



Software Manual

Ladder

V230-21-G23 Rev: 12/04



No part of this document may be used for any purpose other than for the purposes specifically indicated herein nor may it be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and/or recording, for any purpose without written permission from Unitronics.

The information appearing in this document is for general purposes only. Unitronics makes no warranty of any kind with regard to the information appearing in this document, including, but not limited to, implied warranties of merchantability and/or fitness for a particular use or purpose. Unitronics assumes no responsibility for the results, direct and/or indirect, of any misuse of the information appearing in this document nor for any use of the Unitronics products referred to herein in any manner deviating from the recommendations made in this document. Unitronics assumes no responsibility for the use of any parts, components, or other ancillary appliances including circuitry other than as recommended hereunder or other than that embodied in the Unitronics product.

Unitronics retains all rights to its proprietary assets including, but not limited to its software products which are copyrighted and shall remain the property of Unitronics. Copyright protection claimed includes all Forms and matters of copyrightable materials and information legally allowed including but not limited to material generated from the software programs which are displayed on the screen of the Unitronics products such as styles, templates, icons, screen displays, looks, etc. Duplication and/or any unauthorized use thereof are strictly prohibited without prior written permission from Unitronics.

All brand or product names are used for identification purpose only and may be trademarks or registered trademarks of their respective holders.

Unitronics reserves the right to revise this publication from time to time and to amend its contents and related hardware and software at any time. Technical updates (if any) may be included in subsequent editions (if any).

VisiLogic Software Manual - Ladder

Table Of Contents

Ladder.....	1
Ladder Editor.....	1
Calls: Program Control.....	1
Calls, Jumps, and Labels.....	1
Program Sequencing: Modules, Subroutines, Labels & Jumps	2
Labels & Jumps	5
Call Subroutine	8
Subroutine: Return.....	8
Open a Subroutine	10
Ladder Nets with Feedbacks.....	11
Elements	12
Ladder Elements and Functions List.....	12
Placing a Ladder Element in a Net.....	15
Delete Elements	16
Change Element Type.....	17
Contacts	17
Direct Contacts	17
Inverted Contacts	18
Negative Transition Contact	18
Positive Transition Contact.....	19
Coils	20
Direct Coil	20
Inverted Coil	20
Reset Coil.....	21
Set Coil	21
Toggle Coil	21
Immediate Elements.....	21
Immediate: Read Physical Input	22
Immediate: Update High-speed Input	22
Immediate: Write to Output	24
Immediate: Write to Physical Analog Output.....	24
Operands.....	25
Operands	25
Linking Operands to Elements.....	26
Operand Addressing.....	27
Power-up Values.....	27
Constant Values #	27
Operand Types.....	28
Functions	49
Placing a Function in a Net.....	49
FBs Library	50
Compare Functions	52
Logic Functions	56
Math Functions	68
Store & Load Functions	80
Clock Functions	85
Vector Functions.....	104

Strings	119
Com.....	124
HMI Ladder Functions.....	138
Floating Math Functions	142
Interrupt Routines	146
Index.....	149

Ladder

Ladder Editor

Use the Ladder Editor to create the Ladder diagram that comprises your control application. Ladder diagrams are composed of contacts, coils, and function block elements arranged in nets.

In a Ladder diagram, the contacts represent input conditions. They lead power from the left Ladder rail to the right rail. This is why the first element in a net must always touch the left rail. Coils represent output instructions. In order for output coils to be activated, the logical state of the contacts must allow the power to flow through the net to the coil. This is why the elements in a net must be connected. Each net must contain only one rung.

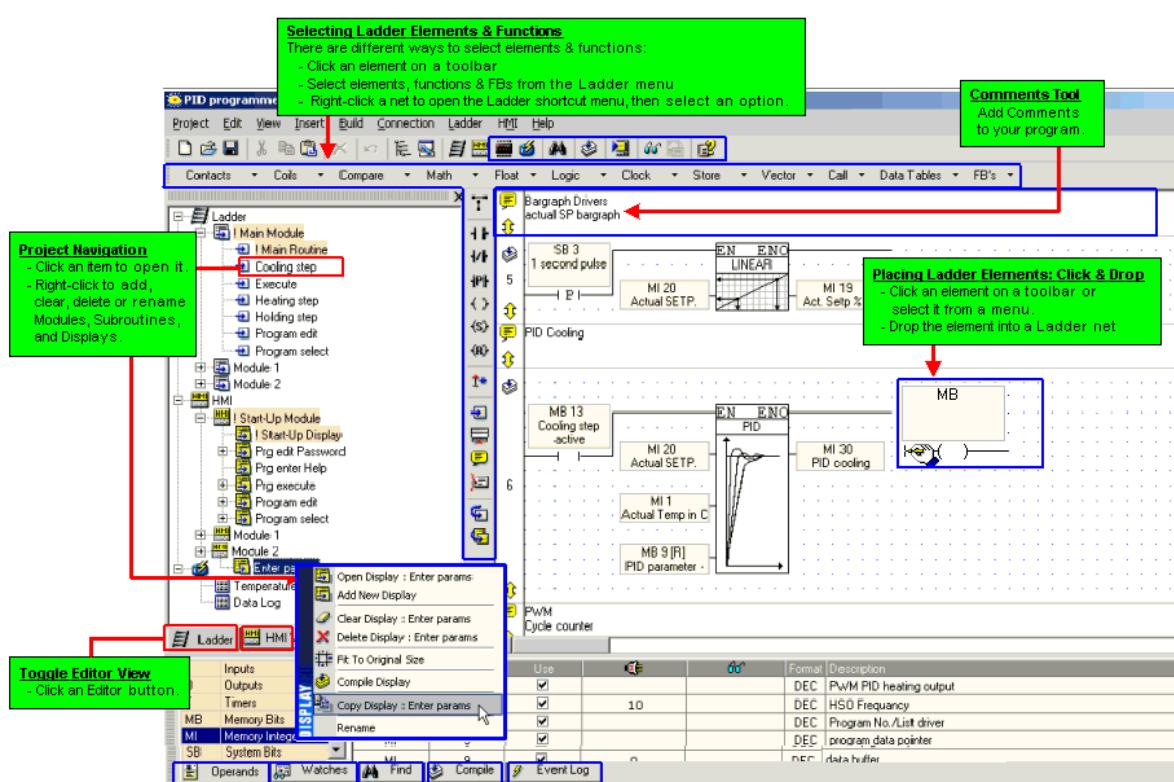
Use the Ladder Editor to:

- Place and connect Ladder Elements.
- Apply Compare, Math, Logic, Clock, Store, and Vector functions.
- Insert Function Blocks (FBs) into your program.
- Build program Modules and Subroutines, and use internal Subroutine Jumps and Labels.
- Place Comments on Ladder nets.

Ladder elements and functions may be dragged and dropped between nets. Hotkeys are also available for easy programming.

To start the Ladder Editor

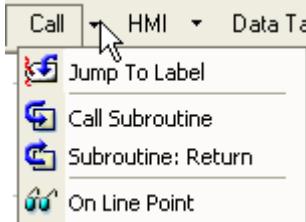
- Click the Ladder button  on the toolbar.



Calls: Program Control

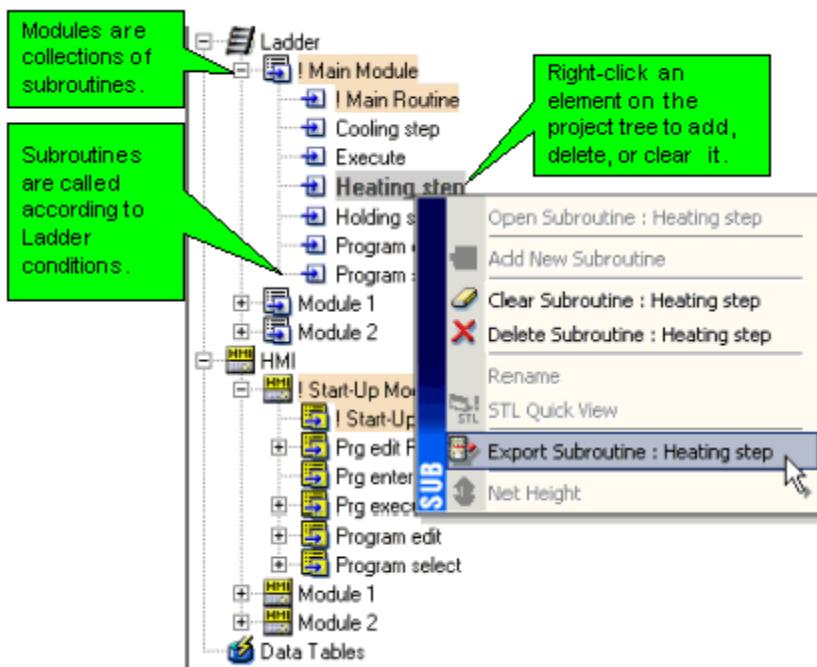
Calls, Jumps, and Labels

The Call menu's functions enable you to set the sequence in which your program runs.



Program Sequencing: Modules, Subroutines, Labels & Jumps

A module is a container of subroutines. Use modules and subroutines to divide your application into program blocks. You can then run these program blocks conditionally, from any point in your control application.



Note • Within the program tree, elements are presented alphabetically. This does not affect the order in which the program runs.

- Ladder Modules and subroutines can be moved via drag-and-drop, as can HMI Modules and Displays. Again, moving elements does not affect the order in which they run.
The Main Ladder Module, Main Subroutine, Start-up HMI Module and the Start-up HMI Display cannot be moved via drag-and-drop or erased. For easy identification, they are always marked in orange.

To control the Ladder program flow sequence and avoid loops, use the Call Subroutine function to conditionally call subroutines. Within a subroutine, you control the sequence by conditionally skipping over nets using Labels and Jump to Label functions. This enables you to shorten the program scan time.

A new VisiLogic project contains the main module and subroutine for the program. Each new subroutine contains a default number of nets and a Subroutine Return function.

Subroutines do not run if they are not called by Call Subroutine. If no Call Subroutine commands are included in the first subroutine of the main module, the program runs until it reaches the Subroutine Return function, and then jumps back to the beginning of the first subroutine.

Note • If a subroutine does not run, the coils in that subroutine will not be updated. For example, Subroutine 4 contains . If MB0 is turned ON in Subroutine 1, but Subroutine 4 is not called, O0 is not updated. The order in which I/Os are updated depend on the PLC program scan.

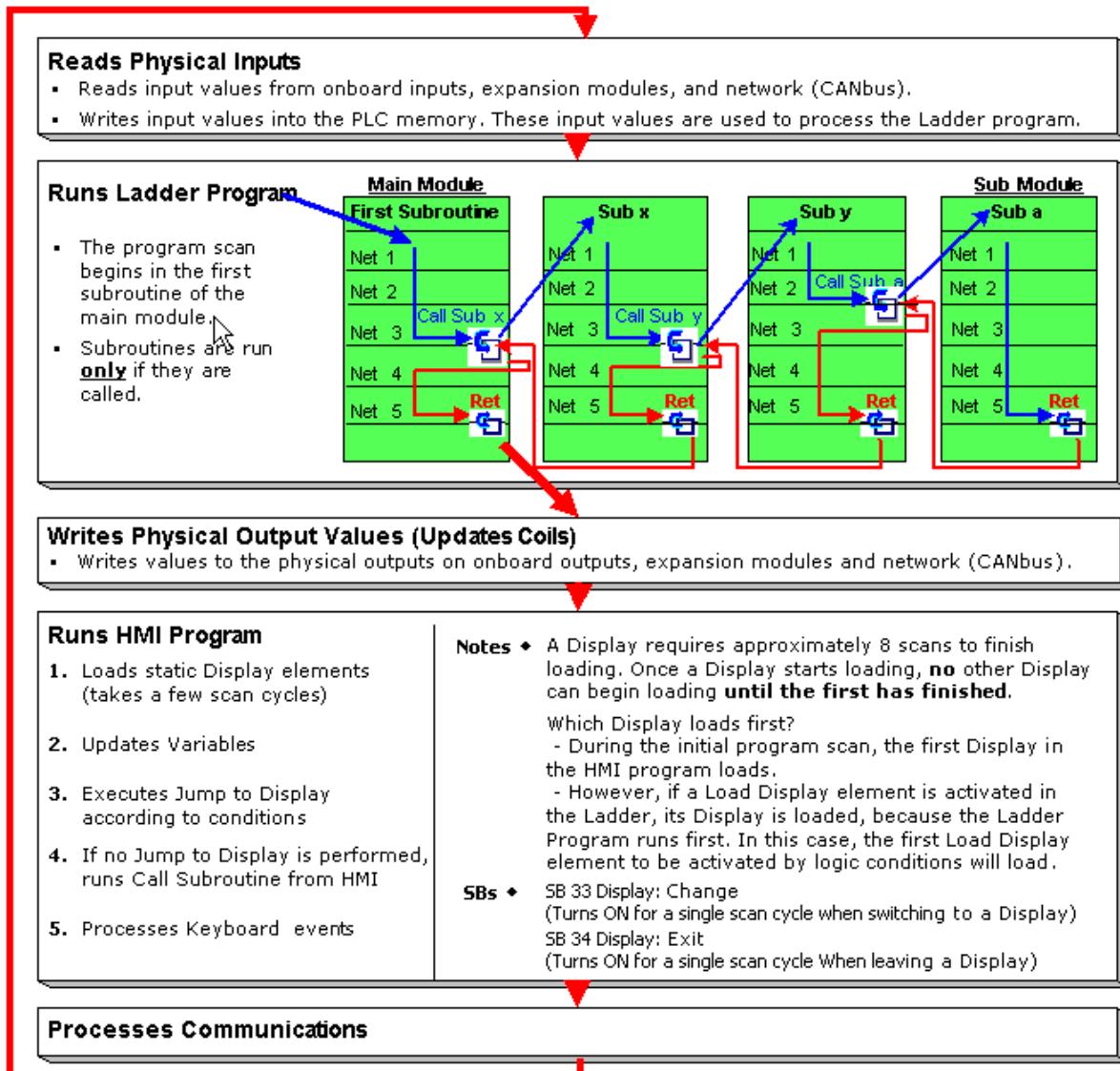
- Some FBs require Configuration, such as SMS. The FB Configuration should be placed in the first subroutine of the main program module. If a Configuration is in a subroutine that is not called into the program, linked FBs will not be processed even if the activating condition for that FB has been turned ON.

Subroutines can be reused as many times as required. Subroutines can also be exported and imported between projects.

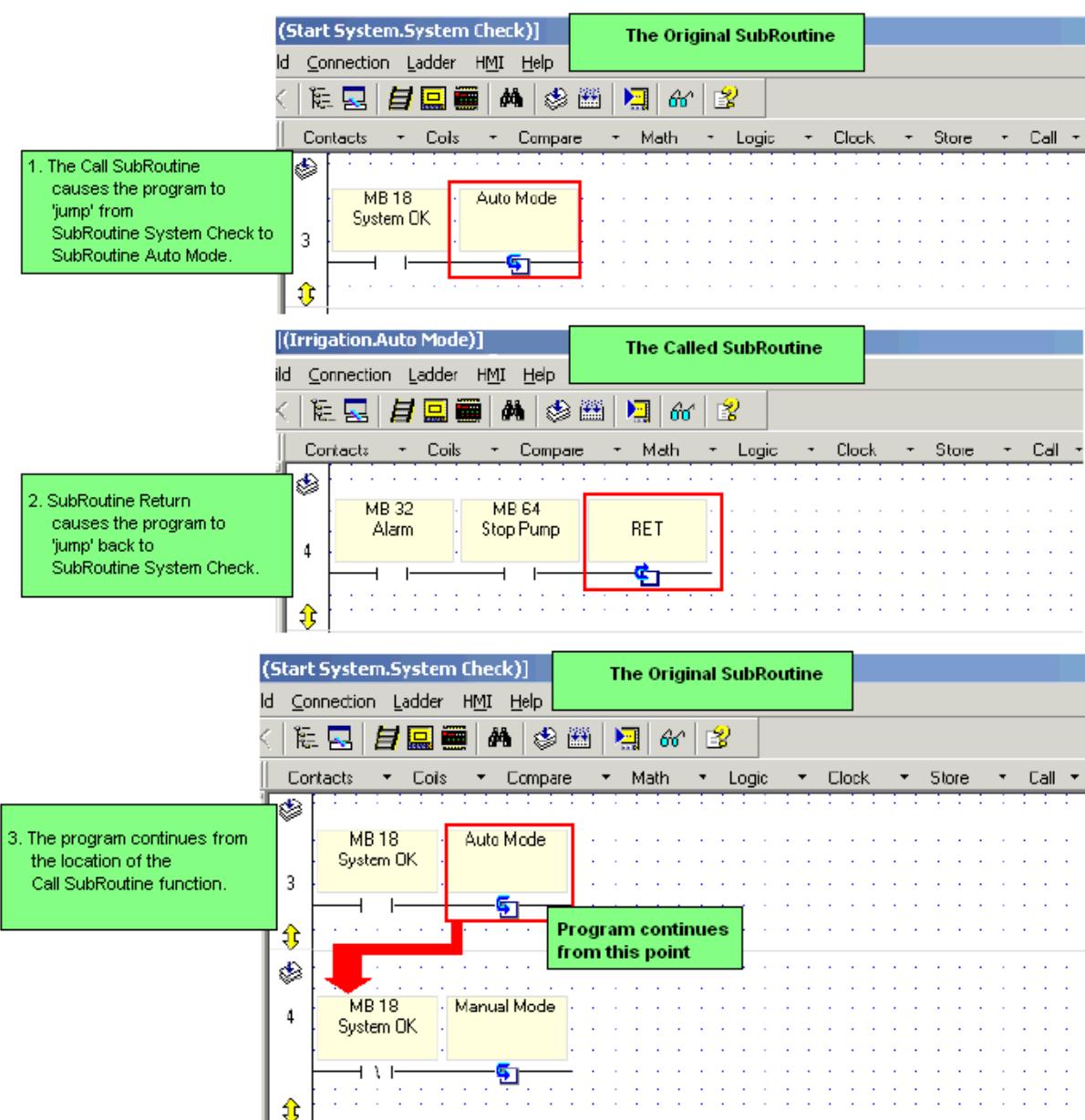
PLC Program Scan

A scan is a complete execution of the controller's entire program. The scan cycle is performed continuously.

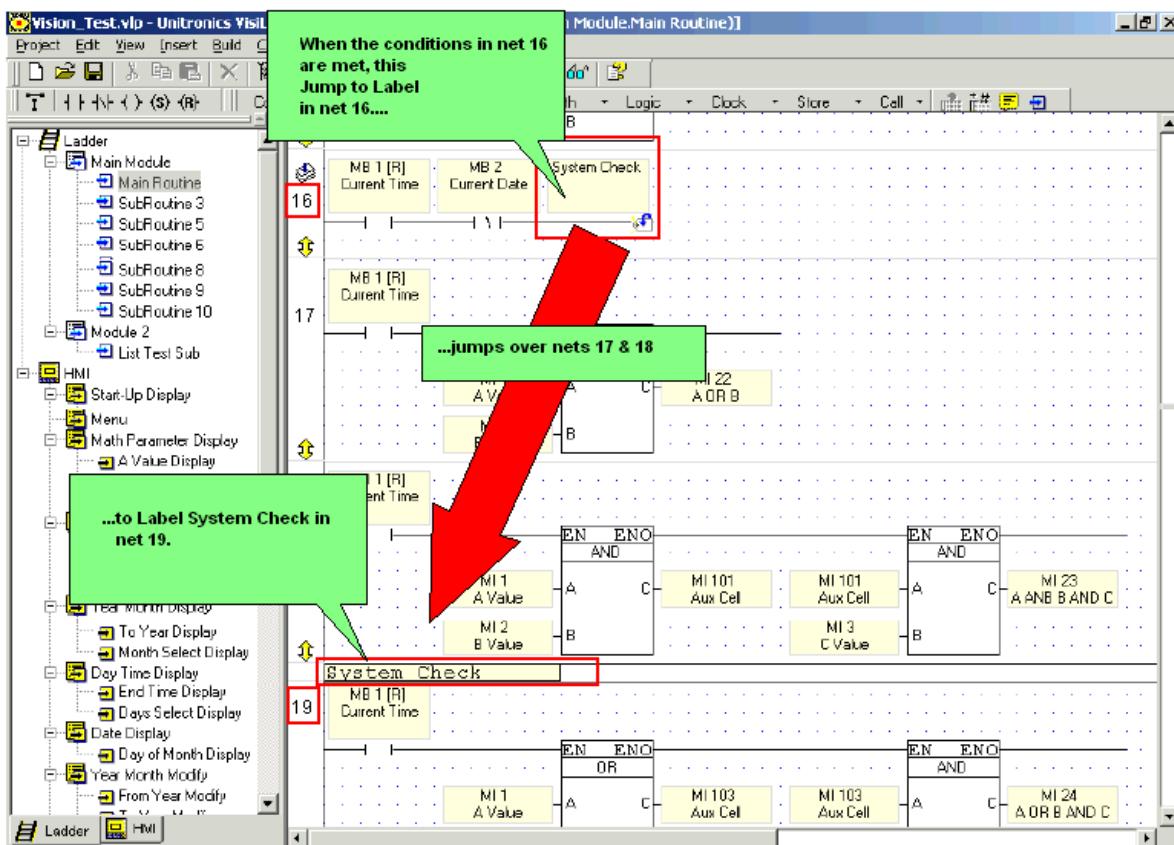
- | | |
|---------------|--|
| Note • | Power-up tasks, relating to the status of SB2 Power-up bit, are performed when the controller is turned on. These tasks are performed before the program scan. |
| • | The scan time is stored in SI 0 Scan Time, Resolution: Units of 10 mSec. |



Call Subroutine & Subroutine: Return



Within Subroutines: Labels & Jumps

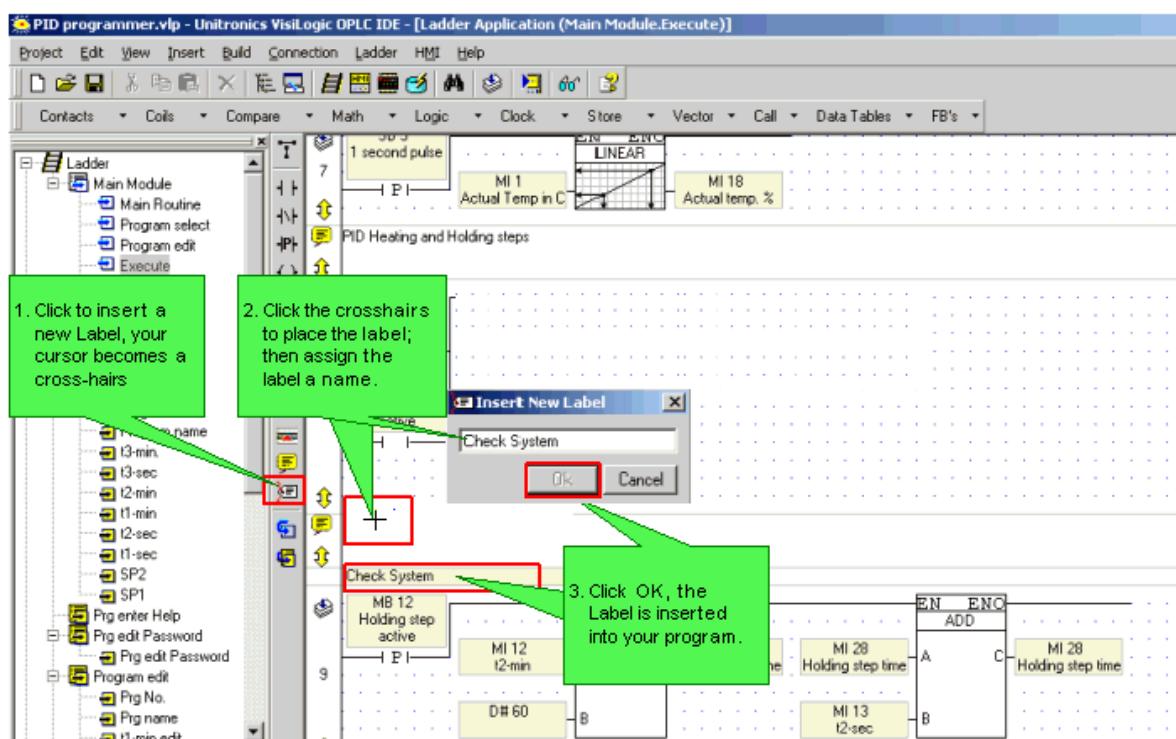


Labels & Jumps

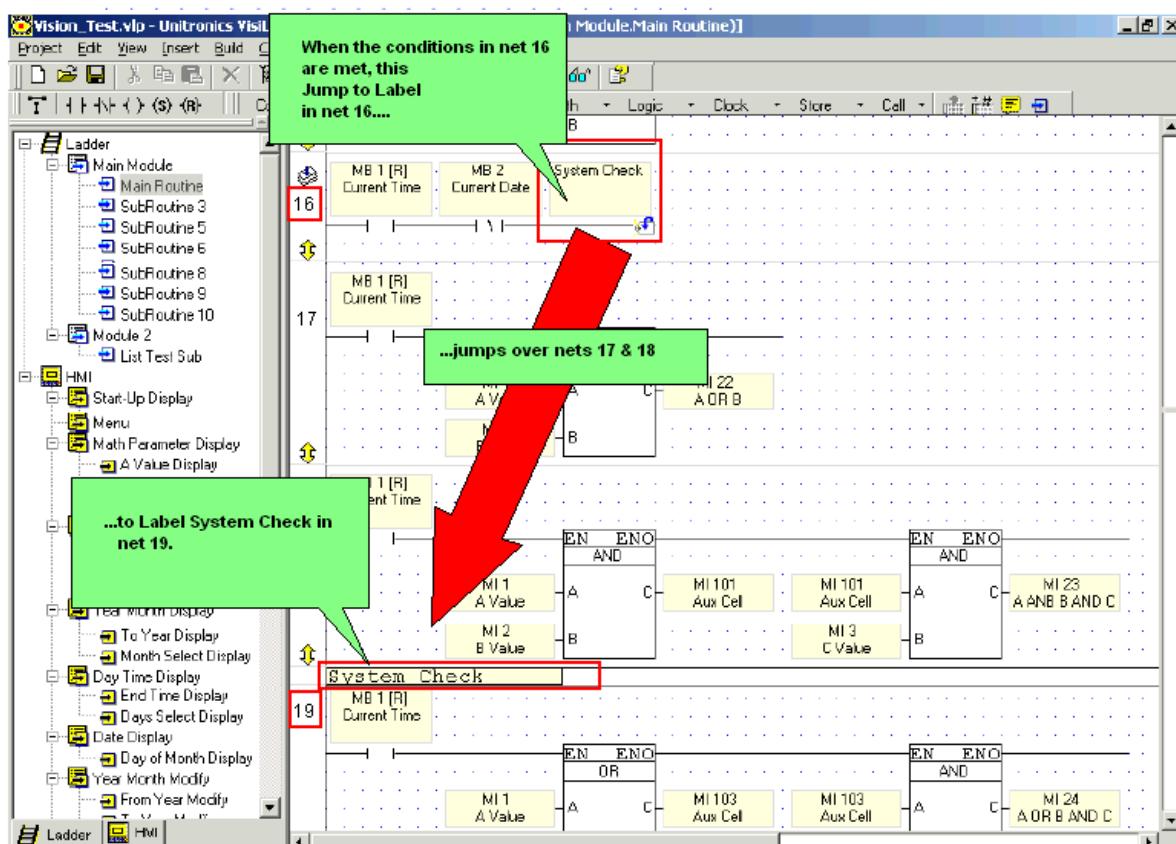
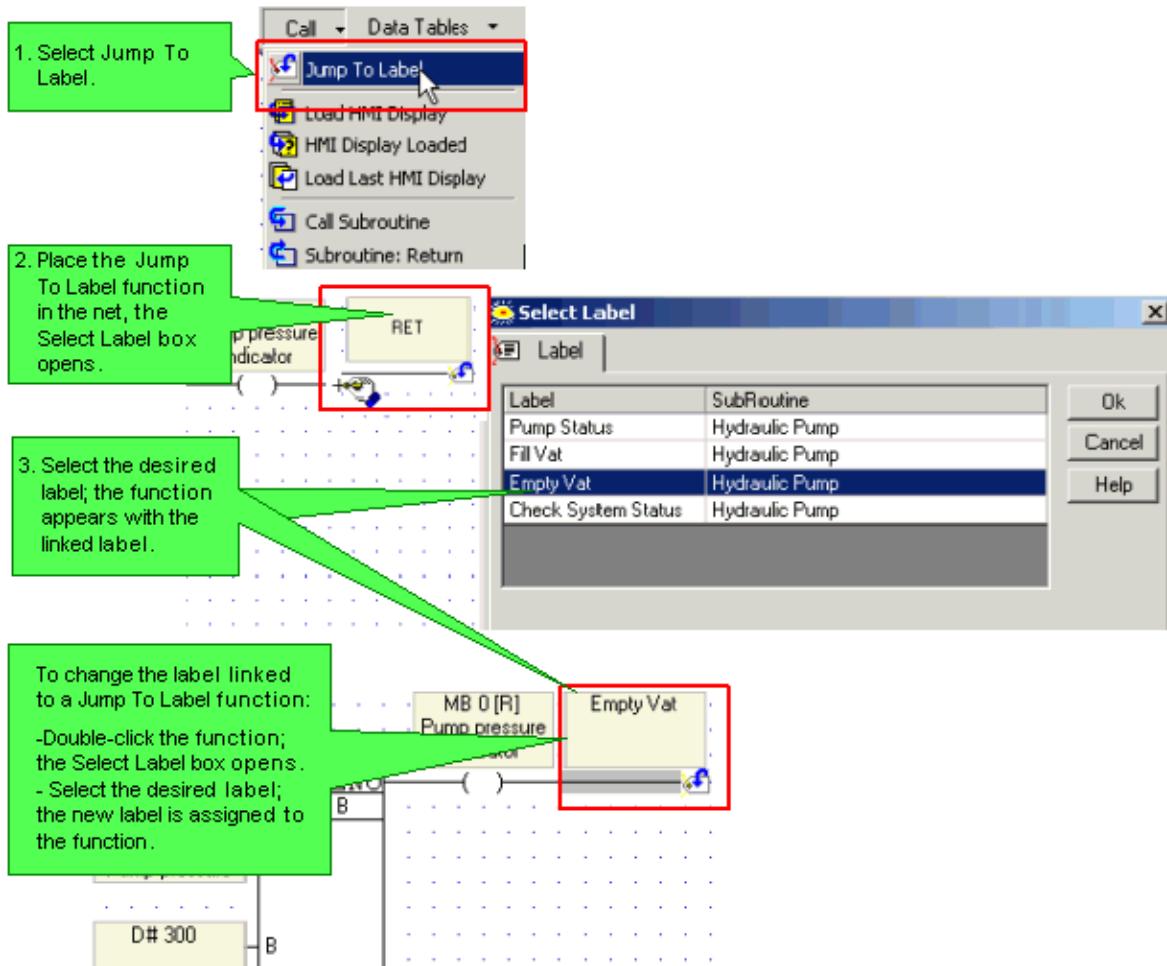
Labels enable you to jump over Ladder nets within a subroutine.

Using Labels

1. Place a Label in a net.

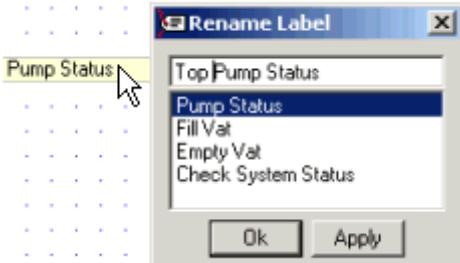


2. Create the condition that will cause the jump condition.
3. Place a Jump after the condition



Renaming Labels

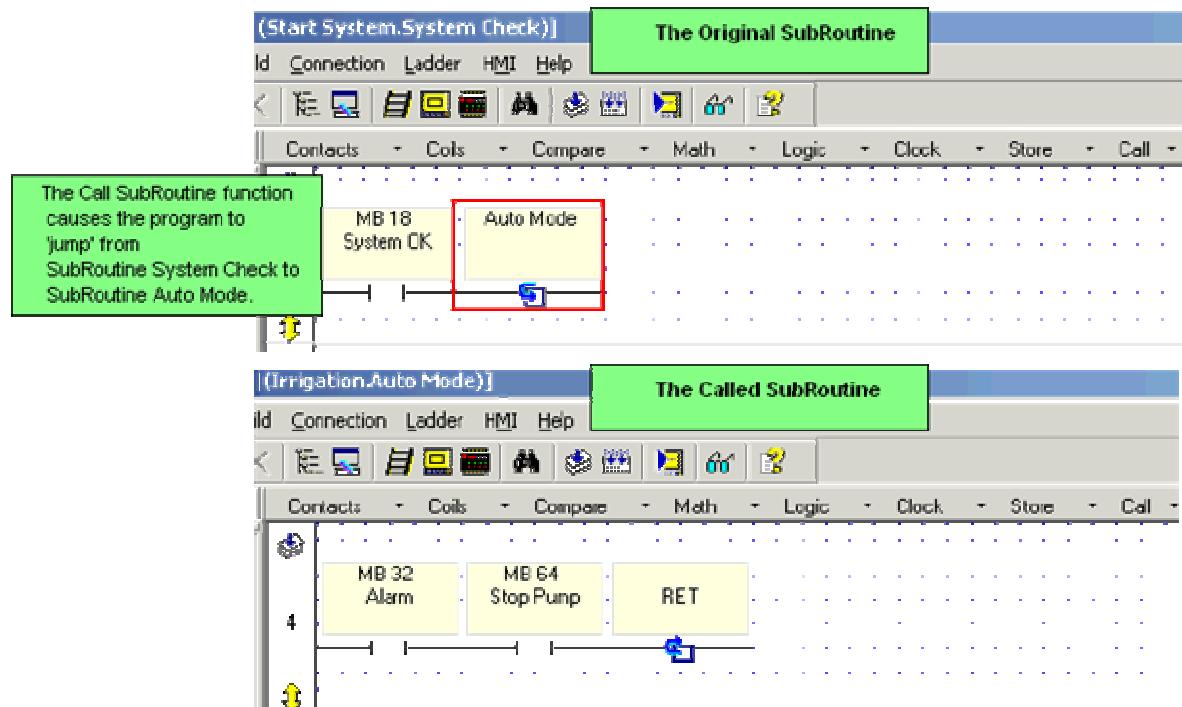
- To rename a Label, double click it, enter the new name and click Apply.



You can also use labels as bookmarks, by using them to mark program sections and then locating them using the Go To Label <Alt> + <Right/Left arrow> and List of Labels <Ctrl> + <L> utility.

Call Subroutine

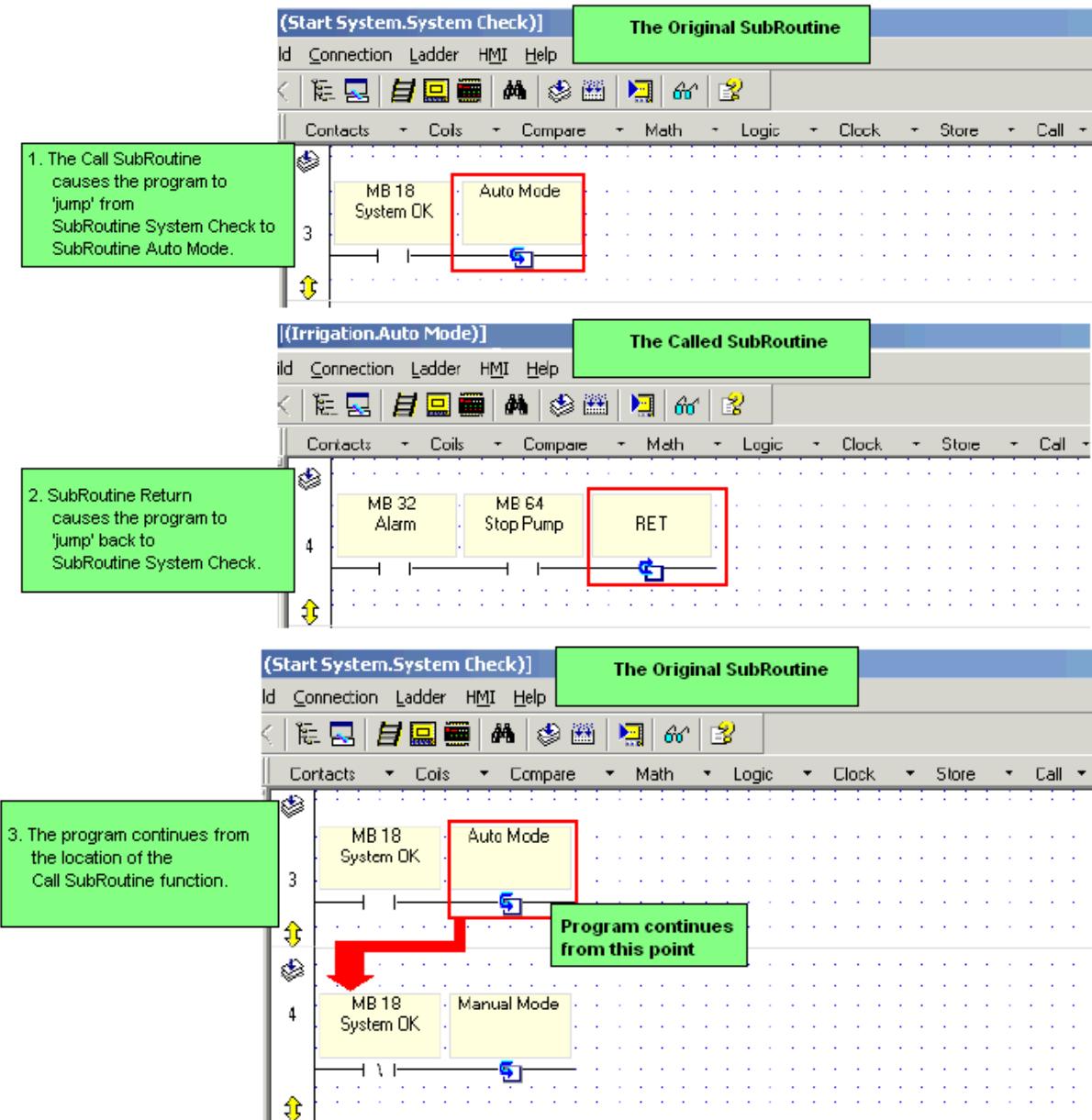
This function causes a subroutine to run in response to a Ladder Condition.



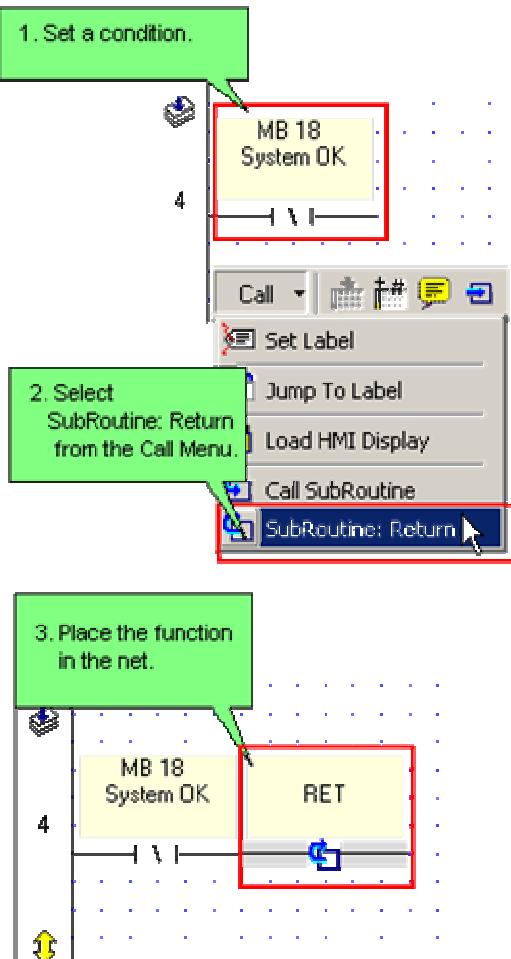
Using Call Subroutine

Subroutine: Return

A subroutine runs until it reaches a Subroutine Return function, and then jumps back to the beginning of the previous subroutine.



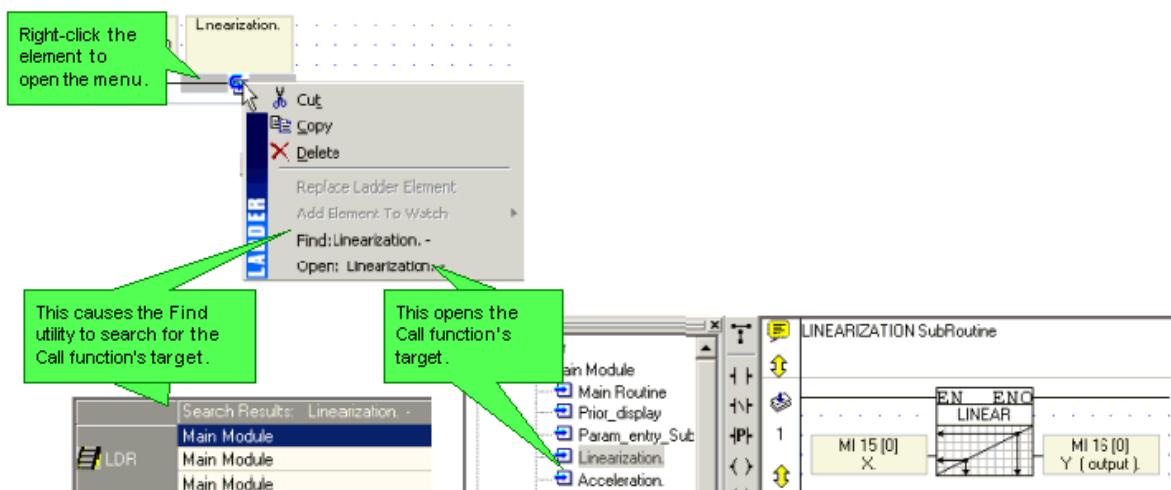
Using Subroutine Return



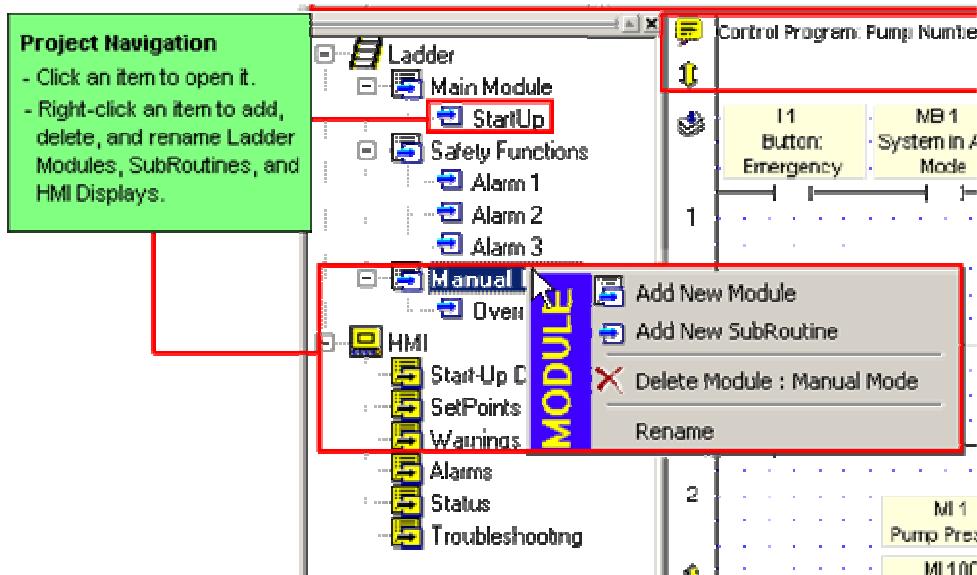
Open a Subroutine

To open a Subroutine for editing:

- Double-click in the Project Explorer tree, -or-
- Right-click the Subroutine in the Project Explorer tree, then select Open, -or-
- Right-click a Call Subroutine element to access the targeted subroutine.



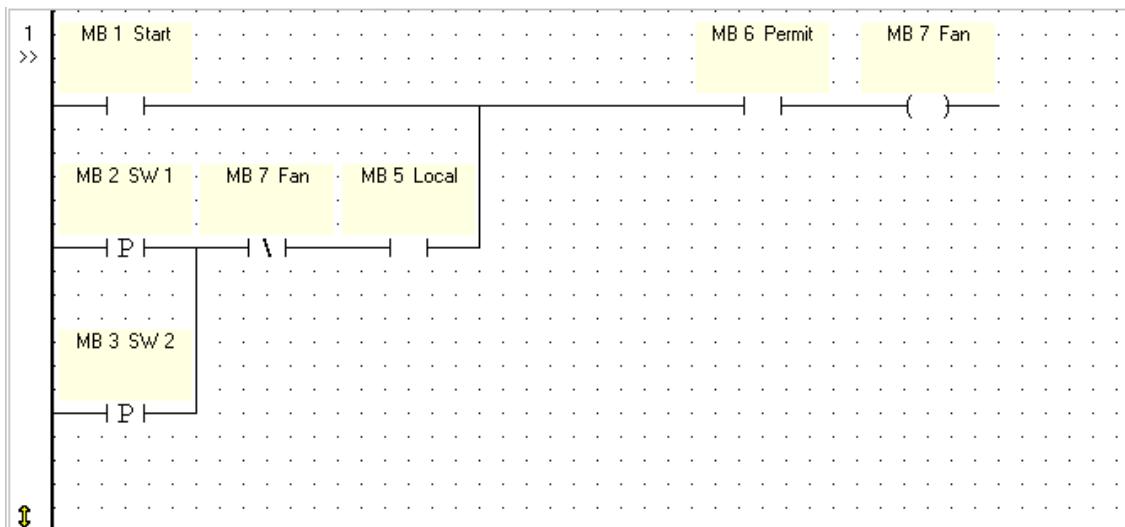
Name-Rename Modules and Subroutines



Ladder Nets with Feedbacks

According to IEC 1131 - 3, it is possible to create Ladder Diagram nets that contain feedback loops, i.e. where an element is used as both contact(s) and coil(s) in the same net.

In Ladder Diagram, all external input values such as those associated with contacts are gathered before each net is evaluated.



In the above example:

Where the net uses the state of its own output, the value of FAN (MB 7) coil associated with an inverted contact of MB 7 is always the value resulting from the previous evaluation.

However, if the value of FAN (MB 7) is used in any following nets, the latest evaluated state is used.

Elements

Ladder Elements and Functions List

Contacts

Direct Contact (NO)	
Inverted Contact (NC)	
Positive Transition (Rise)	
Negative Transition (Fall)	
Immediate: Read Physical Input	
Immediate: Update High-speed Input	

Coils

Direct Coil	
Inverted (negated) Coil	
Set Coil	
Reset Coil	
Toggle Coil	
Immediate: Write to Output	

Compare

Greater Than	
Greater/Equal	
Equal	
Not Equal	
Less/Equal	
Less Than	

Math

Add	
Subtract	

Multiply	
Divide	
Linearization, single value	
Linearization, vector	
Factor	
Power	
Square Root	
Increment/Decrement	

Floats

Basic: Store Direct, Add, Sub, Mul, Div, Abs

Extended: Square Root, Power, Exp, LN, Log10, A Mul (10^B)

Trig: Sin, Cos, Tan, ArcSin, ArcCos, ArcTan, Degrees, Radians

Compare: Greater Than, Greater Equal, Equal, Not Equal, Less Equal, Less Than

Convert: A+B/n, INV (A+B/n)

Logic

AND	
OR	
XOR	
Shift Left/Right	
Rotate Left/Right	
Bit Set/Reset	
Bit Test	
Store Bit Status	
Load Bit Status	
RS-SR Flip-Flop	

Clock

Time	
Day Of Week	

Day Of Month	
Month	
Year	
UTC (Universal Time) functions	

Store

Store Direct Function	
Store Indirect Function	
Store Timer/Counter Preset	
Load Indirect Functions	
Load Timer/Counter Preset	
Store Time/Counter: Current Value	
Load Timer/Counter: Current Value	

Vector

Load	
Load Timer Bit Value	
Store	
Find	
Fill / Fill Offset	
Copy / Copy Offset	
Compare / Compare Offset	
Bit to Numeric, Numeric to Bit	
Get Max	
Get Min	
Vector: Copy Memory	
Shift Byte Left	

Calls

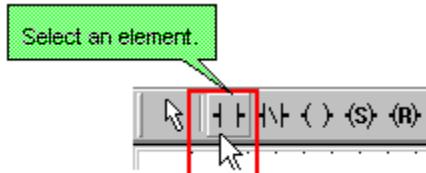
Jump to Label	
Load HMI Display	

HMI Display Loaded	
Load Last HMI Display	
Call Subroutine	
Subroutine Return	
Strings	
Transpose	
Num to ASCII	
Display RTC (ASCII)	
IP to ASCII	
Data Tables	
Read/Write	
Direct Read/Write	
Data Tables: Clear Table	
Data Tables: Find Row	
Immediate Elements	
Immediate: Read Physical Input	
Immediate: Update High-speed Input	
Immediate: Write to Output	
Immediate: Write to Physical Analog Output	

For information regarding advanced functions, such as MODBUS, check the topic FBs Library.

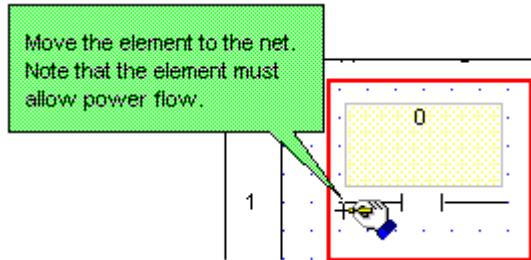
Placing a Ladder Element in a Net

1. Select any type of Ladder element by:
 - Clicking its icon on the Ladder toolbar, -or-

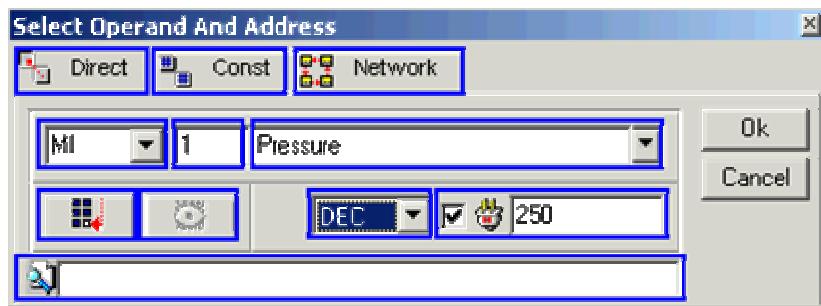


- Selecting it from the Ladder menu, -or-
- Right-clicking on the Ladder to display the Ladder menu and then selecting the element.

2. Move the element to the desired net location, then click.

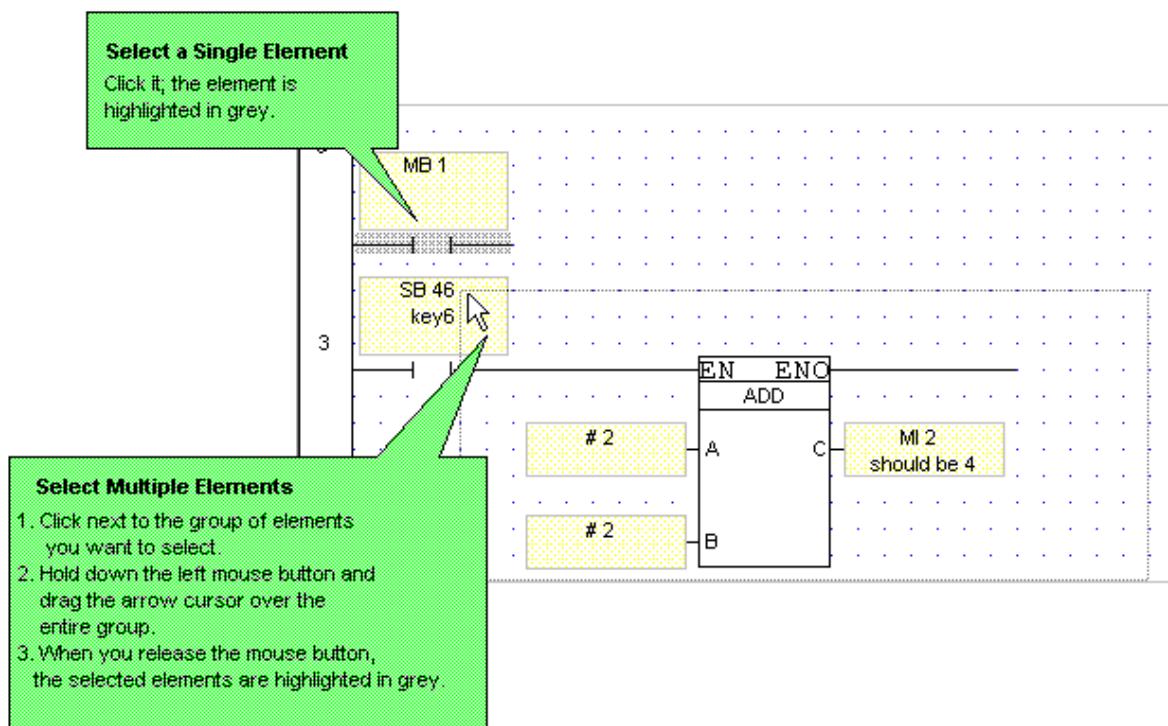


3. Link operands using the Select Operand and Address dialog box shown below.

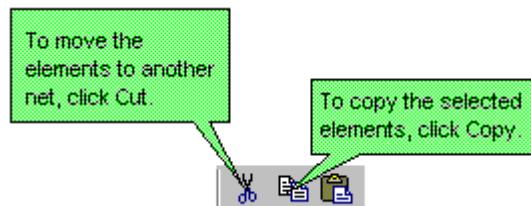


Delete Elements

Select the desired element(s), then



- Select Cut. -or-



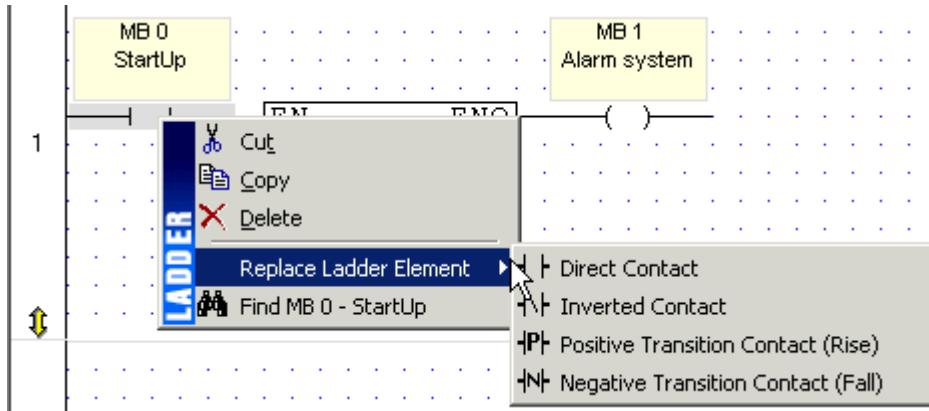
-or-
Select Cut or Copy from the Edit menu.

- Click the Delete button on the toolbar. -or
- Right-click the Element, then select Delete from the menu.

Change Element Type

To change an element type after it is placed in a net and linked to an operand:

- Right-click the element, select Replace Ladder Element, then select the appropriate element type.



After the element has been changed, it remains linked to the same operand.

You can use this method to change contact or coil types, to switch math and other function types while retaining the same input and output operands.

Contacts

A contact represents an action or condition. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Inputs
- Output
- Timer

Each contact condition in a net is loaded into the bit accumulator and evaluated to determine the coil (output or expression) condition. There are 4 types of contacts:

- Direct Contact
- Inverted Contact
- Positive Transition Contact (Rise or One Shot)
- Negative Transition Contact (Fall)

Contacts can be connected in series and in parallel on a Ladder net.

Direct Contacts

A Direct Contact is a normally open (NO) contact condition. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Output
- Timer

A door buzzer is an example of a Direct Contact. When you push the buzzer, power flows through the circuit and the buzzer sounds. When you release the buzzer, the sound stops.

During the system scan, the processor evaluates the program elements net by net.

If the Direct Contact bit operand (the door buzzer) is OFF (logic 0): power will not flow through the Direct Contact. The door buzzer is silent.

If the Direct Contact address (the door buzzer) is ON (logic 1): power will flow through the Direct Contact. The door buzzer sounds.

Inverted Contacts

An Inverted Contact represents a normally closed contact condition. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Output
- Timer

An Inverted Contact condition can be from an external input device (for example: a push button) or from an internal input system element (for example: SB 50 Key +/- is pressed).

An emergency light contains an example of an Inverted Contact.

- Normally, there is power flow through the emergency light's Inverted Coil and the light stays off.
- During an electric power outage, the power flow through the Inverted Coil stops and the emergency light comes on.

During the system scan, the processor evaluates the program elements net by net.

If the Inverted Contact address (power supply) is ON (logic 1): power will flow through the Inverted Contact. The emergency light will stay off.

If the Inverted Contact address (power supply) is OFF (logic 0): power will not flow through the Inverted Contact. The emergency light comes on.

If the power outage ends and power flow is returned to the Inverted Contact, it will close again and the emergency light will go off again.

Negative Transition Contact

A Negative Transition Contact gives a single one-shot pulse when the bit operand it is linked to falls from ON (logic 1) to OFF (logic 0). A Negative Transition Contact registers the fall in status from ON to OFF. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Output
- Timer

A computer ON/OFF button is an example of a Negative Transition Contact. The computer is ON.

If you push the ON/OFF button in without releasing it, the computer will not shut down. But when you release the button, the system registers a change in status from ON to OFF. The computer then shuts down.

During the system scan, a Negative Transition Contact address is evaluated for a transition from ON to OFF. A transition allows power to flow through the Negative Transition Contact for one scan.

At the end of a scan, the Negative Transition Contact is reset to OFF (logic 0). The Negative Transition Contact can only be re-activated when the triggering signal again changes from ON to Off.

Note • A maximum of 255 Rise/Fall elements is allowed in a project. To ascertain how many elements of each type are in the project, use the Find Element utility on the Edit menu. Search for Positive Transition Contact and Negative Transition Contact. The sum of the results must not exceed 255. If a program exceeds this number, Error 1017 results.

Positive Transition Contact

A Positive Transition Contact gives a single one-shot pulse when its reference address rises from OFF (logic 0) to ON (logic 1). A Negative Transition Contact registers the fall in status from OFF to ON. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Network System Bit
- Network System Input
- Output
- Timer

A cellular phone keypad key is an example of a Positive Transition Contact. When you push a key a number is displayed on the screen. It does not matter if you push the key quickly or hold it down for several seconds. The number will only appear once on the screen.

The cellular phone registers the transition from key NOT pressed to key pressed. The length of time the key is pressed is not relevant. You must release the key and press it again to repeat the number on the cellular phone screen.

During the system scan, a Positive Transition Contact address is evaluated for a transition from OFF to ON. A transition allows power to flow through the Positive Transition Contact for one scan.

At the end of a scan, the Positive Transition Contact is reset to ON (logic 1). The Positive Transition Contact is re-activated when the linked signal turns from OFF to ON.

Note • A maximum of 255 Rise/Fall elements is allowed in a project. To ascertain how many elements of each type are in the project, use the Find Element utility on the Edit menu. Search for Positive Transition Contact and Negative Transition Contact. The sum of the results must not exceed 255. If a program exceeds this number, Error 1017 results.

Coils

A Coil represents a result or expression of an action. A coil turns ON when the preceding net conditions are ON, allowing power flow to reach the coil from the net. If the preceding net conditions are OFF, a coil turns OFF. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Output
- Timer

Each contact condition is evaluated in a net to determine the coil (result or expression) condition. Coil types include:

- Direct Coil
- Inverted Coil
- Set Coil
- Reset Coil
- Toggle Coil

Note • | Do not energize a coil more than once in a program.

Direct Coil

An Direct Coil turns ON when the preceding net conditions are ON, allowing power flow to reach the coil from the net. If the preceding net conditions are OFF, an direct coil turns OFF. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Output
- Timer

The coil can represent an external output device (for example: alarm bell) or to an internal system element, as for example, SB 41, which is key #1 on the controller's keyboard..

Inverted Coil

An Inverted Coil turns OFF when the preceding net conditions are ON, allowing power flow to reach the coil from the net. If the preceding net conditions are OFF, an inverted coil turns ON. You can link an Inverted Coil to an:

- Memory Bit
- System Bit
- Output
- Timer

The coil can represent an external output device (for example: alarm bell) or to an internal system element, as (for example, SB 4 Divide by 0).

To place a coil in a Ladder net:

1. Click a Coil icon on the toolbar.
2. Move your cursor to the desired location in the net, then click.
3. The coil drops into place.

Reset Coil

A reset coil turns a set coil OFF (unlatches), when the preceding net conditions are ON, allowing power flow to reach the reset coil from the net.

- Note •** Once a set coil is turned ON, it stays ON, independent of the original set condition, until a reset coil linked to the same address resets (unlatches) the coil condition.

You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Output
- Timer

Do not use a set coil without a reset coil in a program.

Set Coil

A set coil separates the coil from the action or condition that energized the coil. Once energized, a set coil's result is no longer dependant on the action that energized it. A set coil stays energized (latched) until its condition is reset (unlatched) by a reset coil. You can link it to any of the following bit operands:

- Memory Bit
- System Bit
- Output
- Timer

An example of a set coil is an overhead light. When you turn on a light, it stays lit until you turn it off (reset or unlatch it) or the light bulb burns out. You do not have to hold the light switch to keep the light on.

An example of a coil that you do not want to be set (latched) is a car horn. You expect it to toot only when you press on the horn button and you expect it to stop when you stop pressing on the horn button.

Do not use a set coil without a reset coil in a program.

Toggle Coil

A toggle coil changes its state when it is activated. You can link it to any of the following bit operands:

- Memory Bit
- Output

Toggle Coil is fast;the execution time is shorter than Reset Coil.

An example of a toggled coil is an light switch. When you turn on a light, it stays lit until you toggle it; it then turns off. The light stays off until you toggle it back on.

Immediate Elements

Immediate elements are located on the More> Immediate menu.

Generally, I/Os values are read and written to according to the PLC program scan.

Immediate elements immediately update the current value of I/Os--without regard to the program scan. This enables you:

- Write values to inputs, and use the new input value to execute the rest of the PLC program.
- Turn outputs ON, as for example in an emergency routine.

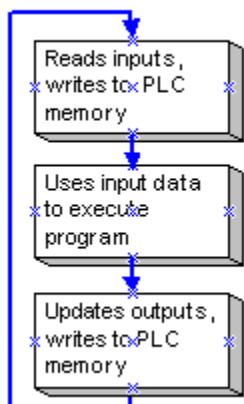
If your program requires you to immediately update an I/O value, use Immediate elements in conjunction with Interrupt routines.

Immediate: Read Physical Input

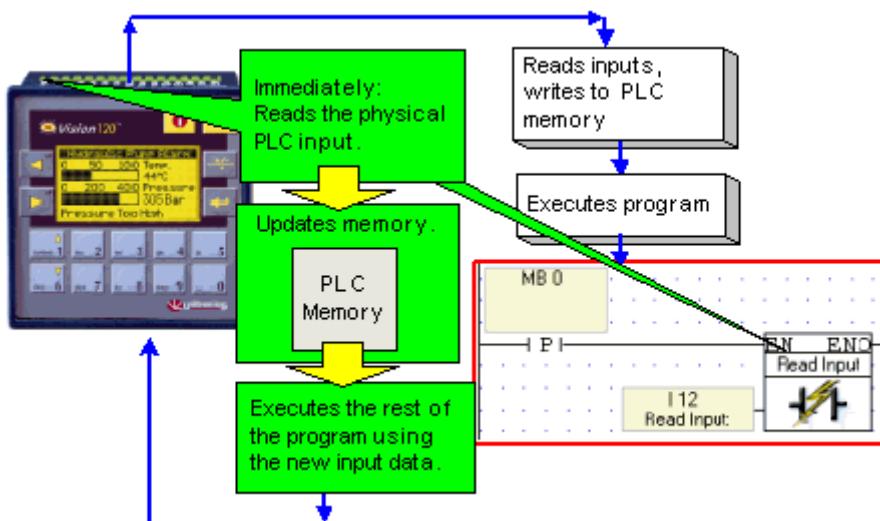
Read Physical Input is located on the More> Immediate menu.. This element can be used to immediately read the current status of a physical, hardwired input and use the new input status to execute the PLC program.

Ordinarily, a PLC program runs like this:

Program Scan



When the program encounters Read Physical Input, the program immediately reads the physical PLC input, updates the PLC memory, and executes the rest of the program using the new input data.

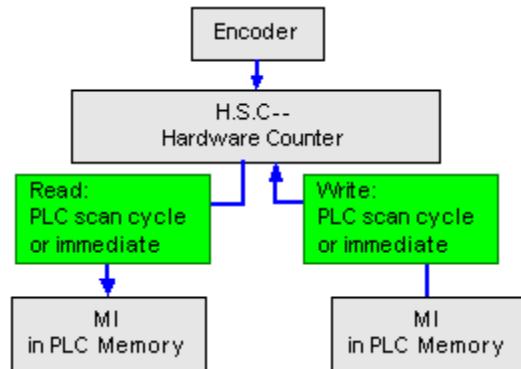


To use Read Physical Input, place it in a net after an activating condition and select the desired input.

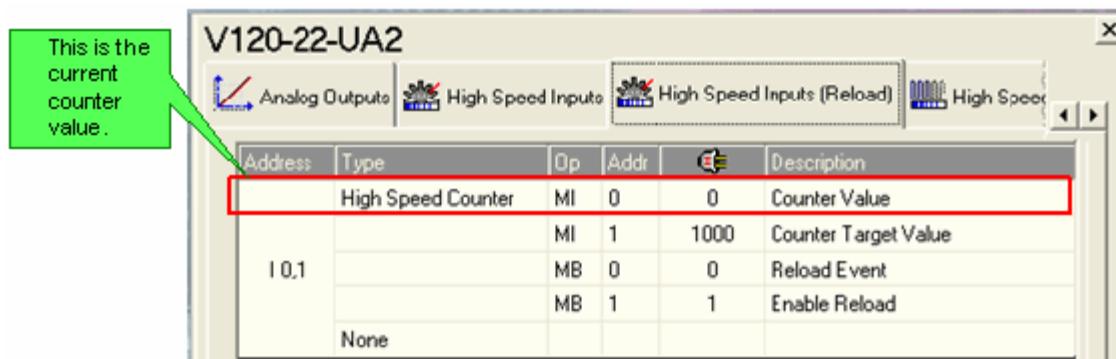
Note • Within a net, Read Physical Input should stand alone except for its activating condition.

Immediate: Update High-speed Input

Update High-Speed Input is located on the More> Immediate menu. This element can be used to immediately update the current value of a physical, hardwired high-speed input--without regard to the program scan--and use the new input value to execute the PLC program.

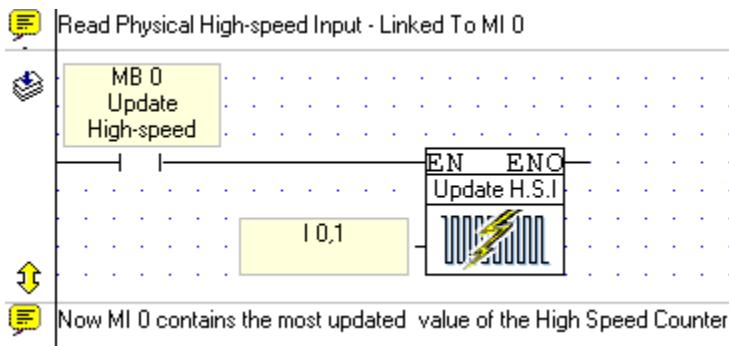


When the program encounters Update High-Speed Input, the program immediately compares the actual, current input value against the value in the MI linked to the input.

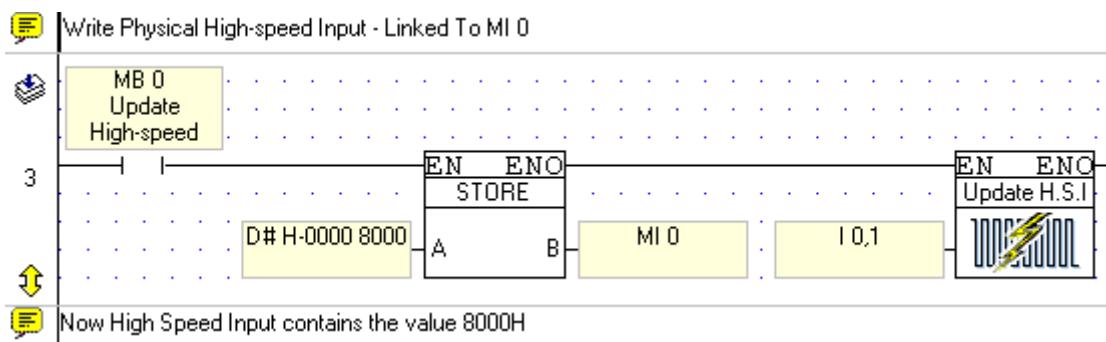


If the values are not equal, the MI is updated with the current input value; the rest of the program executes according to the new input data.

To use Update High-Speed Input, place it in a net after an activating condition and select the desired input.



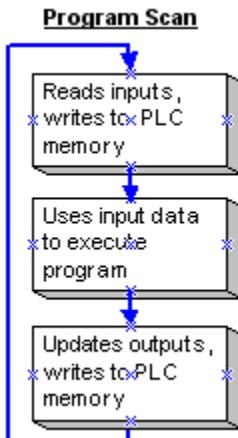
Note • Within a net, Update High-Speed Input should stand alone except for its activating condition.



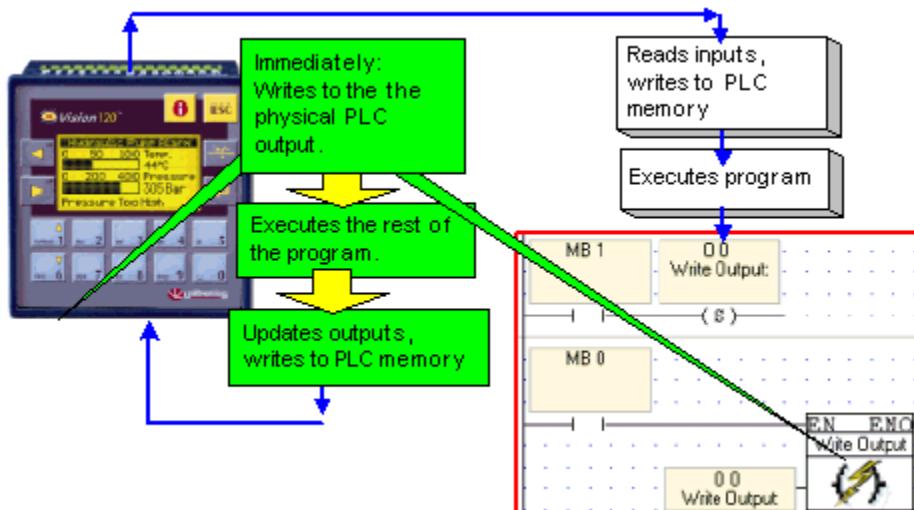
Immediate: Write to Output

Write to Output is located on the More> Immediate menu. This element can be used to immediately update the status of a physical, hardwired output.

Ordinarily, a PLC program scan runs like this:



When the program encounters Write to Output, the program immediately writes the physical PLC output, then executes the rest of the program.



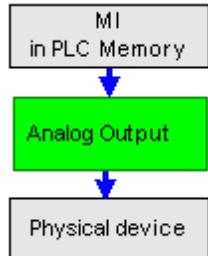
To use Write to Output, place it in a net after an activating condition and select the desired output.

Note •	Within a net, Write to Output should stand alone except for its activating condition.
•	If, after Write to Output has been executed, the same output is updated as the rest of the program runs, the last update is the one written to the PLC memory at the end of the program scan.

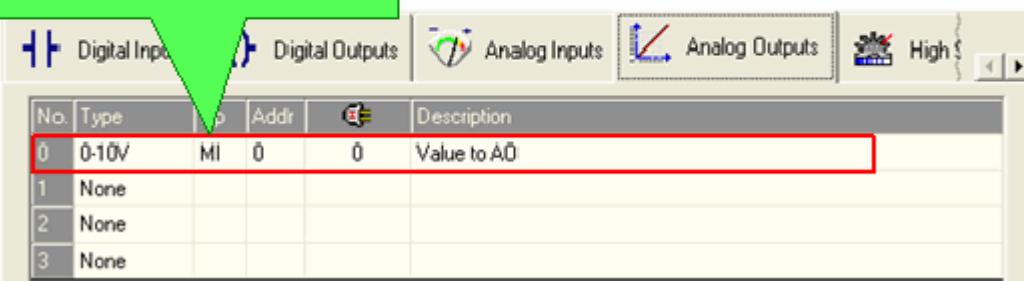
Immediate: Write to Physical Analog Output

Write to Physical Analog Output is located on the More> Immediate menu. This element can be used to immediately write a value into a physical, hardwired output--without regard to the program scan.

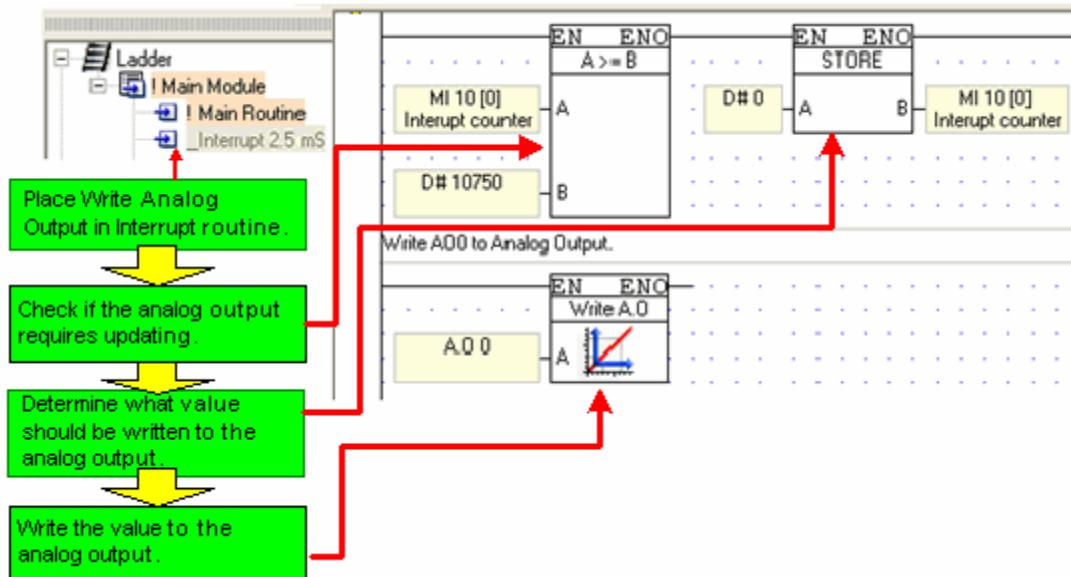
This function is generally included in an Interrupt routine, for example to turn an output ON in case of an alarm or emergency.



When Immediate Write is called, the value in the linked MI is immediately written to the physical analog output



Note • Within a net, Write to Physical Analog Output should stand alone .



Operands

Operands

Ladder elements and functions are linked to operands. Operands contain data. The Ladder elements and functions determine the way that operand data is used in your program. Every Operand has an Address and a Description. When you select a Ladder element and place it in a net, the Select Operand and Address box opens, enabling you to link an Operand type, select an address, and assign a description.

To View Operand Lists

1. Select the Operand tab at the bottom of the Output Window; the operands are displayed.
2. Click an operand type in the left pane; a list of that operand type is displayed.

Note that you can edit values and descriptions in the Output Window.

Operand Types and Symbols

Type	Symbol	Quantity	Value	Address Range
Input	I	544	Bit	I0-I543
Output	O	544	Bit	O0-O543
Timer	T	192	32-bit	T0-T191
Counters (C)	c	24	16-bit	C0-C24
Memory Bit	MB	4096	Bit	MB0-MB4095
Memory Integer	MI	2048	16-bit	MI0-MI2047
Memory Long Integer	ML	256	32-bit	ML0-ML255
Double Word (unsigned)	DW	64	32-bit	DW0-DW63
Memory Floating Point Integer	MF	24	32	MF0-MF24
Constant Value	#	Dynamic		Dynamic

System Operands

System Operands are connected to certain functions and values in the controller's operating system.

Type	Symbol	Quantity	Value	Address Range
System Bit	SB	512	Bit	SB0-SB511
System Integer	SI	512	16-bit	SI0-SI511
System Long Integer	SL	56	32-bit	SL0-SL63
System Double Word (unsigned)	SDW	64	32-bit	

Network Operand Types and Symbols

If a controller is networked, the following operands are accessible to other controllers:

Type	Symbol	Quantity	Value	Address Range
Network System Bit	NSB	8	Bit	SB200-SB207
Network Input	NI	17	Bit	I0-I16
Network System Integer	NSI	2	16-bit	SI200-SI201

Linking Operands to Elements

When you place a Ladder element or function on a net, the Select Operand and Address dialog box opens. All of the operands and operand types that are displayed in the Select Operand and Address dialog box are applicable to the element or function that you have selected. To edit an operand attached to an element, you can also double-click on the yellow Description field of an element after it has been placed in the Ladder.

You can search for a particular operand by using the Search: Symbolic Name function at the bottom of the dialog box.



Operand Addressing

An Operand Address is the physical location in the controller memory where the data is stored.

For example:

- MB 10 - "10" is the address of the MB Operand
- MI 35 - "35" is the address of the MI Operand
- T 12 - "12" is the address of the Timer Operand

You can also assign descriptions to the operands you use in your application.

Power-up Values

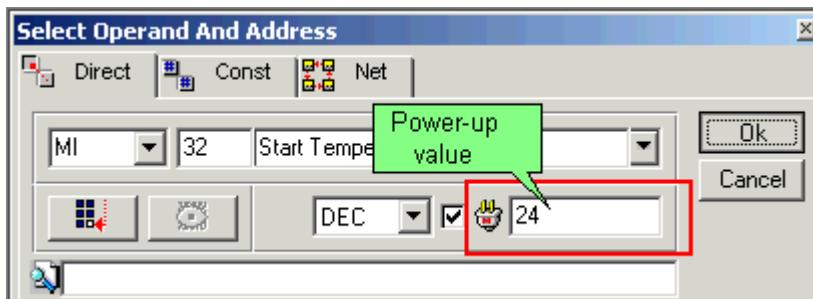
Power-up values can be assigned to most operands. These values are written into the operands when the controller is turned on.

Bit operands can be SET or RESET. Integers, Long Integers, and Double Words can be assigned values that are written into the operand at power-up.

You can assign Power-up Values in the:

- Select Operand and Address Dialog Box

Check the box next to the plug-shaped icon. This enables you to enter a value in the Power-up value fill-in field.



- Operand View Window

1. Select the Operand tab at the bottom of the screen.
2. Click on the Operand type to display the list of operands.
3. Enter Power-up values in the column headed by the Power-up icon.

	Operand	Address	In Use	Power-up value
MI	Memory Integer	0	<input checked="" type="checkbox"/>	24
SI	System Integer	1	<input checked="" type="checkbox"/>	1024
ML	Memory Long			

Constant Values

A Constant Value is an integer number, either signed or unsigned, that is created by the programmer. Constant Values are symbolized by a number sign.

To use a Constant Value in your program, select the Constant option in the Select Operand and Address dialog box and enter a number.

You can also select the unsigned integer option.



When entering the value, you can toggle to Hex via
<CTRL> + <H>.

Constant Value Operands

You can create a list of named Constant Value Operands in the Output Window at the bottom of the screen.

1. Select the Constant tab in the Output Window; the list of Constant Values opens.
2. Enter a Description and a Value; note the Unsigned option.
3. Create a new Constant Value by pressing Enter.

When you create a Constant Value in this way, the program references the value by the description.

By entering the Constant Value's description in the Select Operand and Address dialog box, you can use this Constant Value in your application.

Operand Types

Memory Bits (MB)

Memory Bits are bit operands (0 or 1).

There are 4096 MBs, address MB 0 - MB 4095.

To display a list of operands, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list

Inputs (I)

Inputs are bit operands (0 or 1).

The number of Inputs varies according to the Snap-in I/O Modules and I/O Expansion Modules you integrate into your system.

An Input is an actual hardwired input connection into the controller.

To display a list of operands, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list

Outputs (O)

Outputs are bit operands (0 or 1).

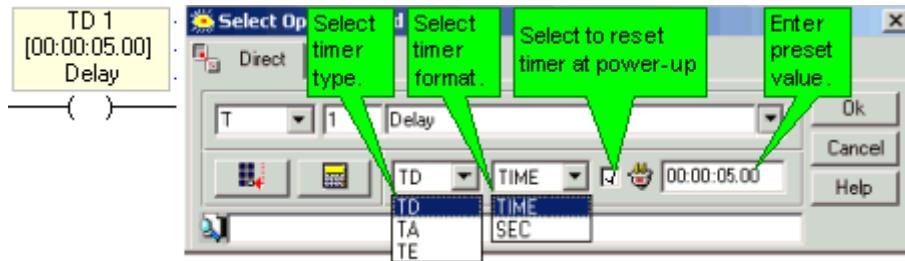
The number of Outputs varies according to the Snap-in I/O Modules and I/O Expansion Modules you integrate into your system.

An Output is an actual hardwired output connection from the controller.

To display a list of operands, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list

Timers (T)

To use a timer in your program, place an element in a net, select T, then define the timer's attributes as shown below.'

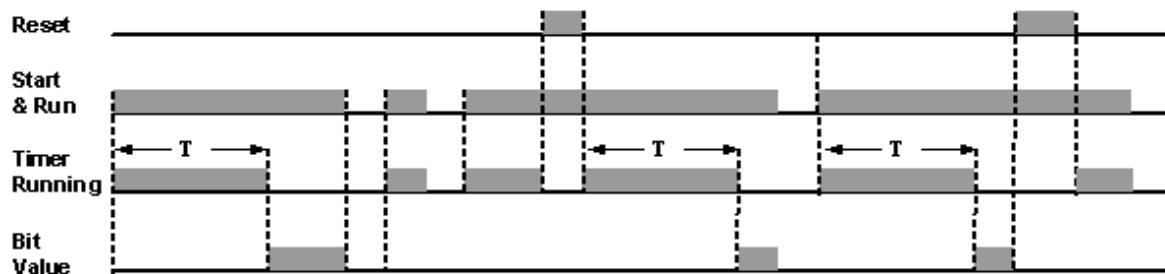


There are 3 types of timers. Each timer type has 3 variables:

- Timer Bit Value: A timer is scanned as a bit data type (scan for OFF, scan for ON). The result of the scan is dependent on the timer type.
- Timer Preset Value. A running timer always decrements (counts down) from the Preset Value. The Preset Values are loaded for all timers at power up. The Preset Value is also loaded into the Current Value when the timer is reset.
- Timer Current Value. The current value of the timer is dependent on the timer type.

All timer types are activated by a rising transition edge, OFF to ON. The condition you use to activate the timer should be scanned only once per PLC program scan

TD- Timer: On Delay



When the timer's Start & Run Condition is OFF, the timer's Bit Value is also OFF.

When the timer's Start & Run Condition rises, the timer's Preset Value is loaded into the timer's Current Value. The timer begins to run. Note that the timer's Bit Value is OFF.

If the timer's Start & Run Condition remains ON during subsequent PLC cycles, the Current Value of the timer continues to decrement.

When the timer has decremented to 0, and the timer's Start & Run Condition is still ON, the timer's Bit Value turns ON. Note that when the timer has finished running, its Current Value is 0.

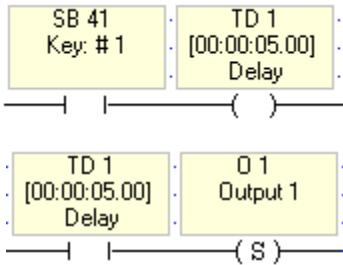
If the timer's Start & Run Condition falls while the timer is decrementing, the timer stops running. The current value of the timer remains.

Timer Reset takes precedence over the timer's Start & Run Condition. When the timer's Reset Condition rises, the timer's Bit Value turns OFF. The timer's Preset Value is loaded into the Current Value, and the timer's Start & Run Condition cannot activate the timer as long as Reset is ON..

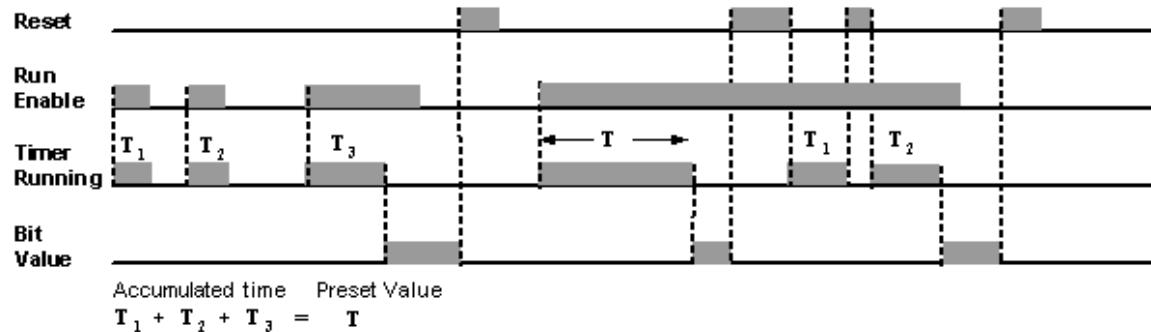
When the timer's Reset Condition falls while the timer's Start & Run Condition is ON, the timer begins to run, exactly the same as when the timer's Start & Run Condition rises.

Below, pressing Key #1 on the Vision keypad activates TD1, which is preset to 5 seconds. If Key #1 is held down for 5 seconds, TD1 decrements to zero. O1 switches on.

If, however, Key #1 is released before TD1 has finished, the timer stops. When Key #1 is pressed again, TD1 again begins to decrement from 5 seconds.



TA Timer: Accumulated



When the timer's Run Enable Condition rises, the timer's Preset Value is loaded into the timer's Current Value. The timer begins to run. Note that the timer's Bit Value is OFF. When the timer's Run Enable Condition remains ON during subsequent PLC cycles, the Current Value of the timer continues to decrement.

When the timer has decremented to 0, and the timer's Start & Run Condition is still ON, the timer's Bit Value turns ON. Note that when the timer has finished running, its Current Value is 0.

If the timer's Run Enable Condition falls while the timer is running, the timer stops running, but the current value of the timer is retained. When the timer is reactivated, it begins decrementing from the retained value.

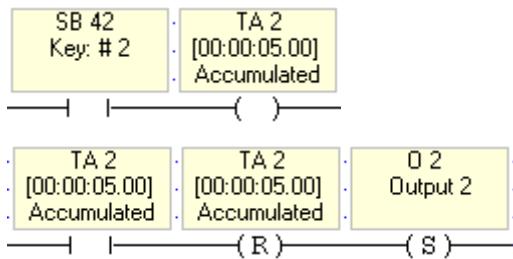
Timer Reset takes precedence over the timer's Run Enable Condition. When the timer's Reset Condition rises, the timer's Bit Value turns OFF. The timer's Preset Value is loaded into the Current Value, and the timer's Run Enable Condition cannot activate the timer as long as Reset is ON.

When the timer's Reset Condition falls while the timer's Start & Run Condition is ON, the timer begins to run, exactly the same as when the timer's Run Enable Condition rises.

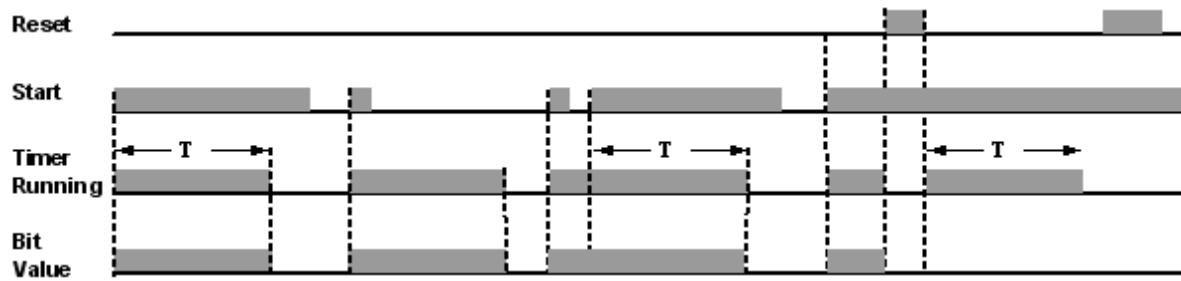
Note • Once a TA Timer has reached its preset value, its Bit Value remains ON until the timer is reset in the program. The timer cannot be activated by Run Enable until it has been reset.

In the net below, pressing Key #2 on the Vision keypad activates TA2, which is preset to 5 seconds. If Key #2 is held down for 5 seconds, TA2 decrements to zero. O2 switches on.

If, however, Key #2 is released after 2.53 seconds--before TA2 has reached the preset value--the timer stops and its current value is retained. When Key #2 is pressed again, TA2 begins to decrement from 2.53 seconds. When TA2 decrements to 0, O2 turns ON.



TE Timer: Extended Pulse



When the timer's Start Condition rises, and the Bit Value is OFF, the timer's Preset Value is loaded into the timer's Current Value. The timer begins to run and the Bit Value turns ON.

If the timer's Start Condition remains ON during subsequent PLC cycles, the Current Value of the timer continues to decrement. However, if the timer's Start Condition rises before the timer has decremented to its Preset Value, the timer reloads the Preset Value into the Current Value, and again begins to decrement. Note that a falling Start condition does not affect the timer.

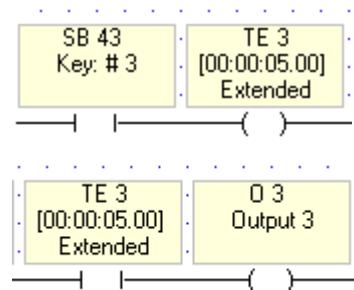
When the timer has decremented to 0 the timer's Bit Value turns OFF. Note that when the timer has finished running, its Current Value is 0.

Timer Reset takes precedence over the timer's Start Condition. When the timer's Reset Condition rises, the timer's Bit Value turns OFF. The timer's Preset Value is loaded into the Current Value, and the timer's Start Condition cannot activate the timer as long as Reset is ON..

When the timer's Reset Condition falls while the timer's Start Condition is ON, the timer begins to run, exactly the same as when the timer's Start Condition rises.

- Note •** Once a TE Timer has reached its preset value, its Bit Value remains OFF until the timer is reset in the program.

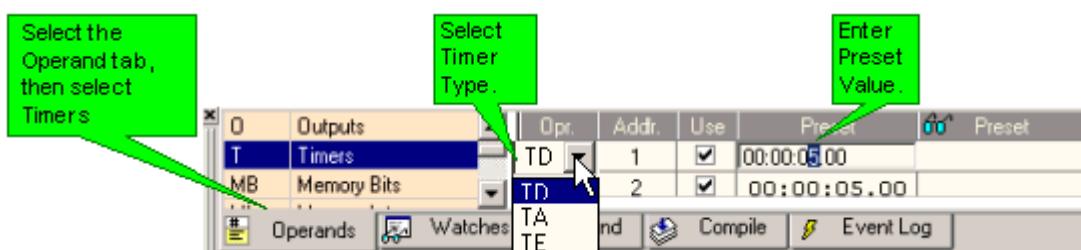
In the nets below, pressing Key #3 on the Vision keypad activates TE3, which is preset to 5 seconds. Once Key #3 is pressed, TE3 decrements to zero. O3 switches on.



- Notes •**
- A Timer value can be displayed in a Display as either a current or elapsed value.
 - The maximum amount of time that you can set a timer for is 99 hours, 59 minutes, and 59.99 seconds.

Viewing and Setting Timers

To display a list of Timers, click on the Operand tab in the Output Window at bottom of the screen, then select Timers. Scroll down to view the list.



Timers can also be preset and edited in the Select Operand and Address dialog box when you insert a timer into your program.

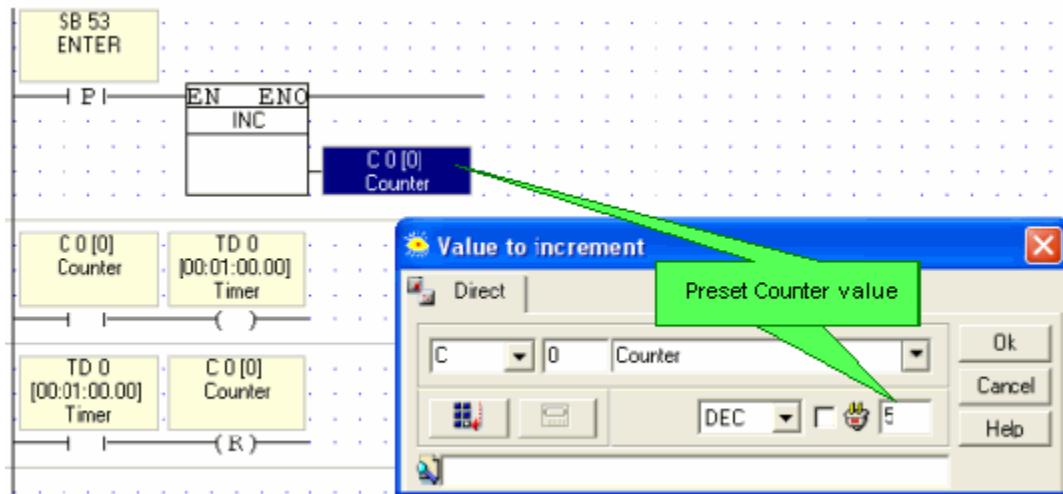
You can also use Information Mode to edit or enter a timer value via the controller keyboard while the controller is running its control program.

Counters (C)

VisiLogic offers 24 built-in counters, represented by the symbol C. To use an Up Counter in your program, place an Increment function in a net and select C. To use a Down Counter in your program, use a Decrement function.

A counter counts rising-edge pulses.

When the accumulated number of pulses equals the counter's preset value, power flows through the function and the counter bit turns ON. Once the preset value is reached, the counter bit stays ON until it is reset via a Reset Coil. This also initializes the counter value.



Note •

Counter values can be displayed on the controller screen via a Counter Variable in the HMI editor. Either the current or the elapsed counter value can be shown in a Display.

Viewing and Setting Counters

A counter's Preset Value can be assigned either in the Select Operand box or in the Output Window. To display a list of Counters, click on the Operand tab in the Output Window at bottom of the screen, then select Counters. Scroll down to view the list.

Op.	Use	Preset	Pres.	Current	On Line	Format	Description
C 2	<input checked="" type="checkbox"/>	144	10	0	0	DEC	Status Sensor 2
C 3	<input checked="" type="checkbox"/>	-6	-6	0	1	DEC	
C 4	<input type="checkbox"/>	0	0	0	1	DEC	
C 5	<input type="checkbox"/>	0	0	0	1	DEC	

Memory Integers (MI)

Memory Integers are 16-bit integer operands that may be signed or unsigned. The range of an MI is -32768 to +32767.

There are 2048 MIs (Address MI 0 - MI 2047).

To display a list of operands, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list

Memory Long Integer (ML)

Memory Long Integers are 32-bit integer operands that may be signed or unsigned, with a range of -2,147,483,648 to +2,147,483,647.

There are 256 MLs (ML 0 - ML 255).

To display a list of operands, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list

Double Word (DW)

Double Words are 32-bit unsigned integer operands, maximum value 4,294,967,296.

There are 64 Double Words, address DW0 to DW63.

Memory Floating Point Integer (MF)

Floating point integers are 32-bit integer operands that may be signed or unsigned, with a range of -3.402823E38 to -1.401298E-45 for negative numbers, and +1.401298E-45 to +3.402823E38 for positive numbers

There are 24 MFs (MF 0 - MF23).

To display a list of operands, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list

System Operands (SI) (SL) (SB) (SDW)

System Operands types include: System Bits (SB), System Integers (SI), System Double Word (SDW), and System Long (SL).

System Operands are used by the controller's operating system to manage certain functions and values. Many System Operands are linked to fixed parameters and are read-only, such as SB 2 Power-up bit, which turns ON for a single cycle whenever the controller powers up.

Other System Operands can be written to by the program, or via INFO Mode. For example, to calculate the current internal temperature of the controller, you can turn on SB 14; the controller will then write the current temperature into SI 14, which is read only.

To display a list of System Operands with their descriptions, click on the Operand tab in the Output Window at bottom of the screen, then select the operand type. Scroll down to view the list.

Note • System Operands have preset descriptions that describe their function. If descriptions have been changed, or if you are opening a project that was written using a different version of VisiLogic, you can display restore descriptions via the Project Menu Project>System Descriptions>Restore all System Descriptions.

- All SBs and SIs which do not have descriptions are reserved for use by the system.

System Bits

General, SBs 0-14

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 0	Always 0	Never	Always	

VisiLogic Software Manual - Ladder

SB 1	Always 1	Always	Never	
SB 2	Power-up bit	Power-up occurs, for 1 scan		
SB 3	1 second pulse			
SB 4	Divide by zero			
SB 5	Outputs short circuit			
SB 6	Keyboard is active			
SB 7	100 mS pulse			
SB 8	Battery low			
SB9	RAM failure :Bit value is not 0 or 1			
SB 10	Float Error	By OS when the result of a float operation is an illegal float value.		By user, or at power-up.
SB 11	User Stack Overflow			
SB 14	Calculate current controller temperature			OS

Touchscreen models only (V280), SBs 16-17, 20-22

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 16	Touchscreen Active	Touchscreen is actually being touched	The screen is not being touched.	OS
SB 17	Enable/Disable Touchscreen indication, Message Board function	User turns ON to enable a message to be handwritten on the touch-screen with a stylus	User turns it off.	User
SB 22	Enable Virtual Keypad	<ul style="list-style-type: none"> • ON by default in Touchscreen-only models (V290) • In Touchscreen + HMI keypad models (V280), user turns ON to enable Virtual keypad. When ON, the normal alphanumeric keypad is suspended. 	<ul style="list-style-type: none"> • Off by default in models with Touchscreen + HMI keypad • May be turned OFF by user. 	User/ OS

HMI Display tasks, SBs 26-34

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 26	<p>Exiting OS Draw Mode (ON for 1 cycle after OS draw)</p> <p>OS Draw Mode means that the controller's Operating System takes control of the LCD screen:</p> <ul style="list-style-type: none"> • During Info Mode • When a Display is entered • When the Virtual Keypad (touch-screen models) is displayed • When 'Symbols' are displayed during 	<p>Turns ON for a single cycle when SB 28 turns OFF. This happens at the following times:</p> <ul style="list-style-type: none"> • When the PLC exits Info Mode. • Rises the cycle after a Display is entered. • When Virtual Keypad mode exits. 	At all other times	OS

	Keypad Entry.	<ul style="list-style-type: none"> After 'Symbols' are displayed during Keypad Entry. 		
SB 27	Enter Display without active Keypad Entry Variables If SB 27 is ON when a Display is shown: <ul style="list-style-type: none"> The user cannot navigate through the variables using the Enter or Right-arrow keys. No Keypad Entry Variable will be marked by the blinking cursor. In this case, a variable may be activated by: <ul style="list-style-type: none"> Touch (V280 only)--assuming it has been assigned the Touch property. By writing the variable ID # into SI 250, either via Info or Online mode. 	By program	By program	
SB 28	LCD controlled by OS (OS drawing) OS Draw Mode means that the controller's Operating System takes control of the LCD screen: <ul style="list-style-type: none"> During Info Mode When a Display is entered When the Virtual Keypad (touch-screen models) is displayed When 'Symbols' are displayed during Keypad Entry. 	<ul style="list-style-type: none"> ON when the PLC is in Info Mode. ON when 'Symbols' are displayed during Keypad Entry. Rises when a Display is entered. In V290, which uses a virtual screen keyboard, SB 28 is always ON. 	<ul style="list-style-type: none"> PLC exits Info Mode After a Display is entered. 	OS
SB 29	Current keypad entry sets SB 30 (HMI keypad entries complete) Turn SB 29 ON after data is keyed into any variable, enabling the user to skip keying in data for all of the variables on-screen. Also refreshes all Display variables on-screen			OS
SB 30	HMI keypad entries completed			OS
SB 31	Refresh current LCD screen display variables			OS
SB 32	HMI keypad entry in progress			OS
SB 33	Display, Call Sub The positive status of SB33 is visible within the specific subroutine only when it runs. Use it to initialize operands in the HMI subroutine.	When a Display containing a Call Sub starts loading, ON for a single scan cycle.		OS
SB 34	Display Exit	Turns ON for a single scan cycle when a display is exited.		OS

OnLine Test SB 35

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 35	OnLine Test Point	During OnLine mode, Single Scan,	One or none instances are	

		when more than 1 instance of OnLine Test Point is activated (receives RLO).	activated	
--	--	---	-----------	--

INFO mode, SB 36

#	Description	Turned ON	Turned Off	Comments
SB 36	INFO mode	By OS, Remote Access, or program	Turns OFF when user exits Info Mode	Delay time to enter Info Mode is 4 seconds, may be modified via SI 50

Invert Touchscreen element pixels, SB 38

#	Description	Turned ON	Turned Off	Comments
SB 38	Invert Touchscreen element pixels (Text, images)	By program	By program	If a Touchscreen text or image element is touched and this bit is on, the pixels in the element reverse color.

Keypad keys, SBs 40-72

Note that the presence of function keys is model-dependant.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 40	Key: # 0			
SB 41	Key: # 1			
SB 42	Key: # 2			
SB 43	Key: # 3			
SB 44	Key: # 4			
SB 45	Key: # 5			
SB 46	Key: # 6			
SB 47	Key: # 7			
SB 48	Key: # 8			
SB 49	Key: # 9			
SB 50	Plus/Minus			
SB 51	Left Arrow			
SB 52	Right Arrow			
SB 53	ENTER			
SB 54	<i>(ON when in Info mode, may be turned ON in order to enter Info, via Remote Access or user program)			

SB 55	Up
SB 56	Down
SB 57	ESC
SB 58	F1
SB 59	F2
SB 60	F3
SB 61	F4
SB 62	F5
SB 63	F6
SB 64	F7
SB 65	F8
SB 66	F9
SB 67	F10
SB 68	F11
SB 69	F12
SB 70	F13
SB 71	F14
SB 72	F15

COM Port/Modem initialization, SBs 80-85

Each port is linked to 2 SBs indicating COM Port/Modem initialization status following COM Init.

Both SBs are initialized to OFF by the OS, at Power-up and at the beginning of COM Init process. When COM Init is complete, one is ON, the other OFF.

#	Description			
SB 80	Modem Initialized: COM Port 1	Example: COM Port 1		
SB 81	COM Port/Modem Initialization Failed: COM Port 1	SB 80	SB 81	
SB 82	Modem Initialized: COM Port 2	0	0	After Power-up, before COM Init
SB 83	COM Port/Modem Initialization Failed: COM Port 2	0	1	Modem Initialization attempt failed, Modem is not initialized
SB 84	Modem Initialized: COM Port 3	1	0	Modem Initialization attempt succeeded, Modem is initialized.

SB 85	COM Port/Modem Initialization Failed: COM Port 3	1	1	Not possible	
-------	--	---	---	--------------	--

Modem connection status, SB 86-88

Each port is linked to an SB indicating modem connection status. These can be used in conjunction with SBs 132-137, which indicate indicating whether incoming or outgoing data is flowing through the port, to troubleshoot problems as shown in the Help topic Modem Troubleshooting.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 86	Modem Connection Status: COM Port 1	PLC receives 'Connect' string from modem	<ul style="list-style-type: none"> • Hang-up • PLC receives string 'No Carrier' • PLC receives break signal 	OS, at Power-up
SB 87	Modem Connection Status: COM Port 2			
SB 88	Modem Connection Status: COM Port 3			

I/O Expansion Modules, SB 90-91

See Help topic Detecting short-circuited end devices

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 90	I/O Expansion error \ I/O Expansion not connected			
SB 91	I/O Exp. Module--Command buffer is full	When an I/O is processing a command.	OFF when an I/O is idle.	

GPRS modem connected, SB 100

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 100	GPRS modem connected	<ul style="list-style-type: none"> • Call Remote device begins. • GPRS incoming call is answered. 	<ul style="list-style-type: none"> • End Session succeeds. • Disconnect from Network succeeds. 	OS

Draw: Out of Range SB 110

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 110	Draw: Out of Range	The OS attempts to draw a line or pixel outside of the legal limits of the controller's LCD.	At the beginning of every cycle	OS

DTR/DSR signals, SBs 120-125

SBs 120-125 register the signals that each port receives from the DTR and DSR pins of a serial communication cable.

The DTR SBs 120, 122, and 124 are also used by the OS to control the DTR signal during RS485 serial communications, and during GPRS communications using the Sony Ericsson GPRS modem.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 120	DTR COM Port 1 (signal output from PLC)	DTR signal present	DTR signal absent	OS, may also be reset by user

SB 121	DSR COM Port 1 (signal input to PLC)	DSR signal present	DSR signal absent	OS
SB 122	DTR COM Port 2 (signal output from PLC)	DTR signal present	DTR signal absent	OS, may also be reset by user
SB 123	DSR COM Port 2 (signal input to PLC)	DSR signal present	DSR signal absent	OS
SB 124	DTR COM Port 3 (signal output from PLC)	DTR signal present	DTR signal absent	OS, may also be reset by user
SB 125	DSR COM Port 3 (signal input to PLC)	DSR signal present	DSR signal absent	OS

COM SBs 132-137

Each port is linked to 2 SBs indicating when incoming or outgoing data is flowing through the port. To troubleshoot problems, use these in conjunction with the Modem Connection Status SBs 86-88, as shown in the topic Modem Troubleshooting.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 132	COM Port 1, Data Transmission	During data send	When data is not being sent	OS
SB 133	COM Port 2, Data Transmission			
SB 134	COM Port 3, Data Transmission			
SB 135	COM Port 1, Data Receive	During data reception	When data is not being received	OS
SB 136	COM Port 2, Data Receive			
SB 137	COM Port 3, Data Receive			

Ethernet-enabled controllers only, SBs 141-158

#	Description	Turns ON when:	Turns OFF when:	Reset by:	Comments
SB 141	Ethernet: Card Exists	Ethernet card is found	No Ethernet card is installed		When the Ethernet: Card Initialization FB runs, the PLC checks whether an Ethernet card is installed.
SB 142	Ethernet: Card Initialized	Ethernet card initialization succeeds	Ethernet card initialization fails		
SB 143	Ethernet: Socket 0 Initialized	Socket 0 initialization succeeds	Socket 0 initialization fails		
SB 144	Ethernet: Socket 1 Initialized	Socket 1 initialization succeeds	Socket 1 initialization fails		
SB 145	Ethernet: Socket 2 Initialized	Socket 2 initialization succeeds	Socket 2 initialization fails		
SB 146	Ethernet: Socket 3 Initialized	Socket 3 initialization succeeds	Socket 3 initialization fails		

SB 147	Ethernet: Socket 0 Connected	Connection established via Socket 0	Socket 0 is free		
SB 148	Ethernet: Socket 1 Connected	Connection established via Socket 1	Socket 1 is free		
SB 149	Ethernet: Socket 2 Connected	Connection established via Socket 2	Socket 2 is free		
SB 150	Ethernet Status: Socket 3 Connected	Connection established via Socket 3	Socket 3 is free		
SB 151	Ethernet Link: Communication established	A link exists	No link exists		
SB 152	Ethernet Link: 10baseT	When a 10baseT link is detected, during data transmit/ receive.	When a 10baseT link is not detected, during data transmit/ receive.		
SB 153	Ethernet Link: 100baseT	When a 100baseT link is detected, during data transmit/ receive.	When a 100baseT link is not detected, during data transmit/ receive.		
SB 154	Ethernet: data collision	More than one device is transmitting data over the Ethernet network	One or no devices are transmitting data over the Ethernet network		
SB 155	Ethernet: Socket 0 Send in Progress	Data is being transmitted via Socket 0	Data is not being transmitted via Socket 0		
SB 156	Ethernet: Socket 1 Send in Progress	Data is being transmitted via Socket 1	Data is not being transmitted via Socket 1		
SB 157	Ethernet: Socket 2 Send in Progress	Data is being transmitted via Socket 2	Data is not being transmitted via Socket 2		
SB 158	Ethernet: Socket 3 Send in Progress	Data is being transmitted via Socket 3	Data is not being transmitted via Socket 3		

SMS message transmission status, SBs 184-185

When the Send process begins, for each and every message, both SB 184 and 185 are OFF. After the message is sent, the relevant bit turns ON, indicating the success or failure of that message. Operands that are linked by the user to SMS FBs may

be found in the topic SMS Operands.

SB	Description	Turns ON when:	Turns OFF when:
184	SMS: Transmission Succeeded	Transmission succeeds	Transmission begins
185	SMS: Transmission Failed	Transmission fails	Transmission begins

CANbus, SBs 200-237

To learn how to use these operands to communicate data, check the topic CANbus Networking.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 200	CANbus Network operand			
SB 201	CANbus Network operand			
SB 202	CANbus Network operand			
SB 203	CANbus Network operand			
SB 204	CANbus Network operand			
SB 205	CANbus Network operand			
SB 206	CANbus Network operand			
SB 207	CANbus Network operand			
SB 208	CANbus Network operand			
SB 209	CANbus Network operand			
SB 210	CANbus Network operand			
SB 211	CANbus Network operand			
SB 212	CANbus Network operand			
SB 213	CANbus Network operand			
SB 214	CANbus Network operand			
SB 215	CANbus Network operand			
SB 236	CANbus Network communication error		Error is fixed.	
SB 237	CANbus Network disable			

Keypad entry, SBs 250-251

To learn how to use these operands to communicate data, check the topic Limit Keypad Entry.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 250	Keypad entry within limits	For a single cycle after keypad entry when entry within limits.		
SB 251	Keypad entry exceeds limits	For a single cycle after keypad entry when entry exceeds limits.		

SMS ASCII, SB 279

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 279	Send SMS messages in ASCII format	User Program Should be turned ON at power-up, before Com Init.		User Program

Reset PLC, SB 300

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 300	Reset PLC	By program or Remote Access	Reset is run	OS

Buzzer SB 310, 311

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 310	Buzzer Turn this ON to sound a buzzer	By user	By user	User
SB 311	Buzzer - Screen Touch Turn this ON to cause a keypad touch (both HMI keypad and Virtual keypad) to sound a buzzer	By user, ON by default in V290/280	By user	User

Data Tables, SB 500

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 500	DT Write: Address exceeds DT range			

System Integers

General, SIs 0-14

#	Description	Value	Comments

SI 0	Scan Time, Resolution: Units of 10 mSec	Updated by the controller at the end of every scan.	A scan is a complete execution of the controller's entire program: reading inputs, executing the Ladder program, updating the outputs, running the HMI program, and processing communications. Scan time depends on the size and complexity of the application. Check the topic Program Sequencing: Modules, Subroutines, Labels & Jumps.
SI 6	Current key pressed		
SI 7	LCD Contrast Control	0=Minimal Contrast 50=Medium Contrast 100=Maximal Contrast	
SI 8	Unit ID (Network)	The ID # 1 is assigned by default.	To learn how to use this operand, check the topic Assigning a Unit ID number
SI 9	LCD Backlight intensity	0 - Off 1 - On (low intensity) (V230 only) 2 - On (max. intensity) - Default	
SI 14	Current controller temperature		Includes decimal point. For example, if the value is 245, the actual value is 24.5.

Real Time Clock, SIs 30-34

#	Description	Value	Comments
SI 30	Current second		According to RTC
SI 31	Current time		
SI 32	Current date		
SI 33	Current year		
SI 34	Current day		

Touch Coordinates

#	Description	Value	Comments
SI 40	Touchscreen is being touched-X coordinates	If the screen is touched, SI 40 shows the current location on the X axis.	When the screen is not touched, SI 40 = -1
SI 41	Touchscreen is being touched-Y coordinates	If the screen is touched, SI 41 shows the current location on the Y axis.	When the screen is not touched, SI 41 = -1

INFO delay time, SI 50

#	Description	Value	Comments

SI 50	INFO delay time	Default by O/S (every power up) = 4 seconds	<ul style="list-style-type: none"> Units: seconds. Legal values: 0, 3 to 20. If you force or store '0' into equal Zero – INFO is disabled. For V290 – Touching the <i> key on the touch screen starts Info Mode – Touching a legal Ladder application variable clears the INFO time.
-------	-----------------	---	--

COM Port: Port/Modem Status, Error codes, SIs 80-85

Each COM Port is linked to 2 SIs; their values and messages are indicated below.

SI 80	Modem Status: COM 1	Error (SI 81,83,85,)		Status (SI 80, 82, 84)	
		Value	Message	Value	Message
SI 81	Error Code: COM 1	0	No error	0	Modem Idle
SI 82	Modem Status: COM 2	1	TimeOut exceeded: no reply	1	Initialization in Progress
SI 83	Error Code: COM 2	2	Reply Error	2	Initialization OK
SI 84	Modem Status: COM 3	3	Wrong PIN number	3	Initialization Failed
SI 85	Error Code: COM 3	4	Registration failed	4	Modem Connected
		5	PUK number needed	5	Hang-up in progress
		10	COM Busy	6	Dial in progress
		11	Reply Busy		
		12	Reply No Dial		
		15	Attempted Initialization during active break signal. Note that a port cannot be initialized while the break signal is active		

Max. Delay between characters, MODBUS + Modem, SI 100

SI	Description	Value
100	Maximum Time Delay between characters (units 2.5ms) MODBUS + Modem	<p>When MODBUS (Serial) is configured to a port linked to a modem, the MODBUS function checks SI 100. If SI 100 = 1, a time interval of up to 2.5 msecs is permitted between characters, if SI 100 contains 2, the permitted interval is 5 msecs (n x 2.5 =interval).</p> <p>Note that:</p> <ul style="list-style-type: none"> - The power-up value is 1, - the application must update SI 100 before the MODBUS configuration is activated.

Ethernet-enabled controllers only, SIs 141-148

Parameter	Description	Value	Comments
SI 141	Ethernet Socket 0: Protocol Type	<ul style="list-style-type: none"> • 0=PC application (default) • 1=MODBUS 	(Read-only) Sockets are set to Protocol Type 0 by default. Activating MODBUS Configuration changes the Protocol Type to 1.
SI 142	Ethernet Socket 1: Protocol Type		
SI 143	Ethernet Socket 2: Protocol Type		
SI 144	Ethernet Socket 3: Protocol Type		

Parameter	Function	SI value	Message
SI 145	Socket 0: Status	0	Initialized to UDP, status: Closed
SI 146	Socket 1: Status	2	Initialized to TCP, status: Listen
SI 147	Socket 2: Status	14	Initialized to UDP, status: Ready
SI 148	Socket 3: Status	15	Initialized to UDP, status: Engaged in Transmit/Receive

GSM Cellular Modem, GSM Signal Quality, SI 185

SI	Description	Value
185	GSM Signal Quality	<ul style="list-style-type: none"> • The value is written during COM Init of the GSM modem. The value is updated whenever the user uses the GSM Signal Quality FB. • A value of -1(FFFF) signifies a modem error. This may be due to a weak signal; try repositioning the antenna. If this has no effect, check the modem.

CANBUS, SIs 200-201, 236-237, 240-243

SI 200	CANbus Network operand											
SI 201	CANbus Network operand											
SI 236	CANbus Network communication error code	<table> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Message</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>No Acknowledgement</td> </tr> <tr> <td style="text-align: center;">2</td> <td>CANbus OFF</td> </tr> <tr> <td style="text-align: center;">4</td> <td>CANbus Warning error</td> </tr> <tr> <td style="text-align: center;">10</td> <td>ISC receiving TimeOut</td> </tr> </tbody> </table>	Value	Message	1	No Acknowledgement	2	CANbus OFF	4	CANbus Warning error	10	ISC receiving TimeOut
Value	Message											
1	No Acknowledgement											
2	CANbus OFF											
4	CANbus Warning error											
10	ISC receiving TimeOut											
SI 237	CANbus Network: failed unit ID											
SI 240	SIs 240-243 comprise a bitmap indicating which unit is in error. If, for example, the network includes unit ID numbers 8, 9 and 13, and PLC #9 cannot be accessed, then the ninth bit in SI240 will turn ON. When the error is fixed, the bit falls to OFF											
SI 241												
SI 242												
SI 243												

HMI Displays, SIs 249-252

SI 249	Last Active Keypad Entry Variable	Contains the ID number of the last active variable.
SI 250	Currently active keypad entry, read/write	<p>Currently active keypad entry, read/write.</p> <p>When either SB 250 'Keypad Entry Within Limits' or SB251 'Keypad Entry Exceeds Limits' turn ON, the index number of the variable is stored here. As you navigate between variables, as for example with the right-left arrow keys, SI 250 will show only the numbers of variables that have not been completed.</p> <p>Note • A value of -1 indicates that, in this particular display, the user has pressed Enter for all the Keypad Entry variables in the Display.</p>
SI 251	Previous HMI Display Number	
SI 252	Current HMI Display Number	To see a list of Displays in a project together with their Display numbers, select HMI Information from the View menu.

Info Mode, SI 253

SI 253	Password: Info Mode	<p>Note that at every power-up, the default password to Info Mode, 1111, is restored. To maintain a different password after power-up, use SB 2-Power-up as a condition to store the desired password value into SI 253.</p> <p>The password may also be modified by accessing the controller via VisiLogic, then running On-line Test mode and changing the value. This value will be erased at power-up.</p>
--------	---------------------	--

Error, General, SI 500

SI 500	General Error	Value	Message
		3	7FFF or 8000 (integer result) FFFF or 0000(unsigned integer result)
		4	+INF or -INF (float result)
		5	0.0 (float result)
		7	+INF or -INF or NaN (float result)
		9	NAN (float result)
		10	0 (integer result)
		11	Floating point stack underflow
		12	Floating point stack overflow

User Stack Depth, SI 503

SI 503	User Stack Depth
--------	------------------

System Long Integers

SL 4	Divide Remainder (signed divide function)
------	---

System Double Words

#	Description	Value	Comments
SDW 0	10mS counter		
SDW 2	SDW 2 Cycle Counter	Increments by 1 every program cycle	
SDW 3	2.5 mS counter		
SDW 4	Divide Remainder		Unsigned divide function
SDW 5	Expansion module short circuit bitmap		
SDW 6	Snap-in module short circuit bitmap		
SDW 10	Keypad entry variable value		When a keypad entry variable value is entered, this SDW 10 holds the value.
SDW 13	Phone number of last received SMS		last 9 digits
SDW 14	Socket 0: Number of sent transmissions	Updated after each data transmission via Socket 0	
SDW 15	Socket 1: Number of sent	Updated after each data transmission via	

	transmissions	Socket 1	
SDW 16	Socket 2: Number of sent transmissions	Updated after each data transmission via Socket 2	
SDW 17	Socket 3 : Number of sent transmissions	Updated after each data transmission via Socket 3	
SDW 18	Socket 0: Number of received transmissions	Updated after each data packet received via Socket 0	
SDW 19	Socket 1: Number of received transmissions	Updated after each data packet received via Socket 1	
SDW 20	Socket 2: Number of received transmissions	Updated after each data packet received via Socket 2	
SDW 21	Socket30: Number of received transmissions	Updated after each data packet received via Socket 3	
SDW 30	Variable display bitmap, 0=Normal, 1=Inverse (or negative)	The value is checked when a display is entered. It is initialized to 0: - At Power-up. - When the program exits the Display.	When a bit is ON, the corresponding variable is displayed in inverted (negative) color; black pixels are changed to white and white to black.
SDW 31	Hide Var	The value is checked when a display is entered. It is initialized to 0 at: - Power-up. - When the program exits the Display.	When a bit is ON, the corresponding variable is hidden
SDW 60	Info Error Status	Error Indication	

On-line Test (Remote Access) Mode, SI 86,88

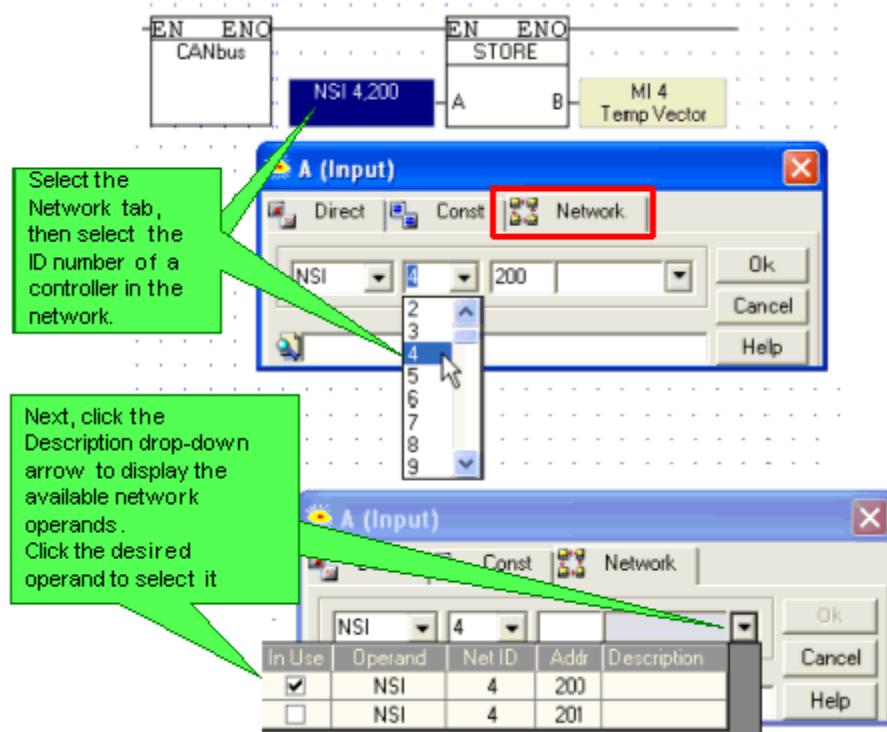
These SIs enable the controller to send SMS messages when the controller is in On-line Test (Remote Access) mode. The SIs do not need to be used in the application because the process is transparent to the user.

SI	Description
86	Modem Connection Status: COM 1
87	Modem Connection Status: COM 2
88	Modem Connection Status: COM 3

Network Operands-Communicating Data Via CANbus

When a controller is integrated into a CANbus network, the data contained in certain system operands is continuously broadcast to the network, together with the controller's unique ID number. The data is contained in 16 System Bits (SB 200 to SB 215(16 Inputs (I 0 to I 15), and 2 System Integers (SI 200 and SI 201).

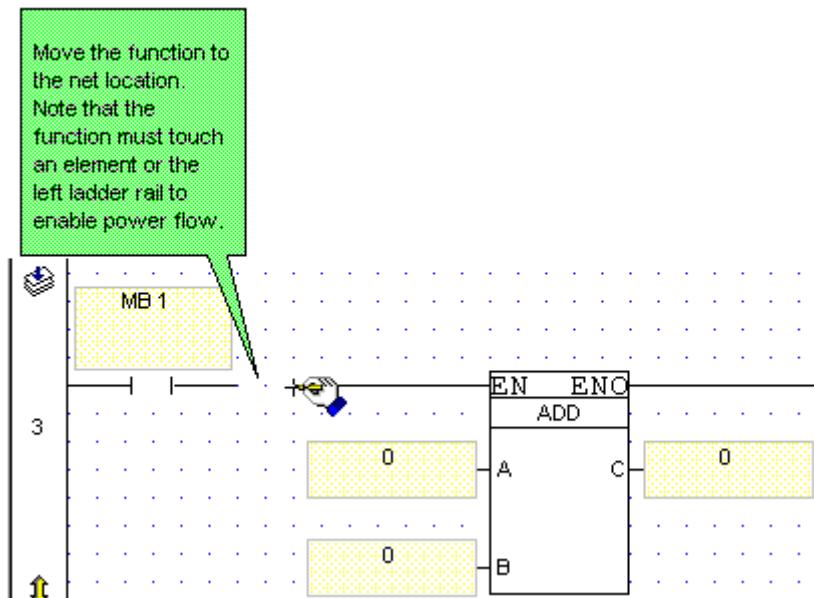
In order to enable a networked controller to read the values from another networked controller, place the desired function in the net. In the Select Operand Address box, click on the Network tab, then select the ID of the target controller and the desired operand.



Functions

Placing a Function in a Net

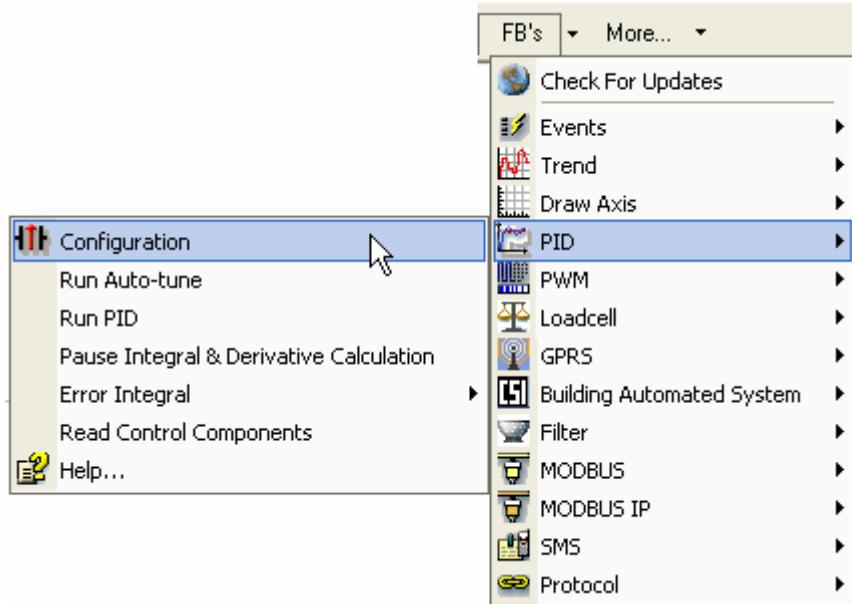
1. Select any type of Ladder function by:
 - Selecting it from the Ladder toolbar, -or-
 - Selecting it from the Ladder menu, -or-
 - Right-clicking on the Ladder to display the Ladder menu and then selecting the function.
2. Move the function to the desired net location, then click.



3. Link operands using the Select Operand and Address.

FBS Library

Unitronics offers an FB library for advanced functions, such as SMS messaging and MODBUS communications. FBs that are currently installed in VisiLogic are listed under the FB's menu.



Note •

You must use a condition (RLO) to activate any FB that requires Configuration in your application, such as MODBUS or SMS.

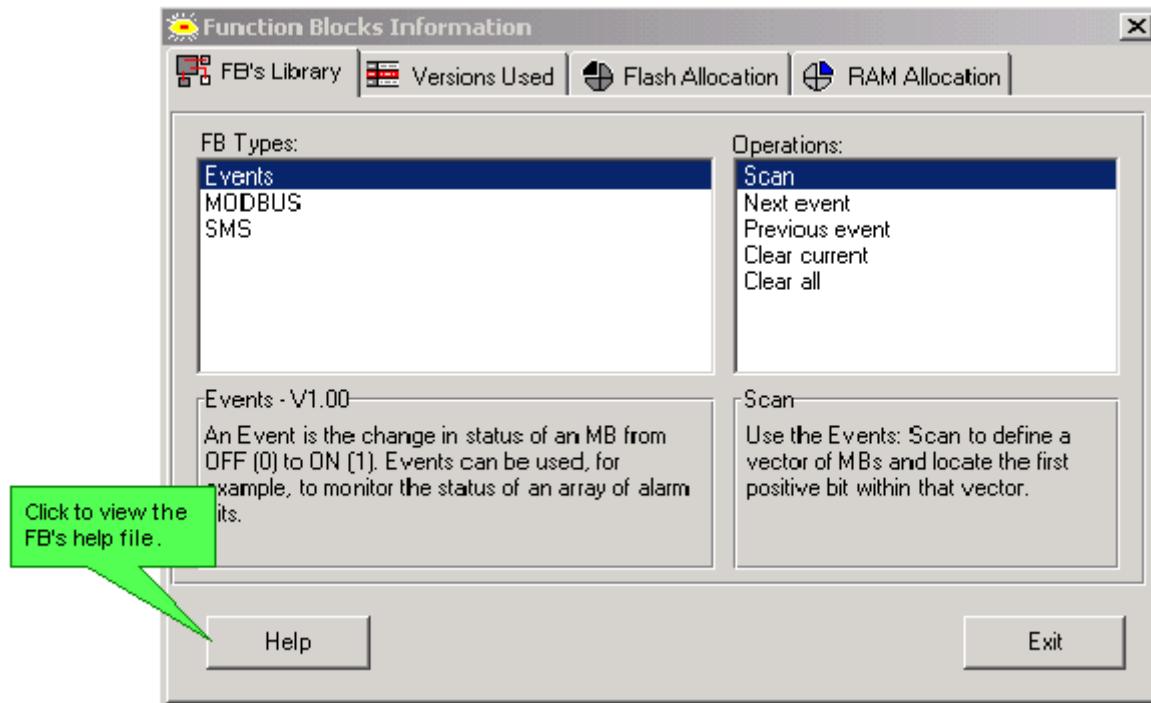
To install an updated FB library, select Update from the Web from the FBS menu or Help menu, then follow the on-screen instructions. Note that at the end of the download, you must close and then restart VisiLogic. The new FBs will appear on the FBS menu.

Note • To enable Live Update, you can select to use a proxy server in Project Properties.

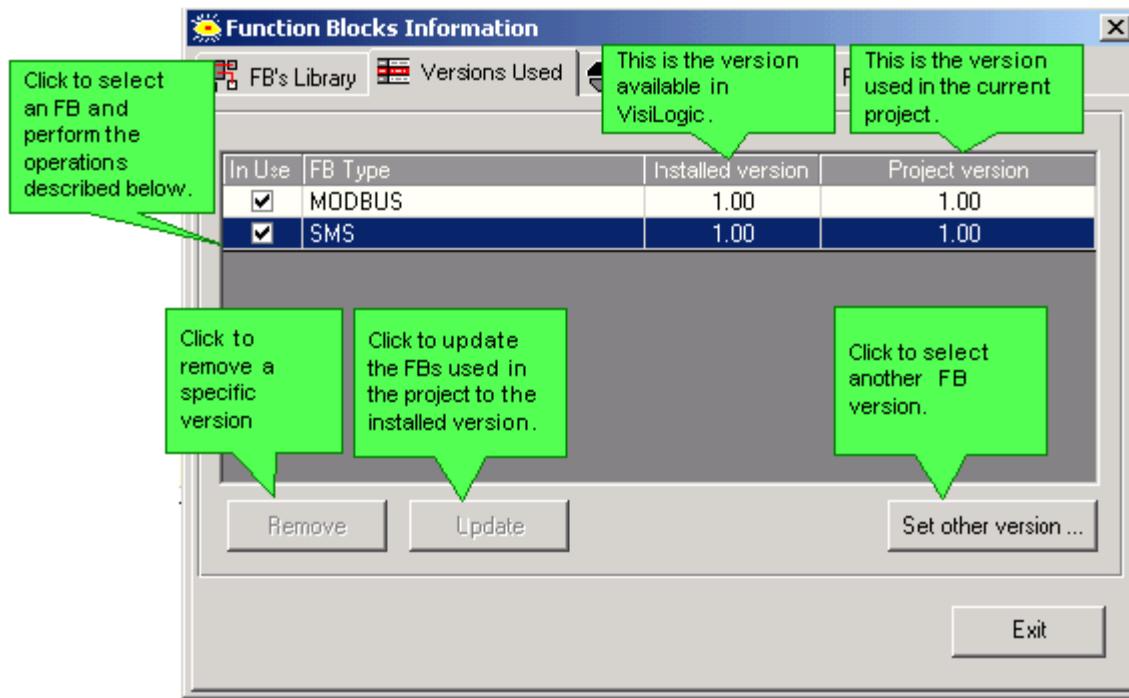
Use Function Block Information, located on the View menu, to check:

- Which FBs are installed in your library.
- Which FB versions are installed, which versions are used in the open project, and to manage FB versions.
- FB memory usage.

FB Library



Versions Used



FBs List

Trends: Real-Time HMI Graph

Draw Axis

PID FB

Events

MB as PWM

Loadcell

Filter

MODBUS, serial

MODBUS, IP

SMS Messaging

GPRS

Communication Protocol

Compare Functions

Compare Functions

A compare function compares two values according to the type of function you select.

If the comparison is true (logic 1): power flows through the block.

If the comparison is false (logic 0): power does not flow through the block.

There are 6 types of Compare Functions:

- Greater Than
- Greater Than or Equal To
- Equal To
- Not Equal To
- Less Than or Equal To

Note • | The Vector menu includes a Compare Vector function.

These values may be compared:

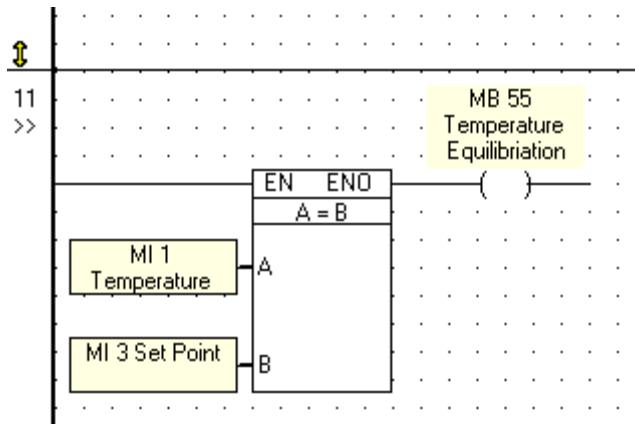
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Equal

The Equal function block compares the value of input A to input B.

If input A is equal to input B : power will flow through the function block.

If input A is not equal to input B: power will not flow through the function block.



According to the above example:

- If MI 1 is equal to MI 3; then MB 55 will go to logic "1" (ON).
- If MI 1 is not equal to MI 3; then MB 55 will go to logic "0" (OFF).

These values may be compared:

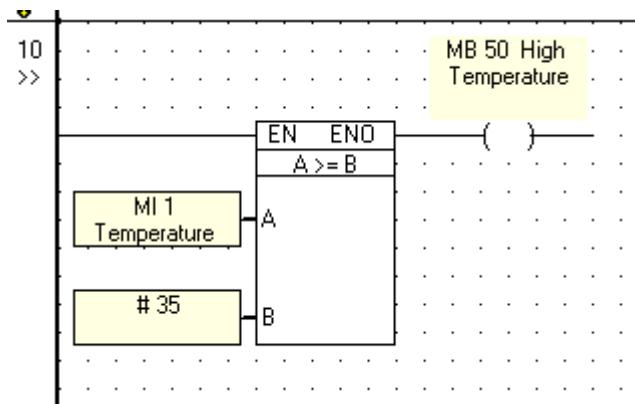
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Greater or Equal to \geq

The Greater Than or Equal function block compares the value of input A to input B.

If input A is greater than or equal to input B: power will flow through the function block.

If input A is not greater than or not equal to input B: power will not flow through the function block.



According to the above example:

- If MI 1 value is greater or equal to constant integer 35; then MB 50 will go to logic "1" (ON).
- If MI 1 value is not greater or equal to constant integer 35; then MB 50 will go to logic "0" (OFF).

These values may be compared:

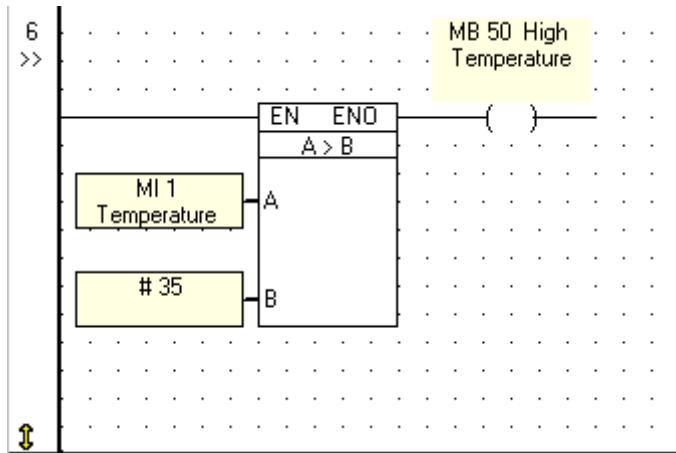
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Greater Than >

The Greater Than function block compares the value of input A to input B.

If input A is greater than input B: power will flow through the function block.

If input A is not greater than input B: power will not flow through the function block.



According to the above example:

- If MI 1 value is greater than 35; then MB 50 will go to logic "1" (ON).
- If MI 1 not greater than 35; MB 50 will go to logic "0".

Note • | Greater and Less Than function blocks do not give an output when input A equals input B.

These values may be compared:

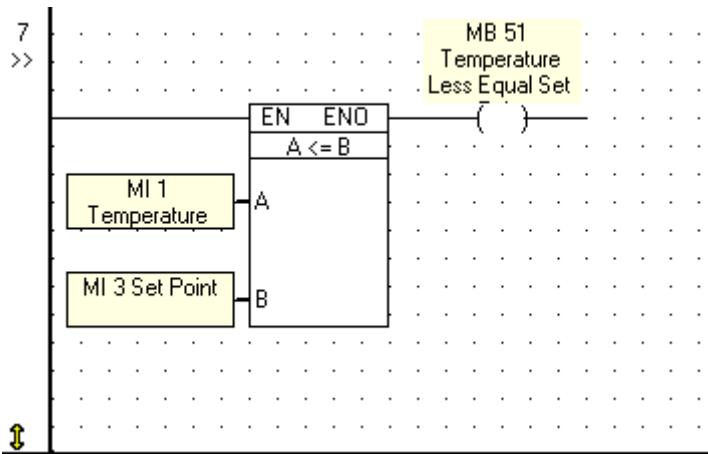
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Less or Equal to <=

The Less Than or Equal To function compares input A to input B. The function is located on the Compare menu.

If input A is less than or equal to input B: power will flow through the function.

If input A is not less than or equal to input B: power will not flow through the function.



According to the above example:

- If MI1's value is less than or equal to MI3's value, then MB 51 will go to logic "1" (ON).
- If MI1's value is less than or equal to MI3's value, then MB 51 will go to logic "0" (OFF).

These values may be compared:

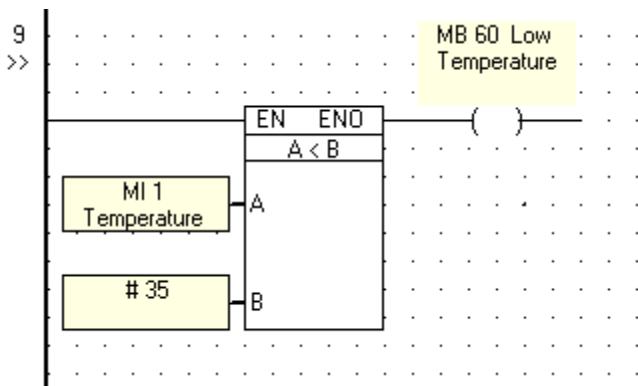
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Less Than <

The Less Than function compares input A to input B. The function is located on the Compare menu.

If input A is less than input B: power will flow through the function.

If input A is not less than input B: power will not flow through the function.



According to the above example:

- If MI 1 value is less than constant integer 35; then MB 60 will go to logic "1" (ON).
- If MI 1 values is not less than constant integer 35; MB 60 will go to logic "0" (OFF).

These values may be compared:

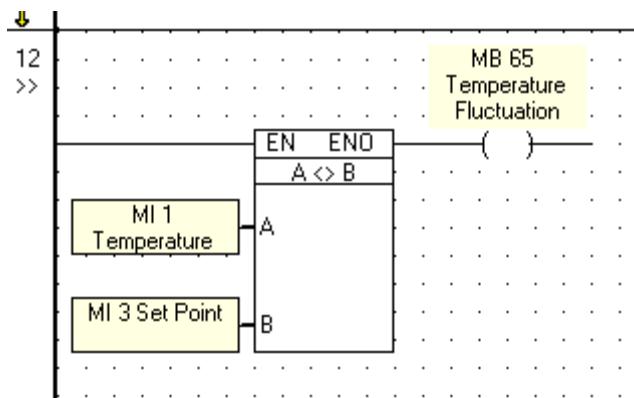
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Not Equal

The Not Equal function evaluates input A to see if its integer value is not equal to input B. The function is located on the Compare menu.

If input A is not equal to input B: power will flow through the function.

If input A is equal to input B: power will not flow through the function.



According to the above example:

- If MI 1 is not equal to MI 3; then MB 65 will go to logic "1" (ON).
- If MI 1 is equal to MI 3; then MB 65 will go to logic "0" (OFF).

These values may be compared:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

Logic Functions

Logic Functions

Function blocks are provided for:

- Bit Test
- Set/Reset Bit
- AND
- OR
- XOR
- Shift
- Rotate
- Convert
- Test Bit
- Store Bit Status
- Load Bit Status
- RS-SR Flip-Flop

The internal operation of a function block is transparent to the user. You select input operands; the result is automatically output by the function block.

The input values in a logic function may be:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

The functions are located under the Logic menu on the Ladder toolbar.

AND

The AND logic function evaluates the state of two integers.

- If a bit is true (logic 1) in both input A and B, then the output C will be true (logic 1).
- If input A and B is false (logic 0), then the output C will be false (logic 0).
- If either input A or B is false (logic 0) - the output C will be false (logic 0).

AND Truth Table		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

The input values in an AND function may be:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

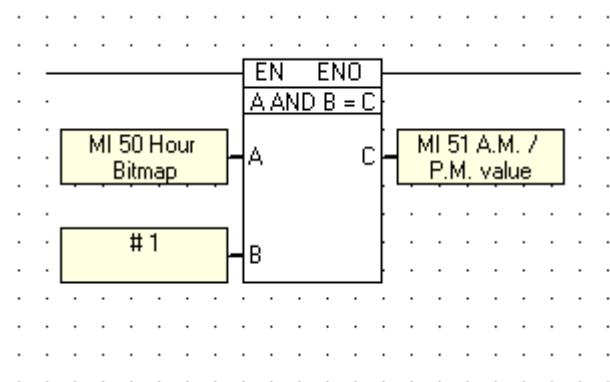
AND can be used to mask out certain bits of an input integer not relevant to a given function.

Example:

If a clock function block uses the first bit of a 16-bit word to decide if a given time is A.M. or P.M., you can mask out the other 15 bits. This will tell you if the current time is A.M. or P.M.

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1
AND																
Mask	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Result	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

All of the non-relevant bits will be turned off (logic 0) except the A.M. / P.M. bit.



The function is located under the Logic menu on the Ladder toolbar.

OR

The OR logic function block can evaluate the state of two integers to see if either input A or B is true. If input A OR B is true - the output C will be true (logic 1). If both input A and B are true (logic 1) - the output C will also be true (logic 1).

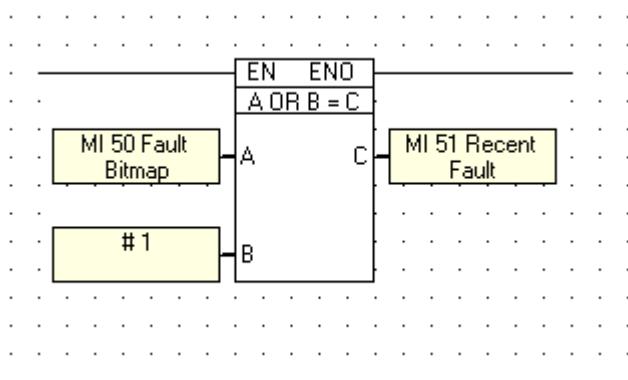
OR Truth Table		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

The input values in an OR function may be:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1
OR																
Compare	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Result	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1



The function is located under the Logic menu on the Ladder toolbar.

XOR

The XOR logic function block can evaluate the state of two integers to see if input A and B are equal. If either input A OR B is true - the output C will be true (logic 1). If both input A and B are true (logic 1) - the output C will be false (logic 0). If both input A and B are false (logic 0) - the output C will be false (logic 0).

XOR Truth Table		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

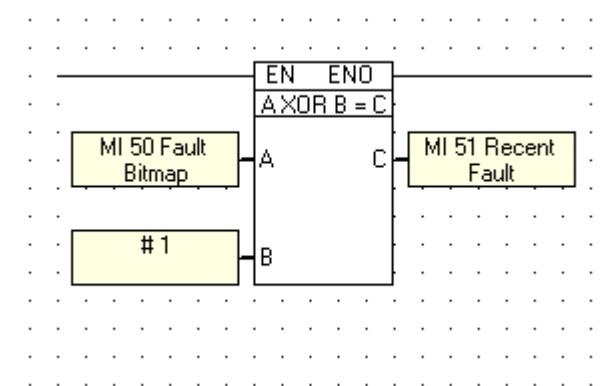
The input values in a XOR function may be:

- Memory Integer (MI)
 - Memory Long Integer (ML)
 - Double Word (DW)
 - System Operands:(SI) (SL)(SDW)
 - Network System Integer (NSI)
 - Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

Use XOR to recognize changes in an integer to check for integer bit corruption. If 2 integers are equal: the result will return logic 0. If there has been bit corruption: the corrupted bit will return logic 1.

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1
XOR																
Compare	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Result	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	0

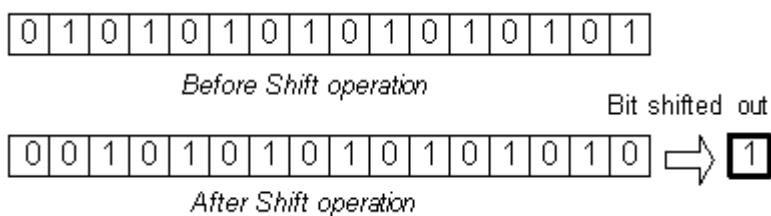


The function is located under the Logic menu on the Ladder toolbar.

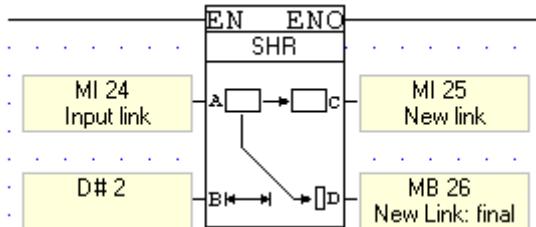
Shift

The Shift function moves the bits in an integer to the left or to the right. Note that any bit shifted out cannot be recovered.

Shift Right



- Operand A: contains the value to be shifted.
- Operand B: contains the number of bits to be shifted (one or more).
- Operand C: contains the resulting value.
- Operand D: shows the status of the final bit in the integer after the operation, regardless of the number of bits shifted out during the operation.



Note that regardless of the number of bits shifted out, Operand D shows the sta

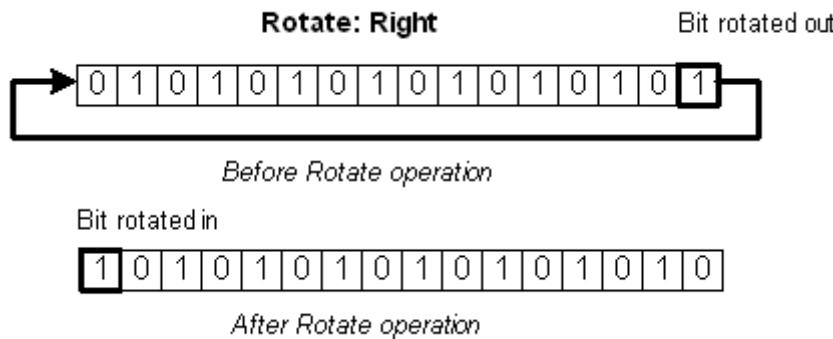
The Shift function may be performed on values contained in the following operands:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)

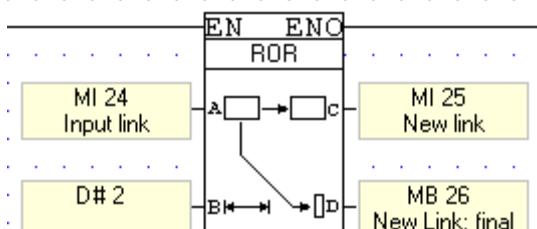
The functions are located under the Logic menu on the Ladder toolbar.

Rotate

The Rotate function moves the bits in an integer to the left or to the right.



- Operand A: contains the value to be rotated.
- Operand B: contains the number of bits to be rotated.
- Operand C: contains the resulting value.
- Operand D: shows the status of the final bit in the integer after the operation.



The Rotate function may be performed on values contained in the following operands:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)

The functions are located under the Logic menu on the Ladder toolbar.

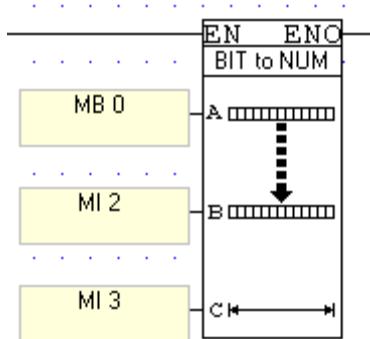
Vector: Bit to Numeric, Numeric to Bit

Use these functions to convert an array of bit values to a numeric value, or a numeric value to an array of bits.

The functions are located on the Vector menu.

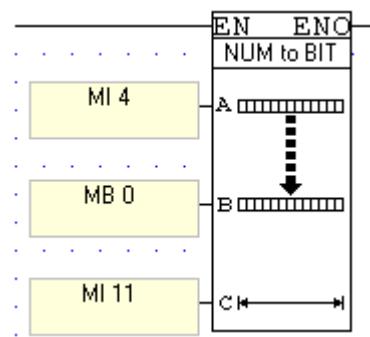
Bit to Numeric

- Operand A: contains the Start Address for the array of bits to be converted.
- Operand B: is the start of the vector that will contain the converted value. Take care in addressing operands, since the converted value may not fit into a single register; the function will overwrite as many consecutive registers as it requires to convert the value.
- Operand C: contains the length of the bit array that will be converted.



Numeric to Bit

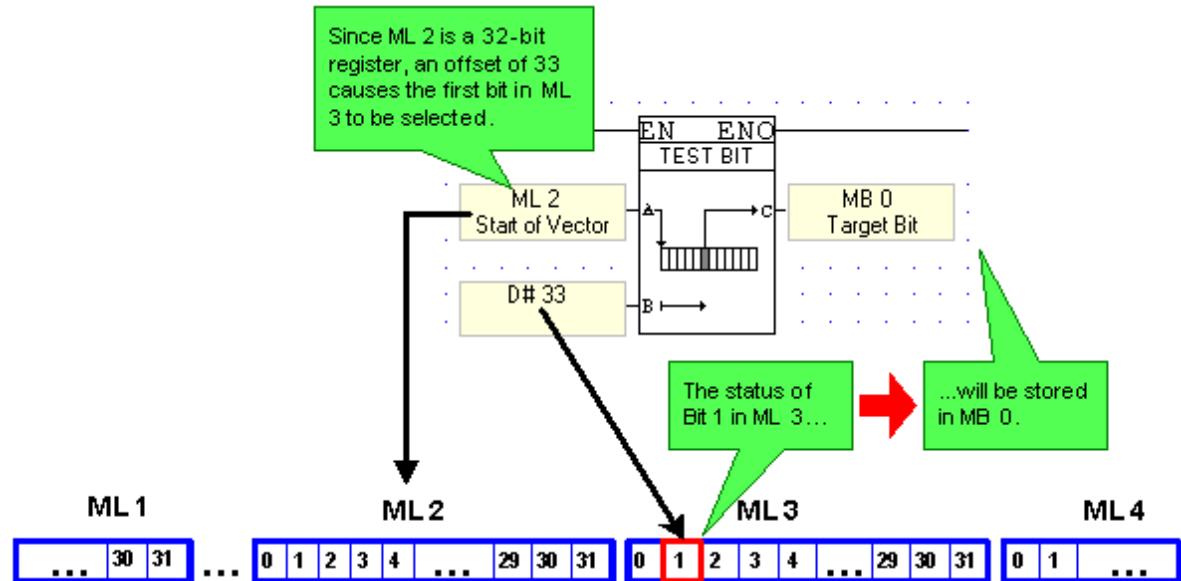
- Operand A: contains the Address of the value to be converted.
- Operand B: contains the Start Address of the bit array that will contain the converted value.
- Operand C: contains the Length of the bit array that will contain the converted value.



Test Bit

Test Bit enables you to select a bit within a vector of registers, and store its status in an MB.

- Operand A, Start of Vector, determines the start of the vector of registers.
- Operand B, Offset in Vector, selects the bit within that vector.
- Operand C, Target Bit, determines where the value of the selected bit will be stored.



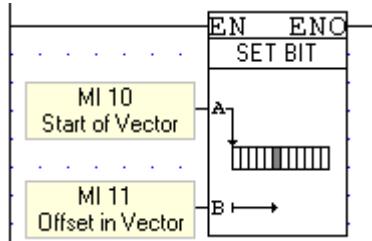
The function is located under the Logic menu on the Ladder toolbar.

Set/Reset Bit

Set Bit enables you to select a bit within a vector of registers, and set it.

Reset Bit enables you to select a bit within a vector of registers, and reset it.

- Operand A, Start of Vector, determines the start of the vector of registers.
- Operand B, Offset in Vector, selects the bit within that vector.

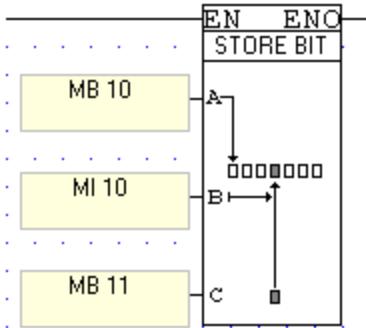


The functions are located under the Logic menu on the Ladder toolbar.

Store Bit Status

Use this to select an MB and store its status in an MB within a defined vector.

- Operand A, Start of Vector, determines where the vector begins.
- Operand B, Offset in Vector, selects the target bit within that vector.
- Operand C, Bit Value, determines the source bit. The status of this bit will be stored into the target bit within the defined vector.

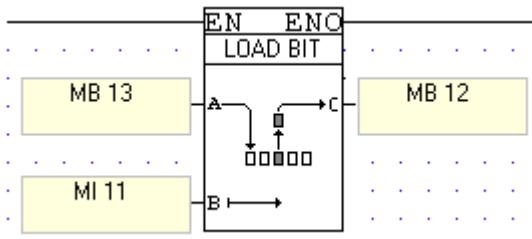


The function is located under the Logic menu on the Ladder toolbar.

Load Bit Status

Use this to select an MB within a defined vector and load its status in an MB outside of that vector.

- Operand A, Start of Vector, determines where the vector begins.
- Operand B, Offset in Vector, selects the source bit within that vector.
- Operand C, Bit Value, determines the target bit--where the value of the source bit will be stored.



The function is located under the Logic menu on the Ladder toolbar.

RS-SR Flip-Flop

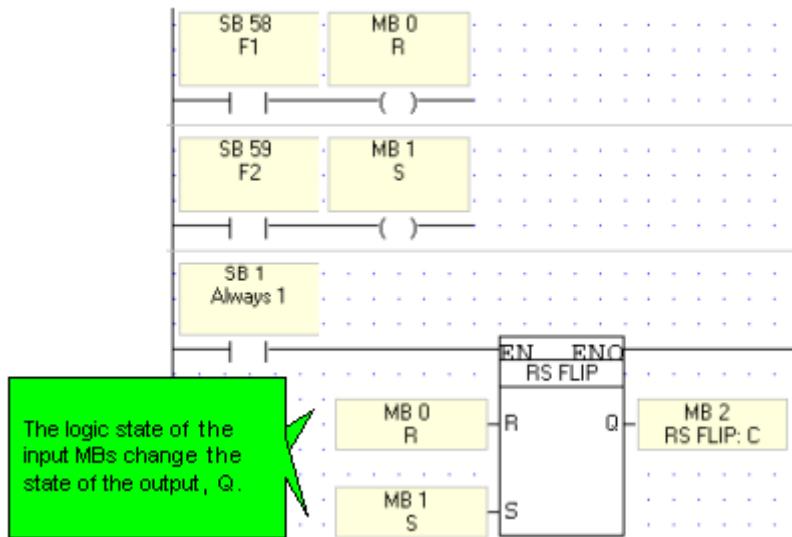
The RS and SR Flip-Flop functions are located on the Logic menu. These functions compare the logic state of two inputs, and use the result to determine an output result in accordance with the tables shown below.

RS Flip-Flop

R (A)	S (B)	Q
0	0	No change
0	1	1
1	0	0

SR Flip-Flop

S (A)	R (B)	Q
0	0	No change
0	1	0
1	1	1

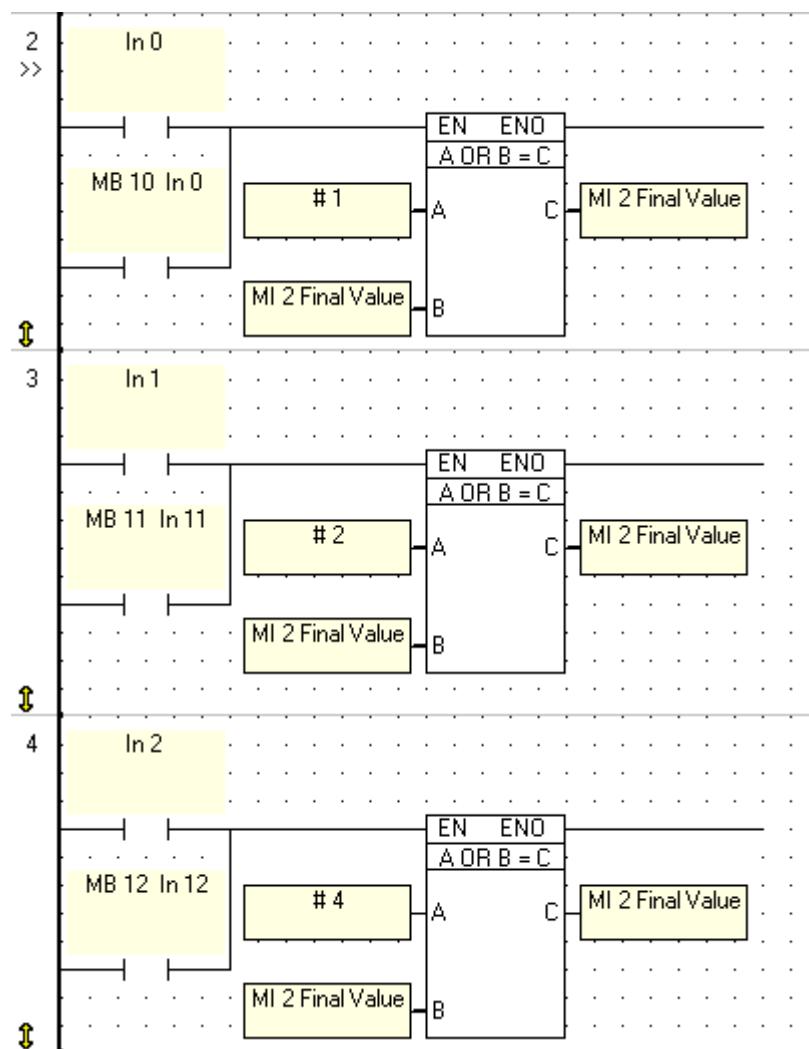


Binary Numbers

Memory Integers and System Integers are 16-bit binary numbers. You enter decimal numbers into Memory Integers and System Integers. The program converts these decimal numbers into binary numbers and performs the specified functions.

You may want to use a logic function to mask out bits or check for bit corruption. You can do this by using a decimal number that converts to the appropriate binary number. The following charts will help you understand why the decimal numbers {0,1,2,4,8,16,32,64,128, etc} were chosen for use with logical OR to evaluate keypad input numbers in the following example.

- ☞ This program shows how to use the logical OR operation. The binary value of 12 inputs is evaluated.
- The value of each input is compared with a number value that is entered from the keypad.
- ☞ The Memory Bits which are parallel to the inputs are used for debugging.



2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	D
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	0	0	0	0	1	1	3
0	0	0	0	0	0	0	0	0	0	1	0	0	4
0	0	0	0	0	0	0	0	0	0	1	0	1	5
0	0	0	0	0	0	0	0	0	0	1	1	0	6
0	0	0	0	0	0	0	0	0	0	1	1	1	7
0	0	0	0	0	0	0	0	0	1	0	0	0	8
0	0	0	0	0	0	0	0	1	0	0	0	1	9
0	0	0	0	0	0	0	0	0	1	0	1	0	10
0	0	0	0	0	0	0	0	0	1	0	1	1	11
0	0	0	0	0	0	0	0	0	1	1	0	0	12
0	0	0	0	0	0	0	0	0	1	1	0	1	13
0	0	0	0	0	0	0	0	0	1	1	1	0	14
0	0	0	0	0	0	0	0	0	1	1	1	1	15
0	0	0	0	0	0	0	0	1	0	0	0	0	16
0	0	0	0	0	0	0	1	0	0	0	0	1	17
0	0	0	0	0	0	0	0	1	0	0	1	0	18
0	0	0	0	0	0	0	1	0	0	1	1	1	19
0	0	0	0	0	0	0	0	1	0	1	0	0	20
0	0	0	0	0	0	0	0	1	0	1	0	1	21
0	0	0	0	0	0	0	0	1	0	1	1	0	22
0	0	0	0	0	0	0	0	1	0	1	1	1	23
0	0	0	0	0	0	0	0	1	1	0	0	0	24
0	0	0	0	0	0	0	0	1	1	0	0	1	25
0	0	0	0	0	0	0	0	1	1	0	1	0	26
0	0	0	0	0	0	0	0	1	1	0	1	1	27
0	0	0	0	0	0	0	0	1	1	1	0	0	28
0	0	0	0	0	0	0	0	1	1	1	1	0	29
0	0	0	0	0	0	0	0	1	1	1	1	0	30
0	0	0	0	0	0	0	1	1	1	1	1	1	31

2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	D
0	0	0	0	0	0	0	1	0	0	0	0	0	32
0	0	0	0	0	0	0	1	0	0	0	0	1	33
0	0	0	0	0	0	0	1	0	0	0	1	0	34
0	0	0	0	0	0	0	1	0	0	0	1	1	35
0	0	0	0	0	0	0	1	0	0	1	0	0	36
0	0	0	0	0	0	0	1	0	0	1	0	1	37
0	0	0	0	0	0	0	1	0	0	1	1	0	38
0	0	0	0	0	0	0	1	0	0	1	1	1	39
0	0	0	0	0	0	0	1	0	1	0	0	0	40
0	0	0	0	0	0	0	1	0	1	0	0	1	41
0	0	0	0	0	0	0	1	0	1	0	1	0	42
0	0	0	0	0	0	0	1	0	1	0	1	1	43
0	0	0	0	0	0	0	1	0	1	1	0	0	44
0	0	0	0	0	0	0	1	0	1	1	0	1	45
0	0	0	0	0	0	0	1	0	1	1	1	0	46
0	0	0	0	0	0	0	1	0	1	1	1	1	47
0	0	0	0	0	0	0	1	1	0	0	0	0	48
0	0	0	0	0	0	0	1	1	0	0	0	1	49
0	0	0	0	0	0	0	1	1	1	0	0	1	50
0	0	0	0	0	0	0	1	1	0	0	1	1	51
0	0	0	0	0	0	0	1	1	0	1	0	0	52
0	0	0	0	0	0	0	1	1	0	1	0	1	53
0	0	0	0	0	0	0	1	1	0	1	1	0	54
0	0	0	0	0	0	0	1	1	0	1	1	1	55
0	0	0	0	0	0	0	1	1	1	0	0	0	56
0	0	0	0	0	0	0	1	1	1	0	0	1	57
0	0	0	0	0	0	0	1	1	1	0	1	0	58
0	0	0	0	0	0	0	1	1	1	0	1	1	59
0	0	0	0	0	0	0	1	1	1	1	0	0	60
0	0	0	0	0	0	0	1	1	1	1	0	1	61
0	0	0	0	0	0	0	1	1	1	1	1	0	62
0	0	0	0	0	0	0	1	1	1	1	1	1	63
0	0	0	0	0	0	0	1	0	0	0	0	0	64

Math Functions

Math Functions

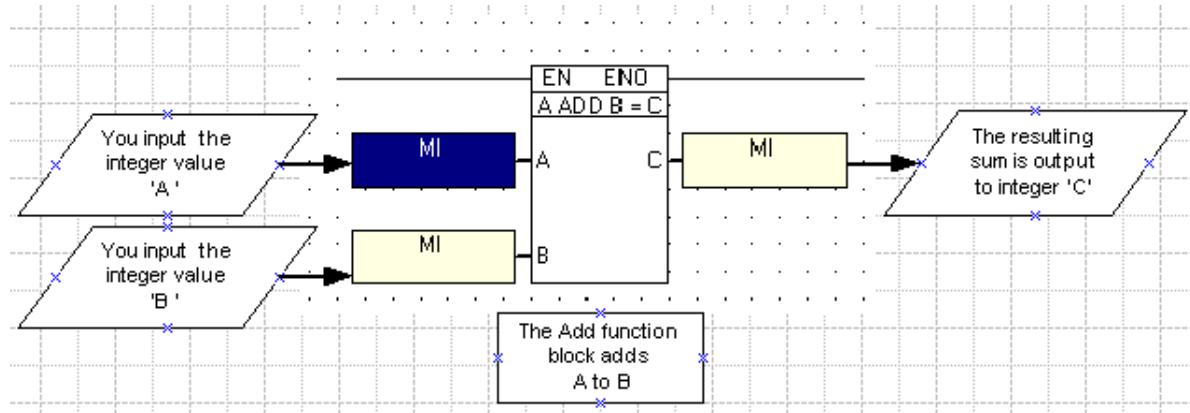
You perform mathematical functions by placing math functions in a net. Math functions, located on the Math menu are provided for:

- Increment/Decrement
- Addition
- Subtraction
- Multiplication
- Division
- Square Root
- Power
- Factor
- Linearization

Each type of math function can use up to 8 input values to compute a single sum.

The internal operation of a function block is transparent to the user.

The example below shows an Add function block with 2 input values.



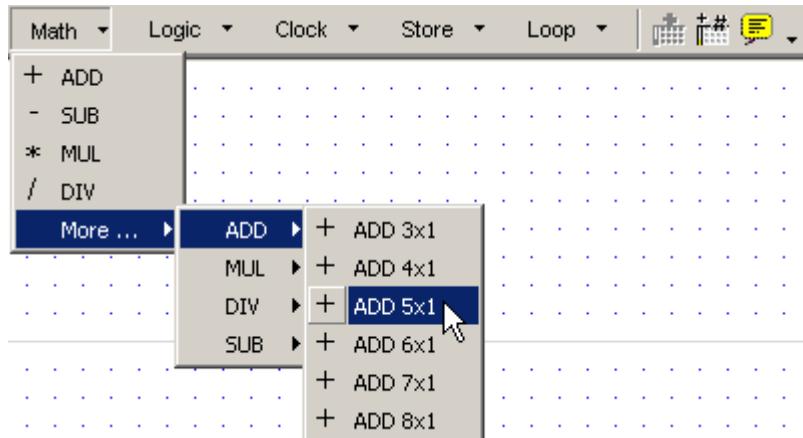
The operands listed below can be used to provide both input and output values, with exception of Constant Values. Constant values can provide input values, but can not contain output values.

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

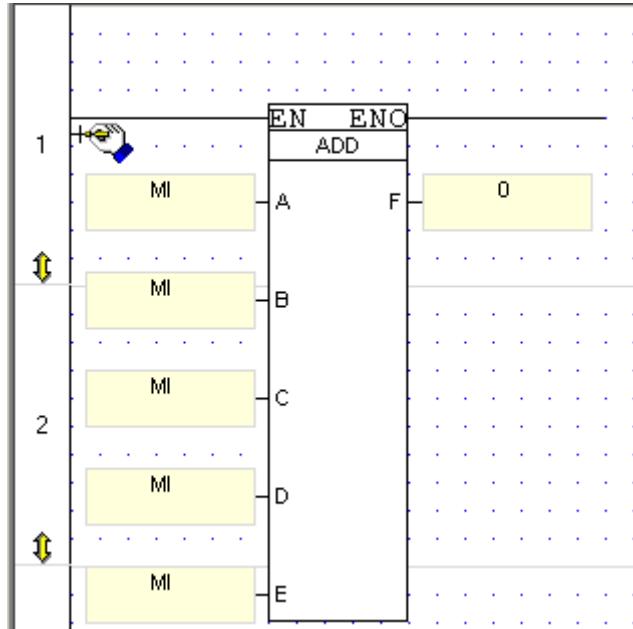
Multiple Input Values in Math Functions

You can input up to 8 values into a math function block. The function will output a single sum. This example shows an Add function that uses 5 input values.

1. Click on the Math button on the Ladder toolbar.
-or-
Right-click on the Ladder to show the Ladder pop-up menu.
2. Select More..., then select the desired function type.
3. Click on the function with the desired number of input values.



4. Move the function to the desired net location, then click. The net automatically enlarges to fit the function



5. Link operands using the Select Operand and Address dialog box. The dialog box opens automatically until all input values and the output value have been linked.

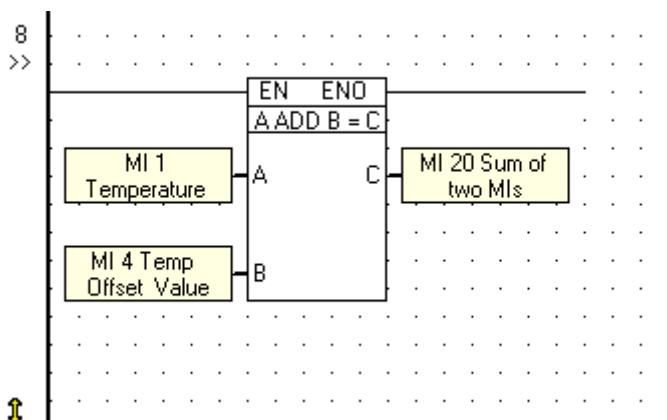
Add

The math function Add is executed by the Add function block shown below. You can choose to add up to 8 input values of the following operand types:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

The example below shows an Add function with two input values.

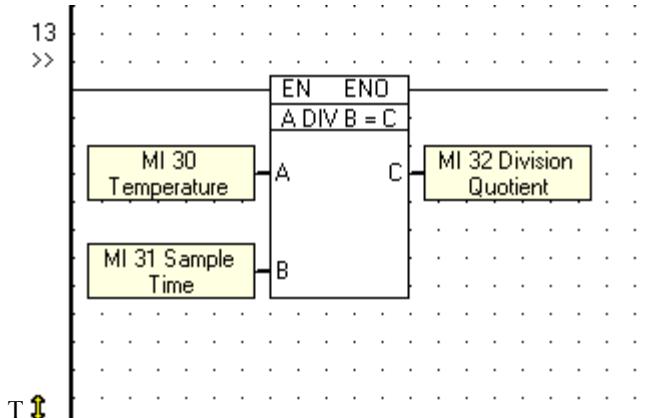


Divide

The math function Divide is executed by the Divide function block shown below. The input values in a Divide function may be:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.



This Divide function can only return whole numbers. To divide floating point numbers, use the Divide function on the Float menu.

Signed remainder values are stored in SL 4 - Divide Remainder (Signed); unsigned results are stored in SDW 4 Divide Remainder (Unsigned).

Note that you must store the remainder values immediately after the division function because these registers will be overwritten by the next division function.

Values may not be divided by zero. In the event that this occurs, System Bit 4 (SB 4 - Divide by Zero) turns ON.

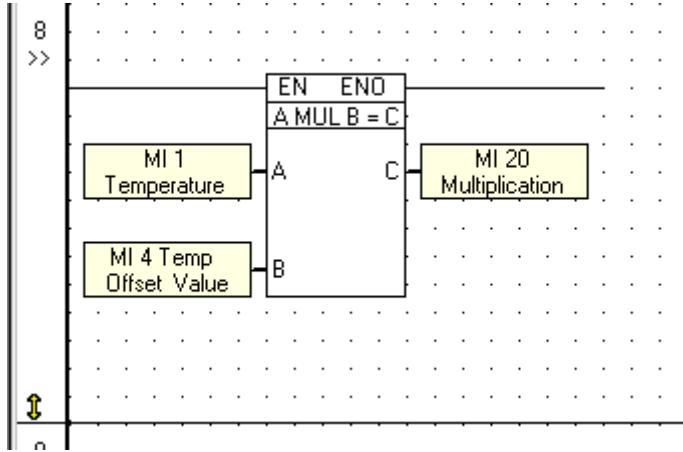
Multiply *

The math function Multiply is executed by the Multiply function block shown below. You can choose to multiply up to 8 input values of the following types:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

The example below shows a Multiply function with two input values.



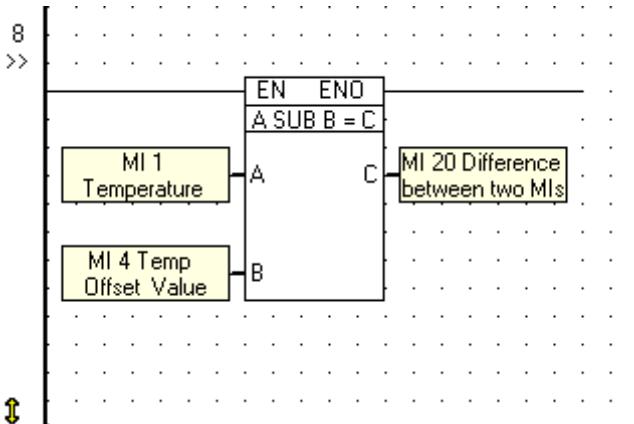
Subtract

The math function Subtract is executed by the Subtract function block shown below. The function is located on the Math menu.

The input values in a Subtract function may be:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Network System Integer (NSI)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

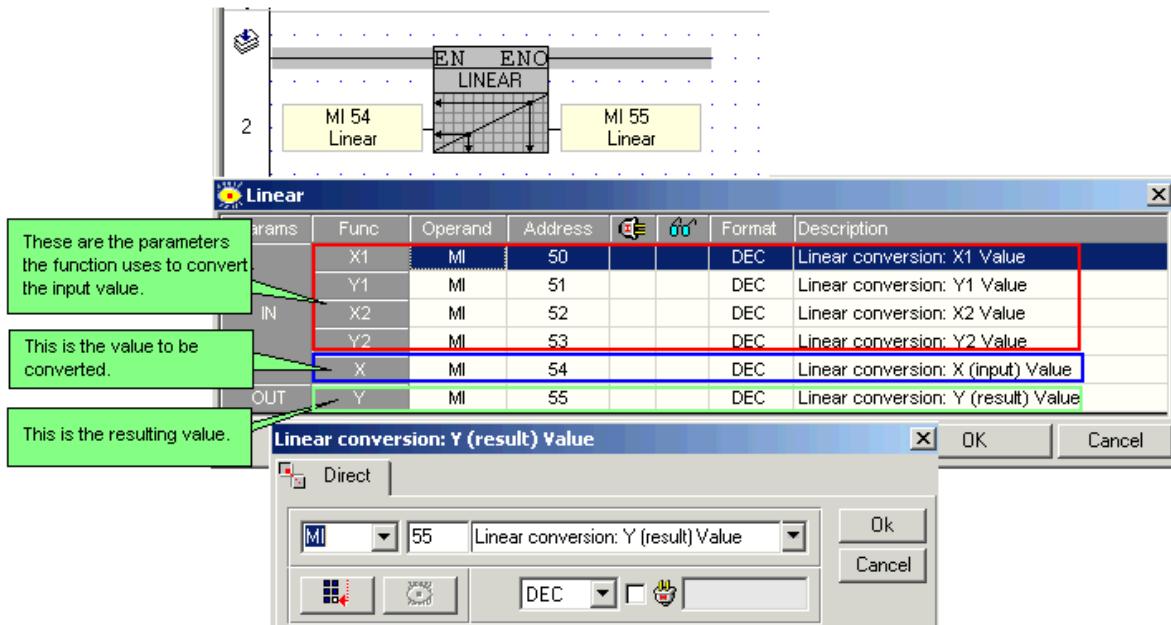


Linearization, Vector Linearization

The Linearization functions, located on the Math menu, enable you to convert values. Use them, for example, to convert analog input values to a values in degrees Celsius.

Linearize a Single Value

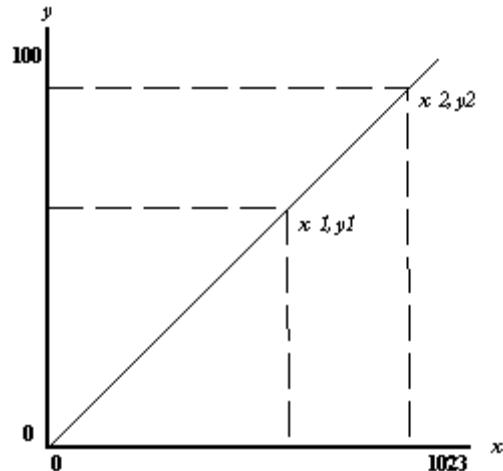
This function linearizes a single source value, then stores it in the target register.



If, for example, X1 and Y1 are 0, and X2=100 while Y2=1023, the output value will be linearized as graphed.

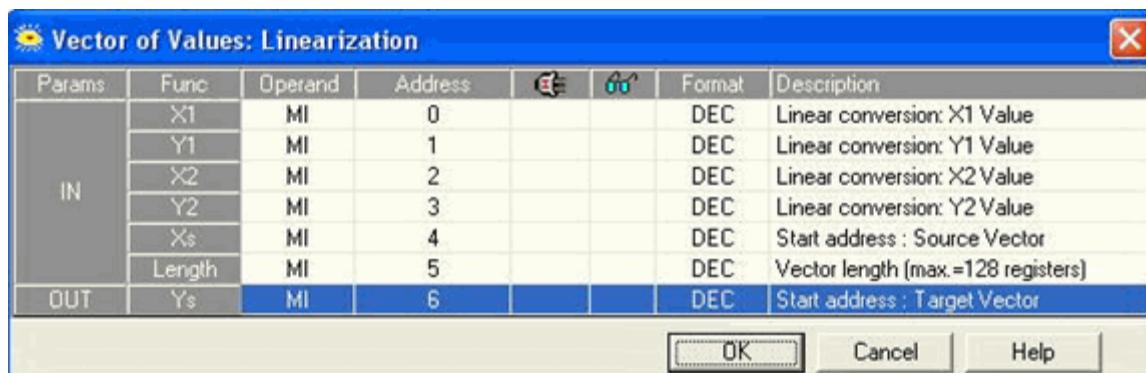
These values would cause:

- A temperature input of 1000 C to be converted to 1023 Digital value.
- A temperature input of 500 C to be converted to 512 Digital value



Linearize a Vector of Values

This function linearizes a vector of source values, then stores the values in the target vector.



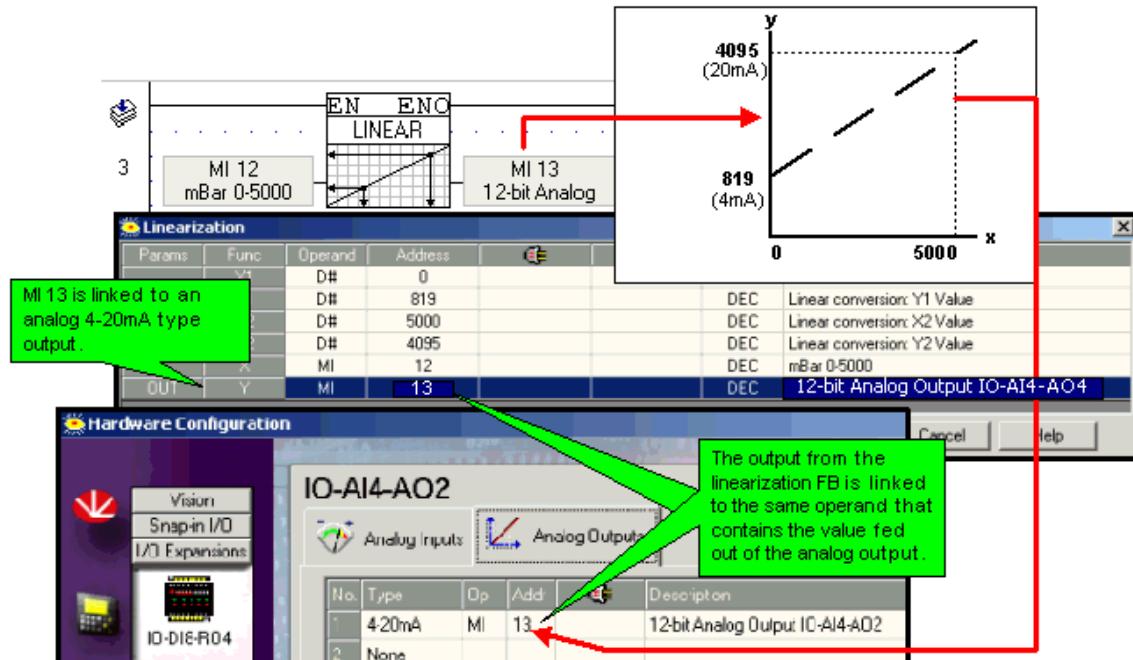
You can convert values contained in the following operand types:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)

With the exception of Constant Value, any of these operands may be used to contain the output value.

Linearizing Analog I/O values

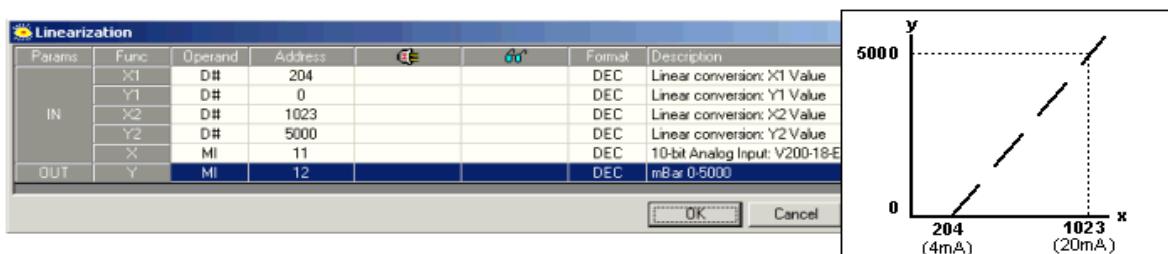
Note • Analog output values are contained in the register that you link to the output in Hardware Configuration.



Working within the 4-20mA range

Available ranges, according to controller and I/O module, are shown in the topic Analog I/O ranges. Note that devices used in conjunction with the controller must be calibrated accordingly. In the examples below, the analog device is a pressure transducer; values are therefore translated to millibars.

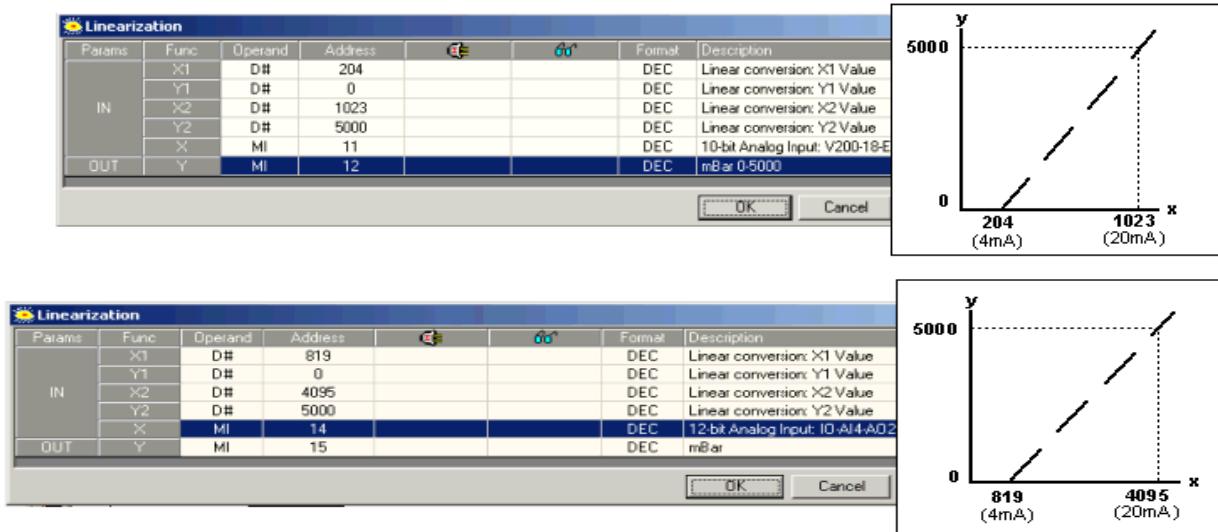
10-bit Analog Input, V200-18-E1



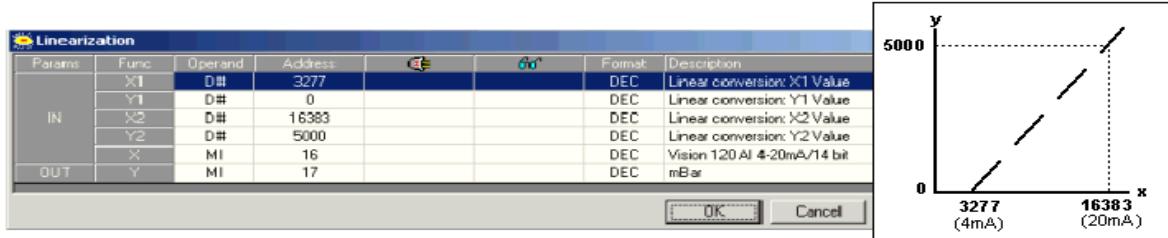
12-bit Analog Output, IO-AI4-AO2



12-bit Analog Input, IO-A14-AO2



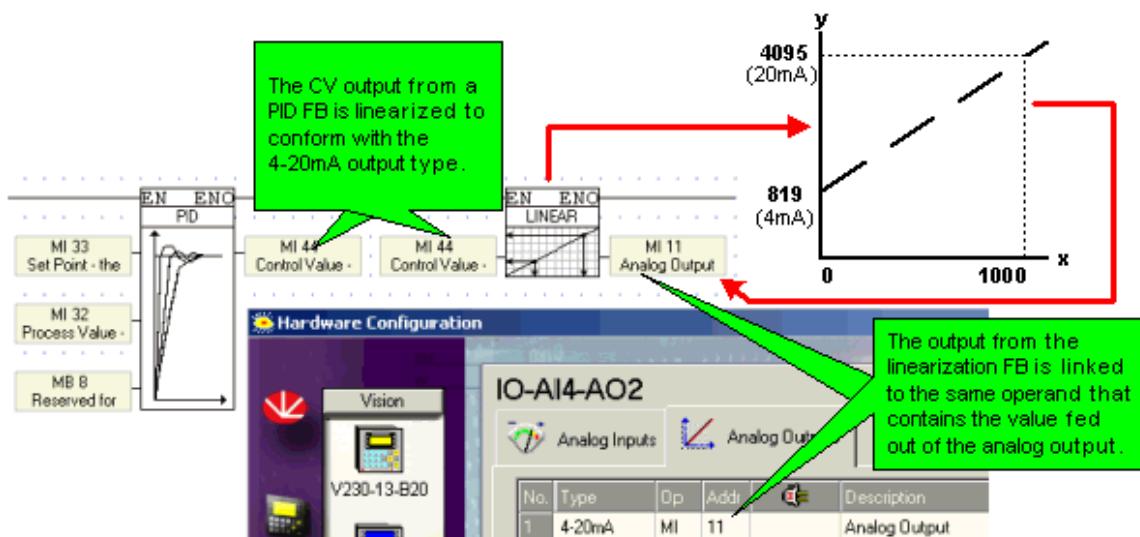
14-bit Analog Input, V120-12-UN2



Linearizing a PID Analog Output Value

Analog values can be converted to physical values, for example Engineering Units (EU) such as degrees Celsius, by using the Linearization FB.

- Note •** Analog output values are contained in the register that you link to the output in Hardware Configuration.

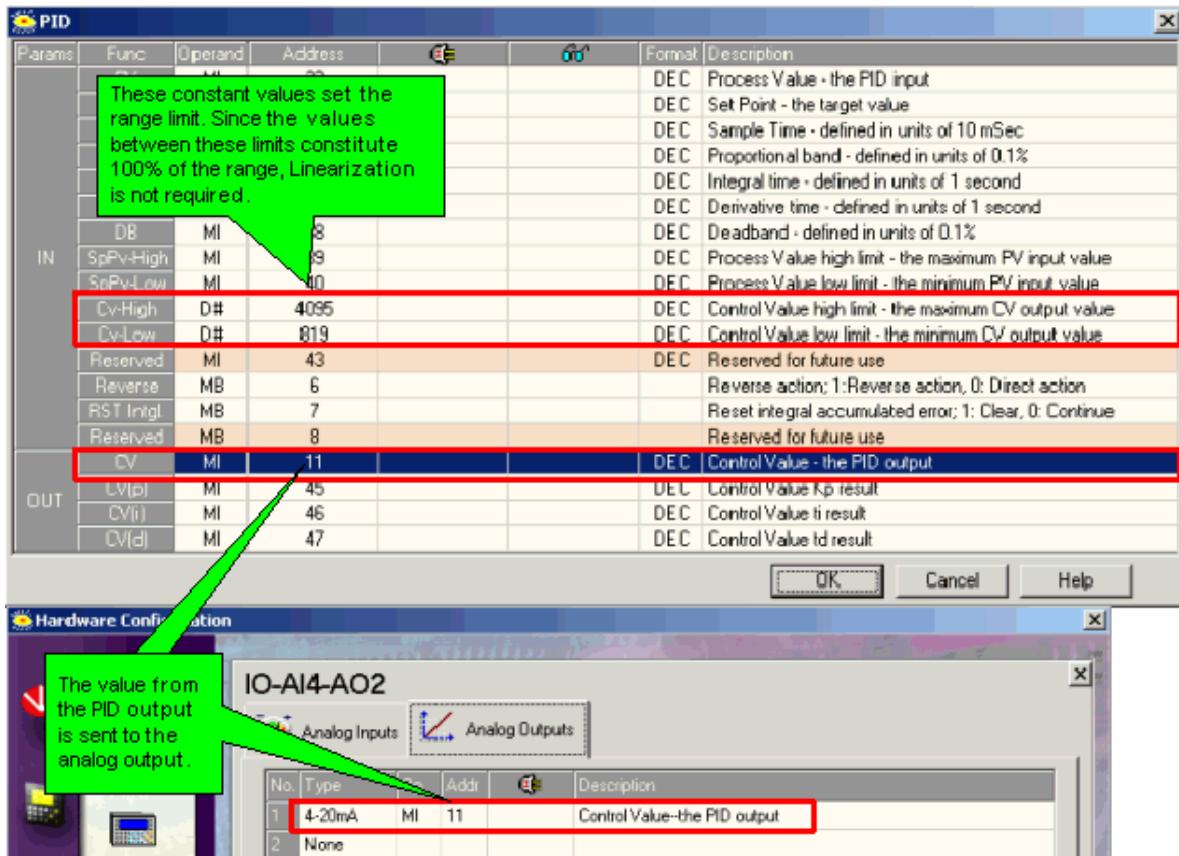


Linearizing a PID output-to-analog output

Working within the 4-20mA range

Available ranges, according to controller and I/O module, are shown in the topic Analog I/O ranges. Note that devices used in conjunction with the controller must be calibrated accordingly.

Limits can be set for the output range, in this case linearization is not required.



Factor



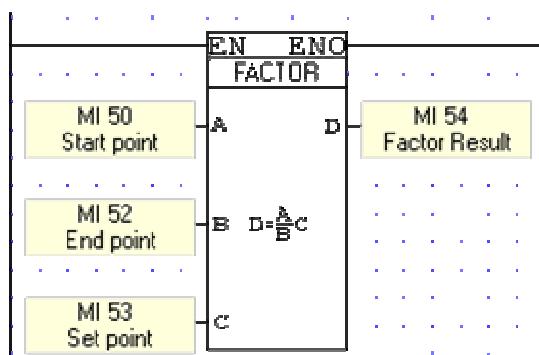
The math function Factor uses 3 input values. Factor divides an A input value by a B input value and then multiplies the result by a C input value. The result is stored in an output operand, D.

You can use the following operand types in this operation:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

The example below shows a Factor function.



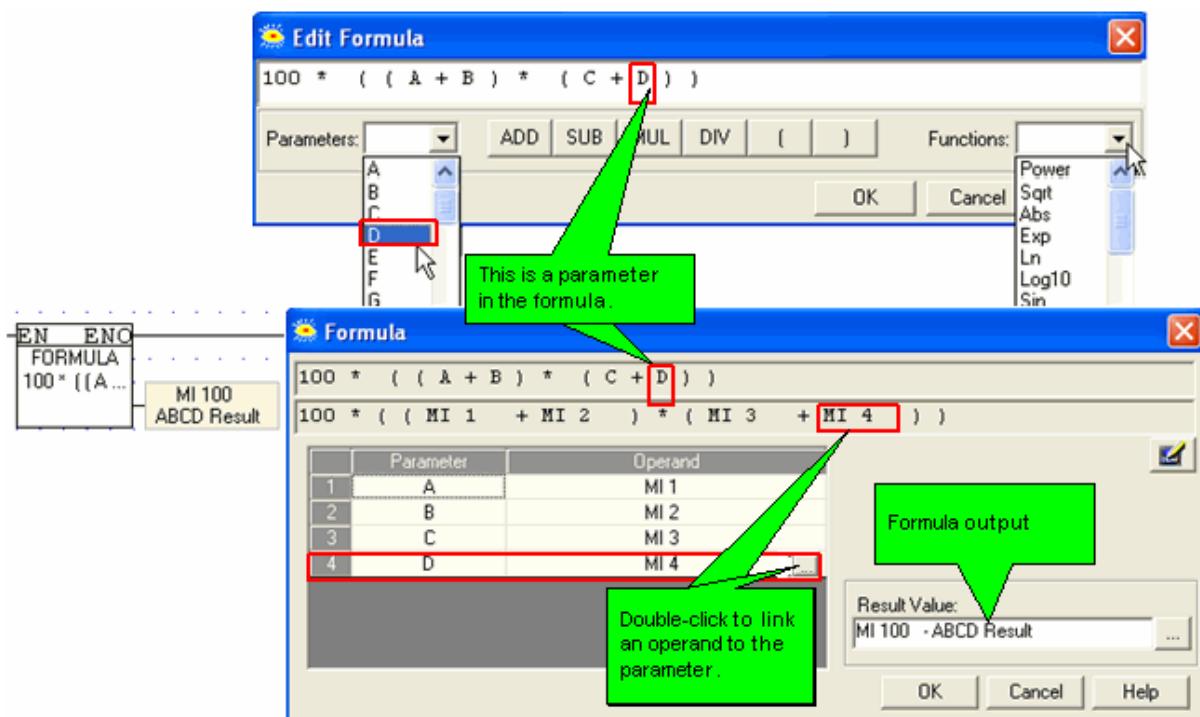
Formula: Build Your Own

The Formula function, located on the Math menu, enables you to apply mathematical operators to operand values, and then output the result to a register.

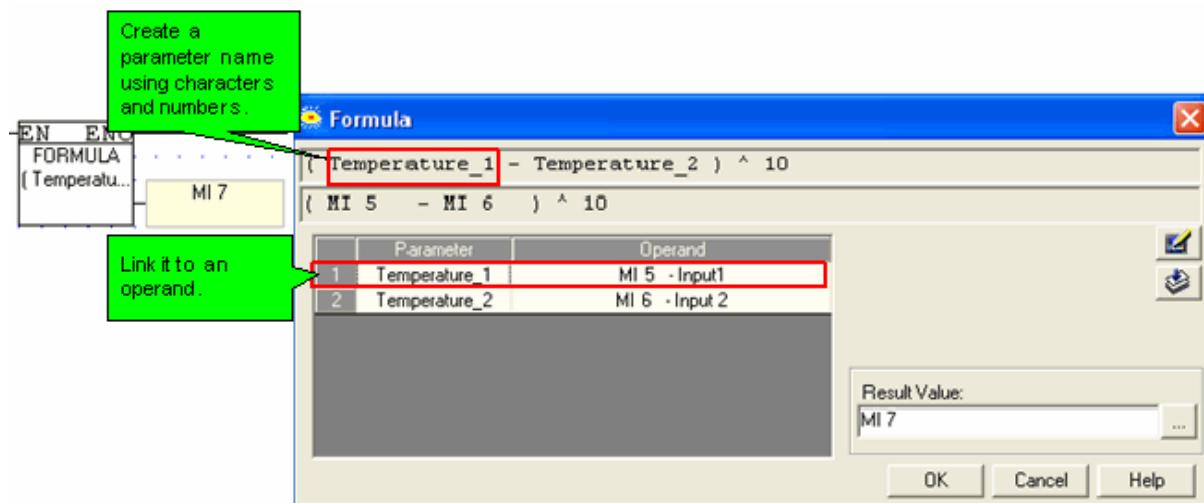
To create a formula, place the Formula function in the Ladder; the Edit formula box opens. You can type in constant numbers, parameters and operators. You can also select parameters and operators from the drop-down lists.

Note • The formula syntax conforms to normal mathematical notation.

- With the exception of the - (minus) sign, binary operators cannot be used to begin a formula. The other binary operators include Add [+], Mul [*], Div [/], Parenthesis [()], and Power. Unary operators, such as Sin, may be used to begin a formula.



You can create a parameter name using a mixture of characters and numbers.



Note • A parameter name may not begin with a number or contain spaces. Use an underscore (_) in place of spaces.

- A constant may not exceed the value of a MF or ML.
- In the following cases, controller will process the formula using floating registers:
 - If the formula contains one or more floating operands.
 - If a constant value in the formula is not a whole number
 - If an operator, such as trigonometric operators, requires that the PLC use a floating register to complete its operation.

Power

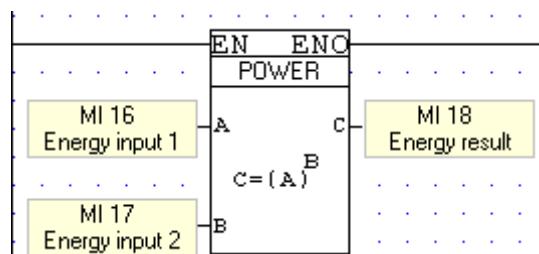
The math function Power uses 2 input values. Power raises an A input value by the power of a B (exponent) input value. The result is stored in an output operand, C. The function is located on the Math menu.

You can use the following operand types in this operation:

- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)
- Constant Value #

With the exception of Constant Value, any of these operands may be used to contain the output value.

The example below shows a Power function.



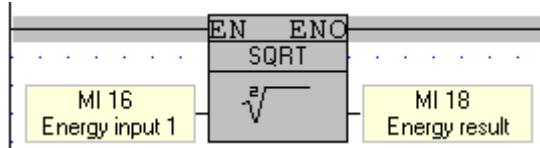
Square Root

This function returns the square root of an input value. The input value serves as the radicand. The result is stored in an output operand. The function is located on the Math menu.

You can find the square root of values contained in the following operand types:

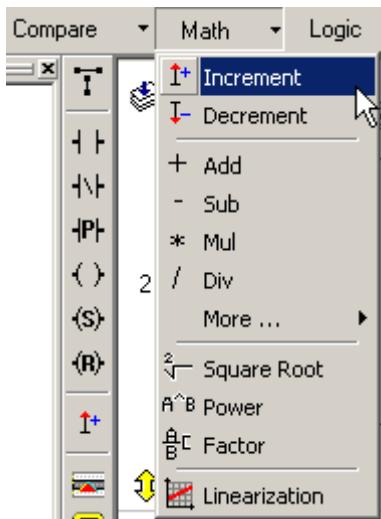
- Memory Integer (MI)
- Memory Long Integer (ML)
- Double Word (DW)
- System Operands:(SI) (SL)(SDW)

The example below shows a Square Root function.



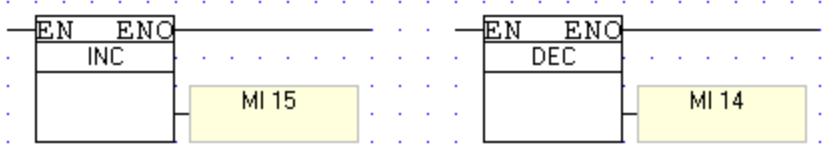
Increment/Decrement

These functions are located on the Math function menu; an Increment button is also located on the shortcut toolbar.



Increment increases the value in the selected operand by 1.

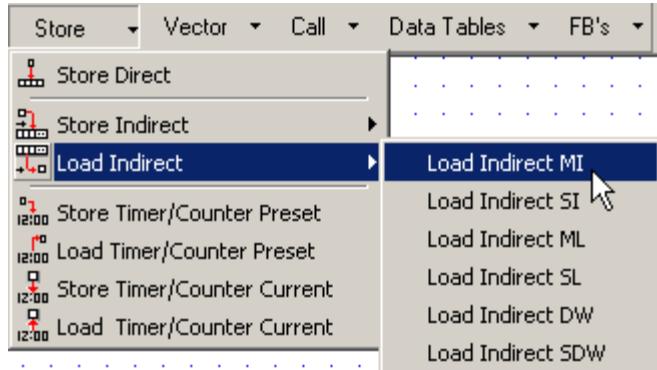
Decrement decreases the value in the selected operand by 1.



You can implement counters in your program by selecting a Counter (C) operand output type.

Store & Load Functions

Store and load functions can be used to copy values from an operand, or range of operands, to another. You access both types of functions from the Store menu.



The available functions are listed below.

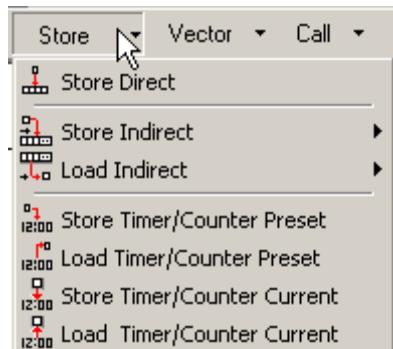
- Store Direct Function
- Store Indirect Function
- Load Indirect Functions
- Store Timer/Counter Preset
- Load Timer/Counter Preset
- Store Timer/Counter: Current Value
- Load Timer/Counter: Current Value

Store Direct Function

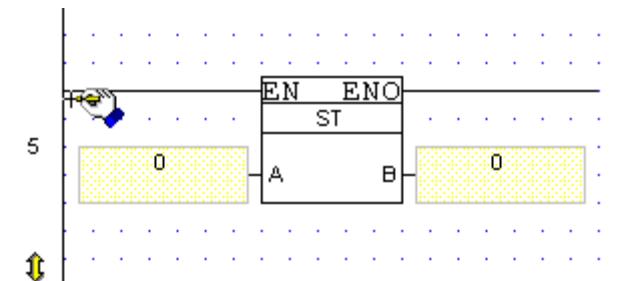
Store Direct allows you to write a value contained in an operand or constant into another operand.

To use the Store Direct function:

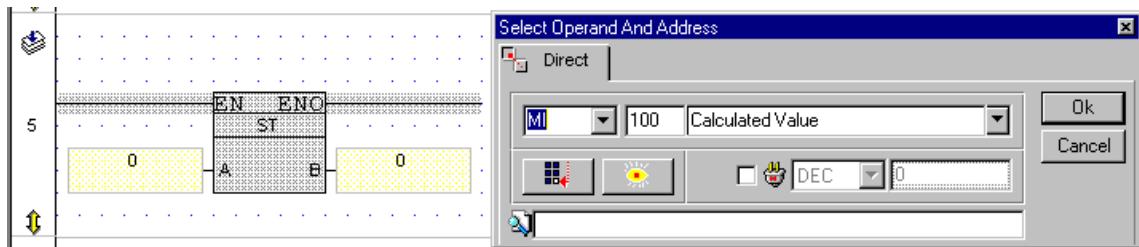
1. Click Store on the Ladder Toolbar.



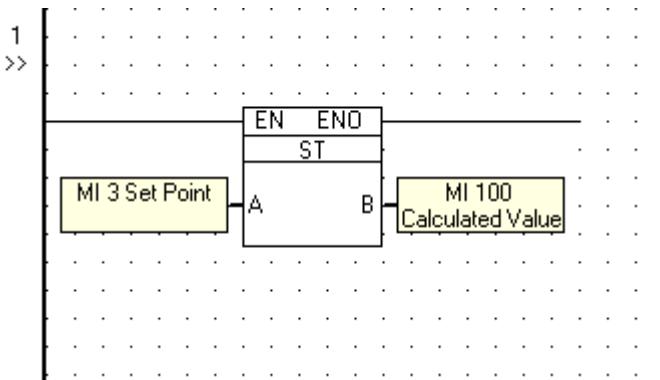
2. Select Store Direct, then place the Store Direct function in the desired net.



3. Enter the desired Operands and Addresses.



4. The Store Direct element appears on the net with the set Operands and Addresses.

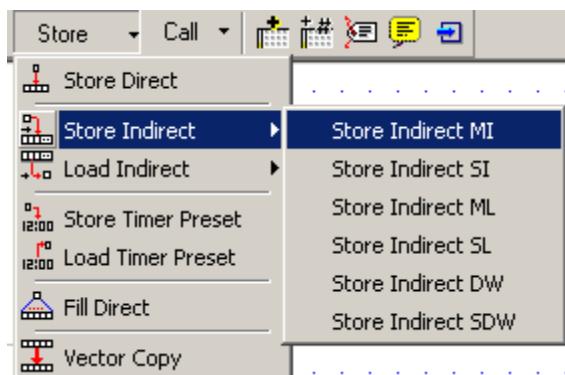


According to the above example, the value in MI 3 will be stored in MI 100. The previous value in MI 100 is overwritten. The current value in MI 3 remains unchanged.

Store Indirect Function

Store Indirect allows you to write a value contained in certain types of operands into another operand using indirect addressing. The 'B' output parameter of the Store Indirect function is actually a pointer to another operand.

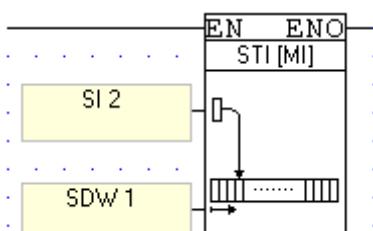
When you select the function type from the Store menu, the program writes the input A value into the address referenced by the output B value--according to the type of function you select.



Example: Store Indirect MI

In the example below, SI2 contains the value 5 and SDW1 contains the value 10. Since the function type is Store Indirect MI, MI10 is where the value in SI2 will be stored.

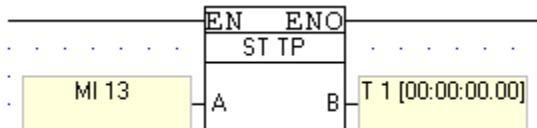
The value 5 will therefore be stored in MI 10.



Store Timer/Counter Preset

You can set a Timer or Counter preset value by storing an operand or constant value into the desired operand.

- Operand A: contains the value to be stored in the timer/counter.
- Operand B: this is the timer/counter to be preset.

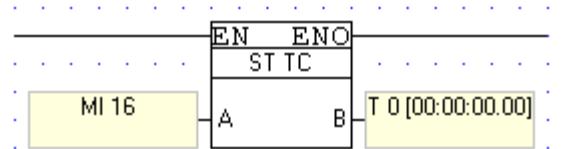


Note • The value that is stored in the Timer is broken down into units of 10 milliseconds. In the above example, if MI 13 is equal to 1023, the value stored into T1 will be 10 seconds and 230 milliseconds.

Store Timer/Counter: Current Value

You can store an operand or constant value into a current Timer or Counter value.

- Operand A: contains the value.
- Operand B: this is the timer/counter where the value will be stored.



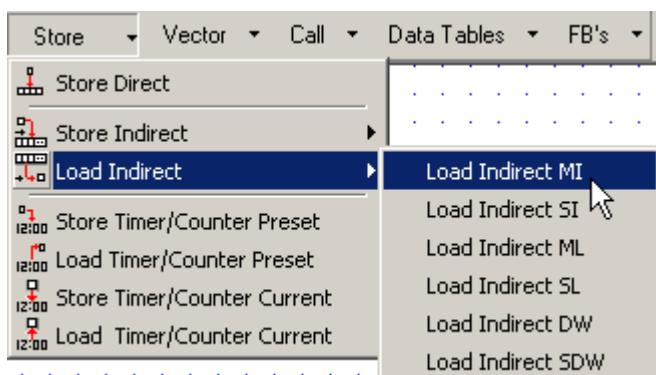
Note • The value that is stored in the Timer is broken down into units of 10 milliseconds. In the above example, if MI 16 is equal to 1023, the value stored into T0 will be 10 seconds and 230 milliseconds.

Load Indirect Functions

Load Indirect allows you to take a value contained in a source operand and load it into a destination operand using indirect addressing.

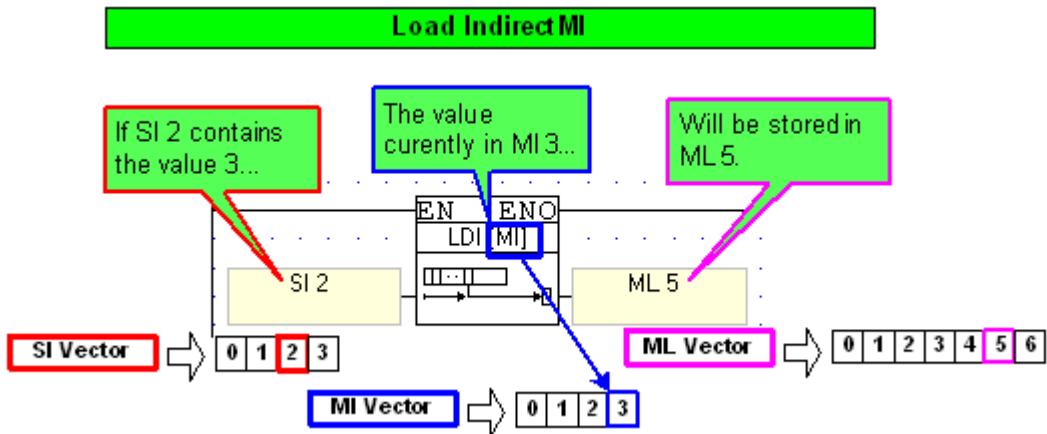
The example below is based on a Load Indirect MI function.

1. Click Store on the Ladder Toolbar, then select Load Indirect MI from the Load Indirect menu.



2. Place the function in the desired net.

3. Link the desired Operands and Addresses. The first operand contains the offset address. In the figure below, SI 2 is linked to the first operand. This is a Load Indirect MI function; therefore if SI contains 3, the function will take the value in MI 3 and store it in ML 5, the second linked operand.

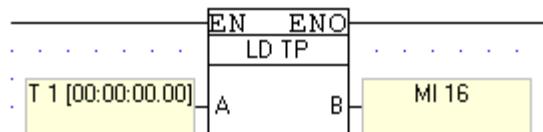


According to the above example, if the value in MI 3 is 986, 986 will be stored in ML 5. The previous value in ML 5 is overwritten. The current value in MI 3 remains unchanged.

Load Timer/Counter Preset

You can load the preset value of a Timer or Counter into an operand.

- Operand A: this is the Timer/Counter preset value.
- Operand B: this is where the value will be stored.

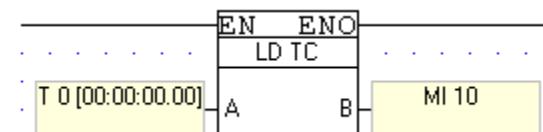


Note • Timer value units are 10 milliseconds. In the above example, if TI is equal to 10 seconds and 23 milliseconds, the value 1023 will be stored into MI 16.

Load Timer/Counter: Current Value

You can load the current value of a Timer/Counter into an operand.

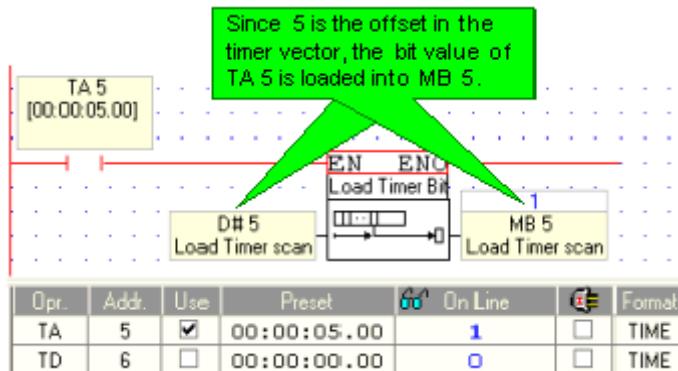
- Operand A: this is the Timer/Counter current value.
- Operand B: this is where the value will be stored.



Note • Timer value units are 10 milliseconds. In the above example, if T0 is equal to 10 seconds and 23 milliseconds, the value 1023 will be stored into MI 10.

Load Timer Bit Value

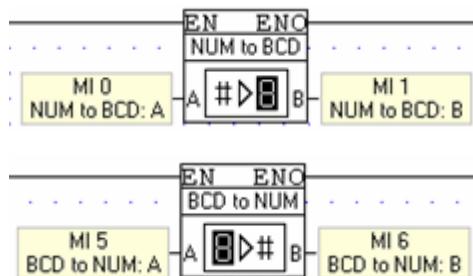
You can use a Ladder condition to load the current bit value of a Timer into an MB. The input to the Load Timer Scan Bit function is the address of the timer within the Timer vector, and may be a constant or a value provided by a register.



BCD to NUM, Num to BDC

You can convert a numeric value into a BCD or a BCD to a numeric value by using the appropriate function.

1. Select the function from the Store menu on the Ladder toolbar.
2. Place the function in the net.
3. Link the parameters to the desired operands.

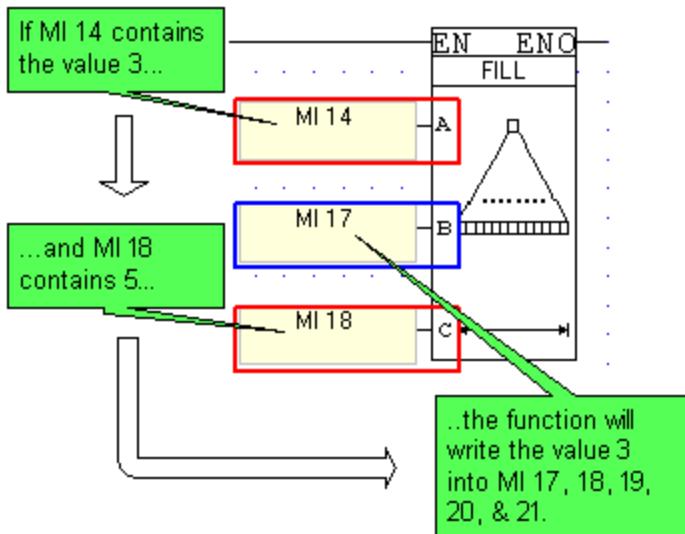


Notes • This type of BCD may be used in seven-segment displays, composed of seven elements.

Fill Direct

Fill Direct enables you to set a range of numeric operands or MBs. The function copies a value from a desired operand, then writes that value into every operand within in the set range.

- Operand A: this is the operand which contains the value to be copied.
- Operand B: this is the first operand in the range.
- Operand C: this sets the length, meaning the number, of operands in that range.



Clock Functions

Program clock and calendar functions in the Ladder by selecting the appropriate functions from the Clock menu on the Ladder toolbar. Function are provided for:

- Time
- Day of the Week
- Day of the Month-Direct and Indirect
- Month
- Year
- UTC (Universal Time) functions

Setting a Clock Function's Time or Date

- Direct Clock function:
The time or date of a Direct Clock function is set within the function you place in your program.
- Indirect Clock Function:
Indirect Clock functions are linked to registers. Values may be placed into the linked register by your application, or may be entered via the controller keypad.

UTC (Universal Time) Functions

VisiLogic offers the following UTC functions:

Clock menu	<p>UTC to RTC</p> <p>The value in a DW is converted to a real-time clock format. Sending the value to SI 30 will set the controller's RTC by automatically overwriting SIs 30-34.</p> <p>RTC to UTC</p> <p>Selecting SI 30 will convert the RTC value into a DW.</p>
Com>TCP/IP menu	<p>RFC-1305</p> <p>Retrieves, via Ethernet UDP, the current time from a PC UTC server. This may be used to synchronize a Vision RTC with UTC.</p>
HMI Clock Variables	<p>Clock Display Variable, UTC</p> <p>This may be set as read only, or as a Keypad Entry variable used to set the RTC.</p>

Note • Note that these functions use the DW as a 32-bit binary number containing the UTC value in seconds, where 1900-01-01 = 00:00.00 UTC. Vision controllers support a range from 2004 to 2024.

• Since the DW is the value in seconds, you can perform time value calculations. For example, you can convert the RTC values to DWs, then calculate the difference in order to figure a time interval.

Both protocols use a standardized data format that refers to UTC (Coordinated Universal Time), and to no other time zones. They are used to synchronize timekeeping among a set of distributed time servers and clients.

RFC-868

The controller sends the time request and receives the response via TCP/UDP port 37. The protocol uses a 32-bit binary number (seconds since 1900-01-01 00:00:00 UTC). This base will serve as the standard until time stamp 4294967295, which will be on 2036-02-07 06:28.14 UTC.

The protocol cannot estimate network delays or report additional information.

RFC-1305

The controller sends the time request and receives the response from the PC server via UDP port 123.

RFC-1305 uses NTP (network time protocol), a very sophisticated protocol between NTP servers and multiple peers, based on unicast and multicast addressing. A NTP timestamp is represented as a 64-bit unsigned fixed-point number (seconds since 1900-01-01 00:00:00 UTC). The integer part is in the first 32 bits and the fraction part of the second is in the last 32 bits. The maximum number is 4294967295 seconds with a precision of about 200 picoseconds.

UTC: Setting/Synchronizing the Real Time Clock (RTC) via Ladder

Via VisiLogic's UTC functions, you can set the Real Time Clock (RTC) within an Ethernet-enabled Vision controller. Via Ethernet, you can:

- Synchronize the RTC's of networked Vision controllers (RFC-868).
- Synchronize the RTC of a controller to a PC server. (RFC-1305)

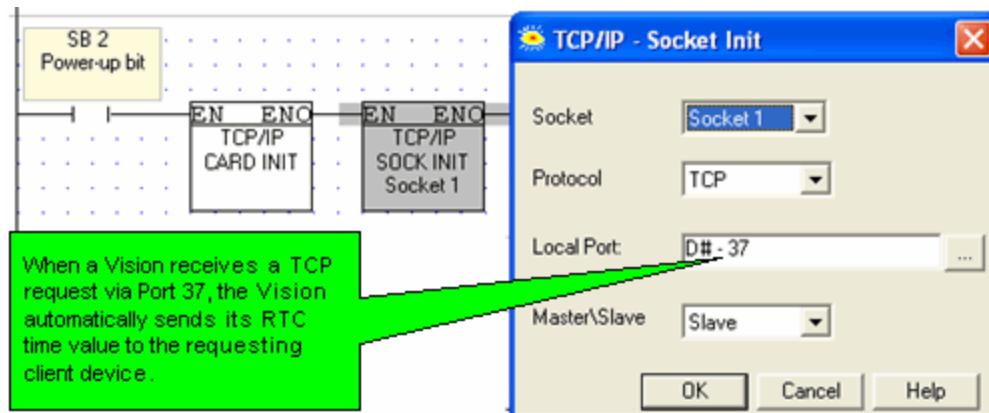
Using RFC-868 to synchronize networked controllers

When a Vision receives a TCP request via Port 37, the Vision 'server' automatically sends its RTC time value to the requesting client device.

In the Vision 'server' :

1. Initialize the TCP/IP card and initialize a socket to TCP, Local Port 37, Slave as shown in the following figure.

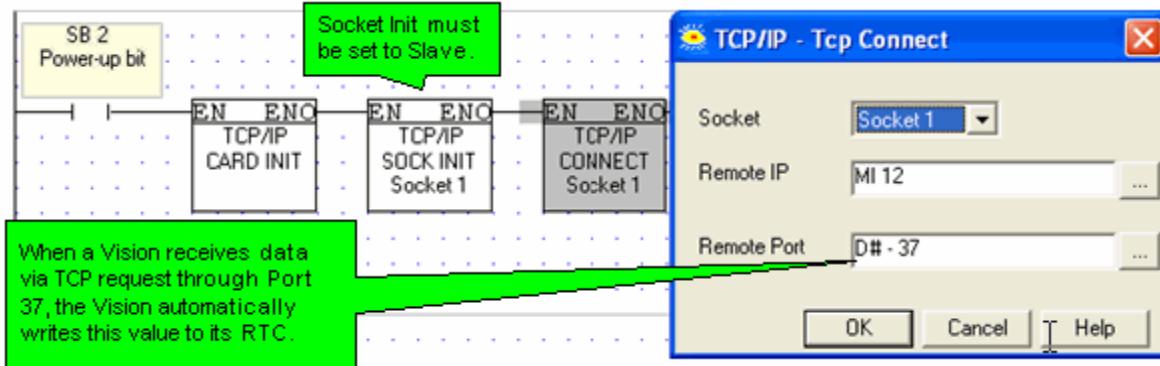
When a Vision receives a TCP request via Port 37, the Vision automatically sends its RTC time value to the requesting client device.



In a Vision requesting the time:

1. Initialize the TCP/IP card and initialize a socket to TCP, Master.
2. Place a TCP/IP Connect function, set to Remote Port 37, as shown in the following figure.

When a Vision receives data via TCP request through Port 37, the Vision automatically sets its RTC, writing this value to all RTC SIs, 30 to 34.



Using RFC-1305 to synchronize a Vision's RTC to a UTC PC server

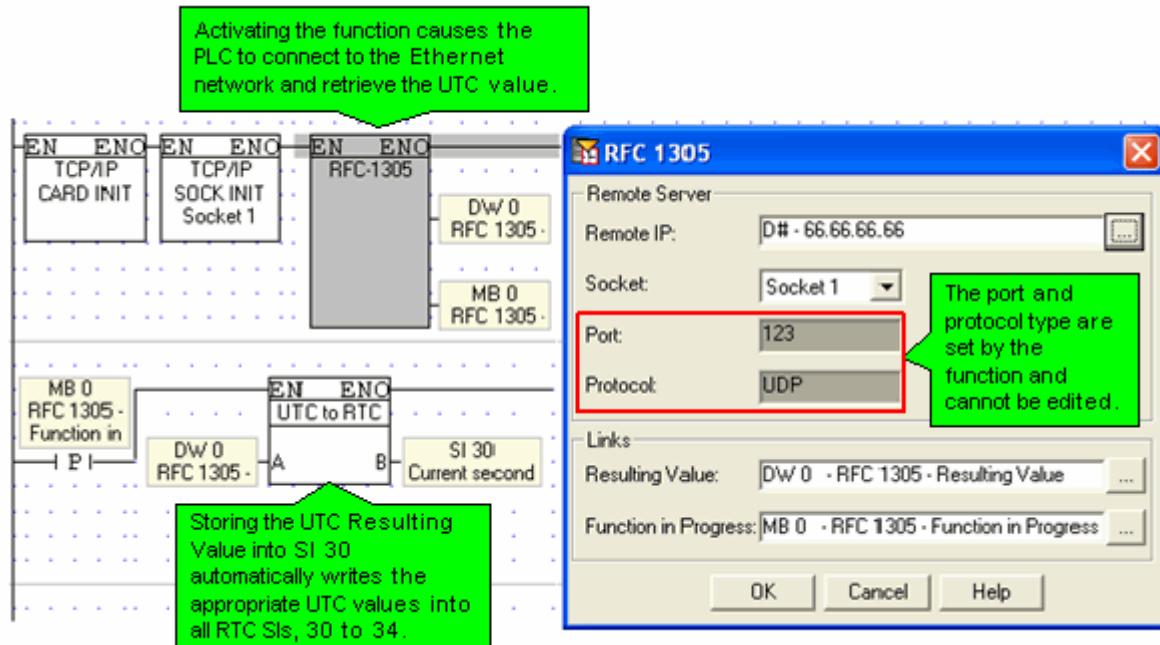
When a UTC PC server receives a UDP request via Port 123, the server automatically sends the time value to the requesting client device.

To request the data from the server, use the RFC-1305 function, located in Com>TCP\IP.

1. Initialize the TCP/IPcard and initialize a socket to UDP.
2. Place the RFC-1305 function in the net, entering the PC server's IP address and the socket set in Socket Init. Note that the Protocol type and Port are set by default.

To write the time value received from the server into the controller and set the RTC, use the UTC to RTC function, located in Clock> UTC.

1. Link a positive transition contact to the RFC-1305 Function in Progress MB
2. Place a UTC to RTC function as shown in the following figure. Storing the UTC Resulting Value into SI 30 automatically writes the appropriate UTC values into all RTC SIs, 30 to 34, setting the RTC.



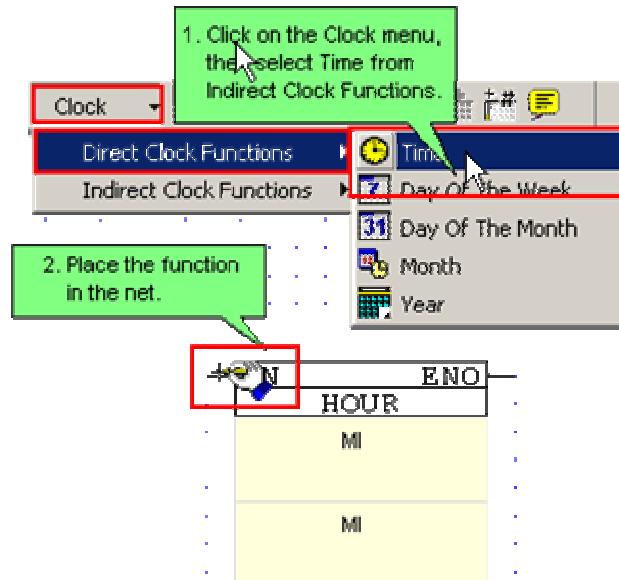
Clock: Direct Function Example

This example shows you how to build a ladder net that drives a coil:

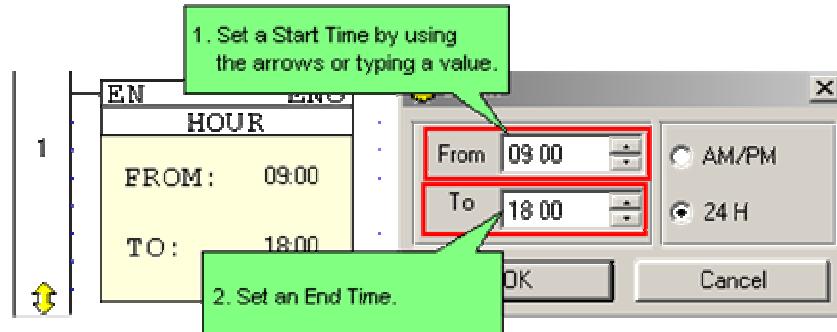
- between the hours 9:00 am and 6:00 PM.
- Monday through Friday
- beginning on the 15th day of a month, until and including the 24th
- in the years 2000 and 2001

Remember that the elements must touch to enable power flow to the coil.

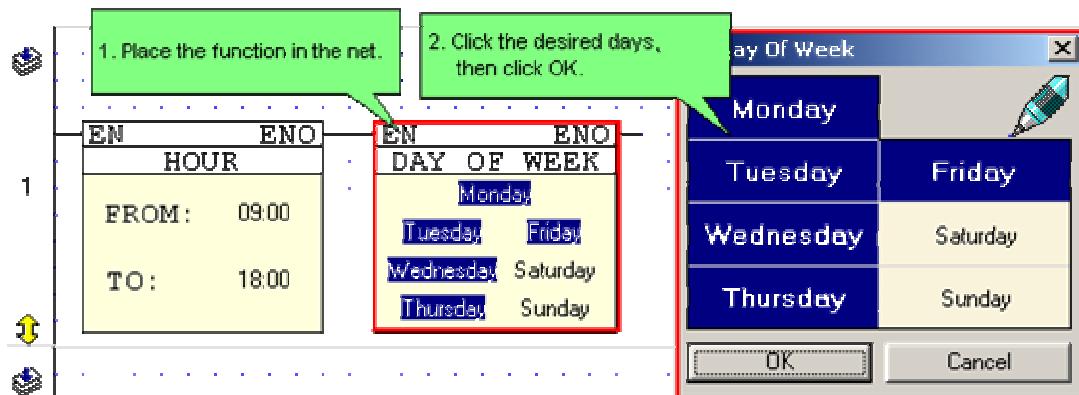
1. Place a Direct Time Function in the net.



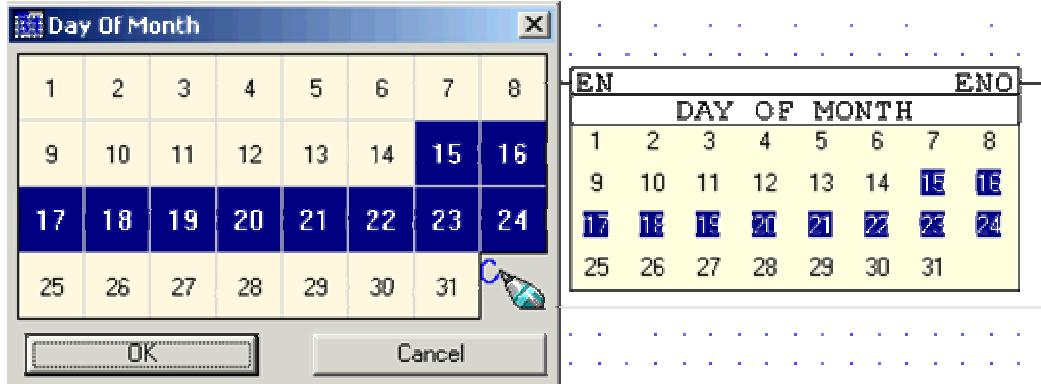
2. Set a Start and End Time. When the RTC is within this range, power flows through the function block.



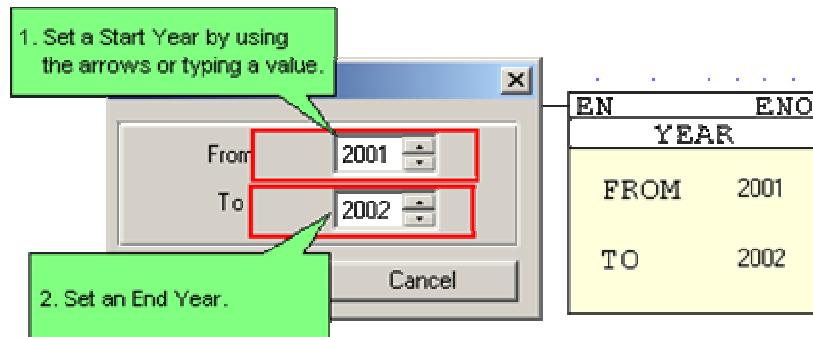
3. Select Day of the Week , place it in the net, then select the desired days.



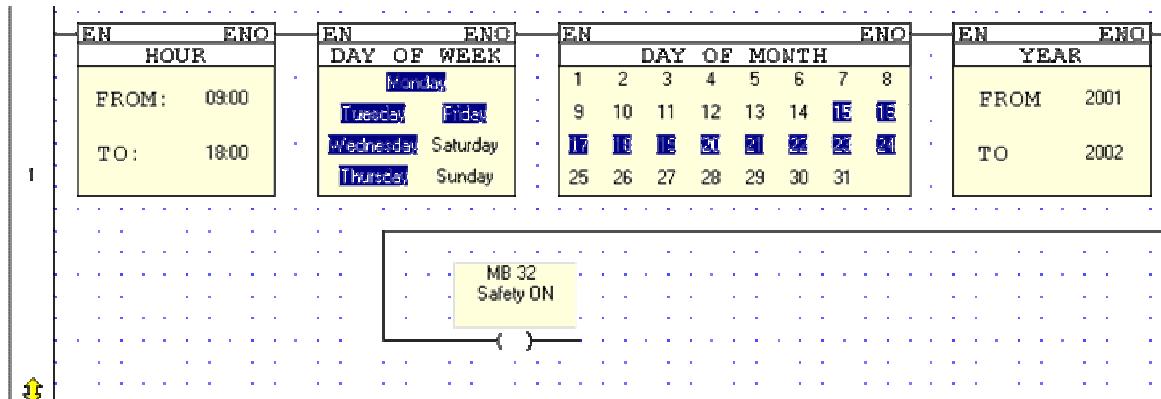
4. Select Day of Month , place it in the net, then select the desired dates.



5. Select Year, then enter the year.



6. Enlarge the net, place and link a coil, then use the Connect Elements Tool to draw lines between the elements.



Clock: Indirect Function Example

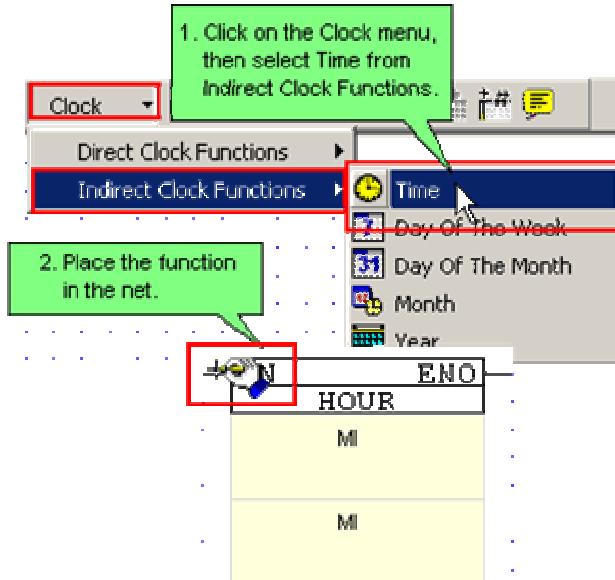
To enable times and dates for tasks or programs to be set from the controller keypad, you:

- Place Clock function blocks in the Ladder.
- Create HMI Displays that include keypad-entry Time Function Variables. This type of Variable accepts a time value that is entered via the controller keyboard, storing the number in the linked operand.

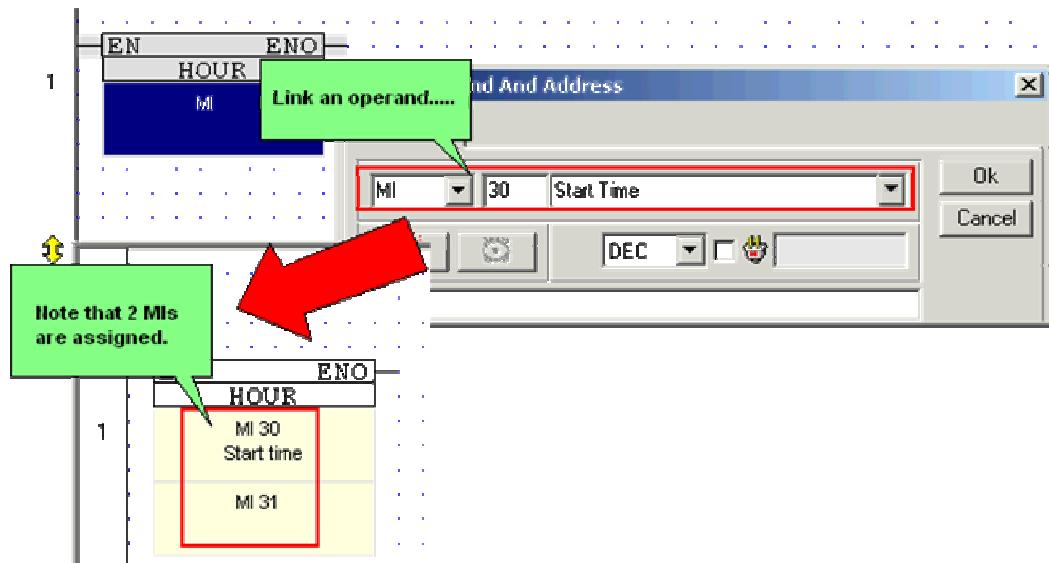
This example shows you how to build a ladder net that drives a coil according to the time and date, and how to build the HMI Displays, add the required Variables and jump between Displays.

Building the Ladder

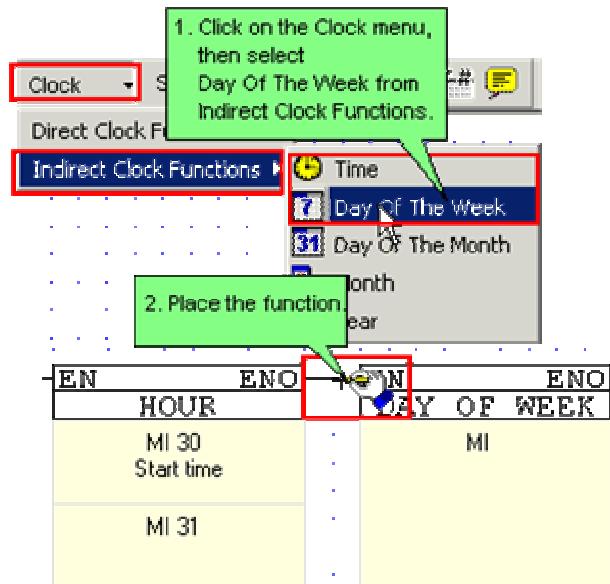
1. Place an Indirect Time Function in the net.



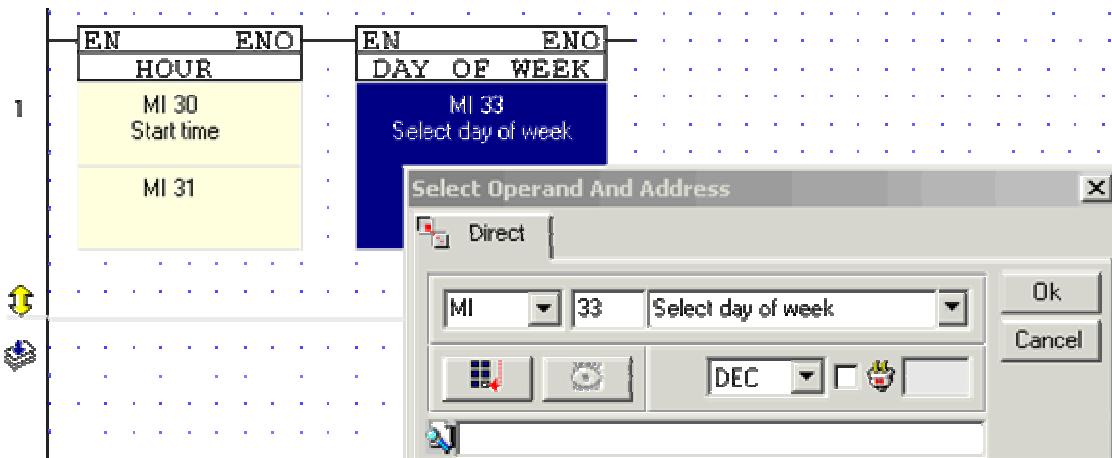
1. Click on the Clock menu, then select Time from Indirect Clock Functions.
 2. Place the function in the net.
2. Link an operand. The Time function requires two consecutive MIs; the second is automatically assigned by the program. These 2 MIs define a time range. The first MI sets the Start Time for the function, the second MI marks the End Time. When the RTC is within this range, power flows through the function block.



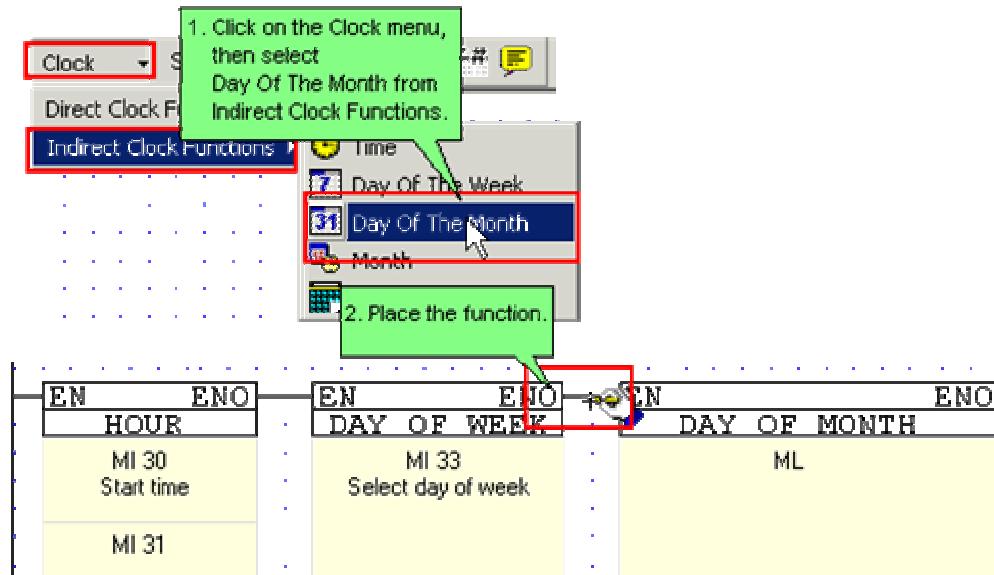
3. Place a Day Of The Week function so that it touches the first function, enabling power flow. This function uses a 16-bit register to contain a 7-bit bitmap representing the days of the week.



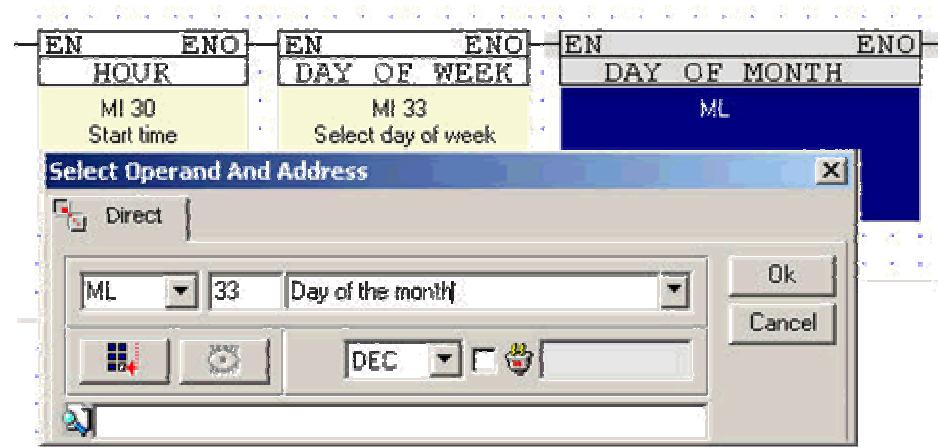
4. Link an operand.



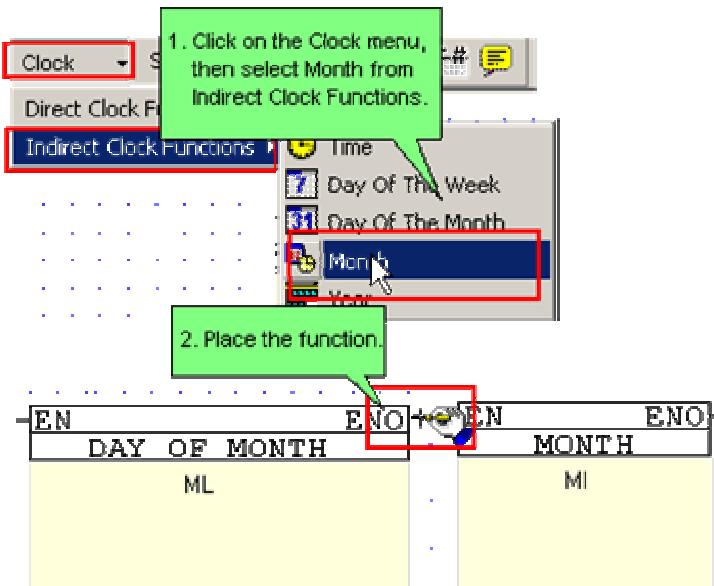
5. Place a Day of The Month function so that it touches the last function.



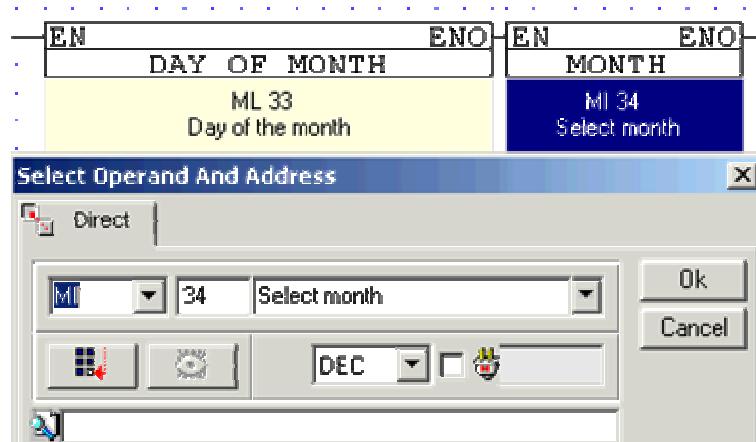
6. Link an operand. This function uses an ML or SL to contain a 32-bit bitmap.



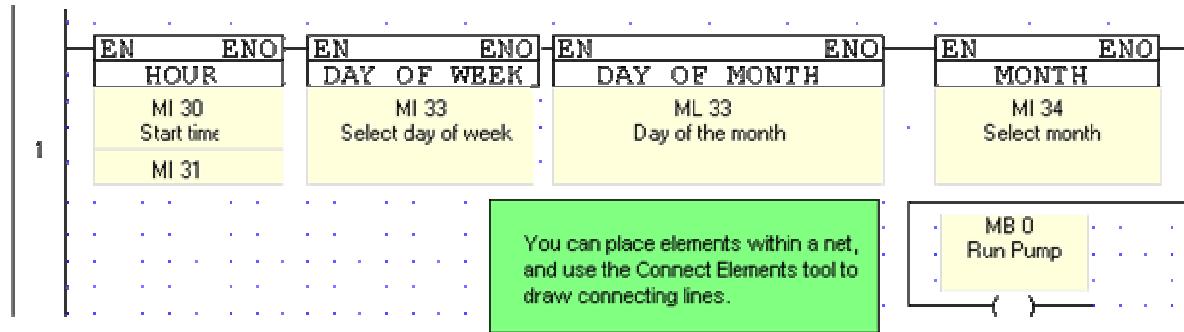
7. Place a Month function so that it touches the last function.



8. Link an operand.



9. Place a Direct Coil in the net as shown below, and link an operand. The Ladder net is complete; now create the supporting HMI Displays and Variables.



You build the net using Indirect Time functions.

Building the HMI Displays

Here, you will create variables that enable Start Time, End Time, Day of Week, and Day of Month, and month to be set from the controller keyboard.

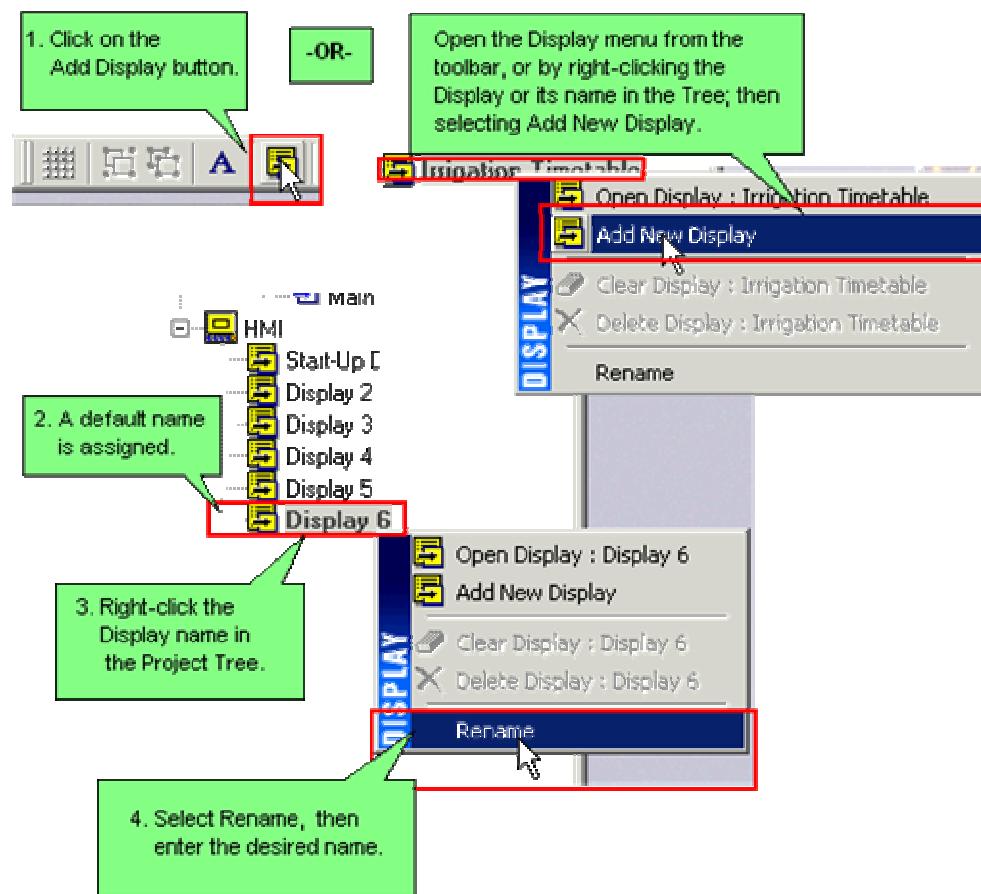
Start & End Time Variables

1. Open the HMI Display editor.

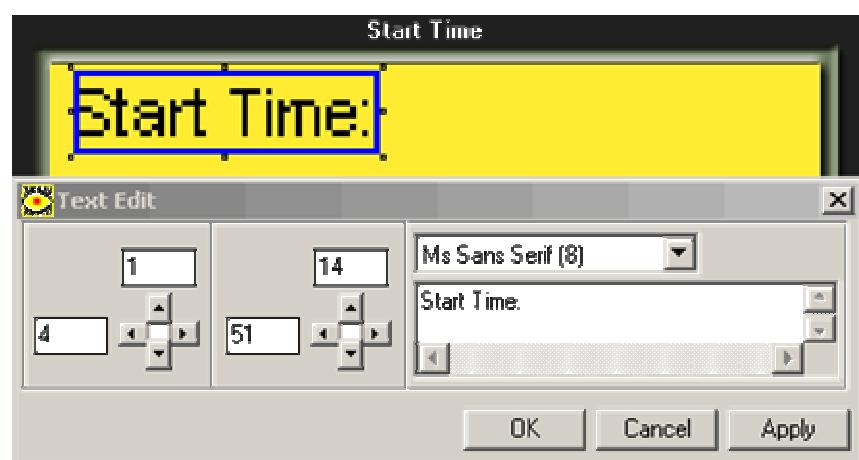
Click the buttons at the bottom of the Program Tree to move between the Ladder Editor and HMI Editor.



2. Create and name a Display: Start and End Time.



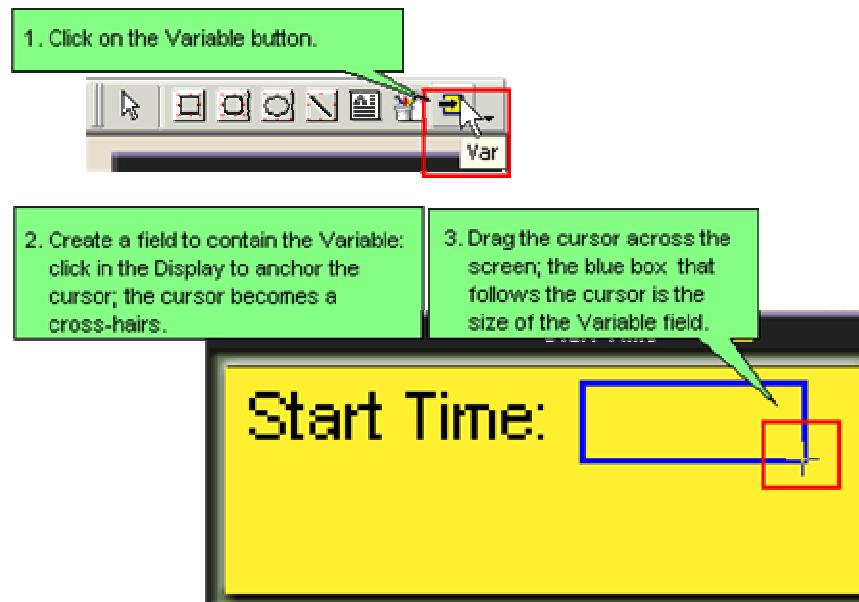
3. Draw a text box, and enter fixed text: Start Time.



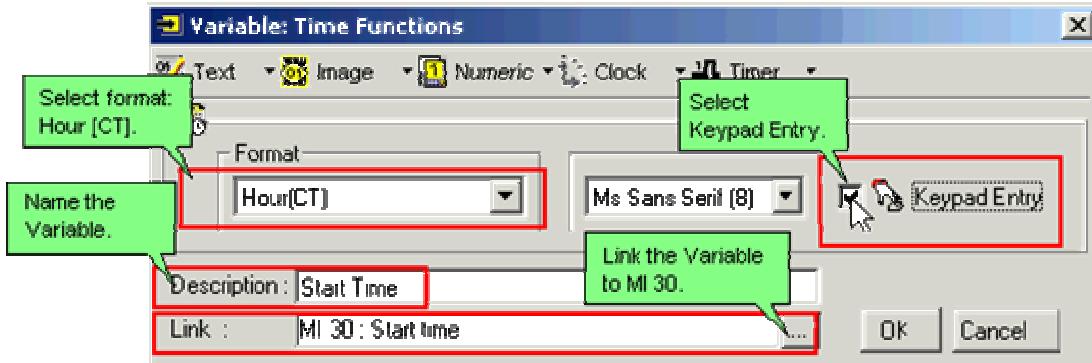
4. Draw another text box, and enter the text: End Time.



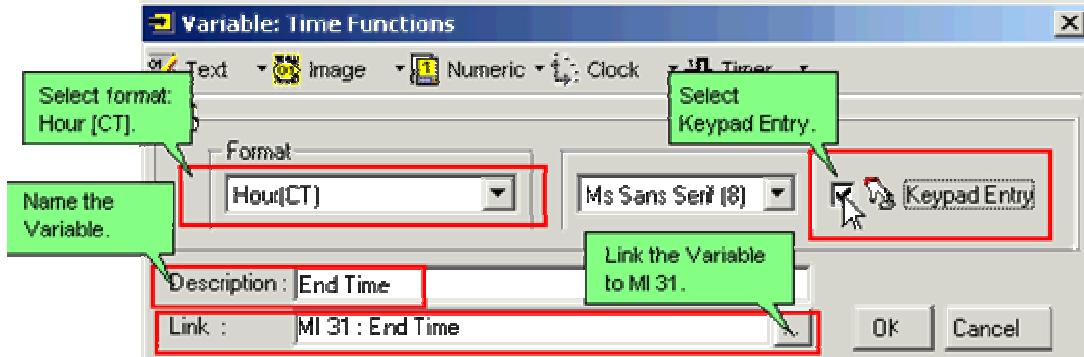
5. Create a field to hold the first Time Function Variable, Start Time.



6. Define the Variable as Keypad Entry and link it as shown below.



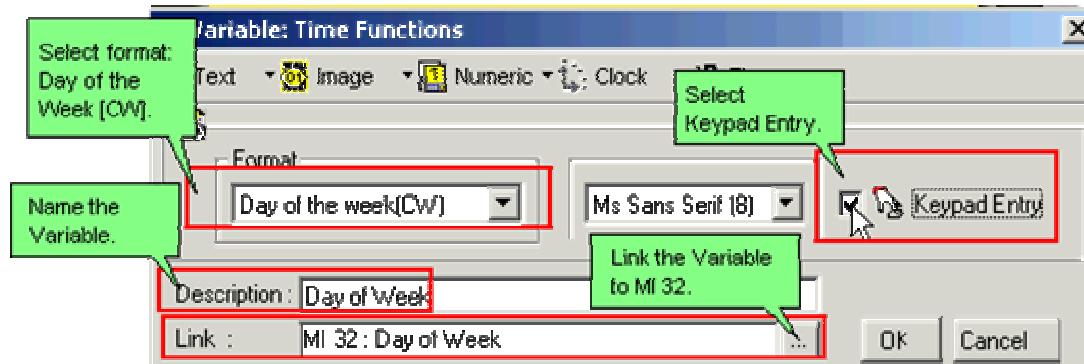
6. Create a field and define the End Time Variable, linking it to MI 31.



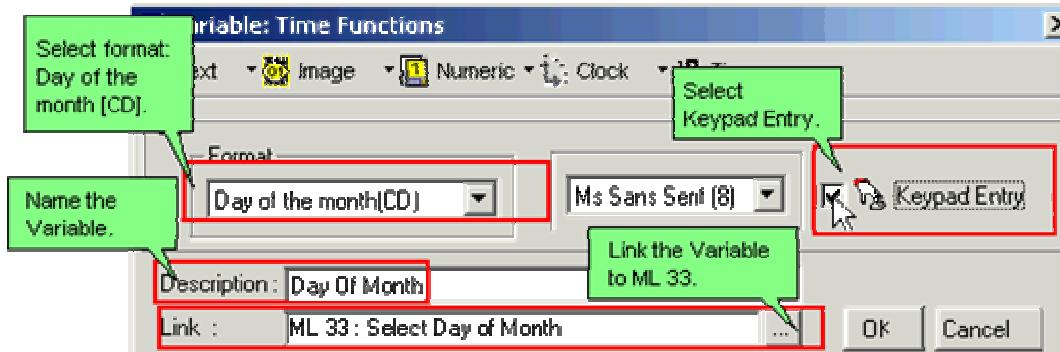
This Display is complete.

Day of Week & Day of Month Variables

1. Create and name a new display; Select Day and Date.
2. Draw a text box, entering the text Select Day.
3. Draw another text box, entering the text Select Date.
4. Create a field to hold the Select Day Variable.
5. Define this variable as Day of Week, and link it to MI 32.



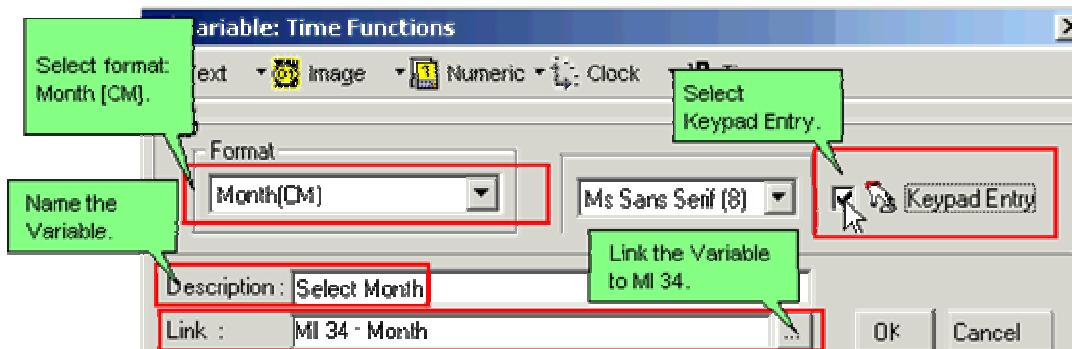
6. Create a field to hold the Select Date Variable.
7. Define this variable as Day of Month, and link it to ML 33.



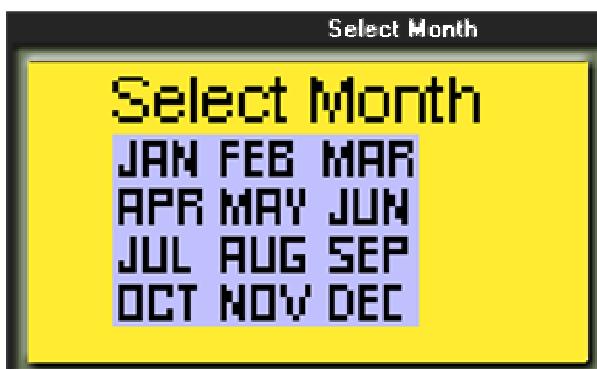
This Display is complete.

Month Variable

1. Create and name a new display; Select Month.
2. Draw a text box, entering the text Select Month.
3. Create a field to hold the Select Month Variable.
4. Define this variable as Month, and link it to MI 34.



This Display is complete.



You must create variables that enable times and dates to be set from the controller keyboard.

Setting Jumps

1. Open Display Start and End Time.
2. Click on the first Jump Condition, and select SB 30: HMI keypad entries completed.
3. Click on Display, and select Display 2.
4. Open Display Select Day and Date, click on the first Jump Condition, and select SB 30..
5. Click on Display, and select Display 3, Select Month.

Set the Jump from Display 3 according to your requirements.

Jumps move from Display to Display, enabling the user to enter the required data.

To see how register values relate to individual functions, refer to the individual topics listed below.

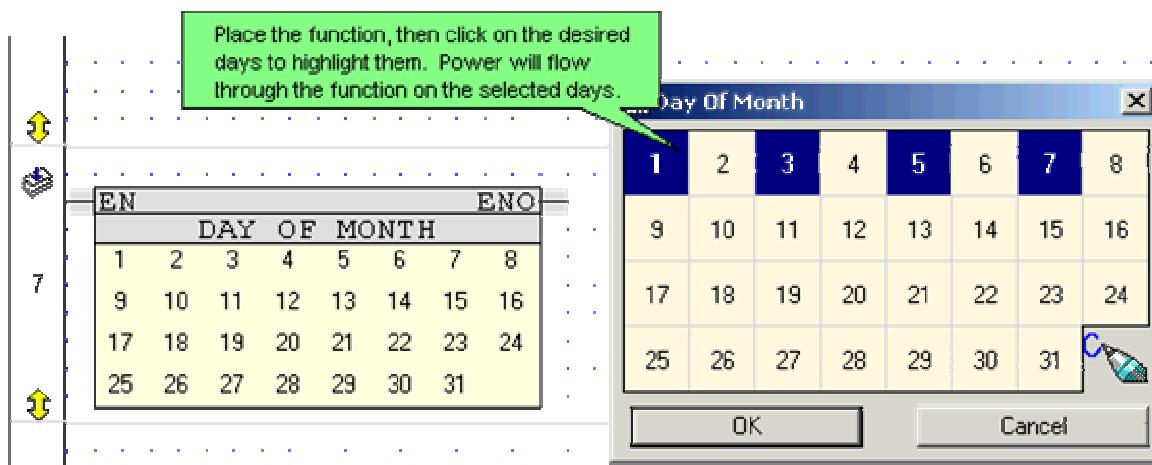
Day of the Month-Direct and Indirect

Day of the Week-Direct and Indirect

Clock: Day of Month-Direct/Indirect

The Day of the Month function enables you to assign tasks or run programs on specific days, such as the 14th and 21st of a month, according to the RTC calendar embedded in the controller.

Direct Day of the Month:



According to the above example:

- On the 1st, 3rd, 5th and 7th the function block's output will be logic "1" (ON).
- On the other days of the month the function block's output will be logic "0" (OFF).

Indirect Day of the Month

Indirect Clock functions are linked to registers. Values may be placed into the linked register by your application, or may be entered via the controller keypad.

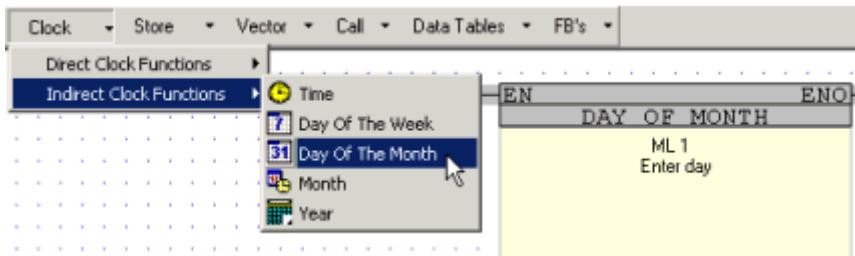
The Indirect Month Time function is linked to a 32-bit ML or SL that provides a bitmap for the functions. The ML value shown below contains the decimal value 271077376 (hexadecimal 10285000). According to this value:

- On the 12th, 14th, 19th, 21st and 28th of the month the FB's output will be logic "1" (ON).
- On the other days of the month the FB's output will be logic "0" (OFF).

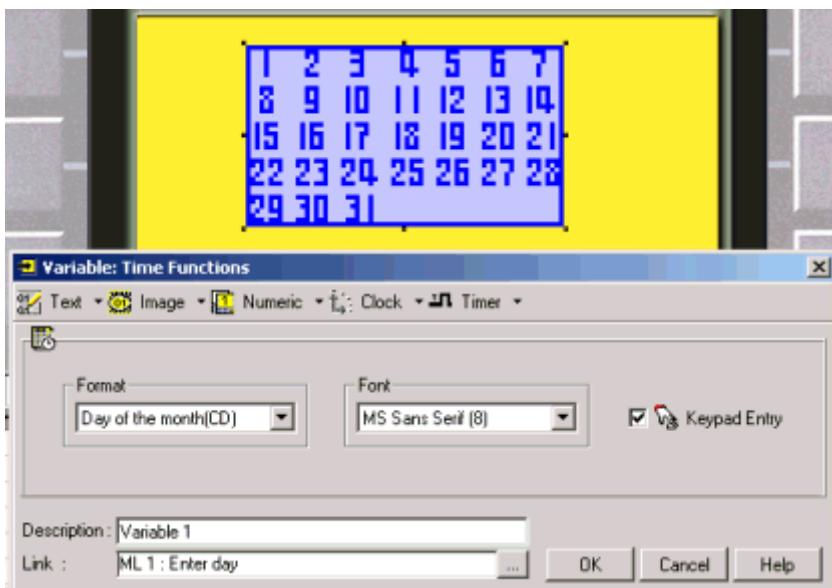
ML	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Bit Status	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Setting Day of Month via Controller Keypad

- Place an Indirect Day of Month clock function in the Ladder.



- Create HMI Displays that include keyboard-entry variables. This type of variable accepts a number entered via the controller keyboard, and stores the number in a linked operand, ML or SL.

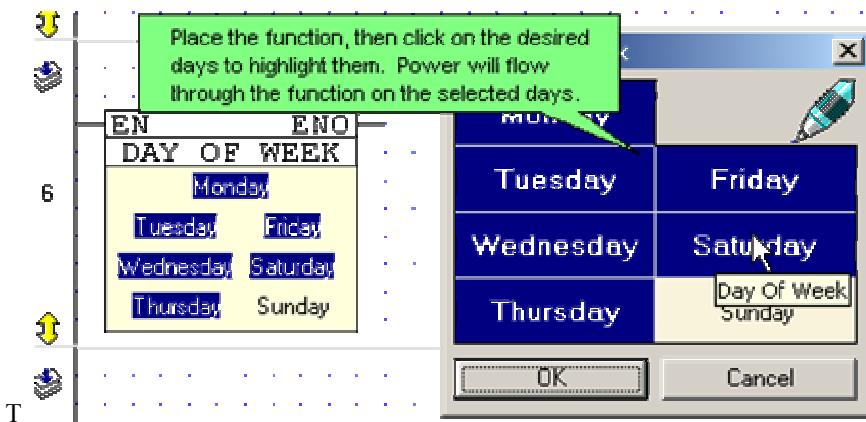


To select the days using the controller's keyboard, the operator uses:

- Up and Down scroll arrow keys to scroll through the days of the month.
- The <Enter>key to select the desired days of the month.

Clock: Day of Week-Direct/Indirect

The Day of the Week function block enables you to assign tasks or run programs on specific days, such as Monday or Tuesday, according to the RTC calendar embedded in the controller..

Direct Day of the Week:

According to the above example:

- On Monday, Tuesday, Wednesday, Thursday, and Friday the function block's output will be logic "1" (ON).
- On Saturday and Sunday the function block's output will be logic "0" (OFF).

Indirect Day of the Week

Indirect Clock functions are linked to registers. Values may be placed into the linked register by your application, or may be entered via the controller keypad.

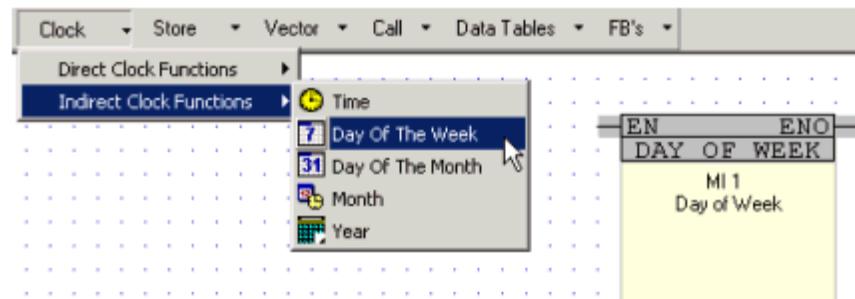
The Indirect Day of Week function is linked to a 16-bit register that provides a 7-bit bitmap in the linked MI. The MI value shown below contains the decimal value 42 (hexadecimal 2A). According to this value:

- On Monday, Wednesday and Friday the function block will go to logic "1" (ON).
- On Sunday, Tuesday, Thursday and Saturday the function block will go to logic "0" (OFF).

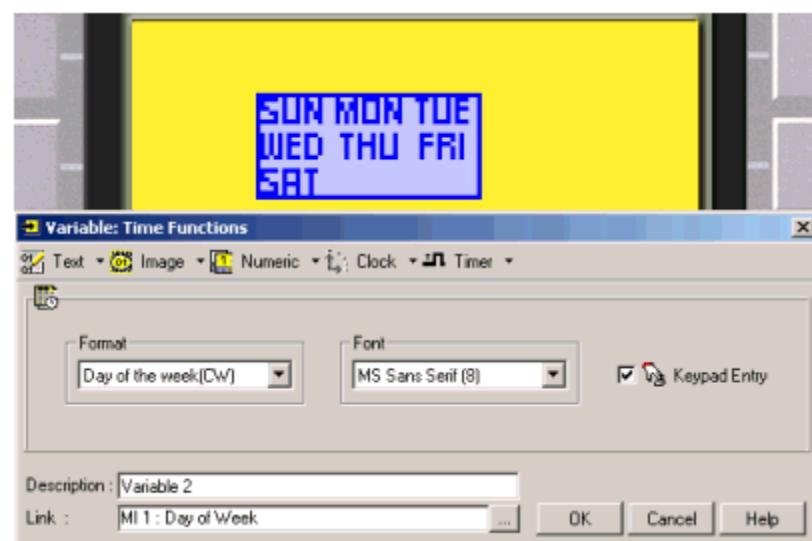
Day	Sat	Fri	Thurs	Wed	Tues	Mon	Sun
MI	6	5	4	3	2	1	0
Bit Status	0	1	0	1	0	1	0

Setting Day of Week via Controller Keypad

- Place an Indirect Day of Week function in the Ladder.



- Create HMI Displays that include keyboard-entry variables. This type of variable accepts a number entered via the controller keyboard, and stores the number in a linked MI, SI, ML or SL.



To select the days using the controller's keyboard, the operator uses:

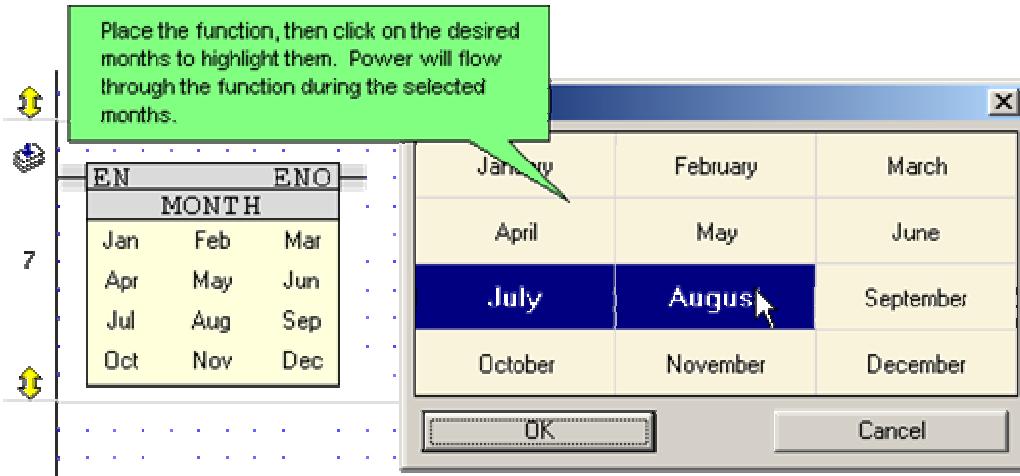
- Up and Down scroll arrow keys to scroll through the days of the week,
- The <Enter>key to select the desired days of the week.

Clock: Month-Direct/Indirect

The Month function block is used for monthly time functions.

Direct Month Function:

The Direct Month function block contains the twelve months of the year.



According to the above example, power will flow through the function during the months of July and August.

Indirect Month Function

Indirect Clock functions are linked to registers. Values may be placed into the linked register by your application, or may be entered via the controller keypad.

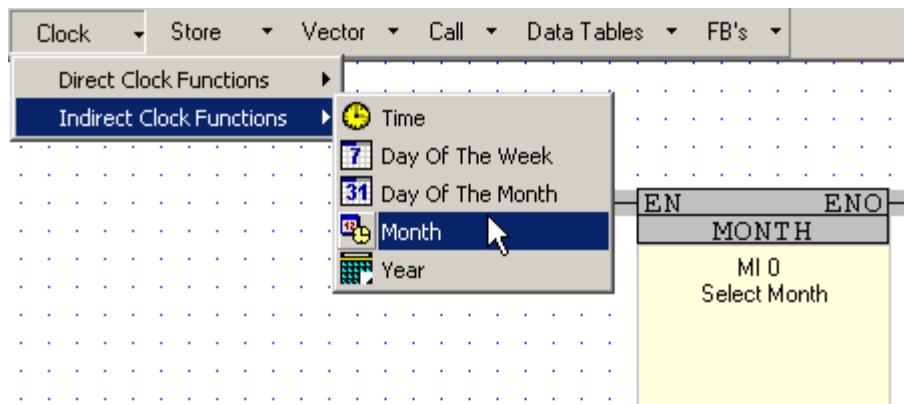
The Indirect Day of Week function is linked to a 16-bit register that provides a 7-bit bitmap in the linked MI. The MI value shown below contains the decimal value 42 (hexadecimal 2A). According to this value:

- On Monday, Wednesday and Friday the function block will go to logic "1" (ON).
- On Sunday, Tuesday, Thursday and Saturday the function block will go to logic "0" (OFF).

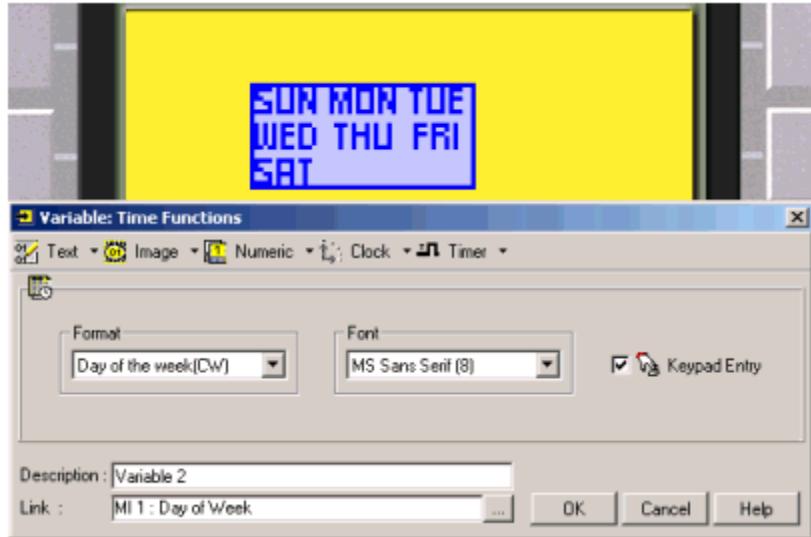
Day	Sat	Fri	Thurs	Wed	Tues	Mon	Sun
MI	6	5	4	3	2	1	0
Bit Status	0	1	0	1	0	1	0

Setting Month via Controller Keypad

- Place an Indirect Month function in the Ladder.



- Create HMI Displays that include keyboard-entry variables.
This type of variable accepts a number entered via the controller keyboard, and stores the number in a linked MI, SI, ML or SL.



- Up and Down scroll arrow keys for scrolling through the months
- +/- keys for selecting the desired months
- enter key for confirming selection

The Indirect Month function values are entered into a 12-bit bitmap in the linked MI. The MI value shown below contains the decimal value 3591 (hexadecimal E07). According to these values:

- During the months of January, February, March, October, November, and December the function block will go to logic "1" (ON).
- During the months of April, May, June, July, August, and September the function block will go to logic "0" (OFF).

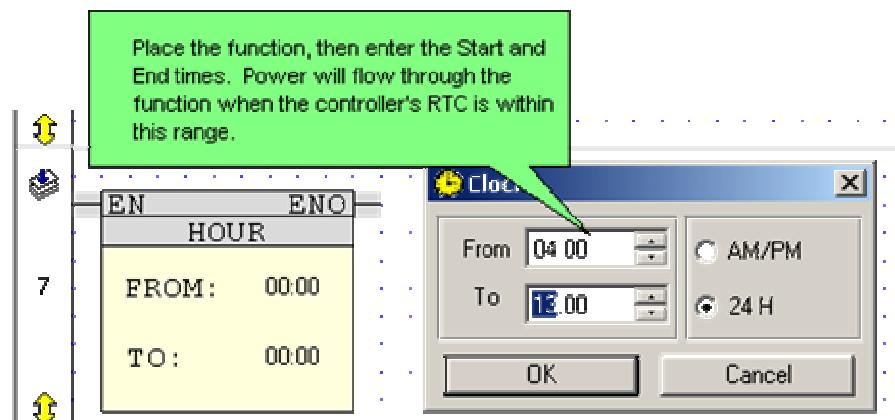
Month	Dec	Nov	Oct	Sep	Aug	Jul	Jun	May	Apr	Mar	Feb	Jan
MI	11	10	9	8	7	6	5	4	3	2	1	0
Bit Status	1	1	1	0	0	0	0	0	0	1	1	1

Clock: Time, Direct/Indirect

The Time function block is used for 24 hour time functions.

Direct Time Function:

The Direct Time function block has a 'from' (start) and a 'to' (end) time set by the programmer.



According to the above example:

Power will flow through the function between 4 A.M. and 1 P.M..

Indirect Time Function

Indirect Clock functions are linked to registers. Values may be placed into the linked register by your application, or may be entered via the controller keypad.

The Indirect Time function is linked to two consecutive registers.

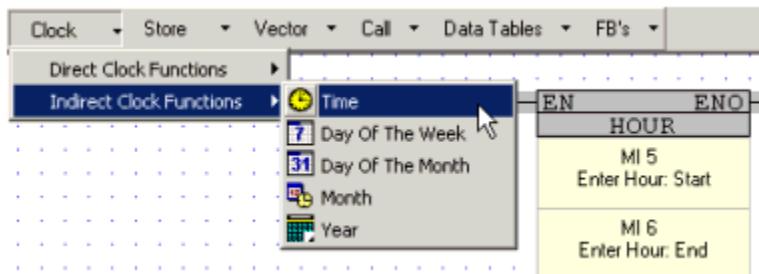
The values are read as hexadecimal (BCD). According to the figure shown below:

- Between the hours of 7:30 and 11:59 P.M., the FB's output will be logic "1" (ON).
- At all other times, the FB's output will be logic "0" (OFF).

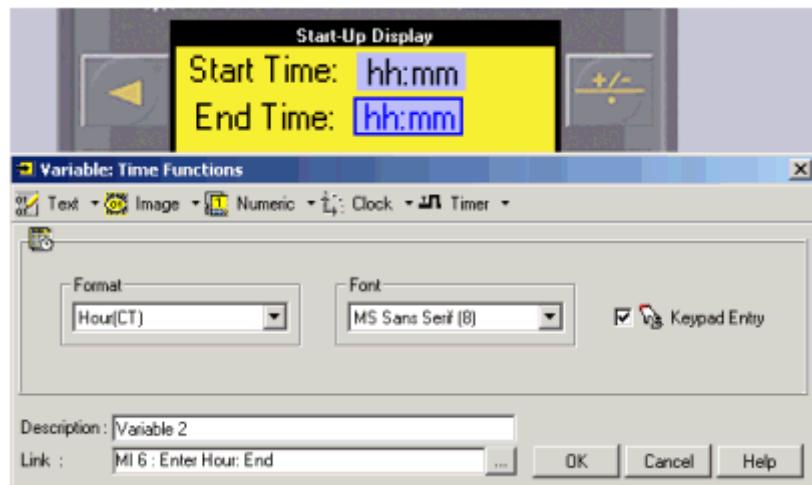
MI 5: Start Time				MI 6: End Time					
Hex	H	H	M	Hex	H	H	M		
1	9	3	0	2	3	5	9		
BCD	0001	1001	0011	0000	BCD	0010	0011	0101	1001

Setting Time (Hour) via Controller Keypad

- Place an Indirect Time clock function in the Ladder.



- Create HMI Displays that include keyboard-entry variables. This type of variable accepts a number entered via the controller keyboard, and stores the number in a linked register.



To select the days using the controller's keyboard, the operator uses:

- The number keys.
- The <Enter> key to confirm the entry.

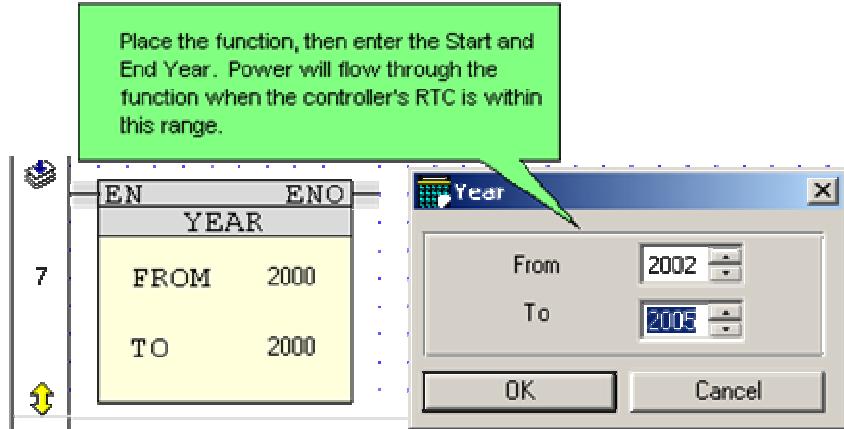
Clock: Year, Direct/Indirect

The Year function block is used for yearly time functions.

Direct Year Function:

The Direct Year function block has a 'from' (start) and a 'to' (end) year set by the programmer.

If the RTC is within this range, power will flow through the function block.



According to the above example:

- Between the years 2002 - 2005, power will flow through the function.

Indirect Year Function:

The Indirect Year function block is linked to two consecutive integers. These integer values are entered by the user via the controller keypad.

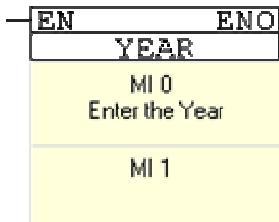
If the RTC is within these two times: power will flow through the function.

If the RTC is not currently within these two times: power will not flow through the function.

You must create a Time Function Variable in Year (CY) format for the user to enter the start and end years.

To select the year using the controller's keyboard, the operator uses:

- Up and Down scroll arrow keys to scroll through the years
- Enter key to select the desired year



Vector Functions

Vector Operations

Vector operations enable you to select an operand type, define a vector within that type, and to perform different actions within the defined vector.

- Bit to Numeric, Numeric to Bit
- Compare
- Copy
- Copy Memory
- Transpose
- Shift Byte Left
- Fill
- Find
- Get Max
- Get Min
- Load
- Load Timer Bit Value
- Store

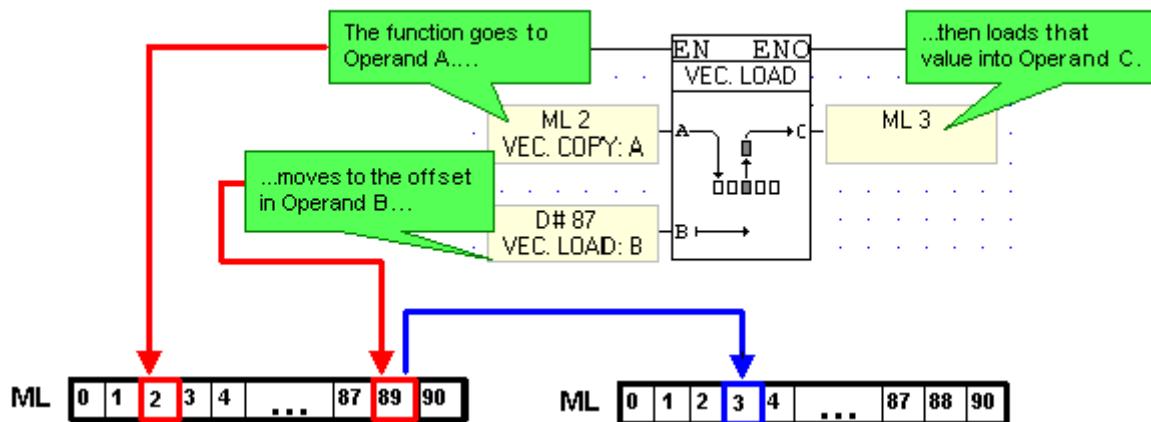
Vector: Load

Load allows you to take a value contained in a source operand and load it into a target operand. This value may be either the status of a bit operand or a register value.

1. Click the Vector menu on the Ladder Toolbar, then select Load.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses. Operands A and B determine the location of the source value. Operand A determines the starting point for the function. Operand B contains the offset value, and the operand linked to Operand C is the target operand.

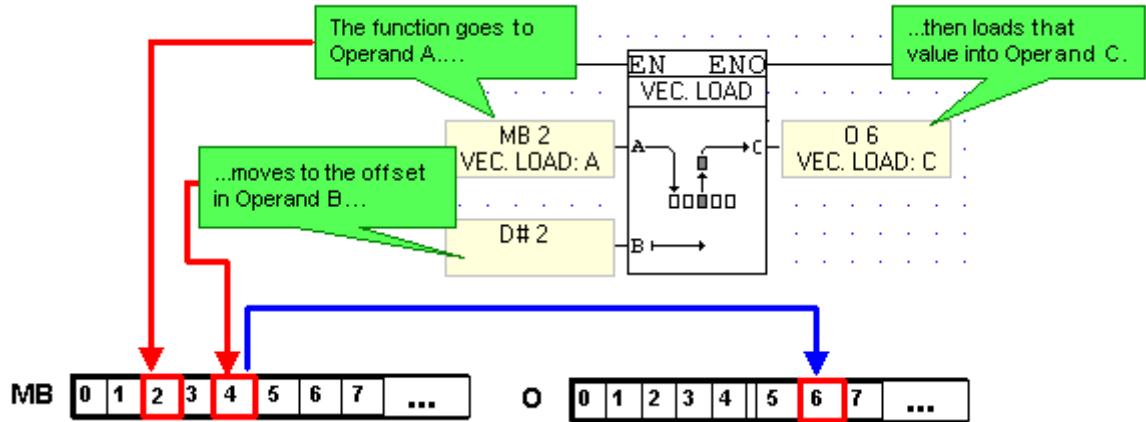
Example: Registers

Below, the value in ML 89 is loaded into ML 3. If the value in ML 89 is 986, 986 will be stored in ML 3. The previous value in ML 3 is overwritten. The current value in ML 2 remains unchanged.



Example: Bit Operands

Below, the status of MB 4 is loaded into O 6. If MB 4 is ON, O 6 will be turned ON. The status of O 6 is overwritten. The status of MB 4 remains unchanged.



Note that:

- If you link a bit operand to Operand A, the function will only allow you to link a bit operand to Operand C.
- If you link a register to Operand A, the function will only allow you to link a register to Operand C.
- If a double register (ML, SL, DW, SDW) is used as the source operand, and a single register (MI), is used as the target, only the first 16 bits will be loaded from the source into the target operand.

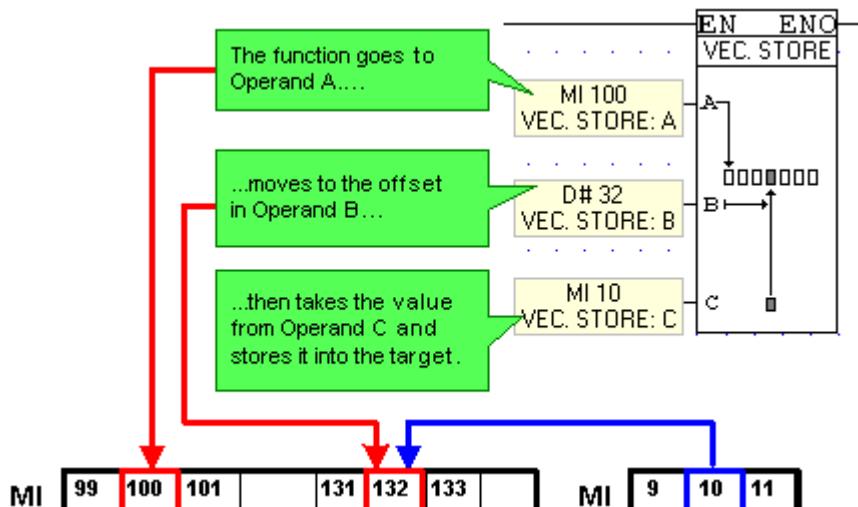
Vector: Store

Store allows you to take a value contained in a source operand and load it into a target operand. This value may be either the status of a bit operand or a register value.

1. Click the Vector menu on the Ladder Toolbar, then select Store.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses. Operands A and B determine the location of the target operand. Operand A determines the starting point for the function. Operand B contains the offset value, and the operand linked to Operand C is the source operand.

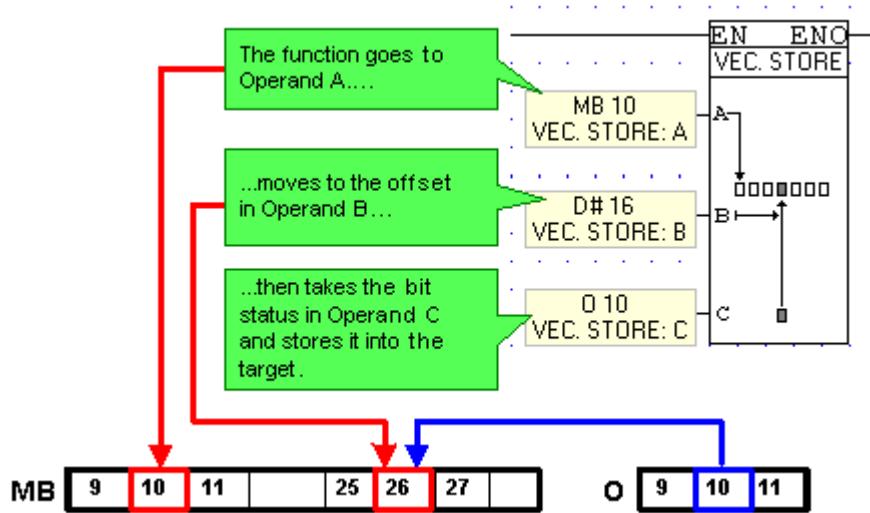
Example: Registers

Below, the value in MI 10 is loaded into MI 132. If the value in MI 10 is 64, 64 will be stored in MI 132. The previous value in MI 132 is overwritten. The current value in MI 10 remains unchanged.



Example: Bit Operands

Below, the status of O 10 is stored into MB 26. If O 10 is ON, MB 10 will be turned ON. The status of MB 10 is overwritten. The status of O 10 remains unchanged.



Note that:

- If you link a bit operand to Operand A, the function will only allow you to link a bit operand to Operand C.
- If you link a register to Operand A, the function will only allow you to link a register to Operand C.
- If a double register (ML, SL, DW, SDW) is used as the source operand, and a single register (MI), is used as the target, only the first 16 bits will be loaded from the source into the target operand.

Vector: Find

The Find function:

- searches through a vector,
 - locates either an integer value or the first bit of a desired status within that vector,
 - records the location of the operand containing the desired value.
1. Click the Vector menu on the Ladder Toolbar, then select Find.
 2. Place the function in the desired net.
 3. Link the desired Operands and Addresses.

Operand A, Locate Value in Vector, determines the value or bit status to be found.

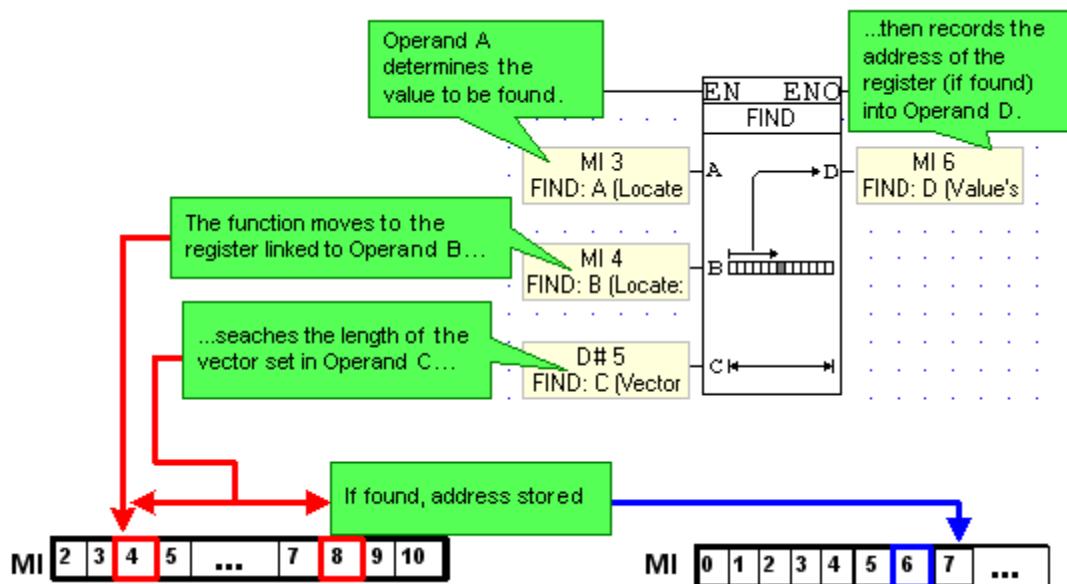
Operand B, Locate Start Address, determines from where the function begins to search. If you select MB 3, for example, the function will search through the MB vector, and will begin to search at MB #3.

Operand C, Vector Length, determines the length of the vector to be searched.

Operand D, Value's Location, is where the function records the location of the operand--if the function finds the value. If the function does not find the value, a linked MI will contain the value -1; a long register will contain FFFFFFFF.

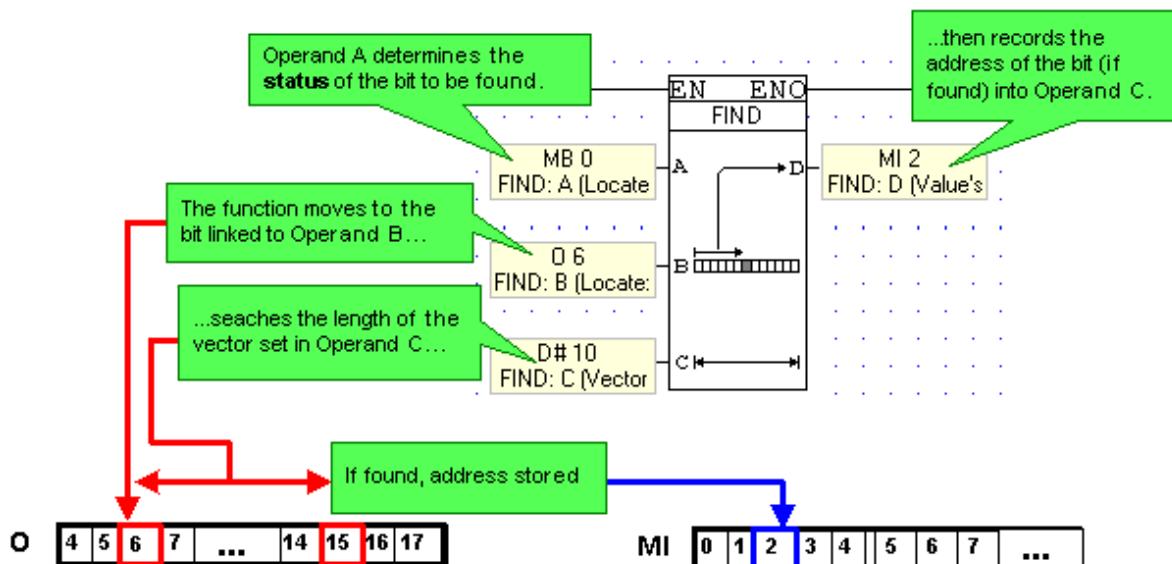
Example: Find Register Value

Below, if MI 3 contains the value 16, the function searches for 16 from MI 4 to MI 8. MI 3. If the value 16 is found in the vector, the address of the operand containing 16 is recorded in MI 6. If the value is not found, MI 6 will contain -1.



Example: Find Bit Status

Below, if MB 0 is OFF, the function searches from O 6 to O 15. If a bit having OFF status is found, the location of the bit is recorded in MI 2.



Note that:

- When the function finds the value, it stops running. This means that if the value is contained by more than one operand in the vector, only the location of the first operand containing that value is recorded.
 - If the value is not found, the function stops until it is reactivated.

Vector: Fill

Fill enables you to:

- select an register, bit operand, or constant value,
 - define a vector of operands,
 - write the selected value into every operand within the vector.

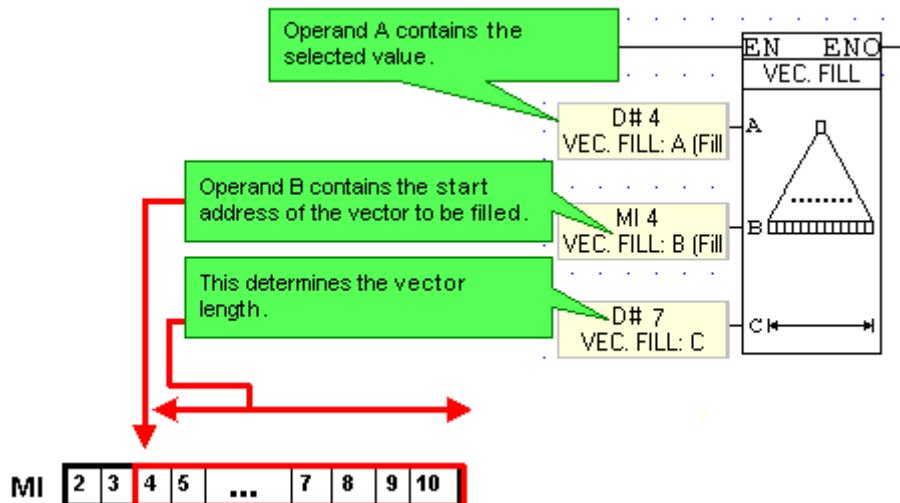
The function is located on the Vector menu.

Vector Fill

1. Click the Vector menu on the Ladder Toolbar, then select Fill.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.
 Operand A: this is the source value.
 Operand B: this is the address of the first operand in the vector.
 Operand C: this is the vector length.

Example:

Below, the constant value 4 is written into MI 4 through 10.

Vector: Fill (Offset)

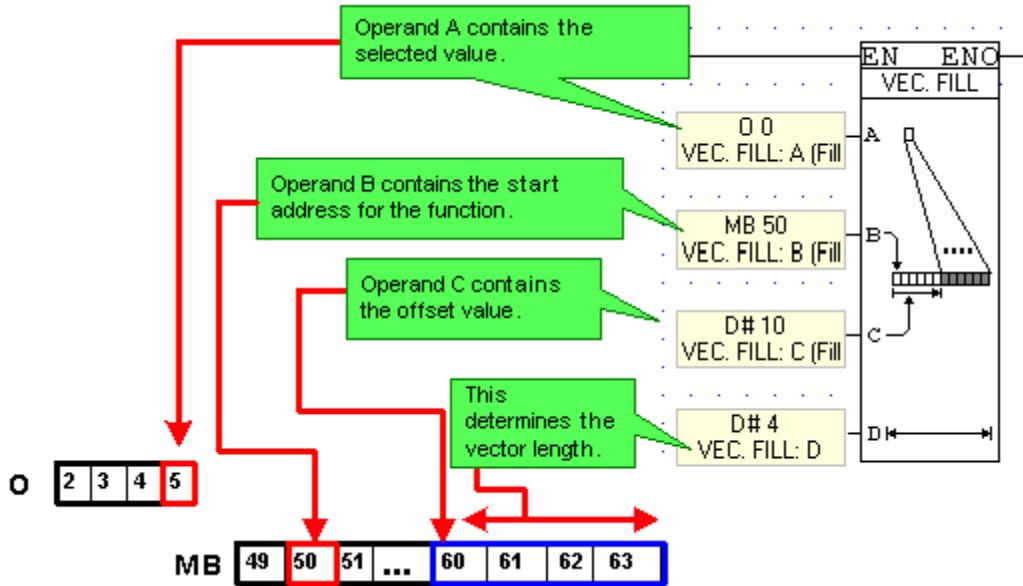
Fill (Offset) enables you to:

- Select an register, bit operand, or constant value,
- define a vector of operands that is offset from a selected start address,
- write the selected value into every operand within the vector.

1. Click the Vector menu on the Ladder Toolbar, click Use Offset, then select Fill.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.
 Operand A: this is the source value.
 Operand B: this is the start address.
 Operand C: this is the offset from the start address.
 Operand D: this is the vector length.

Example:

Below, the status of O 5 is written into MB 60 through 63.



Vector: Copy

Copy enables you to:

- define a vector of operands,
- copy the values or bit status of each operand within that vector,
- write those values or status into a corresponding vector of operands of the same length.

The function is located on the Vector menu.

Copy

1. Click the Vector menu on the Ladder Toolbar, then select Copy.

2. Place the function in the desired net.

3. Link the desired Operands and Addresses.

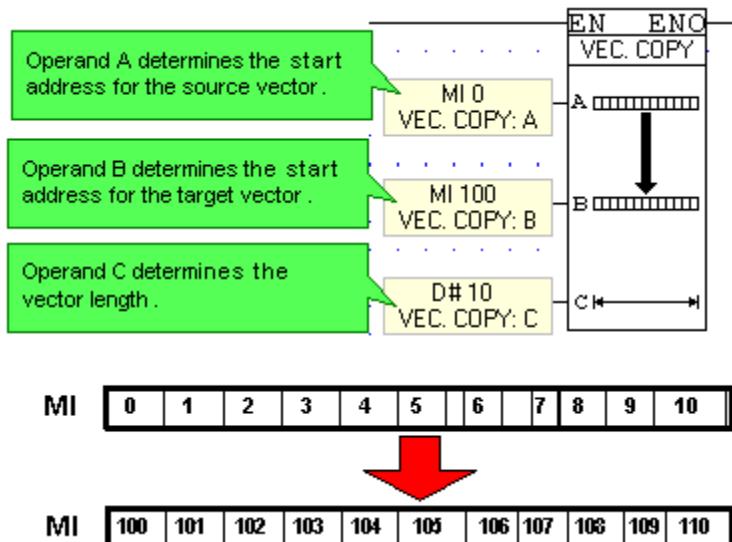
Operand A: this is the range of operands from which the values will be copied.

Operand B: this is the first operand in the vector, the range of operands to which the values will be copied.

Operand C: this sets the length, meaning the number, of operands for both ranges.

Example:

Below, the values in MI 0 through 10 will be copied to MI 100 to 101.



Copy (Offset)

Copy (Offset) enables you to:

- Define a source vector of operands that is offset from a selected start address,
- copy the values or bit status of each operand within that range,
- define a target vector of operands that is offset from a selected start address,
- write the source values or status into the target vector.

1. Click the Vector menu on the Ladder Toolbar, click Use Offset, then select Copy.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.

Operand A: this is the start address for the source vector.

Operand B: this is the offset from the start address.

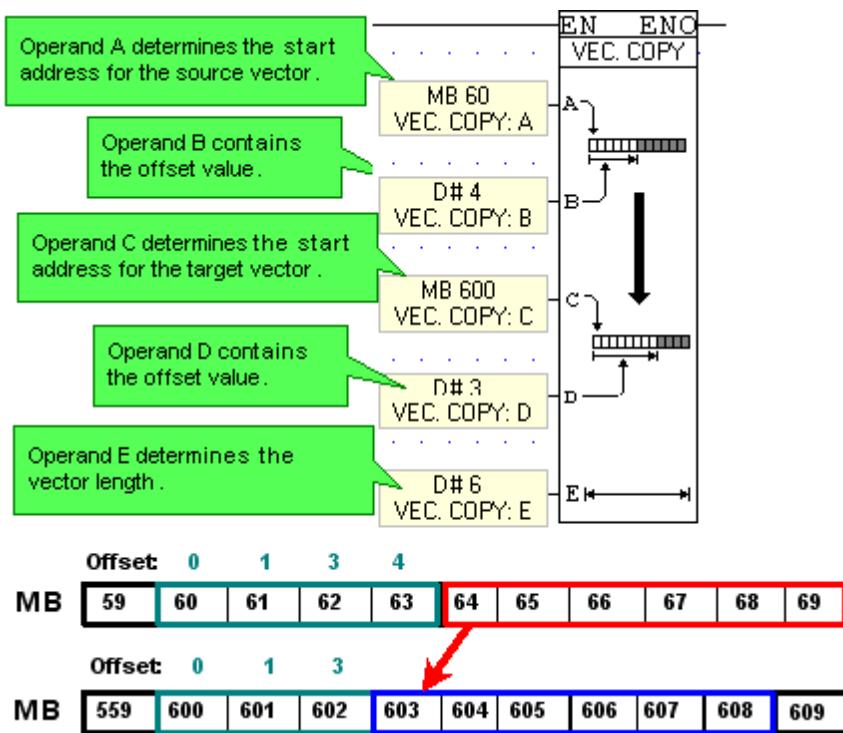
Operand C: this is the start address for the target vector.

Operand D: this is the offset from the start address.

Operand E: this is the vector length.

Example:

Below, the status of MB 64 through MB 69 will be copied to MB 603 through 608.



Strings: Transpose

Transpose enables you to 'compress' MI values into bytes, or 'expand' bytes into MIs:

- Define a source vector of registers that is offset from a selected start address.
- Copy the low byte of each register within that range,
- Define a target vector of operands that is offset from a selected start address.
- Select Conversion type:

MI to Byte (Compress) to write the low byte of each source register into the consecutive bytes of the target vector; thus the low bytes of 3 source registers will occupy 2 MIs.

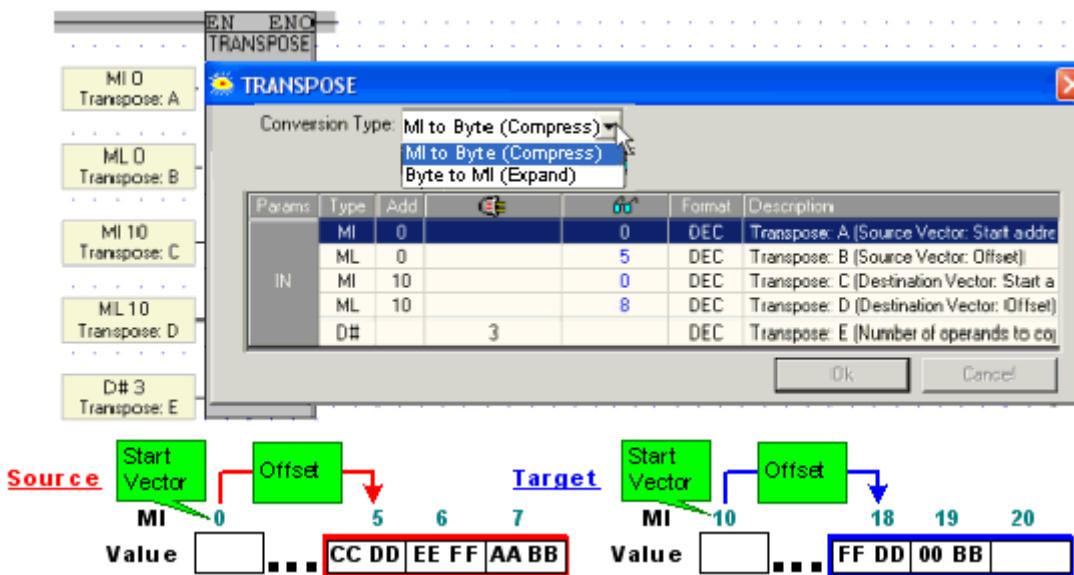
Byte to MI (Expand) to write the consecutive bytes of the source vector into the low byte of each target register, thus the bytes of 3 MIs will occupy the low bytes of 6 MIs.

To use Transpose:

1. Click Strings on the Ladder Toolbar, then select Transpose.
 2. Place the function in the desired net.
 3. Select the type of function.
 4. Link the desired Operands and Addresses.
- Operand A: start address for the source vector.
 Operand B: offset from the start address.
 Operand C: start address for the target vector.
 Operand D: offset from the start address.
 Operand E: vector length.

Example:

Below, the low bytes of MI 5, 6, and 7 are copied into the consecutive bytes of MI 18 and 19.



Vector: Compare

Compare enables you to:

- Define 2 vectors of operands,
- compare the values or bit status of each corresponding operand within that range,
- record the location of the first set of unmatched values found.

The function is located on the Vector menu.

Compare

1. Click the Vector menu on the Ladder Toolbar, then select Compare.

2. Place the function in the desired net.

3. Link the desired Operands and Addresses.

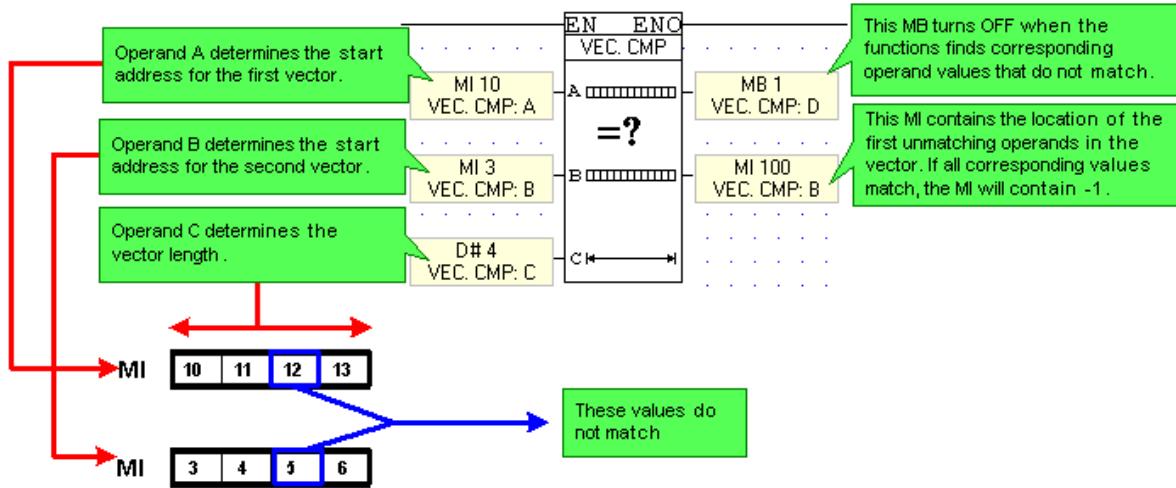
Operand A: this is the start address for the first vector of operands.
 Operand B: this is the start address for the second vector of operands.
 Operand C: this sets the length of both vectors.

Operand D: this MB turns ON when the corresponding values in both vectors match, and turns OFF when corresponding operand values do not match.

Operand E: this MI contains the location of the first set of unmatched operands in the vector. If all of the corresponding values match, the MI contains -1.

Example:

Below, the values in MI 10 through 13 will be compared to MI 3 through 6. MI 12 and MI 5 occupy corresponding locations in the their respective vectors. When the function finds that the values in MI 12 and MI 5 do not match, the function turns MB 1 turns OFF and stores the location of the operands into MI 100.



Compare (Offset)

Compare (Offset) enables you to:

- Define a source vector of operands that is offset from a selected start address,
- define a target vector of operands that is offset from a selected start address,
- compare the values or bit status of each corresponding operand within that range,
- record the location of the first set of unmatched values found.

1. Click the Vector menu on the Ladder Toolbar, click Use Offset, then select Compare.
2. Place the function in the desired net.

3. Link the desired Operands and Addresses.

Operand A: this is the start address for the first vector.

Operand B: this is the offset from the start address.

Operand C: this is the start address for the second vector.

Operand D: this is the offset from the start address.

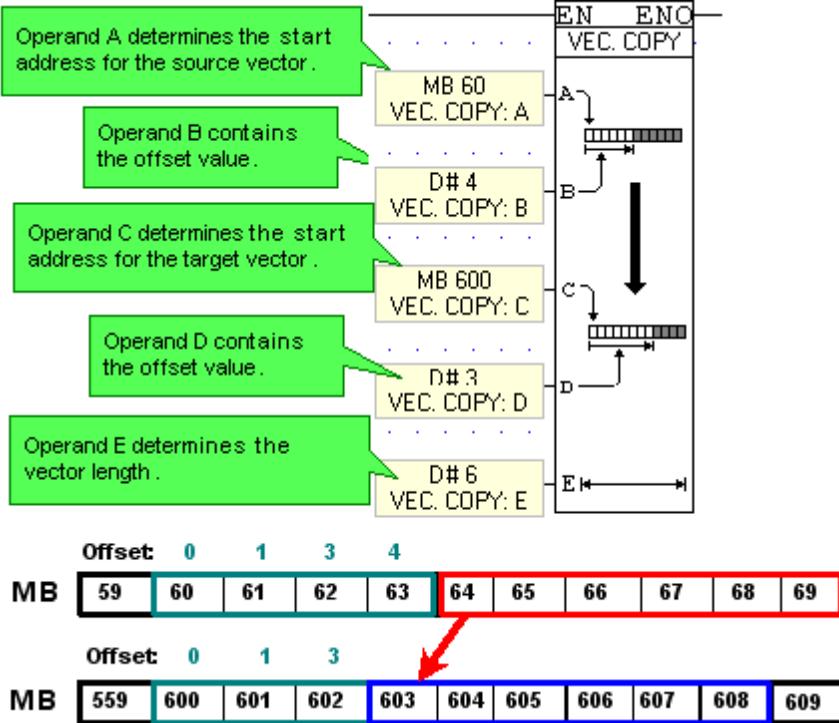
Operand E: this is the vector length.

Operand F: this MB turns ON when the corresponding values in both vectors match, and turns OFF when corresponding operand values do not match.

Operand G: this MI contains the location of the first set of unmatched operands in the vector. If all of the corresponding values match, the MI contains -1.

Example:

Below, the values in MB 4 through MB 11 will be compared to MB 105 through MB 112. MB 12 and MB 110 occupy corresponding locations in the their respective vectors. When the function finds that the values in MB 12 and MB 110 do not match, the function turns MB 2 OFF and stores the location of the operands into MI 6.



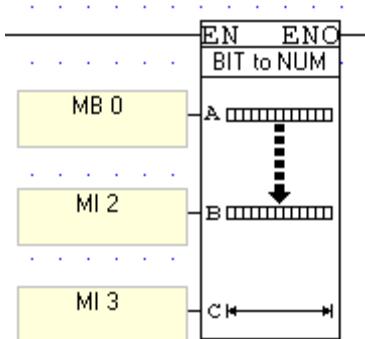
Vector: Bit to Numeric, Numeric to Bit

Use these functions to convert an array of bit values to a numeric value, or a numeric value to an array of bits.

The functions are located on the Vector menu.

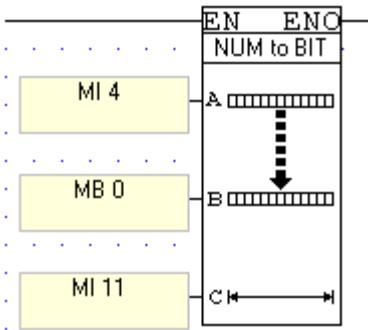
Bit to Numeric

- Operand A: contains the Start Address for the array of bits to be converted.
- Operand B: is the start of the vector that will contain the converted value. Take care in addressing operands, since the converted value may not fit into a single register; the function will overwrite as many consecutive registers as it requires to convert the value.
- Operand C: contains the length of the bit array that will be converted.



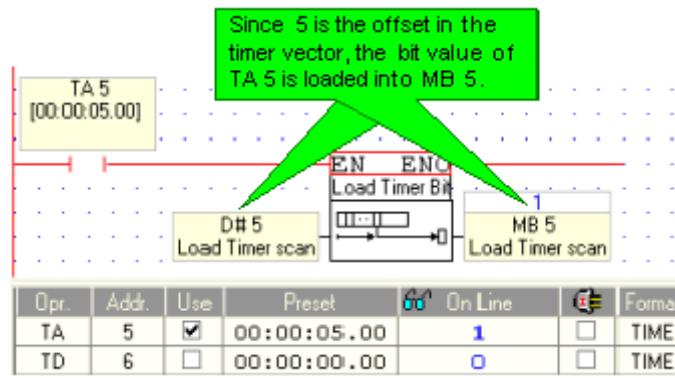
Numeric to Bit

- Operand A: contains the Address of the value to be converted.
- Operand B: contains the Start Address of the bit array that will contain the converted value.
- Operand C: contains the Length of the bit array that will contain the converted value.



Load Timer Bit Value

You can use a Ladder condition to load the current bit value of a Timer into an MB. The input to the Load Timer Scan Bit function is the address of the timer within the Timer vector, and may be a constant or a value provided by a register.

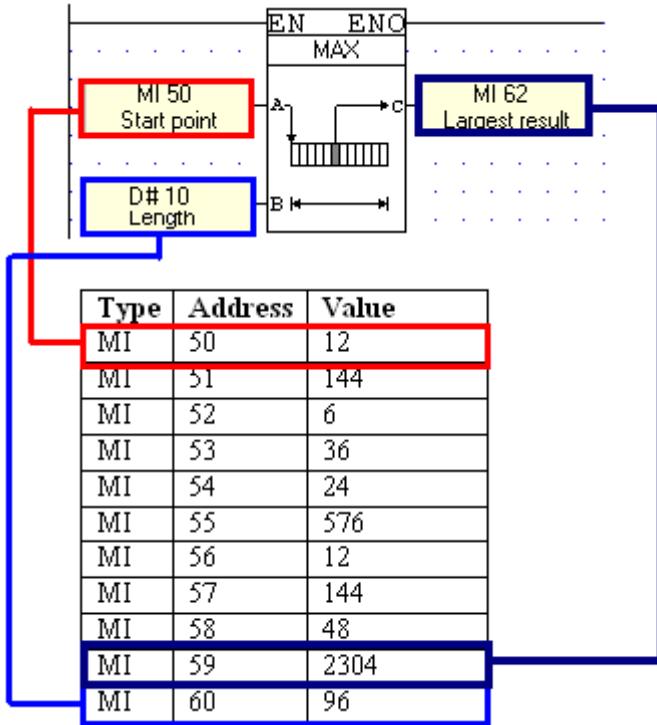


Vector: Get Max

The Get Max function finds the largest value within a range of operands. The function is located on the Vector menu.

Get Max uses 2 input values. The A input sets the beginning of the operand range, the B input sets the end of the range. The result is stored in an output operand, C.

In the example below, the function checks MI 50 through 60. The largest value in the range, 2304, is contained in MI 62; therefore 2304 is stored in MI 59.

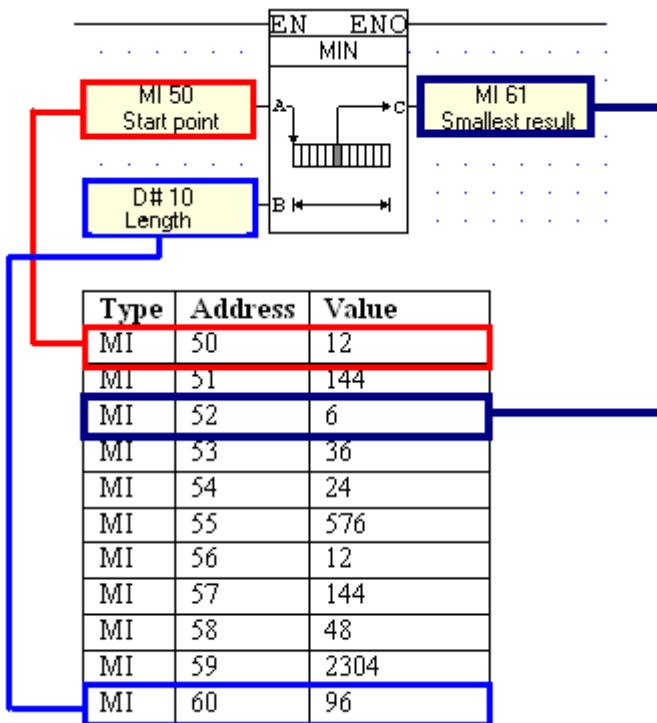


Vector: Get Min

The Get Min function finds the smallest value within a range of operands. The function is located on the Vector menu.

Get Min uses 2 input values. The A input sets the beginning of the operand range, the B input sets the end of the range. The result is stored in an output operand, C.

In the example below, the function checks MI 50 through 60. The smallest value in the range, 6, is contained in MI 52; therefore 6 is stored in MI 61.



Vector: Copy Memory

Copy Memory enables you to:

- define a vector of registers,
- copy the values register within that vector,
- write those values into a corresponding vector of registers of the same length.

To use this function

1. Click the Vector menu on the Ladder Toolbar, click Use Offset, then select Copy Memory.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.

Operand A: this is the start address for the source vector.

Operand B: this is the offset from the start address.

Operand C: this is the start address for the target vector.

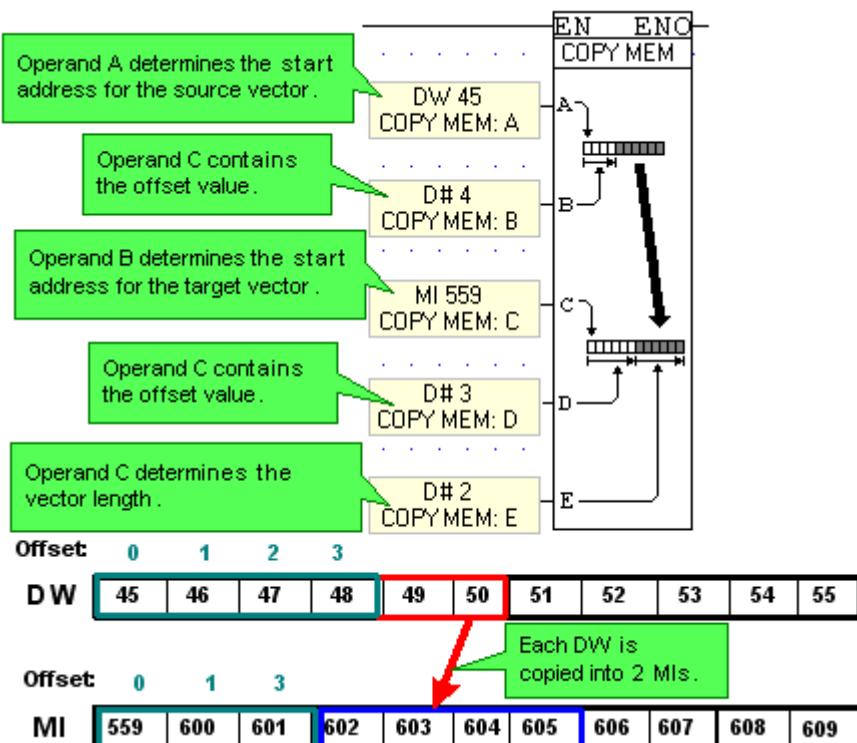
Operand D: this is the offset from the start address.

Operand E: this is the vector length.

Example:

Below, the values within DW 49 and 50 are copied into MIs 602, 603, 604, and 605.

Note • When an MI value is copied into a double register, the MI value will occupy the 2 low bytes of the double register.



Vector: Shift Byte Left

Shift enables you to:

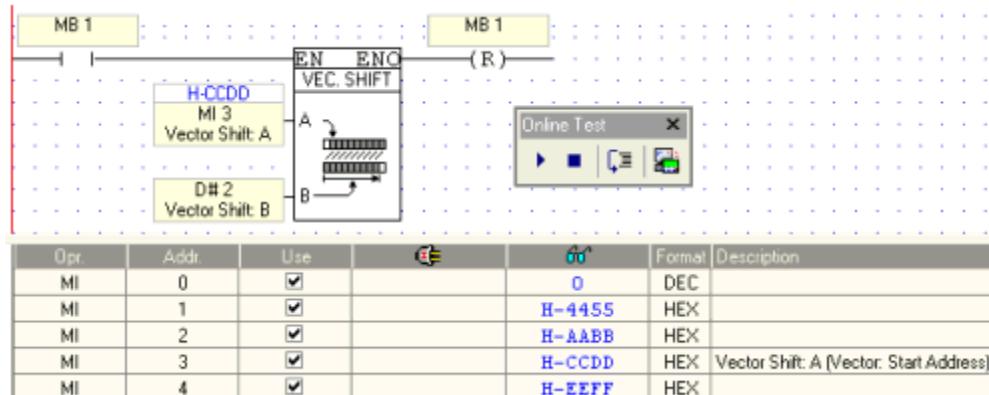
- define a vector of registers,
- shift the bytes within that vector left

To use this function

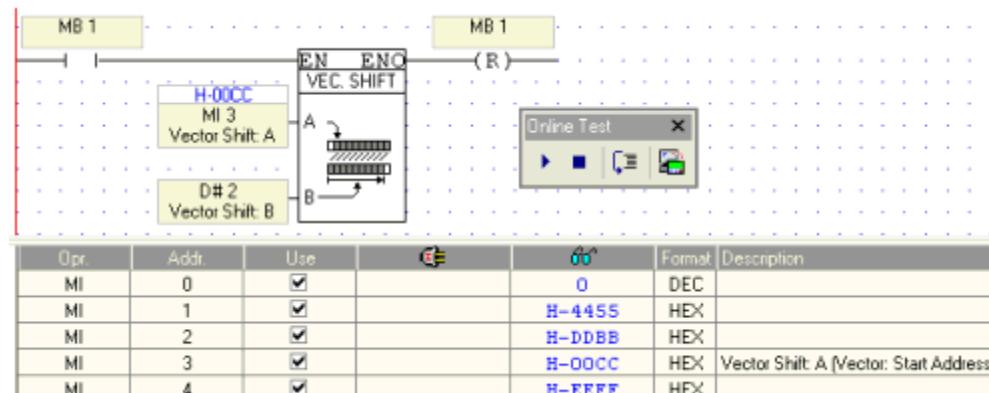
1. Click the Vector menu on the Ladder Toolbar, then select Shift.
 2. Place the function in the desired net.
 3. Link the desired Operands and Addresses.
- Operand A: this is the start address for the source vector.
 Operand B: this is the number of bytes to shift.

Example:

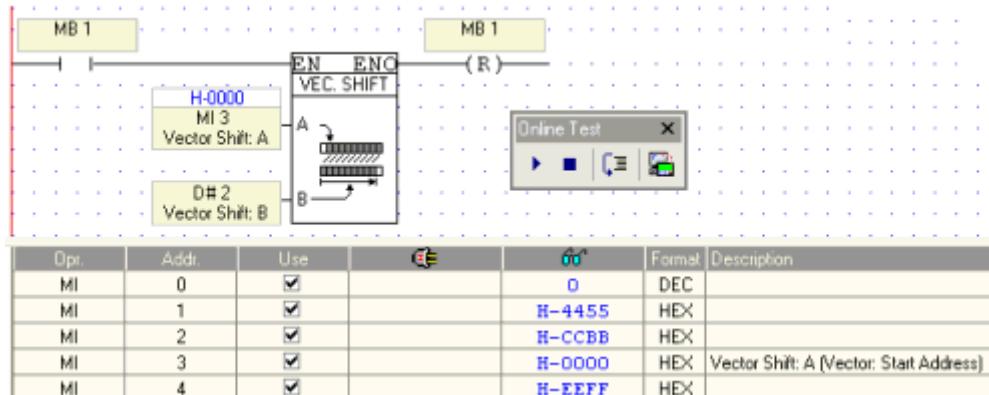
The blue numbers in the figure below show the Online values within the controller. MI 3 is selected for the Shift function.



Below, note the value of MI 3 and MI 4 after the Shift operation.



Repeating Shift leaves MI 3 empty.



Notes •

This function cannot be performed on negative values.

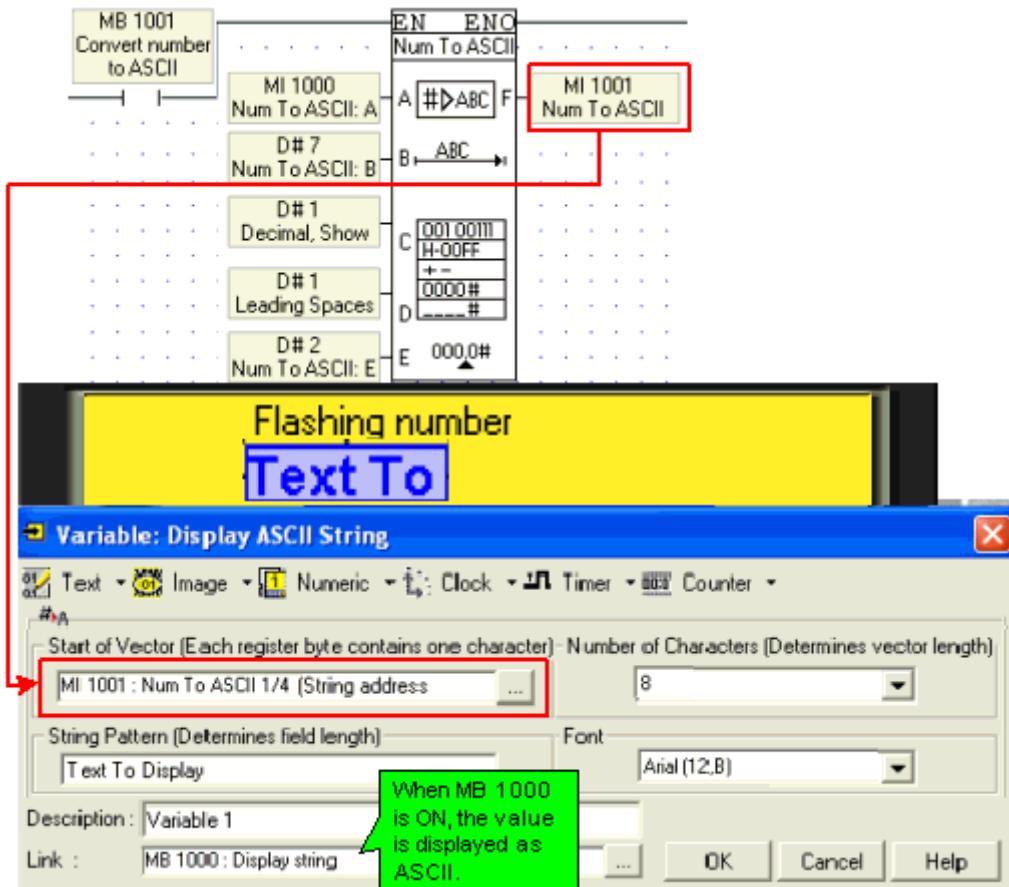
Strings

Time to ASCII

You can display a value as an ASCII string by using the Num to ASCII function together with the Display ASCII String variable.

1. Select NUM to ASCII from the String menu on the Ladder toolbar.
2. Place the function in the net.
3. In the HMI Display, select Display ASCII String from the Text Variable menu.

When the program shown below is downloaded, turning MB 1000 ON will display the value on the Vision's LCD.



- | | |
|----------------|--|
| Notes • | If the vector is not long enough, if for example you convert an ML value of "123456" into ASCII and allow only 5 characters, the function returns a string of question marks (??????). |
| • | Num to ASCII, floating value, is not supported by the V120-12 series. |

Strings: Transpose

Transpose enables you to 'compress' MI values into bytes, or 'expand' bytes into MIs:

- Define a source vector of registers that is offset from a selected start address.
- Copy the low byte of each register within that range,
- Define a target vector of operands that is offset from a selected start address.
- Select Conversion type:
MI to Byte (Compress) to write the low byte of each source register into the consecutive bytes of the target vector; thus the low bytes of 3 source registers will occupy 2 MIs.
Byte to MI (Expand) to write the consecutive bytes of the source vector into the low byte of each target register, thus the bytes of 3 MIs will occupy the low bytes of 6 MIs.

To use Transpose:

1. Click Strings on the Ladder Toolbar, then select Transpose.

2. Place the function in the desired net.

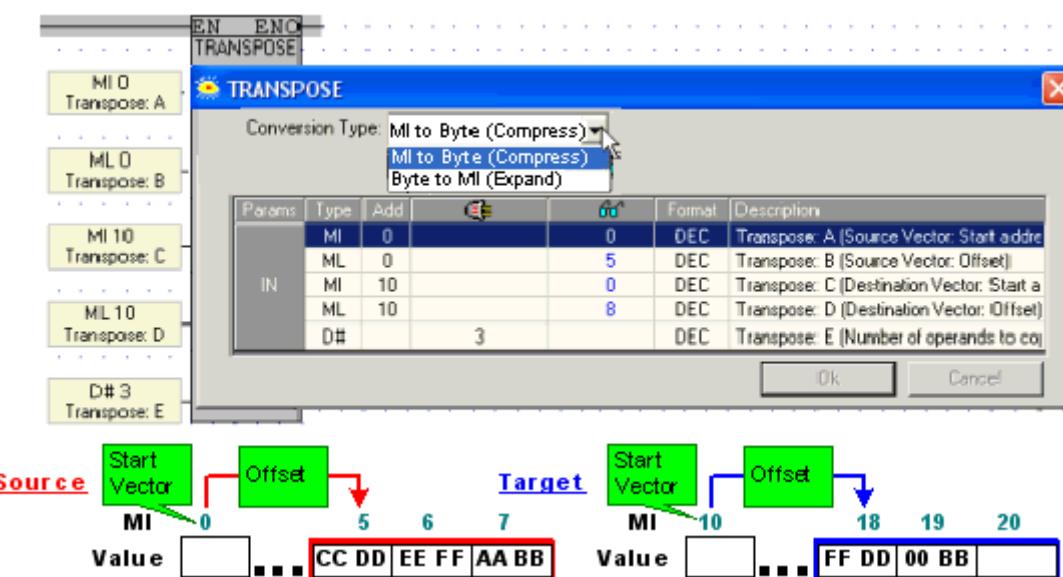
3. Select the type of function.

4. Link the desired Operands and Addresses.

Operand A: start address for the source vector.
Operand B: offset from the start address.
Operand C: start address for the target vector.
Operand D: offset from the start address.
Operand E: vector length.

Example:

Below, the low bytes of MI 5, 6, and 7 are copied into the consecutive bytes of MI 18 and 19.



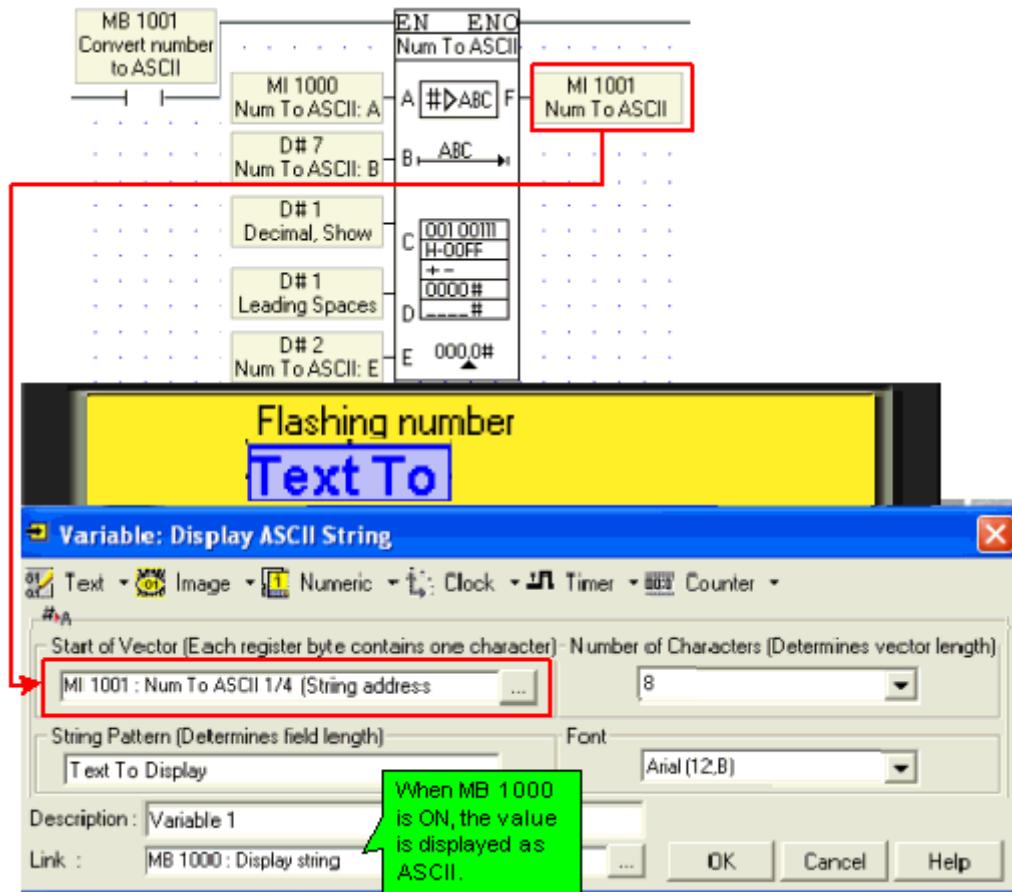
Strings: Num to ASCII, ASCII to Num

Num to ASCII

You can convert a value to an ASCII string and display it by using the Num to ASCII function together with the ASCII String variable.

1. Select NUM to ASCII from the String menu on the Ladder toolbar.
2. Place the function in the net.
3. In the HMI Display, select ASCII String from the Text Variable menu.

When the program shown below is downloaded, turning MB 1000 ON will display the value on the Vision's LCD.



- Notes •**
- If the vector is not long enough, if for example you convert an ML value of “123456” into ASCII and allow only 5 characters, the function returns a string of question marks (??????).
 - Num to ASCII, floating value, is not supported by the V120-12 series.

ASCII to Num

You can convert an ASCII string to a number value by using the ASCII to NUM function.

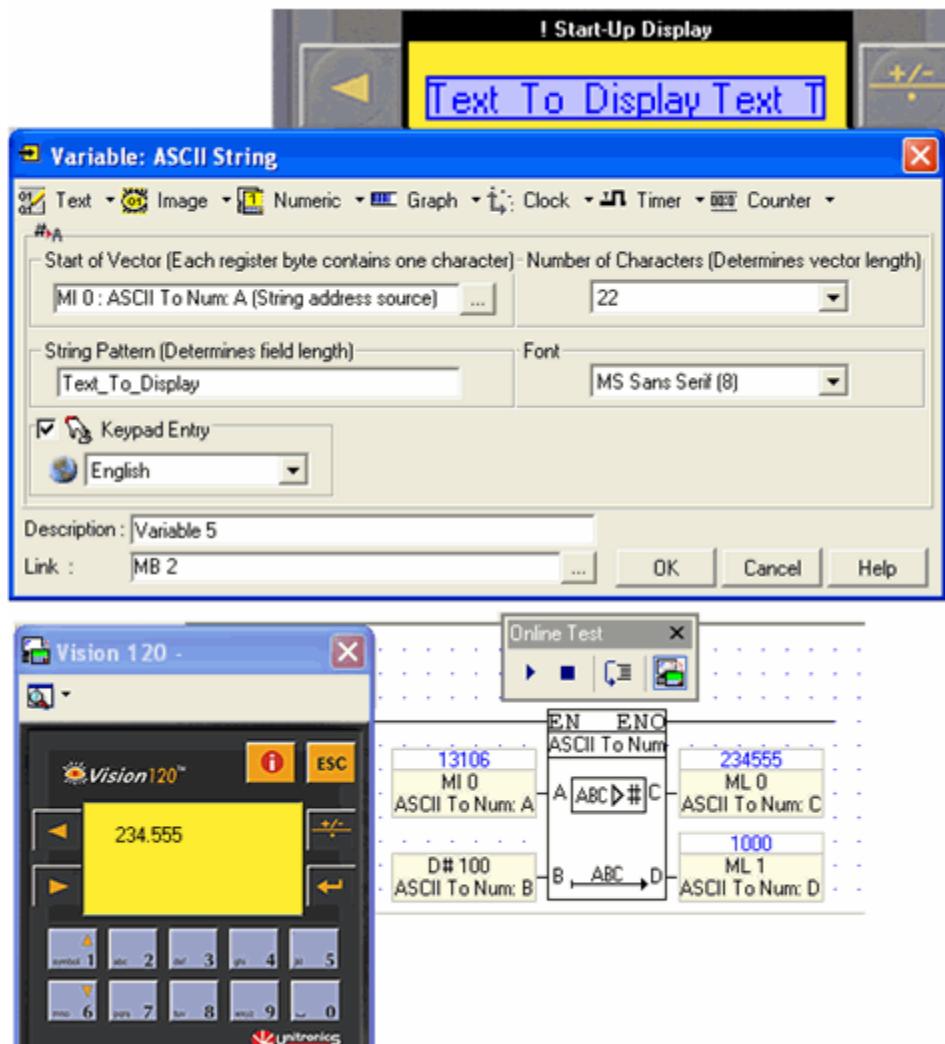
Operand A: Start address for the source vector.

Operand B: Vector length

Operand C: Start address for the destination vector.

Operand D: Factor (decimal point placement).

In the figure below, the value 234.555 is entered via keypad. The value is converted by the function; note that since the ASCII value is 234.555, the Factor is 1000.



Notes • ASCII to Num, floating value, is not supported by the V120-12 series.

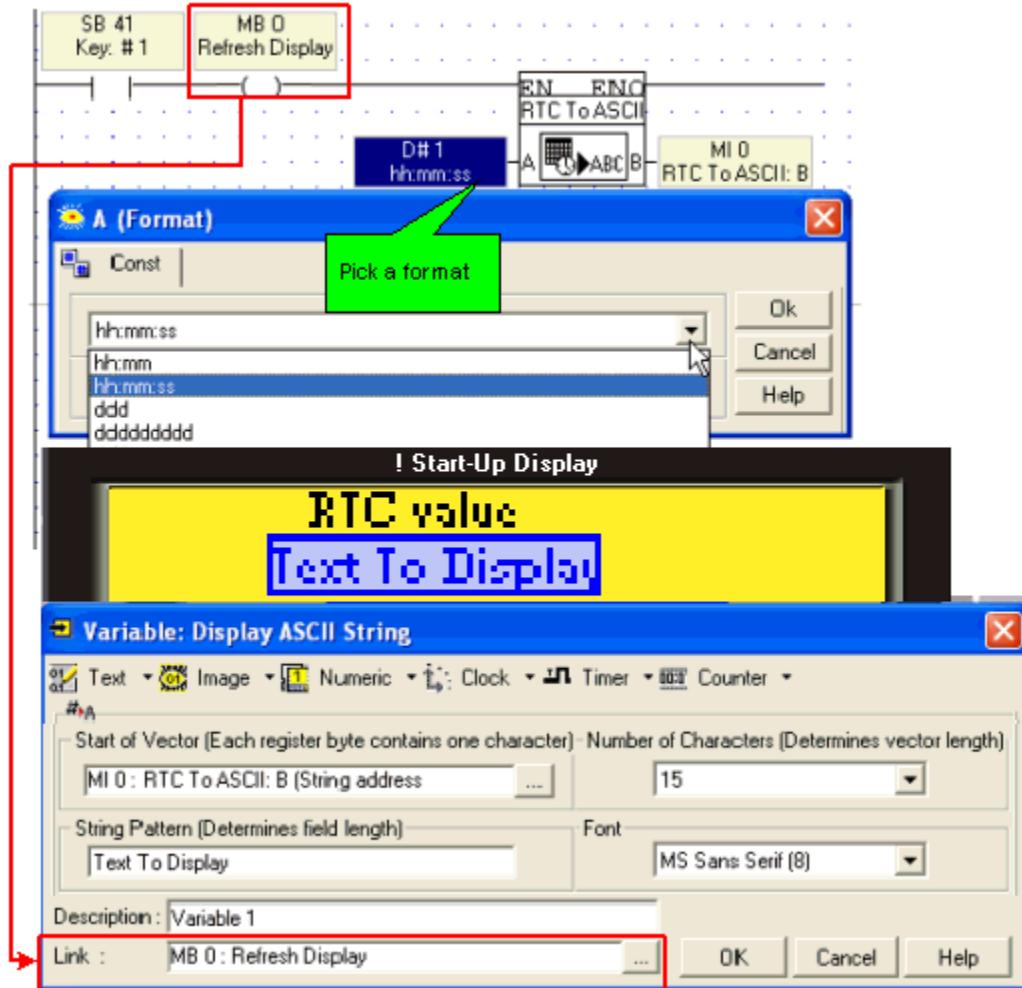
Strings: Display RTC (ASCII)

You can display an RTC value as an ASCII string by using the RTC to ASCII function together with the Display ASCII String variable.

To use Display RTC:

1. Select RTC to ASCII from the String menu on the Ladder toolbar.
 2. Place the function in the net, and select a display format; both European and American format are available.
 3. In the HMI Display, select Display RTC from the Text Variable menu.

When the program shown below is downloaded, pressing key 1 on the Vision's keypad will display the current time on the Vision's LCD.



Strings: IP to ASCII

You can display an value as an ASCII string by using the Num to ASCII function together with the Display ASCII String variable.

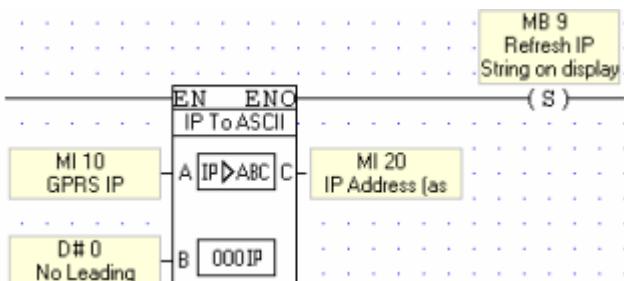
1. Select NUM to ASCII from the String menu on the Ladder toolbar.
2. Place the function in the net.
3. In the HMI Display, select Display ASCII String from the Text Variable menu and link it to the desired MI

When the program is downloaded, turning the linked MB ON will display the value on the Vision's LCD.

Notes •

If the vector is not long enough, if for example you convert an ML value of "123456" into ASCII and allow only 5 characters, the function returns a string of question marks (??????).

- This feature is not supported by the V120-12 series.

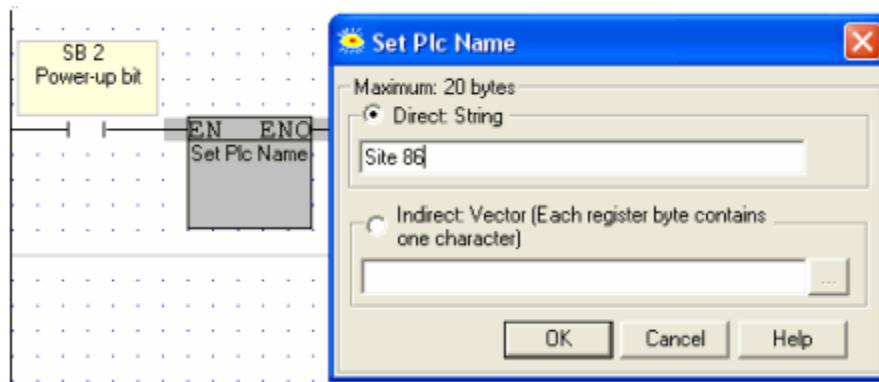


Com

Set PLC Name

Located on the COM menu, this function enables you to assign a unique name to a PLC. This name can, for example, be used to identify the PLC for Ethernet networking purposes.

The PLC name should be assigned as a power-up task.



- Notes •** When the function is activated, the name is written into the PLC. If the name is supplied via Indirect Vector, note that simply storing a value into the vector will not rewrite the PLC name. To rewrite the name, the value must first be stored in the appropriate vector, and then the Set PLC Name FB must be activated.
- This feature is not supported by the V120-12 series.

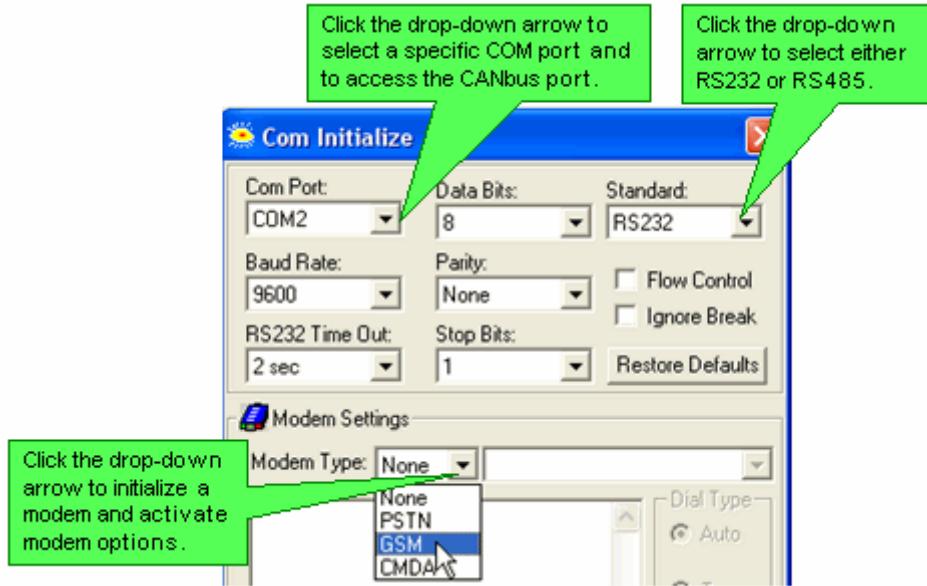
COM Port

COM Port: Init

COM Init is located on the Com menu. Use this function block:

- To initialize serial communication port settings and enable the controller to communicate with networked controllers, using protocols such as MODBUS; or to communicate with external devices such as modems.
- To initialize the CANbus port.
- To synchronize port settings, enabling the controller to engage in inter-device communications via protocols such as MODBUS.

- Notes •** COM Init is generally performed once in a program. It is usually a power-up task, however a one-shot transitional contact may also be used.
- All Vision controllers comprise RS232 serial ports, RS485 ports are not. To learn how to implement RS485 with different Unitronics' controllers, refer to RS485 Options.
 - Note that an Ethernet port is initialized via the Ethernet Card Init FB located on the FBS menu under Ethernet.
 - Where appropriate, use the system operands that are connected to the COM ports and that service communications.



Specific uses of the COM Init FB are detailed in the topics listed below.

- Modems
- CANbus Networking

Examples

The applications below use the COM Init function. To locate application examples, select Examples from the Help menu.

- SMS messaging.vlp
- GPRS.vlp
- MODBUS Slave.vlp
- MODBUS Master.vlp

Dial & Hang-up

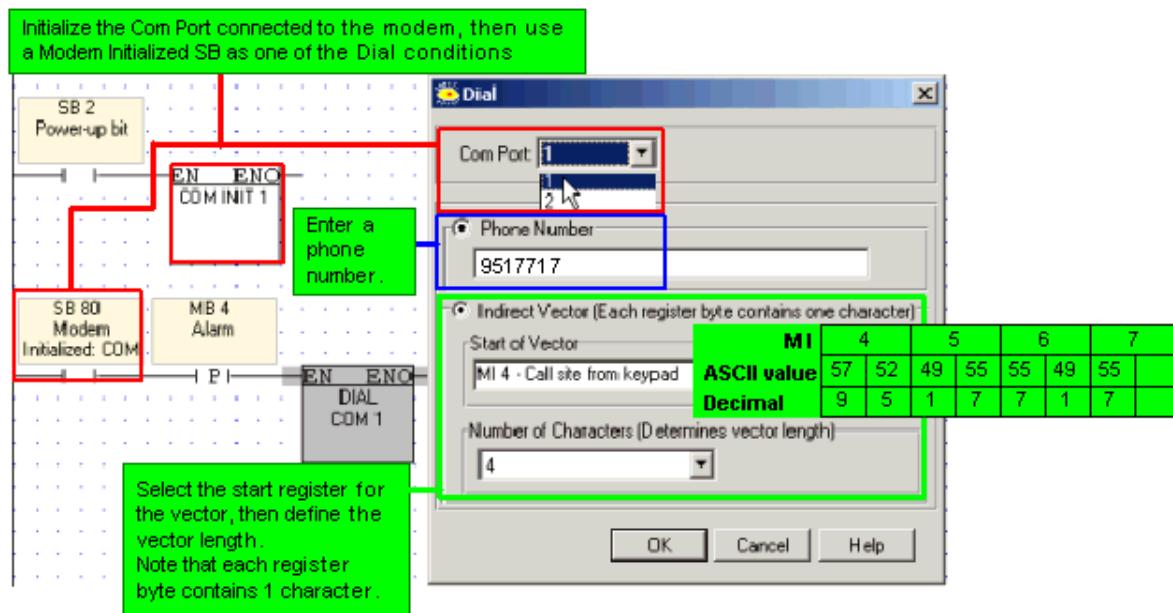
These functions are located on the Com menu. Via the Ladder application, they enable a PLC connected to a modem to establish or terminate a data link to another remote modem.

Before you dial, you must enable the Vision controller to communicate via modem.

Dial

This enables you to use Ladder conditions to dial a modem.

- | | |
|----------------|--|
| Notes • | In the conditions used to activate Dial, include the appropriate Modem Initialized SB: 80, 82, or 84 |
| • | SMS operations can conflict with applications that use the modem for other data communication processes. To prevent conflicts, use the Modem Busy (GSM) MB, and use an MB to indicate when the modem is in use by another data communications process. |



Hang-up

This enables you to use Ladder conditions to break the connection.

Note • Before activating Hang-up, check connection status via a Modem Connection SB: 86, 87, or 88.

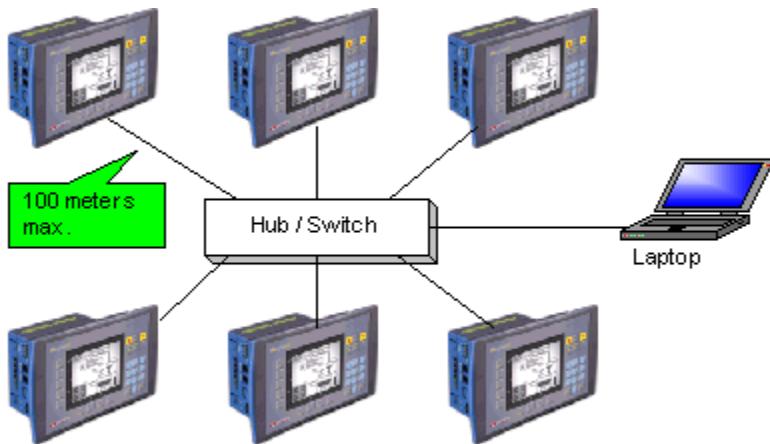


Ethernet

Using Ethernet

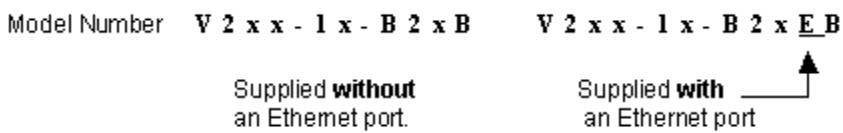
Unitronics currently supports both TCP and UDP protocols, as explained in the topic About Ethernet. This topic also contains general information about Ethernet, IP addressing, sockets, and ports.

Ethernet uses star topology.



In order to use Ethernet, your controller must comprise an Ethernet port.

V2xx Vision OPLCs can be ordered with or without an Ethernet port. The Ethernet port enables you to implement communications via TCP/IP, such as MODBUS over TCP. To check if your Vision controller was supplied with an installed Ethernet port, first check the device's model number. In addition, note that the Ethernet port is an RJ-45-type port that is lined with metal.

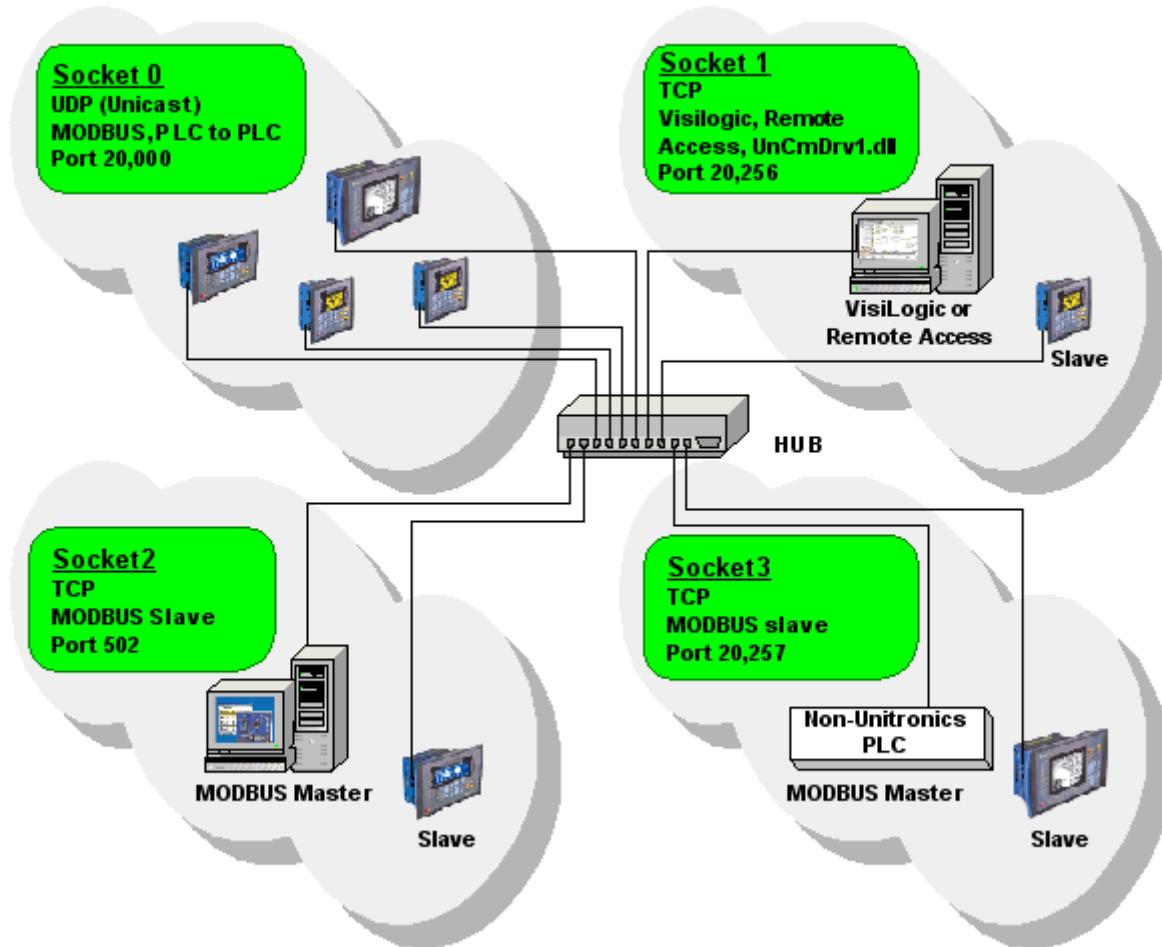


Via Ethernet, you can use the MODBUS IP FB to:

- Communicate data within a PLC network.
- Use a PC to access a PLC via MODBUS over TCP.
- Use MODBUS over TCP to enable non-Unitronics PLCs to access Unitronics PLCs, via MODBUS.

You can also use Ethernet to enable a PC running VisiLogic, Remote Access, or other communication .dll to access a networked PLC.

The default socket configuration enables you to implement these communication options as shown below:



Default Socket Configuration

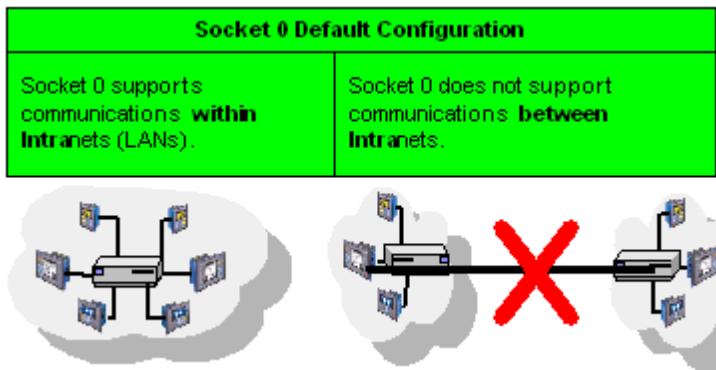
Vision controllers currently offer 4 sockets. The default socket configuration includes:

Socket	Protocol	Port Number	Function
0	UDP	20,000	<p>Enables data to be both transmitted and received within a PLC network, via MODBUS.</p> <p>Note• If you are using the default settings for Socket 0, note that data is sent via Unicast to IP: 255.255.255.255, port: 20,000 plus the last byte of the IP address originally assigned to the device.</p> <p>This is why Port numbers 20,000-20,255 are reserved for Socket 0.</p>
1	TCP	20,256	Enables PC to PLC communication via UnCmDrv1.dll, including VisiLogic, Remote Access, and other Unitronics communication applications.
2	TCP	502	Set to 'listen' as slave (server), enables MODBUS applications such as OPC servers and SCADA systems which use MODBUS TCP over IP.
3	TCP	20,257	Set to 'listen' as slave (server), enables non-Unitronics PLCs to access Unitronics PLCs, via MODBUS.

Note• The default configuration means that, for most applications, you do not need to include a Socket Init FB in the ladder application. However, if, for example, your application requires 4 sockets for TCP, change the default configuration of Socket 0 from UDP to TCP via the Socket Init FBs.

- When using the default socket configuration, Socket 0 cannot be used to communicate data between routers, and therefore cannot transfer data between Intranets as shown in the

figure below. This is because the default configuration for Socket 0 uses Unicast.



- Note that when TCP is used, the formal 'handshake' required by the protocol means that, during each session occurring via a defined socket, other communications cannot flow through that socket until the current session has been terminated.
Such is not the case with UDP. Since there is no formal handshake, communications can continue to flow through a socket even when there are multiple requests.

General

When using Ethernet, use the MODBUS IP FBs. For detailed information regarding MODBUS IP commands, refer to the MODBUS IP help topics.

- | | |
|---------------|--|
| Note • | <p>In order to implement Ethernet, a controller must be assigned an IP address. This is done via the Ethernet Card Init FB, which must be included in the Ladder applications of both master and slave controllers.</p> <p>Class C-type addresses are recommended, as explained in the topic About Ethernet.</p> |
| • | <p>When the Ethernet card finishes initialization, SB 142 rises. Use this as a condition before activating any Ethernet element, such as Socket Connect.</p> |
| • | <p>An activating condition must be placed before the Ethernet Card Init FB. This may be assigned as a power-up task; however a one-shot transitional contact may also be used.</p> |
| • | <p>Unitronics' proprietary COM Protocol FB, located on the FBs menu, which may ordinarily be used to access external slave devices, is not currently compatible with Ethernet.</p> |

Examples

PLC networks, PLC to PLC

Any controller within the network can be both master and slave. In order to be read by the master, a slave's application must contain the MODBUS IP Scan FB.

Using UDP to implement controller-to-controller communication

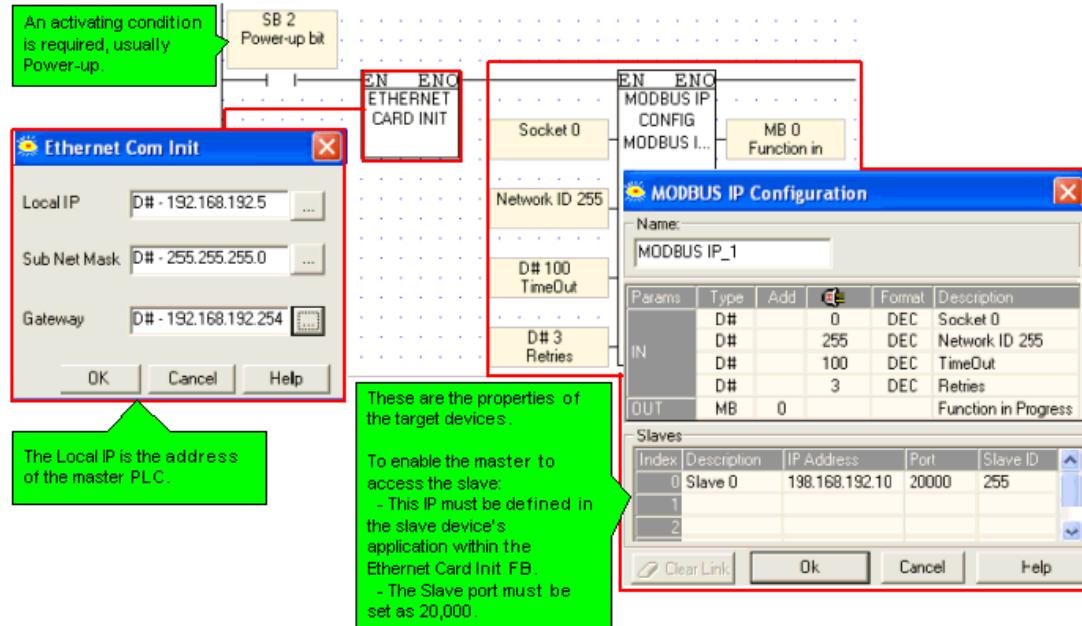
In order to communicate via Ethernet throughout your controller network, you must include an Ethernet Card Init FB in the ladder application of each networked controller. Remember that, when using UDP, do not use the Socket: Connect or Socket: Close elements; these are only required by TCP applications.

- Master

The master PLC Ladder application must include the elements shown below.

Step 1: Initializing the Ethernet card and configuring MODBUS

The MODBUS Configuration is linked to Socket 0, which is by default set to UDP.

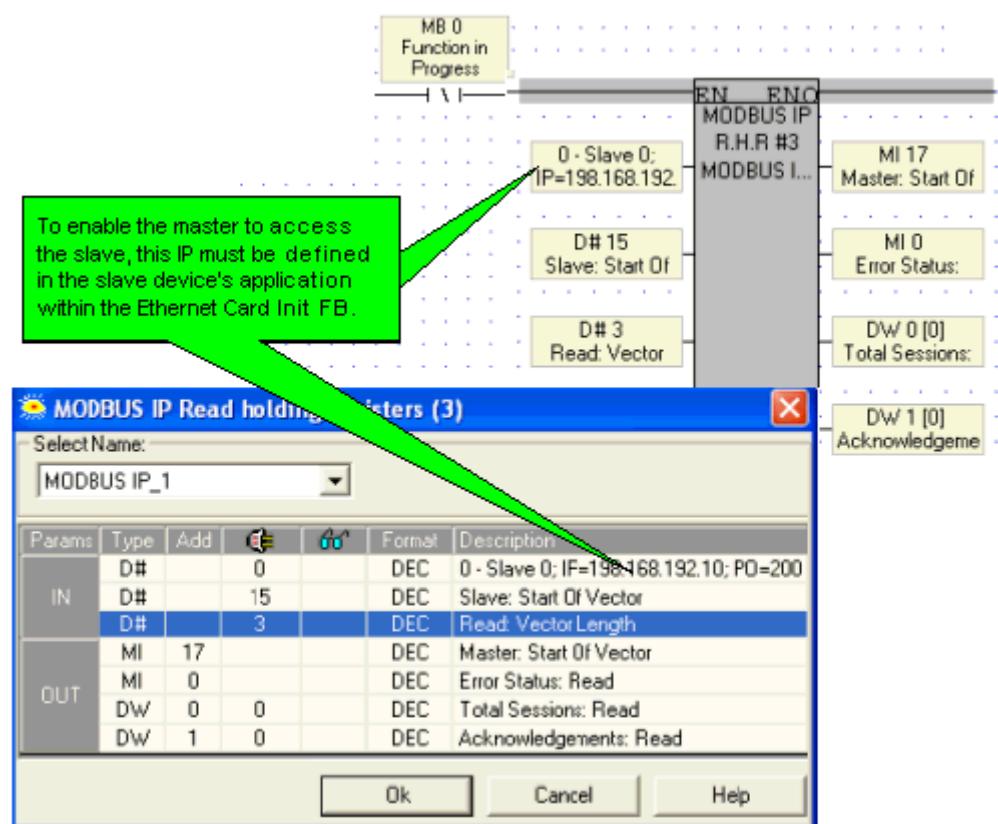


Note • A PLC defined as a UDP master can communicate with a number of slave devices.

Step 2: Using MODBUS Commands

Note • Note that the operand addresses in slave PLCs are indirect addresses (pointers). In the figure below, the Slave: Start of Vector parameter is 15. This means that the master will begin reading from MI 15 in the slave PLC. Since the Read: Vector Length parameter is 3, the function takes the values in MI 15, 16 and 17.

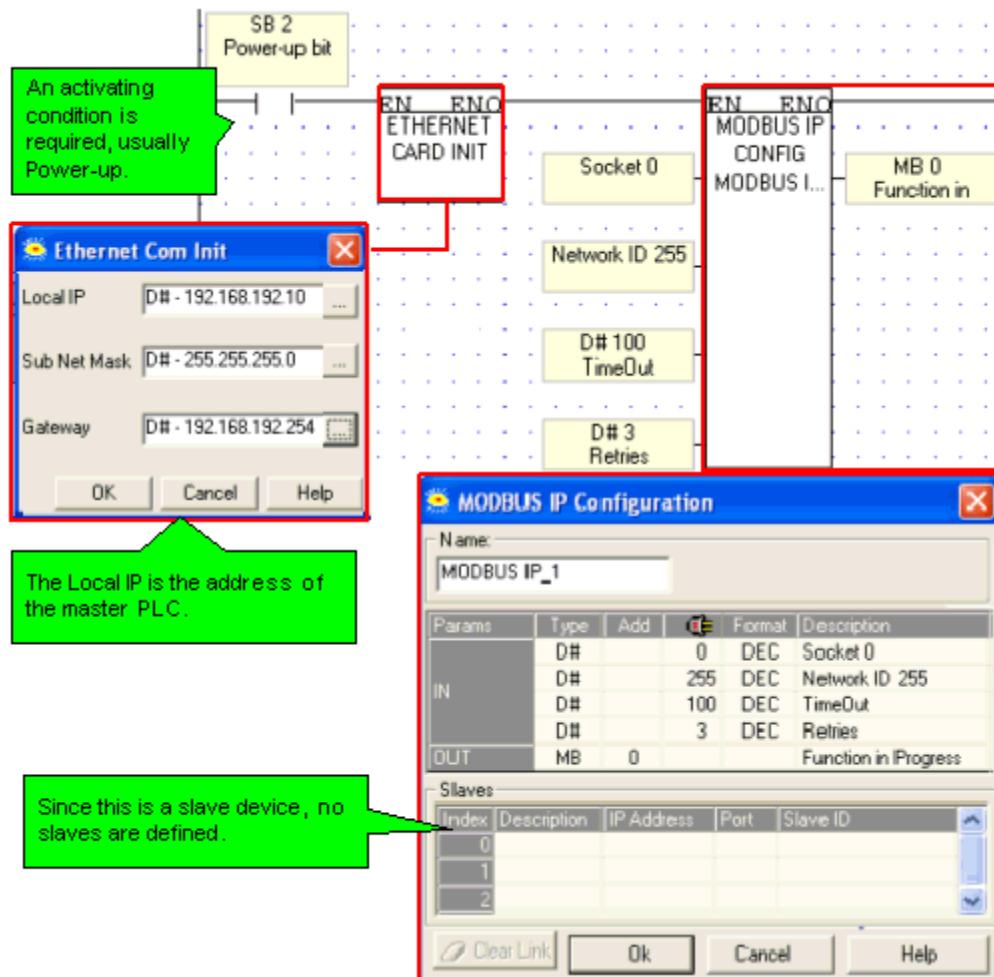
The Master: Start of Vector parameter is 17; therefore the values will be written into MI 17, 18, and 19 in the master device.



- Slave

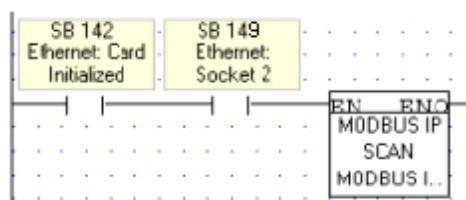
The slave PLC Ladder application must include the elements shown below.

Step 1: Initializing the Ethernet card and configuring MODBUS



Step 2: Scan

To enable the master PLC to access the slave, include a MODBUS Scan FB in the slave's application.



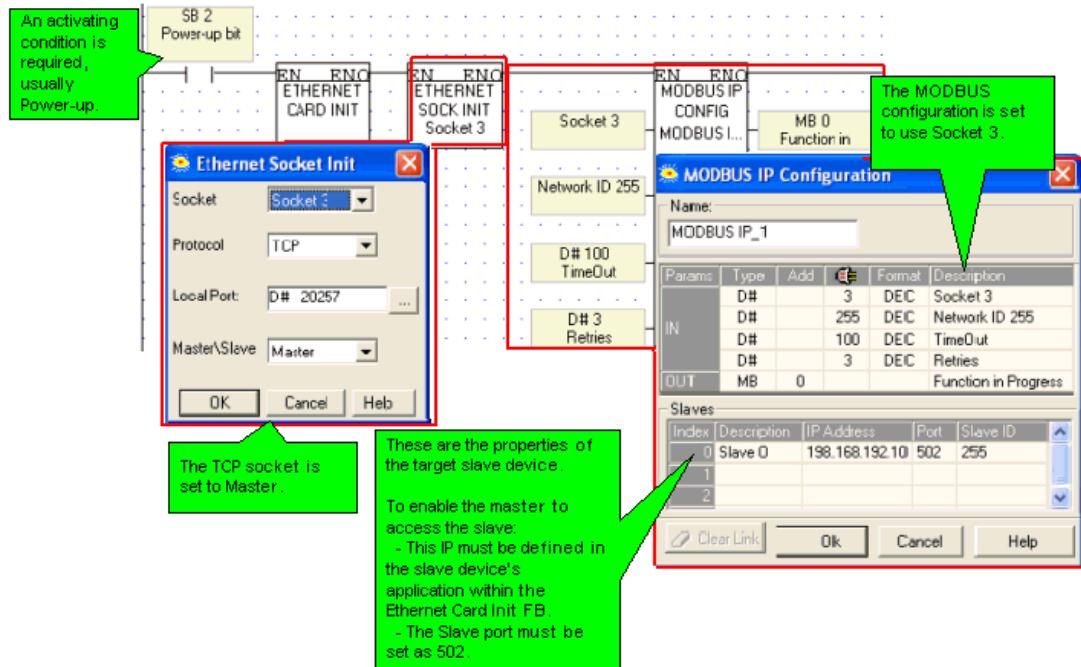
Using TCP to implement controller-to-controller communication

- Master

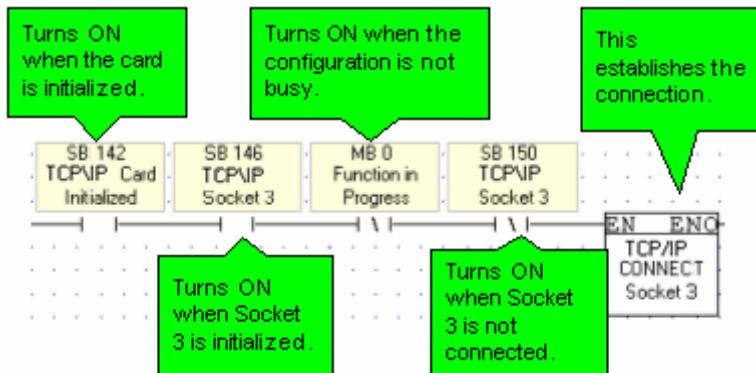
The master PLC Ladder application must include the elements shown below.

Step 1: Initializing the Ethernet card, Socket, and Configuring MODBUS

In the figure below, the socket is configured to use TCP.



Step 2: Establishing the Ethernet Connection: Connect Socket



Note •

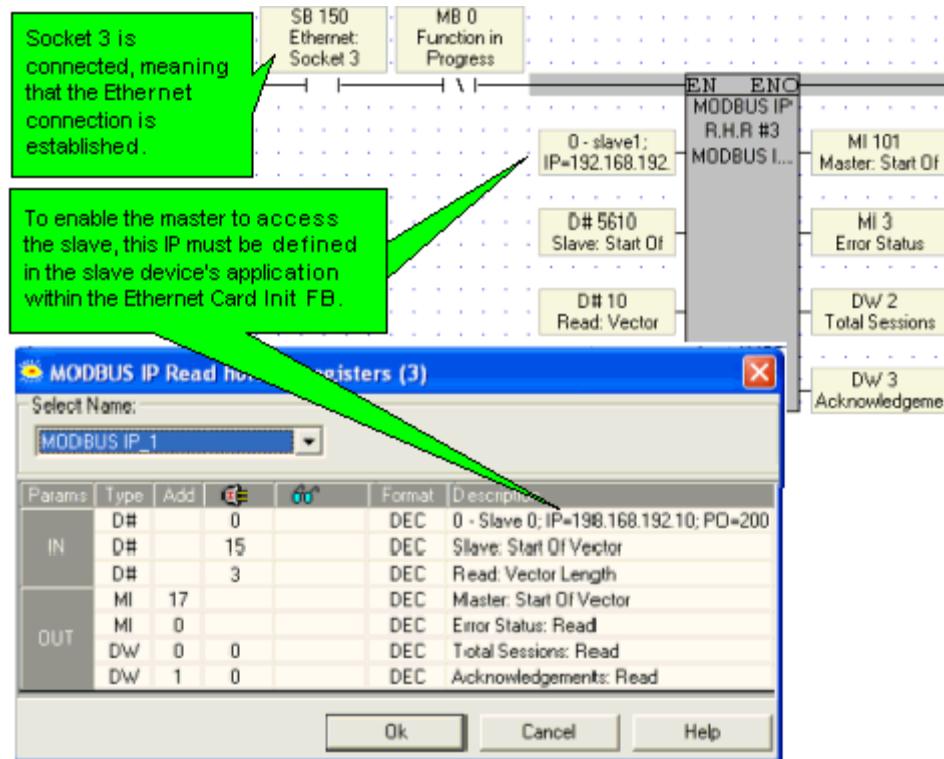
It is recommended that there be a time elapse of a few seconds after the Ethernet Card Initialization and before activating Socket Connect. A timer may be used for this purpose.

Step 3: Using MODBUS Commands

Note •

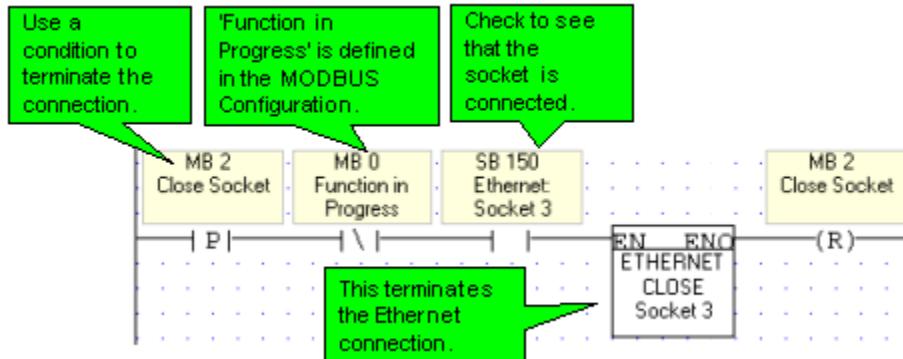
Note that the operand addresses in slave PLCs are indirect addresses (pointers). In the figure below, Below, the Slave: Start of Vector parameter is 15. This means that the master will begin reading from MI 15 in the slave PLC. Since the Read: Vector Length parameter is 3, the function takes the values in MI 15, 16 and 17.

The Master: Start of Vector parameter is 17; therefore the values will be written into MI 17, 18, and 19 in the master device.



Step 4: Terminating the Ethernet connection: Close Socket

When you terminate the connection, use the 'Function in Progress' MB to ensure that you do not terminate the connection while data is being communicated.

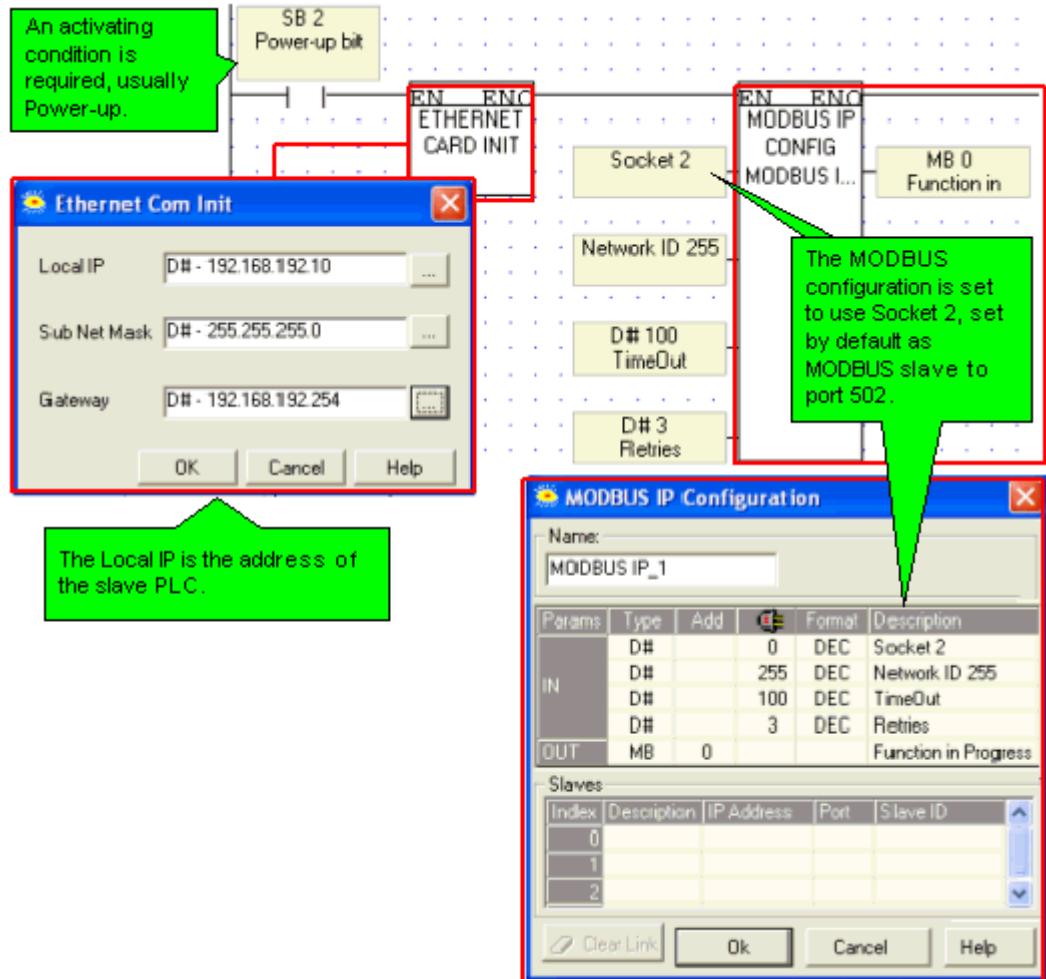


- Slave

The slave PLC Ladder application must include the elements shown below.

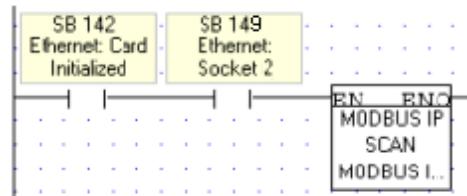
Step 1: Initializing the Ethernet card, Socket, and Configuring MODBUS

In the figure below, the socket is configured to use TCP.



Step 2: Scan

To enable the master PLC to access the slave, include a MODBUS Scan FB in the slave's application.

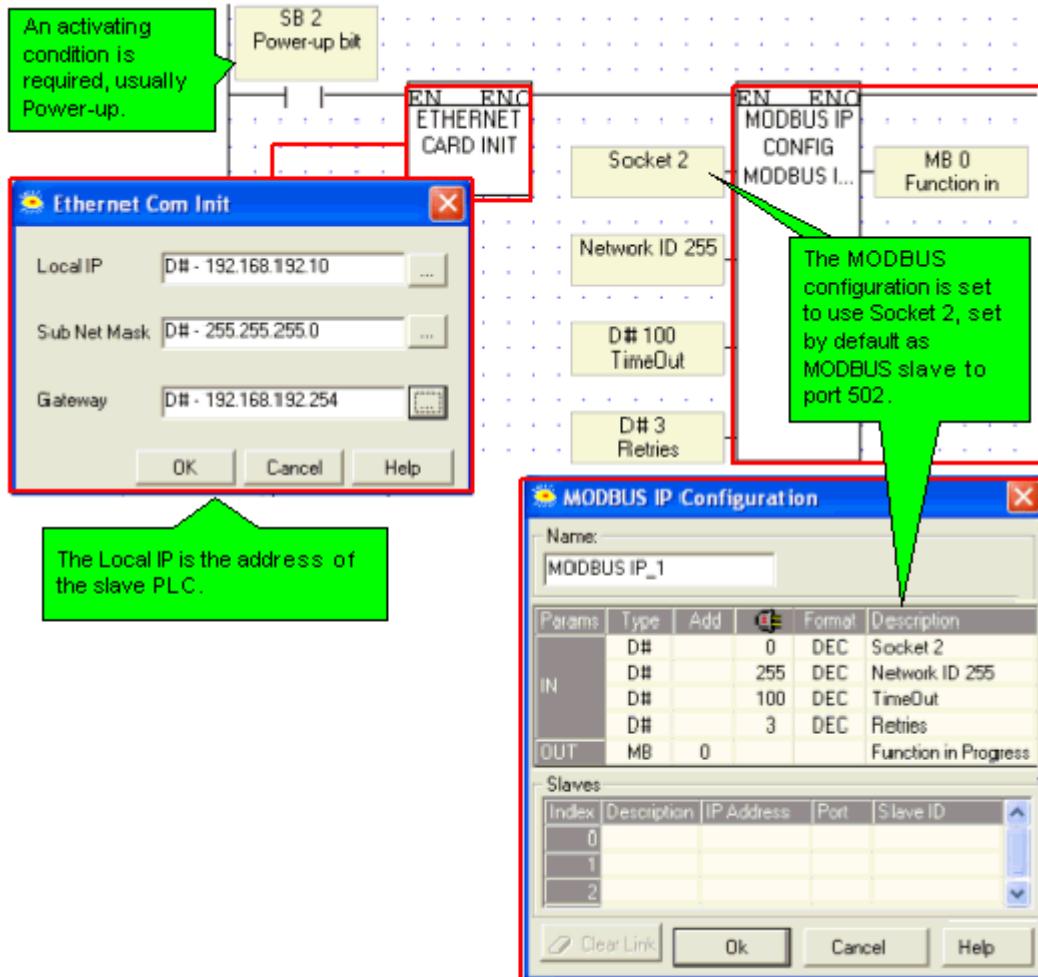


PC to PLC: Accessing PLC via SCADA

To enable the SCADA application to access the PLC, the PLC is defined as a slave device. The slave PLC Ladder application must include the elements shown below.

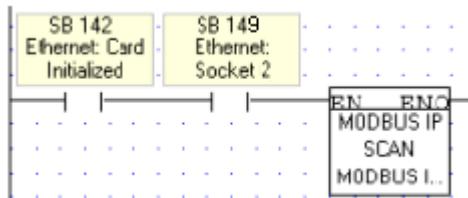
Step 1: Initializing the Ethernet card and configuring MODBUS

Port 502 is the well-known port for MODBUS applications.



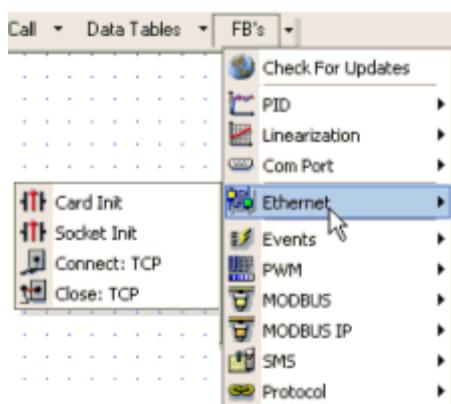
Step 2: Scan

To enable the SCADA application to access the slave, include a MODBUS Scan FB in the slave's application.



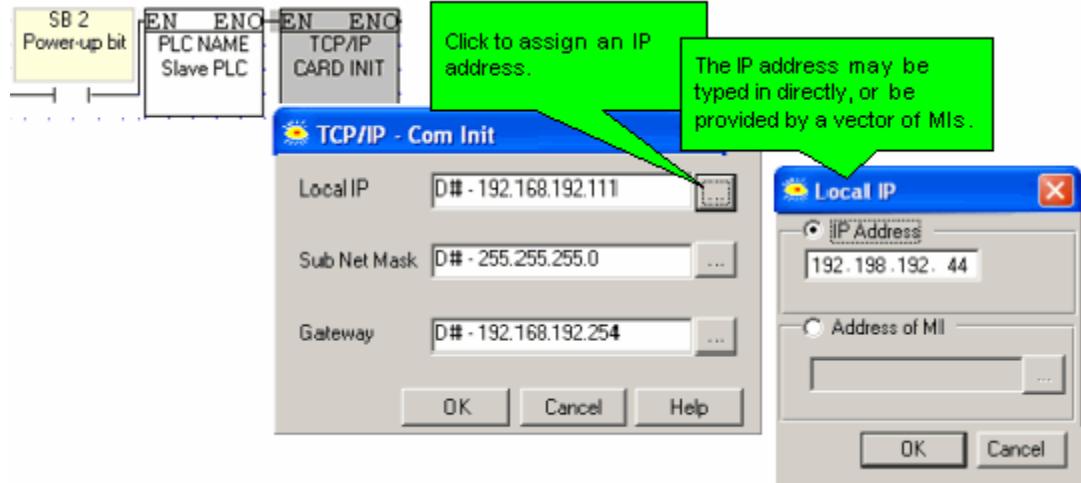
Ethernet Operations

The Ethernet FBs are grouped under Ethernet on the FB's menu.



Ethernet: Card InitEthernet: Socket InitEthernet: TCP Connect \ TCP CloseTCP\IP: Card Init

This function is located on the Com>TCP/IP menu.



If you assign an IP address indirectly, via an MI vector, note that the vector is 4 MIs long. The low byte of each MI provides the number for an octet within the IP address.

If, for example, the IP address is linked to MI 0, and the low bytes of MI 0 to MI 3 contain the values 192, 198, 192, 45, the IP address will be 192.198.192. 45.

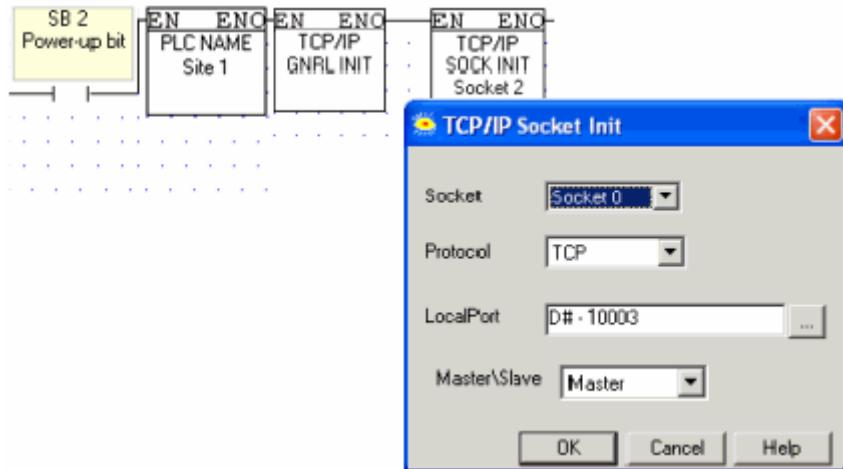
Note •	In order to implement Ethernet, a controller must be assigned an IP address. This is done via the TCP\IP Init FB, which must be included in the Ladder applications of both master and slave controllers. Information on IP addressing is given in the topic About Ethernet
•	When the Ethernet card finishes initialization, SB 142 rises. Use this as a condition before activating any Ethernet element, such as Socket: Connect.
•	An activating condition must be placed before the Ethernet Card Init FB. This may be assigned as a power-up task; however a one-shot transitional contact may also be used.
•	If you have linked the IP address to a vector of MIs, and this condition is not activated, the IP address will not be assigned to the controller. Make sure, for example, that if you have used a power-up condition, that the controller does go through power-up.

TCP\IP: Socket Init

This function is located on the Com>TCP/IP menu.

Vision controllers currently offer 4 sockets.

The default configuration means that, for most applications, you do not need to include a Socket Init FB in the ladder application. However, if, for example, your application requires 4 sockets for TCP, change the default configuration of Socket 0 from UDP to TCP via the Socket Init FBs.



The default socket configuration includes:

Socket	Protocol	Port Number	Function
0	UDP	20,000	<p>Enables data to be both transmitted and received within a PLC network, via MODBUS.</p> <p>Note • If you are using the default settings for Socket 0, note that data is sent via Unicast to IP: 255.255.255.255, port: 20,000 plus the last byte of the IP address originally assigned to the device.</p> <p>This is why Port numbers 20,000-20,255 are reserved for Socket 0.</p>
1	TCP	20,256	Enables PC to PLC communication via UnCmDrv1.dll, including VisiLogic, Remote Access, and other Unitronics communication applications.
2	TCP	502	Set to 'listen' as slave (server), enables MODBUS applications such as OPC servers and SCADA systems which use MODBUS TCP over IP.
3	TCP	20,257	Set to 'listen' as slave (server), enables non-Unitronics PLCs to access Unitronics PLCs, via MODBUS.

Note •

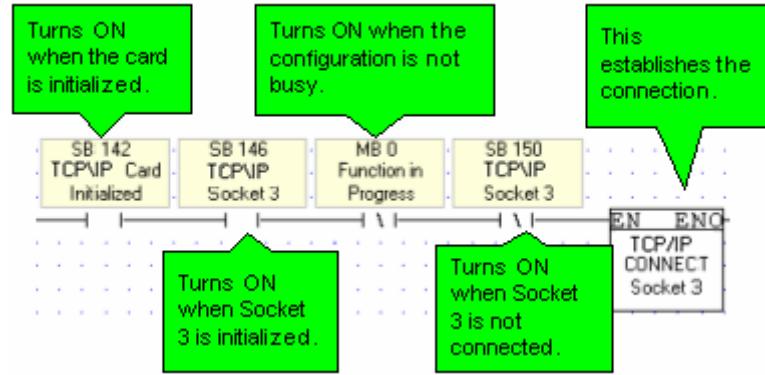
When TCP is used, the formal 'handshake' required by the protocol means that during each session occurring via a defined socket, other communications cannot flow through any of the other sockets until the current session has been terminated.

Such is not the case with UDP. Since there is no formal handshake, communications can continue to flow through a socket even when there are multiple requests.

TCP\IP: TCP Connect \ TCP Close

TCP applications require you to use a TCP: Connect FB to establish the Ethernet connection after the Ethernet card is initialized and before activating any of the MODBUS IP commands.

To terminate the session, use the TCP: Close FB. Both elements are located on the Com>TCP/IP menu.



HMI Ladder Functions

HMI-Ladder: Load HMI Display: Functions

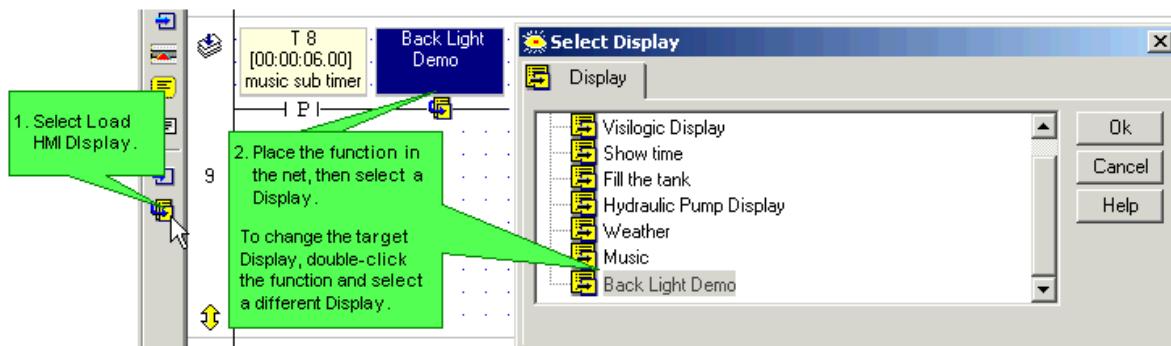
These Ladder functions call HMI Displays.

Note •

Load Display functions should not be placed directly on the Ladder rail, or called by conditions that continually call the Display when it is still loaded on the controller screen. Use these functions to initially load the Display, then to refresh it when your application requires, as, for example, when you want to update variable display.

Load HMI Display

Causes a Display to be shown on the controller's LCD as a response to a Ladder Condition.



HMI Display Loaded

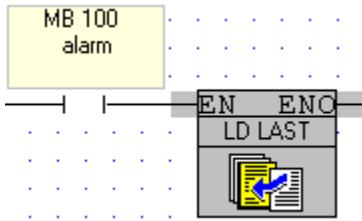
This turns a linked MB ON when a specific Display begins loading. HMI Display Loaded is located on the Calls menu.



Load Last Display

Loads the last Display loaded by the application. The function works according to LIFO list comprising the last 24 active Displays.

This function is located on the Calls menu.

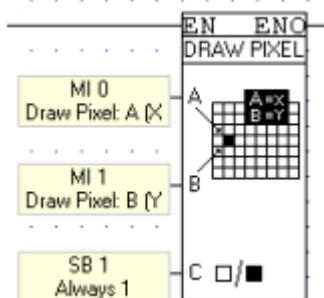


To see a list of HMI Displays in a project, together with the Display number, select HMI Information from the View menu.

HMI-Ladder: Draw Pixel/Line

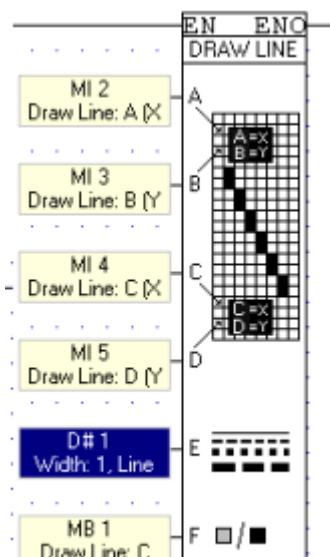
These elements allow Ladder events to color pixels or draw lines on the controller's LCD display. Both Draw Pixel and Draw line are located on the HMI toolbar.

Draw Pixel enables you to color a single pixel located on the x,y axis.



Input	Purpose	Comments
Input A	X location	
Input B	Y location	
Input C	Pixel color	If the value of the linked bit is 1 (set), the pixel will be black, if 0 (reset), the pixel will be negative. SB 1 may be used to color the pixel black, SB 0 to color it negative, or an MB may be used.

Draw line enables you to draw lines in different widths and formats.

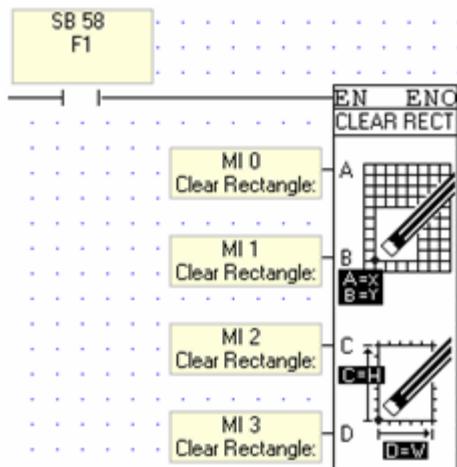


Input	Purpose	Comments
Input A	Start X location	

Input B	Start Y location	
Input C	End X location	
Input D	End y location	
Input B	Format	Select line width: 1 to 4 pixels wide, and line style: solid, dot. or dash.
Input E	Pixel color	If the value of the linked bit is 1 (set), the line will be black, if 0 (reset), the line will be negative. SB 1 may be used to color the line black, SB 0 to color it negative, or an MB may be used.

HMI-Ladder: Clear Rectangle

This element allows Ladder events to 'erase' a rectangular area on the controller's LCD display in response to a Ladder event. Clear Rectangle is located on the HMI toolbar.



The parameters below set the location and size of the rectangle.

Input	Purpose
Input A	Start X location
Input B	Start Y location
Input C	Width
Input D	Height

Inverse Var/Hide Var

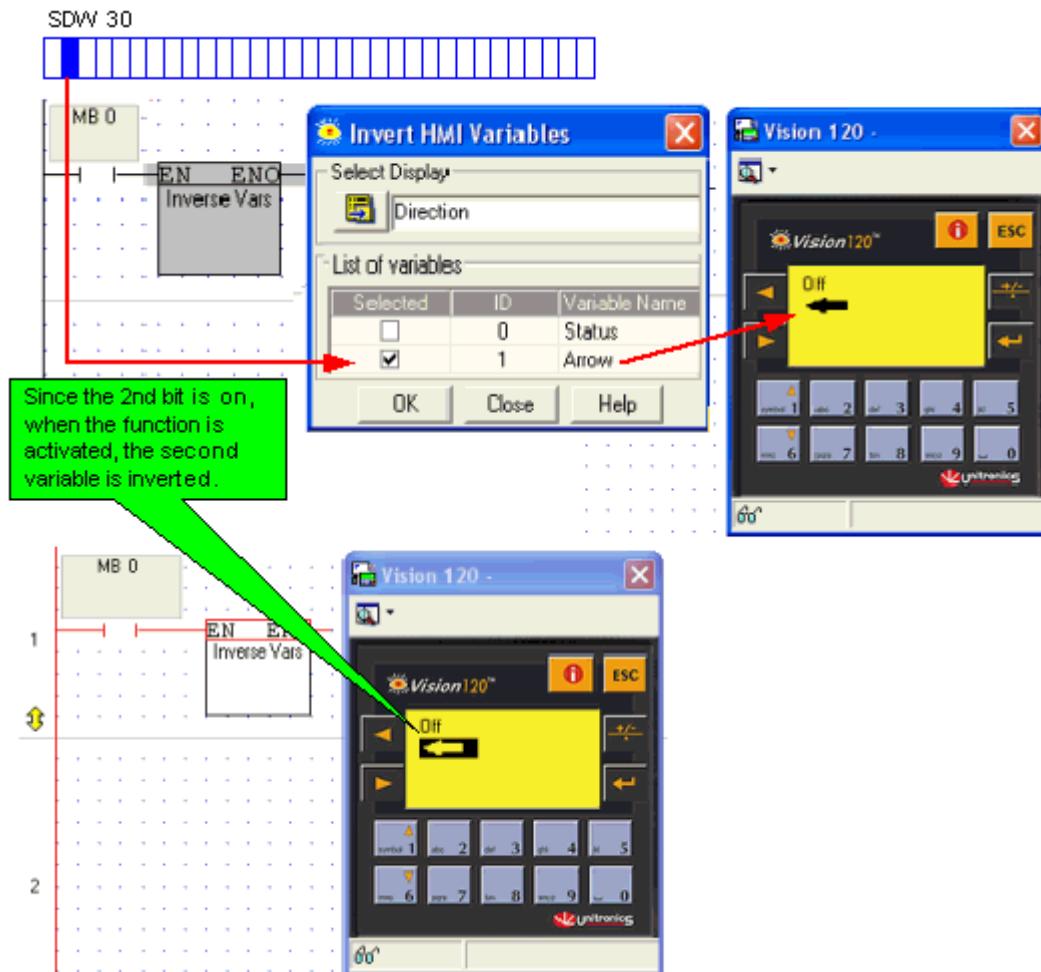
The Inverse Var function 'inverts' the color of a variable, meaning that black pixels are changed to white and white to black.

The Hide Var function hides a variable.

How Inverse/Hide work

Each function is linked to its own SDW, Inverse Var to SDW 30, and Hide Var to SDW 31. The SDW provides a bitmap for the variables in the Display currently shown on the LCD.

Both functions work in the same way: by checking the value of the linked SDW when a Display is activated.



#	Description	Value	Comments
SDW 30	Variable display bitmap, 0=Normal, 1=Inverse (or negative)	The value is checked when a display is entered. It is initialized to 0: - At Power-up. - When the program exits the Display.	When a bit is ON, the corresponding variable is displayed in inverted (negative) color; black pixels are changed to white and white to black.
SDW 31	Hide Var	The value is checked when a display is entered. It is initialized to 0 at: - Power-up. - When the program exits the Display.	When a bit is ON, the corresponding variable is hidden.

How to use Inverse\Hide

- Link the Display containing the variables to a Subroutine.

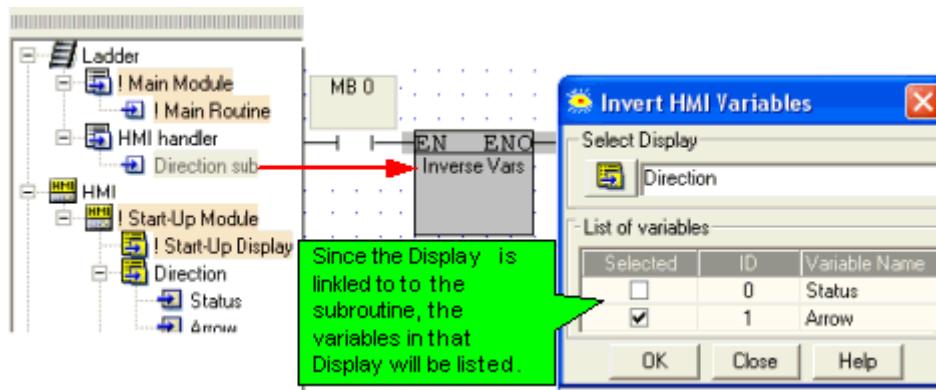


- Place the Inverse\Hide Var function in a subroutine, not in the Main routine

Note •

If the function is in the Main routine, it will not work correctly.

3. Select the desired variables.

**Notes •**

The SDW bits are linked to the variable index number, which changes when variables are added or deleted, as well as during copy/paste. If you edit the variables after inserting Inverse/Hide functions, check that the desired variables remain selected.

- The functions automatically update the variable view of whichever Display is currently on-screen.

Floating Math Functions

Float Functions

Float function blocks enable you to use Memory Float (MF) values in your program.

The Float menu on the Ladder toolbar includes the following functions:

- Basic
- Extended
- Trig
- Compare
- Convert

Note •

Floating point values cannot be directly displayed on the controller screen. In order to display a floating point value, use the Convert Float INV function to express the value in 2 MIs or MLs, and then use a Display number variable to show them on screen.

Float Functions: Basic

These are the basic Float functions:

- Store Direct
Stores a register value into an MF.
- Add
Adds two values and stores the result in an MF.
- Sub
Subtracts two values and stores the result in an MF.
- Mul
Multiplies two values and stores the result in an MF.

- Div
Divides two values and stores the result in an MF.
- Abs
Returns the absolute value of an MF or constant number. The absolute value of a number is the number without its sign.

If, for example, the input value is -2, the absolute number output by the Abs function will be 2.

Float Functions: Extended

These are the extended Float functions:

- Square root
This function returns the square root of an input value. The input value serves as the radicand. The result is stored in an output MF.
- Power
Power uses 2 input values. Power raises an A input value by the power of a B (exponent) input value. The result is stored in an output MF, C
- Exp
Returns the value of the input number raised to the power of 'e'. The constant e equals 2.71828182845904, the base of the natural logarithm.

EXP is the inverse of LN, natural log. If, for example, the value 1 is input to the Exp function, the output result is 2.718282. If the value 2 is input, the output result will be 7.389056.
- LN
Returns the natural logarithm of the input number, using base 'e'. The constant e equals 2.71828182845904.

LN is the inverse of Exp. If, for example, the value 6 is input to the LN function, the output result is 1.791759. If the value 6- is input, the output result will be 4.094345.
- Log10
Returns the logarithm of the input number, using base 10.

If, for example, the value 6 is input to the Log10 function, the output result is 0.7781513. If the value 60 is input, the output result will be 1.778151; an input of 600 results in an output of 2.778151
- A(10^B)
A(10^B) uses 2 input values. The A value is multiplied by 10 to the power of the B value.

If, for example, the A value is 3, and the B value is 2, the output value will be 300: 3(10²). If A is 3 and B is 9, the result will be 3,000,000,000.

A(10^B) uses two input values. The A value is multiplied by 10 to the power of the B value.

If, for example, the A value is 3, and the B value is 2, the output value will be 300: 3(10²). If A is 3 and B is 9, the result will be 3,000,000,000.

Float: Trig Functions

These are the Trigonometric functions:

- Sin
The function's output is the sine of the input value.

- Cos
The function's output is the cosine of the input value.
- Tan
The function's output is the tangent of the input value.
- ArcSin
The function's output is the inverse sine of the input value.
- ArcCos
The function's output is the inverse cosine of the input value.
- ArcTan
The function's output is the inverse tangent of the input value.
- Degrees
Converts the input value into degrees.
- Radians
Converts the input value into radians.

Float: Convert

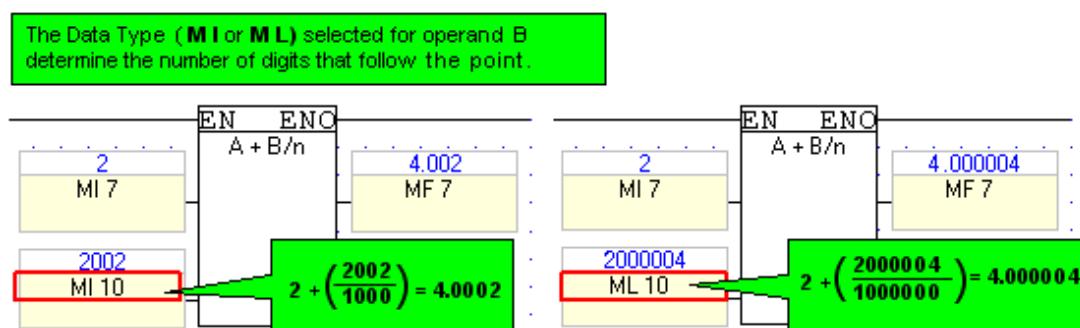
These are the Convert Float functions:

- A+B/n

This function takes 2 non-float values (whole numbers) and creates a single floating value.

The two non-float values are added together; the A input, a whole number, is added to the B input, which is the fractional part of the number following the decimal point.

Note • The Data Type (M I or M L) selected for operand B determine n, the number of digits that follow the point. When an MI is selected, n equals 1000; when an ML is selected, n equals 1,000,000.

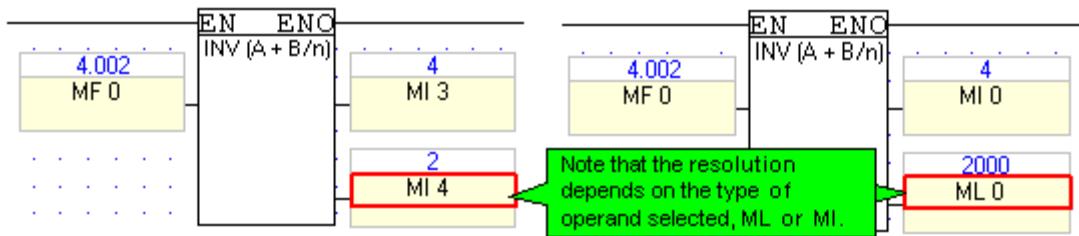


- INV (A+B/n)

Casting separates an MF value into two registers, where output A contains the whole number, and output B contains the fractional part of the number following the decimal point.

This function enables you to show floating-point values on the controller screen, by using 2 Numeric Display Variables, linked to the output MIs.

If an MI is selected for output operand B, it may not be long to hold the fractional value.



- Note •** The Data Type (M I or M L) selected for operand B determine n, the number of digits that follow the point. When an MI is selected, n equals 1000; when an ML is selected, n equals 1,000,000.

Float: Compare Functions

These are the Compare Float functions:

- Greater Than

The Greater Than function compares the value of input A to input B.

When the function is activated:

- If input A is greater than input B, the output MB turns ON.
- If input A is not greater than input B, the output MB turns OFF.

Greater Than or Equal To

The Greater Than or Equal function block compares the value of input A to input B.

When the function is activated:

- If input A is greater than or equal to input B, the output MB turns ON.
- If input A is not greater than or equal to input B, the output MB turns OFF.

Equal

The Equal function block compares the value of input A to input B.

When the function is activated:

- If input A is equal to input B, the output MB turns ON.
- If input A is not equal to input B, the output MB turns OFF.

Not Equal

The Not Equal function evaluates input A to see if its integer value is not equal to input B.

When the function is activated:

- If input A is not equal to input B, the output MB turns ON.
- If input A is equal to input B, the output MB turns OFF.

Less Than or Equal To

The Less Than or Equal To function compares input A to input B.

When the function is activated:

- If input A is less than or equal to input B, the output MB turns ON.
- If input A is not less than or equal to input B, the output MB turns OFF.
- Less Than

The Less Than function compares input A to input B.

When the function is activated:

- If input A is less than input B, the output MB turns ON.
- If input A is not less than input B, the output MB turns OFF.
- Within Range

The Within Range function block checks if the value in input A is within the range of values between input B and input C.

When the function is activated:

- If input A is within the range of values between input B and input C the output MB turns ON.
- If input A is not within the range of values between input B and input C the output MB turns OFF.

Float Errors

When an Float function error occurs, SB 10 Float Error turns on. This SB is reset by the user.

The error code is stored in SI 500 General Error. The codes are shown below.

Value	Message	Result
3	Integer Overflow	7FFF or 8000 (integer result) FFFF or 0000(unsigned integer result)
4	Floating Overflow	+INF or -INF (float result)
5	Floating Underflow	0.0 (float result)
7	Divide by Zero	+INF or -INF or NaN (float result)
9	Undefined Float	NAN (float result)
10	Conversion Error	0 (integer result)
11	Floating point Stack Overflow	Floating point stack underflow
12	Floating point Stack Underflow	Floating point stack overflow

INF Infinite which is the largest absolute floating point number.

NAN Not a Number, special notation for undefined floating point number.

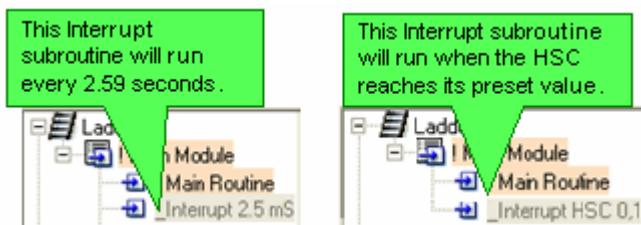
Interrupt Routines

Interrupt routines cause:

- A program to stop immediately, whenever the interrupt is activated, even if the program is in the middle of scanning a net in another subroutine.
- A jump to the Interrupt subroutine. An Interrupt subroutine must have the exact name shown in the examples below.
- When the interrupt routine is finished, the program returns to where it was interrupted, and continues from that point until the next Interrupt arrives.

Interrupt routines are generally used with Immediate elements, for example to turn an output ON in case of an alarm or emergency. To call an interrupt routine:

1. Include an Interrupt subroutine of the correct name in your program; the subroutine is executed automatically when the condition for calling it is filled.



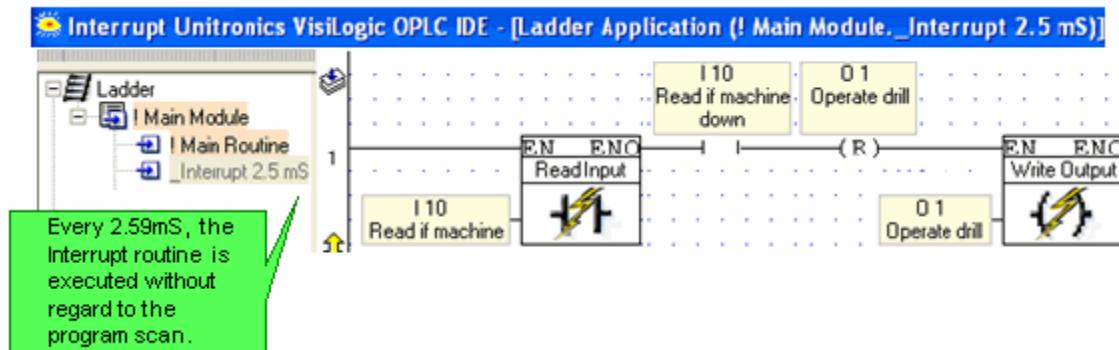
Note • If the name of the subroutine is incorrect, the subroutine will not function as an Interrupt routine.

- Interrupt features are not supported by the V120-12 series.

Sample applications showing how to use Interrupt routines in conjunction with Immediate elements may be located in :::\ProgramFiles\Unitronics\VisiLogic\Examples.

2.5 mS Interrupt Routine

This function is timed-based. The interrupt function is called by naming a subroutine _Interrupt 2.5 mS.



Including a _Interrupt 2.5 mS subroutine in the Ladder application causes:

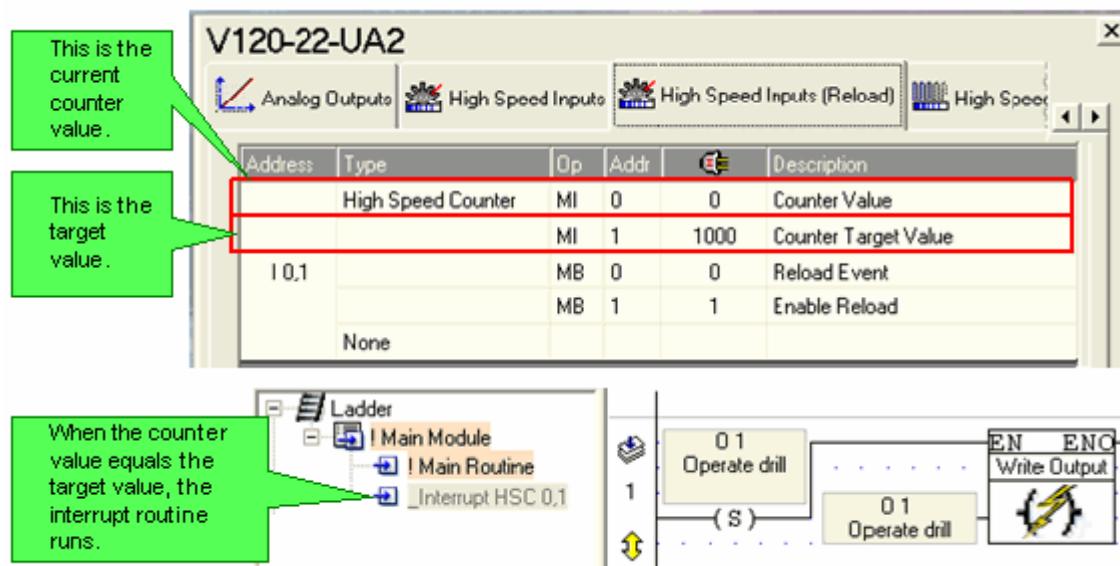
- The program scan to pause every 2.509 msec.
- A jump to the subroutine named _Interrupt 2.5 mS. Note that the interrupt routine should be as short as possible, and must not exceed approximately 0.5 msec.

When the interrupt routine is finished, the program returns to where it was interrupted, and continues from that point until the next Interrupt arrives.

Note • The Subroutine _Interrupt 2.5 mS will run for the first time after the first Ladder scan is run.

Interrupt HSC

This function is called according to the current value of a high-speed counter. The program stops immediately and executes the subroutine when the Counter Value reaches the Counter Target Value.



The interrupt function is included in the program by naming a subroutine `_Interrupt x,x` where the first x is the high-speed counter, and the second x is the reload. These subroutines must be named in accordance with your Hardware Configuration as:

- `_Interrupt HSC 0,1`
- `_Interrupt HSC 2,3`
- `_Interrupt HSC 4,5`

When the interrupt routine is finished, the program returns to where it was interrupted, and continues from that point until the next Interrupt arrives.

Index

2	
2.5.....	21, 146
A	
Add	
Add (math)	70
Add.....	70
Addressing Operands	26
Analog I/Os	
Analog Input.....	21
Configuring I/O Expansion Modules	21
Hardware Configuration	21
AND function.....	12, 57
ASCII String.....	119, 120, 122
B	
Baud	
COM port	124
Binary Numbers	65, 84
Bit	
Bit Functions	62, 107, 108, 114
Memory Bits (MB).....	28
System Bits (SB)	33
Bit.....	21
Bit.....	28
Bit.....	33
Bit.....	63
Bit.....	65
C	
Call.....	1, 8
CANbus.....	124
Casting.....	117, 144
Change Element Type	17
Coil.....	12, 17, 20, 21
COM port	124
Communications	
Initialize COM port	124, 125
Modem	124, 125
Network	127, 136, 137
Communications.....	124
Communications.....	125
Communications.....	136
Communications.....	136
Communications.....	137
Compare	12, 52, 53, 54, 112, 145
Constant Values.....	27
Contacts	12, 17, 18, 19, 22
Controller	27
Convert MB to MI	62, 104, 114
Convert MI to MB	62, 104, 114
coordinated universal time.....	85
Copying Values	62, 104, 105, 106, 107, 108, 110, 111, 112, 114, 115, 116, 117, 119
Counter	21, 32, 82, 83
Counter Values	32, 83
Create.....	144
Cut & Paste.....	16
CY	103
D	
Database, read/write	104
Dates.....	87, 89
Decimal.....	142, 144
Delete.....	16
Descriptions.....	25
Digital I/Os	21
Direct Clock function	87
Direct Coil	20
Direct Contact.....	17
Direct Month Function	100
Display text messages.....	122
Displaying Values.....	119, 122
Displays	119, 138, 140
Divide	69, 70
Double Word	33
DW	33
E	
Edit	17
Edit values	25
Elapsed	28
Element.....	12, 15, 17, 21, 22, 26
Enable Start Time	89
Equal.....	52, 56
Ethernet.....	127, 136, 137
F	
Factor.....	76
Fall edge	18
Fill vector.....	84, 104, 108
Find.....	107
Find Bit	107
Find Value	104, 107
Float functions	142, 143, 144, 145, 146
Flow.....	2
Function...1, 8, 12, 22, 26, 49, 50, 52, 56, 64, 68, 69, 83, 84, 89, 111, 115, 117, 119, 140, 142, 143, 144, 145, 146	

G	
Get Max.....	104, 115
Get Min.....	104, 116
Graphs	139
Greater Than.....	53, 54
H	
High Speed Input.....	22
High Speed Output (HSO).....	24
High-Speed Counter.....	146
HMI	
HMI keypad entries completed.....	89
HMI	89
HMI	138
HMI	139
HMI	140
Hour.....	102
I	
I/Os.....	21, 22, 24
Immediate.....	2, 21, 22, 24, 146
Indirect Clock function.....	89
Indirect Time Function.....	89, 102
Input	21, 22, 28
Integer, Constant	27
Interrupt.....	2, 21, 24, 146
Interrupt HSC	21, 146
Inverted Coil.....	20
Inverted Contact	18
J	
Jumps.....	1, 5
K	
Keypad Entry.....	89
L	
Labels	1, 5
Ladder.....	1, 2, 8, 12, 15, 49
Ladder element.....	12, 16, 17, 26
Ladder Modules.....	2
Latched	21
Less Than	54, 55
Lexical Search	26
Linearization.....	72
Link	26
List.....	25
Load Functions	82, 83, 104, 105, 138
Logic.....	57, 58, 60, 61, 62, 64, 84, 114
Loops	1, 2
M	
Math Functions.....	68, 69, 70, 71, 72, 76, 77, 78, 79, 115, 116, 142
MB-Memory Bits	25, 28
MI-Memory Integers	25, 32
ML-Memory Long Integers.....	25, 33
Modules	1, 2
Month	87, 89, 98, 100
Month Variable.....	89
Move.....	117
Multiple Input Values.....	69
Multiply	69, 71
N	
Negative Transition Contact.....	18
Nets.....	1, 11, 15, 17, 20, 25, 49
Network	25, 136, 137
NI.....	25, 48
Not Equal.....	56
NSB-Network System Bit.....	48
NSI-Network System Integer	25, 48
O	
OFF.....	17, 18, 20, 21, 28
ON	17, 18, 20, 21, 28
O-Output.....	28
Operand	12, 25, 26, 27, 33, 48
Operand Address	27
Operand types	25, 33
OR	58
Outputs	24, 28
P	
Placing function blocks	49, 68
Port	124
Positive Transition Contact.....	19
Power	78
Power-up Values.....	27, 33
Preset Value.....	28
Program Flow	2, 5
Program Sequence	2, 5, 21
R	
Rails.....	1
Reload.....	21
Reset coil	21
RFC 1305.....	85
Rotate	61
RS232 Parameters.....	124
RTC Real-Time-Clock	85, 103, 122
Rung	1
S	
SB-System Bit	25, 33
Scan	2
SDW	33
SDW-System Double Word	25

Select	25, 26	Trig functions	143
Shift	60, 117	Triggering signal.....	18, 19, 21
Signed.....	27, 33	Troubleshooting.....	146
SI-System Integer	25, 33	U	
SL-System Long Integer.....	25, 33	Unlatch	21
Snap-in I/O Module.....	28	Unsigned.....	25, 27, 33
Socket.....	137	UTC	85
Square root	79	utility	146
Store Functions. 12, 80, 81, 82, 83, 84, 104, 106, 144		V	
String.....	119, 122	Values	25, 27
Subroutine		Variable Types.....	122
Return	8	Variables.....	89, 140
Subroutines.....	1, 2, 8, 10	Vector Copy.....	104, 110
Subtract function	69, 72	Vector operations....	72, 83, 105, 106, 107, 108, 110,
Symbols.....	25, 48	111, 112, 115, 117, 119, 120	
System Operands.....	33	View Window.....	25, 27, 28, 32, 33
T		W	
TCP/IP	136, 137	Week.....	87, 99
Text Variable.....	122	X	
Time	119	XOR.....	59
Time function block	102	Y	
Timers.....	28, 82, 83, 115	Year	103
Toggle.....	21		

