



# MotionWorks+™ Windows Software and Icon-Based Programming Manual



**WARNING**

YASKAWA manufactures component parts that can be used in a wide variety of industrial applications. The selection and application of YASKAWA products remain the responsibility of the equipment designer or end user. YASKAWA accepts no responsibility for the way its products may be incorporated into the final system design.

Under no circumstances should any YASKAWA product be incorporated into any product or design as the exclusive or sole safety control. Without exception, all controls should be designed to detect faults dynamically under all circumstances. All products designed to incorporate a component part manufactured by YASKAWA must be supplied to the end user with appropriate warnings and instructions as to that part's safe use and operation. Any warnings provided by YASKAWA must be promptly provided to the end user.

YASKAWA offers an express warranty only as to the quality of its products in conforming to standards and specifications published in YASKAWA's manual. **NO OTHER WARRANTY, EXPRESS OR IMPLIED, IS OFFERED.** YASKAWA assumes no liability for any personal injury, property damage, losses, or claims arising from misapplication of its products.

## Notes

# Contents

<b>1. MotionWorks+™</b>	<b>3</b>
1.1 System Requirements	3
1.2 Installation	3
1.3 Introduction To Software Features	4
1.4 Creating A Project	6
1.5 Saving A Project	8
1.6 The Project Explorer	9
1.6.1 HMI Data	38
1.6.2 Import/Export Initiation	42
1.7 Scope	51
1.8 The Block Toolbar	58
1.9 The Properties Window	59
1.10 The Program Window	60
1.11 Expression Builder	62
1.12 Cross Reference	65
1.13 Search and Replace	66
1.14 Connecting To The Controller	67
1.15 Compiling A Program	71
1.16 Downloading a Project	72
1.17 Saving a Project to Flash	73
1.18 Electronic Cam Tool	74
1.19 Archive	101
<b>2. Icon-Based Motion Control Programming</b>	<b>103</b>
2.1 Programming Tools	103
<b>3. Programming Concept</b>	<b>105</b>

<b>3.1</b>	<b>User Unit Conversion .....</b>	<b>127</b>
<b>3.2</b>	<b>Block Reference.....</b>	<b>130</b>
3.2.1	CALL SUBROUTINE.....	132
3.2.2	CAM .....	133
3.2.3	CAM SHIFT .....	135
3.2.4	CHANGE DYNAMICS .....	137
3.2.5	DEFINE POSITION .....	138
3.2.6	END .....	139
3.2.7	GEAR.....	140
3.2.8	GEAR RATIO .....	141
3.2.9	HOME AXIS .....	142
3.2.10	IF EVENT.....	144
3.2.11	IF FAULT .....	145
3.2.12	INPUT.....	146
3.2.13	JOG AXIS.....	147
3.2.14	LATCH .....	148
3.2.15	LATCH TARGET .....	150
3.2.16	LAUNCH PROGRAM.....	153
3.2.17	MOVE AXIS .....	154
3.2.18	PLS.....	156
3.2.19	RESET FAULT .....	158
3.2.20	SCALE CAM.....	159
3.2.21	SERVO .....	160
3.2.22	SET VARIABLE .....	161
3.2.23	SLAVE OFFSET .....	162
3.2.24	START.....	163
3.2.25	STOP MOTION.....	164
3.2.26	SUSPEND PROGRAM.....	165
3.2.27	TIMER.....	166
3.2.28	TORQUE .....	167
<b>Appendix A</b>	<b>Operational Examples .....</b>	<b>169</b>
<b>Appendix B</b>	<b>Standardized Template Project.....</b>	<b>177</b>
B.1	Summary .....	178
B.2	Programs .....	179
B.3	Subroutines .....	186
<b>Appendix C</b>	<b>MW+ Camming 101.....</b>	<b>201</b>
	<b>Index.....</b>	<b>225</b>

## **Introduction**

This manual is divided into two sections. The first section covers the user interface and other features of the Windows application. The second section covers machine functions in general, and contains details about the icon-based programming environment and its interface to the controller. Program examples are included in the appendix.

This manual was updated to correspond with the software version 2.83.

## Notes

# 1. MotionWorks+™

## 1.1 System Requirements

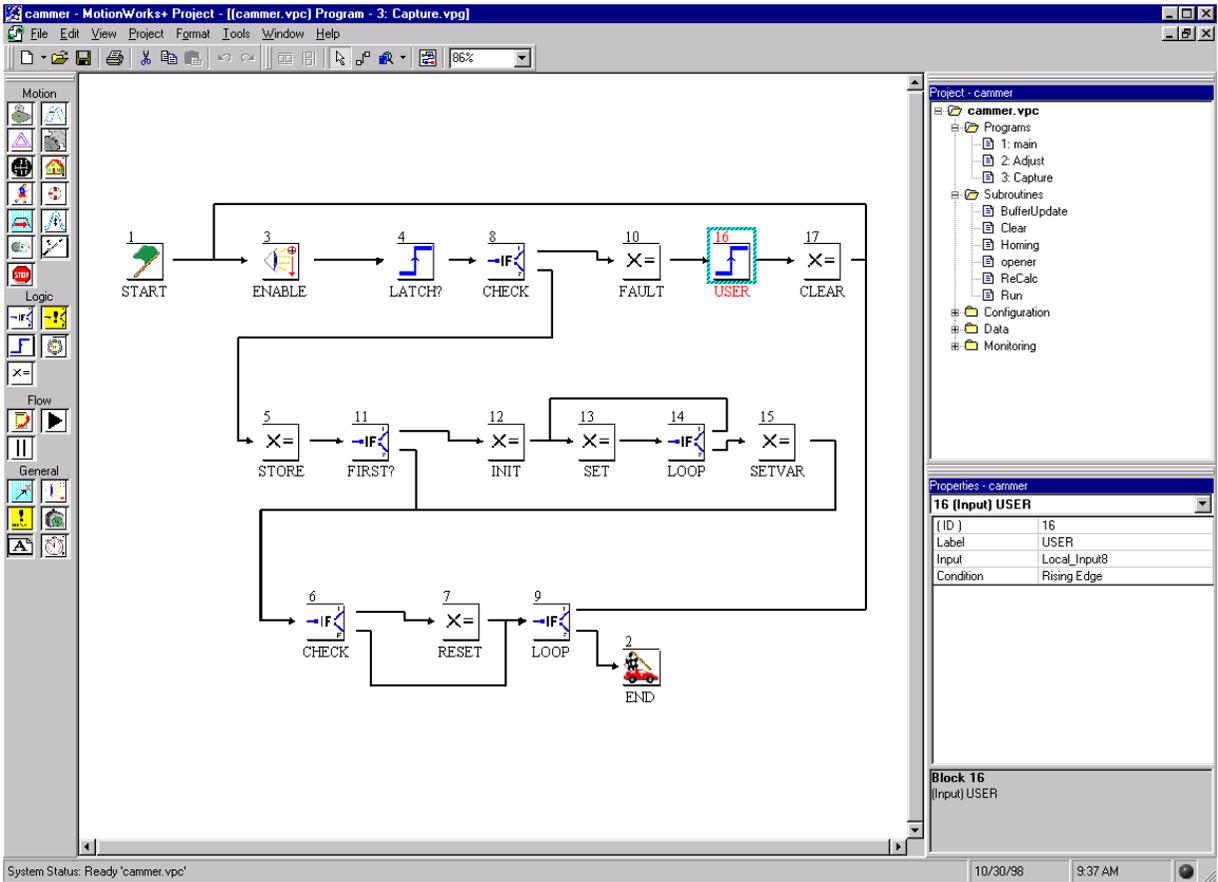
- \* Microsoft(R) 95/98/NT/2000 operating systems.
- \* Minimum - 133MHz, 64MB memory, 100MB hard disk space.
- \* Recommended 350MHz, 128MB memory, 500MB hard disk space.
- \* High Color (16-bit) Color Palette display setting.
- \* Microsoft (R-compatible mouse).
- \* 100MB of free disk space to install the application and all supporting libraries.

## 1.2 Installation

The installation of MotionWorks+™ consists of three separate programs, which are automatically installed by the Install Wizard. They are: MotionWorks+™, Electronic CamTool, and CimScope.

First, remove any previous versions. When removing MotionWorks+™, these three programs must be removed using the Add/Remove Programs feature of the Windows control panel. After installation, Windows must be restarted for the installation to be complete.

## 1.3 Introduction To Software Features



MotionWorks+™ incorporates the following features and tools to aid in the creation of motion control programs.

### 1. Project Window

Displays and allows editing of application information in a file manager-type structure.

### 2. Properties Window

Used to edit and view block information. These are called properties.

### 3. Program Window

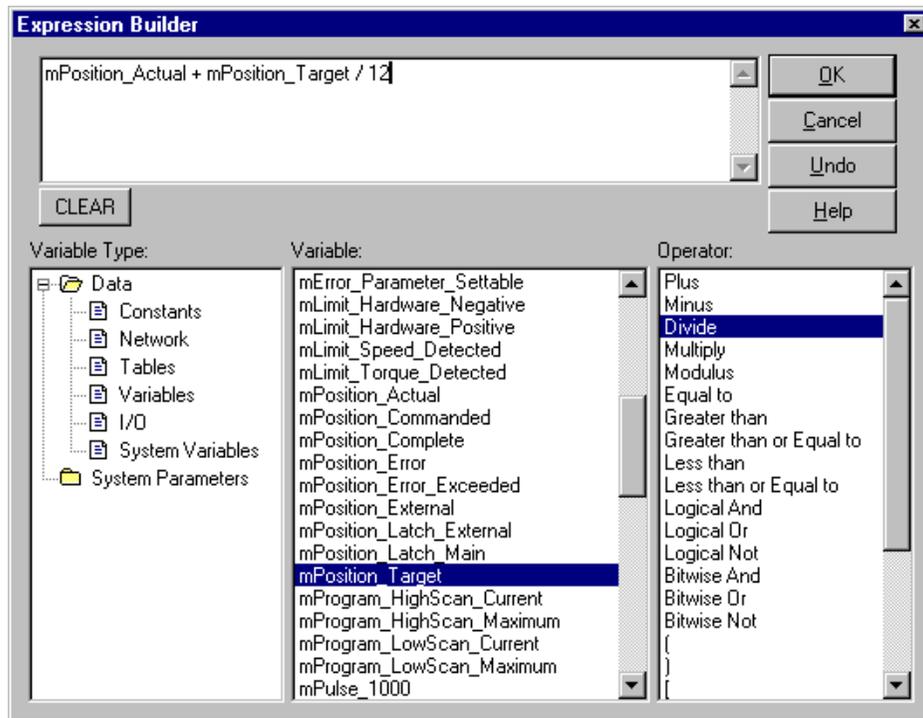
Shows the graphical program layout; i.e., motion control programs are created by connecting blocks together.

### 4. Block Toolbar

Used to drag and drop blocks on the Program Window.

## 5. Expression Builder (below)

Accessible by double-clicking on block properties. It aids in creating calculations for a variety of purposes.



## 6. The *Debugging Tools* consist of the following:

- Monitoring / Data - View and change data values online.
- Monitoring / I/O - Displays a graphical view of I/O data and other system variables for verification of hardware connections and debugging.
- Monitoring / Program - Displays current block numbers for all active programs and shows the program and subroutine names.
- Monitoring/Scope - Records specific information in the controller, then uploads it to MW+ for display as a graph.

## 1.4 Creating A Project

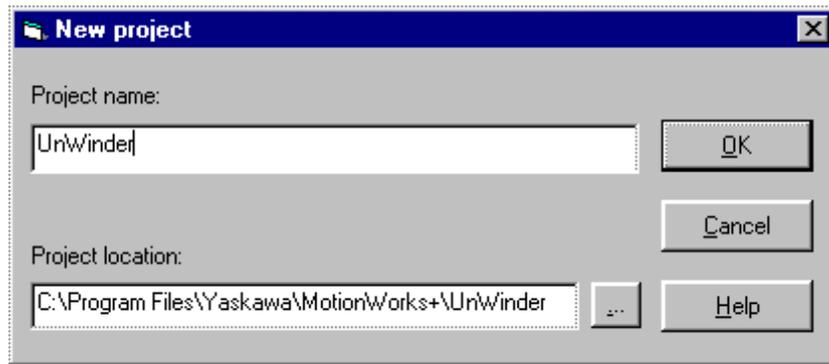
A project is a collection of files, all of which pertain to a specific job. Each project occupies its own directory for easy portability.

### Accessibility

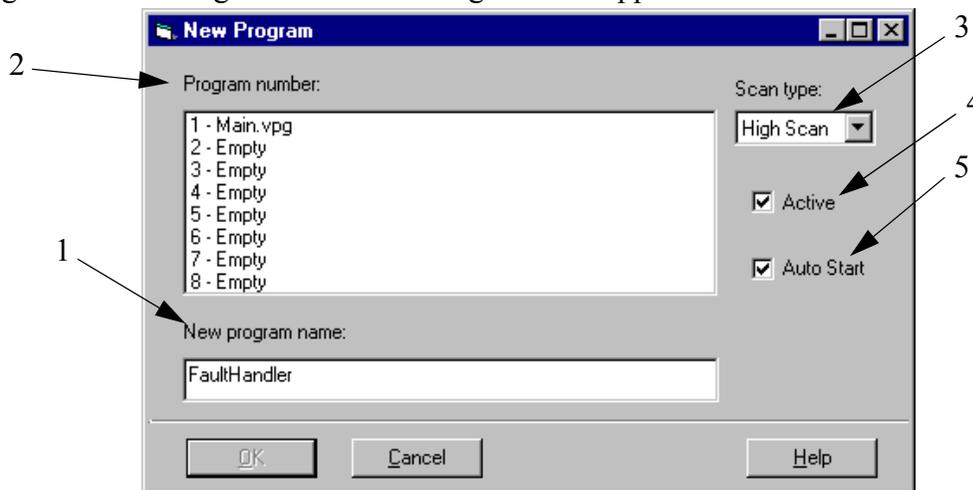
To access the New Project dialog box select:

- From the Main Menu > File > New Project
- From the Standard Toolbar > New > New Project (from the drop down list)
- Hot Key: Ctrl+N

1. Input the project name and file system location. The length of the project name may be up to 255 characters, including spaces. Characters not allowed in the project name are /, \, :, \*, ?, “, <, >, and |.



2. Display the New Program dialog box to add a program to the project. This can be done either from the main menu, or from the Project Explorer by selecting Project > Add - Program > New Program. The following window appears:



**1. New Program Name**

The [character restrictions](#) applying to the project name apply to the program name. Characters not allowed in the project name are /, \, :, \*, ?, “, <, >, and |.

**2. Program Number**

In addition to creating a name, select a program number. Add programs to empty locations or replace existing programs by clicking any of the program numbers in the list.

**3. Scan Type**

There are two choices for scan speed: high and low. Priority levels must be considered when selecting scan speed. For example, high speed is recommended for critical I/O that must be monitored frequently. Low speed is sufficient for low priority events such as operator push buttons. Each program executes one block in the scan period.

Note: High and low scans must be an even multiple of each other. For example: 2/10, 4/16, etc.

**4. Active**

Select the **Active** checkbox to indicate whether the program is to be incorporated during project compilation. During project debugging, it may be helpful to deactivate certain programs to isolate problems. Deactivated programs will not be compiled or downloaded.

**5. Autostart**

The **Autostart** checkbox indicates whether the program automatically executes its START block when the power is initially turned on. If **Autostart** is not selected, programs can be launched later by incorporating a LAUNCH PROGRAM block in another program. At least one program in the project must be selected for autostart in order to ensure execution.

## 1.5 Saving A Project

All files contained in the project are saved when the SAVE icon is pressed. If the project is not saved before exiting MotionWorks+™, a Save Project window appears. All the files that have been changed are displayed. Select the files to be saved at this time. (The default is to save all changed files.) The user must de-select the names of any files that are not to be saved. The **Cancel** button aborts closing MotionWorks+™, and does not save any files.

### Accessibility

To save a project, select

- From the Main Menu > File > Save Project
- From the Standard Toolbar > Save

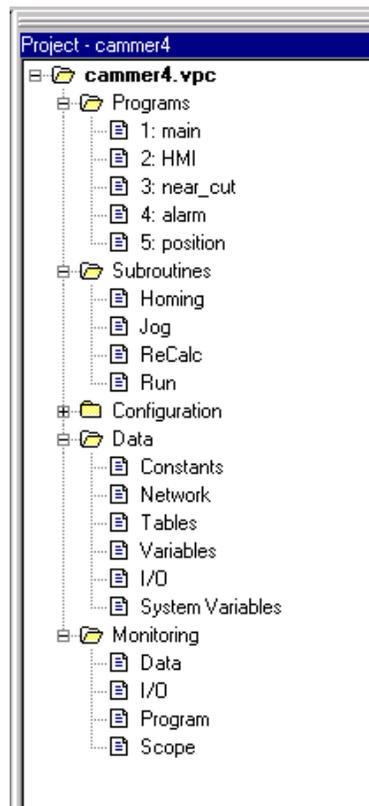
## 1.6 The Project Explorer

Project Explorer is used to navigate among windows in the main programming area. This window can be dragged, resized, docked, and undocked. All folders and their documents are shown below. To view any of the items below, double-click on the file.

### Accessibility

To open the Project Explorer, select

- From the Main Menu View > Project Explorer



### Programs

This folder contains a list of all programs within the project. A maximum of eight programs can be created.

### Accessibility

To view programs and create new programs:

- Right-click on the project name or the Programs folder. Select New Program from the sub-menu.

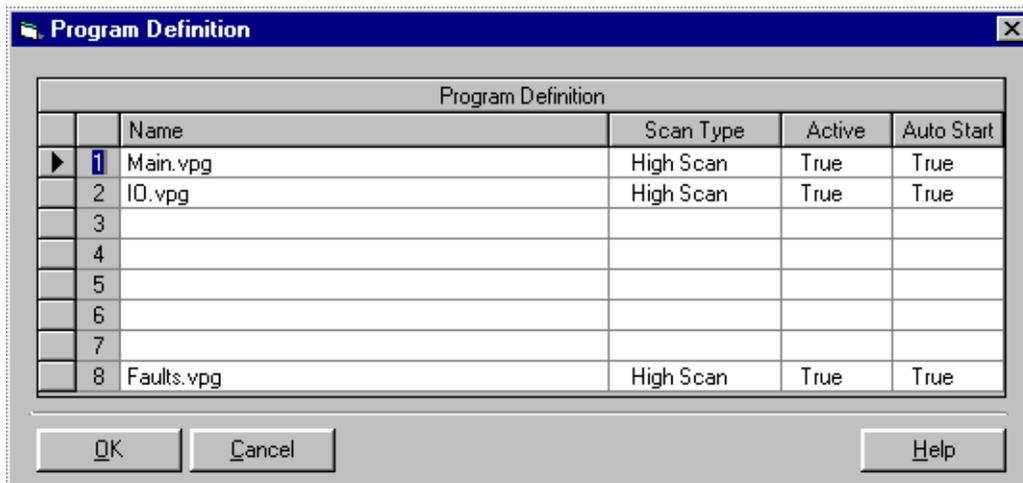
## Program Definition

The Program Definition window displays the properties of each program within the project. The properties may be edited on this screen.

### Accessibility

The Program Definition window is displayed as follows:

- Automatically appears when a new program is created
- From Project Explorer > right-click on the project name > select Program Definition from the sub-menu
- From Project Explorer > right-click on Programs > select Program Definition from the sub-menu
- From the Main Menu > Project > Program > Definition



### Scan Type

Select either High or Low scan for each of the projects programs. This configuration determines the interval that blocks are executed.

### Active

This setting controls if a program is included in the compile & download. Useful for debugging.

## **Auto Start**

The setting controls which programs start executing automatically when the power is applied. Programs set to “Auto Start=false” can be started later using the LAUNCH block.

## **Subroutine Definition**

When New Subroutine is selected, a new untitled program window is added to the project; the START and END blocks automatically appear. The File > Save Subroutine As function is recommended at this time. The new subroutine does not appear in Project Explorer until it has been named.

### **Accessibility**

To access the subroutine definitions, select:

- From the Main Menu > Project > New
- From the Standard Toolbar > New > Subroutine
- From the Project Explorer > right-click on the project name > Subroutine > New

## **Configuration**

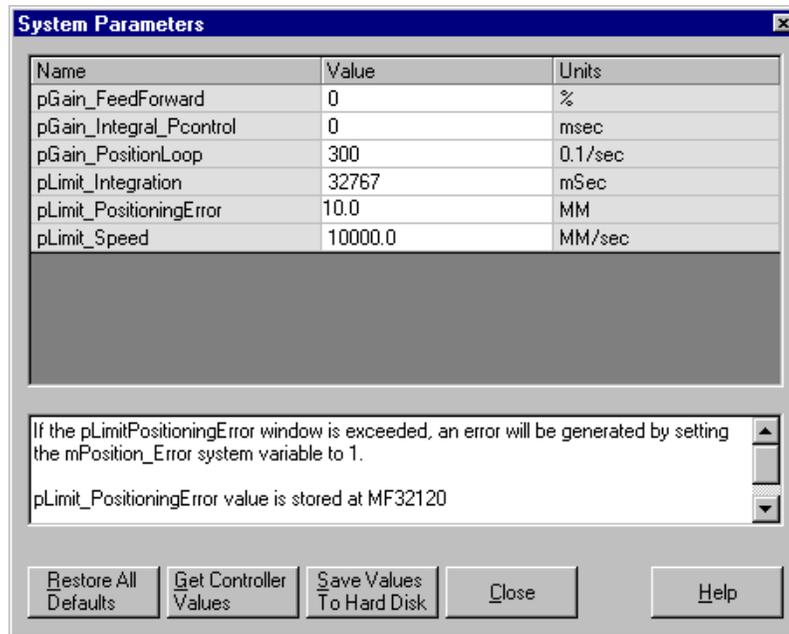
### **System Parameters**

The system parameters are mainly comprised of tuning gains and limit settings. The tool tip of each parameter shows the operation mode in which each gain applies. The parameters can be written to the hard disk. Parameters can also be read from or written to the controller.

### **Accessibility**

To access the system parameters, select:

- From the Project Explorer > Configuration > System Parameters
- From the Main Menu > Project > Configuration > System Parameters



### Important Note!!

**Parameters are updated in the controller after a value is changed if MotionWorks+™ is online with the controller. However, if the new values are to be saved with the project and retained after power cycle, the project must be downloaded.**

## System Properties

Systems properties allow the controller to be configured using the windows shown below.

### Accessibility

To access the System Properties, select

- From the Project Explorer > Configuration > System Properties
- From the Main Menu > Project > Configuration > System Properties
- System Properties already open, but not visible; From the Main Menu > Window List

Right clicking on any of the modules will display a menu with following choices:

- Get from the controller
- Import
- Export
- Restore default
- Send to controller

These menu items are available when right clicking on any of the components in system properties. Their actions affect only the component selected.

Important Note: Performing “Send to Controller” will suffice when making changes that are to be permanently stored in controller memory with the exception of the MP940 and External Encoder properties. For these two modules only, the changes are only temporary, meaning that the power on values will be from the last project download.

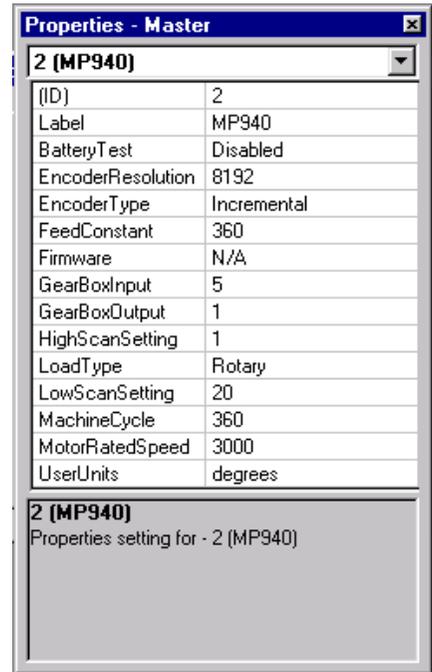
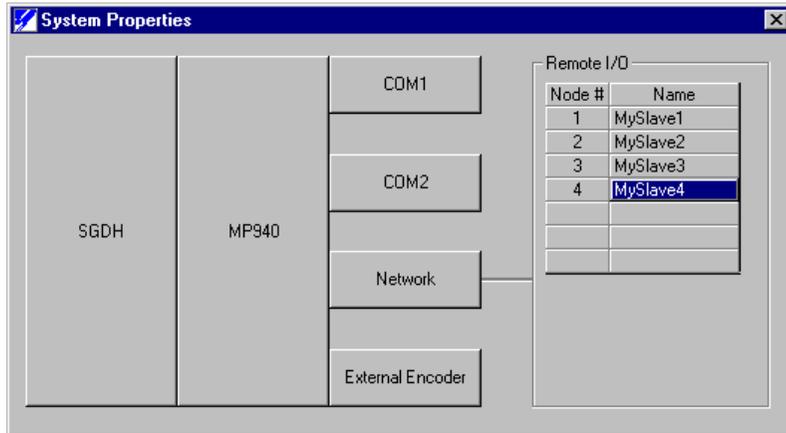
## Machine Cycles

MW+ does not support support machine cycles with fractional pulses. This may present a problem for applications such as flying shear / web handling / camming. In general, axes that are set to “Rotary” mode will suffer from fractional pulses being “lost” each machine cycle. To determine if the system has fractional pulses, perform the following check:

Encoder Resolution	Gear Box	Machine Cycle	Feed Constant	Machine Cycle Counts
--------------------	----------	---------------	---------------	----------------------

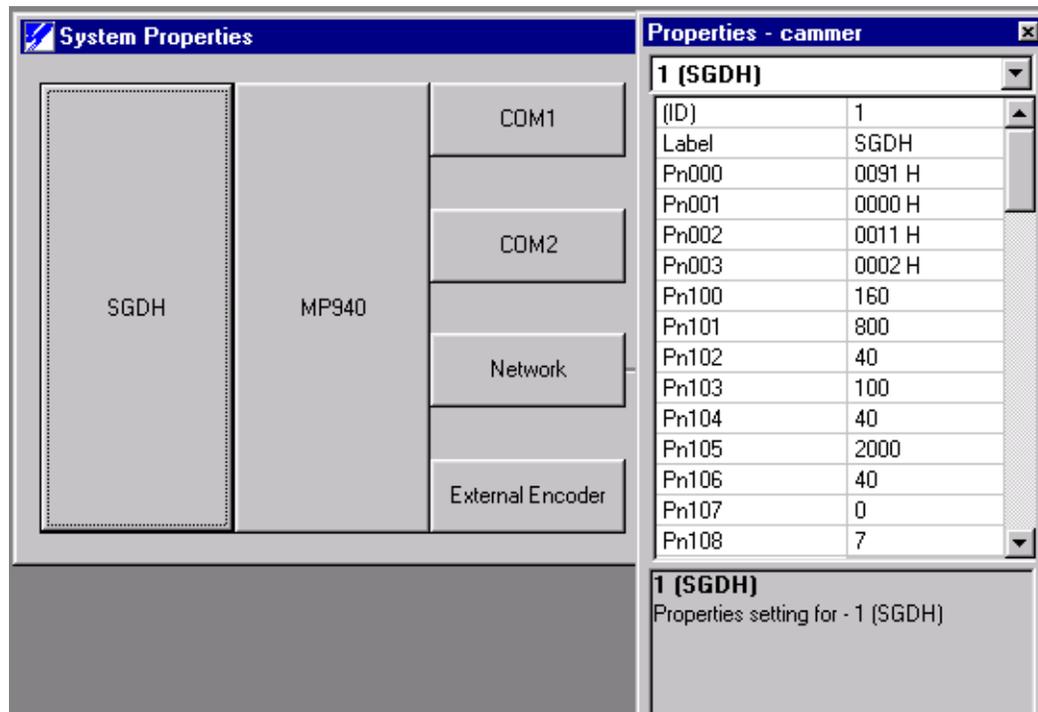
$$131072 \times \frac{43 \text{ Input}}{11 \text{ Output}} \times 17 \div \frac{360 \text{ Units}}{1 \text{ rev}} = 8710330.188888$$

As can be seen from the machine cycle count, a loss of .19 cycles will occur with each revolution. Compensate for this loss by using the Slave Offset function. For an introduction on using the Slave Offset function, see [“How does SLAVE OFFSET work? How would an application benefit from using slave offset?”](#) on page 220.



## SGDH

When the SGDH module is selected, the following properties appear in the properties window: When an MP940 controller is used with an SGDH, it is recommended to set the parameters of the amplifier through the controller instead of the front panel of the amplifier. This method ensures that parameter values are retained with the project and downloaded properly.



Property	Default	Min	Max	Details (online Help message)
Pn000	90	0	0FB1	Function Selection Basic Switches. See sections 5.1.1 and 5.3.5 of SGDH manual.
Pn001	0	0	1122	Function Selection Application Switches 1. See sections 5.1.2, 5.4.2, and 5.5.7 of SGDH manual.
Pn002	11	0	4113	Function Selection Application Switches2. See sections 5.2.8, 5.2.9, and 5.7.2 of SGDH manual.
Pn003	2	0	00FF	Function Selection Application Switches 3. See section 6.5 of SGDH manual.
Pn100	40	1	2000	Speed Loop Gain in Hz. This is an important component of servo tuning, this parameter acts as the derivative gain. See section 6.2.1 of SGDH manual.
Pn101	2000	15	51200	Speed Loop Integral Time Constant in 0.01ms. This is an important component of servo tuning. See section 6.2.1 of SGDH manual.

Property	Default	Min	Max	Details (online Help message)
Pn102	40	1	2000	Position Loop Gain in 1/s. This parameter is not applicable when using MP940. See section 6.2.1 of SGDH manual.
Pn103	100	0	10000	Inertia Ratio in percentage. See sections 6.2.1 and 6.3.3 of SGDH manual.
Pn104	40	1	2000	2nd Speed Loop Gain in Hz. This parameter is switchable with Pn100 based on user input. This is an important component of servo tuning, this parameter acts as the derivative gain. See section 6.2.1 of SGDH manual.
Pn105	2000	15	51200	2nd Speed Loop Integral Time Constant in 0.01 ms. This parameter is switchable with Pn101. This is an important component of servo tuning. See section 6.2.1 of SGDH manual.
Pn106	40	1	2000	2nd Position Loop Gain in 1/s. This parameter is not applicable when using MP940. See section 6.2.1 of SGDH manual.
Pn107	0	0	450	Bias in rpm. See section 6.2.4 of SGDH manual.
Pn108	7	0	250	Bias Width Addition in reference units. See section 6.2.4 of SGDH manual.
Pn109	0	0	100	Feed Forward in percentage. This parameter is not applicable when MP940 is used. Use pGain_FeedForward. See section 6.2.2 of SGDH manual.
Pn10A	0	0	6400	Feed-forward Filter Time Constant in 0.01ms. This parameter is not applicable when MP940 is used. See section 5.2.5 of SGDH manual.
Pn10B	0004	0000	2014	Gain-related Application Switches. See section 6.2.5 of SGDH manual.
Pn10C	200	0	800	Mode Switch Torque Reference in percentage. See section 6.2.5 of SGDH manual.
Pn10D	0	0	10000	Mode Switch Speed Reference in rpm. See section 6.2.5 of SGDH manual.
Pn10E	0	0	3000	Mode Switch Acceleration in 10rpm/s. See section 6.2.5 of SGDH manual.
Pn10F	0	0	10000	Mode Switch Error Pulse in reference units. This parameter is not applicable when using MP940. See section 6.2.5 of SGDH manual.
Pn110	0012	0000	2014	Online Autotuning Switches. See section 6.3.4 of SGDH manual.
Pn111	100	1	100	Speed Feedback Compensation in percentage. See section 6.2.6 of SGDH manual.
Pn200	0	0	1239	Position Control Reference Selection Switches. This parameter is not applicable when using MP940. See section 5.2.2 of SGDH manual.
Pn201	16384	16	16384	PG Divider in pulses per revolution. This parameter is not applicable when using MP940. See section 5.2.3 of SGDH manual.
Pn202	16384	16	16384	Electronic Gear Ratio (Denominator). This parameter is not applicable when using MP940. See section 5.2.5 of SGDH manual.

Property	Default	Min	Max	Details (online Help message)
Pn203	1	1	65535	Electronic Gear Ratio (Denominator). This parameter is not applicable when using MP940. See section 5.2.5 of SGDH manual.
Pn204	0	0	6400	Position Reference Accel/Decel Parameter in 0.01ms. This parameter is not applicable when using MP940. See section 6.1.2 of SGDH manual.
Pn205	65535	0	65535	Multi-turn Limit Setting in revolutions. This parameter is not applicable when using MP940. See section 6.1.2 of SGDH manual.
Pn206	16384	513	65535	Reserved in pulses per revolution. This parameter is not applicable when using MP940. See section 6.1.2 of SGDH manual.
Pn207	0	0	11	Position Control Function Switches. This parameter is not applicable when using MP940. See sections 5.2.9 and 6.1.2 of SGDH manual.
Pn208	0	0	6400	Position Reference Movement Averaging Time in 0.01ms. This parameter is not applicable when using MP940. See section 6.1.2 of SGDH manual.
Pn300	600	150	3000	Speed Reference Input Gain in 0.01volt / rated speed. See section 5.2.1 of SGDH manual.
Pn301	100	0	10000	Speed 1 in rpm. See section 5.2.6 of SGDH manual.
Pn302	200	0	10000	Speed 2 in rpm. See section 5.2.6 of SGDH manual.
Pn303	300	0	10000	Speed 3 in rpm. See section 5.2.6 of SGDH manual.
Pn304	500	0	10000	Jog Speed in rpm. See section 5.3.2 of SGDH manual.
Pn305	0	0	10000	Soft Start Acceleration Time in ms. See section 6.1.1 of SGDH manual.
Pn306	0	0	10000	Soft Start Deceleration Time in ms. See section 6.1.1 of SGDH manual.
Pn307	40	0	65535	Speed Reference Filter Time Constant in 0.01ms.
Pn308	0	0	65535	Speed Feedback Filter Time Constant in 0.01ms. See section 6.2.6 of SGDH manual.
Pn400	30	10	100	Torque Reference Input Gain in 0.1V / rated torque. See section 5.2.7 of SGDH manual.
Pn401	100	0	65535	Torque Reference Filter Time Constant in 0.01ms. See section 6.1.5 of SGDH manual.
Pn402	800	0	800	Forward Torque Limit in % of rated torque. See section 5.1.3 of SGDH manual.
Pn403	800	0	800	Reverse Torque Limit in % of rated torque. See section 5.1.3 of SGDH manual.
Pn404	100	0	800	Forward External Torque Limit in % of rated torque. See section 5.1.3 of SGDH manual.
Pn405	100	0	800	Reverse External Torque Limit in % of rated torque. See section 5.1.3 of the SGDH manual.
Pn406	800	0	800	Emergency Stop Torque in % of rated torque. See section 5.1.2 of SGDH manual.
Pn407	10000	0	10000	Speed Limit during Torque Control in rpm. See section 5.2.7 of SGDH manual.

Property	Default	Min	Max	Details (online Help message)
Pn408	0	0	1	Torque Function Switches. See section 6.1.6 in SGD manual.
Pn409	2000	50	2000	Notch Filter Frequency in Hz. Only effective when the SGD is used in torque mode. See section 6.1.6 in SGD manual.
Pn500	7	0	250	Positioning Completed Width in pulses. It is recommended to use sPosition_CompletionWindow instead. See section 5.5.3 of SGD manual.
Pn501	10	0	10000	Zero Clamp Level in rpm. See section 5.4.3 in SGD manual.
Pn502	20	1	10000	Rotation Detection Level in rpm. See section 5.5.5 of SGD manual.
Pn503	10	0	100	Speed Coincidence Signal Output Width in rpm. See section 5.5.4 of SGD manual.
Pn504	7	1	250	NEAR Signal Width in pulses. See section 5.5.8 of SGD manual.
Pn505	1024	1	32767	Overflow Level in 256 pulses. This parameter is not applicable when the SGD is used with the MP940. See section 6.2.1 of SGD manual.
Pn506	0	0	50	Brake Reference Servo OFF Delay Time in 10ms. See section 5.4.4 of SGD manual.
Pn507	100	0	10000	Brake Reference Output Speed Level in rpm. See section 5.4.4 of SGD manual.
Pn508	50	10	100	Timing for Brake Reference Output during Motor Operation in 10ms. See section 5.4.4 of SGD manual.
Pn509	20	20	1000	Momentary Hold Time in ms. See section 5.4.9 of SGD manual.
Pn50A	2881	0	FFFF	Input Signal Selections 1. See section 5.3.3 of SGD manual.
Pn50B	8883	0	FFFF	Input Signal Selections 2. See section 5.3.3 of SGD manual.
Pn50C	8888	0	FFFF	Input Signal Selections 3. See section 5.3.3 of SGD manual.
Pn50D	8888	0	FFFF	Input Signal Selections 4. See section 5.3.3 of SGD manual.
Pn50E	3211	0	3333	Output Signal Selections 1. See section 5.3.4 of SGD manual.
Pn50F	0	0	3333	Output Signal Selections 2. See section 5.3.4 of SGD manual.
Pn510	0	0	33	Output Signal Selections 3. See section 5.3.4 of SGD manual.
Pn511	6541	0	FFFF	Input Signal Selections 5. See section 5.3.3 of SGD manual.
Pn512	0	0	111	Output Signal Reversal Settings. See section 5.3.4 of SGD manual.
Pn600	0	0	9000	Regenerative Resistor Capacity in watts. See section 5.6.1 of SGD manual.

**COM1**

When the COM1 module is selected, the following properties appear in the properties window.

<b>Property</b>	<b>Default</b>	<b>Minimum</b>	<b>Maximum</b>	<b>Detail</b>
Address	1	0	63	—
Baud Rate	19200	9600	19200	—
Function	Slave	—	—	—
Protocol	Memobus	—	—	Fixed
Transmission Mode	RTU	—	—	Fixed

*Address:* Select between 0 and 63.

*Baud Rate:* The choices are 9600 and 19200bps.

*Function:* Fext at “Slave”.

*Protocol:* This is fixed at Memobus for MW+ projects.

*Transmission Mode:*  
This is fixed at RTU for MW+ projects.

## COM2

When the COM2 module is selected, the following properties appear in the properties window.

Property	Default	Minimum	Maximum	Detail
Address	1	0	63	—
Baud Rate	19200	9600	19200	bps
Function	Slave	—	—	Selection
Protocol	Memobus	—	—	Fixed
Transmission Mode	RTU	—	—	Fixed
Transmission Type	RS422	RS422	RS485	Selection

*Address:* Select between 0 and 63.

*Baud Rate:* The choices are 9600 and 19200

*Function:* Fixed at “Slave”.

*Protocol:* This is fixed at Memobus for MW+ projects.

*Transmission Mode:*  
This is fixed at RTU for MW+ projects.

*Transmission Type:*  
RS 422 and RS 485 are the 2 choices.

**MP940**

When the MP940 module is selected, the following properties appear in the properties window.

Note: When right clicking on this module, the properties can be sent to the controller, but only to RAM. Compile & Download is required to retain values after power cycle.

Property	Default	Minimum	Maximum	Detail
Battery Test	Disabled	Enabled	Disabled	Setting
Encoder Resolution	8192	8	2147483647	Counts
Encoder Type	Incremental	Absolute	Incremental	Selection
Feed Constant	1	0.001	8338608	Constant
Gear Box Input	1	1	32767	Numerator
Gear Box Output	1	1	32767	Denominator
High Scan Setting	1	1	32	ms
Load Type	Linear	Linear	Rotary	Setting
Low Scan Setting	20	10	100	ms
Machine Cycle	1	0.001	2147483647	Modulus
Motor Rated Speed	3000	1000	10000	rpm
User Units	Inch	N/A	N/A	—

*Battery Test:* If enabled, the controller tests the battery voltage and sets the battery alarm LED on the front of the controller accordingly, and sets mAlarm\_Battery.

If disabled, the battery alarm LED does not light.

*Encoder Resolution:* Set the quadrature number of encoder pulses per motor revolution.

*Encoder Type:* Select either “incremental” or “absolute”. This setting determines whether the MP940 automatically reads the absolute encoder at power up.

*Feed Constant:* The number in user units that the load travels for each revolution of the **final output shaft** of the mechanical system.

Final Output Shaft	Detail	Feed Constant
Ball screw	6mm pitch	6
Conveyor belt	roller = 4in diameter	$4 \times \pi$
Belt and pulley	last pulley = 10in diameter	$10 \times \pi$

*Gear Box Input:* If a mechanical gear box is used, enter the value which corre-

sponds to the number of times the input shaft rotates for the number of times the output shaft rotates. These are integer values. If a 10:1 gear box is used, enter “10” for the gear box input.

**Gear Box Output:**

If a mechanical gear box is used, enter the value which corresponds to the number of times the input shaft rotates for the number of times the output shaft rotates. These are integer values. If a 10:1 gear box is used, enter “1” for the gear box output.

*High Scan Setting:*

This value determines the interval at which blocks are executed in high scan programs. One block from each high scan program will execute in this time interval, some will take longer based on specific block conditions.

*Load Type:*

Select either linear or rotary. Linear implies an axis that has finite travel range. Rotary implies an axis that rotates in one direction only, but with a cyclic period. When “Rotary” is selected, the position data is automatically modularized to fit within one cyclic period or machine cycle.

*Low Scan Setting:*

This value determines the interval at which blocks are executed in low scan programs.

*Machine Cycle:*

This is the maximum position value of the main axis. When this value is reached, the position will indicate “0”. Machine cycle is only applicable if Load Type is set to “Rotary”. If the application is a rotary table, this value may be the same as the Feed Constant. Note: If the machine cycle in user units does not equate to an integer number of pulses, the positioning units will drift. In this case, program an offset at periodic intervals to compensate for lost pulses. See note at the beginning of the System Property section about Machine Cycle limitations.

*Motor Rated Speed:*

Enter the rated speed of the motor in rpm.

*User units:*

Select from “centimeters”, “degrees”, “inches”, “microns”, “millimeters”, or “pulses”. Custom units may also be entered.

## External Encoder

When external encoder is selected, the following properties appear in the properties window.

These properties are effective for both real and virtual encoders. A virtual encoder is useful for simulating a machine or creating a time based cam profile. The system properties for External encoder apply for either mode, making the user units the same. The virtual encoder is activated by setting the system variable sExternalMode to “1”.

Note: When right clicking on this module, the properties will be sent to the controller, but only to RAM. Compile & Download is required to retain values after power cycle.

Property	Default	Minimum	Maximum	Detail
Enabled	False	False	True	Setting
Feed Constant	1	0.001	8338608	Constant
Gear Box Input	1	1	32767	Numerator
Gear Box Output	1	1	32767	Denominator
Machine Cycle	1	.001	2147483647	Modulus
Movement Type	Rotary	Linear	Rotary	Setting
Pulse Type	Quadrature	Pulse & Direction	Quadrature	Setting
Resolution	2048	8	2147483647	Counts
User Units	Degrees	N/A	N/A	—

*Enabled:* If *Enabled* is “True”, the MP940 expects an encoder to be connected (i.e., an alarm is set if no encoder is connected). If a virtual encoder is used, set enabled=false, or an A9F alarm will result.

*Feed Constant:* The number in user units that the load travels for each revolution of the *final output shaft* of the mechanical system.

Final Output Shaft	Detail	Feed Constant
Ball screw	6mm pitch	6
Conveyor belt	roller = 4in diameter	$4 \times \pi$
Belt and pulley	last pulley = 10in diameter	$10 \times \pi$

*Gear Box Input:* If a mechanical gear box is used, enter the value which corresponds to the number of times the input shaft rotates for the number of times the output shaft rotates. These are integer values. If a 10:1 gear box is used, enter “10” for the gear box input.

- Gear Box Output:* If a mechanical gear box is used, enter the value which corresponds to the number of times the input shaft rotates for the number of times the output shaft rotates. These are integer values. If a 10:1 gear box is used, enter “1” for the gear box output.
- Machine Cycle:* This is the maximum position value of the external axis. When this value is reached, the position wraps back to “0”. Machine cycle is only applicable if Movement Type is set to “Rotary”. If the application involves camming, the machine cycle may be the product length. Note: If the machine cycle in user units does not equate to an integer number of pulses, the positioning units will drift. In this case, program an offset at periodic intervals to compensate for lost pulses. See note at the beginning of the System Property section about Machine Cycle limitations.
- Movement Type:* Select either “linear” or “rotary”. Rotary implies an axis that rotates in one direction only, but with a cyclic period. The position data is automatically modularized to fit within one cyclic period or machine cycle when rotary is selected.
- Pulse Type:* Select “Quadrature”, “Reverse Quadrature”, “Pulse & Direction”, or “Reverse Pulse & Direction”.
- Resolution:* The number of quadrature pulses per revolution of the external encoder.
- User Units:* Select from “centimeters”, “degrees”, “inches”, “microns”, “millimeters”, or “pulses”. Custom units may also be entered.

## Network

Property	Default	Minimum	Maximum	Detail
Cycle Time	10	0	1000	mSec
Enabled	False	False	True	Setting
Function	Slave	Master	Slave	Selection
Input Words	8	0	8 / 512	Word count
Message Type	Polled	Polled	Strobed	Devicenet
Name	“Network”	0 Characters	8 Characters	—
Node	1	0	99	—
Output Words	8	0	8 / 512	Word count
Refresh Rate	Low	Low	High	Scan
Slave Nodes	N/A	1	99	—
Type	Mechatrolink	DeviceNet	Mechatrolink	Selection

*Cycle Time:* Applicable for Devicenet networks when the MP940 is acting as the master. This is the update rate of the network and effects traffic load.

*Enabled:* Select whether the network is enabled or not.

*Function:* If Function is set to “Master”, the default Node is automatically set to “0”. If Function is set to “Slave”, Node is automatically set to “1” only if Node is currently set to “0”.

*Message Type:* The property is only applicable when the “Type” property is set to Devicenet. Select either Polled or Strobed. This setting determines how messages from the master are send. Strobed can increase network efficiency if there are several identical slaves.

*Input Bytes/Words:* This property is only enabled if the Function property is set to “Slave”. It defines the number of input bytes/words for the MP940 as a slave. If the Function property is set to “Master” then the total input bytes/words are determined from the Slave Node list. Note: If network Type is set to DeviceNet, this property defines the number of Input Bytes.

*Name:* Provide a description of the network device that will be useful in identifying this node. The name will also be used if the network variable “Allocation Wizard” is used.

- Node:* Set the address of the module. The maximum value is dependant on the Type property. Mechatrolink is “29”; DeviceNet is “64”.
- Output Bytes/Words:* This property is only enabled if the Function property is set to “Slave.” It defines the number of output bytes/words for the MP940 when configured as a slave. If the Function property is set to “Master,” the total output words are determined from the Slave Node list. Note: If the network Type is DeviceNet, this property defines the number of Output Bytes.
- Refresh Rate:* Select either “High” or “Low” scan. This is the period in which all network data are updated.
- Slave Nodes:* This property is only enabled if the Function property is set to “Master.” It defines the number of slaves on the network. If Function is set to “Slave,” then Nodes automatically displays “N/A”.
- Type:* Select either DeviceNet or Mechatrolink.

## Remote I/O

Remote I/O only appears on the System Properties window if the Network *Function* is set to “Master”.

Property	Default	Minimum	Maximum	Detail
Message Type	Polled	Polled	Strobed	Devicenet
Input Words	N/A	0	512	Word
Name	N/A	0 characters	8 Characters	—
Node	1	1	99	—
Node Type	N/A	See list below	See list below	Selection
Output Words	N/A	0	512	Word

*Message Type:* The property is only applicable when the Network “Type” property is set to “Devicenet.” Select either Polled or Strobed. This setting determines how messages from the master are send. Strobed can increase network efficiency if there are several identical slaves.

*Input Words:* This property is only enabled if the Node Type property is set to “User Defined”. This is the number of input words allocated to this node. Note: If the network Type is DeviceNet, this property defines the number of Input Bytes.

*Name:* Provide a description of the network device that will be useful in identifying this node. The name will also be used if the network variable “Allocation Wizard” is used.

*Node:* Set the address of the module. This applies to DeviceNet and Mechatrolink. The maximum value is dependant on the Type property. Mechatrolink is “29”; DeviceNet is “64.”

*Node Type:* This is a list of all available Mechatrolink nodes. When a particular module is selected, the input and output words are automatically set.

Mechatrolink Node	Input Words	Output Words
16pt 120VAC Input	1	0
16pt 240VAC Input	1	0
16pt Digital Input	1	0
16pt Digital Output	0	1
2ch Analog Output	2	4
4ch Analog Input	7	2
64/64pt Digital Input/Output	4	4
8pt 120-240VAC Output	0	1

Mechatrolink Node	Input Words	Output Words
8pt Relay Output	0	1
MP940	8	8
User Defined	Undefined	Undefined

*Output Words:* This property is only enabled if the Node Type property is set to “User Defined.” This is the number of output words allocated to this node. Note: If the network Type is DeviceNet, this property defines the number of Output Bytes.

## Data

There are six data types available for use throughout the program:

Constant
I/O
Network
System Variable
Table
Variable

Each is described in the following sections.

### General Information

- Data names may be up to 32 characters.
- All names are case sensitive.
- All names must be unique across all data types.

Data Type	Minimum	Maximum
bit	0	1
float	-3.402832e-38	3.402823e38
integer	-32768	32767
long	-2147483648	2147483647

The resolution for data defined as “float” is seven significant digits as shown above.

To delete data, click on the gray cell to the left of the data column in the row to be deleted, and press the **Delete** key.

To sort data, click on the title row of the column by which sorting should occur.

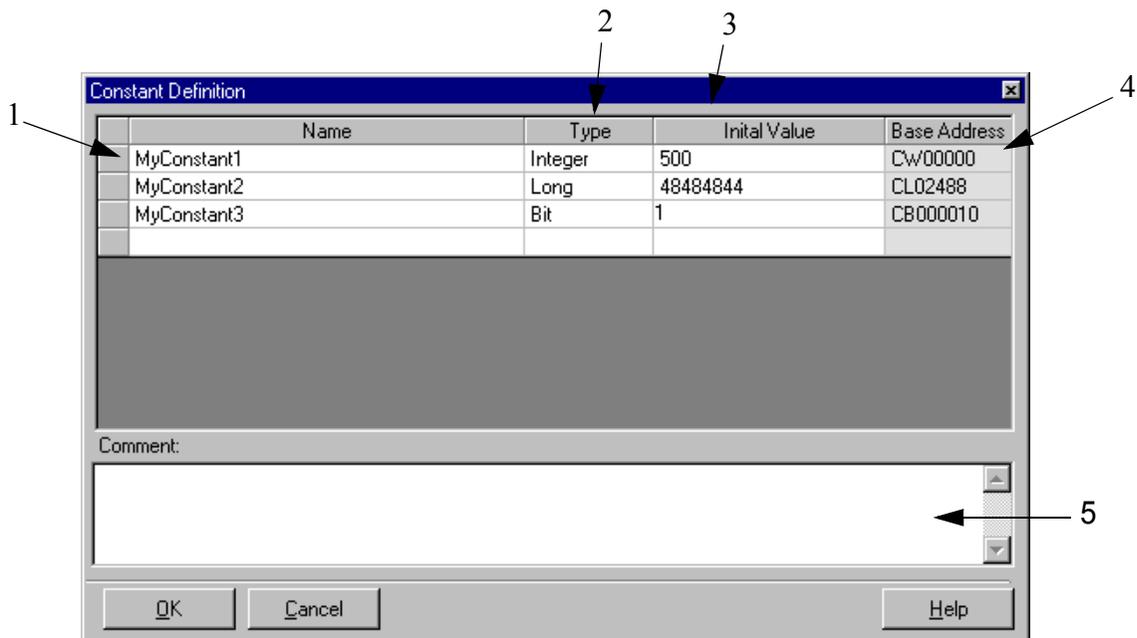
## Constant Definition

Constants are added using the Constant Definition window that appears below. The memory range for constants is 32,767 integers. If longs, floats, and bits are used, the maximum number of constants is affected accordingly.

### Accessibility

To access the constant definitions.

- From the Project Explorer > Data > Constants
- From the Main Menu > Project > Data > Constants



#### 1. Name

The user-given field name of the constant.

#### 2. Type

This field determines the potential magnitude of the constant.

#### 3. Value

This field sets the value of the constant. Constants cannot be changed by the program at run time.

#### 4. Base Address

This field is a read only field indicating the register location of the constant in the controller memory. This is provided for debugging with MotionWorks or accessing data via Memobus serial communication.

## 5. Comments

This field provides a location to document constant usage.

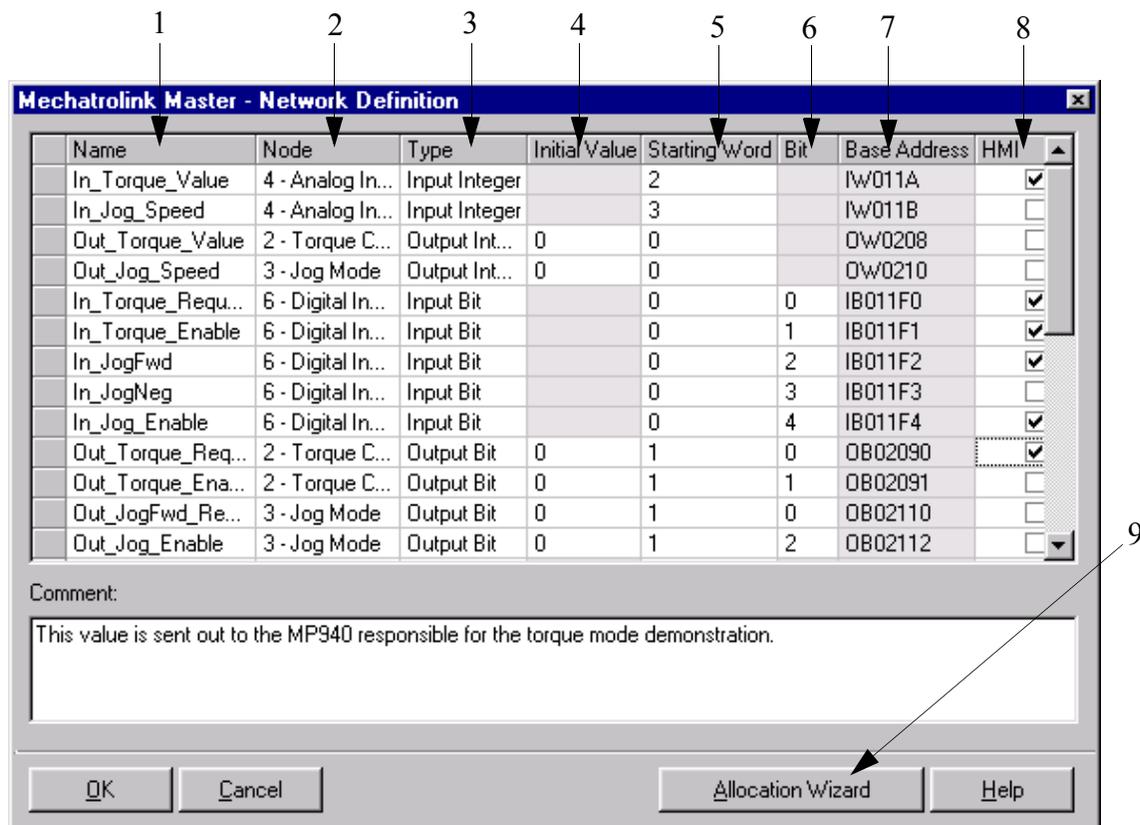
## Network Definition

Network data are added using the Network Definition window that appears below.

### Accessibility

To access the network definitions, select:

- From the Project Explorer > Data > Network
- From the Main Menu > Project > Data > Network



### 1. Name

The user-given name of the network data.

### 2. Node

This column only appears if Network Function is set to “master.” The cells in this column contain a drop-down box that lists the remote I/O nodes configured.

### 3. Type

This field determines the potential magnitude of the data.

#### 4. Initial Value

This field presets the value of the data when the power is turned on. If no initial value is entered, the data is not initialized when the power is turned on (which is useful for preserving data values when the power is cycled).<sup>1</sup>

#### 5. Starting Word

The cells in this column contain a drop-down box. The contents are a list of valid words starting at zero. The maximum value is derived from either of two places, depending on the configuration of the controller, as follows:

- If the controller is configured as a network master, the Starting Word range is defined by the System Properties > Remote I/O Block > Input/Output Word properties.
- If the controller is configured as a network slave, the Starting Word range is defined by the System Properties > Network Block > Input/Output Word properties.

#### 6. Bit

This selection is only available if the Type is “input bit” or “output bit”. Select “0” through “F” from the drop-down list. If a bit type variable is not defined, this cell appears gray, and cannot be set. Also, under DeviceNet configuration, if an odd number of bytes are defined, the drop-down list only shows “0-7” if the final “Starting Word” is selected.

#### 7. Base Address

This field is a read only field indicating the register location of the data in the controller memory. This is provided for debugging with MotionWorks™ or accessing data via Memobus serial communication. Base Address is determined by the information specified in the Node, Starting Word, and Bit columns.

#### 8. HMI Checkbox

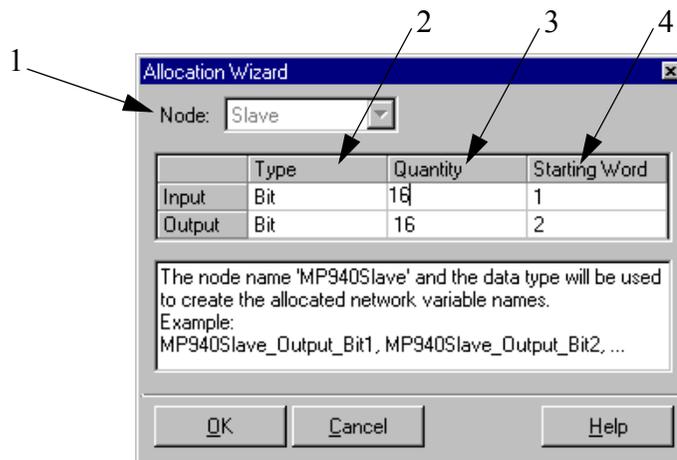
This field is used to set data to be written to the HMI export file.

## Allocation Wizard

If several network data of the same type are required, the Allocation Wizard can automatically create data names based on user-provided input. The automatically-created names include the node name and I/O type.

1. Possible only when the battery option is installed to preserve RAM when power is OFF.

For example, the created names may appear as in the text box of the figure below.



### 1. Node

This drop-down box contains the node number and name of each slave as defined in System Properties of the Configuration. If the controller is configured as a network slave, the *Node* box is disabled.

### 2. Type

This field determines the potential magnitude of the data.

Data Type	Minimum	Maximum
bit	0	1
float	-3.402832e-38	3.402823e38
integer	-32768	32767
long	-2147483648	2147483647

### 3. Quantity

This entry is limited by the Network or Remote I/O configuration properties, depending on whether the controller is configured as a master or a slave.

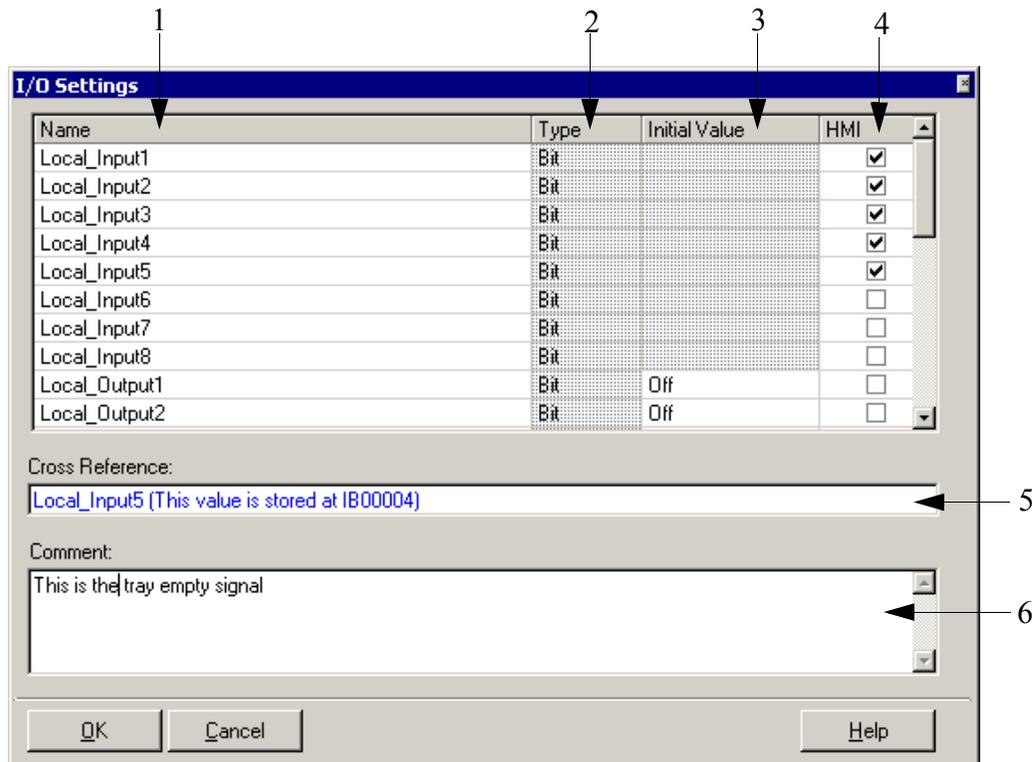
When a bit type is selected and the quantity is greater than the number of bits in a word, allocation automatically proceeds to the next word in sequence. Note: for remote I/O data, more than one data name can be assigned to the same memory location.

### 4. Starting Word

This column contains a list of possible starting words. The list is created based on the input/output properties in the configuration

## I/O Definition

Digital and analog input/output names can be reviewed and edited on the window shown below.



### 1. Name

The inputs/outputs are assigned default names by MotionWorks+™ when a project is created. These names appear in the list shown below. The names can be changed to any character string that more accurately describes their function.

### 2. Type

This field indicates whether the name refers to an individual input or output bit, or an entire word, which contains up to 16 input or output bits. If the I/O is an analog type, the 16-bit word contains the binary representation of the analog voltage.

### 3. Initial Value

This field only applies to outputs. Set the power ON logic state in this field. The default output state is OFF.

#### 4. HMI Checkbox

This field is used to set data to be written to the HMI export file.

#### 5. Cross Reference

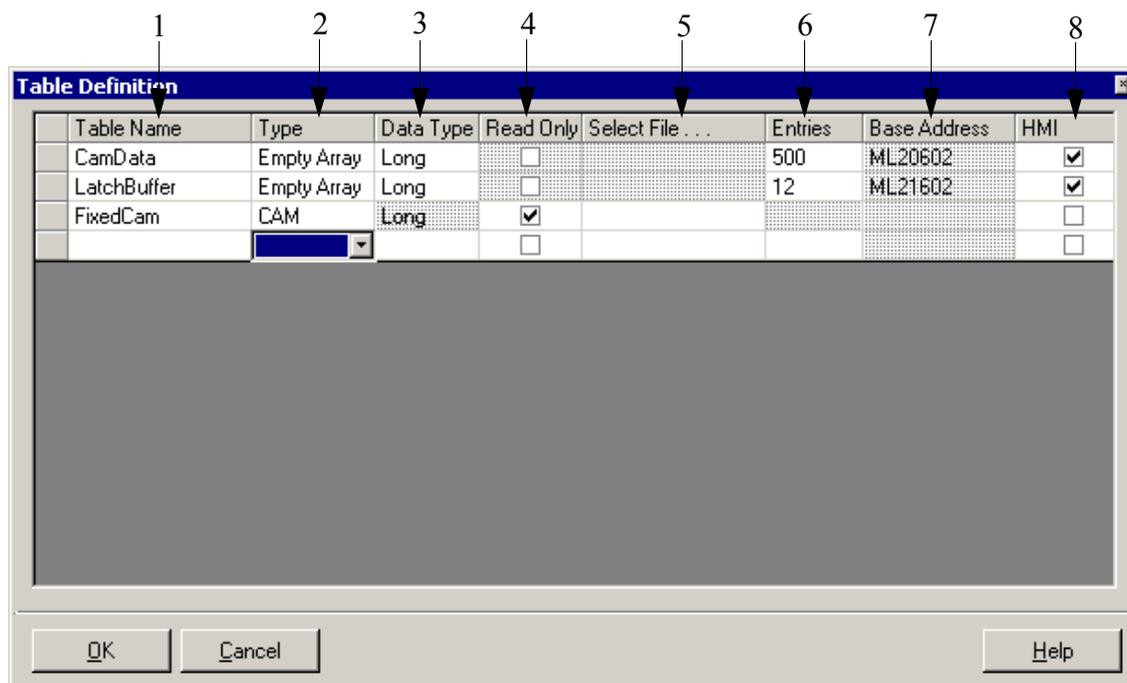
This field retains the default I/O name assigned by MotionWorks+™ and displays the register location of the I/O in the controller. This is provided for debugging with MotionWorks™ or accessing data via Memobus serial communication.

#### 6. Comments

This field provides a location for users to document I/O usage.

### Table Definition

Table data is an array of values, all of which have the same name. They are accessed by using the syntax “MOVE[i] = n”, where “i” is an integer or long, and “n” is a valid value for the table. There are two table formats available - constant and dynamic. If a constant table is defined (read only = true), the table data must come from a .cdd, .txt, .dat, or .csv file. Constant table data cannot be modified by the program. If the table is configured as a dynamic table (read only = false), the data can optionally be loaded from a file, or generated at run time with SET VARIABLE blocks. Note: Tables are zero based, and no bound-



ary checking is provided. Care must be taken by the programmer to remain within the bounds of the table, or other data may get corrupted.

#### 1. Table Name

The user-given name of the table.

**2. Type**

Select from “.cam”, “.txt”, “.dat”, “.csv”, or “Empty Table”. A CAM table is imported from an \*.cdd file and follows the cam tool format. A .csv file is an ASCII file with data separated by commas. An empty table is either downloaded at run time or generated within the MW+™ program with SET VARIABLE blocks.

**3. Data Type**

This field determines the potential magnitude of each item in the table.

**4. Read Only**

This field determines whether items in the table can be changed by the program at run time.

**5. File Name**

For Cam (.cdd) or .csv tables, select the file from which data is imported.

**6. Entries**

Select the maximum number of items the table can contain. If a file is imported, the *Entries* property is automatically determined by the size of the file.

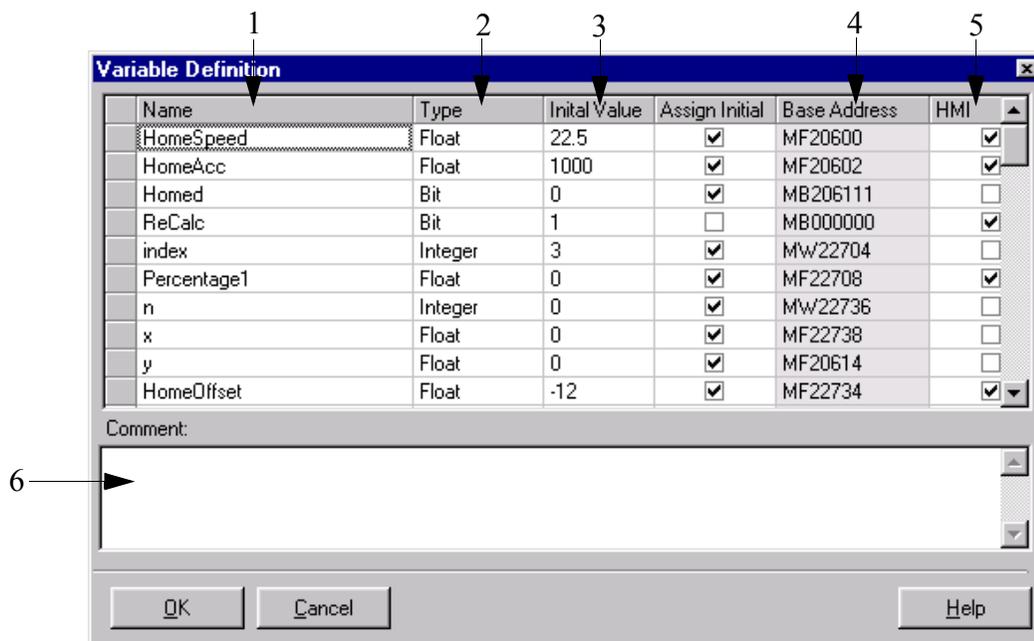
**7. Base Address**

This field is a read only field indicating the starting register location of the table in the controller memory. This is provided for debugging with MotionWorks™ or accessing data via Memobus serial communication.

**8. HMI Checkbox**

This field is used to set data to be written to the HMI export file.

## Variable Definition



### 1. Name

This field is the user-given name of the data.

### 2. Type

This field determines the potential magnitude of the data.

### 3. Initial Value

This field presets the value of the variable when the power is turned on. If no initial value is entered, the data is not initialized when the power is turned on (which is useful for preserving values when the power is cycled).<sup>1</sup>

### 4. Base Address

This field is a read only field indicating the register location of the data in the controller memory. This is provided for debugging with MotionWorks™ or accessing data via Memobus serial communication.

Note: If a bit type variable is defined and checked as an HMI variable, the address of the bit will be moved down to the region MB00000 ~ MB20470.

### 5. HMI checkbox

This field is used to set data to be written to the HMI export file.

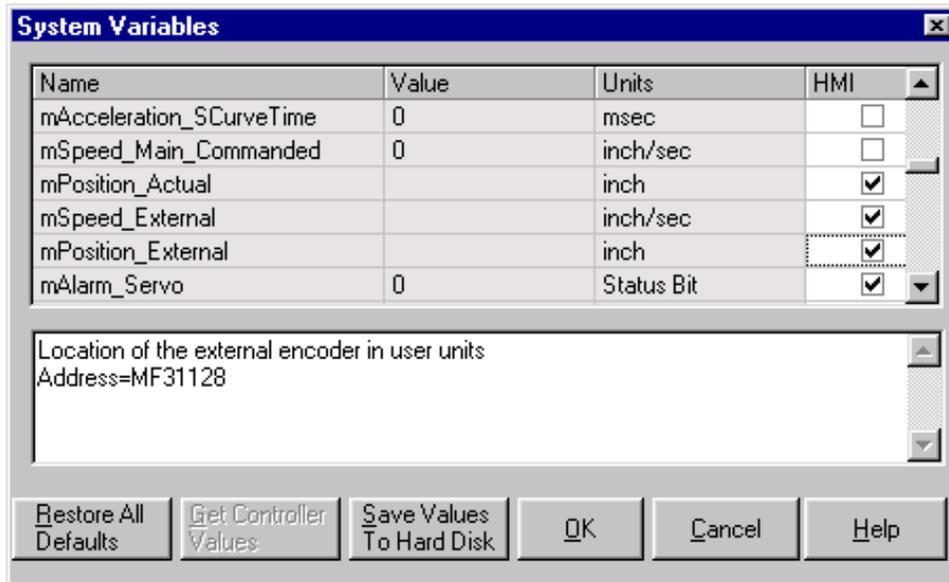
### 6. Comments

This field permits users to document variable usage.

1. Possible only when the battery option is installed to preserve RAM when power is OFF.

## System Variables

System variables are internal keywords which can be stored or accessed in expressions at run time. The names of the system variables cannot be changed by the user.



## 1.6.1 HMI Data

### Accessibility

To access the HMI data, select:

- From the View Menu > HMI Data List

### General

Each data item in the project can be identified as used in conjunction with a Human-Machine-Interface (HMI) by checking the box located next to each user variable or system variable. The HMI data export will automatically occur each time the project is saved.

### Setting data for HMI Export

Simply check the box at the right side of the row.

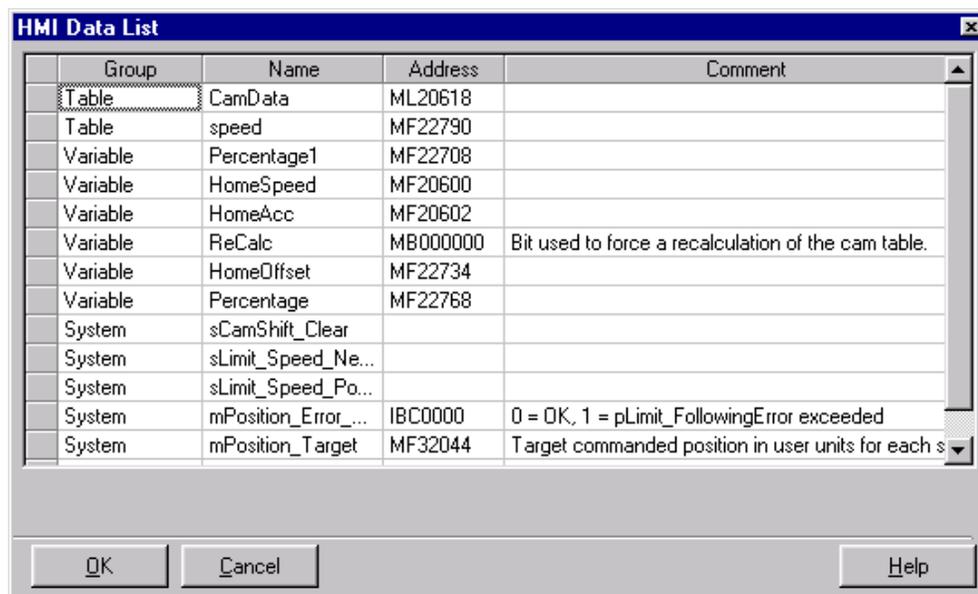
### HMI Export File

Each data type excluding constants has a check box at the right of the line item that can be used to identify project data as exportable to a comma separated value (CSV) file. This file, (called HMI\_xx.DAT) is automatically saved when the project is saved. The following is a description of the file format.

## Viewing HMI data

All data marked for HMI export are identified by the check mark. Alternatively, a window is available to view all items marked for HMI export under the View menu. Importing data is possible by reading a file with the same format that is exported. This can be accomplished by right clicking on the Project Explorer's Data Folder.

The data on this screen is for viewing purposes only, except that the user will be able to remove items from the HMI data list by right clicking on one or several selected rows and choosing "Remove from HMI list" from a sub menu. This has no effect on the data in the project, other than to clear the checkmark on the corresponding data definition window.



The screenshot shows a dialog box titled "HMI Data List" with a table containing the following data:

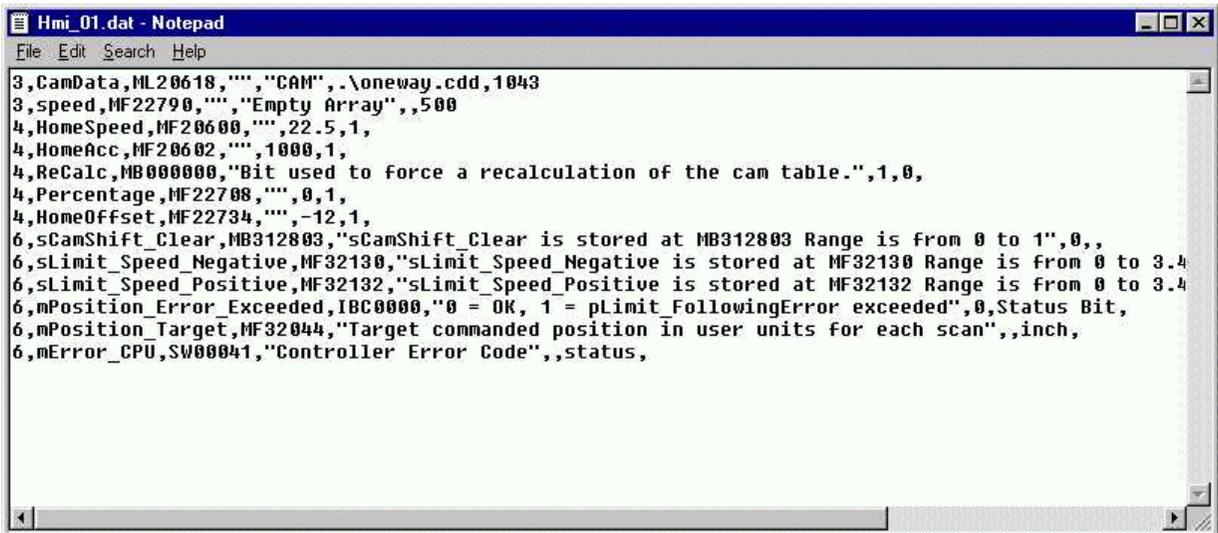
Group	Name	Address	Comment
Table	CamData	ML20618	
Table	speed	MF22790	
Variable	Percentage1	MF22708	
Variable	HomeSpeed	MF20600	
Variable	HomeAcc	MF20602	
Variable	ReCalc	MB000000	Bit used to force a recalculation of the cam table.
Variable	HomeOffset	MF22734	
Variable	Percentage	MF22768	
System	sCamShift_Clear		
System	sLimit_Speed_Ne...		
System	sLimit_Speed_Po...		
System	mPosition_Error_...	IBC0000	0 = OK, 1 = pLimit_FollowingError exceeded
System	mPosition_Target	MF32044	Target commanded position in user units for each s

At the bottom of the dialog box are three buttons: "OK", "Cancel", and "Help".

## File Data Format

The imported/exported CSV file will be of the following format:

<b>(Network)</b>	Code=2	Name	Address	Comment	Initial Value	Network Node	Starting Word
<b>(Table)</b>	Code=3	Name	Address	Comment	Table Type	FileName	Entries
<b>(Variable)</b>	Code=4	Name	Address	Comment	Initial Value		
<b>(I/O)</b>	Code=5	Name	Address	Comment	Initial Value		
<b>(System)</b>	Code=6	Name	Address	Comment	Initial Value		



```

Hmi_01.dat - Notepad
File Edit Search Help
3,CamData,ML20618,"","CAM",.\oneway.cdd,1043
3,speed,MF22790,"","Empty Array",,500
4,HomeSpeed,MF20600,"",22.5,1,
4,HomeAcc,MF20602,"",1000,1,
4,ReCalc,MB000000,"Bit used to force a recalculation of the cam table.",1,0,
4,Percentage,MF22708,"",0,1,
4,HomeOffset,MF22734,"",-12,1,
6,sCamShift_Clear,MB312803,"sCamShift_Clear is stored at MB312803 Range is from 0 to 1",0,,
6,sLimit_Speed_Negative,MF32130,"sLimit_Speed_Negative is stored at MF32130 Range is from 0 to 3.4
6,sLimit_Speed_Positive,MF32132,"sLimit_Speed_Positive is stored at MF32132 Range is from 0 to 3.4
6,mPosition_Error_Exceeded,IBC0000,"0 = OK, 1 = pLimit_FollowingError exceeded",0,Status Bit,
6,mPosition_Target,MF32044,"Target commanded position in user units for each scan",,inch,
6,mError_CPU,SW00041,"Controller Error Code",,status,

```

## Variable Mapping

If a bit type variable is set for HMI usage, it will be mapped within the range of MB000000 through MB02047F. This restriction will be enforced because Memobus can only write bits in the range of MB000000 through MB04096F. When Memobus writes bits outside of this range, the entire word is written, not just the bit, so there is a possibility of corrupted data.

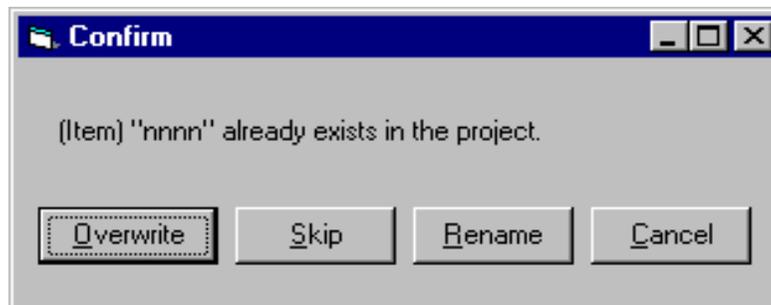
## Importing an HMI Data List

### Accessibility

To import an HMI data list, select:

- From the Project Menu > Data > Import HMI
- From the Project Explorer > Right click on an item in the Data folder and select “Import.”

Import Method: If there are no variables, network, or tables in the project, the import function will proceed without dialog for each type of item that has no existing entries. If there are existing entries, the import will append to the list. If names already exist in the current project, notification will be given to the user by the MW+ Importing Error dialog box shown below.



## 1.6.2 Import/Export Initiation

### General

The import utility will determine what category each imported value falls in.

### All Data Groups

Import will be initiated from Project>Data drop down menu or by right clicking on the Data object in the Project Explorer.

### Individual Data Group

Import will be initiated from Project>Data drop down menu which will expand to show all data types when the user moves the mouse over “Data”, or by right clicking on the specific Data object in the Project Explorer.

Name and address checking will be done on the full data area (Constants, Tables, Network, Variables, I/O, System Variables, System Parameters, System Ladder Registers and System Registers). The name and the address must be unique across the full data area, with the exception of overlapping addresses in the Network data area.

The register range for MW+ system registers (MW20000 ~ MW20599) and the register range for system ladder registers (MW30000 ~ MW32767) are reserved so that user variables are not allowed in these areas; however, the user will be allowed to import values for the system ladder type variables.

### Constants

The name, initial value, address, and comment will be imported. The type will be derived from the address.

### Network

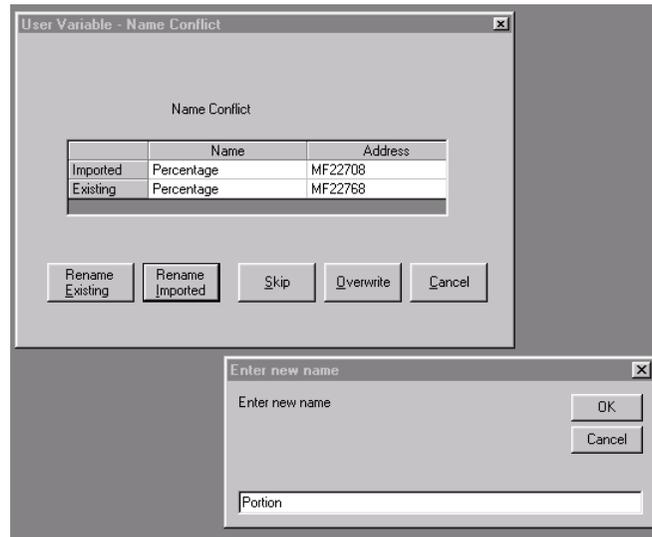
The name, node, initial value (if present), address, starting word, and comment will be imported. The type and bit will be derived from the address.

If configuration for the node is completed and configuration for that node matches configuration from the existing project, import of the network will proceed.

If the configuration for that node does not exist or does not match the configuration from the existing project, a message box will appear stating “The data for node x cannot be imported, because the configuration of node x is missing or does not match.” The user can choose to cancel, skip node, or import the configuration.

1. If the name is unused, the import of that item will proceed without dialog.

- If the imported name and address exactly match an item in the database, the value and the comment will be imported.
- If the imported name matches an item in the database but the address does not, the user will be shown the following grid with the existing name and address and the importing name and address.



A message will be displayed stating the conflict, and the user will be allowed to choose from the following options:

### **Rename Existing**

This button will present the user with a text box where a new name for the existing item can be typed.

### **Rename Imported**

This button will present the user with a text box where a new name for the importing item can be typed.

### **Skip**

Skips importation of this network.

### **Overwrite**

Imports the table and removes the existing item(s).

### **Cancel**

The import operation is canceled leaving the data area unchanged.

- Address overlap will be allowed as currently implemented for the network data area.

## Tables

The name, filename (if present), address, table type, number of entries, and comment will be imported. The type and read only status will be derived from the address. If the filename is present the file will be copied to the project directory, and the data will be transferred with the project.

1. If the name and the full address range is unused the import of that item will proceed without dialog. The full address range check will look for bits, words, longs, and floats that overlap the imported address range.
2. If the imported name and address exactly match an item in the database, the filename (if present), and the comment will be imported.
3. If the imported name matches an item in the database but the address range does not, the user will be shown the following grid with the existing name, number of entries (if table type) and address and the importing name and address range.

	Name	Entries	Address Range
Imported	Position	10	MW2001 ~ MW2010
Existing	Position	N/A	MB1001

A message will be displayed stating the conflict, and the user will be allowed to choose the following options:

### **Rename Existing**

This button will present the user with a text box where a new name for the existing table can be typed.

### **Rename Imported**

This button will present the user with a text box where a new name for the importing table can be typed.

### **Skip**

Skips importation of this table.

### **Overwrite**

Imports the table and removes the existing item(s).

### **Cancel**

The entire import operation will be canceled leaving the data area unchanged.

1. If the imported address matches any existing item within the address range, the following grid will be displayed with all of the existing name(s) and address(s) and the importing name and address.

	<b>Name</b>	<b>Entries</b>	<b>Address</b>
<b>Imported</b>	MyTable1	100	MW1000 ~ MW 1099
<b>Existing</b>	MyTable2	10	MW1000 ~ MW 1009
<b>Existing</b>	MyWord1	N/A	MW1010

A message will be displayed stating the conflict, and the user will be allowed to choose the following options:

**Re-address Existing**

This button will readdress all existing data that is in the imported table space. MW+ will preserve the order of the existing data.

**Readdress Imported**

This button will re-address the imported table. MW+ will find a contiguous area of memory for the table.

**Skip**

Skips importation of this table.

**Overwrite**

Imports the table and removes the existing item(s).

**Cancel**

The import operation is canceled leaving the data area unchanged.

**Variables**

The name, initial value (if present), address, and comment will be imported. The type will be derived from the address.

1. If the name and the full address range is unused the import of that variable will proceed without dialog. The full address range check will look for bits, words, longs, and floats that overlap the imported address.
2. If the imported name and address exactly match a variable in the database, the value and the comment will be imported.

- If the imported name matches an item in the database but the address does not, the user will be shown the following grid with the existing name and address and the importing name and address.

	Name	Address
Imported	MyBit1	MB2001
Existing	MyBit1	MB1001

A message will be displayed stating the conflict, and the user will be allowed to choose the following options:

**Rename Existing**

This button will present the user with a text box where a new name for the existing item can be typed.

**Rename Imported**

This button will present the user with a text box where a new name for the imported variable can be typed.

**Skip**

Skips importation of this variable.

**Overwrite**

Imports the variable and removes the existing item.

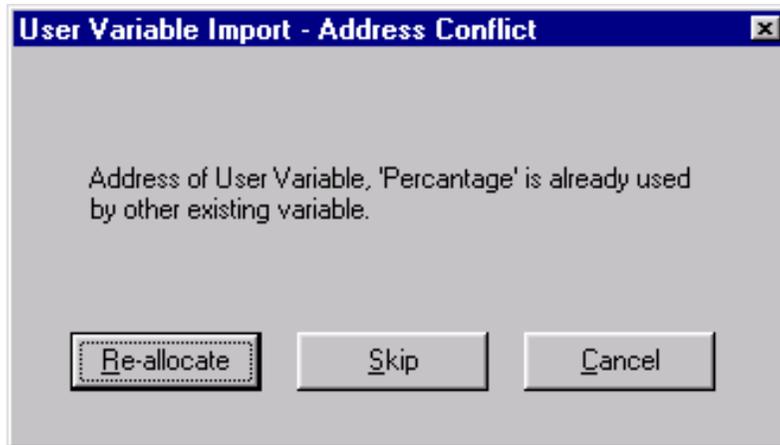
**Cancel**

The entire import operation will be canceled leaving the data area unchanged.

- If the importing address matches any existing item within the address range, the following grid will be displayed with all of the existing name(s) and address(s) and the importing name and address.

	Name	Address
Imported	MWord1	MW100
Existing	MyBit1	MB1001
Existing	MyBit1	MB1002

A message will be displayed stating the conflict, and the user will be allowed to choose the following options:

**Readdress Existing**

This button will readdress all existing data that is in the importing address space. MW+ will preserve the order of the existing variables, so it cannot simply insert the existing variables back into the list.

**Readdress Imported**

This button will readdress the imported variable.

**Skip**

Skips importation of this variable.

**Overwrite**

Imports the variable and removes the existing item.

**Cancel**

The entire import operation will be canceled leaving the data area unchanged.

**I/O**

The name, initial value (if present), address, and comment will be imported. The type will be derived from the address.

**System Variables**

If a register value matches a system variable, the Name will be compared. If the names do not match, a warning is issued to the user that a non-system variable existed in a location reserved by MW+ for system variables.

## Monitoring

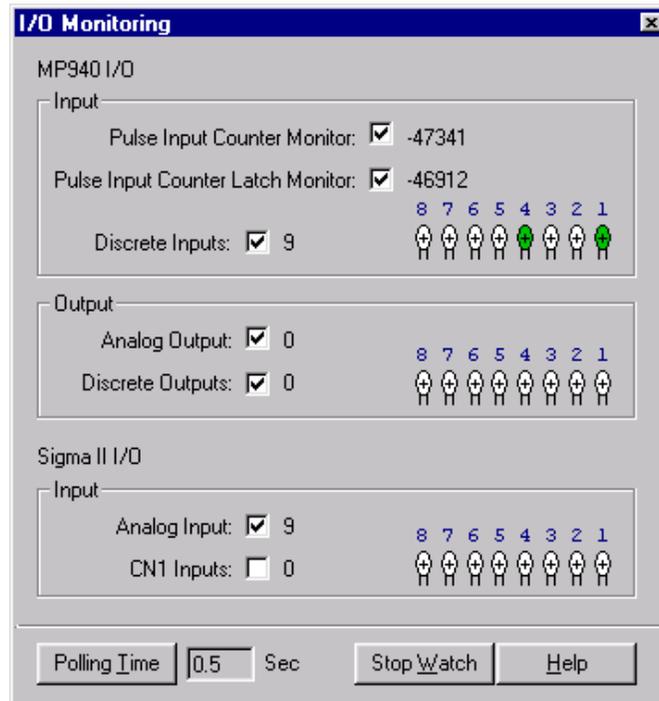
All monitoring functions are available only after going online with the exception of Scope. There are 4 monitoring views available: Data, I/O, and Program and Scope.

### I/O Monitoring

#### Accessibility

To monitor the I/O, select:

- From the Project Explorer > Monitoring > I/O
- From the Main menu > Project > Monitoring > I/O

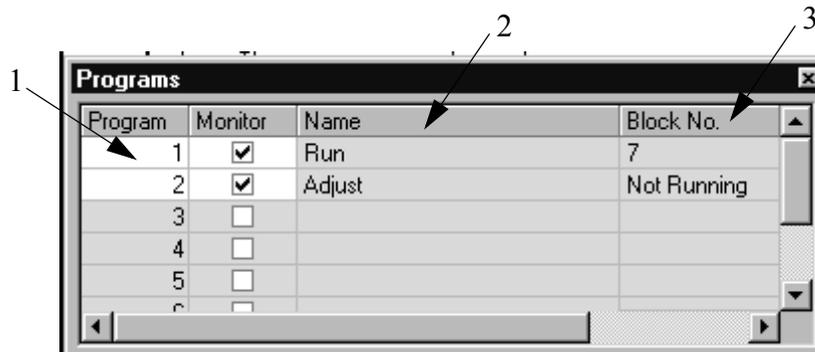


## Program Monitoring

### Accessibility

To monitor the program, select:

- From the Project Explorer > Monitoring > Programs
- From the Main Menu > Project > Monitoring > Programs



### 1. Program

Select the checkbox in this column to enable monitoring for any program.

### 2. Name

This column displays the program or subroutine name currently active.

### 3. Block No.

This column displays the current block being executed and further indicates whether the program is running.

## Monitoring Data

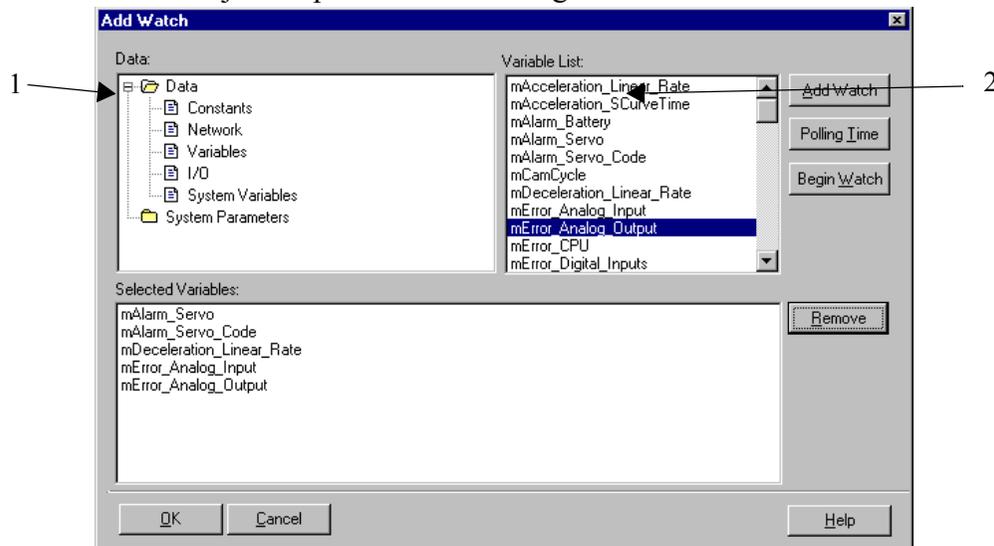
Any data can be monitored while online with the controller via the windows shown below. Read/Write data can be changed on-the-fly using the Watch window. A list of data selected for monitoring is saved so that the same data set can be monitored again during future sessions with the same project.

### Accessibility

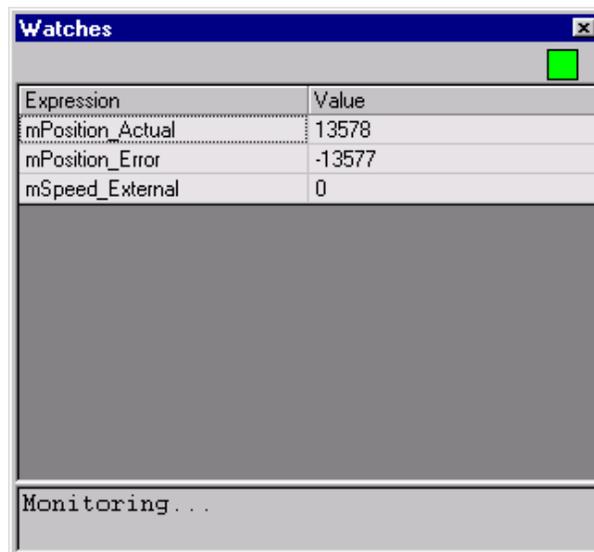
To access the Watch window, select:

- From the Main Menu > Project > Monitoring > Data

- From the Project Explorer > Monitoring > Data



1. Open folders from the Data text box.
2. Double-click on items in the Variable List to add them to the Watch window.



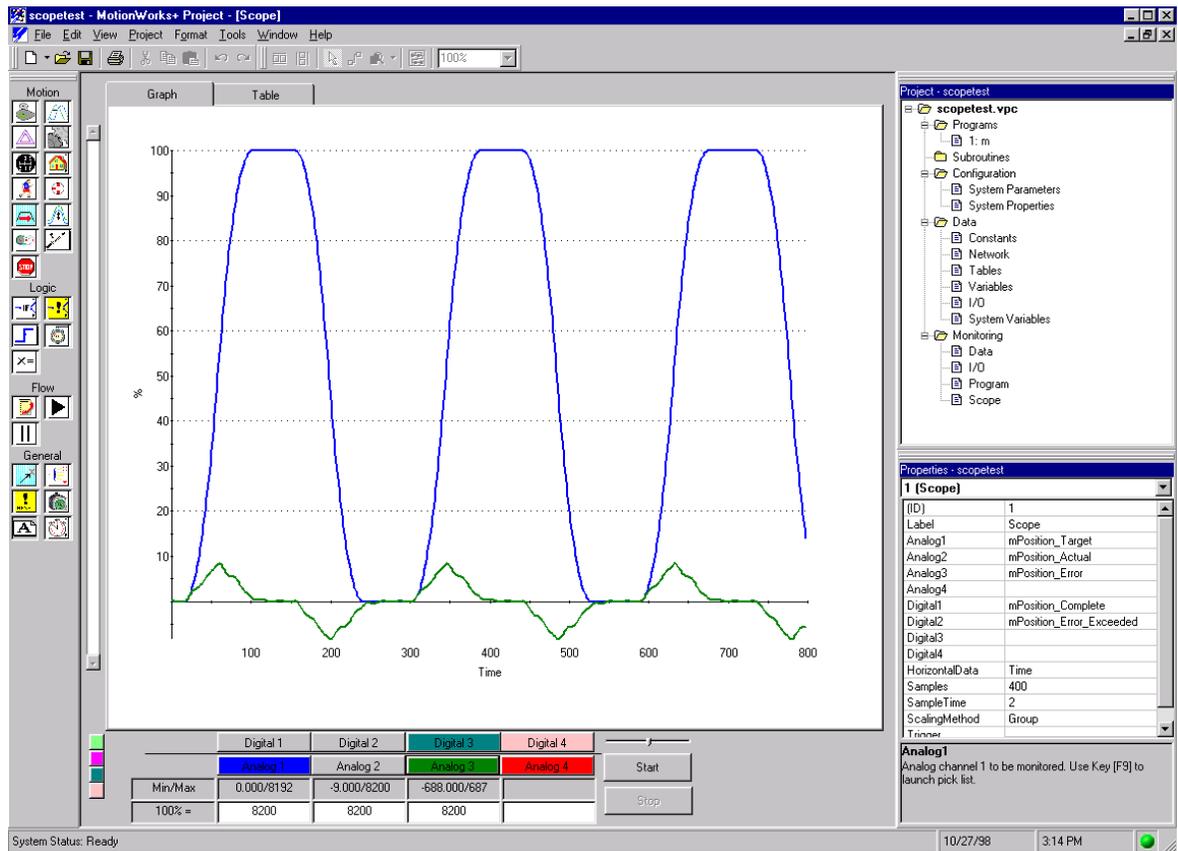
While viewing the watch window, data can be modified by typing a new value in the right column. Variable updates occur as soon as the **Enter** key is pressed or the cell loses focus. While editing data, the Variable Watch window does not update. This is indicated by the status light in the upper right corner of the window.

## 1.7 Scope

### Accessibility

To view the scope display, select

- From the Main Menu > Project > Monitoring > Scope
- From the Project Explorer > Monitoring > Scope



The Scope displays recorded data. Choose data to be recorded and duration in the properties window. The Scope has the capability to display up to eight separate data elements.

There are tabs at the top of the graph that allow the user to switch between viewing data graphically or in a table.

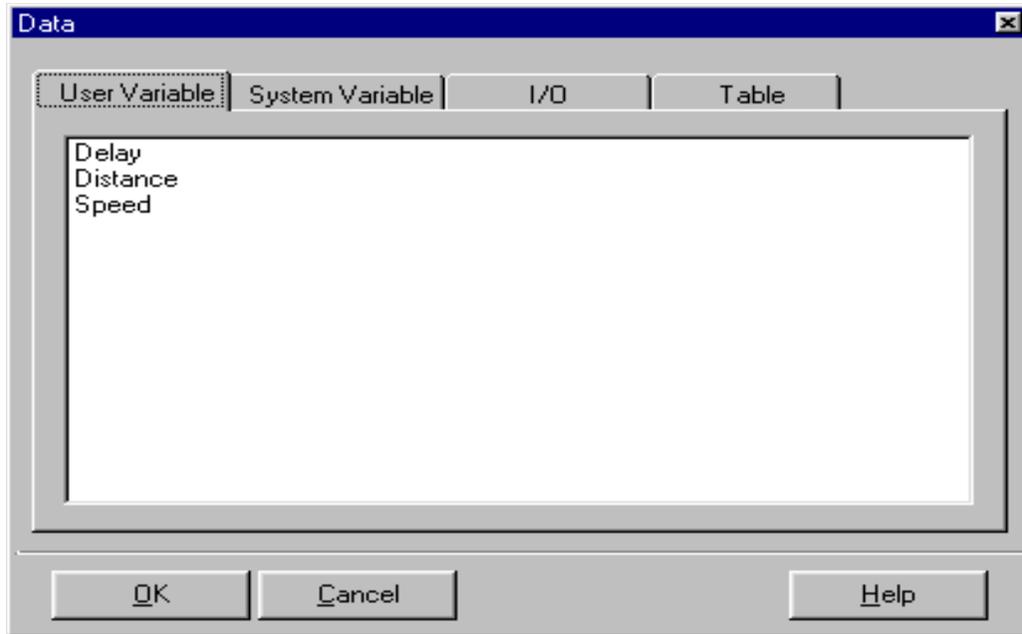
The following are the properties for the Scope. These properties are applied when the user clicks on the start button.

**Properties**

<b>Property</b>	<b>Default</b>	<b>Minimum Value</b>	<b>Maximum Value</b>	<b>Notes</b>
Analog1	Nothing	N/A	N/A	Any Data
Analog2	Nothing	N/A	N/A	Any Data
Analog3	Nothing	N/A	N/A	Any Data
Analog4	Nothing	N/A	N/A	Any Data
Digital1	Nothing	N/A	N/A	Any Bit Data
Digital2	Nothing	N/A	N/A	Any Bit Data
Digital3	Nothing	N/A	N/A	Any Bit Data
Digital4	Nothing	N/A	N/A	Any Bit Data
Horizontal-Data	Time	Analog1	Analog4	N/A
Samples	1000	10	32767	Samples / depend on data types
SampleTime	1	0.5	5000	mSec
Scaling-Method	Custom	Custom	None	Group, Individual, None
Trigger	Nothing	N/A	N/A	Expression builder

**Analog 1~4**

Select the data that will be captured for display on this channel. To use the expression builder, press F9.

**Digital 1~4**

Select the digital data that will be captured for display on this channel. When a digital signal is displayed on the graph, it will occupy 20% of the graph height. To use the expression builder, press F9.

**HorizontalData**

Normally the horizontal time scale is “Time.” It is possible to select any analog channel data as the horizontal time scale. If only one analog channel is defined and no digital channels are defined, the only choice is “Time.”

**Samples**

Specify number of points to be recorded for each channel. The maximum number of storage words is 4096. MW+ will determine if there is enough memory to perform the scope function based on the number of channels selected and notify the user if there is not enough memory.

**SampleTime**

This is the number of milliseconds between samples. The sample time cannot be less than the High scan setting in the MP940 system properties.

**ScalingMethod**

Specify how analog data is displayed on the graph. MW+ will determine the upper and lower bounds of the Y axis and the “100% = ” number for each channel based on the setting of this property. For all cases below, there will always be either a 100% or a -100% boundary, or both.

When this property is set to “group,” each channel will be scaled to the same “100% =” number. Scaling is derived from the channel with the largest absolute min/max value.

When this property is set to “individual”, each channel will be scaled to occupy most of the graph. MW+ will calculate the “100% =” number for each channel to obtain the best possible fit.

When this property is set to “none”, scale values will be used from the previous Scope, and the vertical scale will be:

if all channel data are positive: 100 to 0%

if all channel data are negative: 0 to -100%

else: 100 to -100%.

The vertical axis is always scaled in percent. Each channel can have a different scale such that 100% for analog channel #1 can be different from analog channel #2.

### **Trigger**

The trigger is optional. If the user enters a trigger expression and clicks on start, the Scope will not begin until the expression is true. If no trigger is entered, the Scope will begin immediately after clicking **Start**.

To start **Scoping**, select data for at least one channel and click on the **Start** button. The Scope window will display the following text messages in a text box until data is ready to be displayed:

If a trigger is selected, and triggering has not begun

“Waiting for trigger.”

If recording data then

"Recording data."

If uploading data

"Uploading data"

Notes about Triggering:

This is the basic format of a trigger: [data] [OP] [Constant]

Example of a trigger: mSpeed\_Main >= 200

The following operators may be used:

++
!=
<
>
<>
<=
=>
true
false

If the [Data] is a bit, then the [OP] must be “=”.

If the [Data] is a bit, the the [Constant] must be “ON or “OFF”.

If the [Data] is an integer, then the [Constant] must be an integer.

If the [Data] is a float, then the [Constant] can be integer, float, or in exponential form.

## Analog Channel Controls

To switch the graphed data on and off, click the **Channel Display** buttons. The Channel Display buttons are the same color as the graph line. When the mouse is held over the Channel Display button, a tool tip will display the data name.

The minimum and maximum data value is displayed for each channel.

Each channel indicates 100% value in data units. If “Scaling Method” is “none,” the previous value is retained. If it is “group” or “individual,” the value is calculated by MW+ to maximize the information on the screen. Adjust this value to customize the view.

Use the horizontal slider to determine the current value of each channel at any point along the graph. When the mouse button is held down over the horizontal slider, the “**Min/Max**” field label will change to “**Current value.**” A vertical dashed line appears on the graph at the slider location, and the current value is displayed in the Current Value field. When the button is released, the “Current Value” field will revert to “Min/Max.”

## Digital Channel Controls

To switch digital channels ON or OFF, use the “**Digital Enable**” buttons at the bottom of the Scope window. The channel display buttons are the same color as the digital graph line. When the mouse is held over the channel display button, the data name is displayed.

A vertical slider controls the vertical placement of the digital channels. Only one of the four vertical slider control buttons can be active at any time. The four buttons are color coded with the digital data colors. When the vertical slider control button for digital channel one is active, the scroll bar indicator will be located at the zero level for digital channel one data. It is possible to change the zero level for each digital channel by moving the slider. This can be used to separate digital channels for easier viewing.

## Graph Controls

It is possible to zoom on any portion of the graph. Enter a specific zoom percentage, Fit in Window, or zoom on a specific range (window). To zoom into a specific range on the graph, depress the left mouse button and drag over the graph until the desired range is highlighted. When the button is released, the graph will display the zoomed range.

When the entire graph does not fit in the window, vertical and/or horizontal scroll bars will appear, making it possible to access the non-displayed areas of the graph.

It is possible to display data in a graph or table by selecting the graph or table tab.

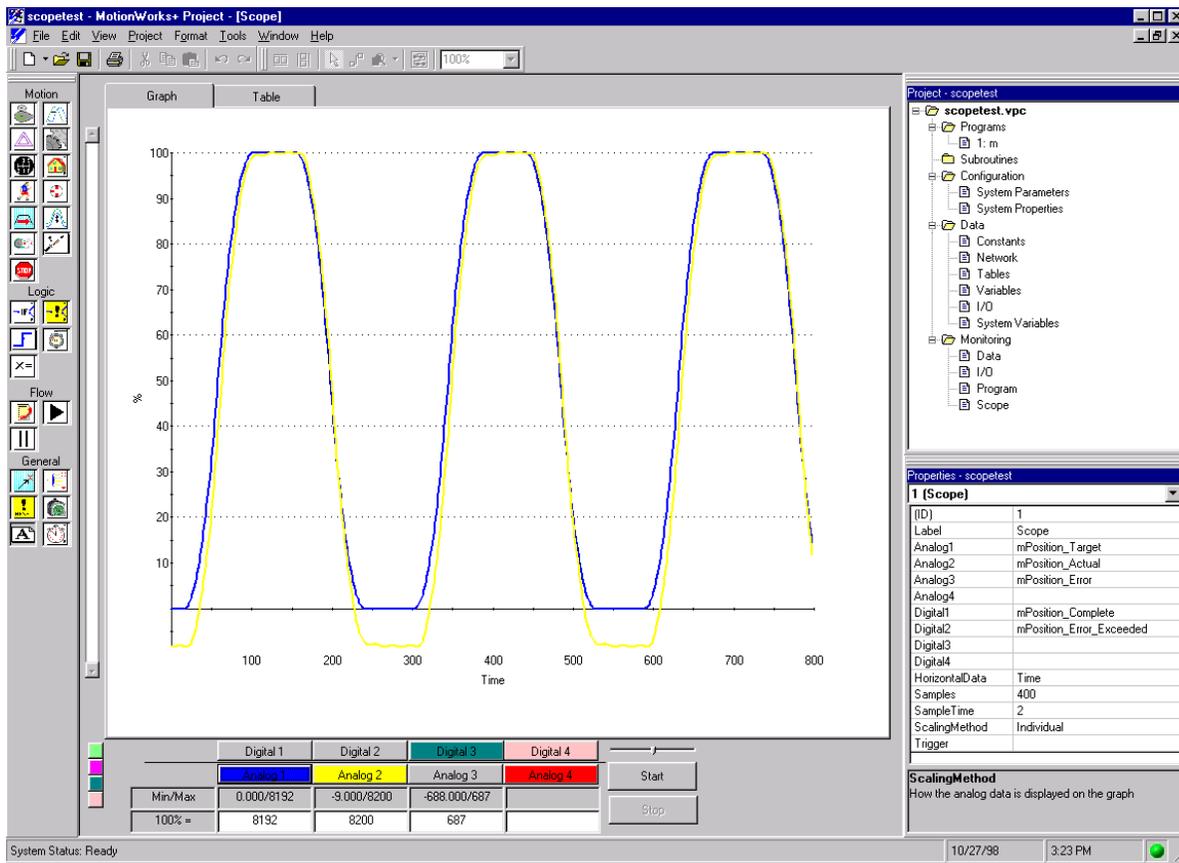
It is possible to import, export, or save recorded data by selecting items from the right click menu while the mouse is over the graph or chart.

## Scope Examples

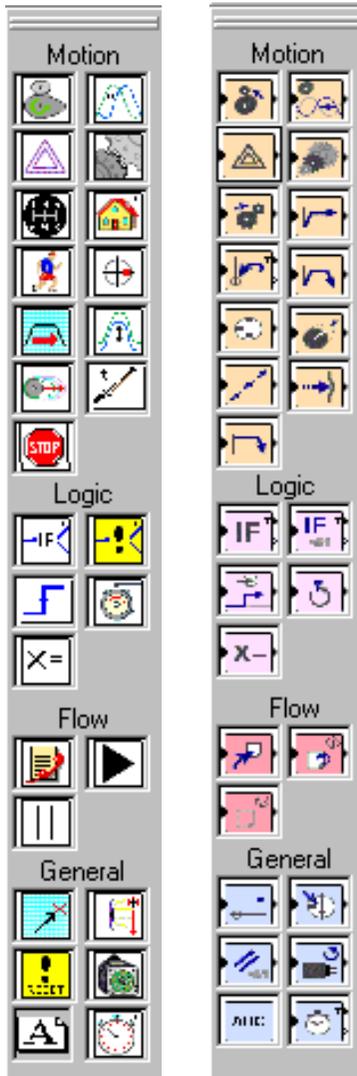
There are two colored graph lines shown in the example below:

The darker graph line indicates Analog 1.

The lighter graph line indicates Analog 2.



## 1.8 The Block Toolbar



The block toolbar contains all the programming icons. The toolbar can either float or dock to the left or top of the screen. The blocks are visually grouped by a dividing line on the toolbar. There are two icon sets available. The one shown on the left is the YEA standard. The one on the right is the YEC standard.

### Accessibility (Tool bar usage)

- From the Main Menu > View > Toolbars > Block

### Accessibility (Graphic selection)

- From the Main Menu > Tools > Options > Block Graphics

## 1.9 The Properties Window

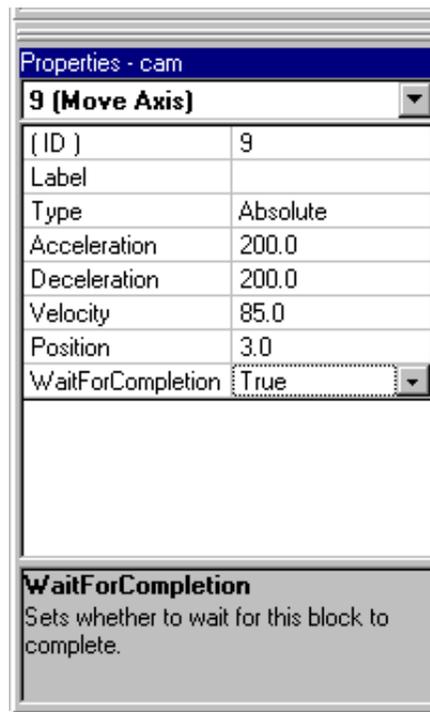
All the details for each block are entered and edited in the Properties window. This window can be dragged, resized, docked, and undocked. Properties change based on the item/block selected. The Properties window has a Help text box that describes each property. If no icon is selected, the properties are shown for the current program or subroutine.

### Accessibility

To access the Properties window, select:

- From the Main Menu > View > Properties
- From any block or configuration item > right-click > Properties
- If the Properties window is closed, double-click on an icon to re-display

The Properties window can be closed, but a right click on an icon allows this window to re-appear. In many cases, the **Expression Builder** is available when double-clicking on a property. This allows selection of data and the creation of formulas from organized lists.



## 1.10 The Program Window

The Program window is the location at which all the files in the project are displayed when opened. There are two types of program windows that appear. The first type is either a program or subroutine. The second type is a form that contains data from monitoring, debug, parameter setup, etc.

### Program Window Behavior

- Program Positioning

The programming area is generally referred to as the “canvas.” Canvas positioning is virtually unlimited, as the scroll bars allow for infinite scrolling. The **Home** key puts the START block at the center of the program window. The **End** key puts the END block at the center of the program window. The zoom range of the program window is between 10% ~ 400%.

- Block Placement

The program window has a snap to grid feature that allows blocks to be placed uniformly. Snap to grid is the default setting, and is global for all programs and subroutines in the project. The default grid size is equivalent to one block. The minimum grid size is ½ block. The maximum grid size is 5 times one block. Blocks cannot be placed one on top of the other.

- Block Editing

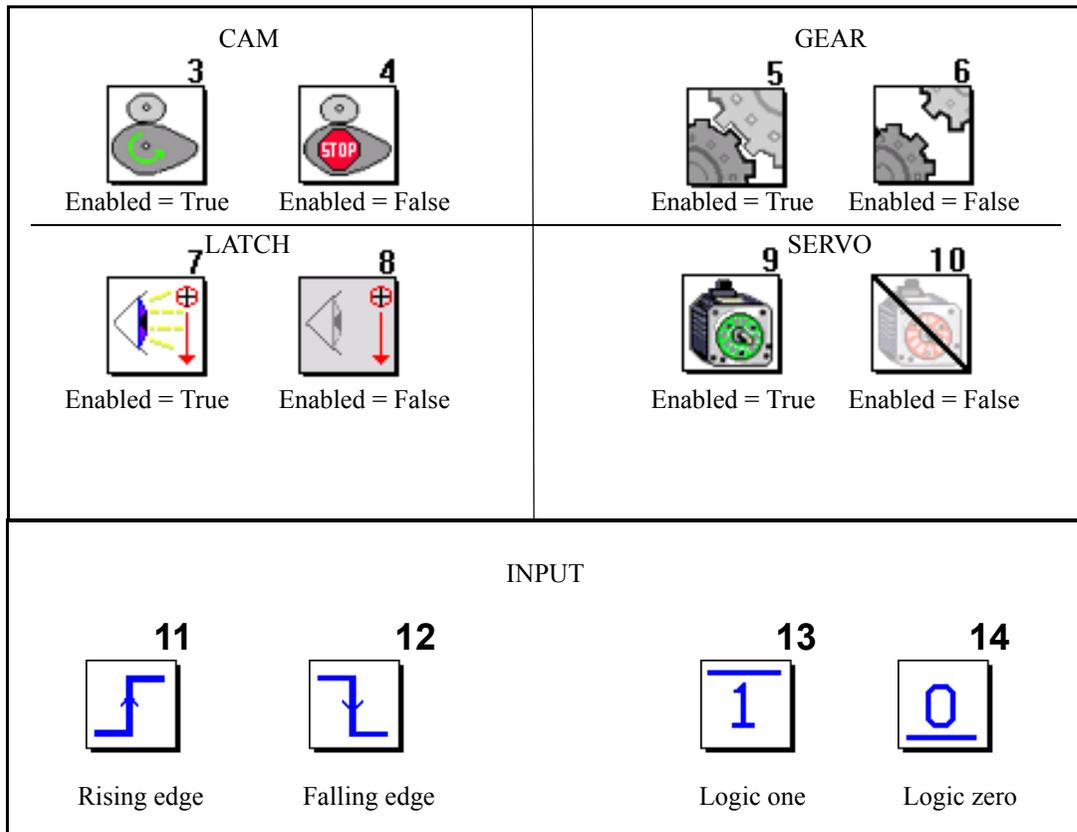
The Cut/Paste/Copy functions conform to Windows standards. The selected block has a highlighted background. To select multiple blocks, hold the **Ctrl** key down while clicking on additional blocks. This allows repositioning of a group of blocks.

- Block Connections

Connect blocks by first selecting the connection mode function from the programming tool bar. The mouse pointer changes to a diagonal line. Click and hold the left mouse button down while near the output port of a block. Move the cursor near the input port of another block to extend the line, then release the mouse button. The connection lines automatically position themselves horizontally or vertically on the canvas. Arrows show the program flow from one block to another. Lines conform to the Windows standards for editing, i.e., when selected, they are highlighted and handles appear for resizing and moving. When blocks with connections are repositioned, only the connection line from the moved block to the first handle repositions itself. Reposition the entire connection path manually if necessary.

- Block Appearance

The bitmaps of some blocks change depending on the property settings for the block.

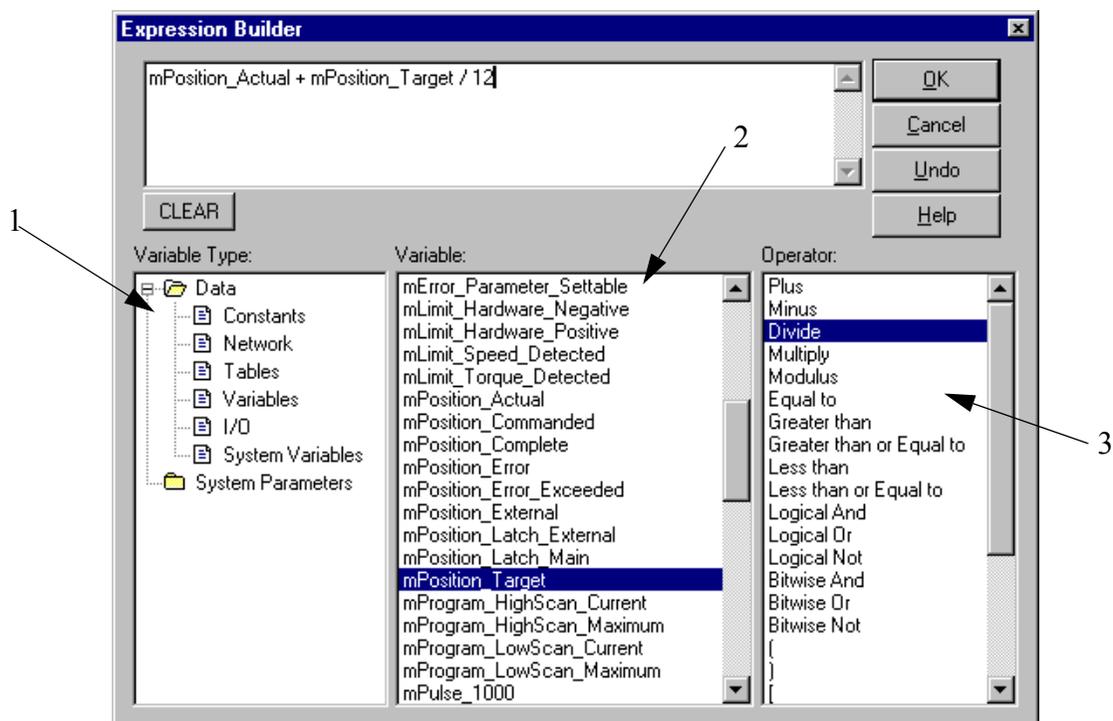


## 1.11 Expression Builder

The Expression Builder is useful when generating complex calculations. The Expression Builder is available when editing properties that can be updated at run time. For example, the properties of the SET VARIABLE, MOVE AXIS, or CHANGE DYNAMICS blocks are designed to be updated as the calculation variables change. Additionally, logical expressions can be created for blocks such as IF EVENT, and IF FAULT.

### Accessibility

- Double-click on any block property that accepts an expression.



### 1. Variable Type

Select Constants, Network, Tables, Variables, I/O, or System.

### 2. Variable

Shows the list of data from the selected folder in the *Variable Type* window.

### 3. Operator

Shows all the available math functions. Select from the list by double-clicking; the selected operator is then inserted at the cursor location in the expression.

## Creating Expressions

Double-click on the folders in the **Variable Type** window. Select an item from the Variable window. The item is inserted at the cursor position.

For the purpose of explanation of the operators, assume the following:.

Variable Name	Type	Value
mPositon_Complete	Bit	1
mSpeed_Main	Float	-36.274
mState_Latch_Main	Bit	1
Mytable[Var2]	Long	10000
Var1	Integer	100
Var2	Integer	10
Var3	Integer	75
Var4	Integer	10
Var5	Bit	1
Var6	Bit	0
Var7	Integer	85 (0x55)
Var8	Integer	170 (0xAA)
Var9	Float	0.1666667

Operator	Symbol	Example Expression	Result
Plus	+	Var1 + Var2	110
Minus	-	Var1 - Var2	90
Multiply	*	Var1 x Var2	1000
Divide	/	Var1 / Var2	10
Modulus	%	Var3 % 4	3
Equal To	==	mPosition_Complete==Var5	1
Greater Than	>	Var1 > Var2	1
Greater Than or Equal to	>=	Var2 >= Var3	0
Less Than	<	Var1 < Var3	0
Less Than or Equal to	<=	Var2 <= Var4	1
Logical And	&&	Var5&&mPosition_Complete	1
Logical Or		Var5  Var6	1
Logical Not	!=	mState_Main_Latch != Var6	1
Bitwise And	&	Var7 & Var8	0
Bitwise Or		Var7   Var8	255 (0xFF)
Bitwise Not	!	—	—
( )	( )	(Var1+Var2) / Var3	11

Operator	Symbol	Example Expression	Result
[ ]	[	MyTable[Var2]	10000
sin	sin	sin(Var9*180)*4096.0	2048
cos	cos	cos(0)	1
tan	tan	tan(45)	1
arctan	acrtan	artan(1)	45
Square Root	sqrt	sqrt(9)	3
Absolute Value	abs	abs(mSpeed_Main)	36.274
True	true	Var5==true	1
False	false	Var5==false	0

Note: If a floating point result is expected, at least one of the terms in the calculation must contain a floating point value. This is true even if the terms are all hard coded.

Example:

ACCEL = 1000 (integer)

VELOCITY = 180 (integer)

Calculation:

Answer = sqrt (ACCEL/VELOCITY)

The above calculation does not produce accurate results because none of the terms are floating points. The answer should be 2.35702, but is reported as 406.

One solution is to multiply the expression by 1.0. Another solution appears on the following page.

Example:

ACCEL = 100.0 (floating point)

VELOCITY = 180 (integer)

Calculations:

Answer = sqrt (ACCEL/VELOCITY) or

Answer = sqrt (ACCEL/180.0) will produce the correct answer: 2.35702.

## More Example Expressions

Expression	Description
JogFwd    JogRev	Assumes both variables are bit type. This is a boolean expression that evaluates to true if JogFwd is true OR JogRev is true.
Start && !Fault	Assumes both variables are bit type. This is a boolean expression that evaluates to true only if Start is true AND Fault is NOT true.
Positions[(Local_Input_Bank & 30)/2]	Assumes Positions is a table and the calculation inside brackets designates pointers for the index in the table. It looks at bits 1, 2, 3, and 4 and masks the rest of the byte by ANDing it with 30 decimal (\$1E hex). After masking, the data is shifted over by dividing by 2. The expression results in a value between 0 and 15.

## 1.12 Cross Reference

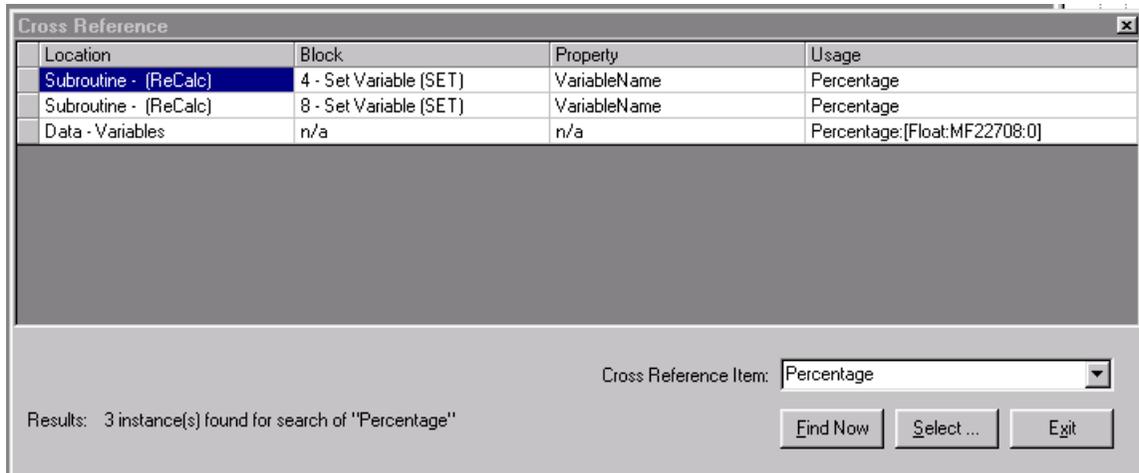
### Accessibility

To view the cross-references, select:

- From the Main Menu > View > Cross Reference

The Cross Reference feature allows searching for all occurrences of a variable, phrase, word, or expression. Double clicking on any cell in the row will display the instance. The drop down box shown on the form will allow the user to search for items in 3 ways:

1. Type the phrase, name, or keyword to search.
2. Select from items in the dropdown list that contains the last 10 searches. This information is stored with each project.
3. Click on the **Select** button and pick an item from the expression builder.



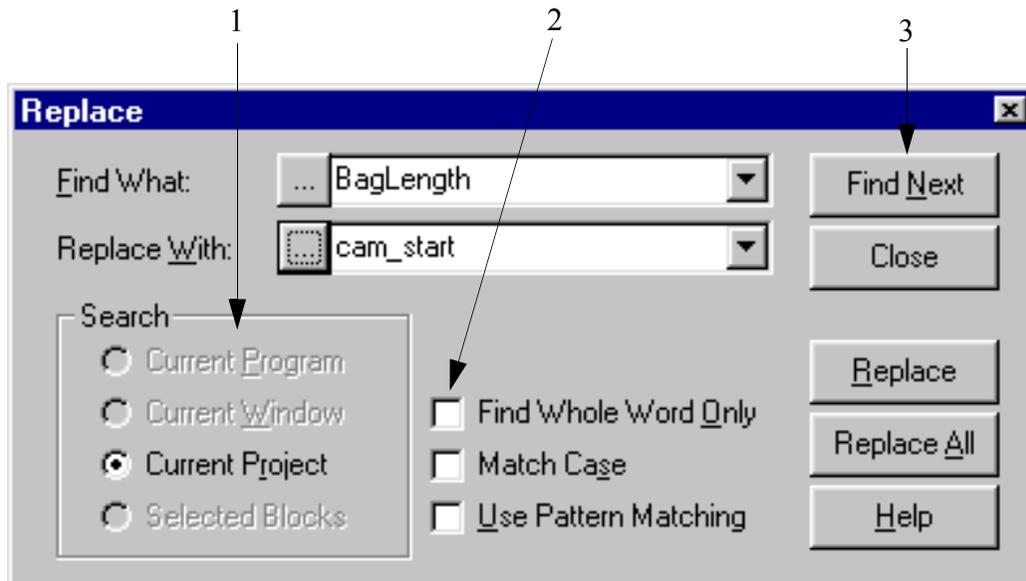
## 1.13 Search and Replace

### Accessibility

To access the search/replace dialog, select:

- From the Main Menu > Edit > Replace

The Search and Replace feature allows for locating words or terms that are already in the program and replacing them with new terminology. Specific functions, i.e.; the Search box (1), check boxes (2), and information tabs (3) make it easier to accomplish this task.



## 1.14 Connecting To The Controller

### Online/Offline

Communication is established between MotionWorks+™ and the MP940 by going online. The status bar on the lower left corner of the screen displays a message indicating the connection status. Also, there is a green or black dot on the lower right portion of the screen to indicate the connection status.

### Accessibility

To go online, select:

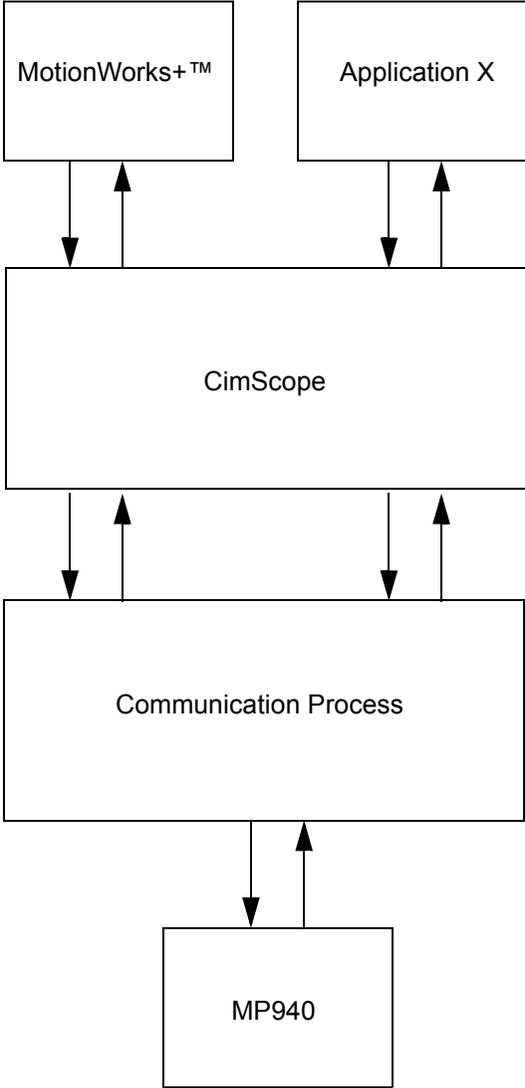
- From the Main Menu > Tools > Online
- Hot Key: F6

### Communication Settings

This section explains how to set communication between MotionWorks+™ and the controller. When the installation is complete, the communication settings are set to the default values, as follows: 19.2kbps; data bits: 8bit; parity: even; stop bits: 1Stop bit.

### Launching the Communication Process

The **Communication Process** is a separate application that launches when attempting to go online with the controller. It is a program that coordinates communication between applications such as MotionWorks+™ and MP-series controllers. The Communication Process is visible in the Task Manager. Double-click on it to view and edit the settings

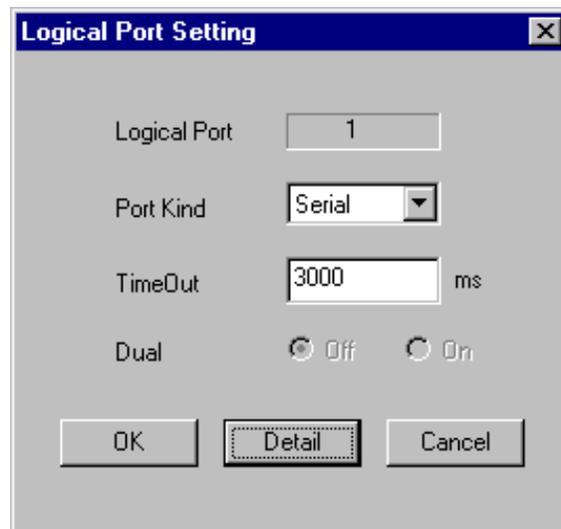


## Setting the Communication Port

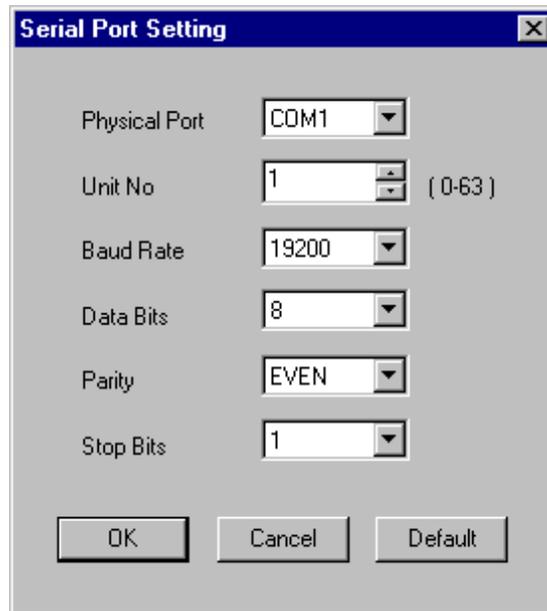
The main screen of the communication process is shown below. Either double-click the **logical port number**, or select a logical port number in the **Logical PT** column, then select **Setting (E)** from the **File** menu.



The Logical Port Setting window is displayed, as in the figure below. Select the **port** type in the **Port Kind** box, then select **Detail**.

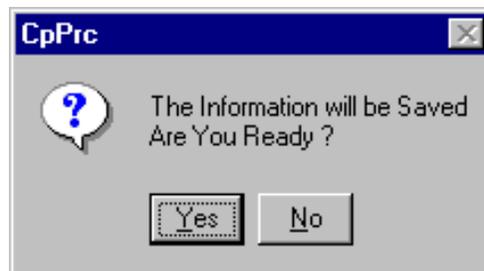


When Detail is selected, the following window is displayed. Set the parameters of the serial port.



### Saving Communication Port Setting Values

Select **File > Save**, and a confirmation message window appears. Choose the **Yes** button to save the information.



The Communication Process provides a link between MotionWorks+™ and the controller. Do not close this program; minimize the Communication Process when the settings are complete.

## 1.15 Compiling A Program

Compiling a program converts the block connections and properties into a format usable by the motion controller.

### Accessibility

To compile a program, select:

- From the Main Menu > Tools > Compile
- Hot Key: F5

### Compiling Errors

If a compiling error has occurred, a window appears with a description of all errors found. Double-click on the error to go to the property. The message box identifies the following information:

1. Program name
2. Block number
3. Property
4. Error description

## 1.16 Downloading a Project

### Accessibility

To download a project, select:

- From the Main Menu > Tools > Download
- Hot Key: Shift+F5

MotionWorks+™ must be online with a controller before downloading. If the controller is running, a message box appears that requests permission to stop the controller prior to downloading. The controller must be stopped for a download to occur. This halts program execution and disables the servo.

The download tool will automatically transfer only System Property modules and programs changed since the last download. Full Compile & Download will force MW+ to download all System Property modules and programs.

After downloading, a message box appears requesting permission to restart the controller. The project is now residing in SRAM, the battery backed memory. This memory is retained by a super capacitor during power OFF for several hours without a battery. If no battery is present, the project must be saved in flash memory for permanent storage. See “Saving a Project to Flash” on the next page.

## 1.17 Saving a Project to Flash

### Accessibility

To save a project to the flash memory, select:

- From the Main Menu > Tools > Controller > Flash Memory

This operation copies the *complete* project from the controller's SRAM to the flash memory, *not* from the computer to flash directly. This means that downloading the project from the computer to the controller's SRAM, as mentioned in the "Downloading a Project" section above, must be performed first.

**IMPORTANT NOTE:** If changes are made in the systems properties and a right click "send to controller" was performed while on the MP940 or External Encoder module, that system property data will not be copied to Flash because "send to controller" is a temporary (RAM only) setting which is overwritten by the application program at power up. Be sure the project has been officially downloaded by the "compile & download" function first.

It is recommended to use both the FLASH and COPY DIP switches when using the Flash mode because the COPY DIP switch copies all register (system variable) data from Flash at power up. Unless using a battery (for data storage), the COPY switch should be used in conjunction with the FLASH switch.

The position of the FLASH DIP switch does not matter when saving the program to FLASH. The DIP switch must be set, however, for the program to be copied from FLASH to RAM at power up.

## 1.18 Electronic Cam Tool

The Electronic Cam Tool is an external application that accompanies MotionWorks+™, and is automatically loaded during installation. This application is specifically for the creation of cam profiles, ranging from simple to complicated, based on user requirements. The profiles created are incorporated into MotionWorks+™ as tables. When a table is created, the data can be used in the *Position Data Table* property of the CAM block.

### Accessibility

To access the electronic cam tool, select:

- From the Main Menu > Tools > Cam Tool

### Getting Started

Select File > New from the Cam Tools main menu when the cam tool program starts.

### Entering Machine Information

In the Set Style window, enter the information necessary for preparing the cam profile data. Set the units of measure for the master (phase) and the slave (position). Machine and motor information is optional, but recommended for higher performance.

The 'Set Style' dialog box is divided into two main sections: '[ Phase/Position Setting ]' and '[ Machine/Motor Information ]'. A 'MEMO' box on the right provides instructions for items marked with an asterisk [\*].

**[ Phase/Position Setting ]**

- 1: Unit(Phase) - Radio buttons for Degree (selected), Pulse, and No Unit.
- 2: Unit(Position) - Radio buttons for mm (selected), Pulse, and No Unit.
- 3: Max Phase Value from the Bottom Dead Center (Where the bottom dead center = 0.) - Input field: 360.00, Unit: Degree.
- 4: Max. Position Value from the Bottom Dead Center (Where the bottom dead center = 0.) - Input field: 0001000.0000000, Unit: mm.

**[ Machine/Motor Information ]**

- 5: Ball Screw Lead - Radio buttons for Not Provided and Provided (selected). Input field: 5.0, Unit: mm.
- 6: Gear Ratio - Radio buttons for Not Provided (selected) and Provided. Input field: 1 / 1.
- 7: Required Time for One Cycle(The shortest time) - Input field: 60.0000, Unit: s.
- 8: Motor Rated Speed [\*] - Input field: 3000, Unit: r/min.
- 9: PG Pulse Number after Multiplication - Input field: 8192, Unit: p/r.
- 10: Rated Torque [\*] - Input field: 0.0000000, Unit: kgf.m.
- 11: Instantaneous Peak Torque [\*] - Input field: 0.0000000, Unit: kgf.m.
- 12: Motor Inertia [\*] - Input field: 0.0000000, Unit: kg.m2.
- 13: Gear+Coupling Inertia [\*] - Input field: 0.0000000, Unit: kg.m2.
- 14: Load Torque(Motor axis conversion) [\*] - Input field: 0.0000000, Unit: kgf.m.
- 15: Load Inertia(Motor axis conversion) [\*] - Input field: 0.0000000, Unit: kg.m2.

**[ MEMO ]**

Input data to items with [\*]. The following Informations are displayed when data is edited.

1. Effective torque as a percent of motor rated torque.
2. Peak torque as a percent of rated torque.
3. Max. speed

Buttons: OK, Cancel

**Properties - twocam**

6 (External Encoder)

(ID)	6
Label	ExternalEncoder
Enabled	True
FeedConstant	199.4911
GearBoxInput	5
GearBoxOutput	1
MachineCycle	554
MovementType	Rotary
PulseType	Quadrature
Resolution	8192
UserUnits	mm

6 (External Encoder)  
Properties setting for - 6 (External Encoder)

**Properties - twocam**

2 (MP940)

(ID)	2
Label	MP940
BatteryTest	Disabled
EncoderResolution	8192
EncoderType	Incremental
FeedConstant	360
Firmware	N/A
GearBoxInput	5
GearBoxOutput	1
HighScanSetting	1
LoadType	Rotary
LowScanSetting	20
MachineCycle	360
MotorRatedSpeed	3000
UserUnits	degrees

**UserUnits**  
Select cm, mm, inches, pulses, degrees, or a user defined type.

**Set Style**

[ Phase/Position Setting ]

Unit(Phase)  Degree  Pulse  No Unit  
 Unit(Position)  mm  Pulse  No Unit

Max Phase Value from the Bottom Dead Center (Where the bottom dead center = 0.)    
 Max. Position Value from the Bottom Dead Center (Where the bottom dead center = 0.)

[ Machine/Motor Information ]

Ball Screw Lead  Not Provided  Provided  mm  
 Gear Ratio  Not Provided  Provided  
 Ball Screw Axis/Motor Axis  /

Required Time for One Cycle(The shortest time)  s  
 Motor Rated Speed [\*]  r/min  
 PG Pulse Number after Multiplication  p/r  
 Rated Torque [\*]  kgf.m  
 Instantaneous Peak Torque [\*]  kgf.m  
 Motor Inertia [\*]  kg.m2  
 Gear+Coupling Inertia [\*]  kg.m2  
 Load Torque(Motor axis conversion) [\*]  kgf.m  
 Load Inertia(Motor axis conversion) [\*]  kg.m2

[ MEMO ]  
 Input data to items with [\*].  
 The following Informations are displayed when data is edited.  
 1. Effective torque as a percent of motor rated torque.  
 2. Peak torque as a percent of rated torque.  
 3. Max. speed

OK Cancel

## Phase/Position Setting

### 1. Unit (Phase)

The master position is generally referred to as the “phase” throughout the Cam Tool. This is typically the external encoder. Select a radio button for degrees, pulses, or no units.

### 2. Unit (Position)

The slave position is generally referred to as “position” throughout the Cam Tool. This is the SGDH servo motor. Select a radio button for mm, pulses, or no units.

### 3. Maximum Phase Value from the Bottom Dead Center

This is the cycle of the master. Enter the maximum phase value of the cam curve data. The default setting is 360.0 degrees.

Units	Minimum	Maximum
Degrees	1.00	360.00
Pulses	1	10000000
No units	1.0000	1000000.0000

### 4. Maximum Position Value from Bottom Dead Center

This is the maximum travel of the slave. Input the maximum position value of the cam curve data. The default setting is 1000.0000000mm.

Units	Minimum	Maximum
mm	1.0000000	1000000.0000000
Pulses	1	10000000
No units	1.0000000	1000000.0000000

## Machine/Motor Information

The remaining information is only required when using the *Speed Data Table* in the CAM block. This information aids in determining the acceleration feed forward value and changes this tuning parameter as the slave follows the master.

### 1. Ball Screw Lead

Select the radio button for **Provided** or **Not Provided**. When **Provided** is selected, the setting range is 0.0 to 99999.9. When **Not Provided** is selected, the value is automatically set to 0. This is the pitch of the ball screw, or the distance the slide travels per rotation.

## 2. Gear Ratio

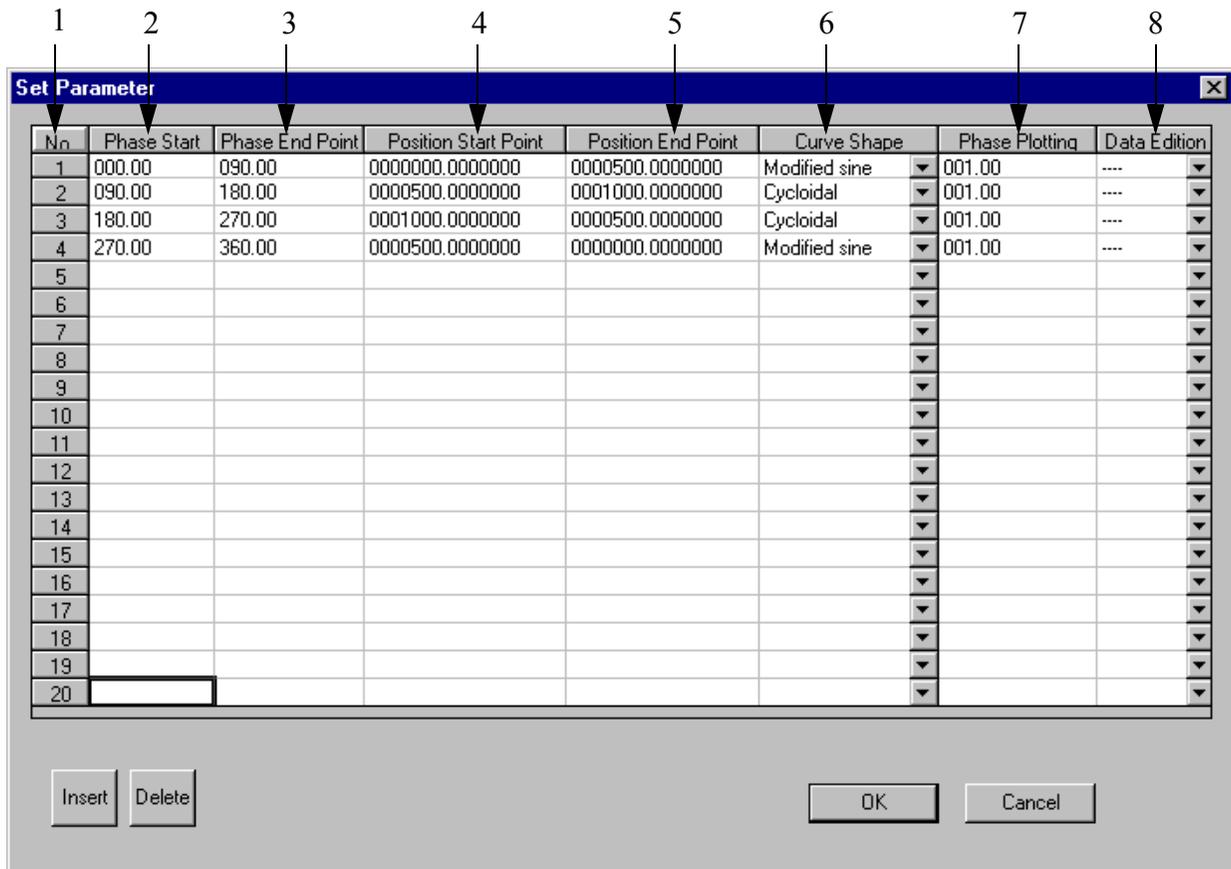
Select the radio button for **Provided** or **Not Provided**. The gear ratio is expressed as ball screw shaft/motor shaft. Enter 1/1 when the gear ratio is not set. The default setting is 1.

Type	Minimum	Maximum
Ball Screw Shaft	1	9999
Motor Shaft	1	9999

Item	Item Name	Minimum	Maximum	Default	Units
7	Required Time for One Cycle (shortest time)	0.0001	9999.9999	60	s
8	Motor Rated Speed	1	99999	3000	rpm
9	PG Pulse Number after Multiplication	1	99999	2048	p/r
10	Rated Torque ( $T_R$ )	0	99999.9999999	0	kgf.m
11	Instantaneous Peak Torque ( $T_{MP}$ )	0	99999.9999999	0	kgf.m
12	Motor Inertia	0	99999.9999999	0	kg.m <sup>2</sup>
13	Gear + Coupling Inertia	0	99999.9999999	0	kg.m <sup>2</sup>
14	Load Torque (motor shaft conversion)	0	99999.9999999	0	kgf.m
15	Load Inertia (motor shaft conversion)	0	99999.9999999	0	kg.m <sup>2</sup>

### Parameter Setting

This window defines the cam profile data. The cam profile can be divided into a maximum of 20 sections. Note: Closing this window without pressing the **OK** button discards all changes.



The table below details the items in the Set Parameter window.

Item	Name	Description
1	Number	Cam data is displayed in sections. Up to 20 sections can be defined.
2	Phase Start	The phase start value of each block is indicated. This column is read only. The values are determined by data entered in the Phase End Point column.
3	Phase End	Set the phase end value of each section of the cam master.
4	Position Start Point	The position start value of each section is indicated. This column is read only. The values are determined by data entered in the Position End Point column.
5	Position End Point	Set the Position End Point value of each section of the slave.

Item	Name	Description
6	Curve Shape	<p>Select the cam curve shape from among the following 21 available shapes. Data points are calculated along each section at the resolution specified in the Phase Plotting column; the units used are those of the master.</p> <ol style="list-style-type: none"> <li>1. Straight line</li> <li>2. Parabolic</li> <li>3. Simple harmonic</li> <li>4. Cycloidal</li> <li>5. Modified trapezoid</li> <li>6. Modified sine</li> <li>7. Modified constant velocity</li> <li>8. Asymmetrical cycloidal</li> <li>9. Asymmetrical modified trapezoid</li> <li>10. Trapecloid</li> <li>11. One-dwell cycloidal m=1</li> <li>12. One-dwell cycloidal m=2/3</li> <li>13. One-dwell trapezoid m=1</li> <li>14. One-dwell trapezoid</li> <li>15. One-dwell trapezoid m=2/3</li> <li>16. One-dwell modified sine</li> <li>17. One-dwell trapecloid</li> <li>18. No-dwell simple harmonic</li> <li>19. No-dwell modified trapezoid</li> <li>20. No-dwell modified constant velocity</li> <li>21. NC2 curve</li> <li>22. Target Matching</li> </ol> <p>Display the combo box and click the desired shape.</p>
7	Phase Plotting	<p>Set the resolution of the specified section. The Cam Tool generates points along the specified curve shape at the interval of the master, in user units. For example, if a certain section of the cam ranges between 0 ~ 90° and a phase plotting value is entered as 0.1, then points will be generated every 0.1° and a total of 900 points will be generated for this section of the cam.</p>
8	Data (Provided/Not Provided)	<p>For a new parameter, “----” is displayed. “Provided” or “Not Provided” is displayed depending on the necessity of the graph data editing. This is useful when switching from the originally entered data and modified cam profiles.</p>

When the cam data has been entered, press **OK** to view the graphs.

## Viewing Cam Data

To change the data of existing files, open the **Data Graph** window and select **Graph** in the menu bar.

1. Set Parameter
2. Set Style
3. Graph Data Editing
4. Shift Phase Direction

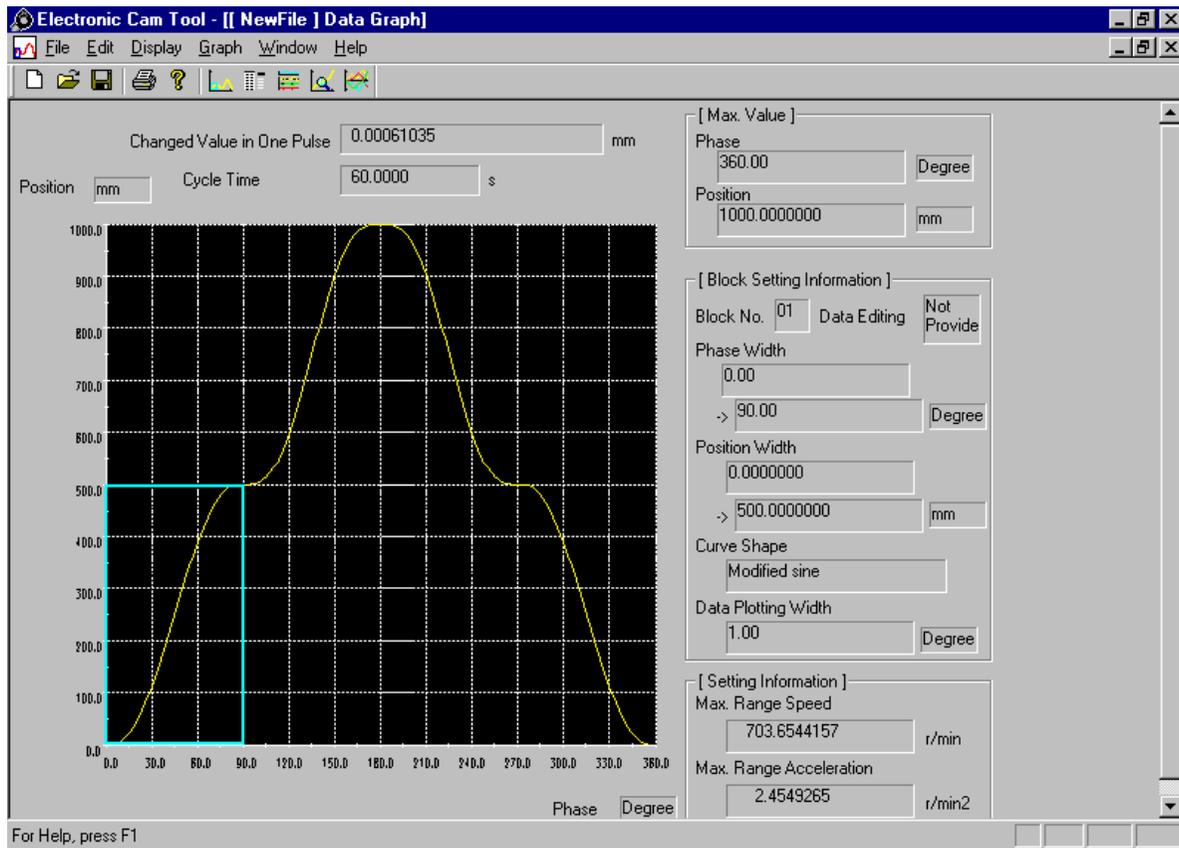
## The Data Graph

### Accessibility

To display the data graph, select:

- From the Main Menu > Display > Data Graph

This window shows the resulting cam profile from the information entered in the set parameter window in graphical format. Use the arrow keys or the mouse to move through the sections of the profile.



## Data List

The numeric data of the cam curve is displayed by section.

1

No.	Phase	Position
1	000.00	0000000.0000000
2	001.00	0000000.0079331
3	002.00	0000000.0632793
4	003.00	0000000.2125291
5	004.00	0000000.5003450
6	005.00	0000000.9686926
7	006.00	0000001.6560236
8	007.00	0000002.5965275
9	008.00	0000003.8194660
10	009.00	0000005.3486037
11	010.00	0000007.2017452
12	011.00	0000009.3903888
13	012.00	0000011.9195812
14	013.00	0000014.7897353
15	014.00	0000017.9999285
16	015.00	0000021.5485012
17	016.00	0000025.4330611
18	017.00	0000029.6504879
19	018.00	0000034.1969405
20	019.00	0000039.0678652
21	020.00	0000044.2580057
22	021.00	0000049.7614143
23	022.00	0000055.5714646
24	023.00	0000061.6808663

2

3

4

For Help, press F1

### 1. List data

The numeric data of the cam curve of the specified section is displayed in each phase division width.

### 2. Display Block No. Designation (Display Section)

Select the section to be displayed.

### 3. Display Block Information (Section Information)

The data quantity and ranges of the phase and position values of the designated section are displayed.

When editing an insertion and/or deletion is performed, the data quantity changes.

#### 4. Setting Information

The effective torque, peak torque, and maximum rotation speed are automatically calculated if the Machine/Motor Information data are set on the Set Style window.

These values are calculated based on the edited data and are then displayed.

The effective torque and peak torque are displayed with the rated torque as 100%.

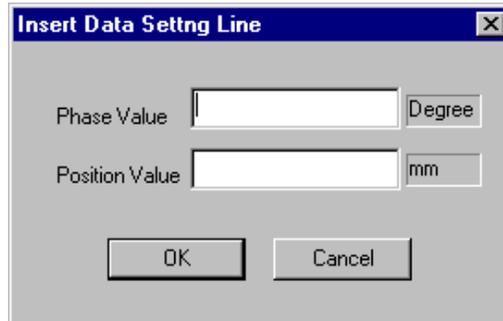
#### Editing the Data List

##### 1. Inserting row data

Move the cursor to a block before which a new line is to be inserted.

Click **Edit** in the menu bar and select **Insert**.

The following dialog box is displayed.



The dialog box titled "Insert Data Setting Line" contains two input fields. The first field is labeled "Phase Value" and has a unit selector set to "Degree". The second field is labeled "Position Value" and has a unit selector set to "mm". At the bottom of the dialog are "OK" and "Cancel" buttons.

Input values for Phase Value and Position Value. Both values must be input. Click the **OK** button, and the input data is inserted above the cell at which the cursor is located.

The data becomes effective when inputting is confirmed and then is reflected in the cam curve.

##### 2. Deleting the data line

Move the cursor to the line to be deleted.

Click **Edit** in the menu bar and select **Delete**. The line at which the cursor is located is then deleted.

##### 3. Undo function

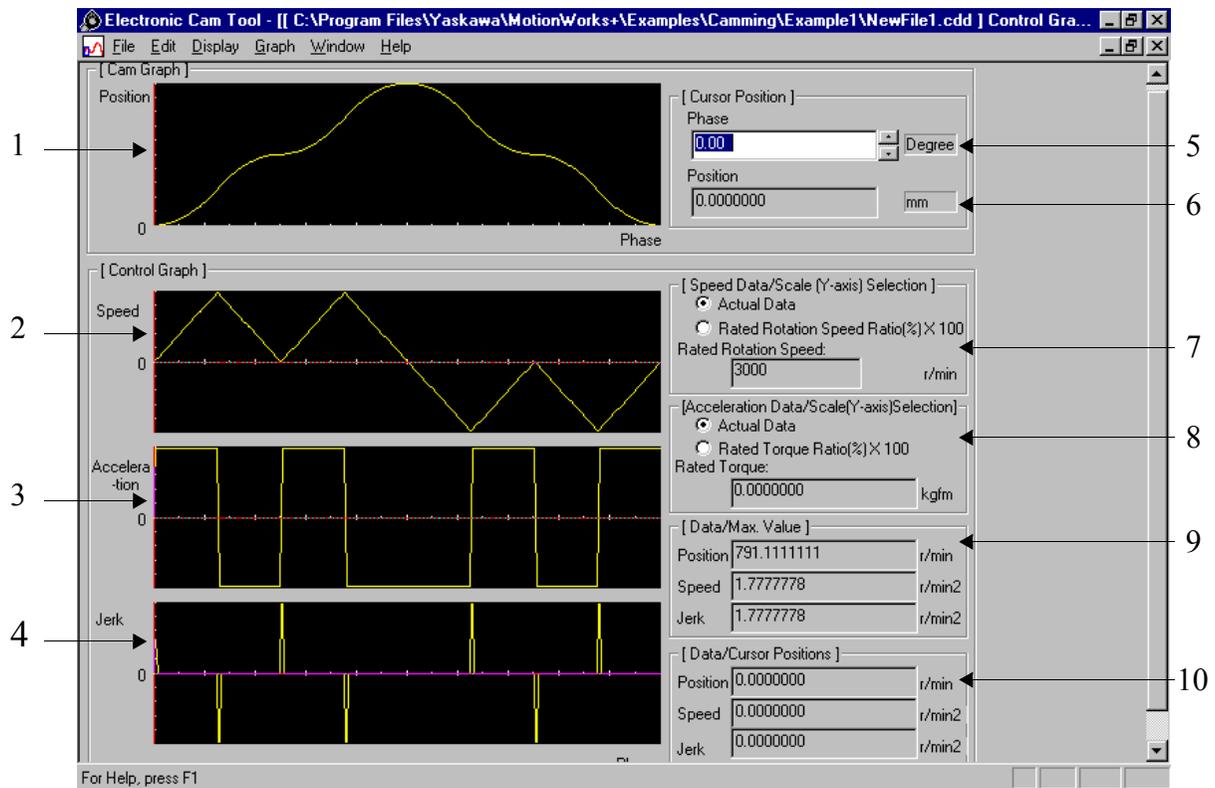
Selecting **Edit** > **Undo** from the menu bar of the Data List window cancels the very last operation or input.

##### 4. Resetting to the initial value

To reset all the edited data to the initial setting, click the **data edit combo box** button, select **Not Provided** or ---- **from Provided, Not Provided**, and ----, and click the **OK** button, which changes the data back to the initial value.

## Control Graph

The speed, acceleration, and jerk are graphically displayed.



### 1. Cam Graph/Position

The cam curve is displayed.

### 2. Control Graph/Speed

The speed data calculated based on the cam curve data is displayed.

### 3. Control Graph/Acceleration

The acceleration data calculated based on the cam curve data is displayed.

### 4. Control Graph/Jerk

The jerk data calculated based on the cam curve data is displayed.

**5. Cursor Position/Phase**

Enter the phase value or set with the arrow buttons.

By clicking a point on the cam graph, the phase and position values at which the mouse is pointing are displayed.

The X axis of the cam graph (1) shows the phase (master) value and the Y axis shows the position (slave) value.

**6. Cursor Position/Position**

The position value corresponding to the phase value set in step 5 (above) is displayed.

**7. Speed Data/Scale (Y-axis) Selection**

Select the graph display type of the speed data graph (2).

By selecting the radio button from Actual Data and Rated Rotation Speed Ratio (%), the speed control graph changes.

**8. Acceleration Data/Scale (Y-axis) Selection**

Select the graph display type of the acceleration data graph (3).

By selecting the radio button from Actual Data and Rated Torque Ratio (%), the acceleration control graph changes.

**9. Data/Maximum Value**

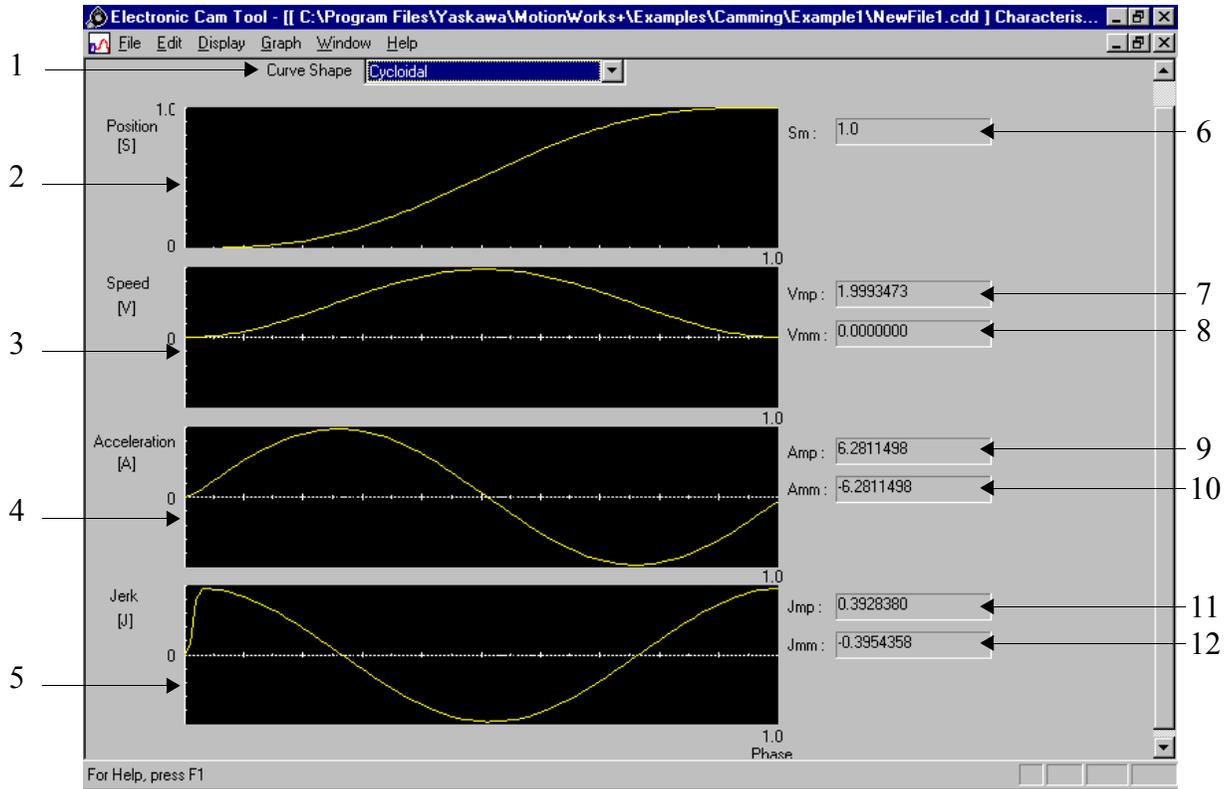
The maximum values of speed, acceleration and jerk are displayed.

**10. Data/Cursor Positions**

The speed, acceleration and jerk of the phase and position values at which the mouse is pointing on the cam graph are displayed.

## Characteristic Curve Graph

The position, speed, acceleration, and jerk of the designated curve shape are displayed as graphics.



### 1. Curve Shape

Display the curve shape combo box, scroll with the scroll bar, and click the curve shape desired. The graph is then displayed in the curve shape desired.

Available curve shapes are presented below.

1. Straight line	2. Parabolic	3. Simple harmonic
4. Cycloidal	5. Modified trapezoid	6. Modified sine
7. Modified constant velocity	8. Asymmetrical cycloidal	9. Asymmetrical modified trapezoid
10. Trapezoid	11. One-dwell cycloidal m=1	12. One-dwell cycloidal m=2/3
13. One-dwell trapezoid m=1	14. One-dwell trapezoid	15. One-dwell trapezoid m=2/3
16. One-dwell modified sine	17. One-dwell trapezoid	18. No-dwell simple harmonic
19. No-dwell modified trapezoid	20. No-dwell modified constant velocity	21. NC2 curve

And New as of September 2002, “Tangent Matching” curve type.

**2. Position**

The position data of the cam curve is displayed in the selected curve shape.

**3. Speed**

The speed data of the cam curve is displayed in the selected curve shape.

**4. Acceleration**

The acceleration data of the cam curve is displayed in the selected curve shape.

**5. Jerk**

The jerk data of the cam curve is displayed in the selected curve shape.

**6. Sm**

The maximum position value is displayed.

**7. Vmp**

The maximum positive speed is displayed.

**8. Vmm**

The maximum negative speed is displayed.

**9. Amp**

The maximum positive acceleration is displayed.

**10. Amm**

The maximum negative acceleration is displayed .

**11. Jmp**

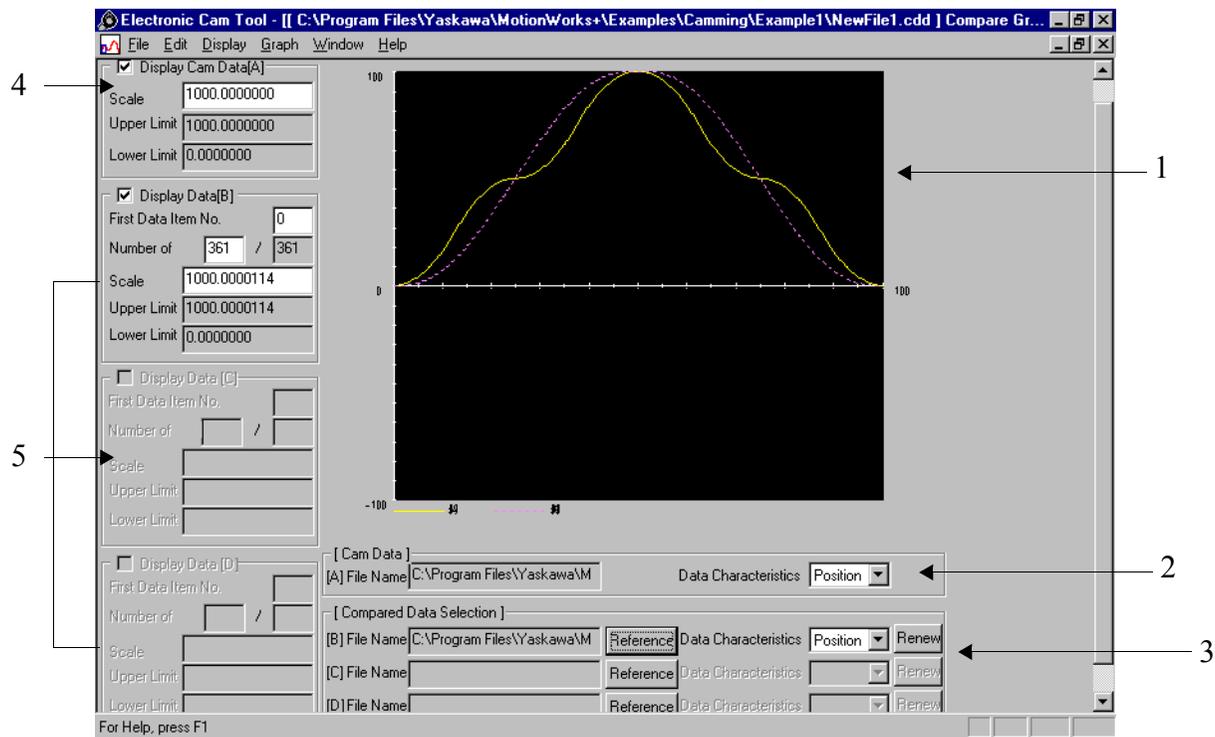
The maximum positive jerk value is displayed .

**12. Jmm**

The maximum negative jerk value is displayed.

## Compare Graphs

The currently edited cam data and either the cam data saved in the file or the external data which was prepared by means other than the cam tool and saved in the .csv format are displayed on the same graph for comparison.



### 1. Graph display

The currently edited cam data and data to be compared are displayed as graphics.

The currently edited cam data is displayed on a yellow line.

### 2. Cam Data

The file name of the currently edited cam data is displayed. Select the data to be displayed.

Data Characteristics ... Position, Speed, Acceleration

### 3. Compared Data Selection

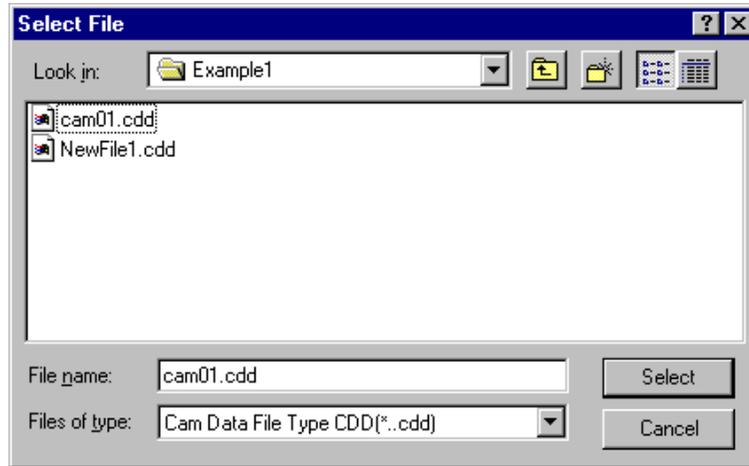
Select the data to be compared with the currently edited data. Up to three files can be selected.

File Name: Displays the file name of the data to be compared.

Reference: Select the data file to be compared.

Data Characteristics: Position, Speed, Acceleration, External

When the data to be compared is the cam data, select from **Position**, **Speed**, and **Acceleration**. When the data to be compared is prepared by means other than the cam tool, select **External**. Clicking the **Reference** button displays the following dialog box. Select the file to be compared.



#### 4. Display Cam Data

Select the display type of the currently edited cam data graph.

*Scale:* Set the scale of the graph (1) vertical axis to a value of 100 or less. The default values are the absolute upper and lower limit values of the cam data.

#### 5. Display Data

Select the display type of the comparison data graph. If the check is removed from the check box, the compared graph disappears. If the check is put in the check box, the graph is displayed again, because the data to be compared has been stored.

*First Data Item No.:* Set the first number of the data to be displayed as graphics.

*Number of:* Set the quantity of data from the first data number to the right of / (the slash); the total data quantity is displayed.

*Scale:* Set the scale of the graph (1) vertical axis to a value of 100 or less. The default values are the absolute upper and lower limit values of the data to be compared.

*Upper Limit:* Displays the upper limit (maximum) value of the data to be compared.

*Lower Limit:* Displays the lower limit (minimum) value of the data to be compared.

### Scale Conversion

The graph scale is set with the maximum absolute value of the displayed data as 100. Change the scale setting when comparing data whose upper and lower limit values are different, or when comparing data at a designated section.

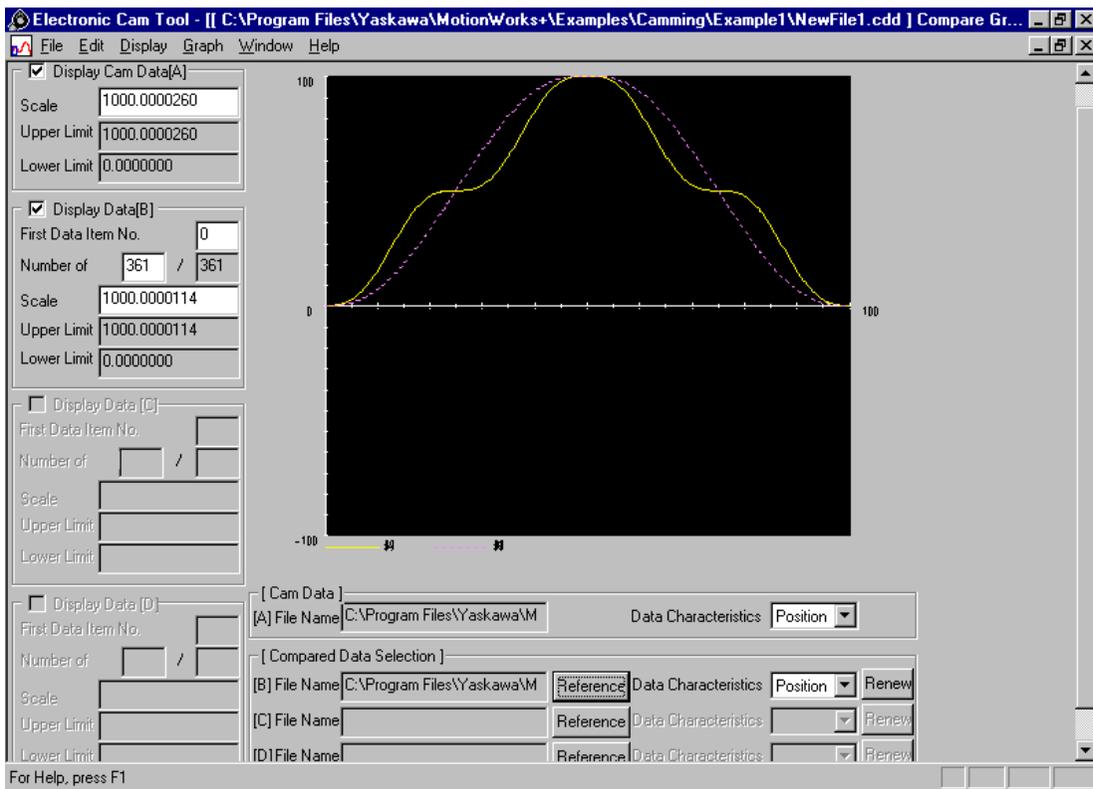
(a) Comparison of data with different upper and lower limit values

(Ex.) Display Cam Data A      Data Characteristics: PositionScale: 1000

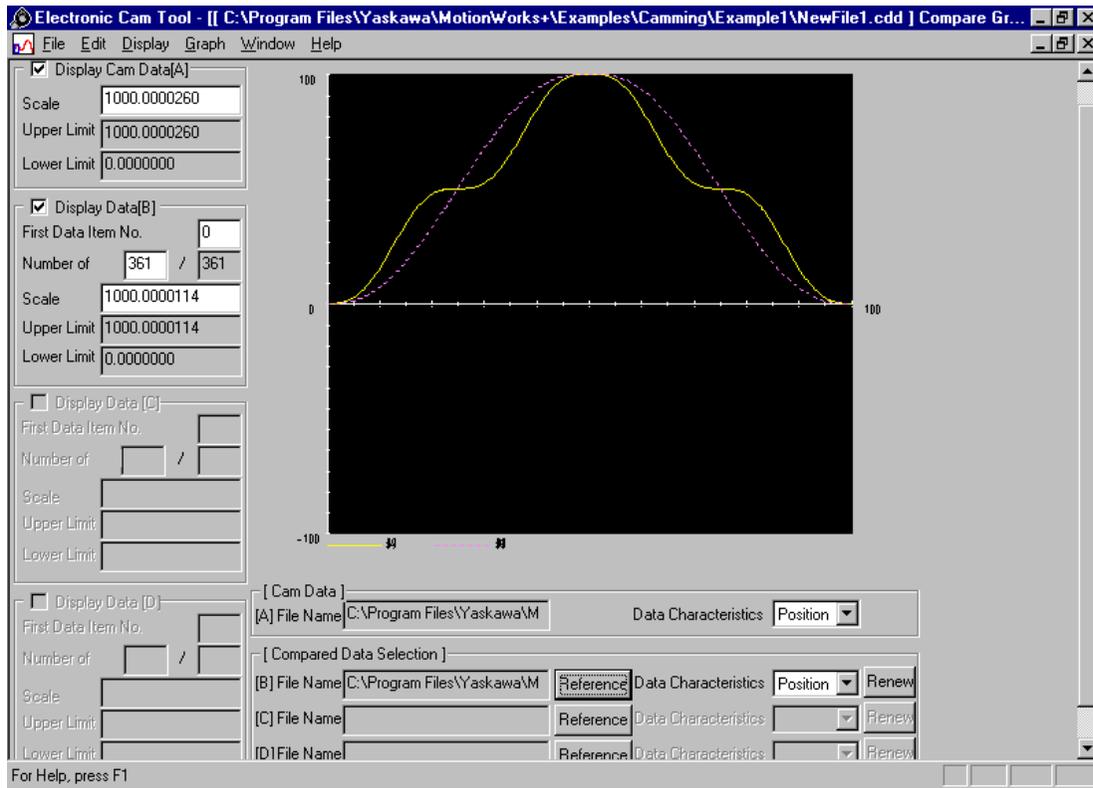
Upper Limit: 1000      Lower Limit: 0

Display Data B      Data Characteristics: PositionScale: 800.0000416

Upper Limit: 800.0000416      Lower Limit: 0



The scales for A and B are set to 1000.



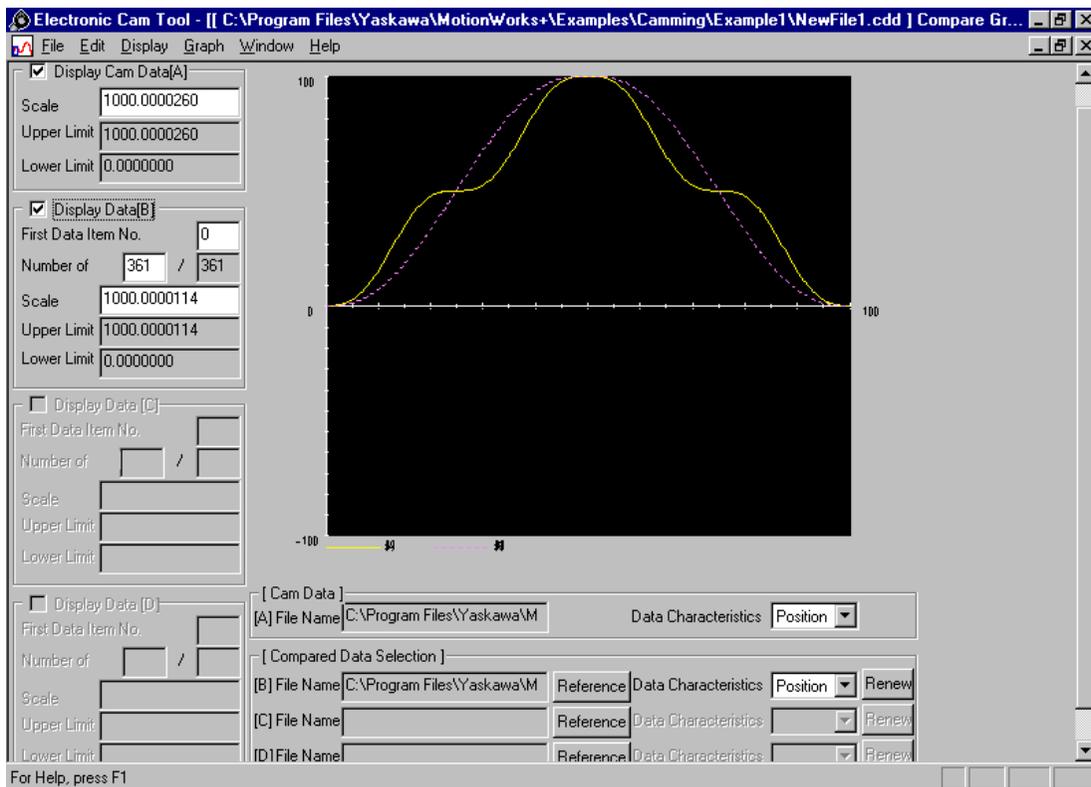
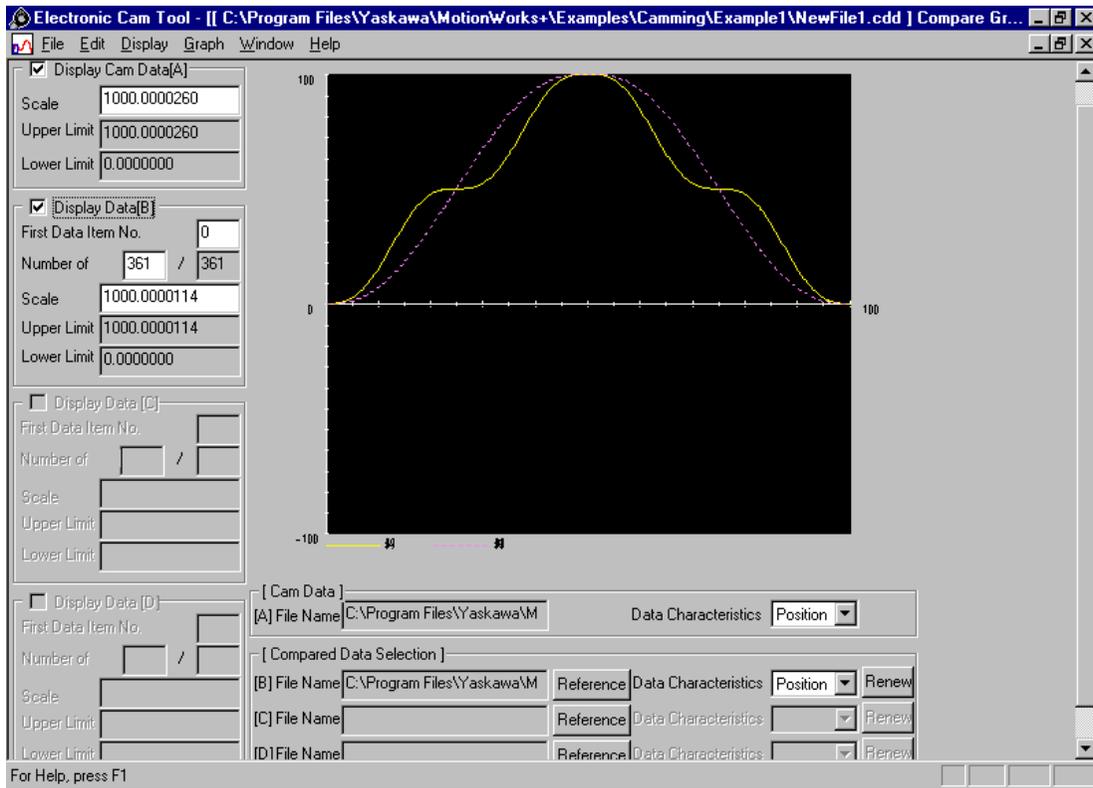
(b) Comparison of data at a designated section

(Ex.) Comparison of A and B

Display Cam Data A	Data Characteristics: Position	Scale: 1000
Upper Limit: 1000	Lower Limit: 0	
First Data Item No.: 0	Number of: 361	
Display Data [B]	Data Characteristics: Position	Scale: 1000
Upper Limit: 1000	Lower Limit: 0	

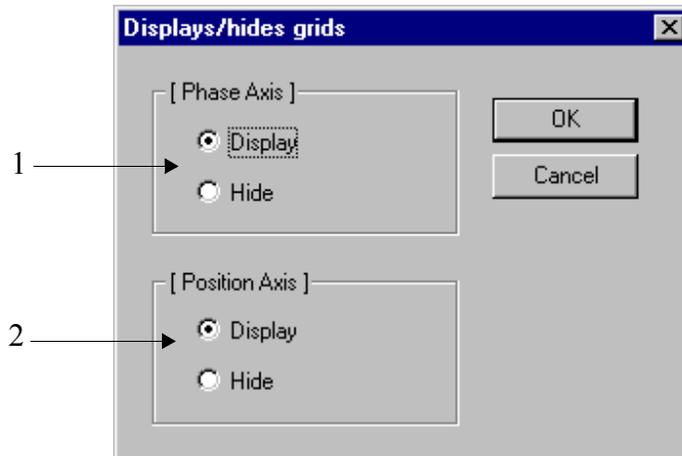
First Data Item No.: 5

Number of: 200



## Setting Grid Line

Set whether to display the graph grid line at the data graph window. When the data graph is displayed, select **Display** in the menu bar and then **Option**. The following dialog box appears.



### 1. Phase Axis

Set whether to display or to hide the phase axis grid line by clicking the radio button.

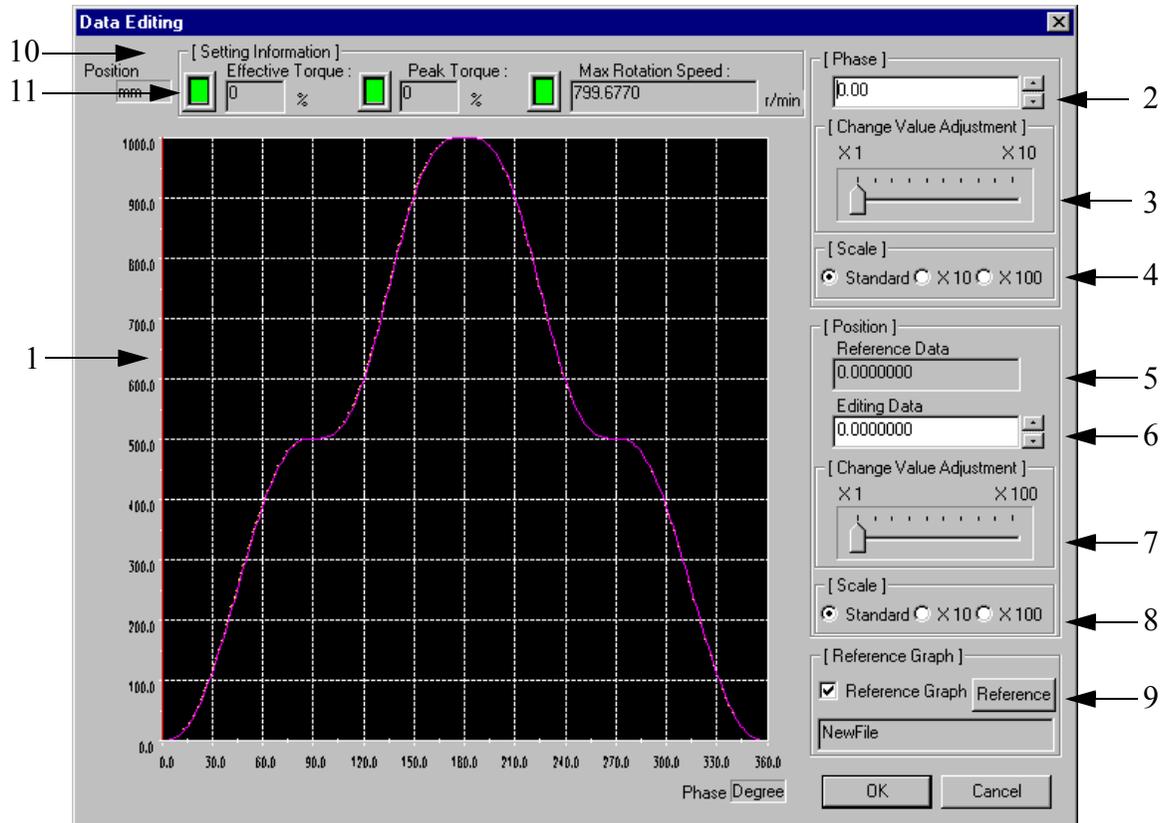
### 2. Position Axis

Set whether to display or to hide the position axis grid line by clicking the radio button.

The initial setting is Display.

## Editing Cam Data

From the main menu of the electronic cam tool, select **Graph > Edit Graph Data**. Position the mouse over a point in the graph. The slave position of that point appears in the editing data box (#6 below).



### 1. Edit graph

The graph can be edited with the mouse. Refer to “Using the Mouse” on page 96.

### 2. Phase

Select a phase (master) value. This allows editing the corresponding position (slave) value.

### 3. Phase/Change Value Adjustment

Sets the adjustment value when the phase value is set with the slide bar.

### 4. Phase/Scale

Select the phase display scale. When the graph is expanded, editing is easy.

### 5. Position/Reference Data

Position value corresponding to the phase value set in item 2 above is displayed.

**6. Position/Editing Data**

Set the position value corresponding to the phase value set in item 2 above.

**7. Position/Change Value Adjustment**

Set the increase/decrease value when the position value is set with the slide bar.

**8. Position/Scale**

Select the position display scale.

You can expand the graph to edit it accurately, using the current cursor position on the graph as the origin.

**9. Reference Graph**

Select the reference graph to be displayed together with the edit graph.

For the setting method, refer to item 4, Setting the reference graph.

**10. Setting Information**

Based on the edit data, the effective torque, peak torque, and maximum rotation speed are displayed.

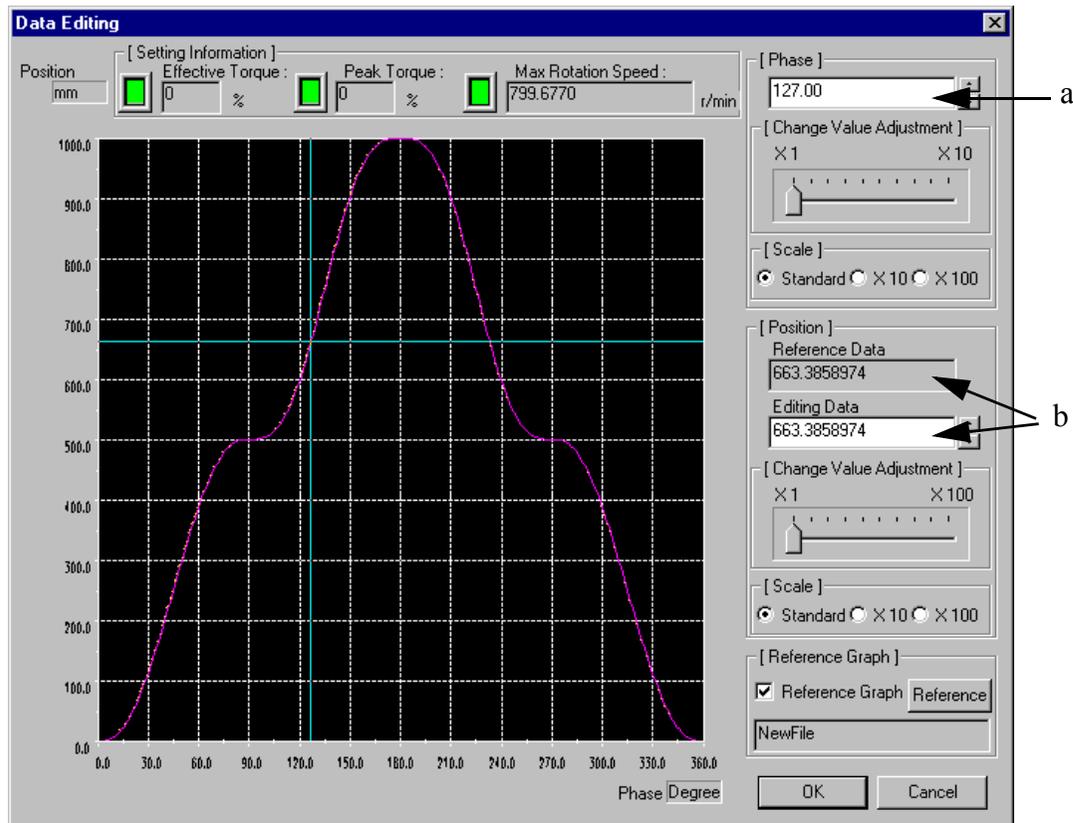
These values are automatically calculated and displayed if the items with an [\*] (for Machine/Motor Information) are set in the Set Style window.

Effective torque and peak torque are displayed as a percentage (%) of rated torque.

**11. Warning display**

The boxes are displayed in green when the set values for the effective torque (no more than 100%), peak torque (no more than 300%), and maximum rotation speed (under the rated rotation speed) are within the tolerances, and displayed in red when they are out of the tolerances.

## Using the Mouse



The phase value at which the mouse is pointing is displayed at 'a' above.

The position value at which the mouse is pointing is displayed at 'b' above.

The position values displayed at 'b' can be changed by moving the mouse up and down.

## Using the Keyboard

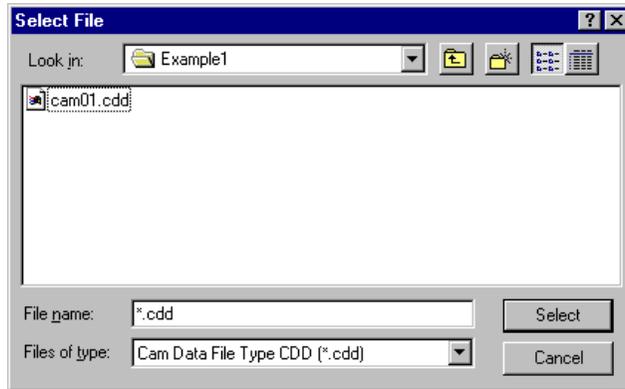
Input new values for the edit boxes 'b'.

## Expanding the Edit Graph

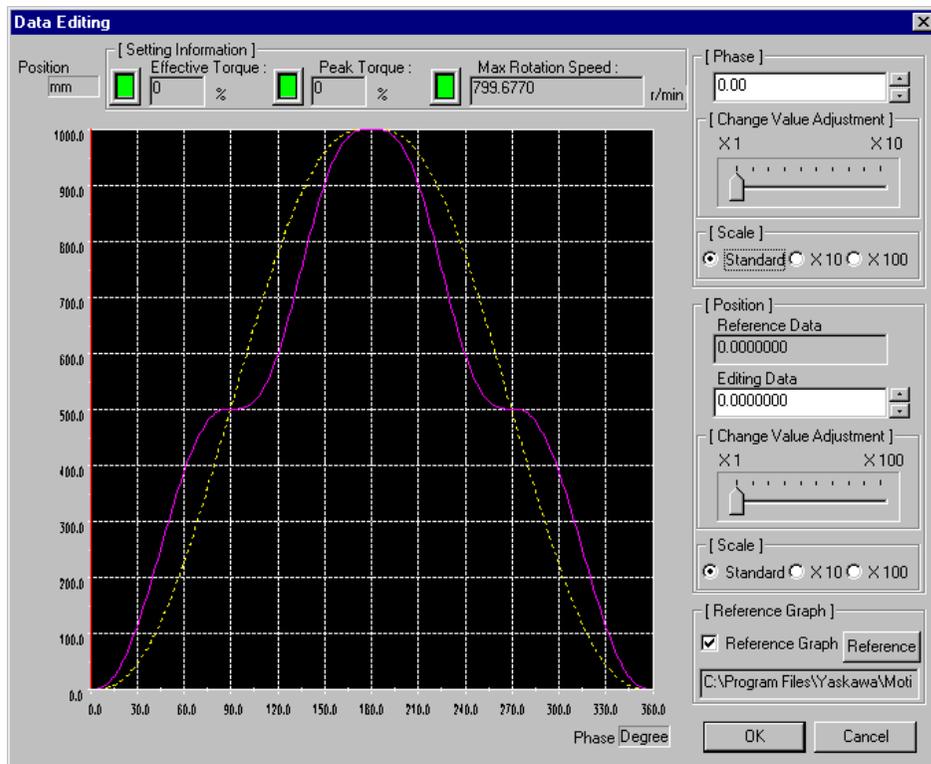
By selecting the scale radio button from X10 and X100, the graph after the current mouse position (phase value) is enlarged on the designated scale.

### Comparing Data to a Reference Graph

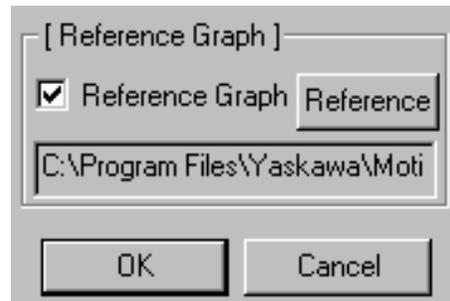
Clicking **Reference** displays the following Select File window.



Click the file to be compared and click the **Select** button.



The dotted line shows the reference graph.



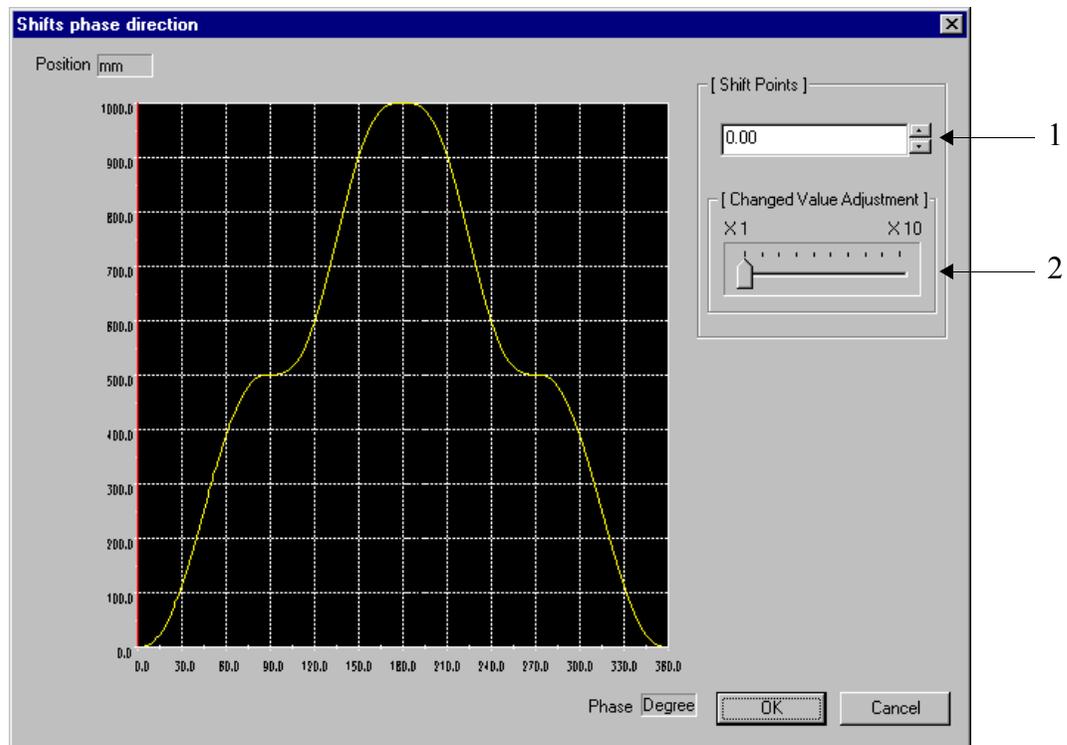
By removing the check from the Reference Graph box, the reference graph disappears.

When editing is completed, click the **OK** button.

As a condition for this display, Provided must be selected for Data Editing in the Set Parameter window.

Reset to the initial data in the Set Parameter window. To reset to the initial data, click the **data edit combo box** button, select **Not Provided** or **---- from Provided, Not Provided**, and **----**, and click the **OK** button.

### Shift Phase Direction



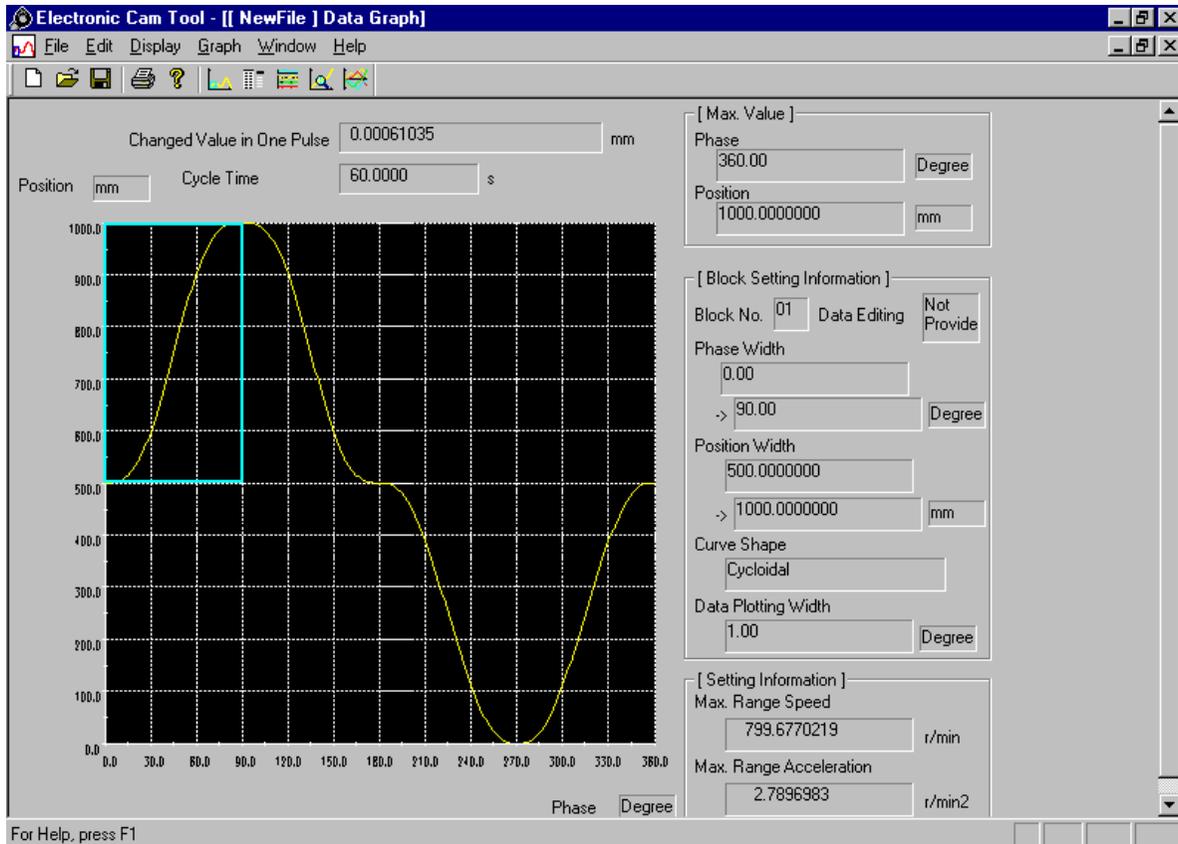
## 1. Changing the phase starting point

The data is rearranged with the phase value for which the shift point is set as a starting point.

The shift point is set at stages of the data division width.

By inputting a value for Shift Points, or by clicking a point on the graph with the mouse, the shift point value is displayed.

## 2. Adjusting the shift amount.

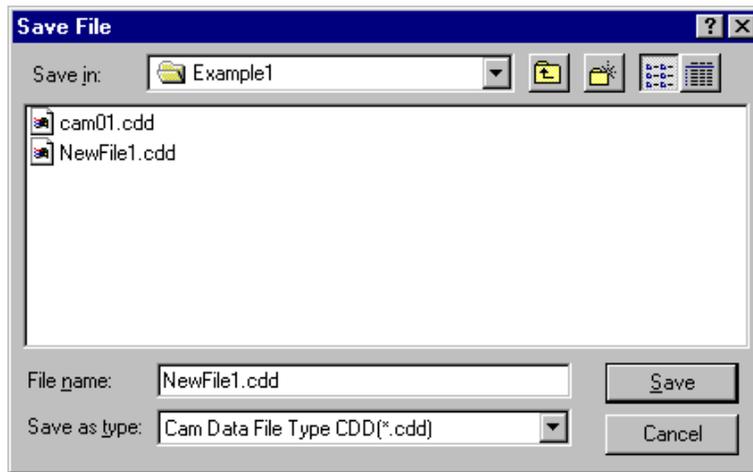


## Saving

### 1. Saving files

Select File > Save.

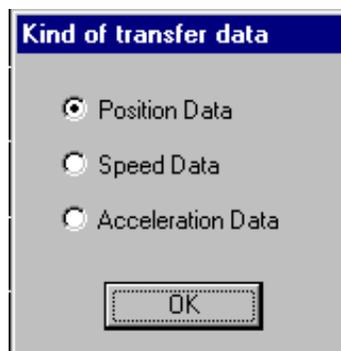
The Save File dialog box is displayed.



The cam data file type must be \*.cdd for use as a MotionWorks+™ table.

(Note: When the cam tool software is launched from MotionWorks+™, settings are passed to the cam tool in this format.)

The Select Save Data dialog box is displayed. Select the data type to be saved, then press the **OK** button.



## 1.19 Archive

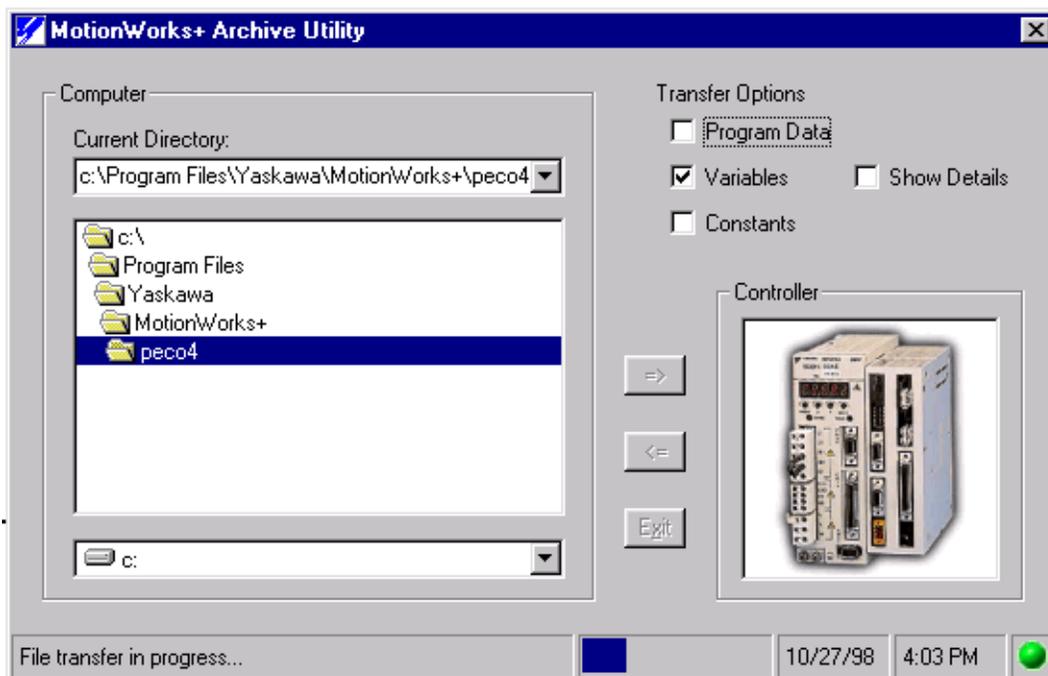
### Accessibility

To access the archive, select:

- From the Main Menu > Tools > Archive

This function is a simple utility for transferring the entire application program to/from the controller. It is intended for use when controller replacement is necessary or program duplication is required. This utility is also available as a stand alone program.

Note: The archive utility version transfers programs to/from the controller RAM. When transferring to the controller, the archiver also copies the project into flash memory.



The Archive Utility supplied with MW+ v2.83 is version 2.1.3 and includes the “Copy to FLASH” function as an automatic part of the process.

**Notes**

## 2. Icon-Based Motion Control Programming

### 2.1 Programming Tools

This section describes all the tools provided to create motion control programs. These tools are:

Blocks	System parameters
Connection lines	User variables
Expression Builder	User constants
System variables	User tables
I/O	Network variables

#### Blocks

The blocks represent specific events in machine operation. Each block has specific properties associated with it that detail its operation.

There are four main block categories: Motion, Logic, Program Flow, and General. They are grouped accordingly on the block toolbar. To place a block in the program, click and hold down the left mouse button while over a block on the toolbar, drag it over a program, and release the left mouse button.

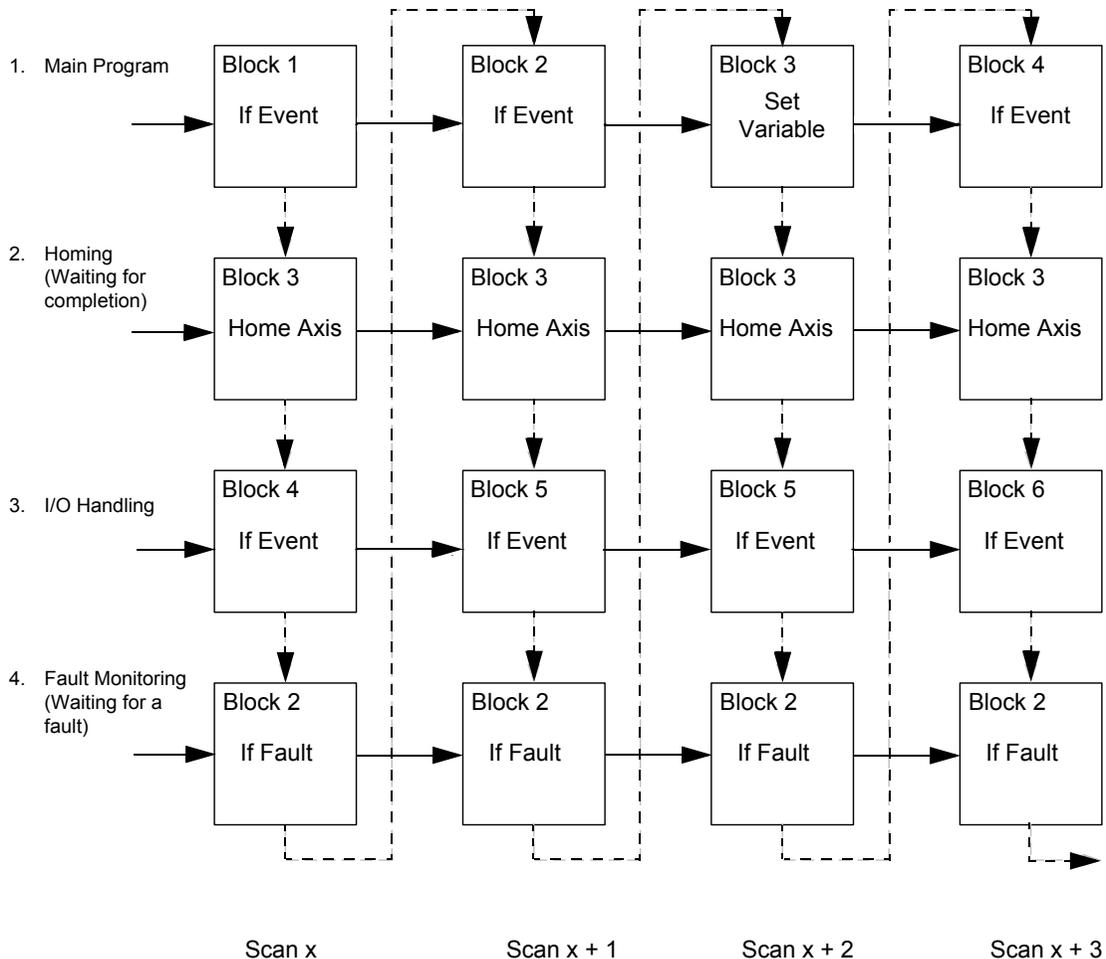
**Block Properties** - Some properties are settable only at design time; that is, they cannot be set with a variable and changed during program run. For example, the SERVO block's *Enabled* property has a true/false drop down box and cannot be equated to a variable. Other block properties can be equated to a variable, or expression. An example is the *Velocity* property of the MOVE AXIS block. If this property is a variable, then the speed of the servo can change every time the MOVE AXIS block executes. The variable may be updated by the program itself in a set variable block, or via Memobus serial communication, or from the Mechatrolink or DeviceNet network interface.

Some blocks have more than one output port. They are: IF EVENT, IF FAULT, TIMER, HOME AXIS. Typically, if the expression or event associated with that block results in a true result, (a logic one), then program flow is transferred to the block connected to the "true" output port. This is usually the upper block connection, and is also identified by the "T" symbol in the block itself. If the result of the block's expression is false, a logic zero is generated, and program flow is transferred to the block connected to the "false" port. This is typically the lower output port, identified by the "F" symbol in the block.

## Notes

### 3. Programming Concept

The programming environment incorporates a time-sharing scheme depicted in the diagram below. When programs are compiled and executing in the controller, they are executed according to a scan-based system. Code from the active block in each program executes at every scan interval. This process continues even if a block has not completed its process. For example, note that program 2 includes a HOME AXIS block in the diagram below. Assuming the *wait for completion* property is true, this block may take several hundred or thousand scans to complete. The program counter for Program 2 remains at block 3 while other program counters in the project continue to advance normally.



#### Number of programs

Up to eight user programs can be created in block format. Programs are activated by setting their Active Property in the Program Definition window.

Active – program is included in compile and download.

Autostart – automatically runs program at power up.

## **Subroutine operation**

Use the CALL SUBROUTINE block to incorporate subroutines in a program. When a CALL SUBROUTINE block is executed, the program counter is set to the first block in the selected subroutine. The main program that called the subroutine is not executing while the subroutine is active. When the end block of the subroutine is executed, the program counter is set to the block after the CALL SUBROUTINE block in the calling program. The maximum subroutine depth is eight levels for each program; a total of 62 subroutines can exist in each program.

## **High Priority Programs**

All high priority programs run code from their active block at precise intervals. The default scan rate is 1ms.

Types of program logic to be inserted into a high priority program

1. Critical Inputs / Outputs
2. Precise Move Timing

## **Low Priority Programs**

All low priority programs run code from the active block at precise intervals. The default is 20ms.

Types of program logic that should be inserted into a low priority program

1. Operator Interface
2. Non-critical or push button I/O

## Default I/O Names

MotionWorks+™ assigns default I/O names when a project is created. The following table is a reference guide showing the default I/O names and their physical locations. These names can be changed to more meaningful descriptions if necessary on a project-by-project basis.

Default I/O Name	Pin #	Minimum	Maximum	Register
Local_Analog_Output	MP940-IO-1	-32768	32767	OW0001
Local_Analog_Output_Voltage	MP940-IO-1	-10	10	MF32054
Local_Input_Bank	N/A	0	255	IW0000
Local_Input1	MP940-IO-14	0	1	IB00000
Local_Input2	MP940-IO-39	0	1	IB00001
Local_Input3	MP940-IO-15	0	1	IB00002
Local_Input4	MP940-IO-40	0	1	IB00003
Local_Input5	MP940-IO-16	0	1	IB00004
Local_Input6	MP940-IO-41	0	1	IB00005
Local_Input7	MP940-IO-17	0	1	IB00006
Local_Input8	MP940-IO-42	0	1	IB00007
Local_Output_Bank	N/A	0	255	OW0000
Local_Output1	MP940-IO-21	0	1	OB00000
Local_Output2	MP940-IO-46	0	1	OB00001
Local_Output3	MP940-IO-22	0	1	OB00002
Local_Output4	MP940-IO-47	0	1	OB00003
Local_Output5	MP940-IO-23	0	1	OB00004
Local_Output6	MP940-IO-48	0	1	OB00005
Local_Output7	MP940-IO-24	0	1	OB00006
Local_Output8	MP940-IO-49	0	1	OB00007
Sigma_Analog_Input	SGDH-CN1-5	-32768	32767	IW0001
Sigma_Analog_Input_Voltage	SGDH-CN1-5	-10	10	MF32052
Sigma_EXT1	SGDH-CN1-44	0	1	IBC0254
Sigma_EXT2	SGDH-CN1-45	0	1	IBC0255
Sigma_HomeInput	SGDH-CN1-41	0	1	IBC0251
Sigma_Latch_Input	SGDH-CN1-46	0	1	IBC0256
Sigma_NOT	SGDH-CN1-43	0	1	IBC0253
Sigma_POT	SGDH-CN1-42	0	1	IBC0252
Sigma_ServoOn	SGDH-CN1-40	0	1	IBC0250

## Internal Data Access

The following table identifies internal variables used by the controller for various functions. The names of these variables cannot be changed. Based on the read/write attribute of each variable, they can be assigned values with the SET VARIABLE block, or used in expressions in the IF EVENT block.

The internal data is comprised of three categories, identifiable by the first character in the name.

- m = Monitor                      read only data (R)
- p = Parameter                  read and write data (R/W)
- s = System Variable read and write data (R/W)

Note: “—” means the minimum/maximum range is calculated based on the project system properties. Use the formulas provided in the User Unit Conversion section to determine your system’s min and max ranges.

**Table 1: Monitors**

Name	Minimum	Maximum	Units	R/W	Address
mAcceleration_Linear_Rate	calculated	calculated	MP user units/s <sup>2</sup>	R	MF32126
mAcceleration_SCurveTime	0	255	mSec	R	OWC014
mAlarm_Battery	0	1	status	R	SB000487
mAlarm_Servo	0	1	status	R	IBC0010
mAlarm_Servo_Code	0	255	status	R	IWC024
mCamCycle	calculated	calculated	Ext user units	R	ML31272
mCamOffset	-2147483648	2147483647	Ext user units	R	MF31282
mCamShift	calculated	calculated	Ext user units	R	MF32200
mDeceleration_Linear_Rate	calculated	calculated	MP user units/s <sup>2</sup>	R	MF32128
mError_Analog_Input	0	1	status	R	SB002140
mError_Analog_Output	0	1	status	R	SW00215
mError_CPU	0	65535	status	R	SW00041
mError_Digital_Inputs	0	1	status	R	SB002120
mError_Digital_Outputs	0	1	status	R	SB002130
mError_Network_A	0	65535	status	R	SW00224
mError_Network_B	0	65535	status	R	SW00225
mError_Network_C	0	65535	status	R	SW00226
mError_Network_D	0	65535	status	R	SW00227
mError_Parameter_Fixed	0	1	status	R	IBC0002
mError_Parameter_Number	0	65535	status	R	IWC00F
mError_Parameter_Settable	0	1	status	R	IBC0001
mExternal_APhase	0	1	status	R	IB0002A

**Table 1: Monitors (Continued)**

Name	Minimum	Maximum	Units	R/W	Address
mExternal_BPhase	0	1	status	R	IB0002B
mLimit_Hardware_Negative	0	1	status	R	IBC0222
mLimit_Hardware_Positive	0	1	status	R	IBC0221
mLimit_Speed_Detected	0	1	status	R	IBC0016
mLimit_Torque_Detected	0	1	status	R	IBC0015
mMachineCycle_External_Counts	1	2147483647	counts	R	ML31130
mMachineCycle_Main_Counts	1	2147483647	counts	R	ML31110
mPosition_Actual	calculated	calculated	MP user units	R	MF31108
mPosition_Actual_Counts	-2147483648	2147483647	counts	R	ML31098
mPosition_Commanded	calculated	calculated	MP user units	R	MF32046
mPosition_Complete	0	1	status	R	IBC000D
mPosition_Error	calculated	calculated	MP user units	R	MF32050
mPosition_Error_Exceeded	0	1	status	R	IBC0000
mPosition_Ext_Mod_Latch_Counts	1	2147483647	counts	R	ML31086
mPosition_External	calculated	calculated	Ext user units	R	MF31128
mPosition_External_Counts	-2147483648	2147483647	counts	R	ML31118
mPosition_External_Shifted	1	2147483647	counts	R	ML31288
mPosition_Latch_External	calculated	calculated	Ext user units	R	MF32040
mPosition_Latch_External_Counts	-2147483648	2147483647	counts	R	IL0008
mPosition_Latch_Main	calculated	calculated	MP user units	R	MF32042
mPosition_Latch_Main_Counts	-2147483648	2147483647	counts	R	ILC006
mPosition_Main_Mod_Latch_Counts	1	2147483647	counts	R	ML31084
mPosition_Target	calculated	calculated	MP user units	R	MF32044
mProgram_HighScan_Current	0	32768	mSec/100	R	SW00005
mProgram_HighScan_Maximum	0	32768	mSec/100	R	SW00006
mProgram_LowScan_Current	0	32768	mSec/100	R	SW00011
mProgram_LowScan_Maximum	0	32768	mSec/100	R	SW00012
mPulse1	0	1	bit	R	SB000010
mPulse1000	0	1	bit	R	SB000012
mPulse500	0	1	bit	R	SB000011
mSpeed_Coincidence	0	1	status	R	IBC0012
mSpeed_External	calculated	calculated	Ext user units/s	R	MF31126
mSpeed_Main	calculated	calculated	MP user units/s	R	MF31106
mSpeed_Main_Commanded	calculated	calculated	MP user units/s	R	MF32048
mState_ABSEncoder_Direction	0	1	status	R	IBC0009
mState_BrakeEnabled	0	1	status	R	IBC0017
mState_Camming	0	4	status	R	MW31261
mState_DIP_Switches	0	255	status	R	SW00048

**Table 1: Monitors (Continued)**

Name	Minimum	Maximum	Units	R/W	Address
mState_ExternalLatch	0	1	status	R	MB315210
mState_Gearing	0	1	status	R	MW31221
mState_LatchTarget	0	1	status	R	MB315413
mState_Main_Latch	0	1	status	R	MB315010
mState_MainPower	0	1	status	R	IBC0019
mState_Program1	0	1	status	R	SB005000
mState_Program2	0	1	status	R	SB005001
mState_Program3	0	1	status	R	SB005002
mState_Program4	0	1	status	R	SB005003
mState_Program5	0	1	status	R	SB005004
mState_Program6	0	1	status	R	SB005005
mState_Program7	0	1	status	R	SB005006
mState_Program8	0	1	status	R	SB005007
mState_ServoReady	0	1	status	R	IBC0014
mState_ServoRotation	0	1	status	R	IBC0013
mState_SVON	0	1	status	R	IBC0018
mTime	-2147483648	2147483647	counter	R	ML31152
mTorque_Actual	-327.68	327.67	% rated torque	R	IWC00E
mWarningOverload	0	1	status	R	MB311400
mWarningRegen	0	1	status	R	MB311401

**Table 2: Parameters and System Variables**

Name	Default	Minimum	Maximum	Units	R/W	Address
pGain_FeedForward	0	0	200	%	R/W	OWC011
pGain_Integral_PControl	300	0	32767	mSec	R/W	OWC035
pGain_PositionLoop	300	0	32767	0.1/Sec	R/W	OWC010
pLimit_Integration	32767	0	32767	—	R/W	OWC036
pLimit_PositioningError	0	calculated	calculated	MP user units	R/W	MF32120
pLimit_Speed	—	calculated	calculated	MP user units/s	R/W	MF32122
sAcceleration_Virtual	0	32767	0	Ext user units/s <sup>2</sup>	R/W	MF31076
sCamShift_Clear	0	0	1	event	W	MB312803
sCamShift_Mode	0	0	1	configuration	R/W	MB312804
sCamUpdate	0	0	1	event	W	MB312604
sExternalMode	0	0	1	configuration	R/W	MB310640
sHome_Offset	0	calculated	calculated	MP user units	R/W	MF32134
sLimit_Speed_Negative	—	calculated	calculated	MP user units/s	R/W	MF32130

**Table 2: Parameters and System Variables (Continued)**

Name	Default	Minimum	Maximum	Units	R/W	Address
sLimit_Speed_Positive	—	calculated	calculated	MP user units/s	R/W	MF32132
sLimit_Torque	-300.00	-327.68	327.67	% rated torque	R/W	OWC002
sMachineCycle_External	1	0.000001	2147483647	Ext user units	R/W	MF31122
sMachineCycle_Main	1	0.000001	2147483647	MP user units	R/W	MF31012
sPosition_CompletionWindow	—	calculated	calculated	MP user units	R/W	MF32124
sPosition_Virtual_Counts	0	-2147483648	2147483647	Ext User Units	R/W	ML31068
sSlaveOffset_Mode	0	0	1	configuration	R/W	MB313011
sSpeed_Virtual	—	0	200.00	Ext user units/s	R/W	MF31070
sTorque_Commanded	0	-327.68	327.67	% rated torque	R/W	OWC01B

## Internal Data Definition

### Monitors

**mAcceleration\_Linear\_Rate** - The current acceleration rate of the servo in MP user units/s<sup>2</sup>. This value is back calculated from the controller units of “mSec to rated speed.” Any of the motion blocks set acceleration, and the setting can be verified by checking this system variable. If it does not agree, it is likely that the original setting was out of range and thus limited by the system.

**mAcceleration\_SCurveTime** - The acceleration S Curve setting in milliseconds. The lower the number, the less the filtering; the higher the number, the smoother the S-Curve smoothing. A given move will take longer by the amount of this variable times two (acceleration & deceleration.)

**mAlarm\_Battery** - A status bit for the lithium battery used for memory backup. A "0" indicates that the battery is OK, a "1" indicates that the battery voltage is low and should be replaced.

**mAlarm\_Servo** - A status bit for the SGDh amplifier. A "0" indicates no alarm, a "1" indicates that there is an alarm, and mAlarm\_Servo\_Code should be checked.

**mAlarm\_Servo\_Code** - The Alarm Code from the SGDh amplifier. The alarm displayed on the front panel is identical, but shown in decimal. A value of 99H on the SGDh is 153 decimal, which means no alarm. See the list below.

SGDH Front Panel (Hex)	mAlarm_Servo_Code (Dec)	Alarm Name	Description
2	2	Parameter Breakdown	EEPROM data in the servo amplifier is abnormal.
3	3	Main Circuit Encoder Error	Detection data for the power circuit is abnormal.
4	4	Parameter Setting Error	The parameter setting is outside the allowable setting range.
5	5	Servomotor and Amplifier Combination Error	Servo amplifier and servomotor capacities do not match each other.
10	16	Overcurrent or Heat Sink Overheated	An overcurrent flowed through the IGBT. Heat sink of servo amplifier is overheated.
30	48	Regeneration Error Detected	Regenerative circuit is faulty. Regenerative resistor is faulty.
32	50	Regenerative Overload	Regenerative energy exceeds regenerative resistor capacity.
40	64	Overvoltage	Main circuit DC voltage is excessively high.
41	65	Undervoltage	Main circuit DC voltage is excessively low.
51	81	Overspeed	Rotational speed of the servo is excessively high.
71	113	Overload: High Load	The motor was operating for several seconds to several tens of seconds under a torque greatly exceeding ratings.
72	114	Overload: Low Load	The motor was operating continuously under a torque exceeding the ratings.
73	115	Dynamic Brake Overload	When the dynamic brake was applied, rotational energy exceeded the capacity of the dynamic brake resistor.
74	116	Overload of Surge Current Limit Resistor	The main circuit was frequently turned ON and OFF.
7A	122	Heat Sink Overload	The heat sink of the servo amplifier overheated.
81	129	Absolute Encoder Backup Error	All of the power supplies for the absolute encoder have failed and position data was cleared.
82	130	Encoder Checksum Error	The checksum results of the encoder memory is abnormal.
83	131	Absolute Encoder Battery Error	Battery voltage for the absolute encoder has dropped.

<b>SGDH Front Panel (Hex)</b>	<b>mAlarm_Servo_Code (Dec)</b>	<b>Alarm Name</b>	<b>Description</b>
84	132	Absolute Encoder Data Error	Received absolute data is abnormal.
85	133	Absolute Encoder Over-speed	The encoder was rotating at a high speed when the power was turned on.
86	134	Encoder Overheated	The internal temperature of the encoder is too high.
99	153	Not an Alarm	Everything in the servo amplifier is OK.
91	145	Overload Warning	This warning occurs before either of the overload alarms (71h or 72h). If the warning is ignored and operation continues, a regenerative overload alarm may result.
92	146	Regenerative Overload Warning	This warning occurs before alarm (32h). If the warning is ignored and operation continues, a regenerative overload alarm may result.
94	148	MP940 Data Setup Warning	Invalid or out of range data. See the MP940 alarms for more information.
95	149	MP940 Invalid Command Warning	Inappropriate command was issued for the current control state.
9F	159	Option Card Warning (MP940)	This warning indicates something is wrong on the MP940. Typically it is an I/O error, such as external encoder disconnected.
B1	177	Reference Speed Input Read Error	The A/D converter for reference speed input is faulty.
B2	178	Reference Torque Input Read Error	The A/D converter for reference torque input is faulty.
BF	191	System Alarm	A system error occurred in the servo amplifier.
C1	193	Servo Overrun Detected	The servo motor ran out of control.
C8	200	Absolute Encoder Clear Error and Multi-Turn Limit Setting Error	The multi-turn for the absolute encoder was not properly cleared or set.
C9	201	Encoder Communications Error	Communications between servo amplifier and encoder is not possible.
CA	202	Encoder Parameter Error	Encoder parameters are faulty.
CB	203	Encoder Echoback Error	Contents of communications with the encoder is incorrect.
CC	204	Multi-Turn Limit Disagreement	Different multi-turn limits have been set in the encoder and servo amplifier.
D0	208	Position Error Pulse Overflow	Position error pulse exceeded parameter Pn505. This is only for position mode, not used when MP940 is connected.

SGDH Front Panel (Hex)	mAlarm_Servo_Code (Dec)	Alarm Name	Description
E0	224	Communication Timeout (SGDH and MP940)	Amplifier could not communicate with the MP940 connected to the DPRAM within 10 seconds of power up.
E2	226	Watch Dog Timer between SGDH and MP940	Out of synchronization. This is normal when downloading a program. If pGain_PositionLoop is set to zero, this can cause AE2 alarm.
E7	231	Option Board Not Connected	An option board (MP940) that was previously connected is no longer present. Use amplifier function Fn14 to reset this alarm.
E9	233	MP940 Alarm	This alarm is generated by the MP940. The MP940 alarms for more information.
EA	234	—	SGDH does not respond at power on or after reset.
EB	235	—	SGDH initial access error. SGDH power on start up confirmed, but response is absent or faulty.
EC	236	—	Watch dog timer error. SGDH ran away or WDT abnormal.
ED	237	—	Command Execution incomplete.
F1	241	Power Line Open Phase	One phase is not connected to the main power supply.
CPF00	—	Digital Operator Transmission Error	—
CPF01	—	Digital Operator Transmission Error	—

**mCamCycle** - The size of the cam cycle in user units of the cam table (master). This value is derived by looking at the last master position in the table.

**mCamOffset** - The relative CAM SHIFT last issued. This value is the value entered in the CAM SHIFT block's *Shift* property.

**mCamShift** - This is the absolute CAM SHIFT in user units of the external encoder. This value is added to the external encoder before `mPosition_External` is modularized for the cam function. Modularization of `mCamOffset` is not required; i.e., a cam profile can be shifted by an amount greater than the machine cycle.

**mDeceleration\_Linear\_Rate** - The current deceleration rate of the servo in user units/ $s^2$ . The same description as `mAcceleration_Linear_Rate` applies.

**mError\_Analog\_Input** - A status bit that indicates if there is a problem with the analog input circuitry. This could indicate either local or Mechatrolink I/O. Check the `mError_Network*` system variables if a mechatrolink network is implemented.

**mError\_Analog\_Output** - A status bit that indicates if there is a problem with the analog output circuitry. This could indicate either local or Mechatrolink I/O. Check the `mError_Network*` system variables if a mechatrolink network is implemented.

**mError\_CPU** - A word containing status bits indicating various errors. If the SGDH displays A9F, check this system variable for specific details.

Bit	Description	Check
F	CERF Error	—
E	Network	<code>mError_Network_A</code> <code>mError_Network_B</code> <code>mError_Network_C</code> <code>mError_Network_D</code>
D	External Encoder	<code>mExternal_APhase</code> <code>mExternal_BPhase</code>
C	SVA - Motion controller section	<code>mPosition_Error_Exceeded</code> <code>mError_Parameter_Fixed</code> <code>mError_Parameter_Number</code> <code>mError_Parameter_Settable</code>
B	Local I/O transmission error	—

Bit	Description	Check
A	Illegal interruption error	—
9	Local I/O	mError_Digital_Inputs mError_Digital_Outputs mError_Analog_Input mError_Analog_Output
8	User operation error	—
7	unused	—
6	unused	—
5	unused	—
4	unused	—
3	unused	—
2	unused	—
1	Program memory error	—
0	Major fault	—

**mError\_Digital\_Inputs** - A "1" indicates that there is an error with the digital input circuitry. This could indicate either local or Mechatrolink I/O. Check the mError\_Network\* system variables if a mechatrolink network is implemented.

**mError\_Digital\_Outputs** - A "1" indicates that there is an error with the digital outputs. The internal fuse could be damaged. This could indicate either local or Mechatrolink I/O. Check the mError\_Network\* system variables if a mechatrolink network is implemented.

**mError\_Network\_A** - Status bits indicating if any of the network slaves have a fault.

Station	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Hex-Mask	\$8000	\$4000	\$2000	\$1000	\$0800	\$0400	\$0200	\$0100	\$0080	\$0040	\$0020	\$0010	\$0008	\$0004	\$0002	\$0001

**mError\_Network\_B** - Status bits indicating if any of the network slaves have a fault.

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Station	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hex-Mask	\$8000	\$4000	\$2000	\$1000	\$0800	\$0400	\$0200	\$0100	\$0080	\$0040	\$0020	\$0010	\$0008	\$0004	\$0002	\$0001

**mError\_Network\_C** - Status bits indicating if any of the network slaves have a fault.

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Station	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Hex-Mask	\$8000	\$4000	\$2000	\$1000	\$0800	\$0400	\$0200	\$0100	\$0080	\$0040	\$0020	\$0010	\$0008	\$0004	\$0002	\$0001

**mError\_Network\_D** - Status bits indicating if any of the network slaves have a fault.

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Station	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Hex-Mask	\$8000	\$4000	\$2000	\$1000	\$0800	\$0400	\$0200	\$0100	\$0080	\$0040	\$0020	\$0010	\$0008	\$0004	\$0002	\$0001

**mError\_Parameter\_Fixed** - Status bit that has a value of "1" when a fixed parameter is out of range. The number of the parameter that is out of range is stored in mError\_Parameter\_Number. Determining what the fixed parameter description may require cross referencing with the MP940 / Motionworks documentation.

**mError\_Parameter\_Number** - If a parameter is out of range, this system variable holds the parameter number. If the value is greater than 100, the fixed parameter # (mError\_Parameter\_Number - 100) is out of range. If the value is less than 100, the settable parameter # (mError\_Parameter\_Number) is out of range.

**mError\_Parameter\_Settable** - Status bit that has a value of "1" when a settable parameter is out of range. The number of the parameter that is out of range is stored in mError\_parameter\_Number.

**mExternal\_APhase** - A status bit that indicates current flow (a connection) of the external encoder's "A" phase. A value of "0" means that there is current flow, and the signal is working properly and a value of "1" means that there is no current flow in the encoder channel, and it may be disconnected.

**mExternal\_BPhase** - A status bit that indicates current flow (a connection) of the external encoder's "B" phase. A value of "0" means that there is current flow, and the signal is working properly and a value of "1" means that there is no current flow in the encoder channel, and it may be disconnected.

**mLimit\_Hardware\_Negative** - Status bit that has a value of "1" when the negative hardware overtravel limit switch on the SGDh is active. Note: The limit switch must be mapped to the default location in the SGDh for this bit to function.

**mLimit\_Hardware\_Positive** - Status bit that has a value of "1" when the positive hardware overtravel limit switch on the SGDh is active. Note: The limit switch must be mapped to the default location in the SGDh for this bit to function.

**mLimit\_Speed\_Detected** - Status bit that has a value of "1" when the servo is commanded to rotate at a speed greater than either the *sLimit\_Speed\_Negative* or *sLimit\_Speed\_Positive*.

**mLimit\_Torque\_Detected** - Status bit that has a value of "1" when the servo is commanded to output a torque greater than *sLimit\_Torque\_Positive*.

**mMachineCycle\_External\_Counts** - The machine cycle of the external encoder as converted by the system ladder from user units into counts. Note: When *sMachineCycle\_External* is changed, it takes up to one low scan time to update the value in counts.

**mMachineCycle\_Main\_Counts** - The machine cycle of the main encoder as converted by the system ladder from user units into counts. Note: When *sMachineCycle\_Main* is changed, it takes up to one low scan time to update the value in counts.

**mPosition\_Actual** - The current position of the servo motor in user units updated every high scan.

**mPosition\_Actual\_Counts** - The current position of the servo motor updated every high scan.

**mPosition\_Commanded** - The servo commanded position in user units. This is the final target of a move. If camming or gearing are enabled, then this is the commanded target at each scan.

**mPosition\_Complete** - Status bit that has a value of "1" when the actual position of the servo (*mPosition\_Actual*) is within the required tolerance (*sPosition\_Completion\_Width*) of the final destination (*sPosition\_Commanded*). Typically this is the *Position* property of the MOVE AXIS block, or the *Target Distance* property of the LATCH block.

**mPosition\_Error** - The difference between the commanded position of the servo (*mPosition\_Target*) and the actual position of the servo (*mPosition\_Actual*). The value is updated at every high speed scan. When this value exceeds *pLimit\_PositioningError*, *mPosition\_Error\_Exceeded* has a value of "1", and an A9F alarm is generated.

**mPosition\_Error\_Exceeded** - Status bit that indicates if the servo position error *mPosition\_Error* has exceeded *pLimitPositionError*. When the limit has been exceeded, the value of this monitor parameter is "1", and an A9F warning displays on the SGDH. An alarm clear is required to reset this error.

**mPosition\_Ext\_Mod\_Latch\_Counts** - This is the external latch in counts minus the cumulative number of machine cycles that have occurred. Note: This value will be inaccurate over time if a non integer based machine cycle is used.

**mPosition\_External** - The current position of the external encoder in user units. The position is updated at every high speed scan. If the external encoder is configured as a rotary axis, this value represents the position within the rotary limits of the machine (the modularized value).

**mPosition\_External\_Counts** - The current position of the external encoder. The position is updated at every high speed scan.

**mPosition\_External\_Shifted** - This is the value that is fed into the cam look up table. It is modularized and includes the absolute cam shift. It is in table units of the master, and represents the exact position of the master as viewed from the cam function's perspective. This value may be useful for forcing the slave to engage immediately, without waiting for a window to appear. For these situations, use this system variable in the CAM engage block's "*External Position*" property.

**mPosition\_Latch\_External** - The location of the external encoder in user units when the external latch input occurred. This value is only valid after *mState\_External\_Latch* has a value of "1", indicating that the input has occurred. If the axis is configured as a rotary axis, then this value will represent the latch position within the machine cycle. Note: If a non-integer machine is used, this value will become inaccurate over time.

**mPosition\_Latch\_External\_Counts** - The external encoder latch in counts. The value is only valid after *mState\_External\_Latch* has a value of "1". This value is always unmodularized.

**mPosition\_Latch\_Main** - The location of the main encoder (SGDH) when the main latch input occurred. This value is only valid after *mState\_Main\_Latch* has a value of "1", indicating that the input has occurred.

**mPosition\_Latch\_Main\_Counts** - The main encoder (SGDH) latch in counts. The value is only valid after *mState\_Main\_Latch* has a value of "1". This value is always unmodularized.

**mPosition\_Main\_Mod\_Latch\_Counts** - This is the modularized version of the latch on the main encoder. If the system properties are not configured for a Rotary axis, this value may be invalid.

**mPosition\_Target** - The commanded position of the servo in user units. This is where the servo should be (calculated position) at each high scan. The profiler determines this value based on the acceleration, speed, and deceleration properties.

**mProgram\_HighScan\_Current** - The amount of time in milliseconds/100 that it actually took to execute all the code for all programs actively running in high scan.

**mProgram\_HighScan\_Maximum** - The longest time it took to run the High Scan Programs since the power was turned on. This value is in milliseconds/100. Example 87 = .87ms.

**mProgram\_LowScan\_Current** - The amount of time in milliseconds/100 that it actually took to execute all the code for all programs actively running in low scan.

**mProgram\_LowScan\_Maximum** - The longest time it took to run the Low Scan Programs since the power was turned on. This value is in milliseconds/100. Example 707 = 7.07ms.

**mPulse1** - A bit that toggles every high scan. This can be useful in the HOME block's "*Home Input*" property when homing only to the "C" channel is required, it effectively makes the HOME block think it already hit the home switch.

**mPulse500** - A bit that changes state every 500ms.

**mPulse1000** - A bit that changes state every 1000ms.

**mSpeed\_Coincidence** - A bit that has a value of "1" when mSpeed\_Main is equal to mSpeed\_Main\_Commanded.

**mSpeed\_External** - The current speed of the external encoder in user units per second. This value is filtered over 20 low scans and is updated every low scan.

**mSpeed\_Main** - The current speed of the servo in user units/second. This value is filtered over 20 low scans and is updated every low scan.

**mSpeed\_Main\_Commanded** - This is the commanded speed of the servo back calculated from the native units of the controller, which is percentage of rated speed. This provides a way to verify that a motion block that set the speed correctly. If a value was entered in a motion block that would cause the native unit to be out of range, the maximum native unit would be set.

**mState\_ABSEncoder\_Direction** - Status bit that has a value of "0" when the absolute encoder is counting up for forward direction, and a value of "1" when the absolute encoder is counting up for the reverse direction.

**mState\_BrakeEnabled** - Status bit that has a value of "1" when the mechanical brake is disabled and the motor is free to rotate.

**mState\_Camming** - Status word that indicates the current condition of the camming function. If the value is 2 and the slave is not moving, there may be a stationary period

State	Description
0	Not camming
1	Slave is waiting to engage
2	Camming is actively synchronized
4	Slave is waiting to disengage

in the cam table, the CAM SCALE may be at 0%, the external encoder may be set to “*Enabled=false*” or if the virtual encoder is selected, it may have a speed of zero.

**mState\_DIP\_Switches** - This is a status word that indicates the configuration of the MP940 DIP Switches. This value could be useful for comparing the DIP switch state of a controller to the required settings intended by the machine designer (i.e. FLASH and COPY switches,) and prohibit operation or provide a message to the operator.

**mState\_External\_Latch** - Status bit that indicates the condition of the external latch input on the MP940. Zero indicates that the latch has not occurred yet; one means the latch has occurred, and the position values stored in *mPosition\_Latch\_External* and *mPosition\_Latch\_External\_Counts* are valid. When the latch has been armed and a latch input received, this status bit remains on (“one”) until the latch is re-armed.

**mState\_Gearing** - Status bit indicating the status of electronic gearing. A "0" indicates that gearing is not enabled; a "1" indicates enabled.

**mState\_LatchTarget** - Indicates the result of the last LATCH TARGET block operation. When the block is executed, the bit is cleared. If the latch was detected within the window, the final position target is updated, and this status bit is turned on (1). When the latch has been armed and a latch input received, this status bit remains on until the latch is rearmed.

**mState\_Main\_Latch** - Status bit that indicates the condition of the main (SGDH) latch input. A "0" indicates that the latch has not occurred yet; a "1" means the latch has occurred, and the position values stored in *mPosition\_Latch\_Main* and *mPosition\_Latch\_Main* are valid. When the latch has been armed and a latch input received, this status bit remains on ("one") until the latch is re-armed.

**mState\_MainPower** - Status bit that has a value of "1" when the SGDH has main power applied. It does not indicate whether control power is applied to the servo amplifier.

**mState\_Program1 - mState\_Program8** - Status bit that has a value of "1" when the program in question is running, and a "0" when the program is not running.

**mState\_ServoReady** - A status bit that has a value of "1" if the servo has no alarms, and the main power is applied. The servo however, may not be enabled. Check the status of *mState\_SVON*, which will indicate if the servo is in RUN mode.

**mState\_SVON** - Status bit that has a value of "1" when the servo is enabled (run), and a value of "0" when disabled. If the main power is removed, or the SGDH has an alarm, this bit will have a value of "0".

**mTime** - A free running counter that counts milliseconds since the power was turned on. The counter is updated every high scan; therefore its resolution is the same as the high scan. The counter uses a 32-bit variable and rolls over automatically as follows:

0 >> 2147483647 >> -2147483648 >> 0 >> 2147483647 etc.

**mTorque\_Actual** - The torque output of the servo expressed as a percentage of rated torque (i.e., 30000 = 300%).

**mWarningOverload** - A status bit that has a value of one when the SGDH has a warning code of A91. (The servo is being overloaded and will cause a fault and shut down soon if the load is not decreased).

**mWarningRegen** - A status bit that has a value of one when the SGDH has a warning code of A92. The servo is decelerating hard with a large load, which is stressing the regenerative capabilities of the unit, and if it persists, the amplifier will cause a fault and shut down soon.

## Parameters

**pGain\_FeedForward** - Describes the amount of bias that is applied to the servo throughout the move. As the servo is commanded to rotate faster, the amount of bias changes to keep following errors to a minimum.

**pGain\_Integral\_PControl** - The integration value used when tuning the servo. The lower the value, the quicker the response.

**pGain\_PositionLoop** - Adjusts the response of the servo. A higher value causes the servo to achieve its commanded position more quickly when acted upon by external forces, or acceleration. If this value is too high for the machine rigidity, vibration results.

**pLimit\_Integration** - Defines a limit to the amount of integration. The lower the number, the less the integration has an effect on the servo tuning. This can help reduce oscillation.

**pLimit\_PositioningError** - Defines the maximum allowable following error (in user units) the servo can accumulate before an alarm is generated. This alarm is set in `mPosition_Error_Exceeded`.

**pLimit\_Speed** - This parameter is only applicable when the servo is in torque mode. It defines the maximum allowable speed of the servo. If the commanded torque causes the servo to go beyond this value, the torque is reduced to keep the speed of the servo within this limit.

## Settable System Variables

**sAcceleration\_Virtual** - If the virtual encoder mode is used, this is the accel and decel rate applied to any speed changes of `sSpeed_Virtual`. This value is in external user units/second<sup>2</sup>.

**sCamShift\_Clear** - Clears the absolute cam shift (`mCamShift`) value to zero. Set this system variable to "1" to enable. It will automatically set itself back to zero when the operation is performed, which takes only one high scan. The cam shift value `mCamShift` can only be cleared if camming is not engaged. If camming is engaged and `sCamShift_Clear` is set to one, the function will not work until camming is disengaged.

**sCamShift\_Mode** - This is a configuration setting which dictates how the CAM SHIFT block will behave. If set to “0,” the *Duration* property of the CAM SHIFT block is time in mSec. The *Shift* will complete in the time specified. If set to “1,” the *Duration* property is a relative change in position of the master, in external user units. The shift will take as long as required for the master to move the relative distance specified. This mode is recommended when it is critical for the correction to be complete within a known distance.

**sCamUpdate** - The bit causes the cam function to redetermine the last master & slave position in the cam table. The bit will automatically turn itself off after completing this task, which only takes one high scan. Use this function if the cam table must be changed on the fly, especially if the last master or slave points are changed. It is necessary because there are supporting functions that must be updated to keep the cam running properly i.e scale the master encoder to the actual range of master positions in the table if necessary, add the slave offset each time the cycle completes.

**sExternalMode** - A configuration setting that switches between the real and virtual internal encoder. If set to “0”, the real external encoder will be used for all system variables concerning the external encoder, i.e. mSpeed\_External, mPosition\_External, mPosition\_External\_Counts. If set to “1,” the virtual encoder will be used and it’s value will be reflected in all related external system variables. If the external encoder is moving at the time sExternalMode is changed to virtual, mSpeed\_Virtual will take on the speed of the real encoder so seamless switchover is possible.

**sHome\_Offset** - The Home Offset in MP940 User Units. After the HOME block is complete, the servo will move this amount and then define the position as zero.

**sLimit\_Speed\_Negative** - Defines the maximum allowable negative speed of the servo for speed or torque modes. If the servo is commanded to go beyond this value, the actual speed of the servo is limited.

**sLimit\_Speed\_Positive** - Defines the maximum allowable positive speed of the servo for speed or torque modes. If the servo is commanded to go beyond this value, the actual speed of the servo is limited.

**sLimit\_Torque** - Defines the maximum amount of torque that may be applied to the servo during operation. This is useful when the force applied to a load must be limited to avoid damage.

**sMachineCycle\_External** - This is the modularization value if the external encoder is configured for rotary mode. This value is identical to the external encoder configuration property *Machine Cycle* and can be changed at run time. Important: when this value is changed, it takes up to a low scan time period for the change to be effective. This is because the conversion from User Units to pulses occurs in the low scan. Care should be taken to be sure the value has actually been updated.

**sMachineCycle\_Main** - This is the modularization value if the main encoder is configured for rotary mode. This value is identical to the MP940 configuration property *Machine Cycle* and can be changed at run time. Important: when this value is changed, it takes up to a low scan time period for the change to be effective. This is because the conversion from User Units to pulses occurs in the low scan. Care should be taken to be sure the value has actually been updated.

**sPosition\_CompletionWindow** - Defines the maximum distance allowed between the final target and the servo (in user units) for a move to be considered complete. If the “Wait for Completion” property is set (MOVE block), it will not complete until the position is within this distance. If this property is set too high, the block will complete before the move is done.

**sPosition\_Virtual\_Counts** - This is the virtual counter which is used as the external encoder if sExternalMode is set to “1”. This value is updated every high scan by an amount consistent with sPSpeed\_Virtual. If the DEFINE POSITION block is used for the external encoder, this value is set also.

**sSlaveOffset\_Mode** - A configuration setting which dictates how the SLAVE OFFSET block behaves. If set to “0,” the *Duration* property of the SLAVE OFFSET block is time in mSec. The *Offset* will complete in the time specified. If set to “1,” the *Duration* property is a relative change in position of the master, in external user units. The Offset will take as long as required for the master to move the relative distance specified. This mode is recommended when it is critical for the correction to be complete within a known distance.

**sSpeed\_Virtual** - If the virtual encoder is selected using sExternal\_Mode=1, this property will set the speed of the virtual counter. If at the time sExternalMode is set for virtual the real external encoder is moving, sSpeed\_Virtual will be set to the same speed as the external encoder. This property is in External User Units/Second.

**sTorque\_Commanded** - This is the same as the Torque property of the TORQUE Block.

### 3.1 User Unit Conversion

Throughout the block definition which follows in this section, the minimum and maximum range of properties are listed in MP940 base units. The following calculations are provided to aid in determining the minimum and maximum values in user units.

#### Accel and Decel Formulas

The following formula shows conversion from *user units/s<sup>2</sup>* to *milliseconds to rated speed*. In the example, assume the rated motor speed is 3000rpm, the feed constant is 9 user units per machine revolution, and the acceleration setting is 100 user units per s<sup>2</sup>.

Accel Setting	Rated Motor Speed	Convert to Seconds	Feed Constant	Gear Box	Convert to Milliseconds	Base Units
---------------	-------------------	--------------------	---------------	----------	-------------------------	------------

$$\frac{1 \text{ s}^2}{100 \text{ Units}} \times \frac{3000 \text{ rev}}{1 \text{ min}} \div \frac{60 \text{ s}}{1 \text{ min}} \times \frac{9 \text{ units}}{1 \text{ rev}} \times \frac{2 \text{ input}}{1 \text{ output}} \times \frac{100 \text{ ms}}{1 \text{ s}} = 9000 \text{ ms}$$

Conversely, the following formula can be used to determine minimum and maximum user units.

Base Units	Rated Motor Speed	Convert to Seconds	Feed Constant	Gear Box	Convert to Milliseconds	ACCEL Setting
------------	-------------------	--------------------	---------------	----------	-------------------------	---------------

$$\frac{1}{32768 \text{ ms}} \times \frac{3000 \text{ rev}}{1 \text{ min}} \div \frac{60 \text{ s}}{1 \text{ min}} \times \frac{9 \text{ units}}{1 \text{ rev}} \times \frac{2 \text{ input}}{1 \text{ output}} \times \frac{1000 \text{ ms}}{1 \text{ s}} = \frac{27.465 \text{ units}}{1 \text{ s}^2}$$

### Position Formulas

The formula below shows the conversion from *user units* to *encoder counts*. In the example, assume the position setting is 100 units, the motor encoder resolution is 8192 quadrature counts, and the feed constant is 9 units per machine revolution. The gear box is 2:1.

Position Setting	Encoder Resolution	Feed Constant	Gear Box	Encoder Counts
100 units	$\times \frac{8192 \text{ counts}}{1 \text{ rev}}$	$\div \frac{9 \text{ units}}{1 \text{ rev}}$	$\times \frac{2 \text{ input}}{1 \text{ output}}$	= 182044 counts

Conversely, the following formula can be used to determine the maximum absolute position for any one move.

Encoder Counts	Encoder Resolution	Feed Constant	Gear Box	Position Setting
2147438647 counts	$\div \frac{8192 \text{ counts}}{1 \text{ rev}}$	$\times \frac{9 \text{ units}}{1 \text{ rev}}$	$\div \frac{9 \text{ input}}{1 \text{ output}}$	= 4718493.12085 units

## Speed Formulas

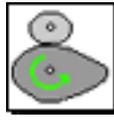
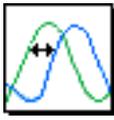
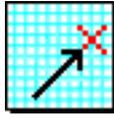
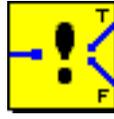
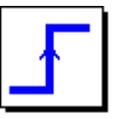
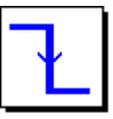
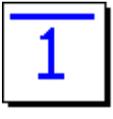
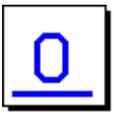
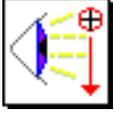
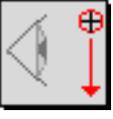
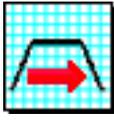
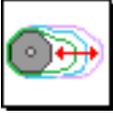
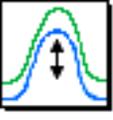
The formula below illustrates the conversion from *user units per second* to *percentage of rated speed*. In the example below, assume the speed setting is 100 units per second, the rated motor speed is 3000rpm, and the feed constant is 9 units per machine revolution.

Speed Setting	Rated Motor Speed	Convert to seconds	Feed Constant	Gear Box	Convert to 10000 = 100%	% of Rated Speed
$\frac{100 \text{ units}}{1 \text{ s}}$	$\div \frac{3000 \text{ rev}}{1 \text{ min}}$	$\times \frac{60 \text{ s}}{1 \text{ min}}$	$\div \frac{9 \text{ units}}{1 \text{ rev}}$	$\times \frac{2 \text{ input}}{1 \text{ output}}$	$\times \frac{10000\%}{100}$	$= 11.11\%$

Conversely, the following formula can be used to determine minimum and maximum speed in user units per second.

% of Rated Speed	Rated Motor Speed	Convert to seconds	Feed Constant	Gear Box	Convert to 10000 = 100%	Speed Setting
150%	$\times \frac{3000 \text{ rev}}{1 \text{ min}}$	$\div \frac{60 \text{ s}}{1 \text{ min}}$	$\times \frac{9 \text{ units}}{1 \text{ rev}}$	$\div \frac{2 \text{ input}}{1 \text{ output}}$	$\div \frac{10000\%}{100}$	$= \frac{1350 \text{ units}}{1 \text{ s}}$

### 3.2 Block Reference

				
Call Subroutine	Cam	Cam	Cam Shift	Change Dynamics
				
Define Position	End	Gear	Gear	Gear Ratio
				
Home Axis	IF Event	IF Fault	Input	Input
				
Input	Input	Jog Axis	Latch	Latch
				
Latch Target	Launch Program	Move Axis	PLS	Reset Fault
				
Scale Cam	Servo	Servo	Set Variable	Slave Offset
				
Start	Stop Motion	Suspend Program	Timer	Torque

## Notes

### 3.2.1 CALL SUBROUTINE



#### Definition

This block represents a subroutine. The program counter is passed to the first block within the subroutine, and passed back to this block after all blocks in the subroutine have executed and the END block of the subroutine is encountered.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Title	N/A	N/A	N/A	N/A	No

*Title:* The name of any subroutine in the project.

#### Required Conditions

1. The selected subroutine must not be executing at the time this block is executed.
2. At least one subroutine must exist within the project.

### 3.2.2 CAM



#### Definition

This block enables or disables the camming function. If the Enabled property is true, the slave engages when the master reaches the position specified in the *External Position* property. If the Enabled property is false, then the slave disengages when the master reaches the *External Position*. The *External Position* window is defined to be 1% of the total cam cycle (sMachineCycle\_External).

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Enabled	True	False	True	Status bit	No
Position Data	N/A	N/A	N/A	Table	No
External Position	0	0	2147483647	Table Units	Yes

*Enabled:* Sets whether the block causes the slave to engage or disengage when the master passes the *External Position*.

*Position Data:* A table name which contains data pertaining to the master and slave positions. A file type must be \*.cdd, created by the Electronic Cam Tool software, provided with the MotionWorks+™ package.

*External Position:* The position of the master at which the slave engages/disengages, i.e., starts/ends synchronization based on the *Enabled* property setting. Note: This property is in the same units as the master in the position data table. This must be a positive number in the range of the master cam data or the slave may not engage properly.

#### Required Conditions

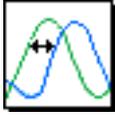
1. A .cdd table must be defined to assign in the Table Definition window.
2. The servo must have been previously enabled using the SERVO block.

### **Additional Information**

To engage or disengage the slave immediately, for situations such as E-Stop, or to re-engage the slave at its current position, use the System Variable `mPositionExternal_Shifted`. This System Variable is in table units, and includes the absolute CAM SHIFT amount. This is the actual master value that is used to determine the slave's position, so using it in the CAM block's *External Position* property will cause the window requirement to be satisfied immediately.

For more information about camming, see the appendix.

### 3.2.3 CAM SHIFT



#### Definition

This block changes the master/slave relationship during camming by the specified relative amount. This shift is a relative, non-modularized value in user units of the master. The shift is added internally to the external encoder position. The shift occurs as a modified sine profile from the current shift to the new shift over the duration specified. This block will perform the shift using one of two methods: time or relative change of the master. Configure the operation by setting system variable sCamShift\_Mode.

The system ladder holds the absolute shift in mCamShift. This value is automatically reset to zero when the external encoder is defined using the DEFINE POSITON block. The programmer can also choose to reset it by using sCamShift\_Clear. This will only take effect if camming is not engaged. This is useful when it is preferred to set an absolute shift (i.e. the first one is always absolute.)

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Shift	0	-2147483648	2147483647	encoder counts	Yes
Duration	0	0	2147483647	mSec / encoder counts	Yes
Wait for Completion	True	False	True	N/A	No

*Shift:* The relative amount in user units of the master to shift the master/slave relationship.

*Duration:* Duration in milliseconds or relative change of the master to complete the shift. Note: If sCamShift\_Mode=1, the shift will occur over a relative position change of the master. If the master stops during the shift, the shift will pause too. The benefit of this method is that the correction will be complete when the machine moves a specific amount, regardless of the machine speed.

*Wait for Completion:*

If enabled, the program counter does not advance to the next block until the shift is complete.

**Required Conditions**

None

**Additional Information**

NOTE: Prior to MW+ release 2.56 and system Ladder 223, this block's *Offset* property was an absolute shift. To improve accuracy, 2.56 MW+ with System Ladder 223 performs relative shifts, while keeping the absolute shift value internally in terms of counts.

NOTE: System ladder 228 or greater allows shifting over time or relative change of the master.

NOTE: If the block is configured for position based shift, if the master stops and moves backwards the shift will "shrink." If the master moves beyond the original location where the CAM SHIFT block first executed, the shift will "advance" again because the absolute value of the relative position is used to determine the required shift amount.

**(Care should be taken that the master does not move backwards, the resulting shift will be incorrect.)**

NOTE: If the STOP block is used while the CAM SHIFT is in progress, the CAM SHIFT will be aborted at it uncompleted location.

### 3.2.4 CHANGE DYNAMICS



#### Definition

This block changes acceleration, deceleration, velocity, and s-curve settings while the servo is already in motion. This block does not initiate motion. A previously set target position will remain unchanged. Leaving a property blank (zero value) does not affect current settings when this block executes.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Acceleration	0	32767	0	ms to rated speed	Yes
Deceleration	0	32767	0	ms from rated speed	Yes
S-curve	0	0	32767	ms	Yes
Velocity	0	-200.00	200.00	% of rated speed	Yes

*Acceleration:* The acceleration rate of the servo in *user units / second<sup>2</sup>*.

*Deceleration:* The deceleration rate of the servo in *user units / second<sup>2</sup>*.

*S-Curve:* The amount of s-curve acceleration/deceleration in milliseconds.

*Velocity:* The velocity of the move profile in *user units / second*.

#### Required Conditions

The servo must already be in motion via a block such as MOVE AXIS, JOG, LATCH TARGET, GEAR for the property updates to have any effect.

### 3.2.5 DEFINE POSITION



#### Definition

This block sets the current encoder location as the value specified in the *Position* property. It does not cause motion. Select either the main or external encoder. This block may take as long as the Low Scan Program time to complete and advance the program counter, even if the block is in a High Scan Program. This block automatically waits until the encoder value has been changed.

Note: It is not recommended to include the DEFINE POSITION block in a program loop that repeatedly defines the position, because a small, cumulative position error will result. The motor may move a slight amount between the time that the DEFINE POSITION block runs and the hardware counter is changed.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Encoder	main	External	Main	N/A	No
Position	0	-2147483648	2147483647	encoder counts	Yes

*Encoder:* Select “Main” for the SGDh encoder, or “External” for a device connected to the encoder input on the I/O connector of the MP940. See the MP940 Machine Controller Hardware Manual for details about external encoders.

*Position:* A number or variable in user units representing the new position.

#### Required Conditions

The selected encoder must not be moving when this block is executed.

#### Additional Information

The virtual encoder will be defined along with the real encoder when External is selected.

### 3.2.6 END



#### Definition

This block identifies the end of a program or subroutine. The block is automatically placed on the programming canvas when a new program or subroutine is created. There is no END icon on the block toolbar.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
None	N/A	N/A	N/A	N/A	N/A

#### Required Conditions

All programs and subroutines must have an END block.

### 3.2.7 GEAR



#### Definition

This block enables or disables the gearing function, based on the property settings. When the gearing function is disabled, the slave continues to run at the speed at which it was running before disengagement. Use the STOP block to stop the slave after disengagement. This guarantees that the slave is stopped.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Enabled	True	True	False	Configuration	No

*Enabled:* The gearing function is enabled or disabled depending on this setting.

#### Required Conditions

The servo must have been previously enabled using the SERVO block.

#### Additional Information

When gearing is enabled, the accel and decel values are set to infinite, and the speed value is set to maximum speed. This allows the position of the servo to track the external encoder exactly. If the external encoder is already moving when gearing is engaged, the CHANGE DYNAMICS block can be implemented immediately after enabling the gearing function to set accel and decel properties to minimize the shock on the slave. Note, however, this reduces the following response; the accel and decel must be again set to infinite when the slave has reached the desired speed.

## 3.2.8 GEAR RATIO



### Definition

This block defines the gear ratio between the main and external axes. The gear ratio can be changed at any time whether the servo is in motion or not. To make adjustments to the main/external relationship, use the SLAVE OFFSET block.

### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Main	1	-32768	32767	numerator	Yes
External	1	-32768	32767	denominator	Yes

The main and external properties define the ratio between the two axes. The gear ratio in MW+ takes into account the physical gearboxes on both the master and slave axes as entered in the system properties:

- $\text{TrueExternal} = \text{Main\_Output} * \text{External\_InputSide} * \text{GEAR\_RATIO\_External}$ .
- $\text{TrueMain} = \text{Main\_Input} * \text{External\_OutputSide} * \text{GEAR\_RATIO\_Main}$ .
- Main input/output is set in the MP940 System Properties.
- External input/output is set in the External Encoder System Properties.
- GEAR RATIO External/Main is set in the GEAR RATIO block.
- Example: If the motor has a 5:1 gearbox and the external encoder has a 2:1 gearbox and the master/slave is to be synchronized 1:1, then to synchronize, enter an electronic gear ratio of 5:2.

### Required Conditions

None

### 3.2.9 HOME AXIS



#### Definition

This block locates the home position within the positioning range. This block moves the servo in search of the home input at a motion profile as specified by the properties of this block. The home input can be either physical or virtual, meaning the home input can be forced on by using a bit type variable. This is useful when homing only to the “C” channel is required. If a timeout value is specified, the program counter advances to the block connected to the false port if the home routine has not been completed or the timeout value has elapsed.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Acceleration	N/A	32767	0	ms to rated speed	Yes
Approach Velocity	N/A	-200.00	200.00	% of rated speed	Yes
Creep Velocity	N/A	-200.00	200.00	% of rated speed	Yes
Deceleration	N/A	32767	0	ms from rated speed	Yes
Home Switch Input	N/A	N/A	N/A	Discrete input or bit variable	No
Homing Direction	Positive	Negative	Positive	N/A	No
Timeout	0	0	32767	Seconds	Yes

*Acceleration:* The acceleration rate of the servo in user units/s<sup>2</sup>.

*Approach Velocity:*

The speed at which the servo travels in user units/s while seeking the home input.

*Creep Velocity:* The speed at which the servo travels in user units/s while seeking the “C” channel.

*Deceleration:* The deceleration rate of the servo in user units/s<sup>2</sup>.

*Home Switch Input:*

The physical or virtual input that causes deceleration and seeking of the “C” channel, or index pulse.

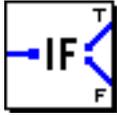
*Homing Direction:* The direction of travel while seeking the home input.

*Timeout:* The amount of time that the block will wait for the servo to find the home input and “C” channel before stopping the servo and passing control to the block connected to the false port. If an error or alarm occurred during homing, motion stops and the timeout results.

**Required Conditions**

The servo must have been previously enabled using the SERVO block.

### 3.2.10 IF EVENT



#### Definition

This block evaluates an expression (event). This could be a physical input, a variable, or mathematical expression. If the expression evaluates to true, the program continues to the block connected to the T (True) output port, otherwise the program continues to the block connected to the F (False) output port. See the Expression Builder section for details on building complex calculations.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Event	N/A	N/A	N/A	N/A	Yes

*Event:* An expression which evaluates to either true or false. The event may contain up to eight sub-expressions.

#### Required Conditions

1. None

#### Additional Information

The IF EVENT and IF FAULT blocks are the only blocks that do not hold up program execution until the next scan to execute the next block, provided that the next block is “down stream.” This means that when an event is evaluated, the down stream block is also executed in the *same* scan.

### 3.2.11 IF FAULT



#### Definition

This block evaluates whether or not there is a fault condition by evaluating the internal system variable `mError_CPU`. If there is a fault, the program continues to the block connected to the T (True) output port, otherwise the program continues to the block connected to the F (False) output port.

#### Properties

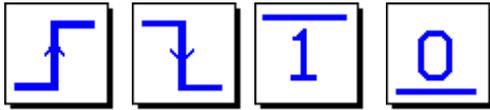
Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
None	N/A	N/A	N/A	N/A	N/A

*None:* There are no properties to set up in this block.

#### Required Conditions

None

### 3.2.12 INPUT



#### Definition

This block waits for an input to change state or be at a certain level based on the condition property. The program counter does not advance to the next block until the input has gone through the specified pattern. If “Rising edge” is selected, this block waits for the pulse to transition from low to high. If “Falling edge” is selected, it waits for the input to change from high to low. If “Logic one” is selected, the block advances the program counter when the specified input is high. If the input is already high when this block executes, the program counter is advanced. If “Logic zero” is selected, the block advances the program counter when the specified input is low. If the input is already low when this block executes, the program counter is advanced.

Tip: The input property can be any logical expression.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Condition	Rising edge	Logic one	Rising edge	N/A	No
Input	Nothing	Any bit type variable	Any bit type variable	N/A	Yes

*Condition:* Select Rising edge, Falling edge, Logic one, or Logic zero.

*Input:* Any bit type physical input, user-defined bit type variable, or logical expression.

#### Required Conditions

None

### 3.2.13 JOG AXIS



#### Definition

This block jogs the axis. Specify the acceleration, deceleration, and velocity. The servo continues to rotate at the specified speed until another JOG AXIS block with zero speed, or the STOP block is executed.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Acceleration	N/A	32767	0	ms to rated speed	Yes
Deceleration	N/A	32767	0	ms from rated speed	Yes
Velocity	N/A	-200.00	200.00	% of rated speed	Yes

*Acceleration:* The acceleration rate in units/s<sup>2</sup>.

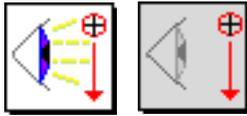
*Deceleration:* The deceleration rate in units/s<sup>2</sup>.

*Velocity:* The jog velocity in units/s.

#### Required Conditions

The servo must have been previously enabled using the SERVO block.

### 3.2.14 LATCH



#### Definition

This block enables or disables the registration latch capture function. It controls when an axis position can be latched, and when it cannot. If necessary, use IF EVENT or INPUT blocks to check for the minimum and maximum realistic latch positions to minimize the chance of latching false registration marks on material that may have other markings. When a latch occurs, the latch function is automatically disabled to avoid receiving multiple latches. Check mState\_Latch\_Main or mState\_Latch\_External to determine if the latch was captured. The axis position at the time of the latch occurrence is stored in mPosition\_Latch\_External. Note: If using the LATCH TARGET block, this block is not required to set the latch window, the LATCH TARGET block contains these settings.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Enabled	True	False	True	N/A	No
Latch Axis	Main	External	Main	N/A	No

*Enabled:* Select whether the latch function is enabled or disabled.

*Latch Axis:* Select whether to latch the main or external encoder position when the registration latch input signal is received for the specified axis.

#### Required Conditions

When using the main registration latch (External Latch Signal 3 - EXT3), the latch must be enabled by parameter setting in the SGD amplifier. Typically, this is accomplished by setting parameter Pn511.3 to “6”. Regardless of the discrete input selected, the state of the latch is stored in mState\_Main\_Latch. MotionWorks+™ default value for Pn511.3 is “6”. Therefore, under normal circumstances, no special configuration is necessary. See the following table for further details.

#### Pn511.3 Settings

Setting	Description	MW+ I/O Name
0 to 3	Sets the signal to always disabled.	—

### Pn511.3 Settings

Setting	Description	MW+ I/O Name
4	Active low input signal from S14 (CN1-44)	Sigma_EXT1
5	Active low input signal from SI5 (CN1-45)	Sigma_EXT2
6	Active low input signal from SI6 (CN1-46)	Sigma_Latch_Input
7	Sets the signal to always enabled	—
8	Sets the signal to always disabled	—
D	Active high input signal from SI4 (CN1-44)	Sigma_EXT1
E	Active high input signal from SI5 (CN1-45)	Sigma_EXT2
F	Active high input signal from SI6 (CN1-46)	Sigma_Latch_Input
9 to F	Sets the signal to always disabled	—

### 3.2.15 LATCH TARGET



#### Definition

The LATCH TARGET block synchronizes a move with a latch input. Specify the move requirements as described in the properties table below. The latch input must occur while the axis is between the *latch start distance* and the *latch finish distance*. Upon successful latch input, the main axis decelerates to a stop at *target distance* + `mPosition_Latch_Main`. If the latch does not occur within the window, the axis stops at the default distance. Note: This block handles the enabling and disabling of the latch; there is no need to use the LATCH block.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Acceleration	N/A	32767	0	ms to rated speed	Yes
Deceleration	N/A	32767	0	ms from rated speed	Yes
Default Distance	N/A	-2147483648	2147483647	Encoder counts	Yes
Latch Finish Distance	N/A	-2147483648	2147483647	Encoder counts	Yes
Latch Start Distance	N/A	-2147483648	2147483647	Encoder counts	Yes
Target Distance	N/A	-2147483648	2147483647	Encoder counts	Yes
Velocity	N/A	0	200.00	% of rated speed	Yes
Wait for Completion	True	False	True	Configuration	No

*Acceleration:* The acceleration of the axis in user units/s<sup>2</sup>.

*Deceleration:* The deceleration of the axis in user units/s<sup>2</sup>.

*Default Distance:* The final target at which the axis stops if a latch input did not occur within the tolerance window. This is a relative distance from the start of the move.

*Latch Finish Distance:*

The final position at which the latch is monitored. If the latch occurs after this position, it is ignored, and the axis stops at the default distance.

*Latch Start Distance:*

The first position at which the latch is monitored. If the latch occurs before this position, it is ignored, and the axis stops at the default distance.

*Target Distance:* The distance from where the latch occurs to the desired stop location. When a valid latch is received, the axis moves this additional distance from the latch location. This supersedes the default distance.

*Velocity:* The velocity of the axis in user units/s.

*Wait for Completion:*

If wait for completion is enabled, the program counter does not advance to the next block until the axis is either stopped at the target *distance* or the *default distance*.

**Required Conditions**

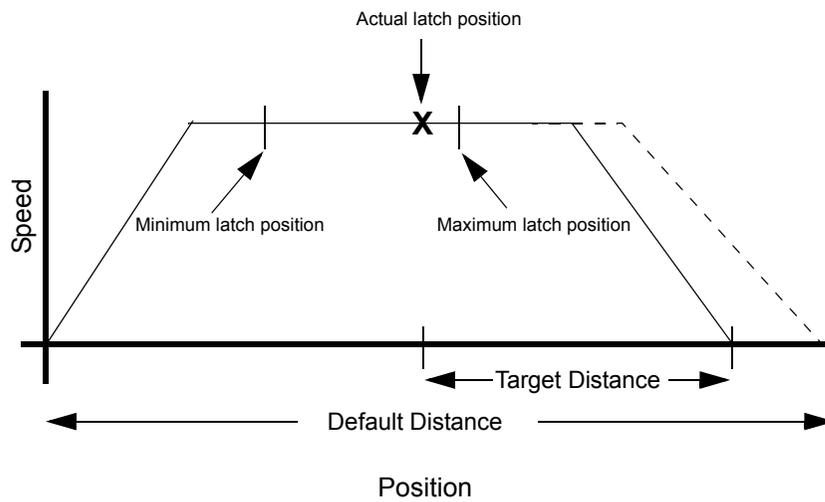
When using the main registration latch (External Latch Signal 3 - EXT3), the latch must be enabled by setting the following parameter in the SGDh amplifier. Typically, this is accomplished by setting parameter Pn511 digit 3 to “6”. Regardless of the discrete input selected, the state of the latch is stored in mState\_Main\_Latch. MotionWorks+™ default value for Pn511.3 is “6”. Therefore, under normal circumstances, no special configuration is necessary. See the table below for further details.

**Pn511.3 Settings**

Setting	Description	MW+ I/O Name
0 to 3	Sets the signal to always disabled.	—
4	Active low input signal from S14 (CN1-44)	Sigma_EXT1
5	Active low input signal from SI5 (CN1-45)	Sigma_EXT2
6	Active low input signal from SI6 (CN1-46)	Sigma_Latch_Input
7	Sets the signal to always enabled	—
8	Sets the signal to always disabled	—
D	Active high input signal from SI4 (CN1-44)	Sigma_EXT1
E	Active high input signal from SI5 (CN1-45)	Sigma_EXT2
F	Active high input signal from SI6 (CN1-46)	Sigma_Latch_Input
9 to F	Sets the signal to always disabled	—

## Example

The following example demonstrates a program that uses the LATCH TARGET block.



An application suitable for this function is a feeder that must start and stop to allow another axis to make a cut at a specific location on product that may have variation or slippage.

### 3.2.16 LAUNCH PROGRAM



#### Definition

This block starts or restarts execution of a program after it has been stopped by a SUSPEND PROGRAM block. This block allows the user to control the execution of programs to ensure proper error recovery. Specify the block number at which to launch a program.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Block ID	N/A	N/A	N/A	N/A	No
Program	N/A	N/A	N/A	N/A	No

*Block ID:* Specify the block ID number at which to start/restart the program.

*Program:* All of the available programs are shown in a list. Specify the program to be launched.

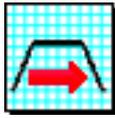
#### Required Conditions

1. The specified program must not already be executing.
2. Only programs can be launched, not subroutines.

#### Additional Information

Do not launch a program that is already executing, as unpredictable operation will result. (i.e. Block counter will be switched.)

### 3.2.17 MOVE AXIS



#### Definition

This block performs motion to a specified location at a profile described by the properties of this block. Motion begins immediately after the block has executed. If the program must wait at this block until motion is completed, set the *wait for completion* property to true.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Acceleration	N/A	32767	0	ms to rated speed	Yes
Deceleration	N/A	32767	0	ms from rated speed	Yes
Move Type	Absolute	N/A	N/A	Absolute, Relative, Rotary	No
Position	N/A	-2147483648	2147483647	Encoder counts	Yes
Velocity	N/A	0	200.00	% of rated speed	Yes
Wait for Completion	True	False	True	Configuration	No

*Acceleration:* The acceleration rate of the axis in user units/s<sup>2</sup>.

*Deceleration:* The deceleration rate of the axis in user units/s<sup>2</sup>.

*Move Type:* **Absolute:** The value or expression entered in the *position* property is made to be the new commanded position of the axis. After the move is complete, the axis is at this position.

**Relative:** The value or expression entered in the position property is added to or subtracted from the current commanded position of the servo. After the move is complete, the servo will be this distance from where it was at the beginning of the move.

**Rotary:** Specifies an axis with a limited positioning range, but the axis can travel in one direction indefinitely. The encoder values are modularized to repeat within a cycle. Positioning values can be either relative or absolute, but they are modularized to fit within one cycle.

*Position:* Specifies the magnitude of the move in user units.

*Velocity:* The speed of the servo in user units/s.

*Wait for Completion:*

When enabled, *wait for completion* freezes the program counter until the servo reaches the *position*. The move is considered complete when the encoder is within a specific range of the target. The system variable for this window is `sPosition_CompletionWindow`. The move can also be monitored via `mPositionComplete`.

### **Required Conditions**

The servo must have been previously enabled using the SERVO block.

### **Additional Information**

Motion is considered complete when the actual position of the axis is within the `mPosition_CompletionWindow` system variable. If the `mPosition_CompletionWindow` is set very wide, it may appear that this block does not wait for completion.

### 3.2.18 PLS



#### Definition

The PLS (programmable limit switch) block sets outputs or variables accordingly based on the *Encoder* property. The *Encoder* property is typically the main axis position; however, any variable can be used. A maximum of eight outputs can be programmed per block. To control more outputs, connect PLS blocks together or assign additional programs for the PLS function.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Encoder	N/A	-2147473648	2147483647	encoder counts	No
Maximum Position	0	-2147483648	2147483647	encoder counts	Yes
Minimum Position	0	-2147483648	2147483647	encoder counts	Yes
Output #	N/A	N/A	N/A	N/A	No
State	Off	Off	On	N/A	No

*Encoder:* The encoder/variable that determines the state of the outputs. Typically this is the main axis position.

*Maximum Position:* The greatest value of *Encoder* which sets the state of the output according to the *Output* property.

*Minimum Position:* The smallest value of *Encoder* which sets the state of the output according to the *Output* property.

*Output #:* The output to be set when *Encoder* is within the range.

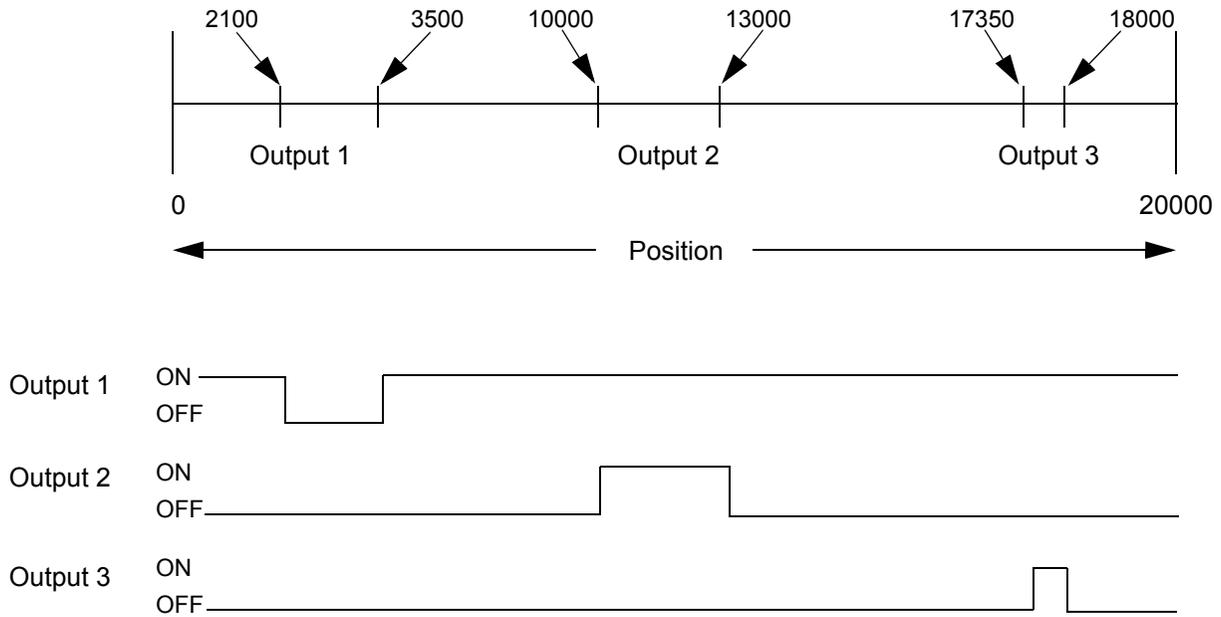
*State:* Select either ON or OFF. The output is set to this state when *Encoder* is within the range, otherwise it is set to the opposite state.

#### Required Conditions

None

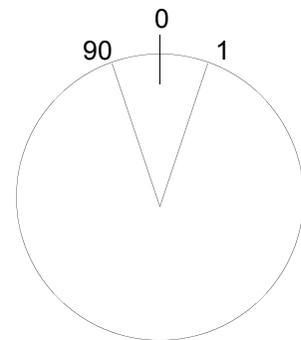
#### Example

The following example demonstrates a program that uses the PLS block.



**Making the PLS Work When the Output Must be ON Across the Modulus Rollover**

Use the entries in the PLS block. One will cover the region from 0 to 1, and the other entry should cover 90 to 100 (Assuming a machine cycle of 100). Next, “OR” the two conditions in a SET VARIABLE block.



### 3.2.19 RESET FAULT



#### Definition

This block is used to reset axis faults. Some faults are not resettable without power cycle.

#### Properties

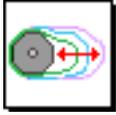
Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
None	N/A	N/A	N/A	N/A	—

*None:* There are no properties to set for this block.

#### Required Conditions

None

### 3.2.20 SCALE CAM



#### Definition

This block scales the slave position by the amount specified in the scale factor. Effective scaling of the cam profile does not require the SCALE CAM block to run at every scan. The cam profile is always scaled by the scale factor; this block only changes the scale value.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Scale Factor	100	-327.68	327.67	%	Yes

*Scale Factor:* Each value in the *Position Data Table* is multiplied by the scale factor. More specifically, when the slave position is determined based on the current master location, the resulting slave value is multiplied by the scale factor.

#### Required Conditions

None

### 3.2.21 SERVO



#### Definition

This block enables or disables the ability of the servo amplifier to apply holding torque to the motor. If the servo is not on when this block executes, it assigns the commanded position equal to the actual position. If the servo is already enabled when this block executes, the commanded position is not changed.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Enabled	True	False	True	Status bit	No

*Enabled:* Specify the servo state. True means the motor has holding torque; false means it does not.

#### Required Conditions

None

### 3.2.22 SET VARIABLE



#### Definition

This block computes an expression and stores the resulting value in a variable or parameter. The expression can be a simple numeric value, or a complex calculation. For example, Boolean type variables can be assigned either a “1” or “0.” Alternatively, the keywords TRUE and FALSE or ON and OFF can be used. A maximum of ten variables are settable within one block.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Expression	N/A	N/A	N/A	N/A	Yes
Variable	N/A	N/A	N/A	N/A	Yes

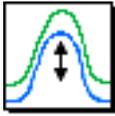
*Expression:* The entity stored in the variable when solved.

*Variable:* The entity to be assigned a value by the expression.

#### Required Conditions

None

### 3.2.23 SLAVE OFFSET



#### Definition

This block changes the slave offset by the specified amount. If gearing is enabled, it advances/retards the slave in relation to the master. If camming is enabled, it adds the value in the *Offset* property to each point in the cam table. If neither camming nor gearing is enabled, this block simply performs the specified move in the specified time. The offset move profile is a modified sine of the value in the *Offset* property over the time period specified in the *Time* property.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Offset	0	-2147483648	2147483647	encoder counts	Yes
Time	0	0	32767	ms	Yes

*Offset:* The amount in user units of the slave to advance or retard the slave.

*Time:* Time in milliseconds to complete the SLAVE OFFSET block.

#### Required Conditions

None

### 3.2.24 START



#### Definition

This block designates the start of a program or subroutine. This block is automatically placed on the programming canvas when the new program or subroutine is created. (There is no START icon on the block toolbar.)

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
None	N/A	N/A	N/A	N/A	N/A

*None:* There are no properties to set for this block.

#### Required Conditions

None

### 3.2.25 STOP MOTION



#### Definition

This block stops motion at the specified deceleration rate. Any previously established motion profile will be canceled. For example, a move to an absolute or relative position will not be completed; if homing is in progress it will be canceled; and if gearing or camming was engaged, it will be disengaged. The servo will remain enabled after deceleration to a stop. If a temporary stop is required, use the CHANGE DYNAMICS block to set the velocity to zero.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Deceleration	N/A	32767	0	ms from rated speed	Yes

*Deceleration:* The deceleration rate of the servo in user units/s<sup>2</sup>.

#### Required Conditions

None

### 3.2.26 SUSPEND PROGRAM



#### Definition

This block suspends the execution of a program. Programs can be restarted by using the LAUNCH PROGRAM block.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Program	N/A	1	8	N/A	No

*Program:* Specifies which program number to suspend.

#### Required Conditions

None

### 3.2.27 TIMER



#### Definition

This block waits a specified amount of time before advancing the program counter to the block connected to the T (True) port. The program counter advances to the block connected to the F (False) output port until the timeout value is reached. The timeout property can either be a discrete value or a mathematical expression. The timeout resolution is to the nearest 10ms.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Timeout	0	0	32767	ms	Yes

*Timeout:* Specify the amount of time to wait in milliseconds.

#### Required Conditions

None

### 3.2.28 TORQUE



#### Definition

This block sets the axis in torque mode. When the axis is in this mode, the controller is only concerned with applying a specific amount of torque, and not concerned with attaining a specific speed or position. If there is not adequate resistance to motion, the axis may achieve its maximum speed or the specified speed limit. If the load drives the servo faster than its speed limit, the servo regenerates power to control the speed of the load.

#### Properties

Property	Default	Minimum Value	Maximum Value	Base Units	Run Time
Speed Limit	0	0	200.00	% of rated speed	Yes
Torque	0	-32767	32767	% of rated torque x 100	Yes

#### Required Conditions

Pn000 digit 1 in the servo amplifier parameters must be set to “9” and Pn0002 digits 0 and 1 must be set to “1”. These are the defaults in the SGDH configuration.

**Notes**

## Appendix A Operational Examples

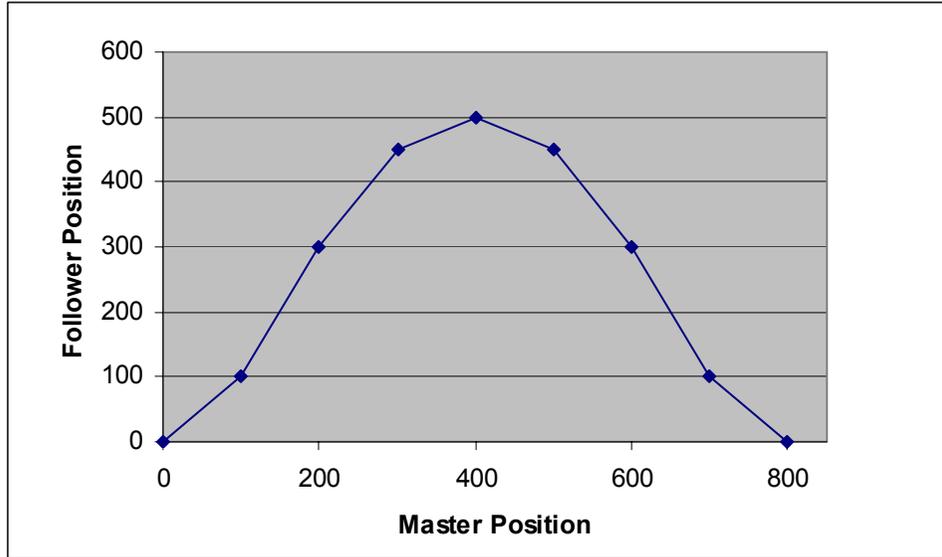
### Example Calculating Cam Tables from within a MotionWorks+™ program

The .cdd table (cam points table) can be calculated within a MotionWorks+™ program. The format of the .cdd table is shown below.

**Table Format**

<b>MW™ Table Syntax</b>	<b>Value</b>	<b>Description</b>	<b>MP940 Address</b>
MyTable[0]	9	Number of Master/Slave Pairs	ML20640
MyTable[1]	0	First master position	ML20642
MyTable[2]	0	First slave position	ML20644
MyTable[3]	100	Second master position	ML20646
MyTable[4]	100	Second slave position	ML20648
MyTable[5]	200	Third master position	ML20650
MyTable[6]	300	Third slave position	ML20652
MyTable[7]	300	Fourth master position	ML20654
MyTable[8]	450	Fourth slave position	ML20656
MyTable[9]	400	Fifth master position	ML20658
MyTable[10]	500	Fifth slave position	ML20660
MyTable[11]	500	Sixth master position	ML20662
MyTable[12]	450	Sixth slave position	ML20664
MyTable[13]	600	Seventh master position	ML20666
MyTable[14]	300	Seventh slave position	ML20668
MyTable[15]	700	Eighth master position	ML20670
MyTable[16]	100	Eighth follower position	ML20672
MyTable[17]	800	Ninth master position	ML20674
MyTable[18]	0	Ninth follower position	ML20676

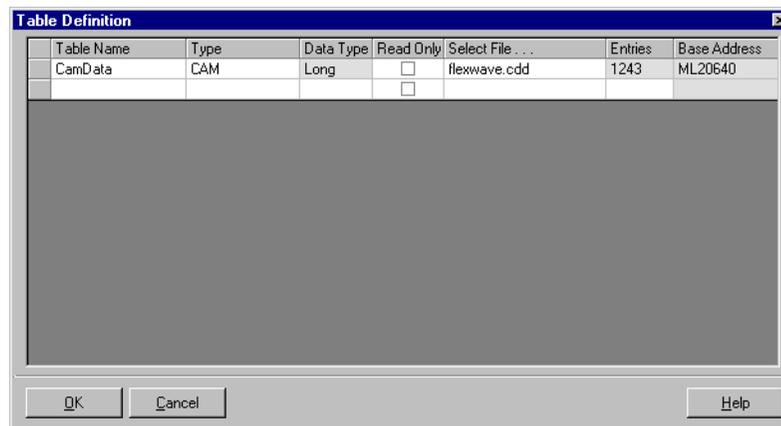
## Graphical View



## Requirements

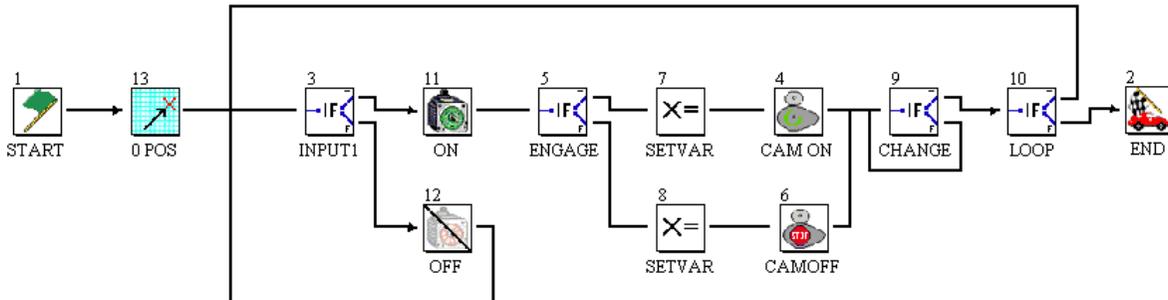
A .cdd file must be imported into the project of an appropriate data size. This determines the camming resolution.

The table must be defined as a variable table, that is, the “Read Only” checkbox must not be checked.



**Program 1: MW+™ Cam Example with Cam Table Recalculation.**

Program #1 turns the servo ON if Input #1 is ON. If Input #2 is ON, the program saves the state of Input #2, and camming mode is engaged. The program waits at block #9 until the state of Input #2 has changed.



*Program #1*

Properties - camcalc2

3 (If Event) INPUT1

{ ID }	3
Label	INPUT1
Event	Local_Input1

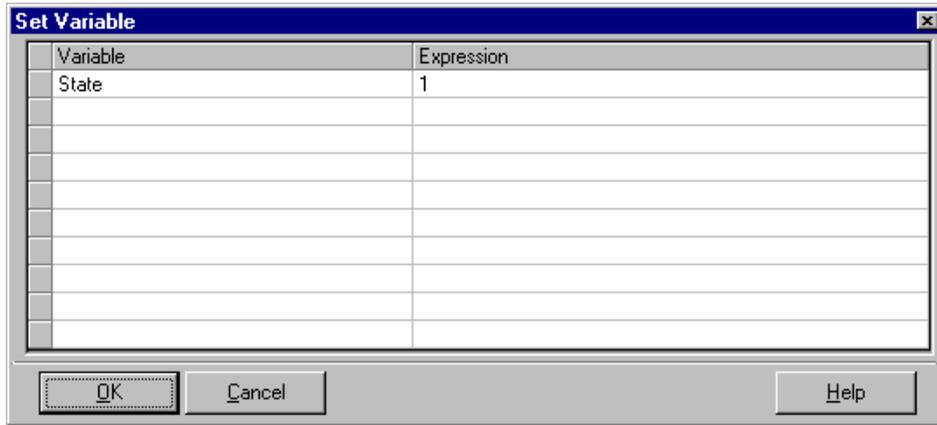
**Label**  
Returns/sets a label which is displayed below a block.

Properties - camcalc2

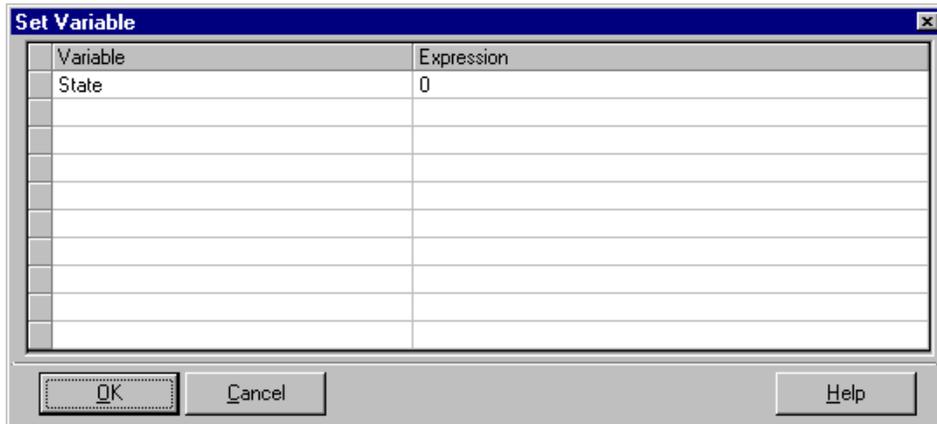
5 (If Event) ENGAGE

{ ID }	5
Label	ENGAGE
Event	Local_Input2

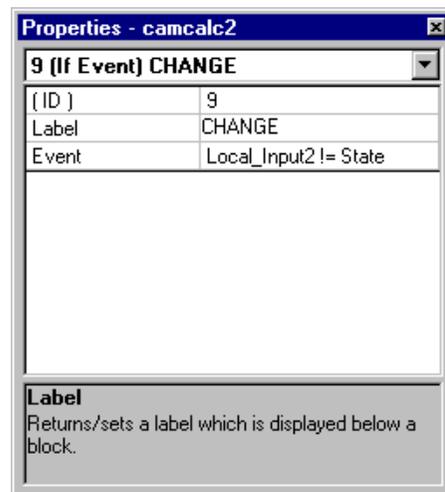
**Label**  
Returns/sets a label which is displayed below a block.



**Block #7**

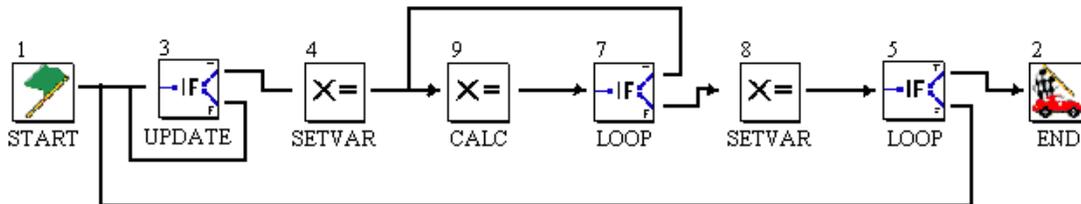


**Block #8**



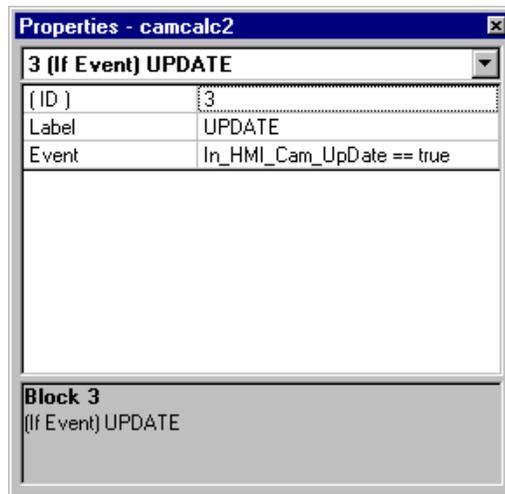
### Program #2: Cam Table Calculation

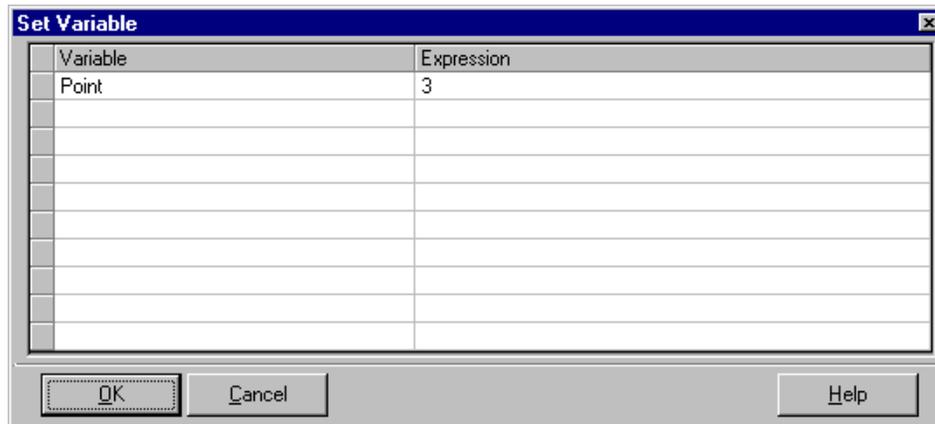
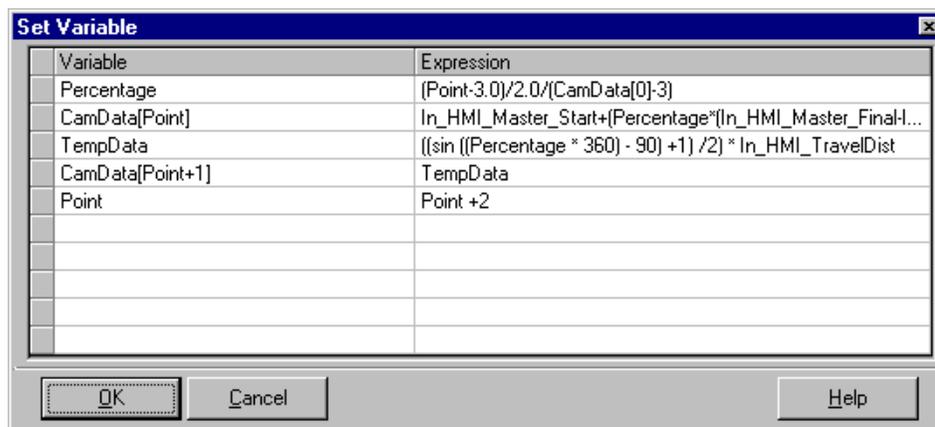
This program waits at block #3 until the bit variable “Update” becomes a “1”. The program initializes a table index variable called “Point”. Block #9 has calculations for the master and slave for all 619 cam points. Block #7 checks for the end of the cam table. Block #8 sets the final pair of cam points, and sets the “Update” variable back to “0”.



#### Program #2

Note: The cam example calculated here has a deadband at the beginning and at the end. By manipulating variables “MasterStart” and “MasterFinal”, dwell period can be shifted, and the move time can be increased or decreased.



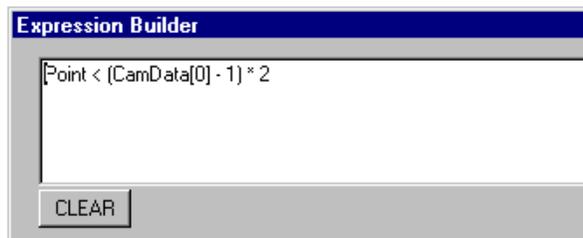
**Block #4****Block #9**

**Percentage** is a floating point number ranging from 0.0000 to 1.0000 through the cam profile. Point-3.0 subtracts off the first entry, which holds the number of master/slave pairs, and also the first pair, which is the very first position of the table, which does not change. For the percentage, only information on the pair of points is necessary. The reason for /2.0 is that the master and slave data makes the table twice as large. The reason for (CamData[0]-3) is that the percentage is only for the number of pairs in which the slave has the sine function applied to it. This is the “total number of cam points” - “first pair” - “last pair” - “first pair from sine function”. Finally, one important note is the reason for using 3.0 and 2.0 in the calculation. MotionWorks+™ follows the rules for “C” syntax. To force the result to be a floating point number, a decimal point value must be included in the equation. If not, the compiler uses longs for the temporary registers to generate the answer, and the percentage is always zero.

**CamData[Point]** is the master value of the current “Point”. MasterStart is the dwell distance at the beginning of the cycle. MasterFinal is the dwell distance at the end of the cycle. The master is always the MasterStart offset plus some percentage of the total master travel while the slave moves.

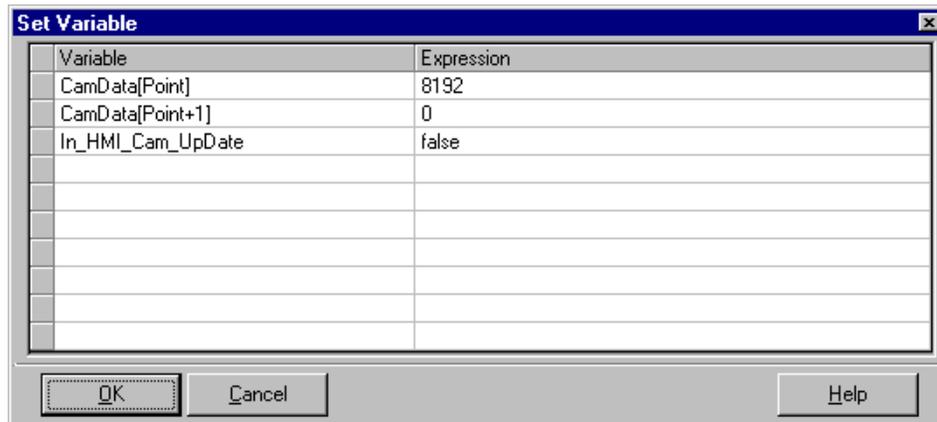
**TempData** is a floating point which is the sine of the percentage times 180. This value is scaled by the maximum travel of the slave. In the example, the number is 4096 encoder counts, or 1/2 of a revolution. This number may also be a variable, if desired. TempData is transferred to CamData[Point+1].

**Point** is incremented by two to reflect the number of cam pairs, not each individual value in the table.



*Block #7*

The loop keeps running while Point is less than the “total number of pairs” - “one pair” times 2. The reason for “times 2” is that the value of Point is doubled, to account for master and slave.



*Block #8*

**Final Notes**

From the Data Monitor Window, view MasterStart, MasterFinal, and Update. Change the MasterStart and MasterFinal values to any value between 0 and 8192, then change Update to 1. The program recalculates the table, and sets Update back to zero when complete. The table can be changed while the cam profile is running, but a drastic change will cause the slave to jump.

In less than one second the program recalculates all the cam data points.

Any combination of calculations can be implemented by this same method.

## Appendix B Standardized Template Project

This appendix illustrates a standardized template project recommended for programming the MP-940 with MotionWorks+. The template includes three main programs:

- 1) Supervisor
- 2) Manual
- 3) Automatic

In addition, eight subroutines are called from two of the main programs. These subroutines include:

- 1) Jog Forward
- 2) Jog Reverse
- 3) Homing
- 4) Indexing with Programmable Limit Switch (PLS)
- 5) Gearing
- 6) Camming
- 7) Torque
- 8) Latch with PLS

The project described in this document is meant to be used as a starting point for virtually any MP-940 MotionWorks+ application. Make use of the subroutines that are appropriate for the application and discard the subroutines that are not.

## B.1 Summary

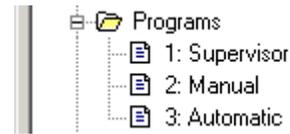
This appendix outlines the details of a template example program for an MP-940/SGDH system. In particular, it was designed with the Yaskawa demonstration (YEA Part # DEMO4700) unit in mind; however, it can be modified to suit virtually any application, and is commonly used as a starting point when programming the MP940 using MotionWorks+.

There are three main programs: Supervisor, Manual, & Automatic. The Supervisor program is the only of the three that is auto-starting. Once it starts and conditions are satisfied, it starts both the Manual & Automatic programs. In addition, there are seven subroutines: 02 Jog+, 03 Jog-, 04 Home, 05 Move, 06 Gear, 07 Cam, 08 Torque, and 09 Latch.

While the Manual & Automatic programs are running, various conditions must be met in order for them to start a subroutine. Both of these programs, as well as the configuration for the system, will be discussed. This will be completed in the order in which a program in MotionWorks+ is laid out, according to the Project Explorer window.

## B.2 Programs

The Programs folder contains the following: Supervisor, Manual, and Automatic programs.



### *Program Definition*

	Program Name	Scan Type	Active	Auto Start
1	Supervisor.vpg	High Scan	True	True
2	Manual.vpg	High Scan	True	False
3	Automatic.vpg	High Scan	True	False
4				
5				
6				
7				
8				

Buttons: OK, Cancel, Help

This template project has one program that is auto starting (Supervisor). Once that program has started and the various conditions have been satisfied, it will start the other programs. The supervisor will stop the other programs if it detects a fault, error, or other event. This programming methodology creates a solid infrastructure to build from so that each individual program does not have to monitor for errors as there is one program which does that and coordinates appropriately.

### *Program Guidelines*

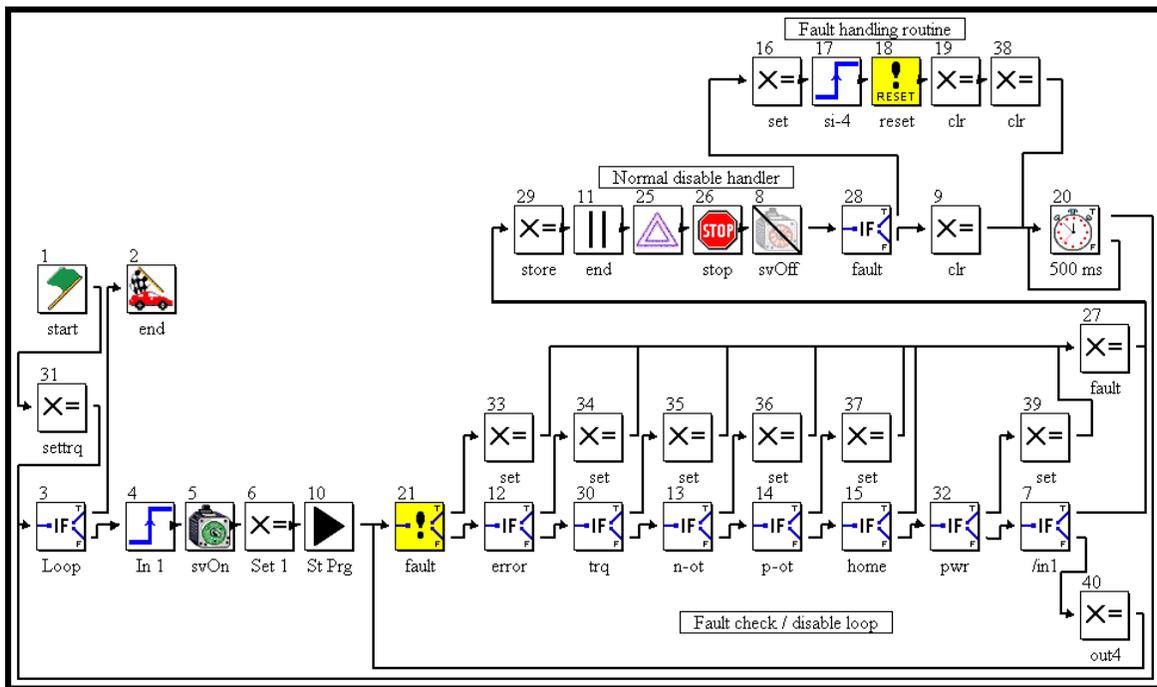
As a guideline it is recommended that each “program” contain a maximum of 64 total blocks including all subroutine blocks called from within the program (not including start and stop blocks). In this example, the Supervisor program utilizes 35 blocks, the Manual program utilizes 43 blocks, and the Automatic program utilizes 58 blocks. In addition, it is also recommended that execution of motion blocks, only be active in one program at a time. For this reason, the Manual and Automatic routines are interlocked such that they can only operate exclusively.

Motion blocks include the following:

CAM	Change Dynamics	Define Position
Gear	Home	Jog
Latch Target	Move Axis	Scale CAM
Servo Enable	Slave Offset	Stop
Torque		

The Supervisor routine utilizes the Servo ON, Stop, and Change Dynamics block, but is closely monitoring the other programs to ensure that there is no overlapping. Some precautions must still be followed with the use of Motion blocks.

### Supervisor Program



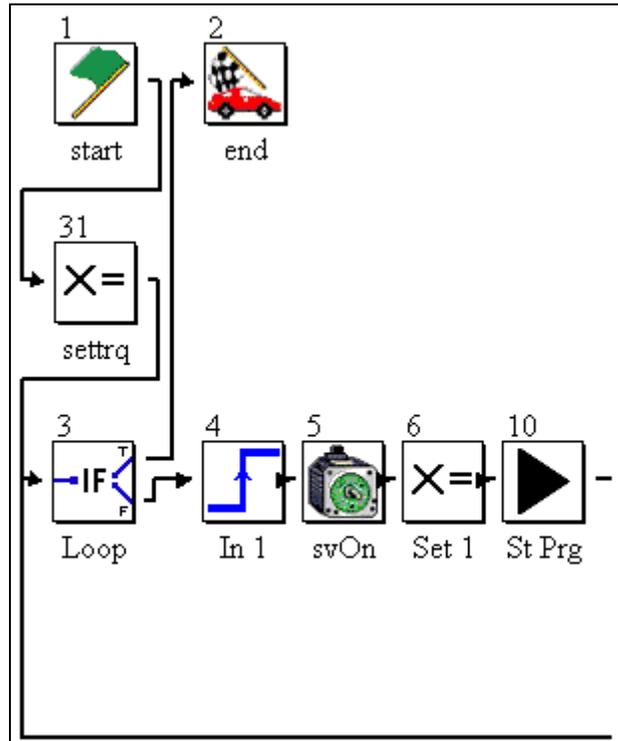
The Supervisor can be separated into four distinct sections: Start-up, Fault Detection, Disable Handler, and Fault Recovery.

### *Start-up*

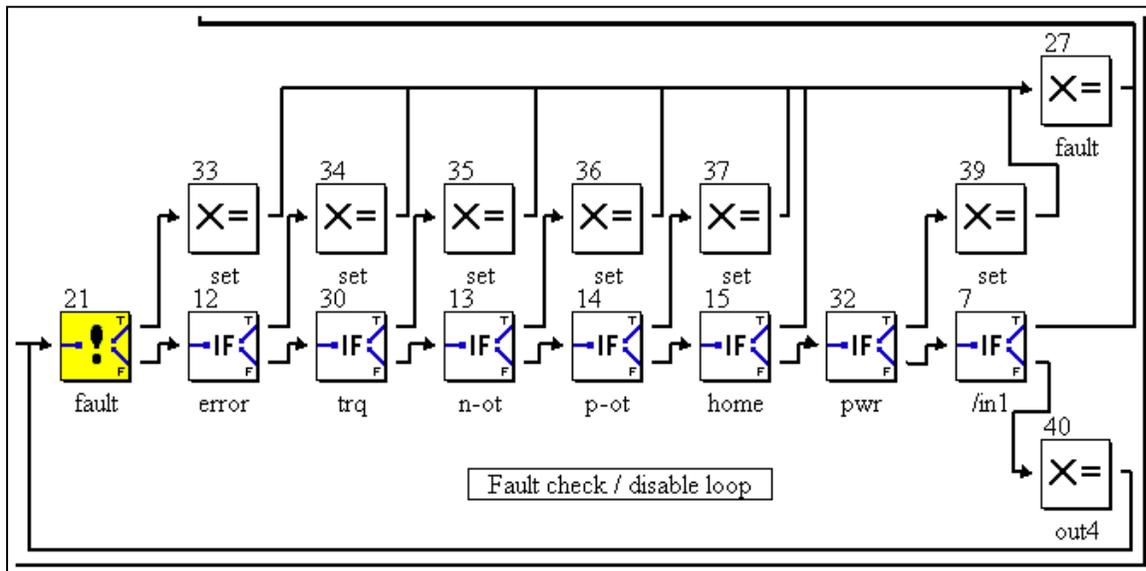
The start-up section encompasses blocks 3-6, 10, and 31. It is responsible for startup. Block 31 is executed only once (upon power up) and can be useful for setting user variables, or outputs that need to be re-initialized at power up but may be variables themselves. Block 3 has the condition “False” and provides a point to loop back at the end of the flow chart, and ensures that all blocks have connections. The remaining blocks regulate a normal start up.

Block 4 must see the rising edge of Local\_Input1, this assures that input was activated to enable the system, rather than just left on all of the time. Block 5 enables the servo.

Local\_Output1 and (user variable) [SystemOk] is set in Block 6. The [System Ok] variable is used by the two other main programs as a signal that it is Ok to execute. The last block (10) starts the other main programs (Manual and Auto-

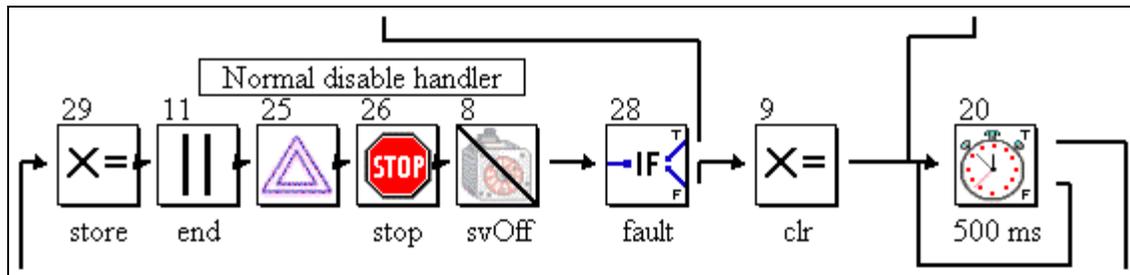


### Fault Detection



After the start-up section has been completed successfully, the next section is the Fault Detection / Disable loop. This is the section that the program will execute most frequently. All of the IF EVENT (including a special IF FAULT) blocks are designed to detect various faults and errors in the system. When an error is detected, the next block executed is a SET VARIABLE block that sets an internal flag to trap the type of error that occurred and stores pertinent data. Lastly, the SET VARIABLE block (27) is executed, which sets an internal fault flag and clears the SystemOk flag. Execution then continues to the disable handler. Also, if the user simply turns off of the enable input (Local\_Input1), execution continues to the disable handler. The last block that deserves discussion is the SET VARIABLE block (40); this block handles the homed/homing output, discussed later in the homing subroutine. Additionally it takes care of the modality, that is to say, which mode (manual or automatic) the machine is in and setting the appropriate outputs.

### Disable Handler

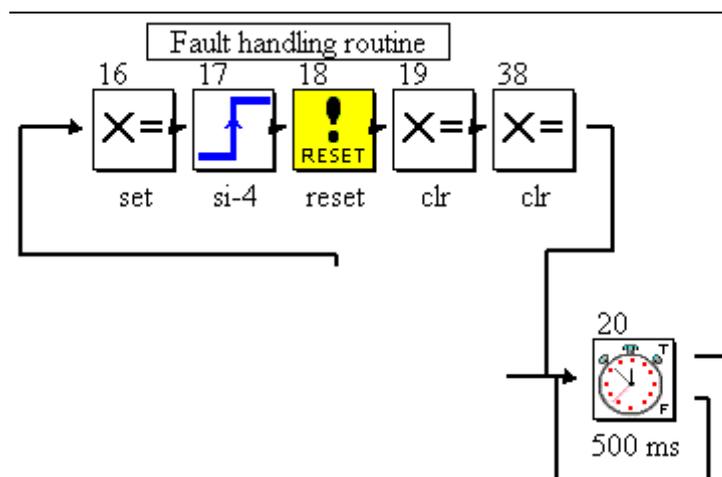


When a fault or normal disable is detected, this section of code is executed. The first block (29) stores the commanded and actual position (these may be useful in a recovery routine). Next, the other main programs are halted; the servo is commanded to go to zero speed (25), then stop (26), and lastly the SERVO Off block (8) is executed.

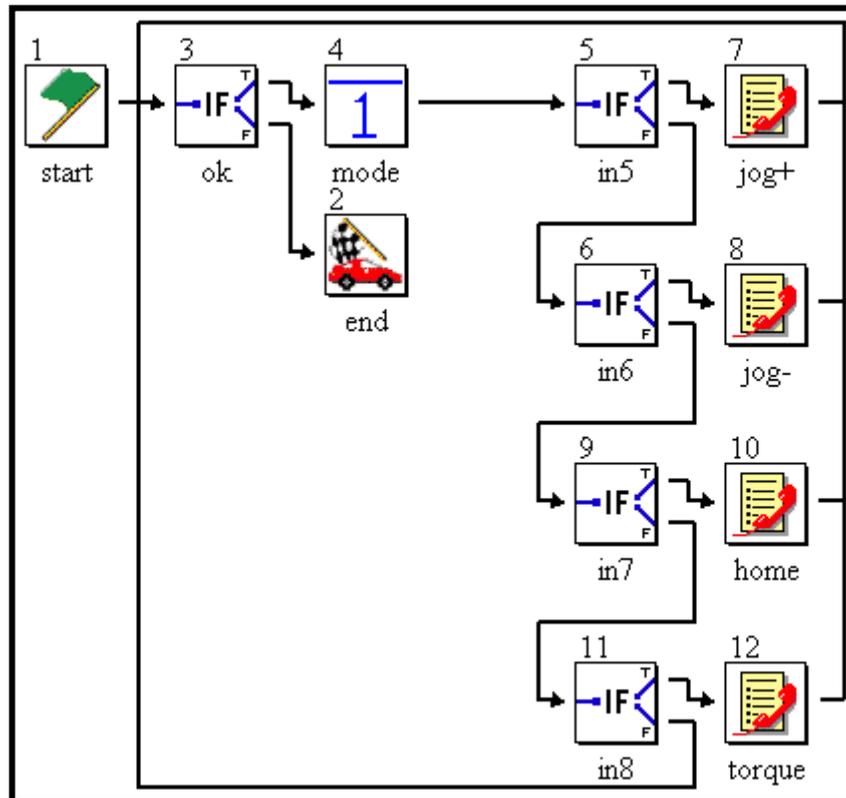
If a fault has occurred, block (28) will direct execution to the fault recovery section, otherwise block (9) is executed. Block (9) clears all of the outputs that may have been inadvertently left on. The Disable handler and the fault recovery routine both make use of Block (20). It is simply a timer that ensures time for recovery before attempting a restart. After block (20) execution continues back to block (3).

### Fault Recovery

Once it has been determined that a fault occurred and the appropriate blocks have executed, the program ends up in the fault recovery section. Block (16) sets an output to indicate that a fault has occurred (in the case of the demo box, it actually sets all eight outputs). INPUT block (17) waits to see the rising edge transition of SGDH input SI-4 (coincidentally, the Servo Alarm Reset input when the servo amplifier is used alone). RESET FAULT block (18) will reset any servo amplifier alarm that does not require a power cycle to reset. Lastly, SET VARIABLE blocks 19 and 38 clear the alarm output and all of the internal error bits. Block (20) was discussed above in the Disable Handler section.



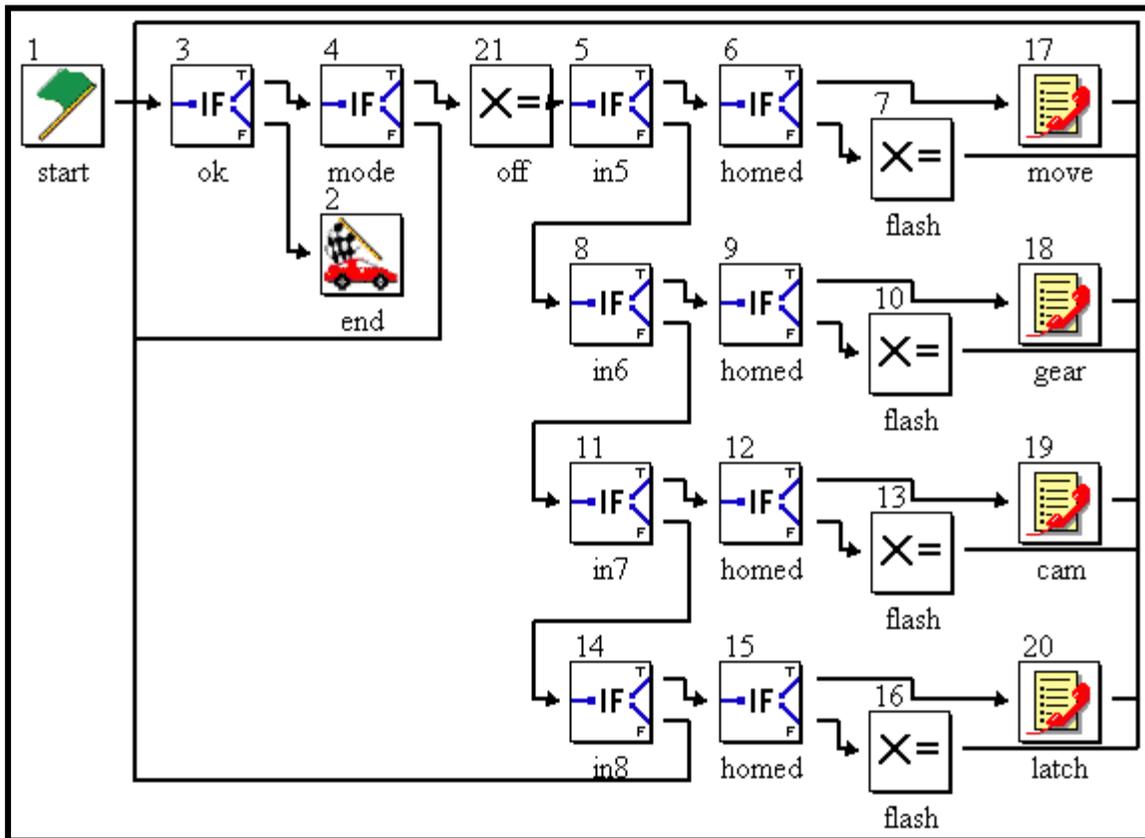
## Manual Program



Functions typically performed while in a “manual” mode are included in the manual main program. While this program is executing, it is waiting for the user to activate an input to select a subroutine program. As long as no input is selected, the program scans the blocks in order of 3, 4, 5, 6, 9, and 11 then back to 3.

Two key features in this program are important to mention. First, the IF EVENT block (3) detects if the user variable [SystemOk] is true. As long as this condition is true the program will stay running (remember, SystemOk is controlled by the supervisor program). Second, the input block (4) ensures that the user variable [ModeManual] has been selected. This is a critical interlock that guarantees that more than one main program is not attempting motion at the same time. This can be especially disruptive if the “Automatic” program is running the servo in camming mode and the “Manual” program is attempting to jog.

## Automatic Program



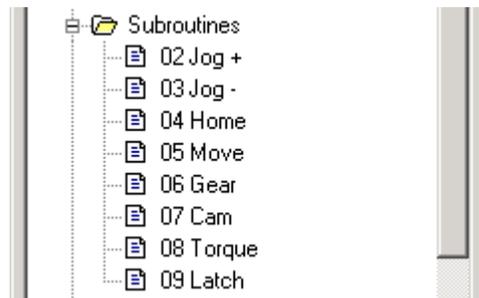
Functions typically performed while in an “automatic” mode are included in the automatic main program. While this program is executing, it is waiting for the user to activate an input to select a subroutine program. As long as no input is selected, and not in manual mode, the program scans the blocks in order of 3, 4, 21, 8, 11, 21 then back to 3.

Two key features in this program are important to mention. First, IF EVENT block (3) detects if the user variable [SystemOk] is true. As long as this condition is true the program will stay running (remember, SystemOk is controlled by the supervisor program). Second, the IF EVENT block (4) ensures that the user variable [!ModeManual] has been selected (in other words that the machine is not in manual mode, but rather automatic mode). This is a critical interlock that guarantees that more than one main program is not attempting motion at the same time. This can be especially disruptive if the “Automatic” program is running the servo in camming mode and the “Manual” program is attempting to jog.

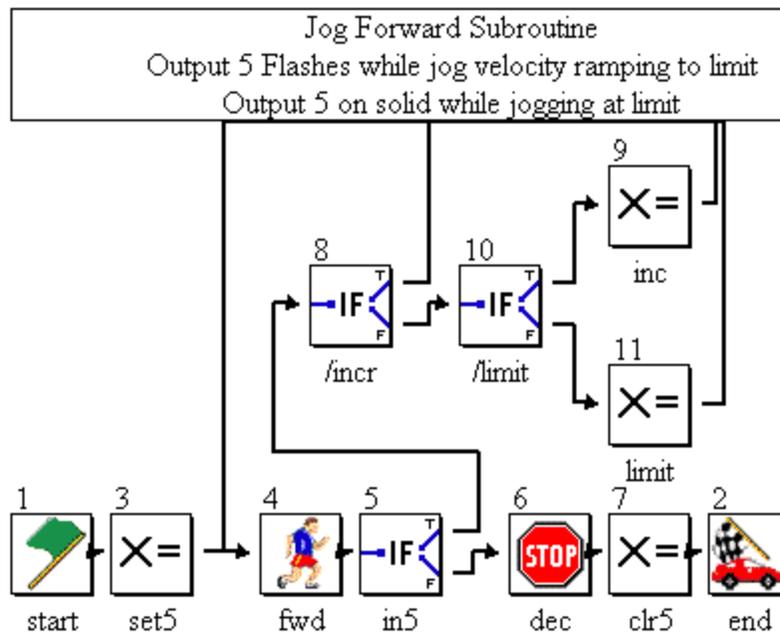
Automatic mode operation additionally requires that the system be homed prior to activating any of the automatic subroutines. If a user puts the machine in automatic mode, and attempts to execute an automatic subroutine that has not been homed, the corresponding output will flash instead of operating normally as described in “**B.3 Subroutines**” below.

### B.3 Subroutines

The Subroutines folder contains the following:



A program can have up to 62 subroutines, and subroutines may be nested up to eight deep.

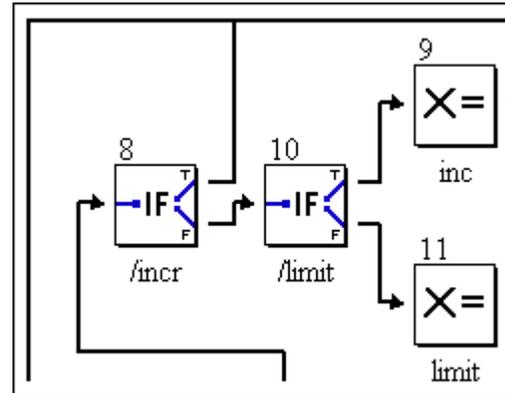


#### *Jog + Subroutine*

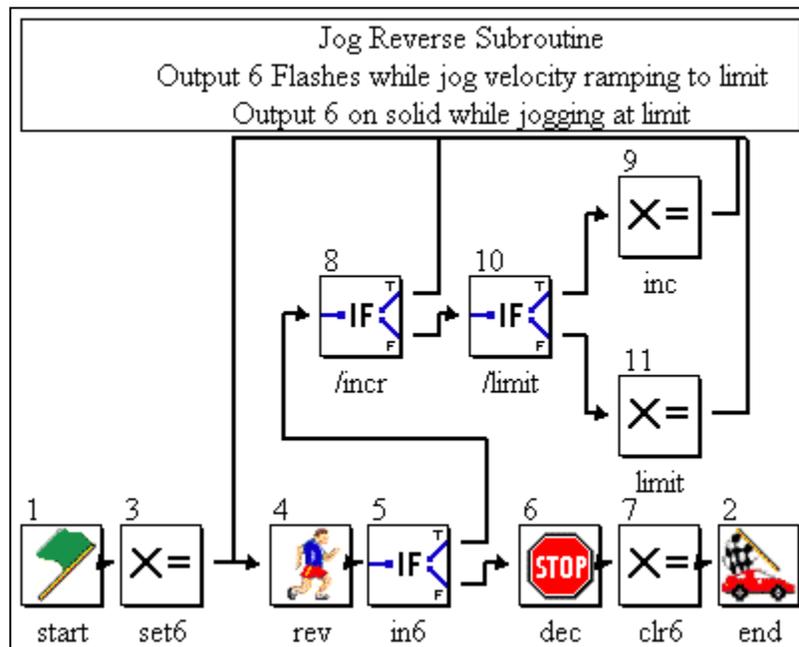
The jog + subroutine jogs the servo in the forward direction while the jog forward request (Local\_Input2) activated. When the user deactivates the jog forward request, the servo decelerates to a stop and execution is returned to the main program. LocalOutput2 is activated while the servo is in motion jogging at the limit (discussed below), otherwise the output flashes.

### Increasing Jog Speed

Blocks 8 – 11 implement a jog speed increase while the jog forward request is activated up to a user defined speed limit. Block (8), IF EVENT, allows the incremental velocity increase to occur at each user variable [VelJogIncrementTime] time period. Block (10), IF EVENT, verifies that the velocity has not reached the limit user variable [VelJogLimit]. If the limit has not been reached, SET VARIABLE block (9) increments the velocity in user variable [VelJogIncrement]. Otherwise, SET VARIABLE block (10) maintains the jog velocity at the limit.



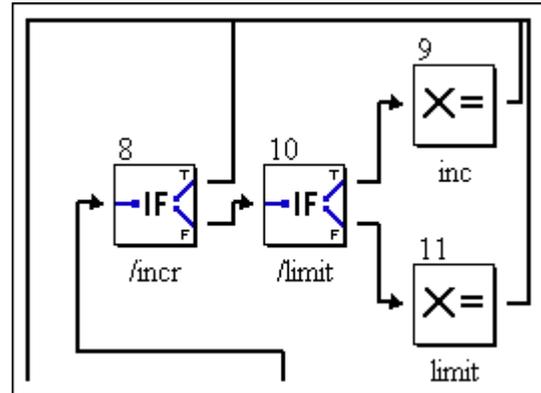
### Jog – Subroutine



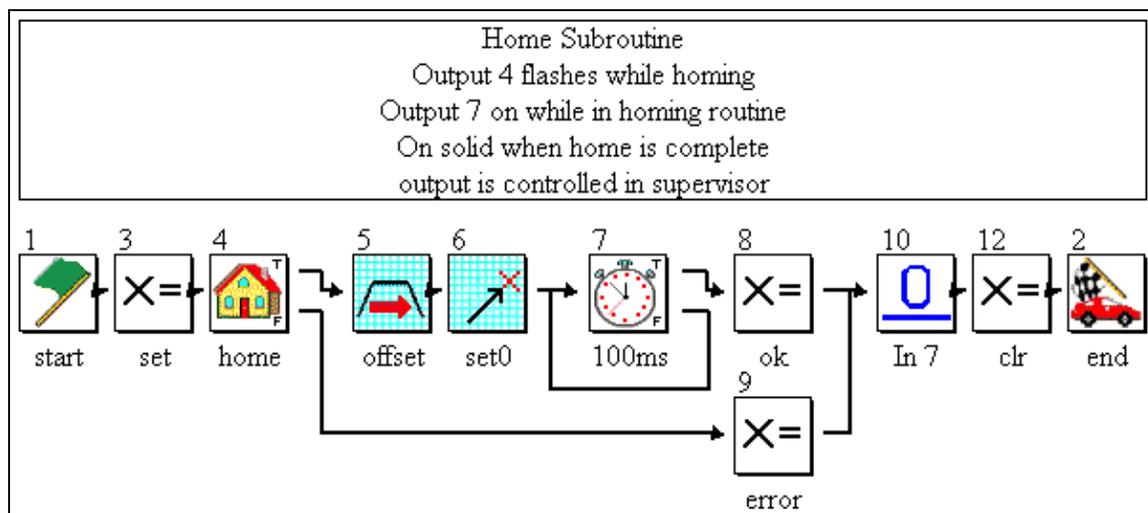
The jog – subroutine jogs the servo in the reverse direction while the jog reverse request (Local\_Input3) is activated. When the user deactivates the jog reverse request, the servo decelerates to a stop and execution is returned to the main program. LocalOutput3 is activated while the servo is in motion jogging at the limit (discussed below), otherwise the output flashes.

### Increasing Jog Speed

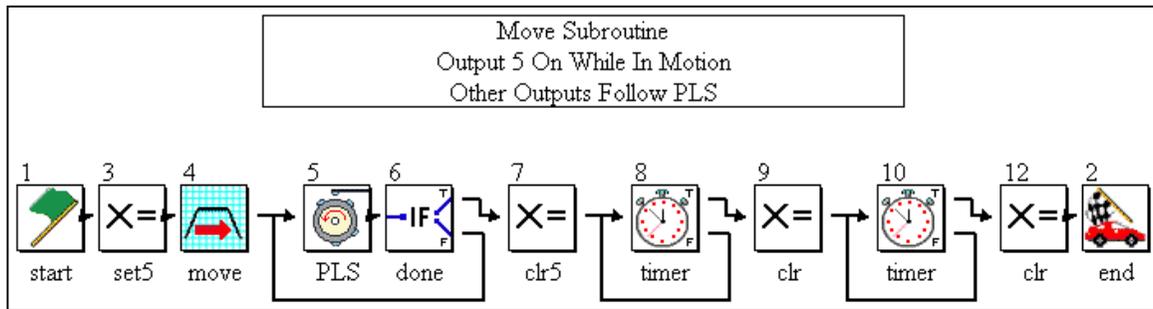
Blocks 8 – 11 implement a jog speed increase while the jog reverse request is activated up to a user defined speed limit. Block (8), IF EVENT, allows the incremental velocity increase to occur at each user variable [VelJogIncrementTime] time period. Block (10), IF EVENT, verifies that the velocity has not reached the limit set in user variable [VelJogLimit]. If the limit has not been reached, SET VARIABLE block (9) increments the velocity by user variable [VelJogIncrement]. Otherwise, the SET VARIABLE block (10) maintains the jog velocity at the limit.



### Home - Subroutine



The home subroutine takes advantage of the HOME block (4) which has built in functionality for homing to and through a deceleration switch then to the C (zero) pulse of the encoder. After that is complete, an offset move is completed, then the DEFINE POSITION block (6) re-defines the position to 0.0 (this could be a user variable). In this program, that same block also sets the external position to 0.0 (again that could be a user variable also). Blocks 3 and 8 take care of activation and deactivation of some internal user variables [Homed] and [Homing]. Block (8) sets a user error variable [ErrorHoming] which is detected elsewhere. Lastly, INPUT block (10) ensures that the user has deactivated the homing request input (Local\_Input4). If this block was not in place and the user left the home request on, the machine would home over and over.

*Move - Subroutine*

The move subroutine is simply a relative move, which incorporates a programmable limit switch (PLS) block and a couple of timers. This subroutine differs slightly from the others in that there is no event that locks it in the subroutine. Once the move is complete and the timers have timed out, the subroutine ends and execution is returned to the main program. If the input (input 5) that calls the subroutine is left on, the subroutine will be executed again (and again).

The SET VARIABLE block (3) activates output 5, and the corresponding block (7) deactivates the output making the output correspond to an “in motion” indicator. The next block, TIMER (8) creates a delay by the user variable [MoveDelay]. SET VARIABLE block (9) deactivates any outputs that may be left on from the PLS. Execution continues to another TIMER block, with the same user variable as the earlier TIMER block.

### ***Programmable Limit Switch (PLS)***

Value >	Value <	Output	On/Off
0	Position/3	Local_Output6	<input checked="" type="checkbox"/>
Position/3	Position*(2.0/3.0)	Local_Output7	<input checked="" type="checkbox"/>
Position*(2.0/3.0)	Position	Local_Output8	<input checked="" type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>

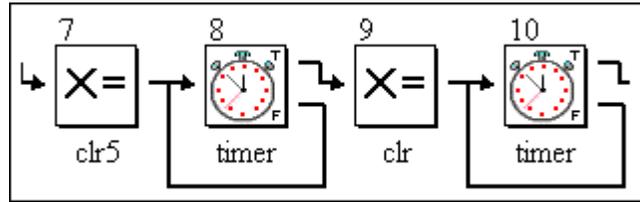
Because the controlled axis is configured in rotary mode, the system variable `mPosition_Actual` rolls over automatically at the system variable `sMachineCycle_Main`. Therefore, `mPosition_Actual` can be used as the “Encoder Variable” in the PLS block (5). For systems where rotary mode is not applicable, `mPosition_Actual` (in a user variable, such as `[PositionCapture]`) can be captured in the SET VARIABLE block (5) prior to motion. Then, to make the PLS easy to setup (i.e. elements relative to move length) the “Encoder Variable” could be configured as a calculation; `mPosition_Actual – Position-Capture`.

The PLS block implemented in this project is executed in this subroutine and therefore relies on the MOVE block (4) Wait for Completion box to be un-checked. It is used in conjunction with the IF EVENT Block (6) that checks for move completion (system variable `mPosition_Complete`). The PLS is updated while in motion.

Entries for (Value >) and (< Value) can be implemented as fixed numbers, user variables, system variables, and any combination thereof. In addition, the entries can be calculations, especially useful for applications where PLS outputs also depend on machine speed. Take care to use ‘C’ syntax in calculations. Lastly, when implementing values that go through (past) zero, it is best to split them up in to two segments `[(Value>) to 0]` and `[0 to (<Value)]` that set internal User bits. Then in a SET VARIABLE block, use those internal user bits ORed together to set the correct output.

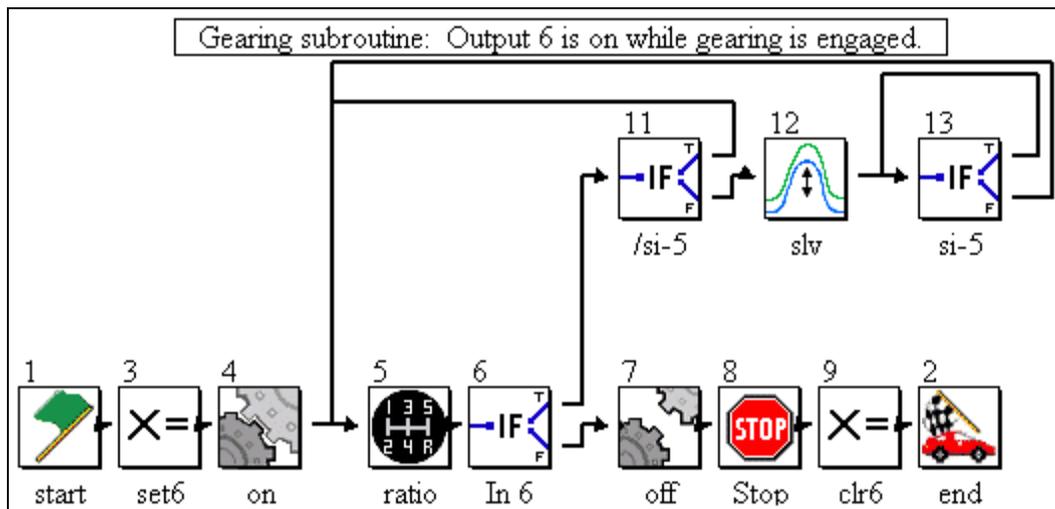
### Timer

The last few blocks (7-10) are comprised of SET VARIABLE and timer blocks. Some special attention should be brought up regarding the TIMER block. The TIMER block, while it has two OUT ports, the FALSE port should always be looped back to the IN port of the block. Time only accumulates while the block is being executed, thus the need for the loop back. The timer may take additional time to complete if items are connected between the FALSE port and the return to the TRUE port. For example, if a block is inserted between the FALSE port and the return to the TRUE port, and it takes one scan to execute that block, the amount of time will be doubled. One last thing to remember about the Timer block is that the time value in it is retentive. In other words, if the timer starts timing and for some reason gets interrupted, (program is halted, etc.) upon re-entering said timer block it will finish timing.



There is an alternative way to construct a timer using the system variable mTime, which counts milliseconds. Capture the value of mTime in a SET VARIABLE block. In the next block, an IF EVENT block, compare if  $mTime > CaptureTime + DesiredTime$ . While that condition is false, the program can monitor other events and return to the IN port of the IF EVENT block. Be careful with this method, as the timed value can vary if there is a lot of logic between the FALSE port and the return to the IN port.

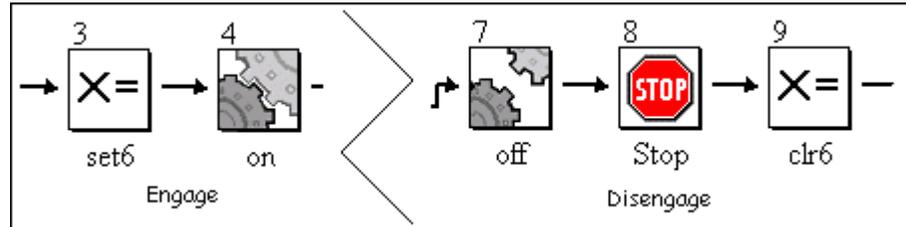
### Gear - Subroutine



The Gear subroutine is fairly straightforward, for discussion purposes it is broken down into two sections. The two sections include: Engaging/Disengaging, and Running.

### Engaging/Disengaging

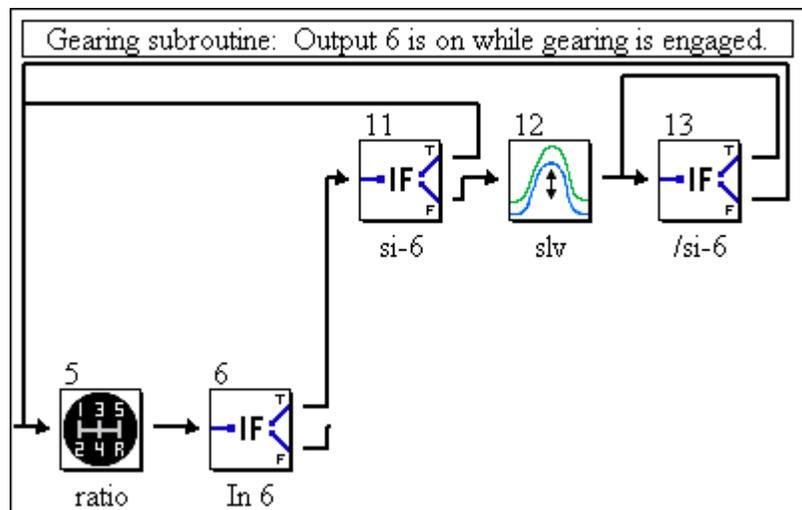
Blocks 3 – 4 & 7 – 9 comprise engaging and disengaging of gearing. Blocks 3 and 9 take care of activating /



deactivating Output 6 while entering / leaving the gearing subroutine. The key element to remember is to include a STOP block. If the STOP block is not included and the slave is disengaged while the master is motion, the slave will continue to rotate at the last known calculated speed of the master axis. Additionally, the STOP block will switch the axis to position mode, providing a controlled deceleration to zero speed and will maintain position once stopped.

### Running

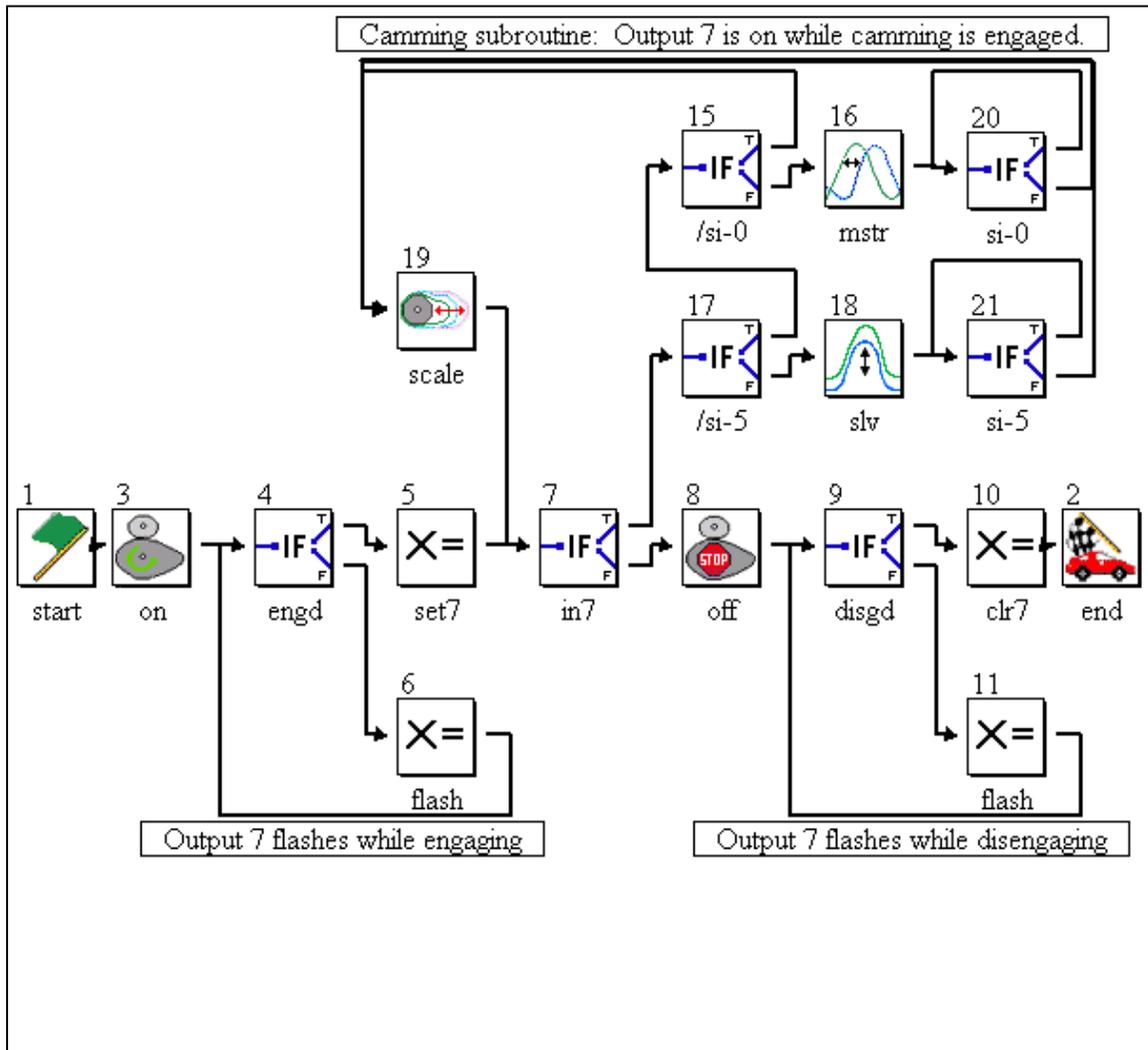
The rest of the blocks are executed while the system is engaged in gearing (or running). The user is able to adjust the gearing ratio by modifying the variables (user variables [Gear-Master] divided by [GearSlave]) in the Gear Ratio block (5). The gear ratio is a fraction of two integer numbers. Be careful not to use floating-point variables for



these. Doing so will truncate the values, resulting in lost motion. In addition, shifting of the Slave position is possible by activating input SI-6. The amount of offset is relative (in user units, based on the user variable [GearSlaveOffsetPosition]) to the current position of the respective axis. The offset can be accomplished over a given amount of time or distance (in user units, based on user variable [GearSlaveOffsetDuration]). System variable sSlaveOffset\_Mode determines if the duration is time or position based.

(sSlaveOffset\_Mode = 0, time based offset; sSlaveOffset\_Mode = 1, master position based offset)

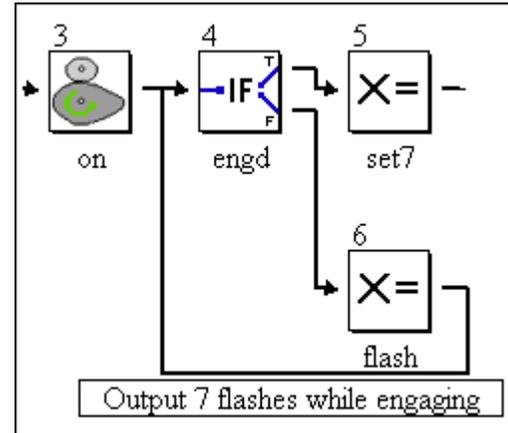
*Cam - Subroutine*



The three sections include: Engaging, Disengaging, and Running. (Note: Cam table generation and Cam Tool usage are beyond the scope of this document)

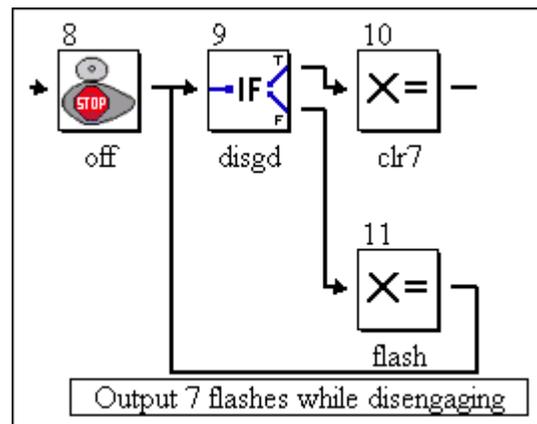
### ***Engaging***

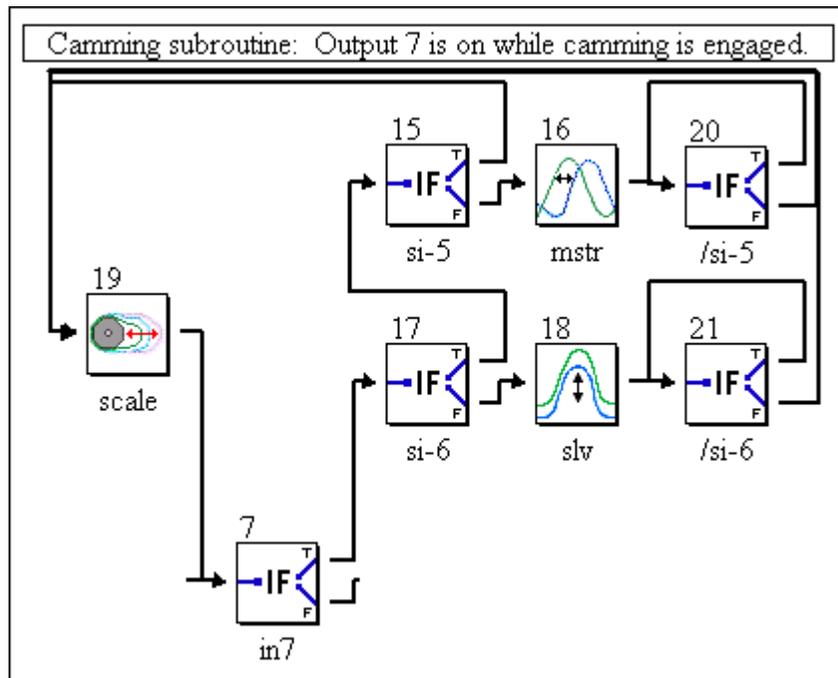
The CAM engage consists of the blocks 3 – 5. The Cam engages at a particular Master position based on the value entered in block three, for this program 0 is the position used. Block 3 also sets the system variable mState\_Camming = 1, this indicates that the system is waiting to engage. While waiting to engage (mState\_Camming = 1) output 7 flashes. Once engaged (mState\_Camming = 2) output 7 is turned on solid.



### ***Disengaging***

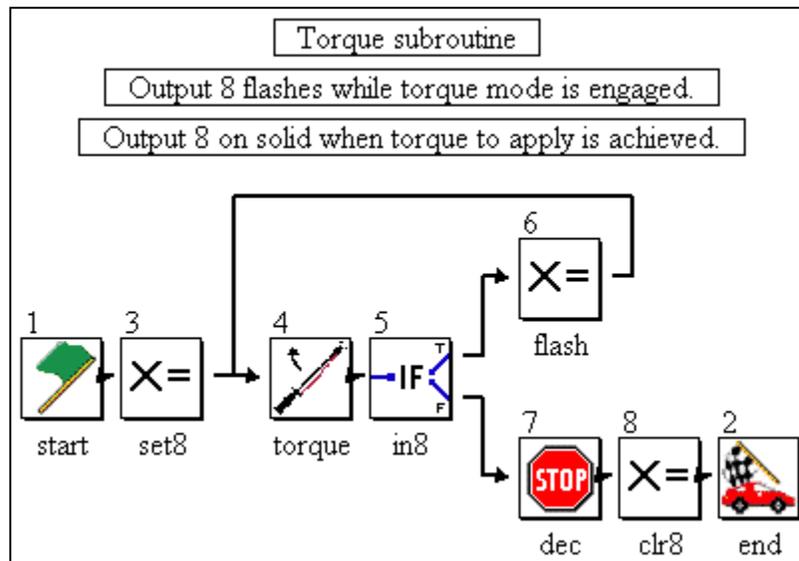
Disengagement begins after input 7 has been deactivated and consists of block 8 – 11. This works quite similarly to engaging, in Block 8 the disengage position is specified. Again, for this program 0 is the position used. Block 8 sets mState\_Camming = 4, while disengaging output 7 again flashes. Once the disengage position has passed (mState\_Camming = 0), output 7 is deactivated and the subroutine is exited.



**Running**

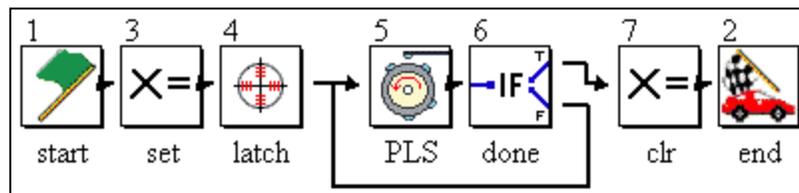
The rest of the blocks are executed while the system is engaged in camming (or running). The user is able to scale the cam by modifying the variable (user variable [CamScale]) in the CAM SCALE block (19). The scale factor is a percentage of the original cam, where 100.00% represents the original cam size. A value larger than 100% equates to an expanded cam, less equals a contract cam. In addition, shifting of the Master or Slave position is possible by activating inputs SI-5 or SI-6, respectively. Amount of offset is relative (in user units, based on user variable [CamMasterShiftPosition] or [CamSlaveOffset-Position]) to the current position of the respective axis. The offset can be accomplished over a given amount of time or distance (in user units, based on user variable [CamMaster-ShiftDuration] or [CamSlaveOffsetDuration]). System variable sSlaveOffset\_Mode determines if the duration is time or position based. (sSlaveOffset\_Mode = 0, time based offset; sSlaveOffset\_Mode = 1, master position based offset)

### Torque - Subroutine



The motor applies a variable amount of torque (based on a user variable [TorqueToApply]). It also limits the velocity of the motor (based on a user variable [Vel]). This continues for as long as input 8 is activated. Output 8 flashes while applying torque and goes on solid when motor is operating below the velocity limit. Once input 8 is deactivated the servo decelerates to stop, output 8 is deactivated and the subroutine is exited.

### Latch - Subroutine



The latch subroutine works very similarly to the move subroutine. The main difference (besides the lack of timers in this subroutine) is the move block has been replaced with a Latch Target block (4). Additionally, the values in PLS block (5) have been modified to be more suitable for the Latch routine. The routine has been configured to activate output during the various states of latching.

### ***Latch Target Block***

The latch target block is a pre-configured block that allows the user to make use of high speed (captures position in 30 microseconds or less) without knowing the ins and outs arming/disarming/windowing the latch signal. The block is much like a MOVE AXIS block; in fact if a latch is not received it will work identically to a MOVE AXIS block (Default Distance in Latch Target block = Position in move block). However, if a Latch signal is received, the servo will move the Distance After Latch from the point where the latch was detected.

Distance After Latch:	LatchDistanceAfter	degrees
Default Distance:	LatchDefaultPosition	degrees
Velocity:	Vel	degrees/sec
Acceleration:	Acc	degrees/sec <sup>2</sup>
Deceleration:	Dec	degrees/sec <sup>2</sup>
Latch Start Distance:	LatchStartWindow	degrees
Latch Finish Distance:	LatchStopWindow	degrees

Wait for Completion:

OK Cancel Help

Occasionally false latches can be a problem, causing the servo to move an incorrect distance. The Latch Start and Finish Distance parameters are useful in removing unwanted latch signals. They are used to set-up a window in which the latch is expected.

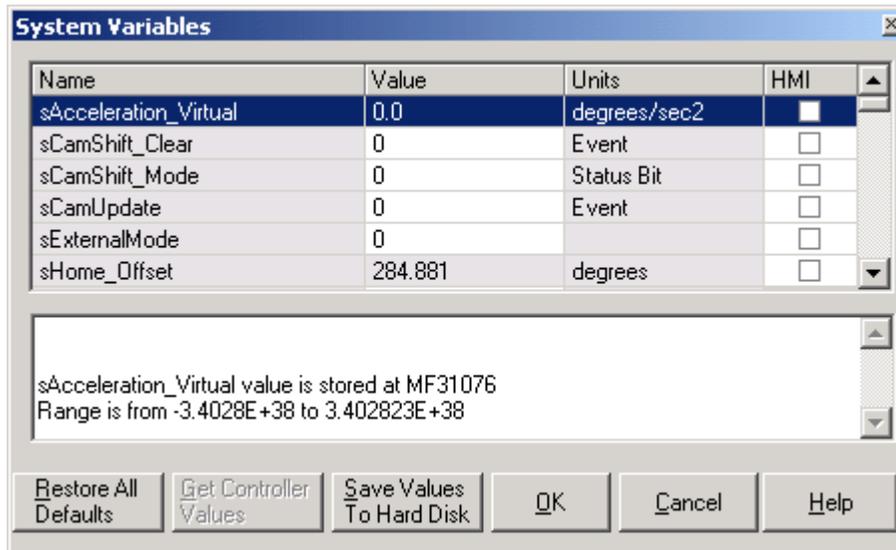
***Local Input functionality***

Input	Functionality		Description
	Manual	Automatic	
MW+ Name (SGDH Ref)			
Local_Input1	Servo Enable		Enables Servo and Starts manual/auto programs
Local_Input2	Off*	On*	*Off Selects Manual, On Selects Automatic
Local_Input3	-	-	unused
Local_Input4	-	-	unused
Local_Input5	Jog Forward	Index	Mode Dependent
Local_Input6	Jog Reverse	Gear	Mode Dependent
Local_Input7	Home	Cam	Mode Dependent
Local_Input8	Torque	Latch	Mode Dependent
Sigma_ServoOn (SI-0)	°	Cam Shift*	*Only applicable while in Camming
Sigma_HomeInput (SI-1)	Home SW	°	Home Switch Input
Sigma_POT (SI-2)	P-OT		Positive Over Travel
Sigma_NOT (SI-3)	N-OT		Negative Over Travel
Sigma_EXT1 (SI-4)	Alarm Reset*		*Only applicable after alarm has occurred
Sigma_EXT1 (SI-5)	°	Slave Offset*	
Sigma_Latch_Input (SI-6)	°	Latch Input*	

***Local Output Functionality***

<b>Output</b>	<b>Functionality</b>	<b>Description</b>
<b>MW+ Name</b>	<b>All Modes</b>	
° <b>All Modes</b> °		
Local_Output1	Servo Enabled	On when Servo Enabled
Local_Output2	Manual Mode	On when in Manual Mode
Local_Output3	Automatic Mode	On when in Automatic Mode
Local_Output4	Homed / Homing	On when homed, flashing while homing, Off otherwise
° <b>Manual Mode</b> °		
Local_Output5	Jogging Fwd	Flashing when jogging forward, until max jog speed achieved, then on solid
Local_Output6	Jogging Rev	Flashing when jogging reverse, until max jog speed achieved, then on solid
Local_Output7	Homing	Flashing while homing, on solid when homed, off otherwise
Local_Output8	Applying Torque	Flashing while applying torque, on solid when applied torque=requested torque
° <b>Automatic Mode</b> °		
Local_Output5	Index	On while indexing
Local_Output6	Gear	On while Gearing
Local_Output7	Cam	On while Camming, Flashing when engaging/disengaging camming
Local_Output8	Latch	On during latch routine
° <b>While Indexing</b> °		
Local_Output6	PLS	On first third of move
Local_Output7	PLS	On second third of move
Local_Output8	PLS	On last third of move
° <b>While Latching</b> °		
Local_Output5	PLS	On before Latch Window Starts
Local_Output6	PLS	On during Latch Window
Local_Output7	PLS	On after Latch Window Ends until default distance reached

## System Variables



Only four parameters were modified in the System Variables to correspond with the demo unit, they are as follows:

- sLimit\_Speed\_Negative: 30,000 degrees/sec
- sLimit\_Speed\_Positive: 30,000 degrees/sec
- sLimit\_Torque -30000 (.01% rated torque)
- sPosition\_CompletionWindow: 1.0 degree

These were modified to allow full peak motor rpm (5000), and allow full motor peak torque be developed. Lastly, in the completion window it is possible for an incompletely tuned system to be operated with consistency.

## Appendix C MW+ Camming 101

### 1. What is a cam profile?

A cam profile is simply a list of number pairs that describe the relationship of two axes. The simplest cam table consists of two pairs of points, the beginning, and the end. This would make a “straight line” relationship, identical to a gearing relationship, where the gear ratio is the last pair of numbers. A cam profile can be much more complex. It can consist of thousands of pairs of numbers, describing a mathematical formula, such as a sine wave, or any arbitrary relationship that best applies to the process. With a cam table in place, the servo can be precisely positioned at every step along the machine cycle. The possibilities are almost limitless, the slave can match speed, accelerate ahead of the master, remain motionless for a time, or move backwards; all speeds dictated by the speed of the master. Most importantly, the cam profile contains position data that synchronizes two axes, regardless of speed.

### 2. How are cam profiles created in MW+?

Two ways: either with Cam Tool, or calculated within MW+ blocks.

Set Parameter							
No.	Phase Start	Phase End Point	Position Start Point	Position End Point	Curve Shape	Phase Plotting	Data Edition
1	0000000000	0000019000	0000000000	0000327680	Straight line	0000001000	Not Provid
2							
3							

Figure 1: This is a simple profile, with only a straight-line segment.

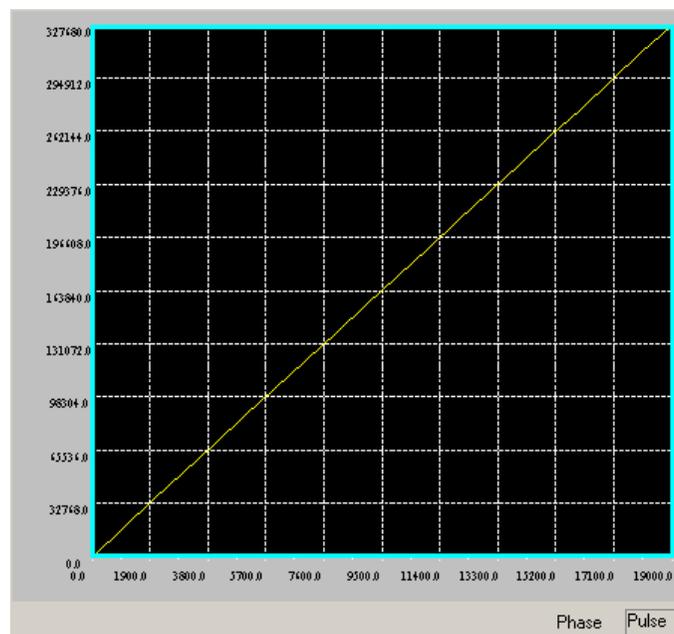


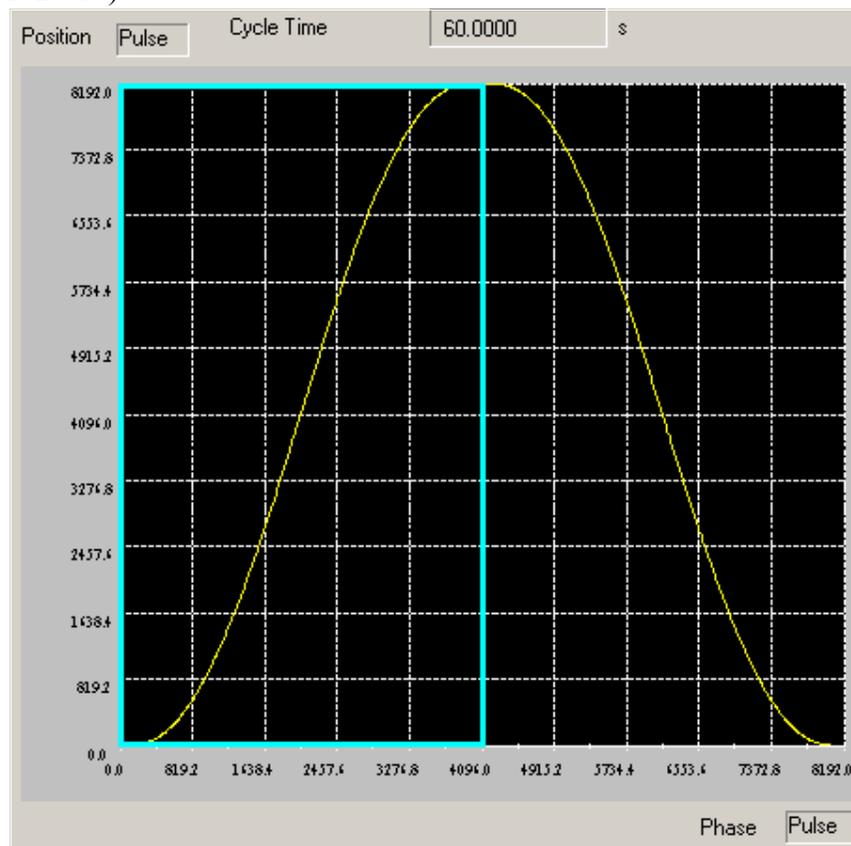
Figure 2: Cam Tool graph showing the master / slave relationship

MW+ comes with “Cam Tool” which is a separate program that can also be used with MotionWorks, or as a stand-alone package to create cam profiles and save them as a CSV file for any other controller. Cam Tool allows the user to create complex cam profiles with up to 20 different segments or profiles in chart format. The concept is to specify the “Phase” & “Position” which correspond to “Master” & “Slave” for each segment of the required profile. Once the master & slave numbers are entered, select a formula to be used in determining all the points between the first and last point of the segment.

Set Parameter							
No.	Phase Start	Phase End Point	Position Start Point	Position End Point	Curve Shape	Phase Plotting	Data Edition
1	0000000000	0000004096	0000000000	0000008192	Modified sine	0000000010	Not Provid
2	0000004096	0000008192	0000008192	0000000000	Modified sine	0000000010	Not Provid
3							
4							

Figure 3: A cam profile that moves the slave out and back in a sine wave pattern as the master moves forward

Note that “Phase Plotting” is the resolution of the segment, meaning that an actual data point will be generated for every x units of travel of the master. This defines the actual precision of the path, because the controller will create straight-line segments between each resulting value in the controllers memory. There will be 820 pairs for this cam profile. (8192 / 10)

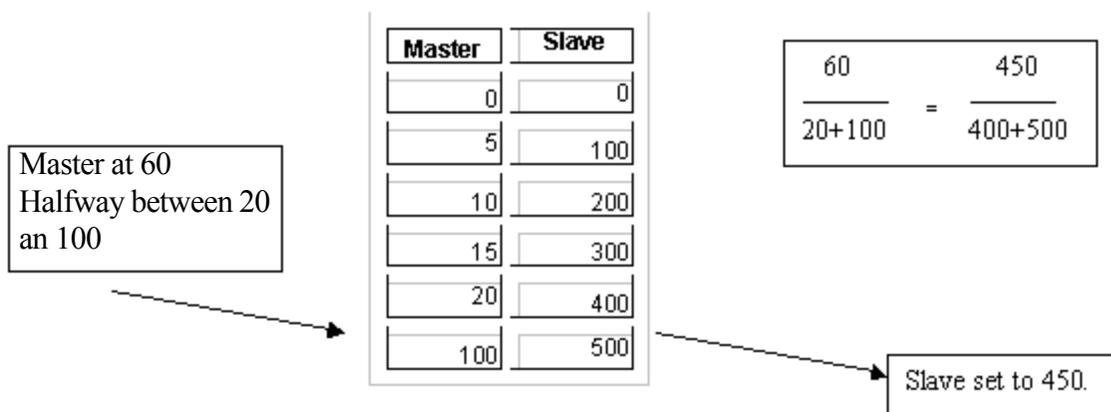


Cam Tool has some advanced features that allow the user to determine the servo's ability to follow the profile at given master speeds. The "Set Style" window allows the user to enter details about the inertia of the motor, load, and machine cycle time. Graphs indicate by color if the servo is capable of supplying enough torque to perform the move.

### 3. Resolution Considerations

It is usually impossible to include a data pair for every possible position of the master, and sometimes it is not necessary to enter every position. Slave position resolution is a direct result of the value output from the cam look-up table. The controller uses a function that provides straight-line linear interpolation to find the slave position.

Look at the following example:



A little background about cam tool; master units are "what you enter is what you get." No conversion takes place when the CDD file is written. However, Cam Tool always generates slave data in pulses. MW+ is designed such that the slave data MUST be in pulses. In Cam Tool, either program the slave in pulses, or determine the number of pulses per unit travel of the slave. Unfortunately, MW+ and Cam Tool are not able to share information about the system properties.

A breakdown of the equivalence is shown on the next page.

The image shows three software dialog boxes. The top-left box is 'Properties - twocam' for encoder '6 (External Encoder)'. The top-right box is 'Properties - twocam' for encoder '2 (MP940)'. The bottom box is 'Set Style'.

**Properties - twocam (6 (External Encoder))**

(ID)	6
Label	ExternalEncoder
Enabled	True
FeedConstant	199.4911
GearBoxInput	5
GearBoxOutput	1
MachineCycle	554
MovementType	Rotary
PulseType	Quadrature
Resolution	8192
UserUnits	

**Properties - twocam (2 (MP940))**

(ID)	2
Label	MP940
BatteryTest	Disabled
EncoderResolution	8192
EncoderType	Incremental
FeedConstant	360
Firmware	N/A
GearBoxInput	5
GearBoxOutput	1
HighScanSetting	1
LoadType	Rotary
LowScanSetting	20
MachineCycle	360
MotorRatedSpeed	3000
UserUnits	degrees

**Use/Units**  
Select cm, mm, inches, pulses, degrees, or a user defined type.

**Set Style**

[ Phase/Position Setting ]

Unit(Phase)  Degree  Pulse  No Unit  
 Unit(Position)  mm  Pulse  No Unit

Max Phase Value from the Bottom Dead Center (Where the bottom dead center = 0.)    
 Max. Position Value from the Bottom Dead Center (Where the bottom dead center = 0.)

[ Machine/Motor Information ]

Ball Screw Lead  Not Provided  Provided  mm  
 Gear Ratio  Not Provided  Provided  
 Ball Screw Axis/Motor Axis  /

Required Time for One Cycle(The shortest time)  s  
 Motor Rated Speed [\*]  r/min  
 PG Pulse Number after Multiplication  p/r  
 Rated Torque [\*]  kgf.m  
 Instantaneous Peak Torque [\*]  kgf.m  
 Motor Inertia [\*]  kg.m2  
 Gear+Coupling Inertia [\*]  kg.m2  
 Load Torque(Motor axis conversion) [\*]  kgf.m  
 Load Inertia(Motor axis conversion) [\*]  kg.m2

[ MEMO ]  
 Input data to items with [\*].  
 The following informations are displayed when data is edited.  
 1. Effective torque as a percent of motor rated torque.  
 2. Peak torque as a percent of rated torque.  
 3. Max. speed

OK Cancel

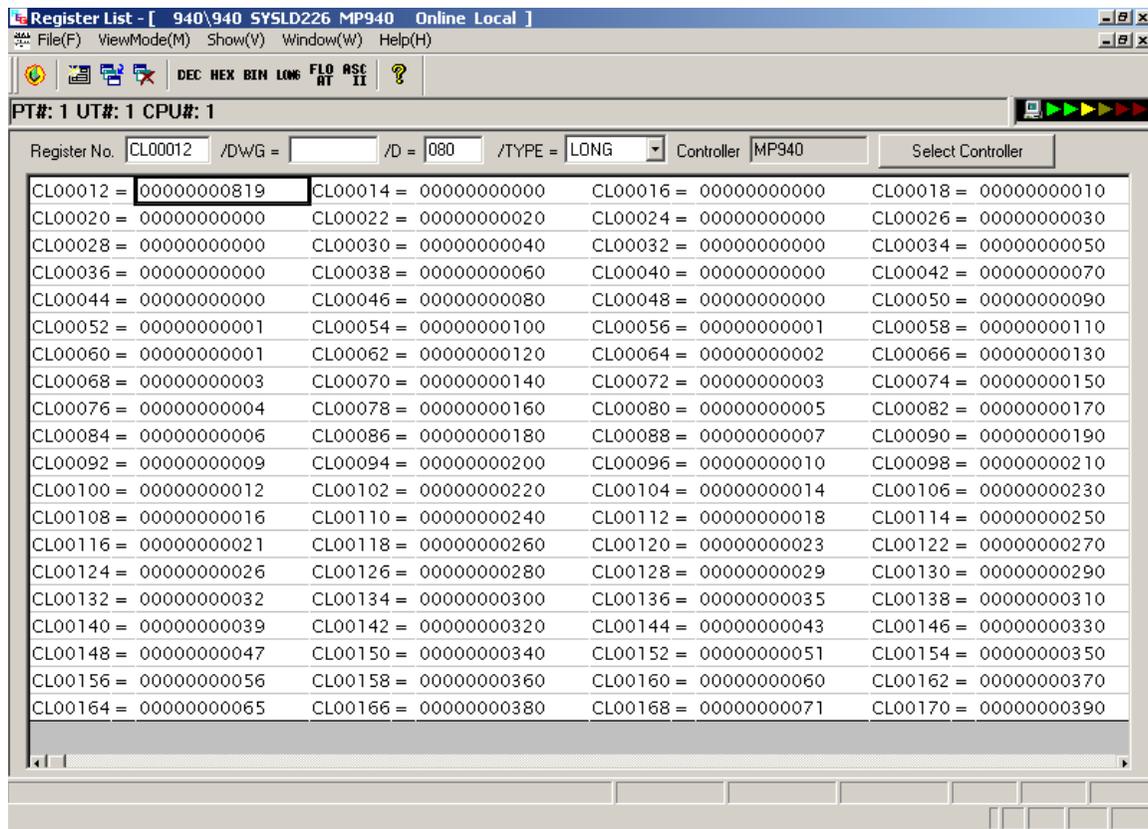
“Max phase value” is the same as the External Encoder Machine Cycle, in pulses. Use the Position formulas in the MW+ manual to determine the number of pulse in the machine cycle.  $554 * 8192 / 199.4911 * 5 = 113463$ .

“Max Position Value” is the same as the MP940 Machine Cycle. “Ball screw lead” is equivalent to MP940 FeedConstant.

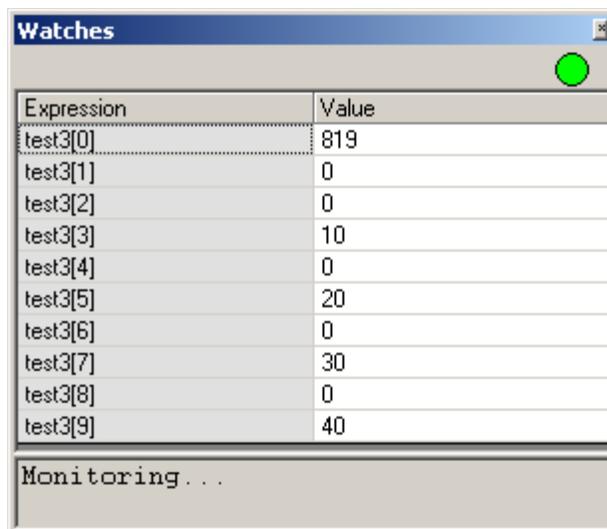
Although it is possible to program the master in user units, or magnified user units, such as  $36000 = 360$  degrees, the best resolution usually results from programming the master in pulses. The exception to this would be if the number of pulses in the master cycle is less than the number of magnified user units.

*For example, if the master cycle were 32768 pulses, resolution would be just as sufficient at  $36000 = 360$  degrees. Resolution would suffer however, if the cam table were programmed in  $360 = 360$  degrees. The reason for this is that all cam data is stored as long registers. This means no fractions. It also means that there are only 360 possible points in the table. THE CONTROLLER WILL NOT PERFORM THE ABOVE CALCULATION TO DETERMINE FRACTIONAL DEGREES. Note that  $32768 / 360 = 91$  pulses. This is the distance the master will travel and still generate the same data point for the slave. The effect is a stair stepping motion.*

#### 4. What does the cam table look like in the controller?



Above is the view from MotionWorks Register List, which displays up to 80 registers at a time. If you were to look at the table definition window of MW+, it would indicate that the table starts at CL0012. Below is the same data as seen MotionWorks+ v2.82 Data monitor window:



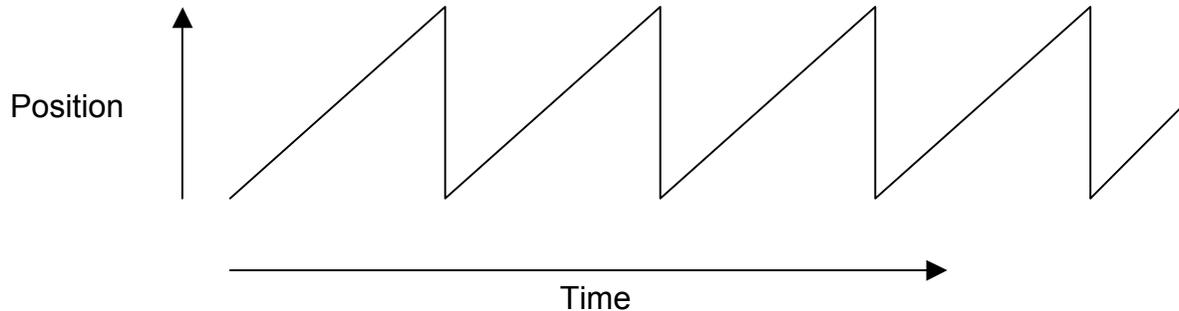
In either case, the raw data used by the controller can be viewed. The very first element in the table contains the number of master / slave pairs. Following the first entry is master position1, slave position1, master position2, slave position 2, etc., etc.

Understanding the data format in the table is important when implementing an algorithm in MW+ blocks to calculate a cam profile.

## 5. How does camming work?

### *The Master*

First, camming requires that the master position be modularized to the external encoder machine cycle, which must also match the last master point in the table. The system ladder modularizes the master automatically when camming is engaged. A modularized position can also be selected in the system properties, so position of the master (mPosition\_External) will look like this:



This modularized value is used as a pointer to the cam table to find the slave position. However, an important conversion happens first. The modularized master value is scaled by the last value found in the cam table. The conversion looks like this:

Modularized Master in counts.	<b>Xlast</b>	<b>PulsMCE</b>	
└ DL00042	× DL00090	÷ ML31130	
0000000047	0000000360	0000000100	
			<b>M(TUnit)</b>
			⇒ DL00082
			Value used as 0000000169
			pointer to cam table

This conversion can be the source of trouble if the external encoder's machine cycle and the last master position are not the same. The purpose is to convert the master in counts to "table units" if they are not already the same. In the above example, the machine cycle is 100 counts, but the last master position in the table is 360. The conversion takes place, and  $47/100 = 169/360$ , or about 50% through the cycle. If the cam master were programmed in counts, then the conversion would be 1:1. This same feature can be a benefit if the application can benefit from stretching or shrinking the profile by changing sMachineCycle\_External.

## *The Slave*

Most cam profiles can be classified as one of the two below. The slave will either move out and back to the same position, or continue in one direction each cycle.

## *Engaging*

Engaging is the process of synchronization. During this process, mState\_Camming is set to one. When the CAM block is executed, the master may not be at a location for the slave to start following. A pre-defined window of about 1% of the master's total travel through the cycle is calculated. The formula looks like this:

$$\begin{array}{rcl} \mathbf{Xlast} & & \mathbf{Window} \\ \vdash \mathbf{DL00090} & \div & \mathbf{DL00094} \\ \mathbf{0000000360} & \mathbf{00120} & \mathbf{000000003} \end{array}$$

Note that dividing by 120 is the same as 0.83%. In this example, the master table is in degrees; the engage window is 3 degrees. Remember, it is not recommended to use degrees as the units of the master. Use 36000 or 360000, or counts. Degrees are shown here for simplicity.

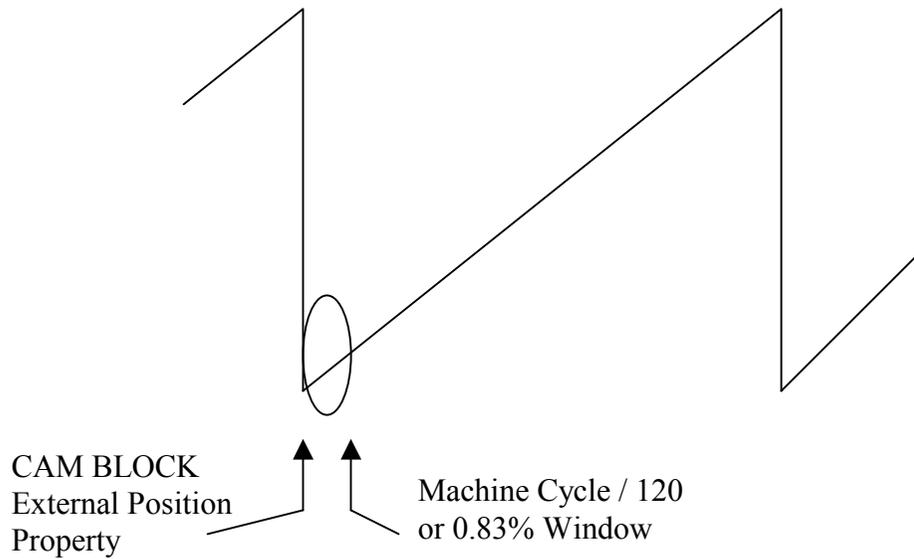
The “External Position” property of the CAM block in MW+ is the position of the master in table units, where the slave must begin following. The master position is compared to this value every high scan. When the master position is greater than the window, mState\_Camming changes to two, and the slave begins to follow because it's commanded position register is updated by data coming out of the cam table.

When the slave becomes engaged, an initial offset is determined. The current position of the slave at the time of engagement is stored as an offset, and camming starts from this point. The reason for this is so that the slave will never jump when camming is engaged, if it is not already positioned at zero. This also gives the user the freedom to position the slave axis properly before camming takes place.

### *Why is a window required?*

The window is a built in function of MW+. It is required because in the dynamic world of motion, it is quite difficult if not impossible to compare to an exact position. Each time the controller scans the position of the external encoder; it's only a snapshot. If the encoder is moving, the "snapshots" may never match the exact value entered as the "External Position."

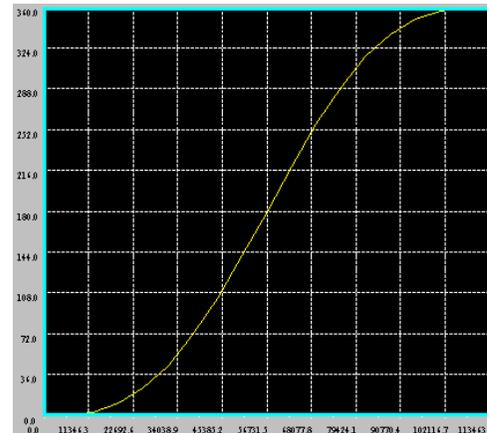
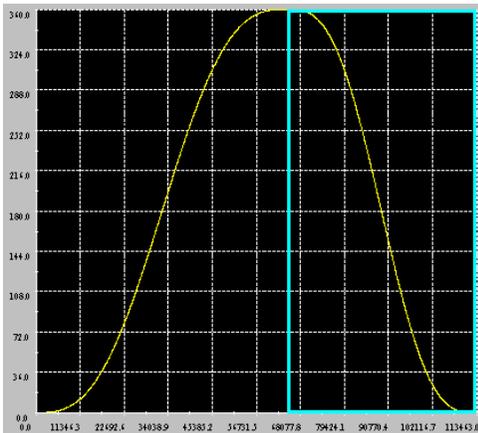
Note: Note: The window spans from the position defined in the CAM block forward. It is not equally centered on the position specified.



## Running

During run, the following happens at each high speed scan: The controller reads the external encoder, modularizes it, scales it to the cam table, sends it to a function that finds the required slave position, and adds this result to the slaves cumulative offset position, and then adds the SLAVE OFFSET. This forces the slave to move as the master moves.

There are basically two types of cam tables; “one way cam,” and “out and back.” The controller detects the type of cam by comparing the first slave position to the last slave position.



Out and back is simple; the values in the cam table can be considered the absolute positions of the slave, it will never be outside the range of values in the table.

A one-way cam uses the absolute slave positions in the table once. After that, the slave is outside the range of the values found in the table. To account for this, the controller detects when the master position exceeded the machine cycle, and adds an offset to the slave according to the following formula:

$$\text{Offset} = \text{Offset} + ((\text{SlaveLast} - \text{SlaveFirst}) * \text{ScaleCam})$$

When the next scan starts, the modularized master position will be pointing to the beginning of the table. A small slave position will result. This small position is added to the offset from the last cycle, and the slave continues to move forward.

## *Disengaging*

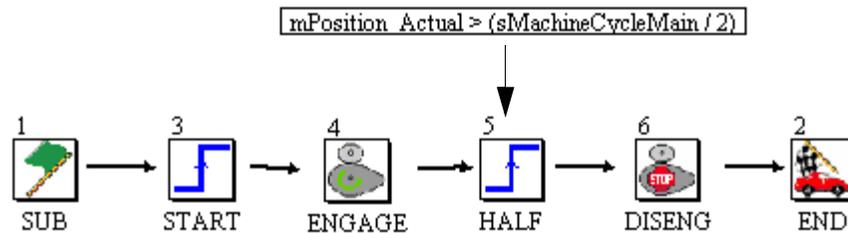
Disengaging is similar in concept to engaging. During this process, mState\_Camming is set to four. The master may not be at a location to disengage at the time the CAM disengage is executed. The “External Position” property of the CAM block specifies the master position at which the slave must stop following the master. The same window size is used to determine when it is OK to disengage.

On the scan when it is determined that the master is in the window, a switch takes place, and instead of using the actual master value for the cam look up table, the system ladder uses the value or variable entered in the CAM disengage block’s “External Position” property as if it is the actual master position during that scan. This is done because slight errors in the slave position could result if on the last scan the master is not perfectly at the disengage position, and the cam table is designed such that the slave is not exactly at its intended resting position when disengagement occurs.

This phenomenon would most likely appear in applications that engage & disengage frequently. (A perceived drift would result if the slave were repeatedly disengaged at a slightly different location than the engage point.)

### *Performing a cam “one shot.”*

A cam one shot is useful when the cam profile must only run long enough to complete one cycle. This is easily accomplished by engaging, waiting for the slave engage or waiting for the slave to move an appreciable distance of travel, and then disengaging. Of course, the action of actually engaging and disengaging takes place when the master is within the window. Here is an example of the MW+ code to perform a cam one shot.



The subroutine could be enhanced to loop back and repeat the one shot each time the start condition is true.

The reason for waiting to disengage: If the CAM disengage block is issued immediately after the CAM engage block, there is a good chance that the cam profile will never run. The master may still be in the window, and that would cause disengagement immediately. There are other tricks to this. Given the application, it may be possible to set up slightly different engage and disengage positions, and place the disengage position before the engage position. This would work best if there were a sizeable portion of the cam table that does not move the slave right away.

Remember that the “Sync Position” or External Position is in master table units!

## How does CAM SHIFT work?

The purpose of the CAM SHIFT is to redefine the position of the master while camming is engaged. The function that performs cam shift is much more than just a simple redefinition of the master. This is one of the most complicated concepts in MW+ camming.

Surprisingly, CAM SHIFT does nothing to the master at all. It affects another value that is always added to the master. The reason for this is to prevent lost pulses. When redefining a moving axis, it is difficult to do so without losing a pulse here and there. This error is caused by speed variation of the axis between scans. By the time it is redefined, it may have moved from the known location.

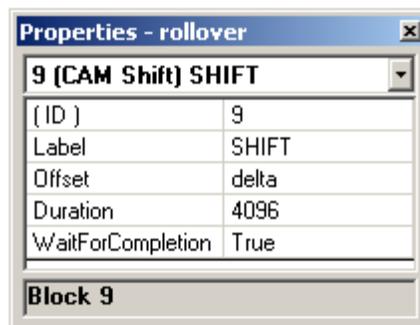
In the “How does camming work?” section, we said,

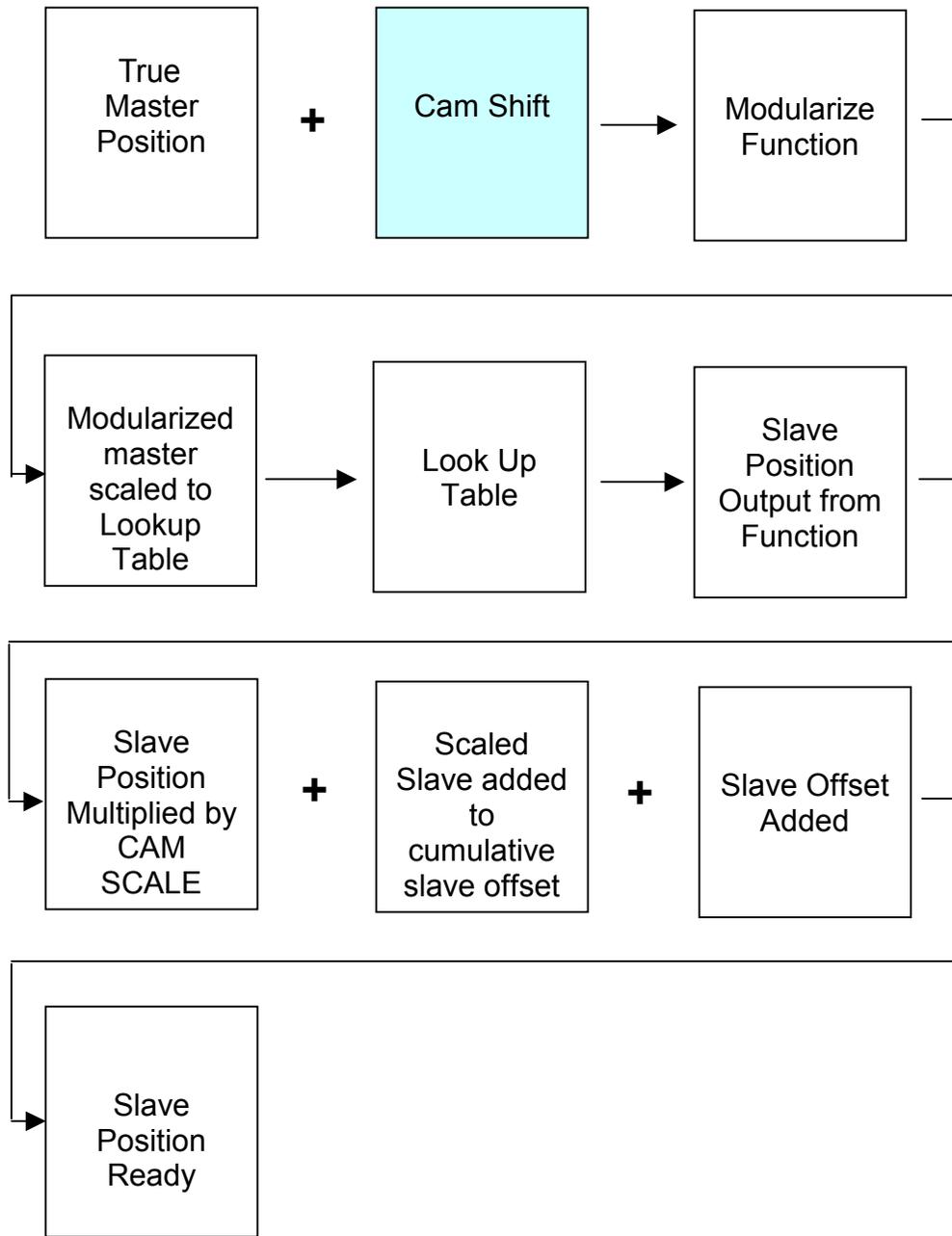
*The controller reads the external encoder, modularizes it, scales it to the cam table, sends it to the function that finds the required slave position, and adds this result to the slaves commanded position. This forces the slave to move as the master moves.*

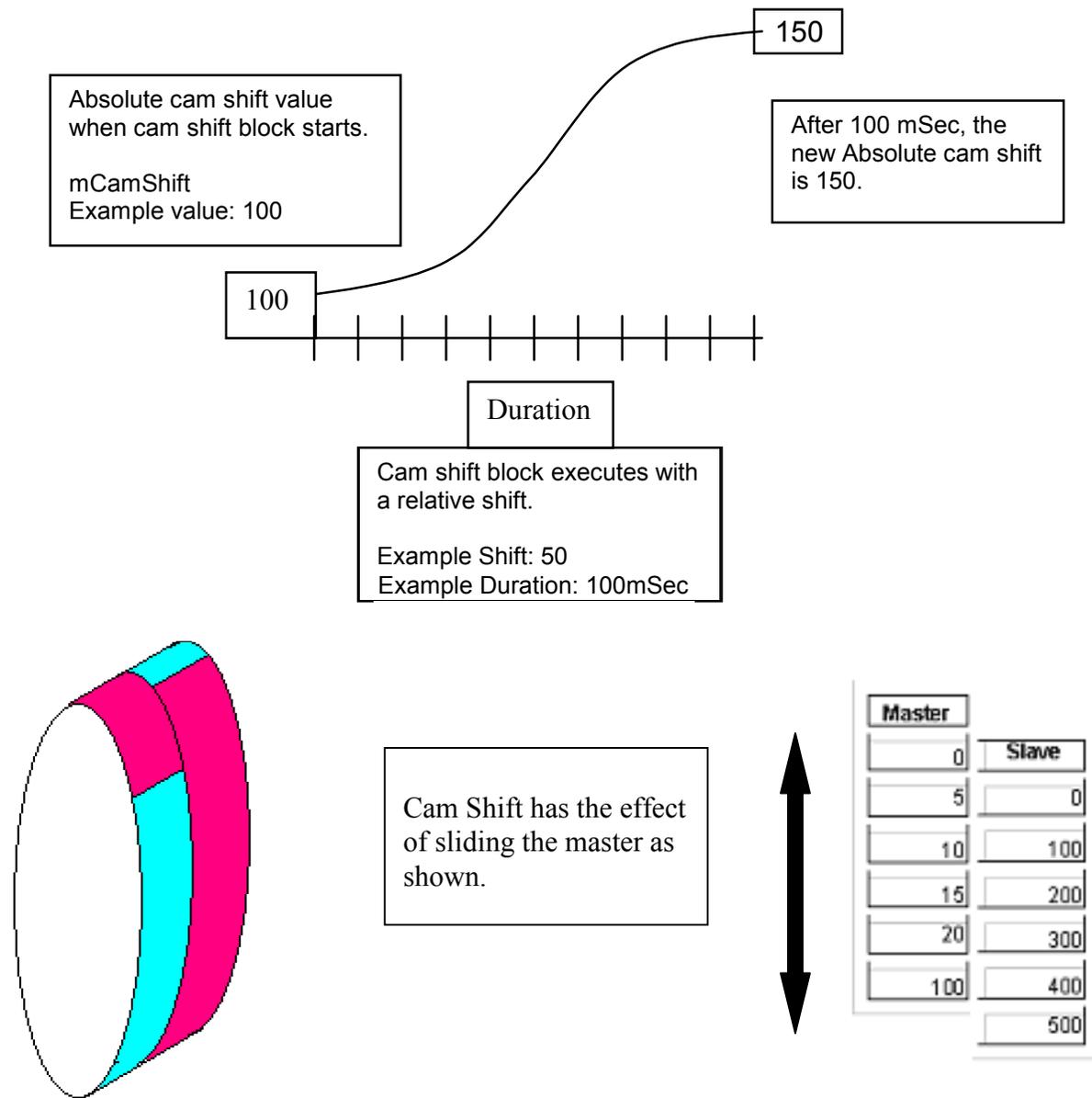
What actually occurs is slightly different. It really works like this:

*The controller reads the external encoder, **adds the absolute cam shift**, modularizes it, scales it to the cam table, sends it to the function that finds the required slave position, and adds this result to the slaves commanded position. This forces the slave to move as the master moves.*

The CAM SHIFT block adjusts the master a relative amount in a modified sine profile over the duration specified.







The CAM SHIFT function can operate in two modes. The adjustment can take place over a time specified in milliseconds, or over a relative change in position of the master. Operation is changed by setting sCamShift\_Mode to either zero or one. Zero is the time-based setting, and one is the position-based setting. The position setting is more practical, because the time to complete the shift will vary as the machine speed varies. It is easy to control the corrections and be sure they are complete when the master reaches a certain position. If the master stops, the correction waits partially corrected.

### *What applications benefit from CAM SHIFT?*

- A process that must accurately track randomly spaced products on a conveyor belt.
- A process where slippage, stretch, or shrinkage requires realignment of the master and slave to continue working properly.

By using the external latch, MW+ can calculate the distance between products, and continually adjust so the slave motion is always synchronized to the product, not simply the master position.

The following table demonstrates an effective way to understand what happens when the product stretches or products are coming at random. This was created as an Excel sheet to understand the formulas before entering them in a MW+ program.

After the second product has been latched, it is determined that the product has stretched 0.1 inches. A CAM SHIFT of an equal and opposite amount is given to negate the stretch. This amount can be added over time, such that the adjustment is subtle, and causes little if any negative effects due to the slight de-synchronization that occurs during the shift.

<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>
5.5	Modulus (Machine Cycle)				
2.75	MaxShift (Mid Cycle)				
<b>Product Number</b>	<b>Current Shift</b>	<b>Hypothetical Latch</b>	<b>Relative Shift Amount</b>	<b>Resulting Abs Shift</b>	<b>Shifted Master</b>
1	0.00	0.00	0.00	0.00	0.00
2	0.00	5.60	-0.10	-0.10	5.50
3	-0.10	11.20	-0.10	-0.20	11.00
4	-0.20	16.80	-0.10	-0.30	16.50
5	-0.30	22.40	-0.10	-0.40	22.00
6	-0.40	28.46	-0.56	-0.96	27.50
7	-0.96	35.00	-1.04	-2.00	33.00
8	-2.00	42.11	-1.61	-3.61	38.50
9	-3.61	48.00	-0.39	-4.00	44.00
10	-4.00	54.00	-0.50	-4.50	49.50

- Assume the external encoder machine cycle (default product length) is 5.5 inches. This is the nominal product length.
- MaxShift is something that can be used mainly for random product infeed, where you need to determine which way is better to shift, to the previous machine cycle, or the next. This will cause the slave to either slow down or speed up.

- When the next product arrives, the position is recorded with the latch function. The latch value was 5.6. Since the machine cycle is 5.5, it is expected that the next latch will arrive at 5.5. If it comes later, the product has stretched. NOTE: For simplicity, we are using user units in this discussion. In MW+, use the `mPosition_Latch_External_Counts`, which is the unmodularized encoder latch in counts.

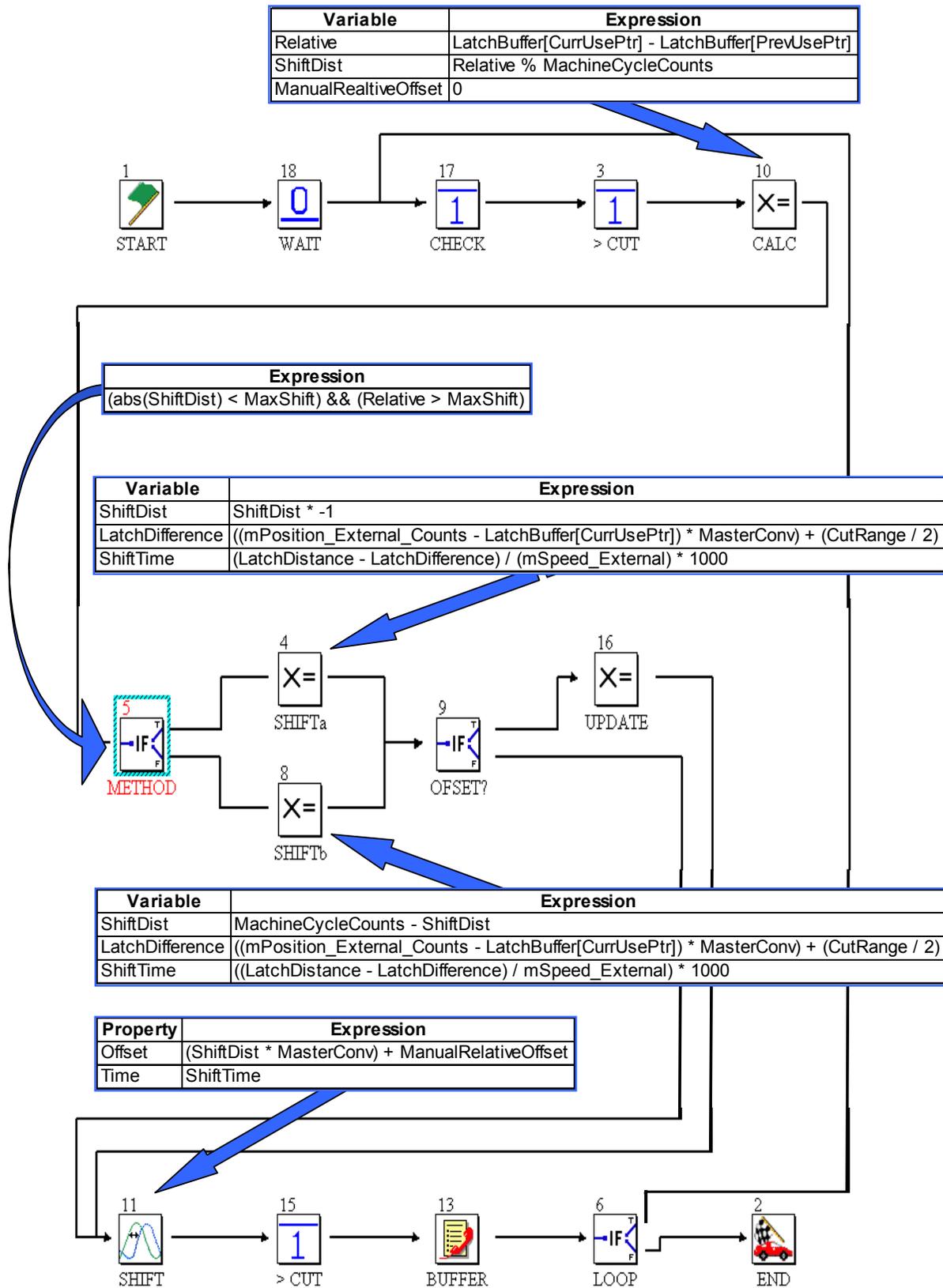
- Here is the EXCEL formula for column D:

```
=IF ( ABS ( MOD ( C6 - C5 , $A$1 ) ) < $A$2 , MOD( C6 - C5 , $A$1 ) * -1 , $A$1-MOD ( C6 - C5 , $A$1 ) )
```

- Here are the equivalent formulas to determine the adjustment in MW+:

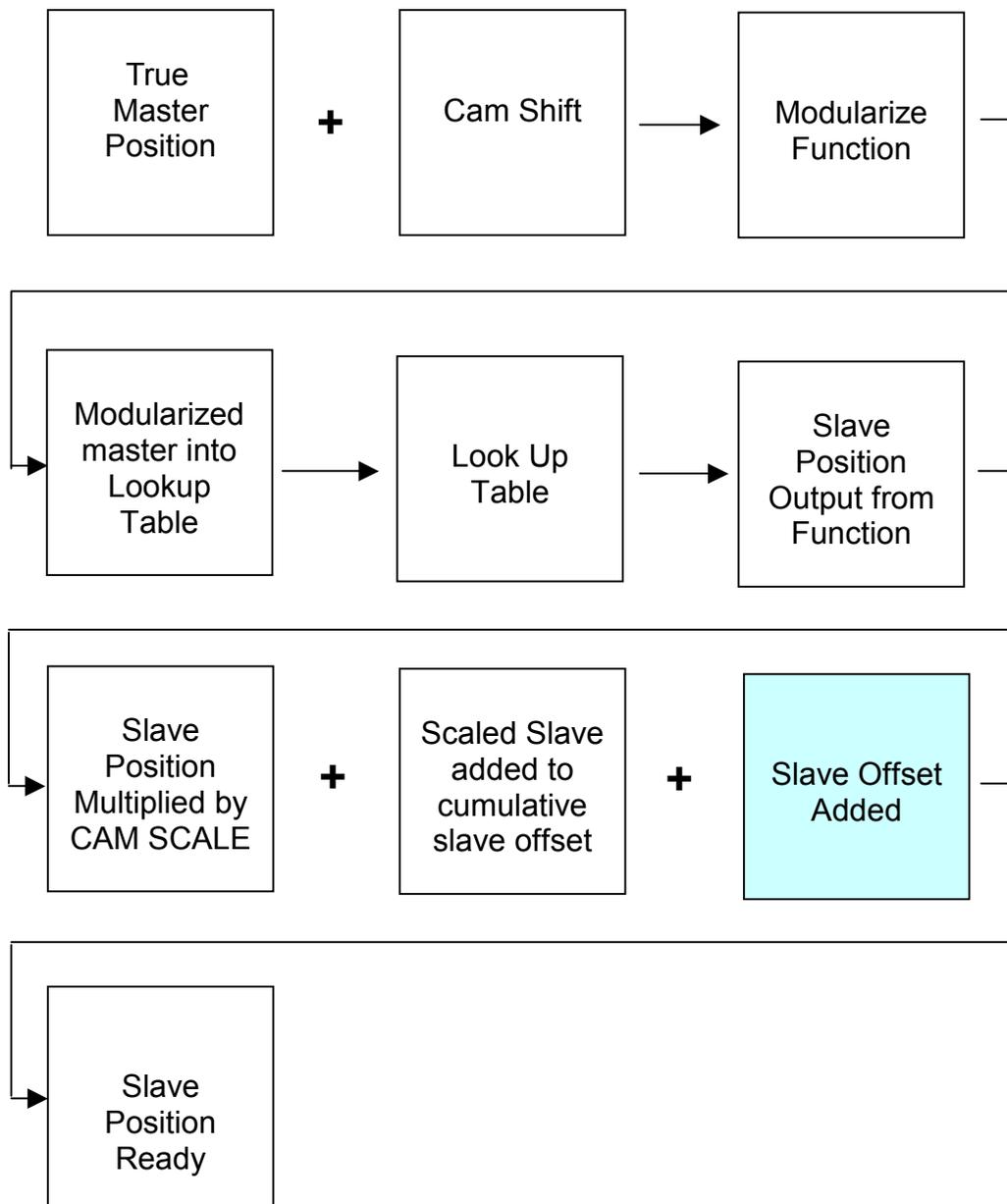
```
IF (Abs(CurrentLatch - PreviousLatch) % mMachineCycle_External_Main) < MaxShift THEN RelativeShift = (CurrentLatch - PreviousLatch) % mMachineCycle_External_Counts) * -1
ELSE
RelativeShift = sMachineCycle_External - (CurrentLatch - PreviousLatch % sMachineCycle_External))
END
```

- The following is the equivalent block structure in MW+. The following is a diagram of a complete program thread that handles adjustments. Blocks are included to control the timing of the correction, to assure that it takes place when a cut is not occurring. Another thread is dedicated to capturing the registration (not shown.)



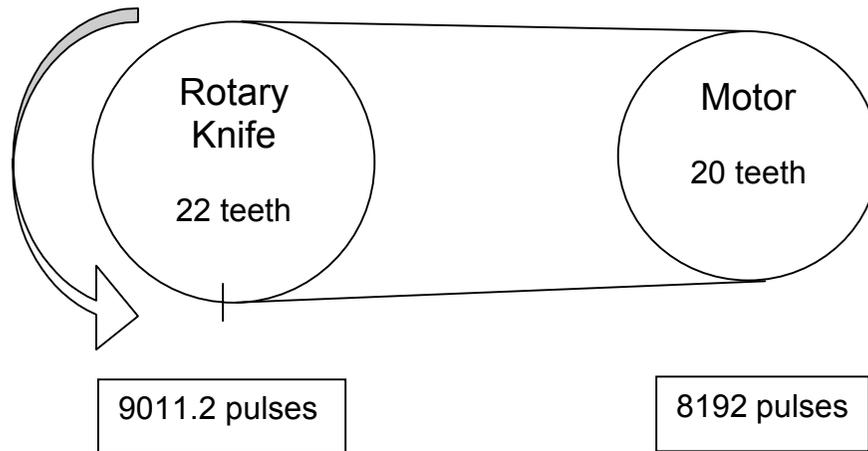
*How does SLAVE OFFSET work? How would an application benefit from using slave offset?*

The slave offset works similar to the CAM SHIFT. It uses an S-Curve profile to change the offset from the current to the new absolute offset. The slave Offset is simply a value added to the slave position after it's position has been determined from the cam table. It is the very last operation in the cam function.



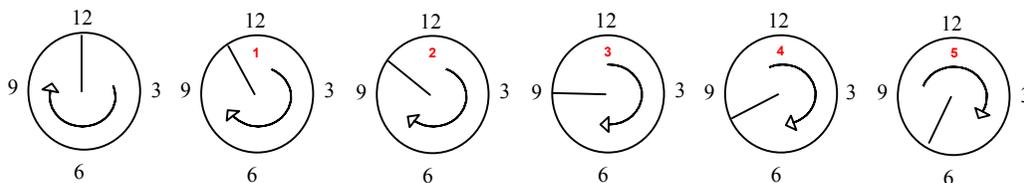
The slave offset is useful when the slave must be adjusted at run time a uniform amount across the entire cam profile. This can account for changes in material, temperature, humidity etc.

The slave offset is also useful when the machine cycle is not an integer. Imagine a machine where a timing belt is used as shown:



Every time the knife rotates, 0.2 pulses required to keep the knife orientated in the correct position are lost. This error will add up quickly, and after about 11,000 cycles, the knife will be off by 45 degrees.

A remedy to solve this is to count the machine cycles, and use the SLAVE OFFSET block to add back the lost pulses. After 100 cycles, the knife will be behind by 20 pulses. Using the SLAVE OFFSET block will correct the physical error of the knife, but numerically, the knife's position will still seem to be drifting.



Imagine a clock, but the machine cycle cannot be set to exactly 12, the closest you can set it is 11. The line in the clock indicates the position where the machine cycle numerically indicates 12 O'Clock (or 00:00 hours military time.)

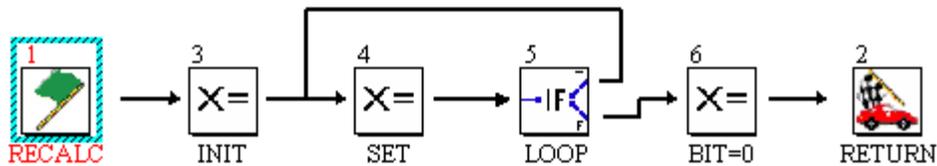
The slave offset can move the pointer ahead one hour each time and keep the physical position correct. Notice that if you want to position the pointer at the machine's 12 O'Clock position after it has run 5 cycles, you would actually have to position it to 5 O'Clock. Keeping track of the total SLAVE OFFSET added (total of all relative offsets given) is the key to keeping the position under control.

*Can cam profiles be calculated using a formula in the MW+ program when compile & downloading a new cam profile is not an option?*

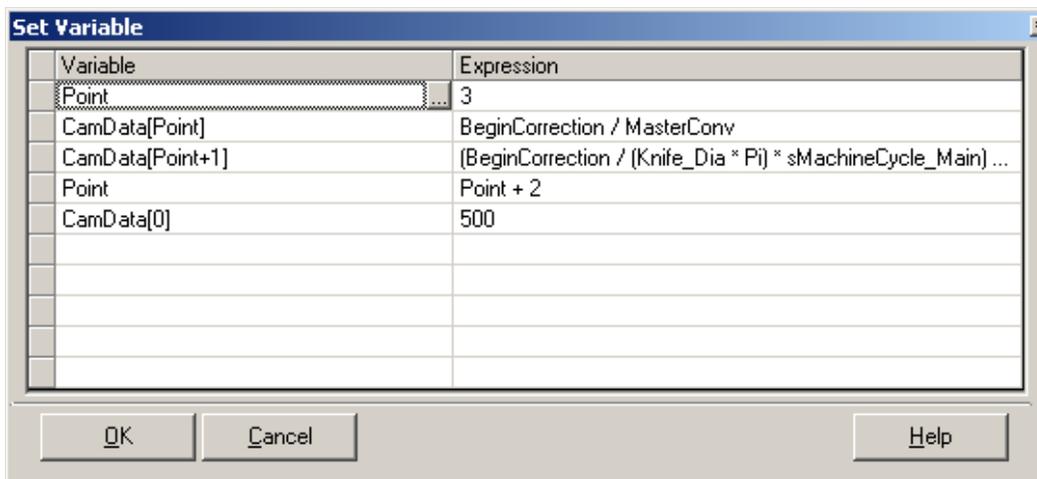
Yes. It is possible to write directly to the cam table from within MotionWorks+. The table structure is easy to understand.

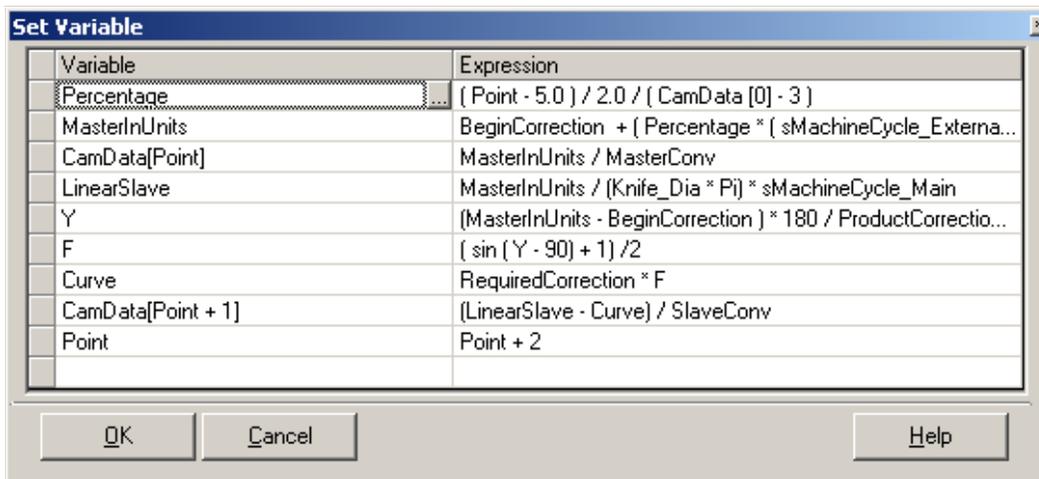
You must first create a table using CAM TOOL, and it must be at least as large as the data points you will calculate in MW+. Even if you intend to calculate your own cam table, you must still select a CAM type table.

The following is an example of the cam calculations of a subroutine called “ReCalc.”



### ***INIT BLOCK***



**SET BLOCK****LOOP**

The actual calculations are beyond the scope of this document. An excel file describing the formulas, using the same variable names is available at [ftp://motion:controls@ftp.yaskawa.com/Applications/MW+\\_Examples/](ftp://motion:controls@ftp.yaskawa.com/Applications/MW+_Examples/)

## Notes

## Index

### A

#### Analog

- input 107
- output 107

### B

Ball screw lead 205

#### Base Units

- Encoder counts 128
- Milliseconds from rated speed 137, 147, 150, 154
- Milliseconds to rated speed 127, 137, 147, 150, 154
- Percentage of rated speed 129, 137, 147, 150, 154

Block toolbar 58

#### Blocks 103

- Active 105
- Appearance 61
- CALL SUBROUTINE 106, 132
- CAM 74, 133
- CAM SHIFT 135
- CHANGE DYNAMICS 137, 164
- Connection lines 60
- Connection mode 60
- Connections 60
- DEFINE POSITION 138
- Editing 60
- END 139
- GEAR 140
- GEAR RATIO 141
- General 103
- HOME AXIS 142
- IF EVENT 144, 148
- IF FAULT 145
- INPUT 146
- JOG AXIS 147
- LATCH 148
- LATCH TARGET 148, 150
- LAUNCH PROGRAM 153, 165

Lines 60

- Logic 103
- Motion 103
- MOVE AXIS 103, 137, 154
- Output port 60, 103, 144, 145
- PHASE SHIFT 141
- Placement 60
- PLS 156
- Program Flow 103
- Properties 103
- RESET FAULT 158
- SCALE CAM 159
- SERVO 160
- SET VARIABLE 34, 161
- SLAVE OFFSET 162
- START 163
- STOP 147
- STOP MOTION 164
- SUSPEND PROGRAM 153, 165
- TIMER 166
- TORQUE 167

Boolean type variables 161

### C

Cam profile 201, 207

Cam profiles 209

CAM SHIFT 214

CAM TOOL 222

#### Camming 164

- Cam profile 79
- CDD 34, 133
- Disengage 133
- Electronic Cam Tool 3, 74, 133
- Engage 133
- Machine Information 75
- Master position 77
- Master/Slave relationship 135
- Phase 77
- Set Style 75
- Slave position 77

CDD file 203

Characters 6

CimScope 3

Communication

    Communication Process 67

    Logical port 69

    Settings 67

Compile

    Compiling errors 71

    Program 71

Constants 29

CSV file 202

## D

Data

    Internal data access 108

    Monitor 108

    Parameter 108

    System variable 108

Debugging 7, 29, 31, 35, 36

Default scan rate 106

DeviceNet 103

## E

Error recovery 153

Expression 144, 166

Expression Builder 59, 62

    Calculations 62

    Creating 63

    Examples 65

    Logical expressions 62

External encoder 77

External Position 209, 212, 213

## F

FeedConstant 205

Formulas

    Accel and decel 127

    Position 128

    Speed 129

## G

Gains 11

Gearing 164

    Master/Slave offset 162

## H

Holding torque 160

Home key 60

Homing 142

    C channel 142

    Home input 142

## I

I/O 33

    Default names 33, 107

    Default output state 33

    Monitoring 48

    Type 33

Input

    bank 107

Install Wizard 3

## L

Limit

    Negative overtravel (NOT) 107

    Positive overtravel (POT) 107

Linear interpolation 203

## M

Machine Cycle 205

Math functions 62

Max Phase Value 205

Max Position Value 205

Mechatrolink 103

Memobus 34, 103

Move type

    Absolute 164

Relative 164

## O

Online 67

Output

bank 107

## P

Parameter 161

Phase 202

Phase Plotting 202

Position 202

Positioning range 142

Program 9

Active 7, 105

Autostart 7, 105

Definition 10, 105

Flow 103

High priority 106

Low priority 106

New program name 7

Number 7, 105

Priority levels 7

Program counter 105, 132, 166

Scan Type 7

Time sharing 105

Program Window 60

Canvas 60

Grid size 60

Positioning 60

Zoom range 60

Programmable limit switch 156

Project

creating 6

Downloading 72

Name 6

Properties

Acceleration 137, 147, 150, 154

Block ID 153

Deceleration 137, 147, 150, 154, 164

Default distance 150

Enabled 133, 140, 148, 160

Encoder 138, 156

Event 144

Expression 161

Home switch input 142

Homing direction 142

Latch axis 148

Latch finish distance 150

Latch start distance 150

Maximum position 156

Minimum position 156

Move type 154

Absolute 154

Relative 154

Rotary 154

Offset 162

Output 156

Position 154

Position Data 133

Position Data Table 74, 159

Program 153, 165

Range 127

Scale factor 159

S-curve 137

State 156

SyncPosition 133

Target distance 150

Time 162, 167

Timeout 142, 166

Timeout resolution 166

Title 132

Variable 161

Velocity 103, 137, 147, 150, 154

Wait for completion 105, 135, 150, 154

Properties Window 59

Docked 59

Undocked 59

## R

Registration

latch 148

latch input 107

Resolution 205

**S**

Scan-based system 105  
S-Curve 137  
Shrinkage 217  
Slave offset 220  
Slippage 217  
Stretch 217  
Subroutine 106, 132, 153  
    Maximum depth 106  
    New 11  
    Operation 106  
Synchronization 209  
System Requirements 3

**T**

Table 222  
Table units 208, 209  
Target position 137  
Template 177  
Torque mode 167

**V**

Variables  
    Array 34  
    Base address 29  
    Initial value 36  
    Keyword 37  
    Name 29  
    System 37  
    Tables 34  
    Type 29, 30, 36  
    Value 29

**YASKAWA ELECTRIC AMERICA, INC.**

2121 Norman Drive South, Waukegan, IL 60085, U.S.A.  
Phone: (847) 887-7000 Fax: (847) 887-7310 Internet: <http://www.yaskawa.com>

**MOTOMAN INC.**

805 Liberty Lane, West Carrollton, OH 45449, U.S.A.  
Phone: (937) 847-6200 Fax: (937) 847-6277 Internet: <http://www.motoman.com>

**YASKAWA ELETRICO DO BRASIL COMERCIO LTDA.**

Avenida Fagundes Filho, 620 Bairro Saude Sao Paulo-SP, Brasil CEP: 04304-000  
Phone: 55-11-5071-2552 Fax: 55-11-5581-8795 Internet: <http://www.yaskawa.com.br>

**YASKAWA ELECTRIC CORPORATION**

New Pier Takeshiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo, 105-6891, Japan  
Phone: 81-3-5402-4511 Fax: 81-3-5402-4580 Internet: <http://www.yaskawa.co.jp>

**YASKAWA ELECTRIC (SHANGHAI) CO., LTD.**

4F No. 18 Aona Road, Waigaoqiao Free Trade Zone, Pudong New Area, Shanghai 200131, China  
Phone: 86-21-5866-3470 Fax: 86-21-5866-3869

**BEIJING OFFICE**

Room No. 301 Office Building of Beijing International Club,  
21 Jianguomanwai Avenue, Beijing 100020, China  
Phone: 86-10-6532-1850 Fax: 86-10-6532-1851

**SHANGHAI OFFICE**

27 Hui He Road Shanghai 200437 China  
Phone: 86-21-6553-6600 Fax: 86-21-6531-4242

**SHANGHAI YASKAWA-TONJI M & E CO., LTD.**

27 Hui He Road Shanghai 200437 China  
Phone: 86-21-6533-2828 Fax: 86-21-6553-6677

**BEIJING YASKAWA BEIKE AUTOMATION ENGINEERING CO., LTD.**

30 Xue Yuan Road, Haidian, Beijing 100083 P.R. China  
Phone: 86-10-6232-9943 Fax: 86-10-6234-5002

**SHOUGANG MOTOMAN ROBOT CO., LTD.**

7, Yongchang-North Street, Beijing Economic Technological Investment & Development Area,  
Beijing 100076 P.R. China  
Phone: 86-10-6788-0551 Fax: 86-10-6788-2878

**YASKAWA ELECTRIC (HK) COMPANY LIMITED**

Rm. 2909-10, Hong Kong Plaza, 186-191 Connaught Road West, Hong Kong  
Phone: 852-2803-2385 Fax: 852-2547-5773

**YASKAWA ELECTRIC KOREA CORPORATION**

Kfpa Bldg #1201, 35-4 Youido-dong, Yeongdungpo-Ku, Seoul 150-010, Korea  
Phone: 82-2-784-7844 Fax: 82-2-784-8495

**YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.**

151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, Singapore  
Phone: 65-282-3003 Fax: 65-289-3003

**TAIPEI OFFICE (AND YATEC ENGINEERING CORPORATION)**

Shen Hsiang Tang Sung Chiang Building  
10F 146 Sung Chiang Road, Taipei, Taiwan  
Phone: 886-2-2563-0010 Fax: 886-2-2567-4677

**YASKAWA ELECTRIC TAIWAN CORPORATION**

Shen Hsiang Tang Sung Chiang Building  
10F 146 Sung Chiang Road, Taipei, Taiwan  
Phone: 886-2-2563-0010 Fax: 886-2-2567-4677

**YASKAWA ELECTRIC EUROPE GmbH**

Am Kronberger Hang 2, 65824 Schwalbach, Germany  
Phone: 49-6196-569-300 Fax: 49-6196-888-301 Internet: <http://www.yaskawa.de>

**MOTOMAN ROBOTEC GmbH**

Kammerfeldstrabe 1, 85391 Allershausen, Germany  
Phone: 49-8166-900 Fax: 49-8166-9039

**YASKAWA ELECTRIC UK LTD.**

1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, United Kingdom  
Phone: 44-12-3673-5000 Fax: 44-12-3645-8182

**MOTOMAN ROBOTICS EUROPE AB**

Box 504 S38525, Torsas, Sweden  
Phone: 46-486-48800 Fax: 46-486-41410