'yyyy' was not found in the table of VME address widths.

{"A16", "A24", "A32"}

Line xx, Invalid VME byte order 'yyyy'!

VME byte order 'yyyy' must start with 'I' or 'M', 'I' is for Intel order, 'M' is for Motorola. It can be a single character or the name, Intel or Motorola. It is case sensitive, so 'I' and 'M' must be capitalized.

Line xx, Invalid VME count 'yyyy'!

The count of VME objects to transfer 'yyyy' was evaluated to be less than one or greater than 32767.

Line xx, Invalid VME data width 'yyyy'!

The data width 'yyyy' was not found in the table of VME bus widths.

{"D32", "D16", "D8"}

Line xx, Invalid VME data format 'yyyy'!

The data format 'yyyy' was not found in the table of VME data formats.

{"I32", "I16", "I8", "U32", "U16", "U8", "F32", "POINT"}

Line xx, Left term 'yyyy' of equation must not be constant!

A calculation must have a variable(Fx, GFx, GIx, Ix) or changeable table element(ABS[1].x, EVT[3].d, etc.) as its term to the left of the equal sign.

Line xx, Maximum number of terms reached.

When parsing the line 'xx', the number of terms exceeded 32. A term is one or more alphanumeric characters followed by a space, comma or other non-alphanumeric character. This error usually only occurs in

message statements with many short words. Try a message with fewer words.

Line xx, Maximum size (20) of term exceeded!

While parsing line 'xx' for arguments a string of more than 20 characters was encountered. Arguments and argument labels are limited to 20 characters. Check label length and use of commas between arguments.

Line xx, Message exceeds 80 characters!

The number of characters used in the message exceeds 80 characters. This count includes spaces.

Line xx, missing argument(s)!

One or more additional arguments were expected.

Line xx, missing beginning quotes of message!

On line 'xx', quotes were expected to denote the start of the message. Diagnostic, status and event messages are specified within quotes in textual language programs. Also, use quotes when using the "CALC" icon to set an event message.

Line xx, missing closing bracket ']'!

The closing bracket used to denote the end of the index of a data structure was not found.

Line xx, missing closing curly brace '}'!

The closing brace used to denote the end of initialization data for a data structure was not found. Other causes are extra arguments or the wrong character.

Line xx, missing closing quotes of message!

On line 'xx', quotes were expected to denote the end of the message. Diagnostic, status and event messages are specified within quotes in textual language programs. Also, use quotes when using the "CALC" icon to set an event message.

Line xx, missing mark name!

The argument of branch command does not start with an alpha character. Check for missing or misspelled argument.

Line xx, Parameter <type> must be 'A', 'C', 'D' or 'T'

The parameter class was not found to be 'A', 'C', 'D', 'T', or equivalent label. Check for missing or misspelled argument.

Line xx, point element 'yyyy' missing or entered with spaces!

On line 'xx', the compiler has not found a "]" in the point string 'yyyy'. It uses this to position to the start of the point element. The point element {x, y, z, b, s, a, d, j, e1, e2, e3, e4, r, p, ya, el} must follow immediately.

Line xx, point element 'yyyy' unknown!

The point element 'yyyy' was not found in the point element table, check manual for exact syntax.

Line xx, register number 'yyyy' out of range (1-zzzz)!

The register number 'yyyy' is less than one or greater than the maximum register 'zzzz'.

Line xx, table or array index out of range 'yyyy'!

The table or array index 'yyyy' is less than one or greater than number declared by DATA/SIZE command or by the default declaration.

Line xx, table or array label index out of range 'yyyy'!

The table or array index label 'yyyy' is evaluated to be less than one or greater than number declared by DATA/SIZE command or by the default declaration.

Line xx, Task must be 'A', 'B', 'C' or 'D'!

The compiler is expecting a task argument (A, B, C, or D) and has not found it. This may result from a missing argument or arguments out of sequence.

Line xx, too many arguments!

More terms than expected were found following the command. Check for extra arguments, extra commas or terms with spaces in them.

Line xx, unknown mnemonic operator - 'yyyy'!

On line 'xx', the string 'yyyy' is assumed to be a command, but was not found in the list of valid commands. This error is most often generated from textual language programs when the command is misspelled or from incorrect syntax.

Line xx, unknown or out of range variable 'yyyy'!

On line 'xx', the string 'yyyy' is not of the type expected. Check for argument type(float where integer should be used, etc.), or for missing or misspelled arguments.

Line xx, unresolved index 'yyyy'!

The index 'yyyy' could not be resolved, check for missing or misspelled label. Labels are case sensitive and cannot contain spaces.

Line xx, unresolved index label 'yyyy'!

The index label 'yyyy' could not be resolved to an integer or integer variable, check for missing or misspelled label. Labels are case sensitive and cannot contain spaces.

Line xx, unsupported data transfer! VME bus width 'yyyy', VME format 'zzzz', local

The selected VME data transfer is not supported in this product. Check VME format and local format for possible erroneous selection.

Line xx, Unsupported structure transfer!

The data structures equated to each other are not of the same type. The data structure transfers supported are: Point to Point, Event to Event, and Zone to Zone.

Line xx, Valid modes are 1=single axis, 2=ELS synchronized!

The second argument of the "ELS/MODE" command is missing or out of range. This can also be generated if the first argument is invalid and appears as two or more arguments to the compiler.

Line xx, variable table 'yyyy' index unknown!

The closing bracket is missing or other delimiters found in the index term of a variable or register with index format.

Mark table filled - yyyy, reduce number of subroutine calls.

The total number of marks used exceeds the table space provided. Marks are the location tags of the start of tasks, event functions and subroutines, or, the destination of a branch or Goto. Try to optimize your program to reduce the number of branches. If the problem persist, contact your Rexroth Indramat representative.

Upon successful completion of the compile, the number of marks and labels used are displayed in the completion window.

No main task (A, B, C, or D) found!

After compiling the program, no marks were found for Task_A, Task_B, Task_C, or Task_D. One or more of these tasks marks must be used. If it's a textual language program, check spelling and the underscore. The marks for the main tasks are case insensitive.

Sequencing error in output file!

While computing byte offsets for branches and subroutine calls, an unknown command op-code was encountered. This error can occur in a corrupted Windows memory system or a compiler bug. Try rebooting your computer and compiling again. If the problem persist, contact your Rexroth Indramat representative.

Size of program exceeds compiler space!

The compiler has 48k of space available for program development, this error is outputted when that space is filled. Variables and tables are not included in this space. Try reworking your program to fit it in the space.

Unable to allocate memory for compiler!

The 2nd pass compiler uses a large block of memory (48K) allocated from the Windows operating system to build the program. When Windows fails to allocate this memory, this error is outputted. Try closing other applications or rebooting Windows to free needed memory.

Unable to open source file.

This error is issued on failing to open the file "CLCCODE.TXT." Some possible causes are:

File "CLCCODE.TXT" is not in the "\\\Indramat\VisualMotion 8\" directory. This file is created by compiling a textual or icon program.

The maximum number of file is already open. DOS file "CONFIG.SYS" configures the maximum number of files.

The file is already open and cannot be shared.

Write to file error!

This error is outputted when the number of bytes sent to the output file doesn't match the number of bytes written in the output file. Check for available hard drive disk space or write protection on the output file (\\\Indramat\VisualMotion 8\project*.exc).

13 Index

*	New.....	6-13, 7-15, 8-12
C		
C-0-2635.....	6-17, 7-19, 8-17	
Cam synchronization	2-2	
Cascading ELS Groups	5-24	
channels		
Cyclic Data Channel configuration	6-3, 7-2	
multiplex.....	6-2, 7-2	
Non-Cyclic Channel	7-2	
Parameter Channel.....	6-2	
Real-Time Channel	6-2, 6-3	
single.....	6-2, 7-2	
CLC Compiler Base Code	12-2	
CLC register structure.....	5-1	
command execution.....	11-1	
Configuration	8-2	
Configuration word		
ELS Group.....	5-12	
connection box.....	5-26	
control word	6-5, 7-4, 8-4	
coordinated motion	3-41	
Cyclic Channel.....	7-2	
Cyclic Data Channel		
configuration.....	6-3, 7-2	
Cyclic Data Valid....	6-15, 7-17, 8-15	
D		
Data Channels		
Configuration.....	8-2	
Data Exchange Objects	6-27, 7-6, 7-30, 8-25	
5E70.....	6-27, 8-25	
5E71.....	6-27, 8-25	
5E72.....	6-27, 8-25	
5E73.....	6-27, 8-25	
Class 100, Instance 1-4, Attribute 100.....	7-30	
Class 100, Instance 1-4, Attribute 100	7-6	
D		
Data Menu		
I/O Mapper	3-29	
Variables.....	3-24	
Data Sizes		
Cyclic.....	6-4, 7-3, 8-3	
data types		
multiplex.....	7-2	
single.....	7-2	
Data Types		
Cyclic.....	6-4, 7-3, 8-3	
Multiplex	6-3, 6-4, 7-3, 8-3, 8-4	
Single.....	6-3, 6-4, 7-3, 8-3	
DDE client interfaces	9-32	
DDE monitor menu	9-20	
DDE communications.....	9-21	
DDE conversations	9-20	
network monitors	9-22	
DDE Server.....	9-8	
item name.....	9-9	
menus.....	9-10	
service name	9-9	
topic name.....	9-9, 9-30	

Wonderware	9-37
DDE settings menu.....	9-11
AT modem communications	9-13
network communications	9-16
serial communications.....	9-13
server configuration	9-11
DDEInitiat	9-34
Default Register Labels	
ELS Group 1-8.....	5-8
Virtual Master 1 & 2	5-4
DeviceNet Error Codes.....	7-34
Drive Parameter Editor	10-1
analog outputs	10-4
drive direction	10-7
drive language selection	10-7
drive limits	10-10
drive monitoring.....	10-8
drive name.....	10-8
drive reference.....	10-11
drive tuning	10-9
electronic cam shaft.....	10-34
encoder 2.....	10-13
load default parameters	10-2
mechanical	10-18
minimize parameter lists	10-2
parameter overview	10-20
phase synchronization	10-31
scan for SERCOS drives	10-4
transfer parameters	10-3
velocity synchronization	10-30
Dynamic Data Exchange	9-8
Dynamic Synchronization	5-30, 5-35
E	
Edit ELS Group Master.....	5-17
Edit ELS Masters.....	5-16
Edit Virtual Masters	5-18
ELS Group.....	4-2
ELS Group configuration word .	5-12
ELS Group Master.....	4-2
ELS Group Masters	4-1, 5-15
ELS Groups	5-27
ELS Master Assignment	5-23
ELS Master connection box.....	5-26
ELS Master types	5-23
ELS Masters	5-15, 5-25
ELS Runtime Utility.....	5-15
Edit ELS Group Masters	5-17
Edit ELS Masters	5-16
Edit Virtual Masters	5-18
ELS Slave.....	5-28
ELS System Masters	4-2
ELS_MSTR_A#	5-7
ELS_MSTR_EC#	5-7
ELS_MSTR_FLTR#.....	5-7
ELS_MSTR_FREQ#.....	5-7
ELS_MSTR_M#	5-7
ELS_MSTR_N#	5-7
ELS_MSTR_TYPE#.....	5-7
Error Reaction	
Ignore	6-18, 7-20, 8-18
Shutdown CLC.....	6-18, 7-20, 8-18
Warning Only.....	6-18, 7-20, 8-18
Errors	
208 Lost Fieldbus Connection	6-18, 7-20, 8-18
209 Field Bus Mapper Timeout	6-18, 7-20, 8-18
F	
FB Card Found	6-16, 7-18, 8-16
FB Init OK	6-15, 7-17, 8-15
FB Slave Ready.....	6-15, 7-17, 8-15
Fieldbus Diagnostics	6-16, 7-18, 8-16
Fieldbus Error Reaction	6-17, 7-19, 8-17
Fieldbus Mapper	6-2, 7-2, 8-2
Fieldbus Master	6-1, 7-1, 8-1
fieldbus message header	7-28
Fieldbus Resource Monitor	6-16, 7-18, 8-16
Fieldbus Slave	6-1, 7-1, 8-1
Fieldbus Status.....	6-15, 7-17, 8-15
File Extensions	
.acc	3-25
.exb	3-25
.exc	3-25
.iom	3-25
.lss	3-25
.lst	3-25
.map	3-25
.pos	3-25
.prm	3-25
.str	3-25
.tbl	3-25
.txt	3-25
.vel	3-25
.csv	3-25
File Types	3-25
First In First Out	11-2

First Pass Errors.....	12-17	single axis.....	2-1
Float variables	5-3	velocity mode	2-1
floating point variable.....	3-24	Motion Capabilities	2-1
G		multiplex data types	7-2
Global variables.....	3-24	Multiplex Data Types6-3, 6-4, 7-3, 8-3, 8-4	
I		Multiplexing	
I/O Mapper.....	3-29	Control and Status Words.....	7-4
Display Strings.....	3-29	multi-turn encoder.....	10-12
open default *.iom file	3-30		
Uploading I/O Mapper	3-29		
Icon Language Warnings and Error Messages	12-15		
Icons			
Start.....	3-11		
Ignore (Fieldbus Error Reaction)6- 18, 7-20, 8-18			
Immediate.....	5-30		
Immediate switching of ELS Group	5-33		
Initialization of VisualMotion's Multi- Master Functionality	5-1		
input.....	6-2, 7-2, 8-2		
Integer variables.....	5-3		
Integers.....	3-24		
Integral Action Time.....	10-9		
J			
Jog Controls	5-29		
Jogging an ELS Group in Local Mode.....	5-29		
L			
Labels.....	3-21		
least significant bit . 6-15, 7-17, 8-15			
Local Mode	5-30		
Local variables.....	3-24		
Lock Off cam profile.....	5-37		
Lock On / Lock Off.....	5-36		
Lock On cam profile.....	5-37		
M			
Mapped Data	7-6		
Mapping List			
Add Button.....	6-13, 7-15, 8-12	Parameter Channel.....	6-2, 6-5
Delete Button	6-13, 7-15, 8-12	Short Format 3	6-6
Edit Button.....	6-13, 7-15, 8-12	VisualMotion ASCII Format	6-6
Insert Button	6-13, 7-15, 8-12	Parameter Channel Error Codes6- 32	
New Button	6-13, 7-15, 8-12	parameter overview	
Maximum Model Deviation	10-8	custom list.....	10-26, 10-28
measuring wheel example.....	10-15	custom list group	10-28
most significant bit . 6-15, 7-17, 8-15		edit parameter	10-22
motion		find a parameter	10-23
coordinated.....	2-1	parameter access	10-22
circular interpolation.....	2-2	parameter list	10-24
constant speed.....	2-2	system configuration	10-21
kinematics.....	2-2	path planner.....	11-1
linear interpolation.....	2-2	Phase Correction	5-31
ELS	2-2	Phase synchronization.....	2-2
phase synchronous mode	2-2	PLC Programming	6-19
velocity synchronous mode....	2-2	Non-Cyclic Data (via Data Exchange Objects)	6-27, 7-30, 8-25
non-coordinated	2-1	Position Mode	5-23
		predefined register.....	5-1

Program Debugging and Monitoring	12-1
program execution	3-26
Programming Concepts	11-1
Data Transfer	11-6
event arming/disarming	11-7
Event Function	11-1
event processing	11-7
event queue	11-2
Event Type	11-1
event types	11-8
Events	11-1
Extended User Input (EUI)	11-26
First In First Out (FIFO)	11-2
parameter transfer tool	11-5
stack	11-2
transfer events	11-5
Proportional Gain	10-9
R	
Real Masters	4-1
Real-Time Channel	6-2, 6-3
Register 19 Definition (Fieldbus Status)	6-15, 7-17, 8-15
Register 20 Definition (Fieldbus Diagnostics)	6-16, 7-18, 8-16
Register 26 Definition (Fieldbus Resource Monitor)	6-16, 7-18, 8-16
S	
Second Pass Compiler Errors	12-17
serial communications	9-13
Serial I/O	3-1
server configuration	9-11
Short Format 3	6-6
Shutdown CLC (Fieldbus Error Reaction)	6-18, 7-20, 8-18
single data types	7-2
Single Data Types	6-3, 6-4, 7-3, 8-3
Smoothing Time Constant	10-9
status word	6-5, 7-4, 8-4
T	
Stop Ramp	5-29
Subroutine	3-11
Synchronization	5-30
Synchronization Setup	5-31
V	
Variables	3-21, 3-24
ELS Group	5-10
ELS Master	5-6
Virtual Master	5-6
Velocity Mode	5-22
Velocity synchronization	2-2
View Menu	
Event functions	3-24
Subroutine	3-11
Task	3-11
Virtual Master	5-19
Virtual Master and ELS Group	
Default Registers	5-2
Virtual Master Compile Time Initialization	5-19
Virtual Masters	4-1, 5-15
VisualMotion ASCII Format	6-6
VisualMotion GPS/GPP Overview	4-1
VisualMotion System Overview	1-1
W	
Warning Only (Fieldbus Error Reaction)	6-18, 7-20, 8-18
Wonderware	
tagnames	9-38

14 Service & Support

14.1 Helpdesk

Unser Kundendienst-Helpdesk im Hauptwerk Lohr am Main steht Ihnen mit Rat und Tat zur Seite. Sie erreichen uns

- telefonisch: **+49 (0) 9352 40 50 60**
über Service Call Entry Center Mo-Fr 07:00-18:00
- per Fax: **+49 (0) 9352 40 49 41**
- per e-Mail: **service@indramat.de**

Our service helpdesk at our headquarters in Lohr am Main, Germany can assist you in all kinds of inquiries. Contact us

- by phone: **+49 (0) 9352 40 50 60**
via Service Call Entry Center Mo-Fr 7:00 am - 6:00 pm
- by fax: **+49 (0) 9352 40 49 41**
- by e-mail: **service@indramat.de**

14.2 Service-Hotline

Außerhalb der Helpdesk-Zeiten ist der Service direkt ansprechbar unter

oder **+49 (0) 171 333 88 26**
+49 (0) 172 660 04 06

After helpdesk hours, contact our service department directly at

+49 (0) 171 333 88 26
or **+49 (0) 172 660 04 06**

14.3 Internet

Ergänzende Hinweise zu Service, Reparatur und Training sowie die **aktuellen** Adressen unserer Service- und Vertriebsbüros finden Sie unter **www.indramat.de** – einige Angaben in dieser Dokumentation können inzwischen überholt sein.

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem lokalen Ansprechpartner auf.

- | | |
|---|----------------------------------|
|  | Verkaufsniederlassungen |
|  | Niederlassungen mit Kundendienst |

Additional notes about service, repairs and training as well as the **actual** addresses of our sales- and service facilities are available on the Internet at **www.indramat.de** – some information in this documentation may meanwhile be obsolete.

Please contact the sales & service offices in your area first.

- | | |
|---|---------------------------|
|  | sales agencies |
|  | offices providing service |

14.4 Vor der Kontaktaufnahme... - Before contacting us...

Wir können Ihnen schnell und effizient helfen wenn Sie folgende Informationen bereithalten:

detaillierte Beschreibung der Störung und der Umstände.

Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern.

Tel.-/Faxnummern und e-Mail-Adresse, unter denen Sie für Rückfragen zu erreichen sind.

For quick and efficient help, please have the following information ready:

1. Detailed description of the failure and circumstances.
2. Information on the nameplate of the affected products, especially typecodes and serial numbers.
3. Your phone/fax numbers and e-mail address, so we can contact you in case of questions.

14.5 Kundenbetreuungsstellen - Sales & Service Facilities

Deutschland – Germany

vom Ausland: (0) nach Landeskennziffer weglassen!
from abroad: don't dial (0) after country code!

Vertriebsgebiet Mitte Germany Centre	SERVICE CALL ENTRY CENTER MO – FR von 07:00 - 18:00 Uhr from 7 am - 6 pm Tel. +49 (0) 9352 40 50 60 service@indramat.de	SERVICE HOTLINE MO – FR von 17:00 - 07:00 Uhr from 5 pm - 7 am + SA / SO Tel.: +49 (0)172 660 04 06 oder / or Tel.: +49 (0)171 333 88 26	SERVICE ERSATZTEILE / SPARES verlängerte Ansprechzeit - extended office time - ♦ nur an Werktagen - only on working days - ♦ von 07:00 - 18:00 Uhr - from 7 am - 6 pm - Tel. +49 (0) 9352 40 42 22
Vertriebsgebiet Süd Germany South	Gebiet Südwest Germany South-West Bosch 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	Vertriebsgebiet Ost Germany East Bosch Rexroth AG Beckerstraße 31 09120 Chemnitz Tel.: +49 (0)371 35 55-0 Fax: +49 (0)371 35 55-333	Vertriebsgebiet Ost Germany East Bosch Rexroth AG Regionalzentrum Ost Walter-Köhn-Str. 4d 04356 Leipzig Tel.: +49 (0)341 25 61-0 Fax: +49 (0)341 25 61-111
Vertriebsgebiet West Germany West	Vertriebsgebiet Mitte Germany Centre Bosch Rexroth AG Regionalzentrum Mitte Waldecker Straße 13 64546 Mörfelden-Walldorf Tel.: +49 (0) 61 05 702-3 Fax: +49 (0) 61 05 702-444	Vertriebsgebiet Nord Germany North Bosch Rexroth AG Walsroder Str. 93 30853 Langenhagen Tel.: +49 (0) 511 72 66 57-0 Fax: +49 (0) 511 72 66 57-95	Vertriebsgebiet Nord Germany North Bosch Rexroth AG Kieler Straße 212 22525 Hamburg Tel.: +49 (0) 40 81 955 966 Fax: +49 (0) 40 85 418 978

Europa (West) - Europe (West)

vom Ausland: (0) nach Landeskennziffer weglassen,
from abroad: don't dial (0) after country code,
Italien: 0 nach Landeskennziffer mitwählen
Italy: dial 0 after country code

Austria - Österreich	Austria – Österreich	Belgium - Belgien	Denmark - Dänemark
Bosch Rexroth GmbH Bereich Indramat Stachegasse 13 1120 Wien Tel.: +43 (0)1 985 25 40 Fax: +43 (0)1 985 25 40-93	Bosch 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	Bosch Rexroth AG Electric Drives & Controls Industrielaan 8 1740 Ternat Tel.: +32 (0)2 5830719 Service: +32 (0)2 5830717 Fax: +32 (0)2 5830731 indramat@boschrexroth.be	Bosch Rexroth A/S Zinkvej 6 8900 Randers Tel.: +45 (0)87 11 90 60 Fax: +45 (0)87 11 90 61
Great Britain – Großbritannien	Finland - Finnland	France - Frankreich	France - Frankreich
Bosch Rexroth Ltd. Rexroth Indramat Division Broadway Lane, South Cerney Cirencester, Glos GL7 5UH Tel.: +44 (0)1285 863000 Fax: +44 (0)1285 863030 sales@indramat.co.uk service@indramat.co.uk	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	Bosch Rexroth S.A. Division Rexroth Indramat Avenue de la Trentaine BP. 74 77503 CHELLES CEDEX Tel.: +33 (0)164 72-70 00 Fax: +33 (0)164 72-63 00 Hotline: +33 (0)608 33 43 28	Bosch Rexroth S.A. Division Rexroth Indramat 1270, Avenue de Lardenne 31100 Toulouse Tel.: +33 (0)5 61 49 95 19 Fax: +33 (0)5 61 31 00 41
France - Frankreich	Italy - Italien	Italy - Italien	Italy - Italien
Bosch 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	Bosch Rexroth S.p.A. Via G. Di Vittorio, 1 20063 Cernusco S/N.MI Tel.: +39 02 2 365 270 Fax: +39 02 700 408 252378	Bosch Rexroth S.p.A. Via Paolo Veronesi, 250 10148 Torino Tel.: +39 011 224 88 11 Fax: +39 011 220 48 04	Bosch Rexroth S.p.A. Via del Progresso, 16 (Zona Ind.) 35020 Padova Tel.: +39 049 8 70 13 70 Fax: +39 049 8 70 13 77
Italy - Italien	Italy - Italien	Netherlands - Niederlande/Holland	Netherlands - Niederlande/Holland
Bosch Rexroth S.p.A. Via Mascia, 1 80053 Castellamare di Stabia NA Tel.: +39 081 8 71 57 00 Fax: +39 081 8 71 68 85	Bosch Rexroth S.p.A. Viale Oriani, 38/A 40137 Bologna Tel.: +39 051 34 14 14 Fax: +39 051 34 14 22	Bosch Rexroth B.V. Kruisbroeksestraat 1 (P.O. Box 32) 5281 RV Boxtel Tel.: +31 (0)411 65 19 51 Fax: +31 (0)411 65 14 83 indramat@hydraudyne.nl	Bosch Rexroth Services B.V. Kruisbroeksestraat 1 (P.O. Box 32) 5281 RV Boxtel Tel.: +31 (0)411 65 19 51 Fax: +31 (0)411 67 78 14
Norway - Norwegen	Spain - Spanien	Spain - Spanien	Sweden - Schweden
Bosch Rexroth AS Rexroth Indramat Division Berghagen 1 or: Box 3007 1405 Ski-Langhus 1402 Ski Tel.: +47 (0)64 86 41 00 Fax: +47 (0)64 86 90 62 jul.ruud@rexroth.no	Bosch 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 Grandmontagne no.2 20018 San Sebastián 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	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
Sweden - Schweden	Switzerland West - Schweiz West	Switzerland East - Schweiz Ost	
Rexroth Mecman Svenska AB Indramat Support Ekvändan 7 254 67 Helsingborg Tel.: +46 (0) 42 38 88 -50 Fax: +46 (0) 42 38 88 -74	Bosch 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	Bosch Rexroth Schweiz AG Geschäftsbereich Indramat Hemrietstrasse 2 8863 Buttikon Tel. +41 (0) 55 46 46 205 Fax +41 (0) 55 46 46 222	

Europa (Ost) - Europe (East)

vom Ausland: (0) nach Landeskennziffer weglassen
from abroad: don't dial (0) after country code

Czech Republic - Tschechien	Czech Republic - Tschechien	Hungary - Ungarn	Poland – Polen
Bosch -Rexroth, spol.s.r.o. Hviezdoslavova 5 627 00 Brno Tel.: +420 (0)5 48 126 358 Fax: +420 (0)5 48 126 112	DEL a.s. Strojírenská 38 Zdar nad Sázavou 591 01 Czech republic Tel.: +420 616 64 3144 Fax: +420 616 216 57	Bosch Rexroth Kft. Angol utca 34 1149 Budapest Tel.: +36 (1) 364 00 02 Fax: +36 (1) 383 19 80	Bosch Rexroth Sp.zo.o. Biuro Poznan ul. Dabrowskiego 81/85 60-529 Poznan Tel.: +48 061 847 64 62 /-63 Fax: +48 061 847 64 02
Rumania - Rumänien	Russia - Russland	Russia - Russland	Turkey - Türkei
Bosch Rexroth Sp.zo.o. Str. Drobet nr. 4-10, app. 14 70258 Bucuresti, Sector 2 Tel.: +40 (0)1 210 48 25 +40 (0)1 210 29 50 Fax: +40 (0)1 210 29 52	Bosch Rexroth Wolokolamskoje Chaussee 73 Zimmer 406, 408 RUS – 123424 Moskau Tel.: +7 095/ 232 08 34 +7 095/ 232 08 35 Fax: +7 095/ 232 08 36 info.rex@rexroth.ru	ELMIS 10, Internationalnaya Str. 246640 Gomel, Belarus Tel.: +375/ 232 53 42 70 Fax: +375/ 232 53 37 69 elmis_ltd@yahoo.com	Bosch Rexroth Otomasyon San & Tic. A.S. Fevzi Cakmak Cad No. 3 34630 Sefaköy İstanbul Tel.: +90 212 541 60 70 Fax: +90 212 599 34 07
Slowenia - Slowenien			
DOMEI Otoki 21 64 228 Zelezniki Tel.: +386 5 5117 152 Fax: +386 5 5117 225 brane.ozbek@domei.si			

Africa, Asia, Australia – incl. Pacific Rim

vom Ausland: (0) nach Landeskennziffer weglassen!
from abroad: don't dial (0) after country code!

Australia - Australien	Australia - Australien	China	China
AIMS - Australian Industrial Machinery Services Pty. Ltd. Unit 3/45 Horne ST Campbellfield , VIC 3061 Melbourne Tel.: +61 (0) 393 590 228 Fax: +61 (0) 393 590 286 Hotline: +61 (0) 419 369 195 terryobrien@aimservices.com.au	Bosch Rexroth Pty. Ltd. No. 7, Endeavour Way Braeside Victoria, 31 95 Melbourne Tel.: +61 (0)3 95 80 39 33 Fax: +61 (0)3 95 80 17 33 mel@rexroth.com.au	Bosch Rexroth Ltd. Wai Gaoqiao Free Trade Zone No.122, Fu Te Dong Yi Road Shanghai 200131 - P.R.China Tel.: +86 21 58 66 30 30 Fax: +86 21 58 66 55 23 roger.shi_sh@boschrexroth.com.cn	Bosch Rexroth (China) Ltd. 15/F China World Trade Center 1, Jianguomenwai Avenue Beijing 100004, P.R.China Tel.: +86 10 65 05 03 80 Fax: +86 10 65 05 03 79
China	China	Hongkong	India - Indien
Bosch Rexroth (China) Ltd. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023, P.R.China Tel.: +86 411 46 78 930 Fax: +86 411 46 78 932	Bosch Rexroth (Changzhou) Co.Ltd. Guangzhou Repres. Office Room 1014-1016, Metro Plaza, Tian He District, 183 Tian He Bei Rd Guangzhou 510075, P.R.China Tel.: +86 20 8755-0030 +86 20 8755-0011 Fax: +86 20 8755-2387	Bosch Rexroth (China) Ltd. 6 th Floor, Yeung Yiu Chung No.6 Ind Bldg. 19 Cheung Shun Street Cheung Sha Wan, Kowloon, Hongkong Tel.: +852 22 62 51 00 Fax: +852 27 41 33 44 alexis.siu@boschrexroth.com.hk	Bosch 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
India - Indien	Indonesia - Indonesien	Japan	Japan
Bosch 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 Building # 202, Cilandak Commercial Estate Jl. Cilandak KKO, Jakarta 12560 Tel.: +62 21 7891169 (5 lines) Fax: +62 21 7891170 - 71	Bosch Rexroth Automation Corp. Service Center Japan Yutakagaoka 1810, Meito-ku, NAGOYA 465-0035, Japan Tel.: +81 (0)52 777 88 41 +81 (0)52 777 88 53 +81 (0)52 777 88 79 Fax: +81 (0)52 777 89 01	Bosch Rexroth Automation Corp. Rexroth Indramat Division 1F, I.R. Building Nakamachidai 4-26-44, Tsuzuki-ku YOKOHAMA 224-0041, Japan Tel.: +81 (0)45 942 72 10 Fax: +81 (0)45 942 03 41
Korea	Malaysia	Singapore - Singapur	South Africa - Südafrika
Bosch Rexroth-Korea Ltd. 1515-14 Dadae-Dong, Saha-Ku Rexroth Indramat Division Pusan Metropolitan City, 604-050 Republic of South Korea Tel.: +82 (0)51 26 00 741 Fax: +82 (0)51 26 00 747 gyhan@rexrothkorea.co.kr	Bosch Rexroth Sdn.Bhd. Head Office No. 3, Block B, Jalan SS 13/5 Subang Jaya Industrial Estate 47500 Petaling Jaya - Selangor Tel.: +60 (0) 3 73 44 870 Fax: +60 (0) 3 73 44 864 hockhwa@hotmail.com	Robert Bosch (SEA) Pte Ltd. Dept. RBSI-R/SAT 38-C Jalan Pemimpin Singapore 577180 Tel.: +65 35 05 470 Fax: +65 35 05 313 kenton.peh@sg.bosch.com	TECTRA Automation (Pty) Ltd. 28 Banfield Road, Industria North RSA - Maraisburg 1700 Tel.: +27 (0)11 673 20 80 Fax: +27 (0)11 673 72 69 Hotline: +27 (0)82 903 29 23 georgv@tectra.co.za
Taiwan	Thailand		
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 indramat@mail.net.tw	NC Advance Technology Co. Ltd. 59/76 Moo 9 Ramintra road 34 Tharang, Bangkhen, Bangkok 10230 Tel.: +66 2 943 70 62 +66 2 943 71 21 Fax: +66 2 509 23 62 sonkawin@hotmail.com		

Nordamerika – North America

USA Hauptniederlassung - Headquarters	USA Central Region - Mitte	USA Southeast Region - Südwest	USA SERVICE-HOTLINE
<p>Bosch Rexroth Corporation Rexroth Indramat Division 5150 Prairie Stone Parkway Hoffman Estates, IL 60192-3707</p> <p>Tel.: +1 847 6 45 36 00 Fax: +1 847 6 45 62 01 service@indramat.com</p>	<p>Bosch Rexroth Corporation Rexroth Indramat Division 1701 Harmon Road Central Region Technical Center Auburn Hills, MI 48326</p> <p>Tel.: +1 248 3 93 33 30 Fax: +1 248 3 93 29 06</p>	<p>Bosch Rexroth Corporation Rexroth Indramat Division Southeastern Technical Center 3625 Swiftwater Park Drive Suwanee, Georgia 30124</p> <p>Tel.: +1 770 9 32 32 00 Fax: +1 770 9 32 19 03</p>	<p>- 7 days x 24hrs -</p> <p>+1-800-860-1055</p>
<p>USA East Region –Ost</p> <p>Bosch Rexroth Corporation Rexroth Indramat Division Charlotte Regional Sales Office 14001 South Lakes Drive Charlotte, North Carolina 28273</p> <p>Tel.: +1 704 5 83 97 62 +1 704 5 83 14 86</p>	<p>USA Northeast Region – Nordost</p> <p>Bosch Rexroth Corporation Rexroth Indramat Division Northeastern Technical Center 99 Rainbow Road East Granby, Connecticut 06026</p> <p>Tel.: +1 860 8 44 83 77 Fax: +1 860 8 44 85 95</p>	<p>USA West Region – West</p> <p>Bosch Rexroth Corporation 7901 Stoneridge Drive, Suite 220 Pleasant Hill, California 94588</p> <p>Tel.: +1 925 227 10 84 Fax: +1 925 227 10 81</p>	
<p>Canada East - Kanada Ost</p> <p>Bosch Rexroth Canada Corporation Burlington Division 3426 Mainway Drive Burlington, Ontario Canada L7M 1A8</p> <p>Tel.: +1 905 335 55 11 Fax: +1 905 335-41 84 michael.moro@boschrexroth.ca</p>	<p>Canada West - Kanada West</p> <p>Bosch Rexroth Canada Corporation 5345 Goring St. Burnaby, British Columbia Canada V7J 1R1</p> <p>Tel. +1 604 205-5777 Fax +1 604 205-6944 david.gunby@boschrexroth.ca</p>	<p>Mexico</p> <p>Bosch Rexroth S.A. de C.V. Calle Neptuno 72 Unidad Ind. Vallejo MEX - 07700 Mexico, D.F.</p> <p>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</p>	

Südamerika – South America

Argentina - Argentinien	Argentina - Argentinien	Brazil - Brasilien	Brazil - Brasilien
<p>Bosch Rexroth S.A.I.C. "The Drive & Control Company" Acassuso 48 41/47 1605 Munro Prov. Buenos Aires</p> <p>Tel.: +54 (0)11 4756 01 40 Fax: +54 (0)11 4756 01 36 mannesmann@mannesmannsaic.com.ar</p>	<p>NAKASE Servicio Tecnico CNC Calle 49, No. 5764/66 1653 Villa Balester Prov. - Buenos Aires</p> <p>Tel.: +54 (0) 11 4768 36 43 Fax: +54 (0) 11 4768 24 13 nakase@usa.net nakase@nakase.com</p>	<p>Bosch Rexroth Ltda. Av. Tégula, 888 Ponte Alta, Atibaia SP CEP 12942-440</p> <p>Tel.: +55 (0)11 4414 56 92 +55 (0)11 4414 56 84 Fax sales: +55 (0)11 4414 57 07 Fax serv.: +55 (0)11 4414 56 86 alexandre.wittwer@rexroth.com.br</p>	<p>Bosch Rexroth Ltda. R. Dr.Humberto Pinheiro Vieira, 100 Distrito Industrial [Caixa Postal 1273] BR - 89220-390 Joinville - SC</p> <p>Tel./Fax: +55 (0)47 473 58 33 Mobil: +55 (0)47 9974 6645 prochnow@zaz.com.br</p>
<p>Columbia - Kolumbien</p> <p>Reflutec de Colombia Ltda. Calle 37 No. 22-31 Santa Fe de Bogotá, D.C. Colombia</p> <p>Tel.: +57 1 368 82 67 +57 1 368 02 59 Fax: +57 1 268 97 37 reflutech@inter.net.co</p>			



2 8 9 8 7 4

Printed in the U.S.A.

