SOFTWARE MANUAL

VISILOGIC: LADDER PROGRAMMING

V230-21-G23 Rev: 3:00

UNITRONICS®

Table of Contents

VisiLogic: Ladder Programming

Ladder Editor

Use the VisiLogic 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.

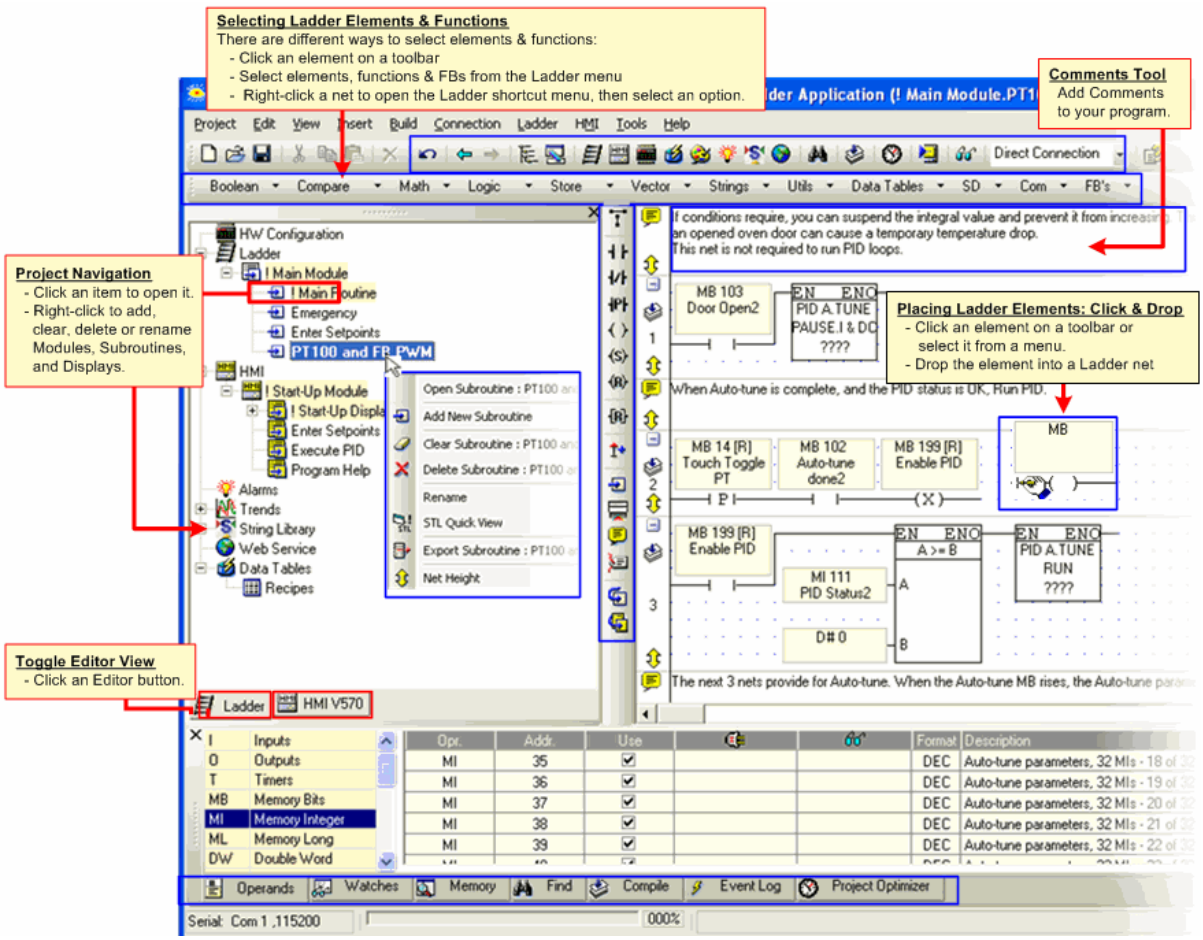
Use the Ladder Editor to:

- 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.


Selecting Ladder Elements & Functions
There are different ways to select elements & functions:
- Click an element on a toolbar
- Select elements, functions & FBs from the Ladder menu
- Right-click a net to open the Ladder shortcut menu, then select an option.
Comments Tool
Add Comments to your program.
Project Navigation
- Click an item to open it.
- Right-click to add, clear, delete or rename Modules, Subroutines, and Displays.
Placing Ladder Elements: Click & Drop
- Click an element on a toolbar or select it from a menu.
- Drop the element into a Ladder net
Toggle Editor View
- Click an Editor button.
Project Edit View Insert Build Connection Ladder HMI Tools Help
Direct Connection
Boolean Compare Math Logic Store Vector Strings Utils Data Tables SD Com FB's
HW Configuration
Ladder
! Main Module
! Main Routine
Emergency
Enter Setpoints
PT100 and FB_PWM
HMI
! Start-Up Module
Enter Setpoints
Execute PID
Program Help
Alarms
Trends
String Library
Web Service
Data Tables
Recipes
Open Subroutine : PT100
Add New Subroutine
Clear Subroutine : PT100
Delete Subroutine : PT100
Rename
STL Quick View
Export Subroutine : PT100
Net Height
MB 103 Door Open2
PID A.TUNE PAUSE.I & DC ????
MB 14 [R] Touch Toggle PT
MB 102 Auto-tune done2
MB 199 [R] Enable PID
MB
MI 111 PID Status2
D# 0
A >= B
PID A.TUNE RUN ????
Ladder
HMI V570
Inputs
Outputs
Timers
Memory Bits
Memory Integer
Memory Long
Double Word
Operands Watches Memory Find Compile Event Log Project Optimizer
Serial: Com 1 ,115200

Ladder Logic

You use Ladder Logic to write your project application. Ladder is based on Boolean principals and follows IEC 1131-3 conventions.

Ladder Diagrams are composed of different types of contact, coil and function block elements. These elements are placed in nets.

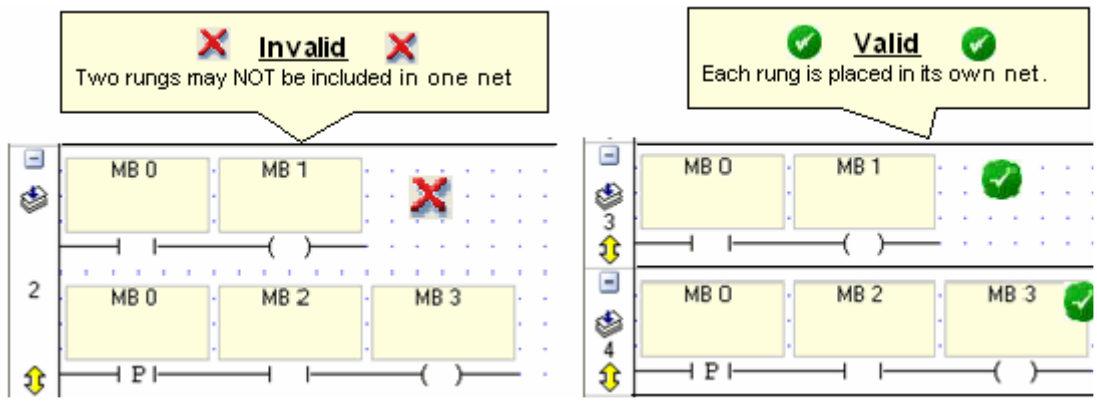
In any Ladder Diagram, the contacts represent input conditions. They lead power from the left rail to the right rail. Coils represent output instructions. In order for output coils to be activated, the logical state of the contacts must allow the power to flow through the net to the coil.

Ladder Net

A Ladder net is the smallest division of a ladder diagram.

The Ladder diagram contains a left and right rail. Between these rails, the control application is arranged in nets. A net contains a row of Ladder elements that drive a coil.

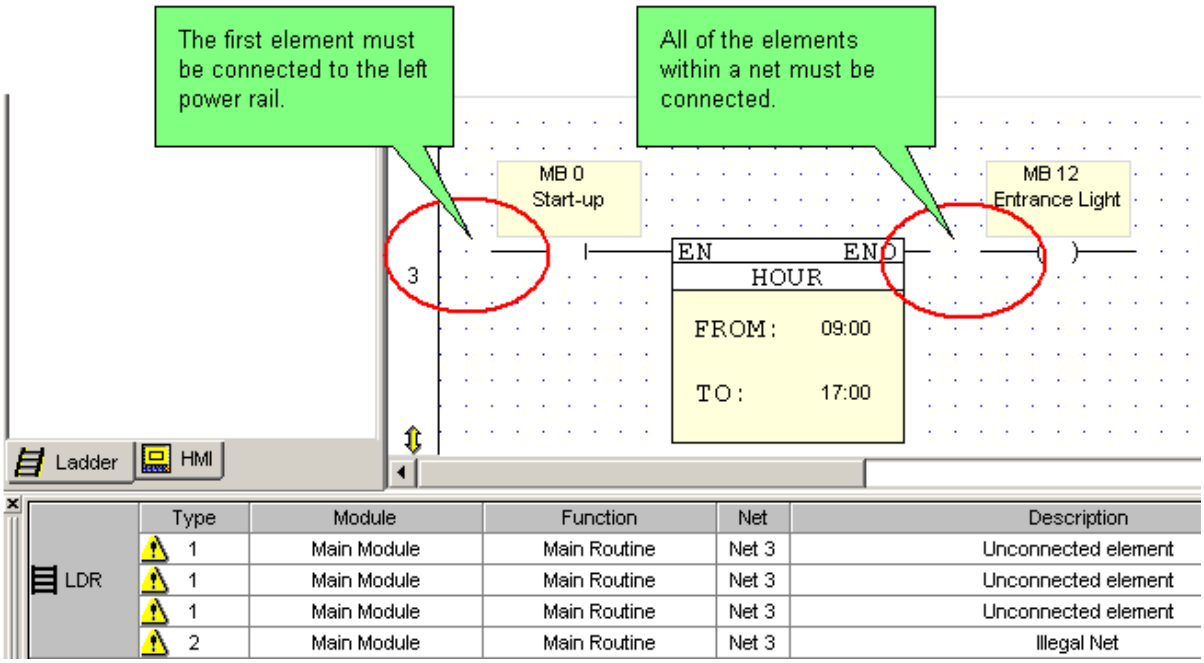
Each net must contain only **one** rung.



Power flows through the ladder elements in a net from left to right.

This is why the first ladder element in the net **must touch the left Ladder rail**. All of the elements in a net must be connected to allow power flow. You do **not** need to connect the last element on the right to the right side of the ladder in each net.

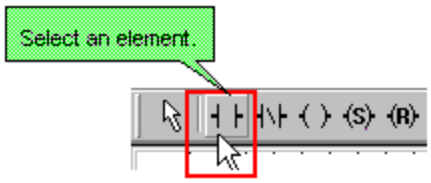
If the elements in a net are not connected, the software displays an error message at compilation.



Placing a Ladder Element in a Net

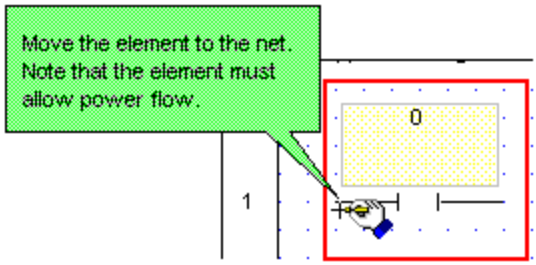
1. Select any type of Ladder element by:

- Clicking its icon on the Ladder toolbar, -or-

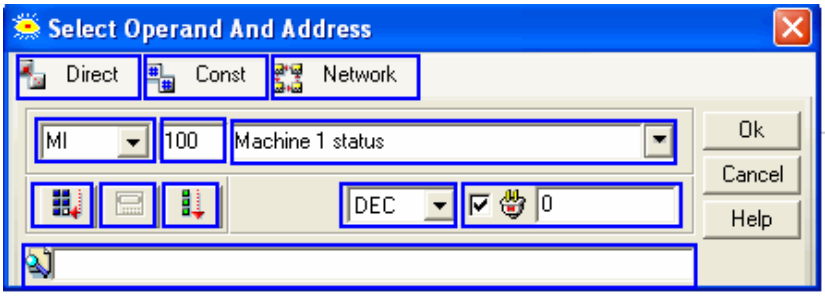


- Selecting it from the Ladder menu, -or-
- Right-clicking on the Ladder to display the Ladder menu and then selecting the element.

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



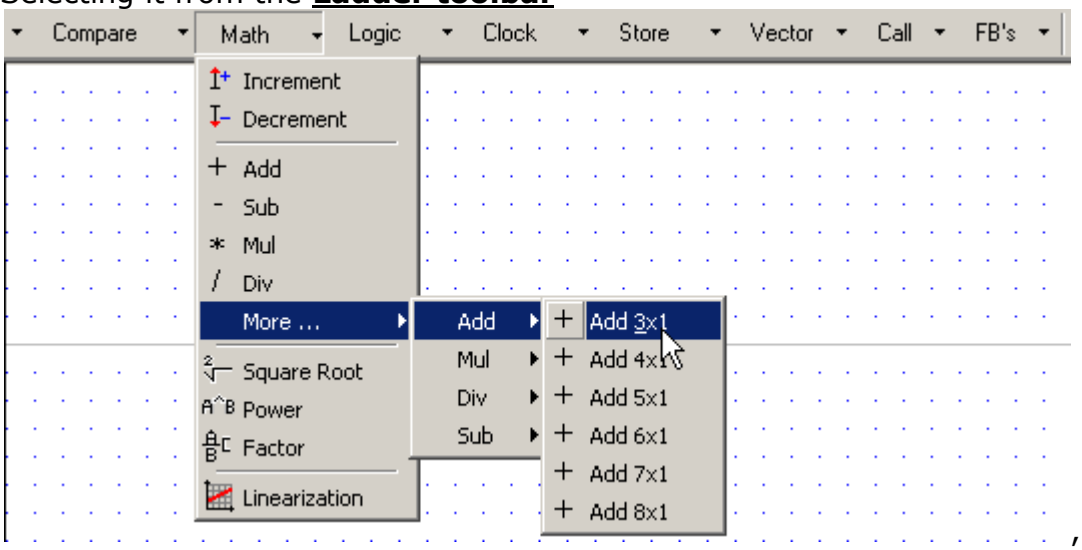
3. Link operands using the Select Operand and Address dialog box shown below.



Placing a Function in a Net

1. Select any type of Ladder function by:

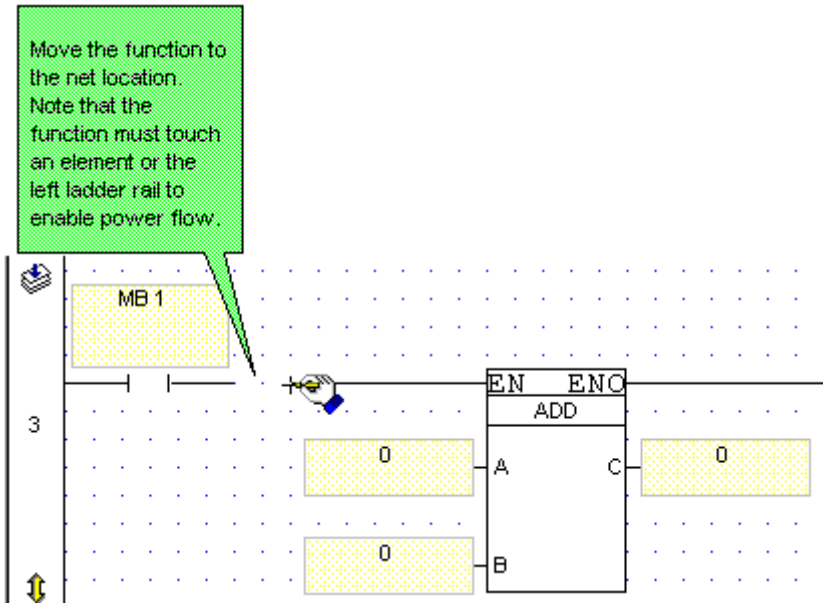
- Selecting it from the **Ladder toolbar**



-or-

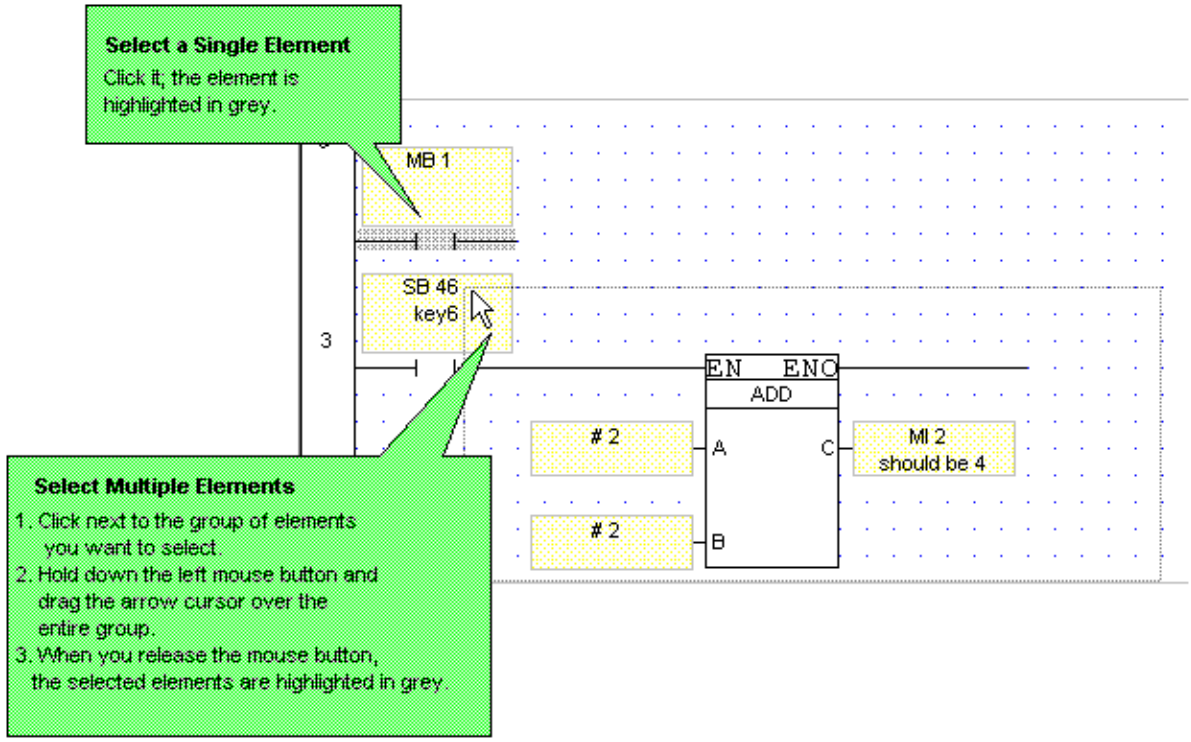
- Selecting it from the Ladder menu, -or-
- Right-clicking on the Ladder to display the Ladder menu and then selecting the function.

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

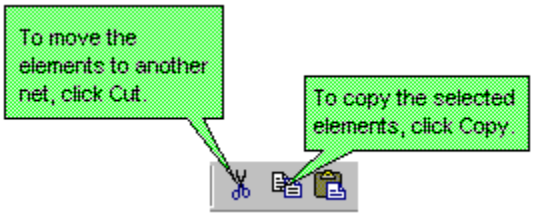


Delete Elements

Select the desired element(s), then



- Select Cut. or Copy from the Edit menu.



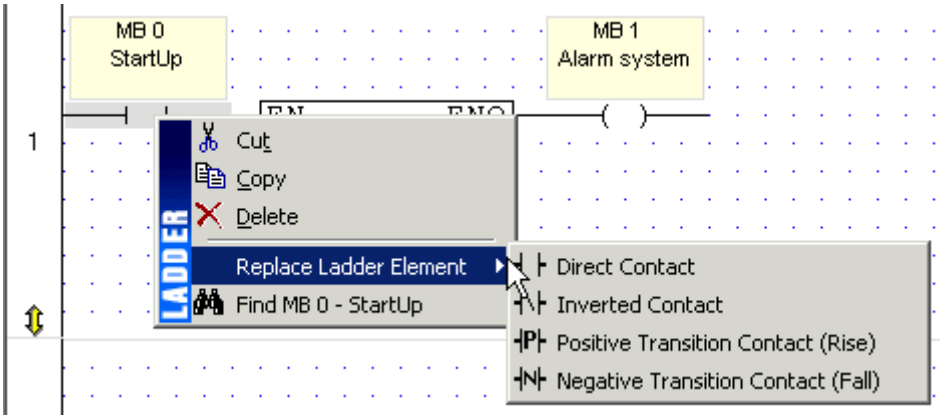
-or-

- 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.



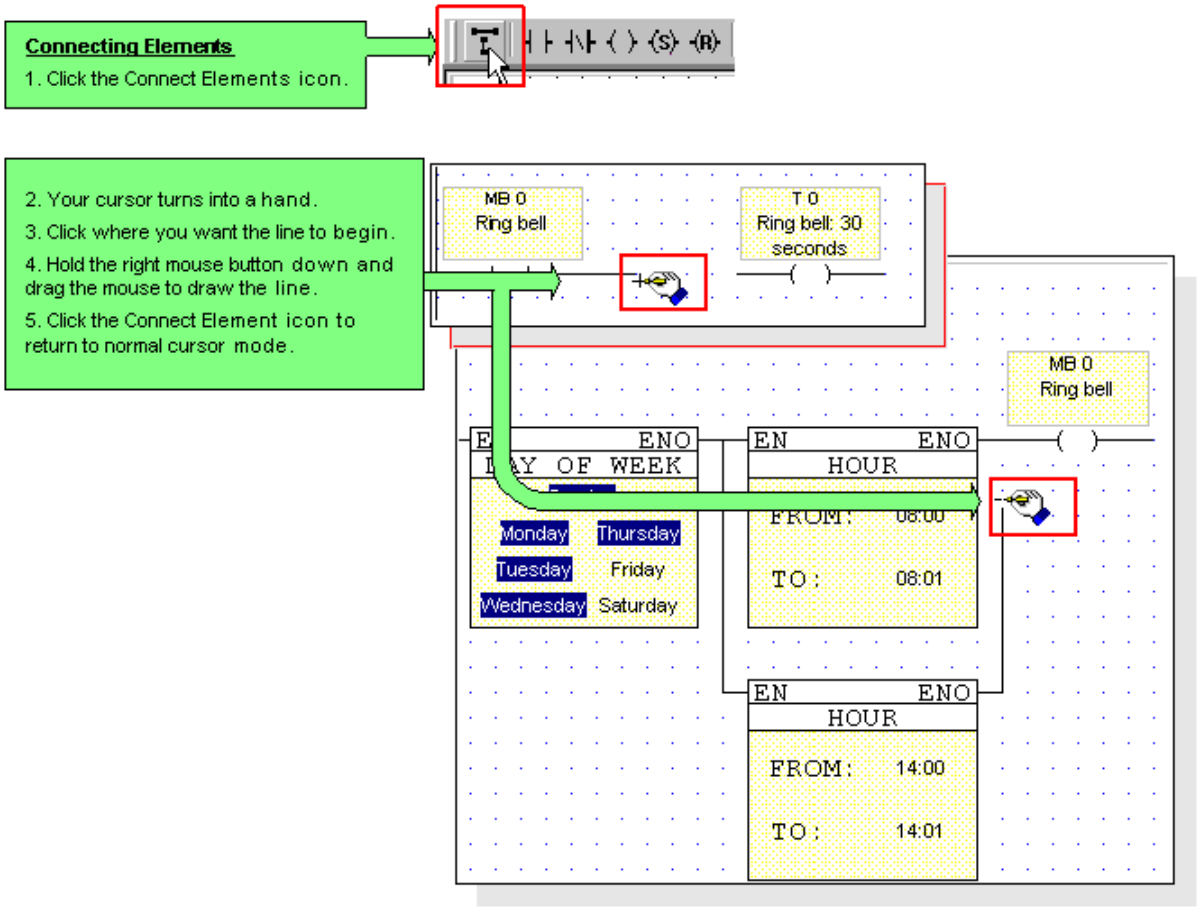
After the element has been changed, it remains linked to the same operand.

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.

Connecting Ladder Elements and Functions

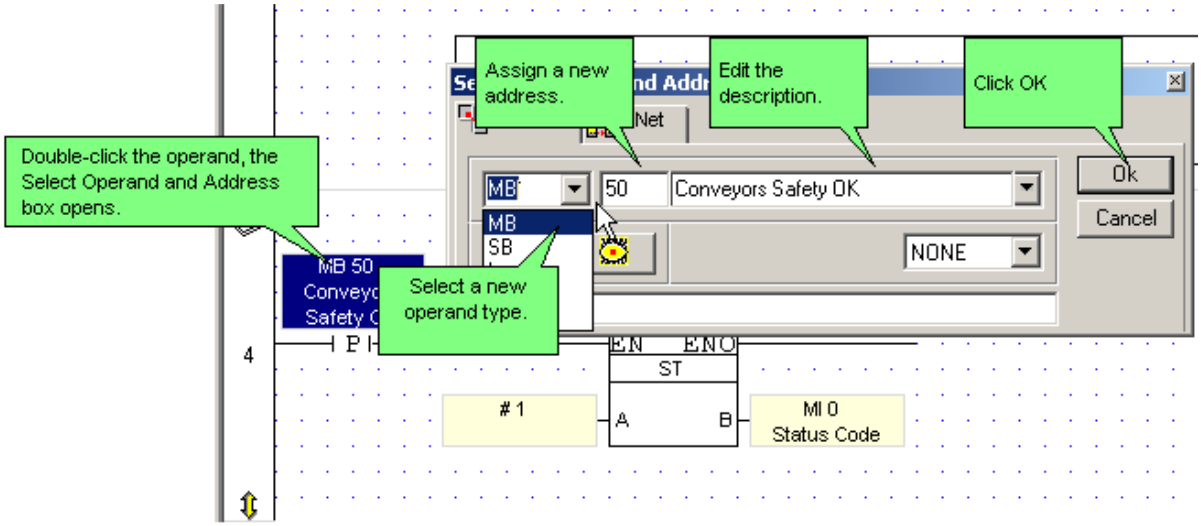
Use the Connect Elements tool to connect two or more elements or functions in a net. All net elements must be connected in order to allow power to flow through the net. If they are not connected you will not be able to compile your application.

Connecting Elements



Changing an Element's Operand

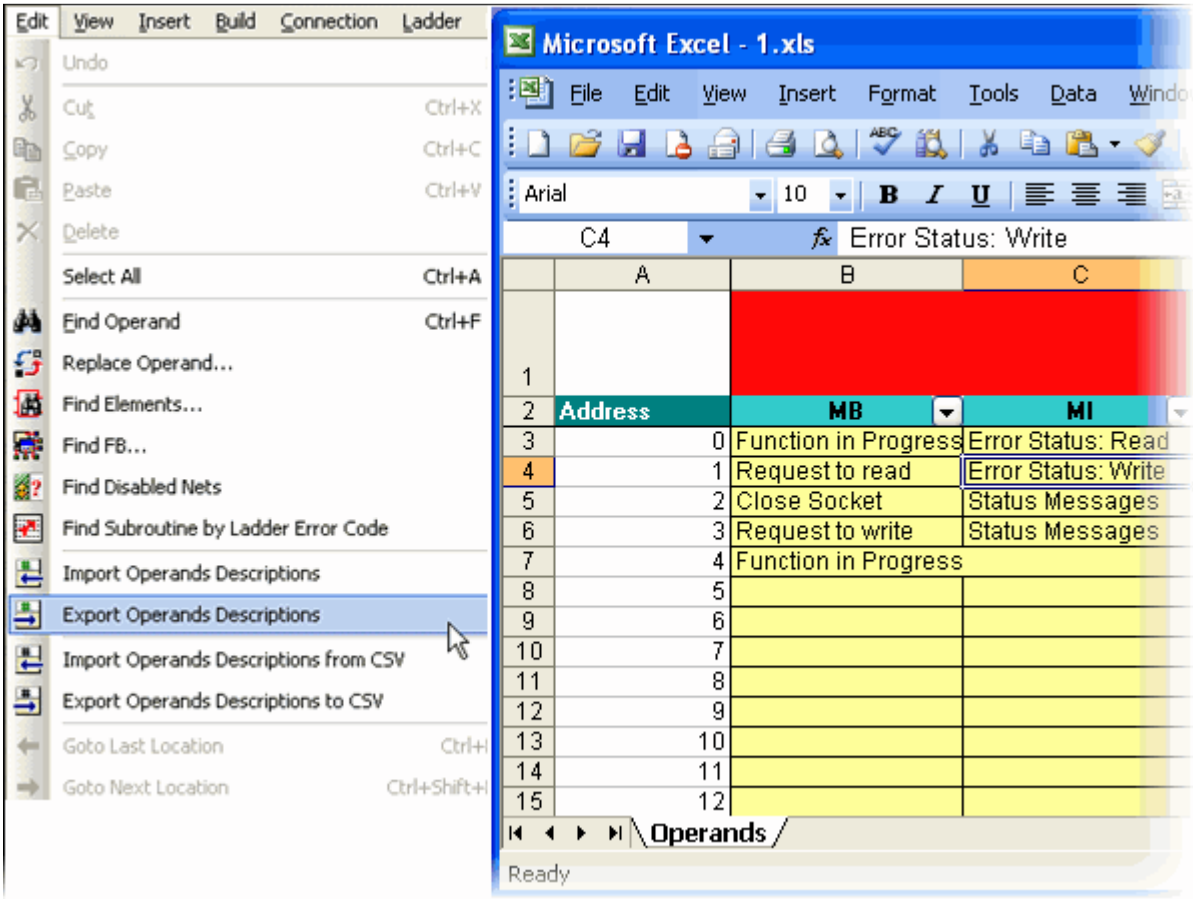
To edit an element's operand:



The element appears on the net with the new Operand, Address and symbol.

Import-Export Operand Descriptions

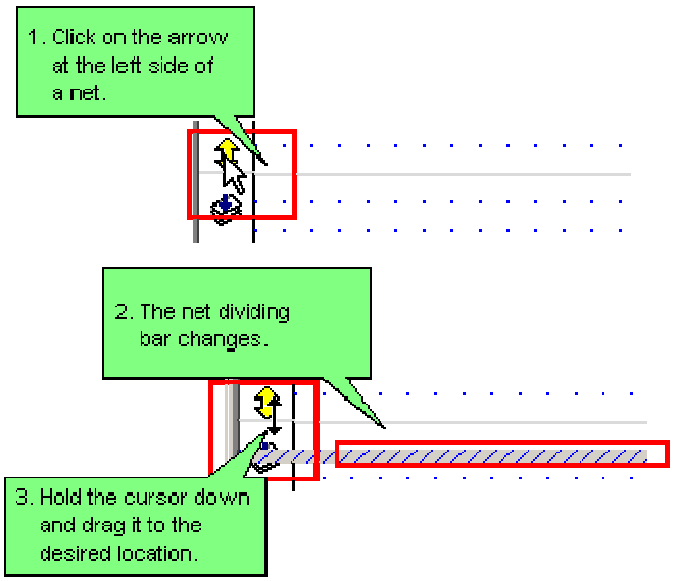
You can export operand descriptions to Excel or other .csv editor, edit them, then import them back into VisiLogic via the Import-Export Operands Description on the Edit menu.



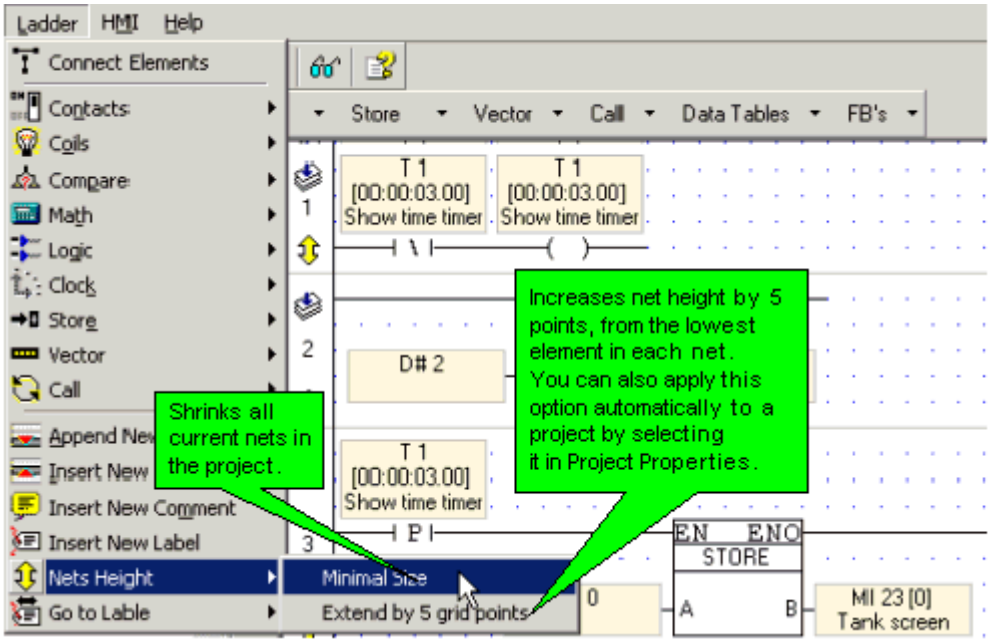
Nets: Sizing and Resizing

To shrink a net to its minimum height, double-click the net's left-hand rail.

Nets can also be manually resized.

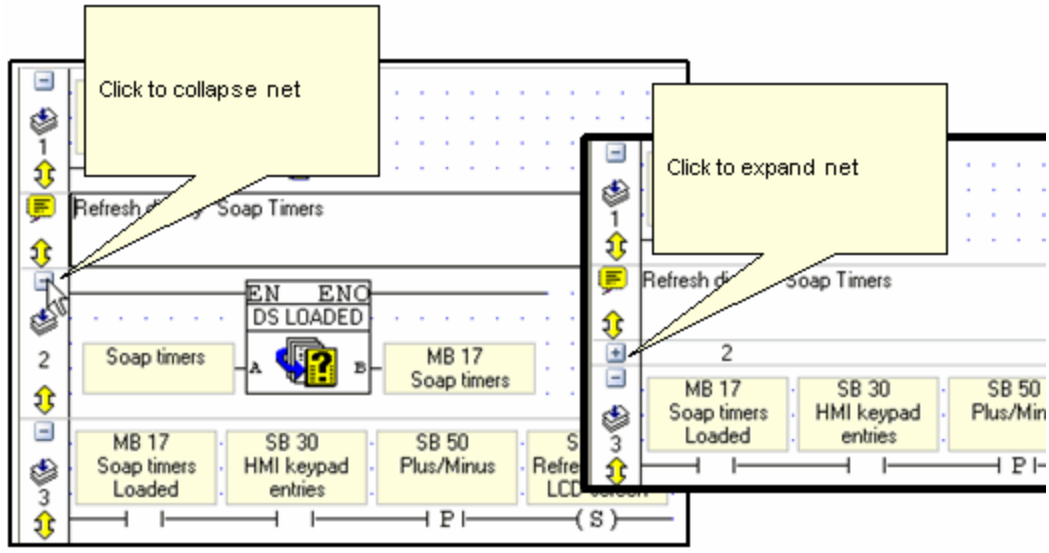


The Ladder menu contains two options that enable you to resize nets throughout a project.



Collapse, Expand Nets

You can collapse and expand individual nets by clicking the button in the upper left corner of the net. To expand all nets at once, click the Ladder menu and select Expand All Nets.



Adding and Inserting Nets

To add a net to the bottom of your Ladder:

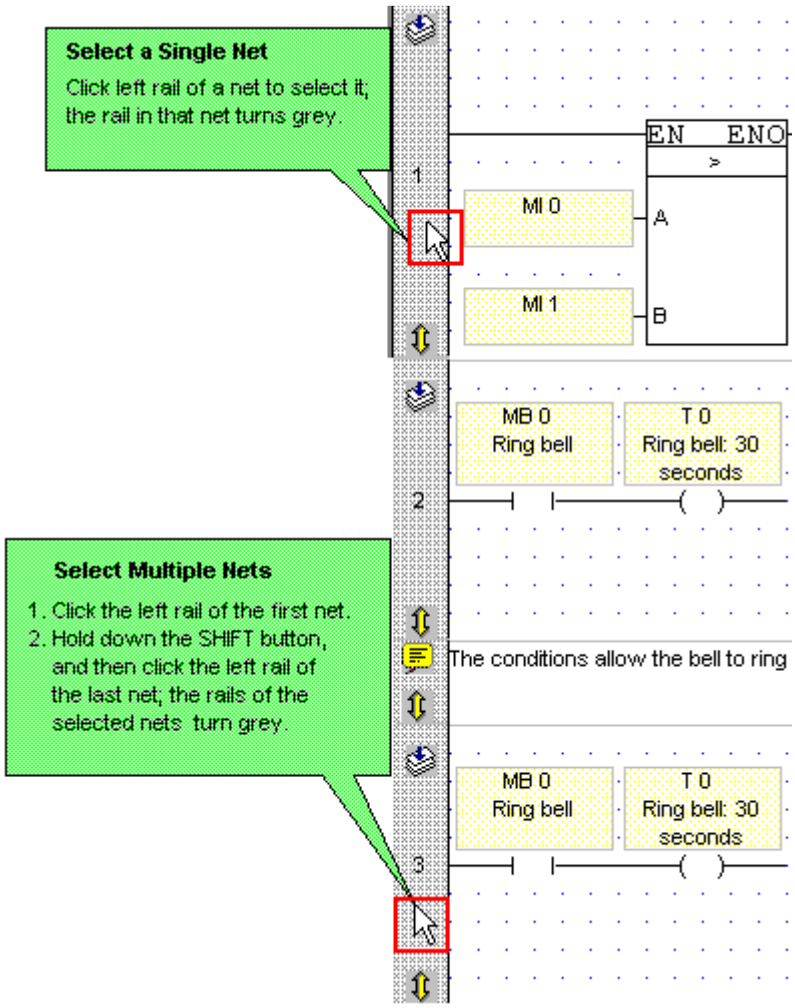
- Select the Append Nets icon from the Insert menu;three nets are added to the bottom of the Ladder application. .

To insert a Ladder net:

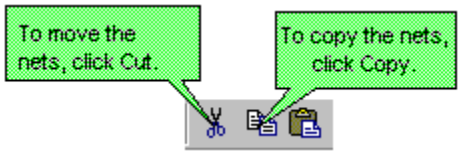
1. On the Ladder toolbar, click on the Insert Net icon ; your cursor changes into **cross-hairs** .
2. Click on a net; the new net is inserted above the net you clicked on.

Move, Copy, & Paste Nets

1. Select the desired net(s).

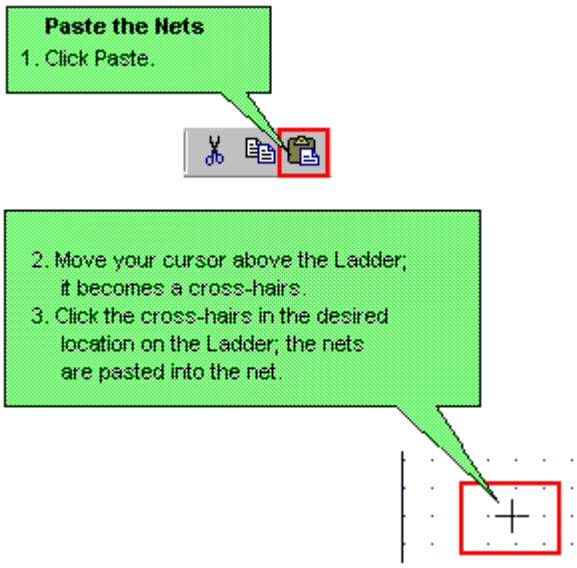


2. Select the desired operation.



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

3. Place the elements in the net.

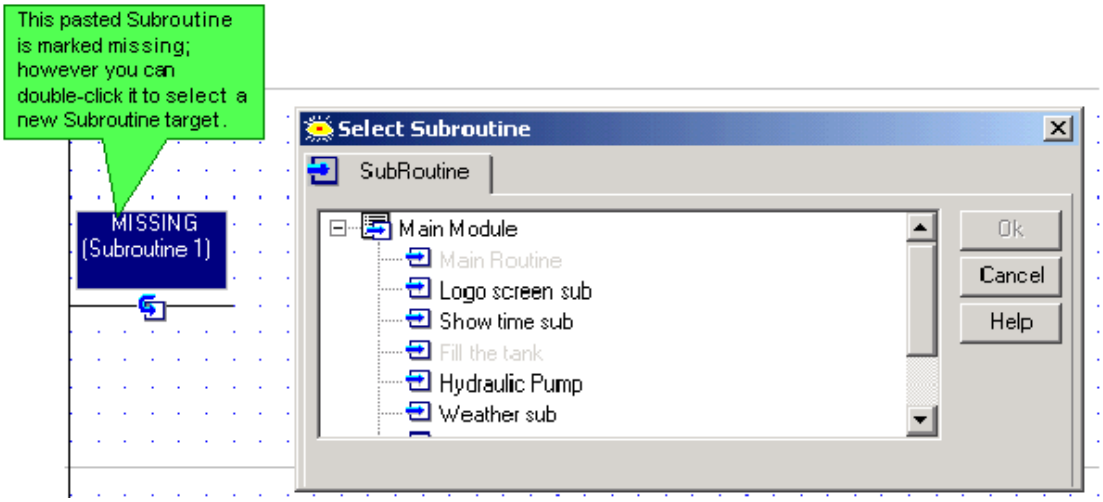


-or-
Select Paste from the Edit menu.

You can also cut, copy and paste nets **between projects**, subject to the information listed below:

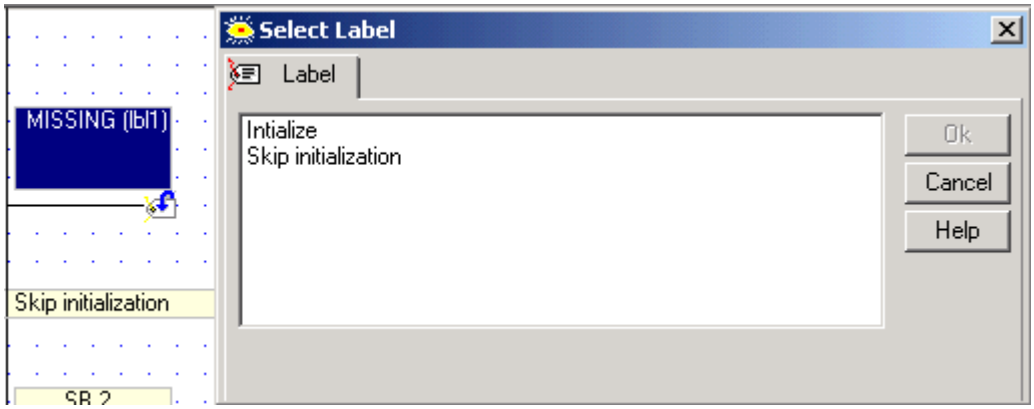
- Once you have cut or copied your selection from the source project, open a target project without closing VisiLogic, either by using the New Project or Open project buttons or via these options on the Project menu. If you close VisiLogic, the selection will be lost.

- If the source project contains Call Subroutine or Load HMI operations, note that the referenced elements will be marked as **missing**, even if the target project contains elements of the same name. Note that you can reassign the references.

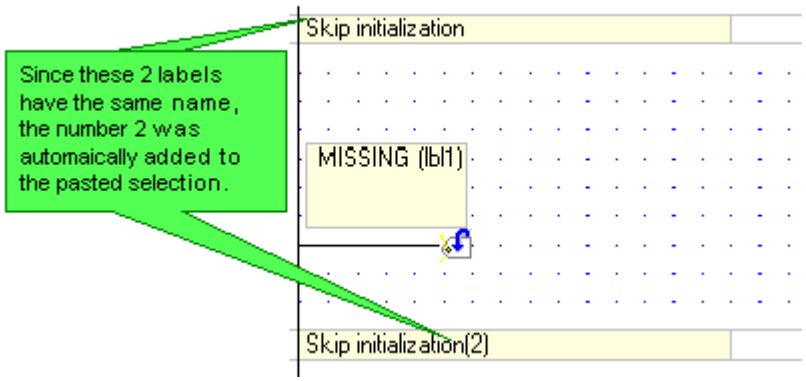


- If the selection contains FBs, and no FBs of that type currently exist in the target project, the pasted FBs will be the version currently in VisiLogic FB library--in other words, if the source selection contains older FB versions, they are automatically updated during the Paste operation.

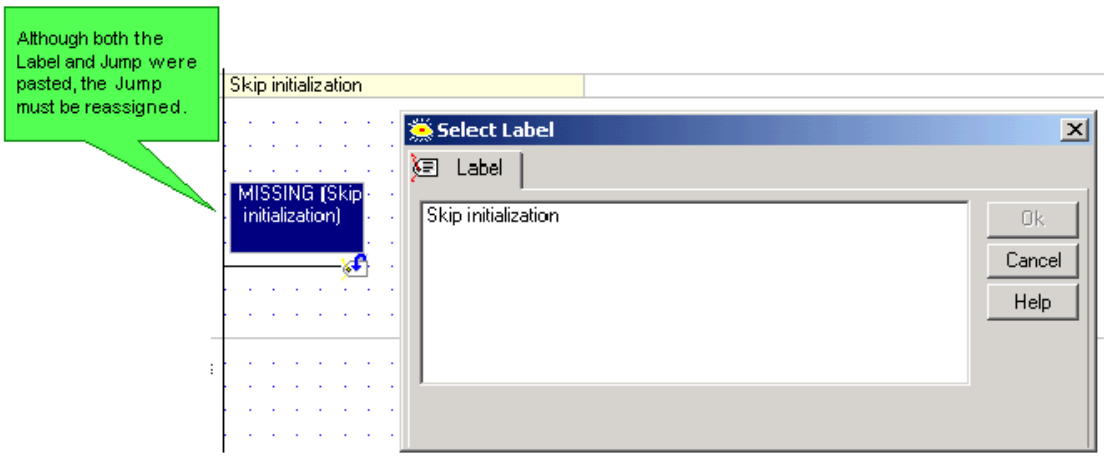
- If the selection contains FBs, and FBs of that type currently exist in the target project in a **different** version, Paste cannot be completed.
- If your selection contains only Labels, without the attendant Jump to Label, they will be marked as **missing**, even if the target project contains Jumps of the same name. Note that you can reassign the references.



- If the selection contains Labels or Jumps with the same name as those in the target project, these will be automatically renamed by the program when they are pasted.



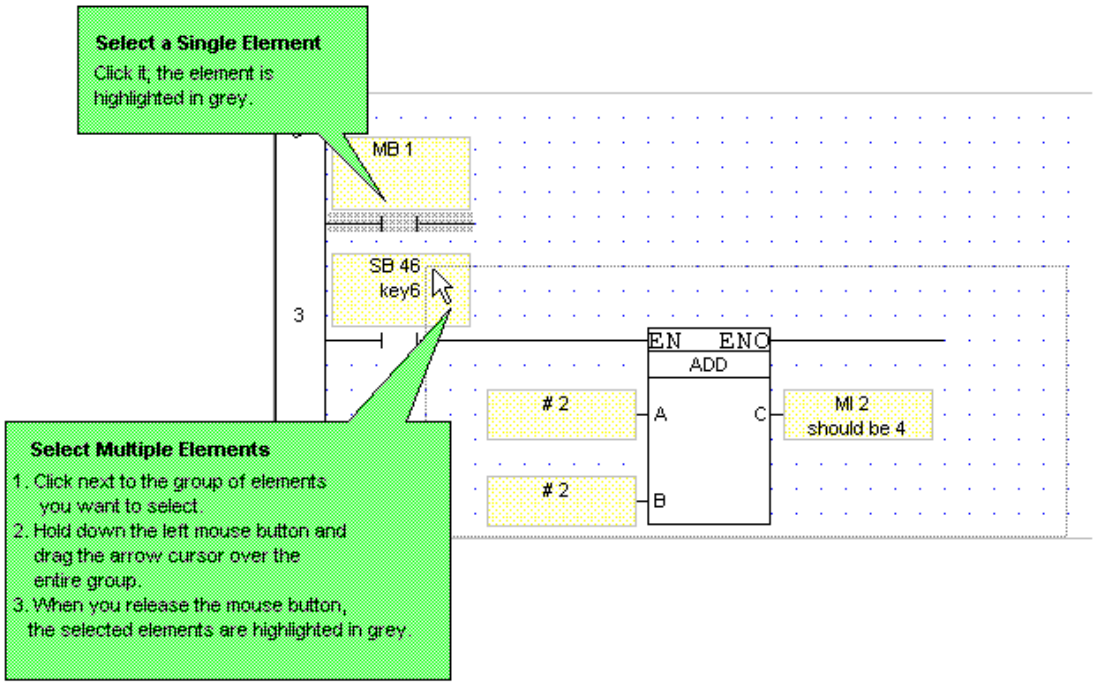
- If you copy both Labels and Jump to Label, the Jumps will be marked as missing. Note that you can reassign the references.



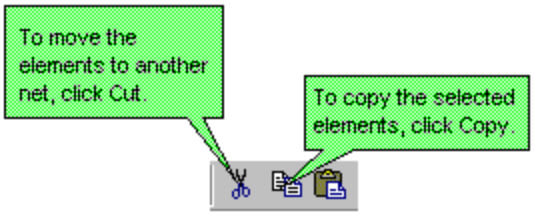
Move, Copy & Paste Elements

Ladder elements and functions may also be dragged and dropped between nets.

1. Select the desired element(s).



2. Select the desired function.



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

3. Place the elements in the net.

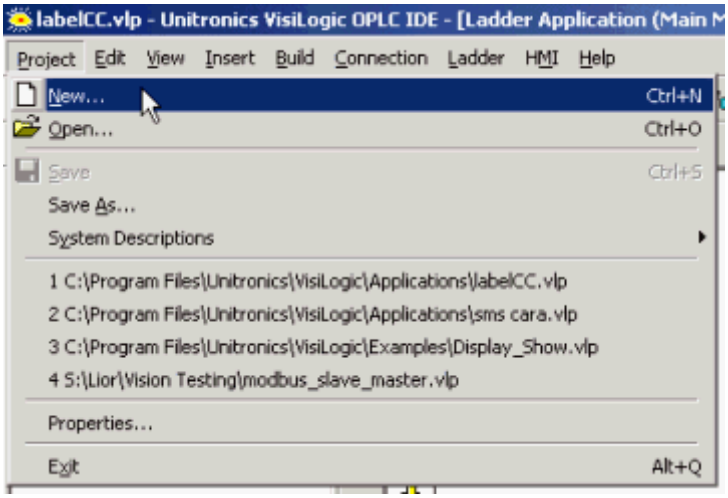
-or-
Select Paste from the Edit menu.

Note that when you paste elements into a net, the elements paste into the same relative location in the new net. The elements 'remember' their original net location. Therefore, before you paste elements into a net that already contains elements, move any elements that occupy the same position as the paste selection.

Move, Copy, & Paste between Projects

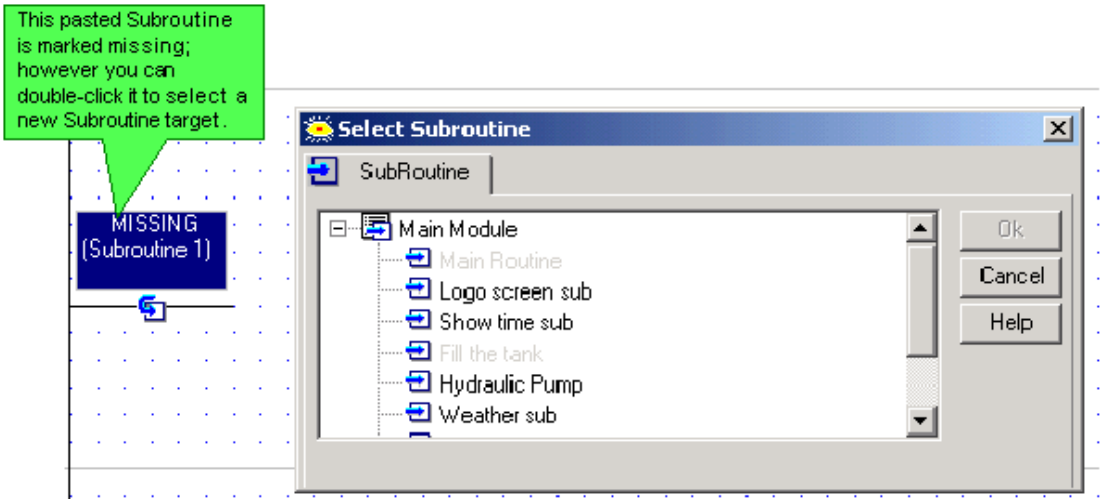
You can cut, copy and paste both HMI Displays and Ladder nets **between projects**, subject to the information listed below.

Once you have cut or copied your selection from the source project, open a target project without closing VisiLogic, either by using the New Project or Open project buttons or via these options on the Project menu. If you close VisiLogic, the selection will be lost.

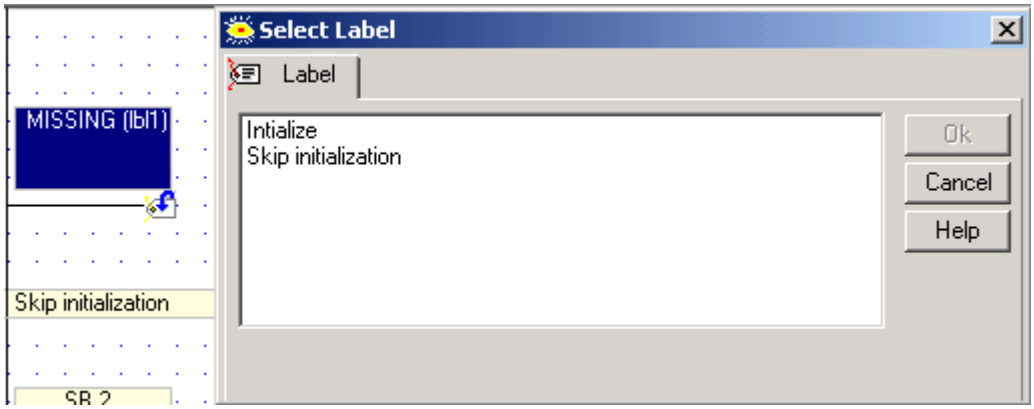


Ladder

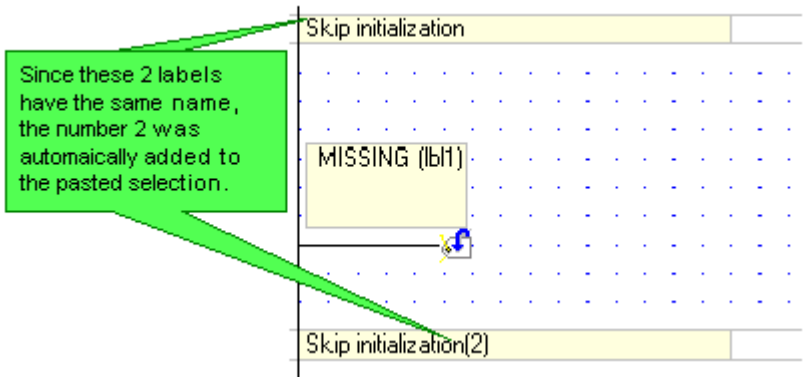
- If the source project contains Call Subroutine or Load HMI operations, note that the referenced elements will be marked as **missing**, even if the target project contains elements of the same name. Note that you can reassign the references.



- If the selection contains FBs, and no FBs of that type currently exist in the target project, the pasted FBs will be the version currently in VisiLogic FB library--in other words, if the source selection contains older FB versions, they are automatically updated during the Paste operation.
- If the selection contains FBs, and FBs of that type currently exist in the target project in a **different** version, Paste cannot be completed.
- If your selection contains only Jumps, without the attendant Labels, they will be marked as **missing**, even if the target project contains Labels of the same name. Note that you can reassign the references.



- If the selection contains Jumps and Labels with the same name as those in the target project, the Jump, Label and link between them will be automatically recreated by VisiLogic when they are pasted.



In this way, VisiLogic maintains the integrity of the links between Jumps and their corresponding Labels.

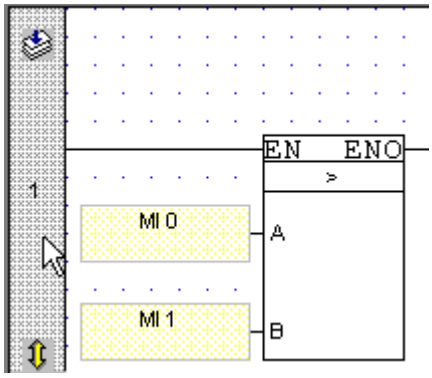
Display elements

- When you paste elements into a Display, the elements paste into the same relative area in the new net. The elements 'remember' their original location. Therefore, before you paste elements into a Display that already contains elements, move any elements that occupy the same position as the Paste selection.
- If you paste variables that are linked to named constant values, note that the constant's description is lost during the paste operation.
- Variables do not retain their descriptions; they are renamed as Variable 1, Variable 2, etc..

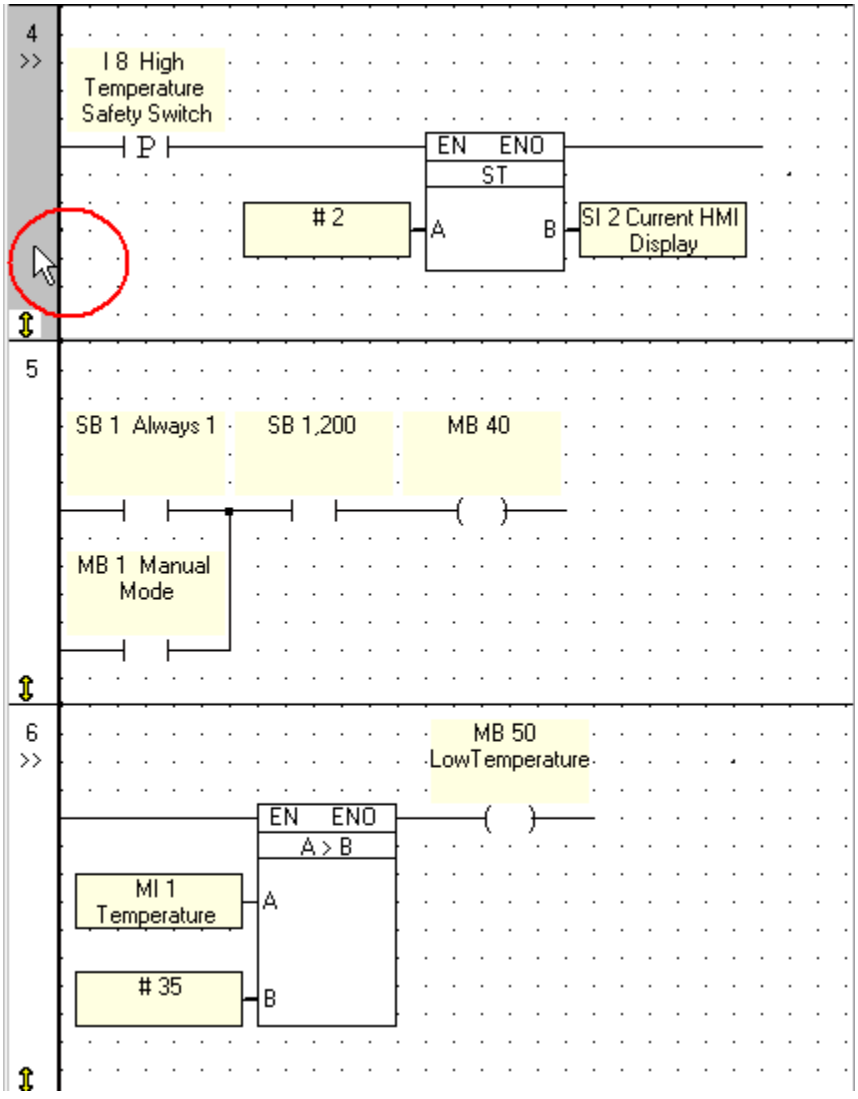
Deleting Nets

Select the desired nets.

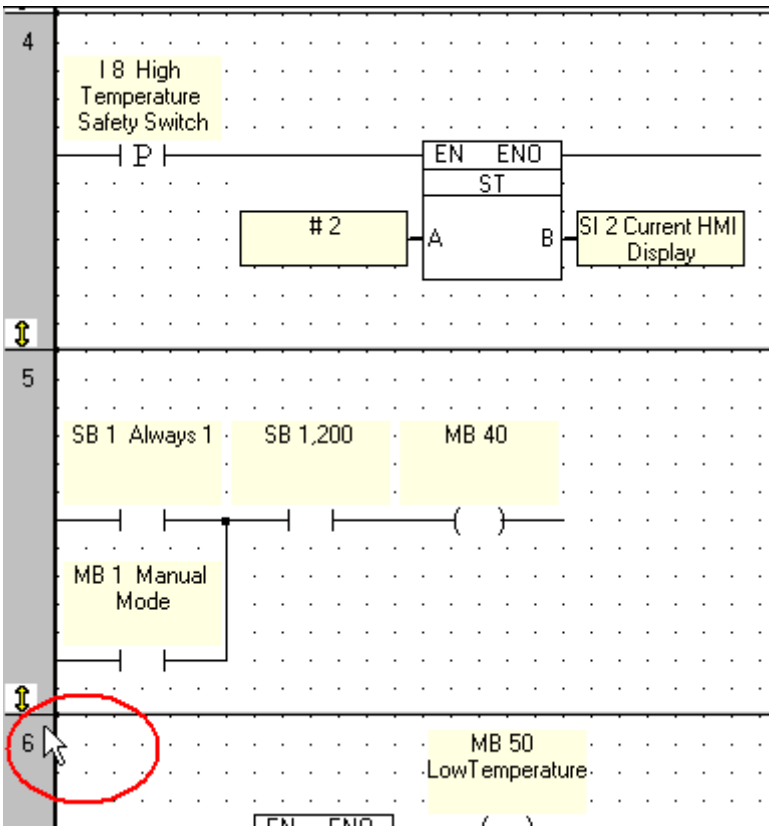
- To select one net , click on the left rail of a net to select it; the rail in that net turns grey.



1. To select more than one net, select the first net by clicking on the left net bar.



2. Hold the **Shift** button and click on the last net in the range that you want to delete.



3. Press the Delete button on your computer keyboard; the net is deleted and all of the nets in your project move up.

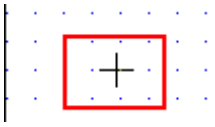
Comments Tool

Ladder Editor Comments enable you to place remarks above program nets. Comments can be written directly into the Comment pane, or written in Notepad and pasted into the pane.

Comments are not downloaded to the controller. To toggle Comments in and out of view, press **<Alt> + <C>**, or select the option from the View menu.

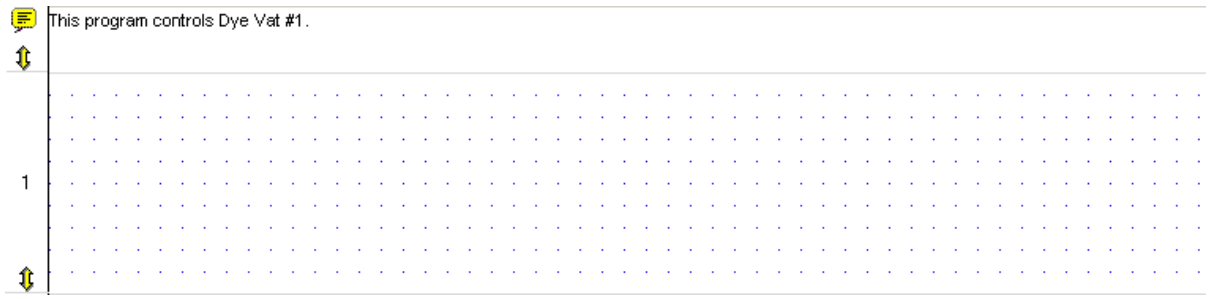
Insert a comment:

1. Click on the Comment icon  ; your cursor changes into a **cross-hairs**



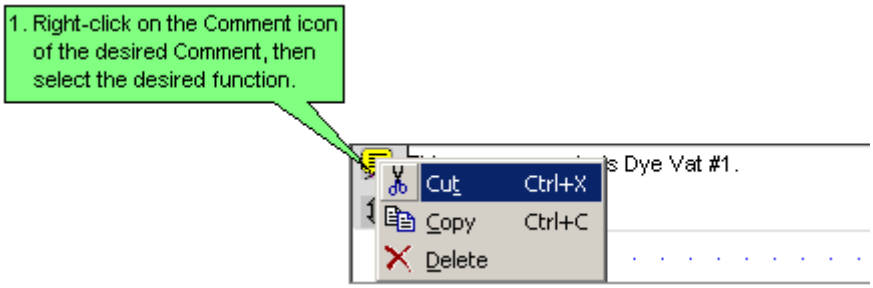
-or-
Select Insert Comment from either the Insert or Ladder menu.
-or-
Right-click on the Ladder, and then select Insert Comment.

2. Click on a net; a Comment field opens in the net you clicked.
3. Type text in the field.

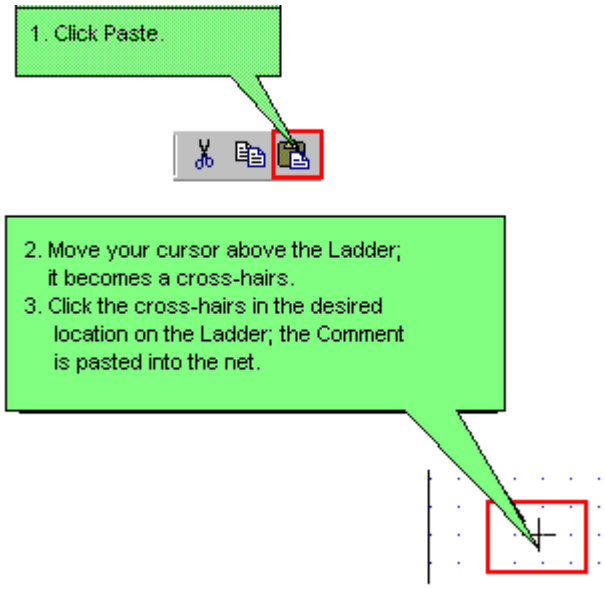


Move, Copy, and Paste Comments:

4. Select the Comment.



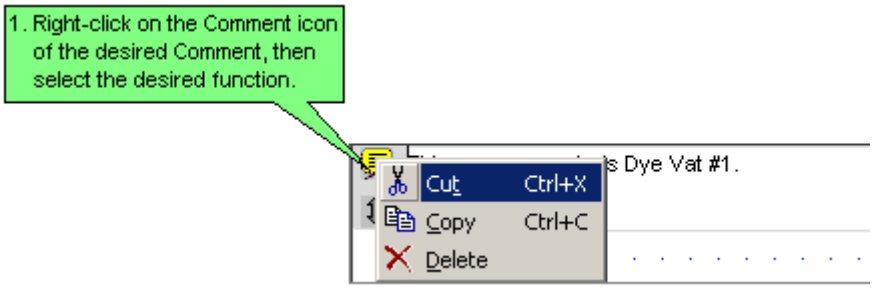
5. Place the Comment in the net.



-or-
Select Paste from the Edit menu.

Delete a Comment

1. Select the Comment.

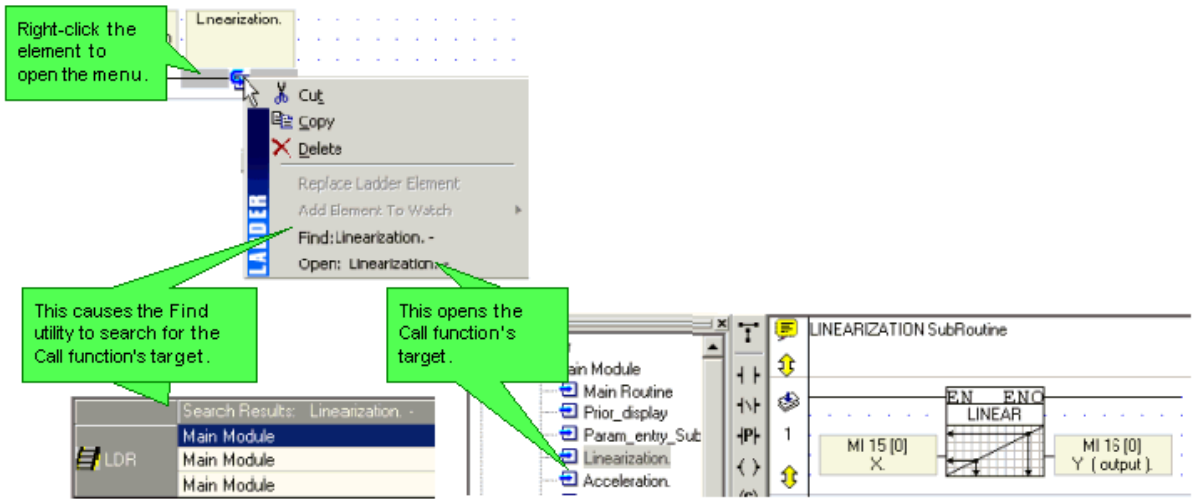


2. Select Delete.
   -or-
   Press the Delete button on your PC's keyboard.

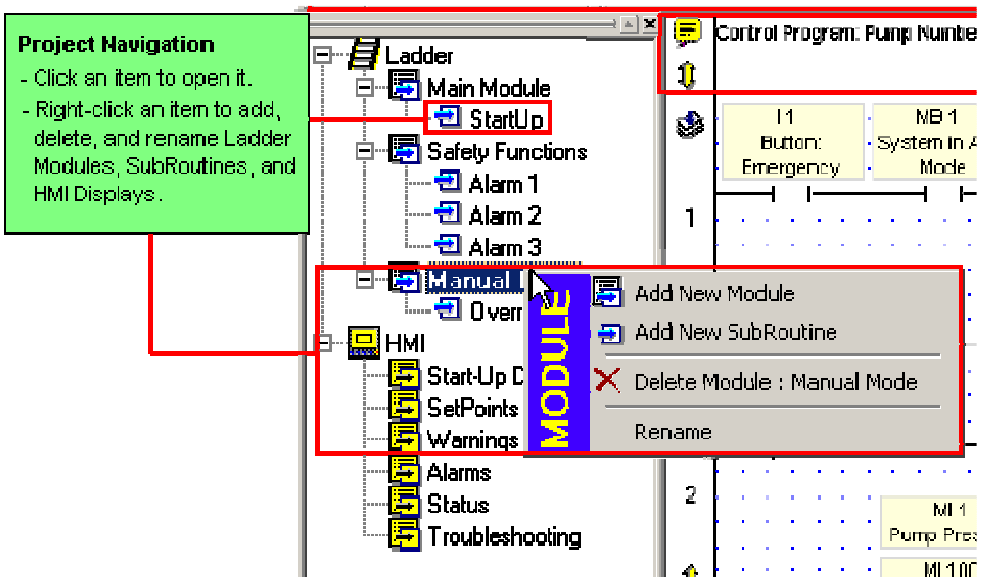
Open a Subroutine

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.

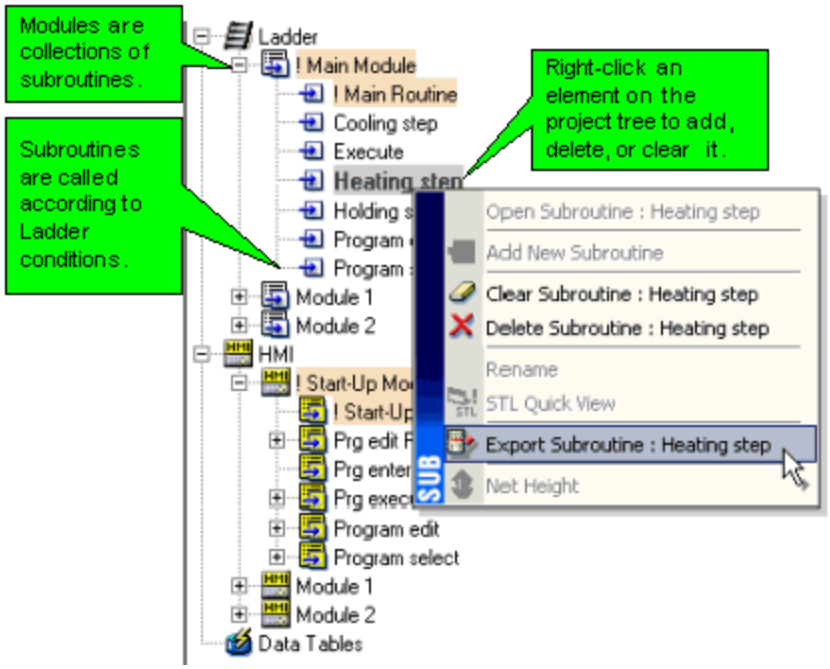


Name-Rename Modules and Subroutines



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.

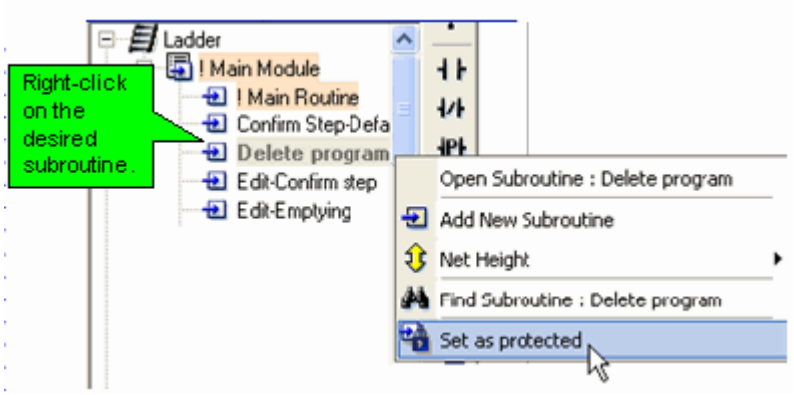
Protecting Subroutines

You can create a Ladder Password, then apply it to protect multiple subroutines and hide their content. When a subroutine is protected, a user cannot export/import it. In addition, the user cannot open, copy, or print it without supplying the password.

Creating and Using a Password

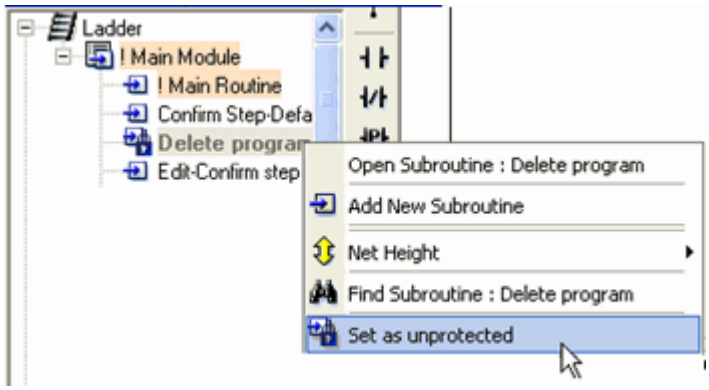
1. To create a password, select File>Set Ladder Password; then fill in the password field.

2. To apply the password to a subroutine, right-click the subroutine's name in the Project Navigation window, then select Set as Protected; a small padlock icon is displayed nest to the subroutine's name.
   You can also right-click a module's name and select Protect All Subroutines in Module.

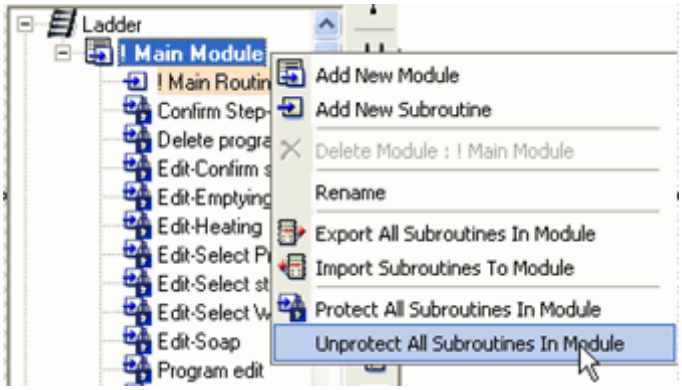


Note • Protection is applied **after** VisiLogic (not just the project) is closed and reopened.

3. To remove protection from a subroutine, right-click the protected subroutine's name, then select Set AS Unprotected; the padlock icon disappears.



You can remove protection from a module in the same way.



Note • The same password may be used for different projects.

Deleting a Ladder Password

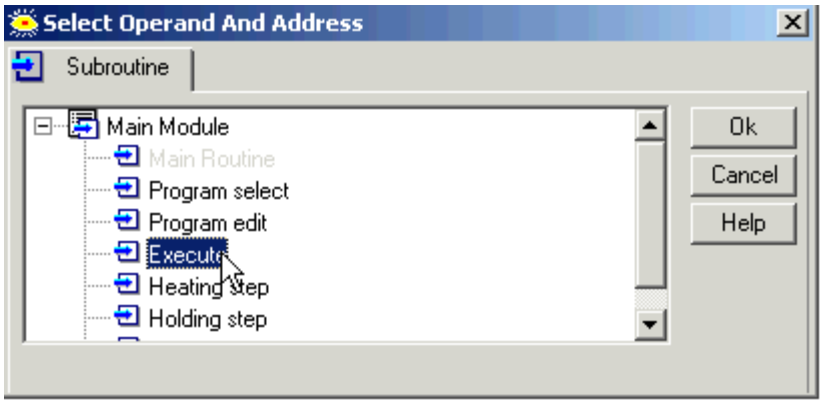
1. To delete a Ladder password from a project, select File>Unset Ladder Password.

Import/Export Subroutines

You can export Subroutines and save them as .vlx files, then import them into other projects. You can import/export single Subroutines, or all of the subroutines in a Module. Note that you cannot export Subroutines from the Main Module.

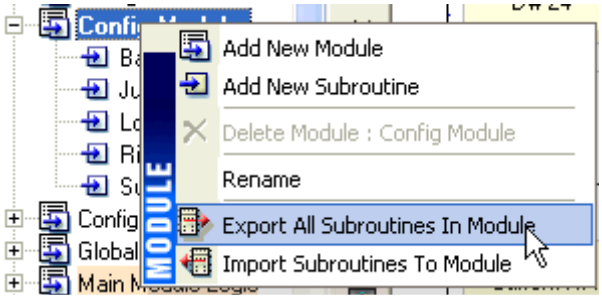
Exporting a single Subroutine

1. Right-click the desired Subroutine and select Export Subroutine,
   -or-
   select Export Subroutine from the Project menu; the Select Subroutine box opens.
2. Select the desired subroutine, then save it to the desired folder.



Exporting all of the Subroutines in a Module

1. Right-click the desired module and select Export All Subroutines.

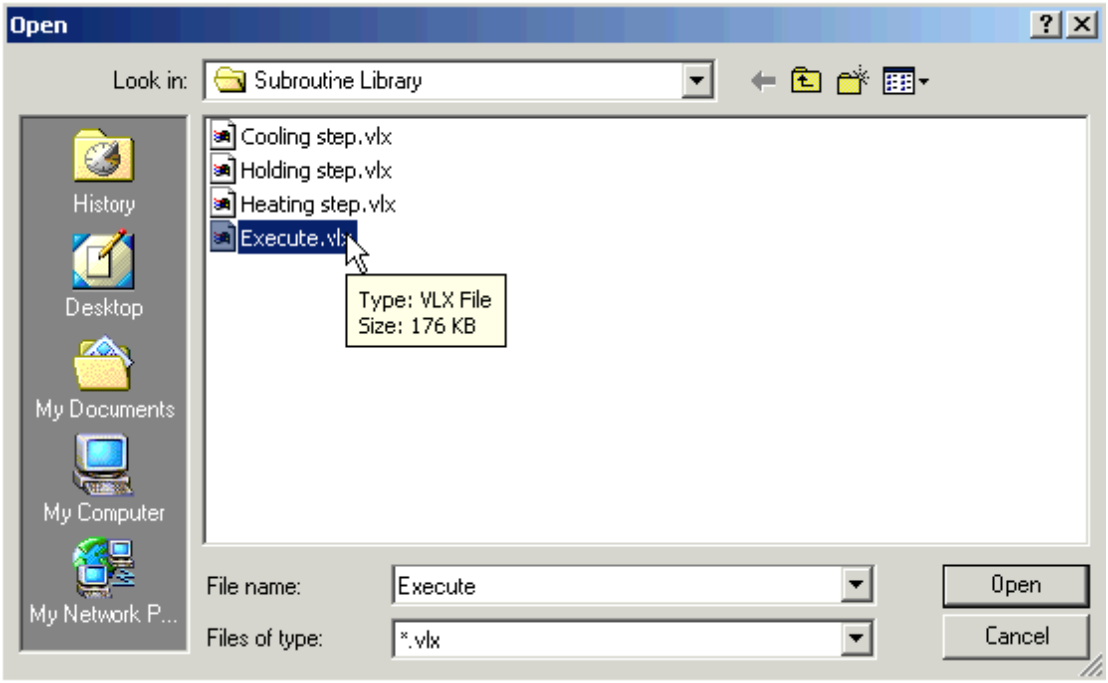


2. Save the .vlx file to the desired folder.

   Note that when you import this .vlx file, **all** of the Subroutines it contains will be imported.

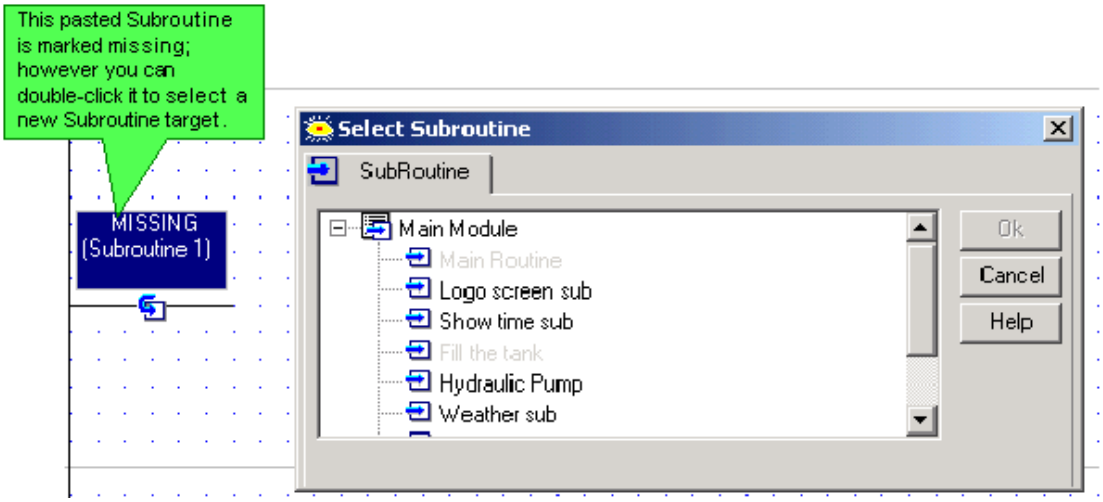
Import

1. Right-click a module name and select Import Subroutine,
   -or-
   Select Import Subroutine from the Project menu; the Open box appears.
2. Select the desired subroutine, then save it to the desired folder.

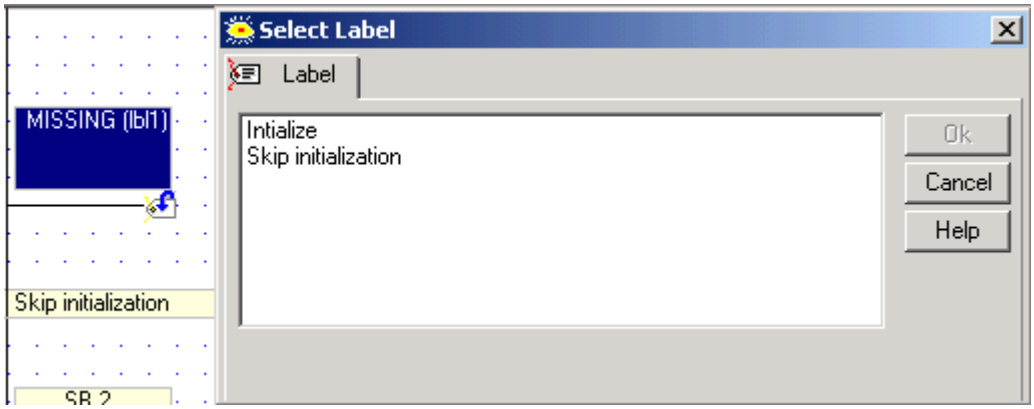


Import/Export is subject to the limitations below.

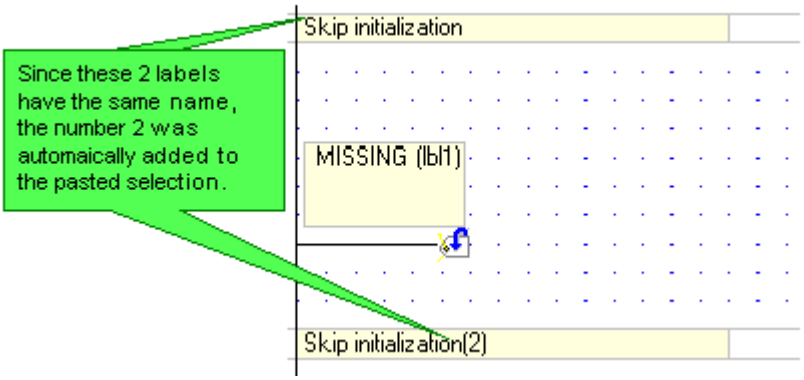
- If the source project contains Call Subroutine or Load HMI operations, note that the referenced elements will be marked as **missing**, even if the target project contains elements of the same name. Note that you can reassign the references.



- If the selection contains an FB operation related to an FB Configuration, and is imported into an application containing an FB Configuration of the same name, the links will be retained.
  If, for example, you export a subroutine containing an SMS Send FB linked to SMS Configuration 'Denmark' and then import this subroutine into another application containing an SMS Configuration 'Denmark', the SMS Send FB will automatically link to 'Denmark'.
- If the selection contains FBs, and no FBs of that type currently exist in the target project, the pasted FBs will be the version currently in VisiLogic FB library--in other words, if the source selection contains older FB versions, they are automatically updated during the Paste operation.
- If the selection contains FBs, and FBs of that type currently exist in the target project in a **different** version, Paste cannot be completed.
- If your selection contains only Jumps, without the attendant Labels, they will be marked as **missing**, even if the target project contains Labels of the same name. Note that you can reassign the references.



- If the selection contains Jumps and Labels with the same name as those in the target project, the Jump, Label and link between them will be automatically recreated by VisiLogic when they are pasted.
  In this way, VisiLogic maintains the integrity of the links between Jumps and their corresponding Labels.



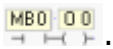
- Note that the following symbols cannot be used in subroutine names: **/ \ | * : ! " < >**. In addition, please note that a name may not include a period followed by a space (for example **`My. Subroutine`**).
  When importing/exporting from older VisiLogic programs containing such symbols, they will be automatically replaced by underscore characters.

Program Control and Sequencing

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.

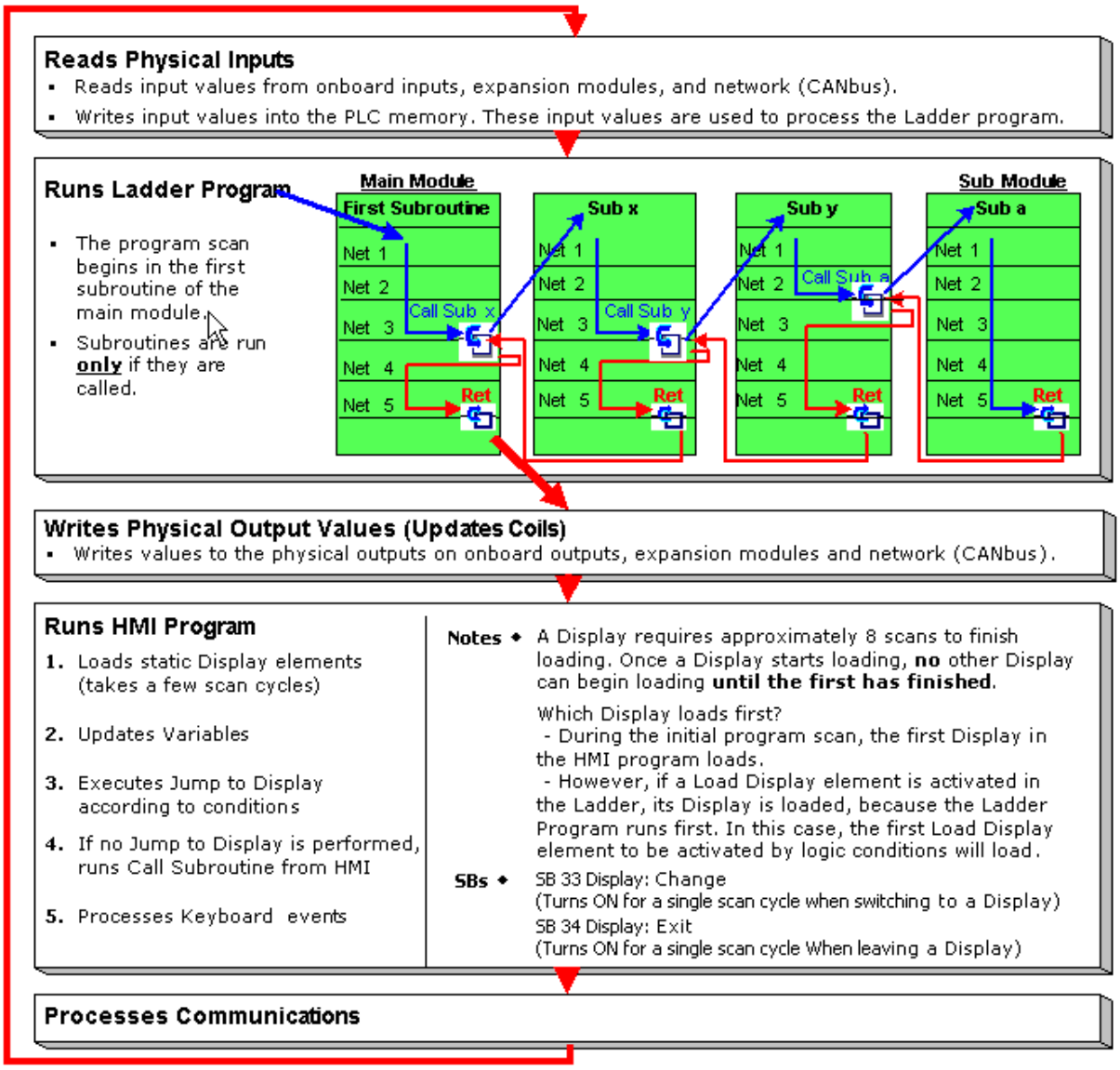
Note •	If a subroutine **does not run**, the coils in that subroutine **will not be updated**. For example, Subroutine 4 contains  . If **MB0** is turned **ON** in Subroutine 1, but Subroutine 4 is **not** called, **O0 is not updated**. The order in which I/Os are updated depend on the PLC program scan.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



Disable-Enable Nets

Disabling a net causes the program scan to skip over it.

To disable a net, right-click the left-hand Ladder rail and select the Disable option from the menu. The disabled net rail is colored green.

To re-enable the net, right-click the left hand Ladder rail of the disabled net and select Enable.

Calls, Jumps, and Labels

The Call menu functions are located on the Utils menu. They enable you to set the

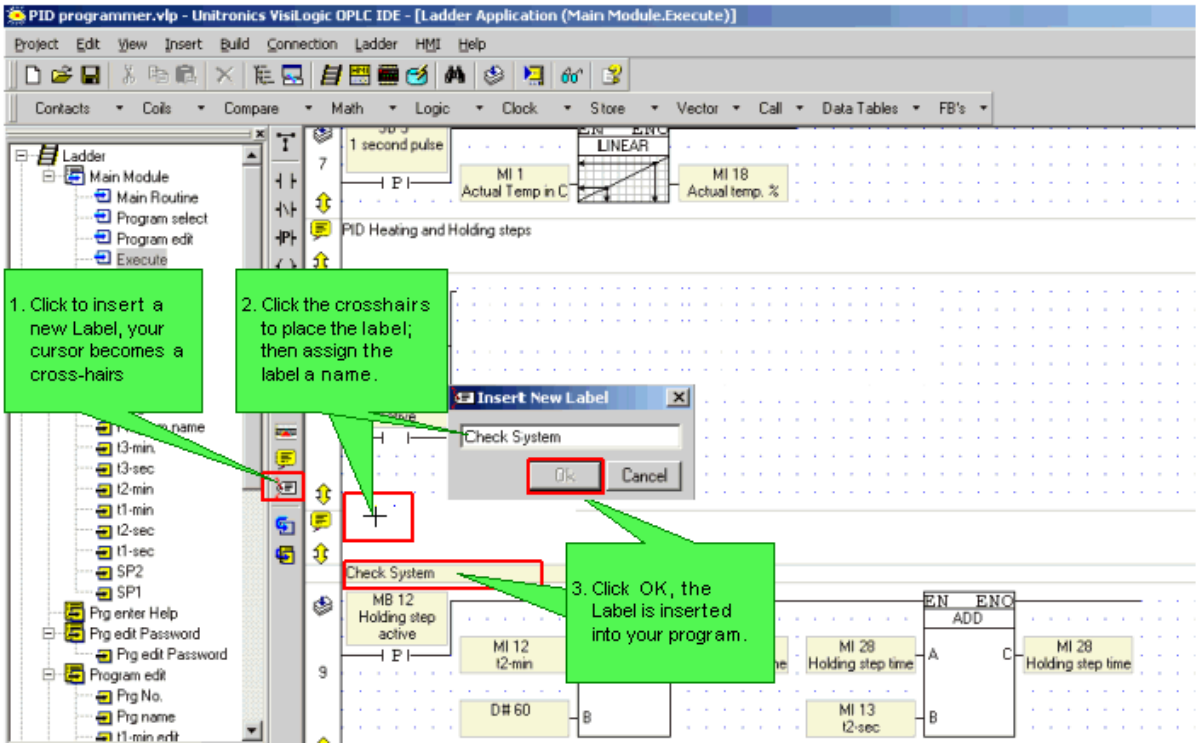
sequence in which your program runs.

Labels & Jumps

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

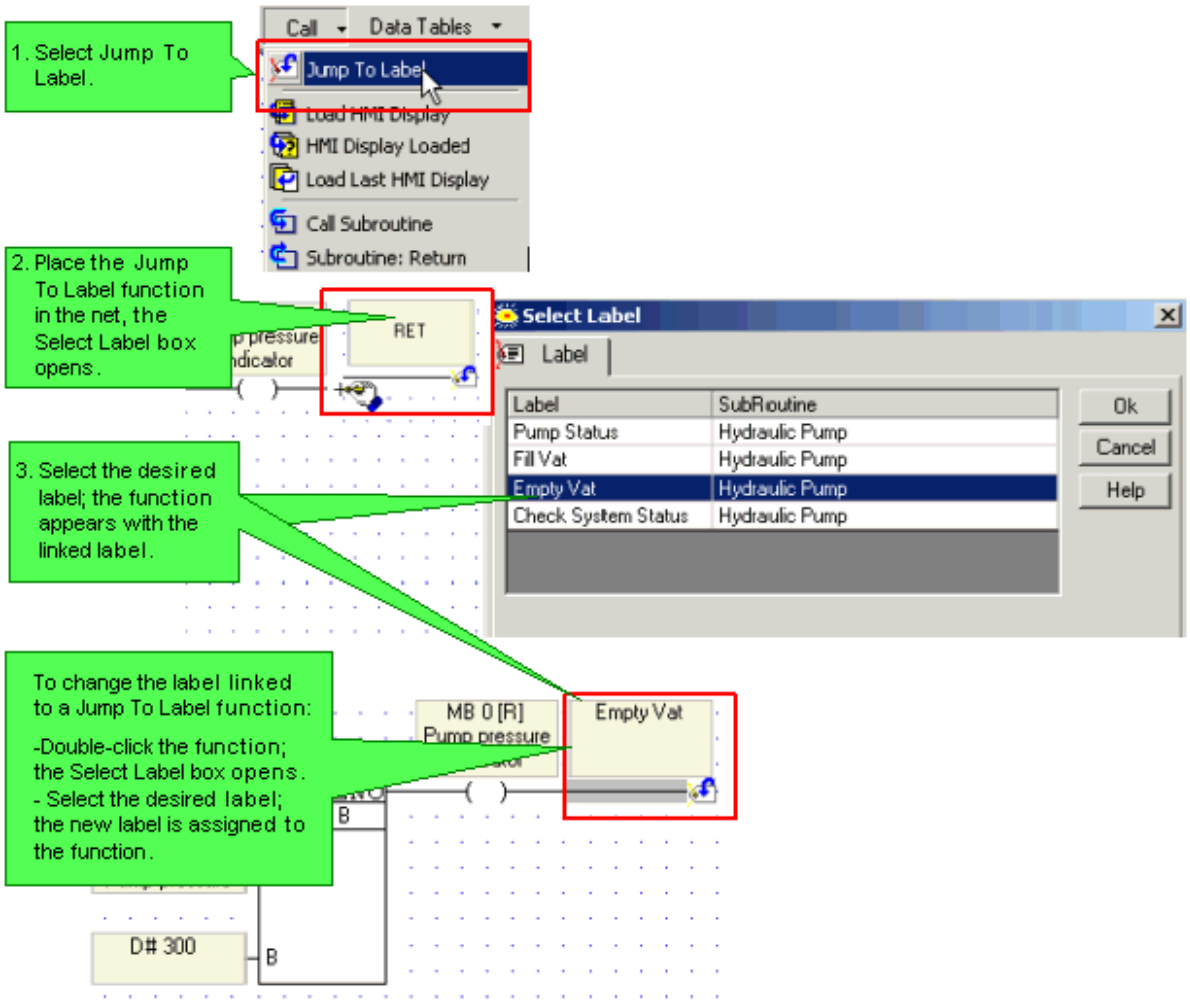
Using Labels

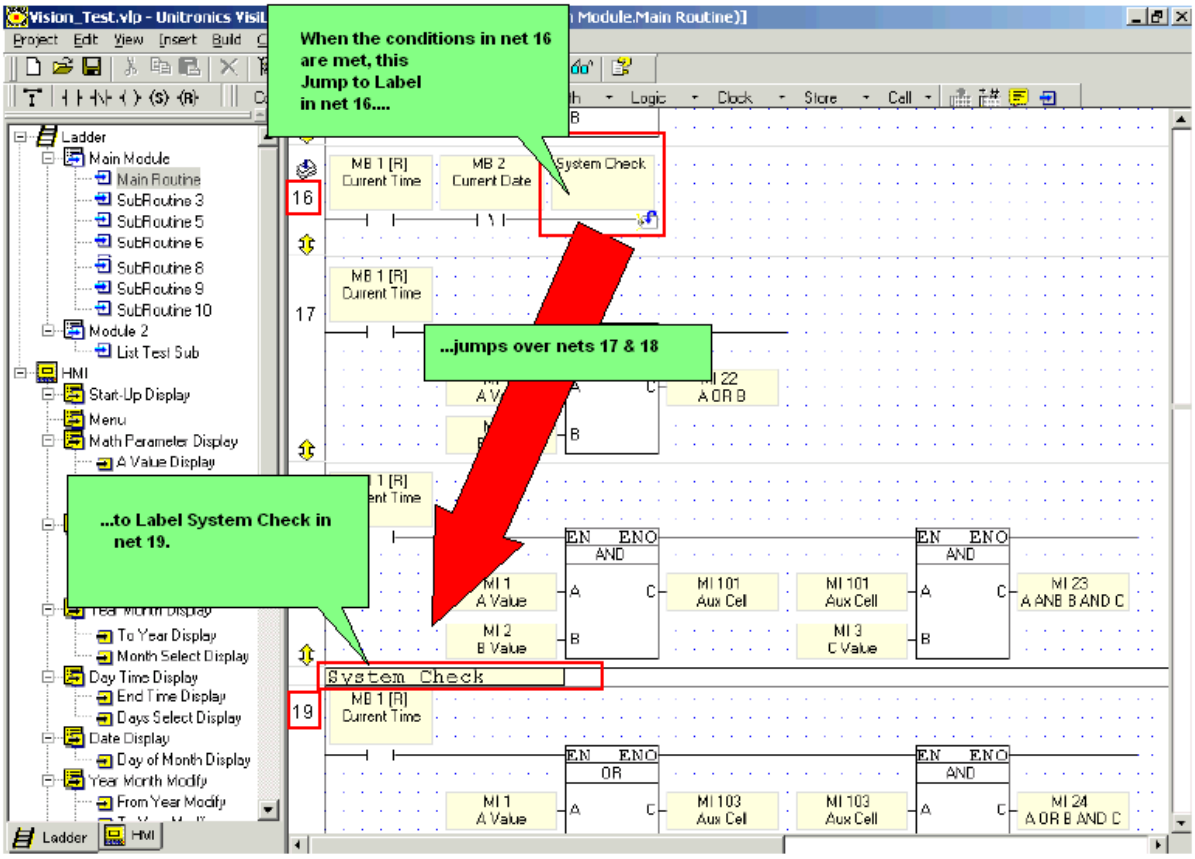
1. Place a Label in a net.



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

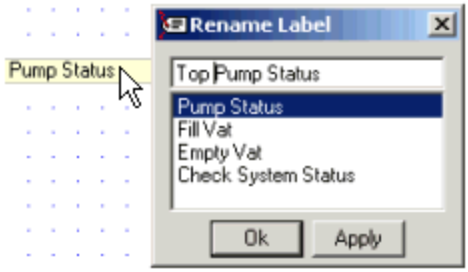
.


1. Select Jump To Label.
Call
Data Tables
Jump To Label
Load HMI Display
HMI Display Loaded
Load Last HMI Display
Call Subroutine
Subroutine: Return
2. Place the Jump To Label function in the net, the Select Label box opens.
RET
Select Label
Label
Label
SubRoutine
Pump Status
Hydraulic Pump
Fill Vat
Hydraulic Pump
Empty Vat
Hydraulic Pump
Check System Status
Hydraulic Pump
Ok
Cancel
Help
3. Select the desired label; the function appears with the linked label.
To change the label linked to a Jump To Label function:
-Double-click the function; the Select Label box opens.
- Select the desired label; the new label is assigned to the function.
MB 0 [R]
Pump pressure
Empty Vat
B
D# 300
B


When the conditions in net 16 are met, this Jump to Label in net 16....
...jumps over nets 17 & 18
...to Label System Check in net 19.
System Check
Ladder
Main Module
Main Routine
SubRoutine 3
SubRoutine 5
SubRoutine 6
SubRoutine 8
SubRoutine 9
SubRoutine 10
Module 2
List Test Sub
HMI
Start-Up Display
Menu
Math Parameter Display
A Value Display
To Year Display
Month Select Display
Day Time Display
End Time Display
Days Select Display
Date Display
Day of Month Display
Year Month Modify
From Year Modify
MB 1 [R] Current Time
MB 2 Current Date
AND
OR
MI 1 A Value
MI 2 B Value
MI 101 Aux Cell
MI 3 C Value
MI 23 A AND B AND C
MI 103 Aux Cell
MI 24 A OR B AND C

Renaming Labels

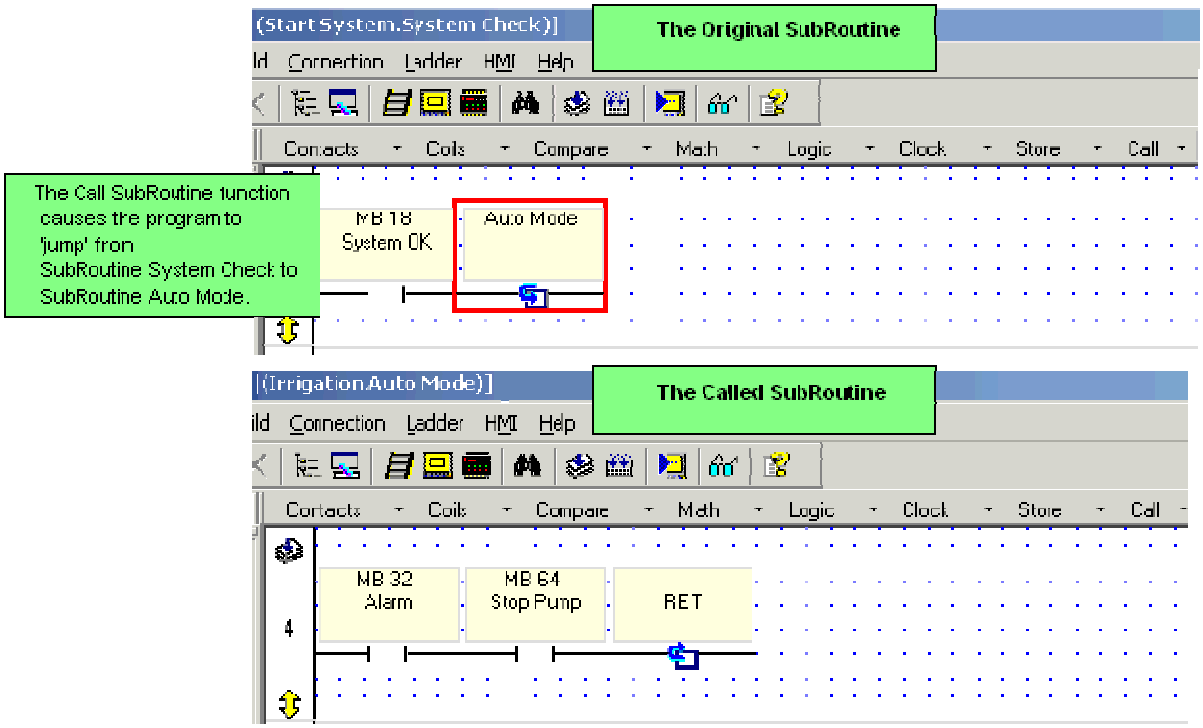
1. To rename a Label, double click it, enter the new name and click Apply.



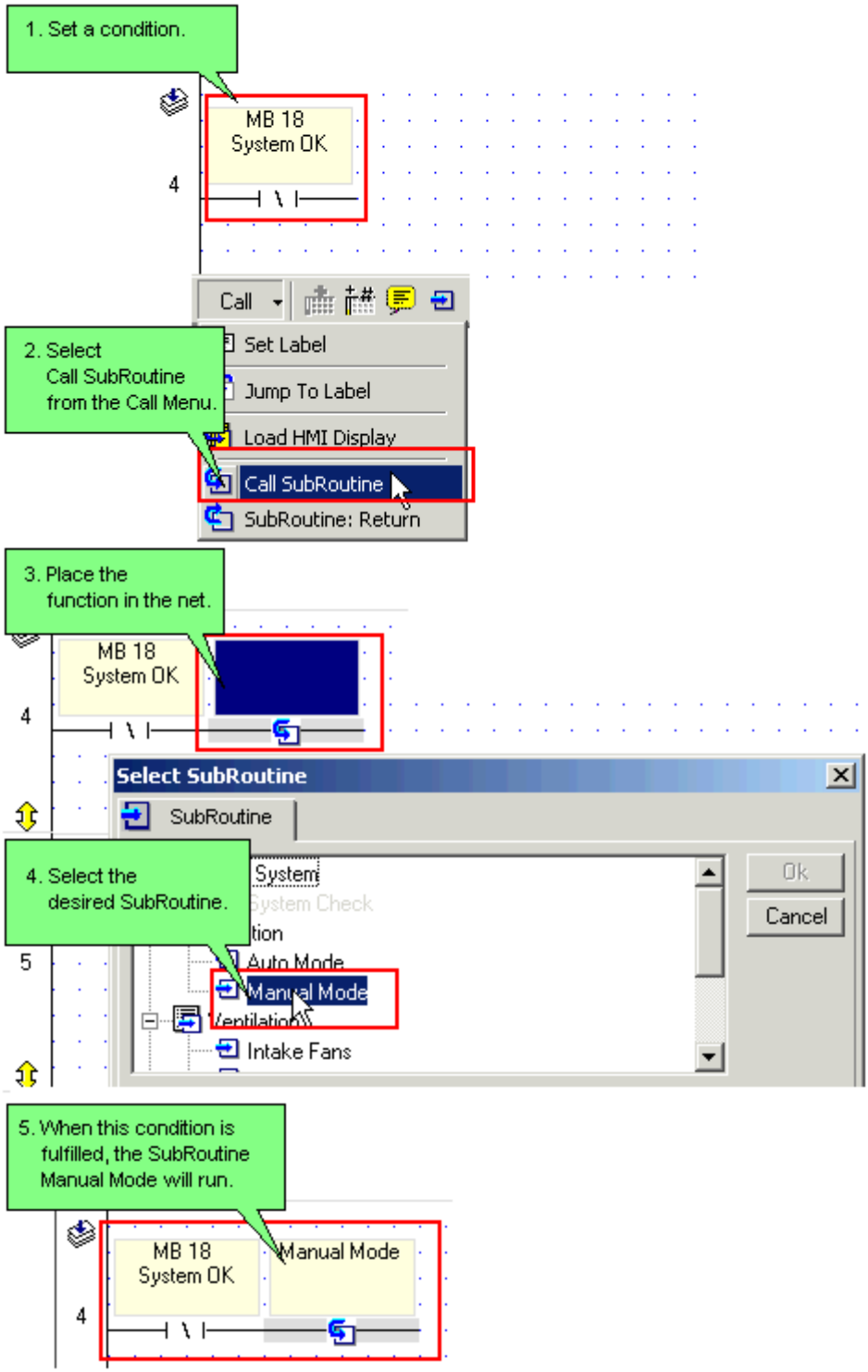
You can also use labels as bookmarks, by using them to mark program sections and then locating them using the Go To Label <Alt> + <Right/Left arrow> and List of Labels <Ctrl> + < L> utility.

Call Subroutine

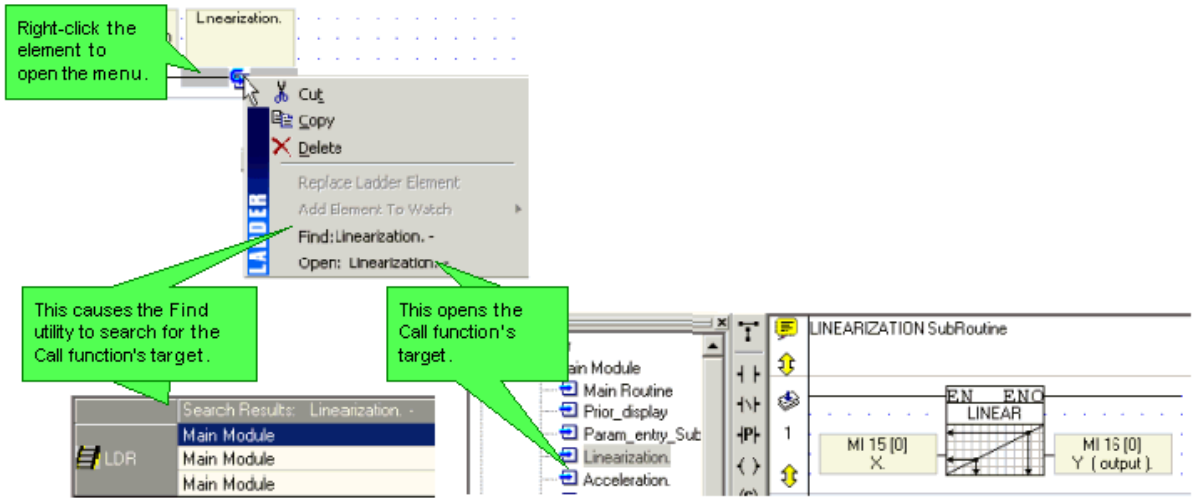
This function causes a subroutine to run in response to a Ladder Condition.



Using Call Subroutine

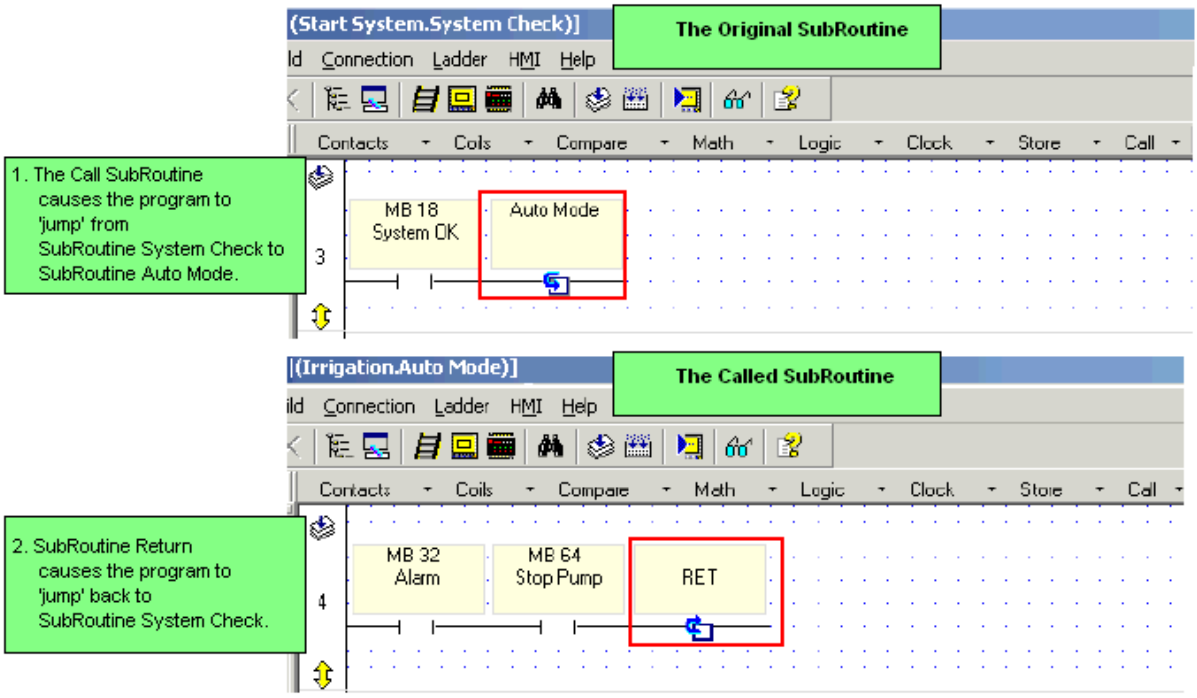


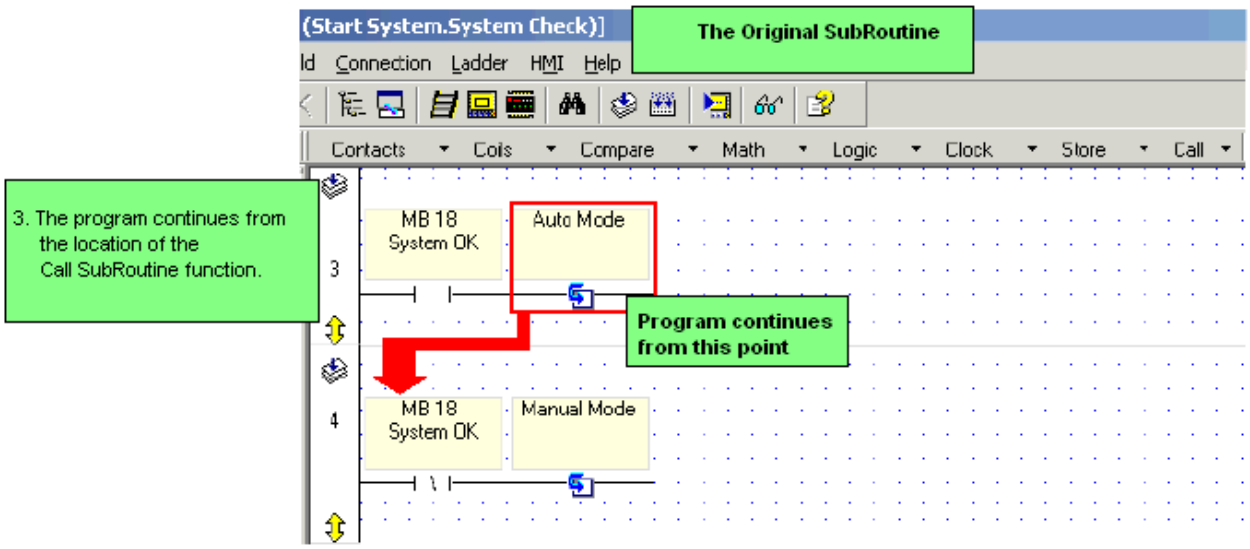
Accessing a Call Subroutine Target



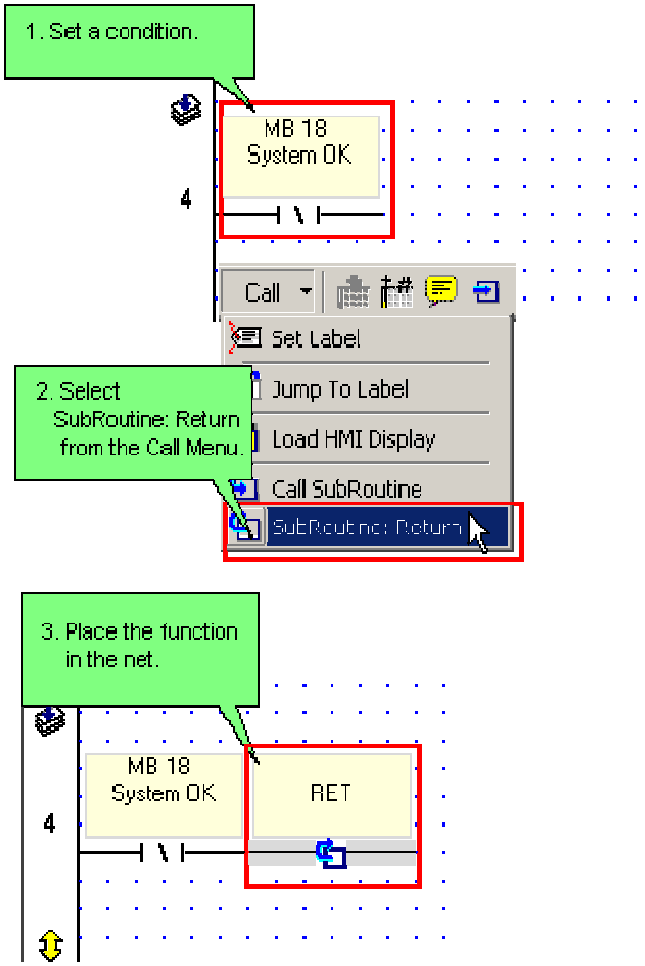
Subroutine: Return

A subroutine runs until it reaches a Subroutine Return function, the function then causes the program to jump back to the previous subroutine. The program returns to the same point from it exited.





Using Subroutine Return



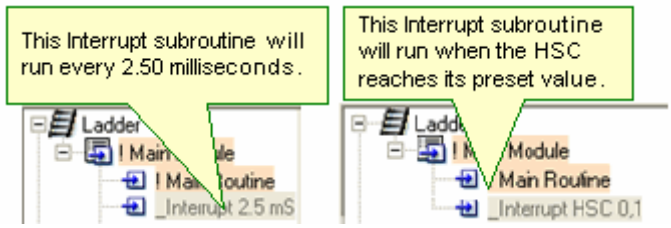
Interrupt Routines

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.



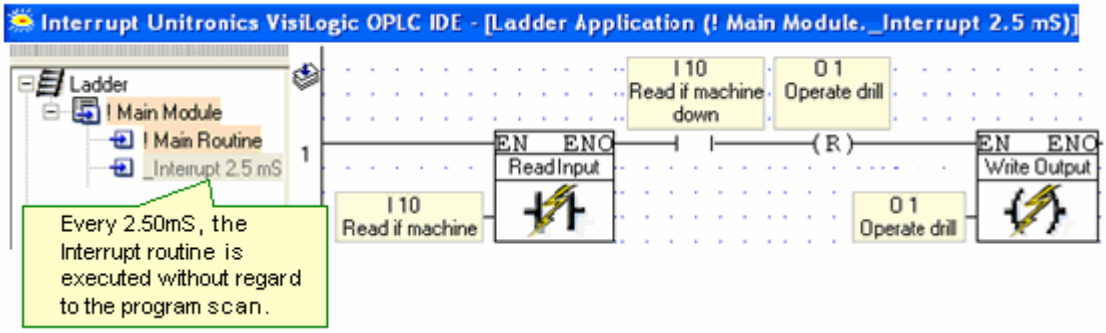
Note

- 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. Call it by naming a subroutine **_Interrupt 2.5 mS**



Including an **_Interrupt 2.5 mS** subroutine in the Ladder application causes:

- The program scan to pause every 2.50 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 •

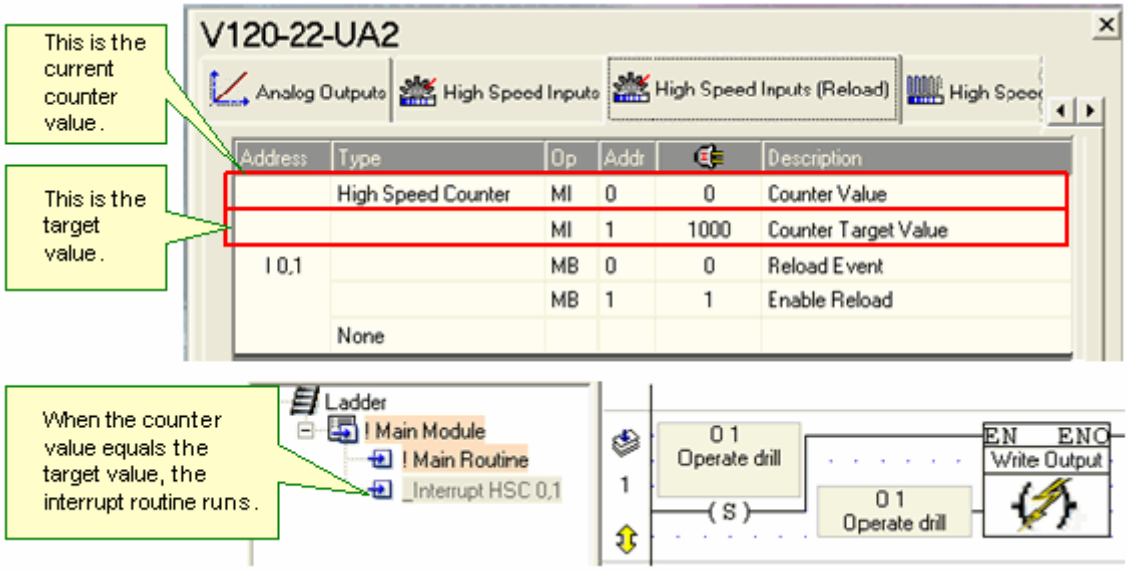
The Subroutine _Interrupt 2.5 mS will run for the first time **after** the first Ladder scan is run.

1.25 mS Interrupt Routine

This function is supported by Enhanced Vision models **only**. Call it by naming a subroutine **_Interrupt 1.25 mS**
It functions exactly like the 2.5mS Interrupt routine described above.

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.



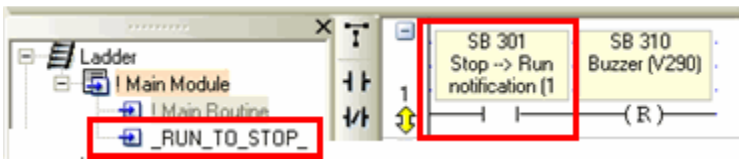
The interrupt function is included in the program by naming a subroutine **_Interrupt x,x** where the first x is the high-speed counter, and the second x is the reload. These subroutines must be named in accordance with your Hardware Configuration as:

- _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.

Stop Mode Subroutine

If you include the exact name of the subroutine:
_ RUN_TO_STOP_
in your program, this subroutine will run a single time when the PLC enters Stop Mode.



Note the related SBs:

- SB 301 PLC exits Stop and returns to Run Mode; turns ON for 1 scan

- SB 302 Stop Mode ON, turns ON when entering Stop Mode, OFF when exits to Run Mode

Note •	If the **name** of the subroutine is **incorrect,** the subroutine **will not run when the PLC is in Stop mode**.
•	The PLC enters Stop mode at the end of the program scan. When the PLC exists Stop Mode, it will start a program scan.
•	This features is not supported by the V120-12 series.

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 

Within Range

Math

Add 

Subtract 

Multiply 

Divide 

Modulo 

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 

RS-SR Flip-Flop

RLO to Bit

Clock

Time 

Day Of Week 

Day Of Month 

Month 

Year 

UTC (Universal Time) functions

Store

Reset Numeric

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

Load Timer Bit Value

BCD to NUM, Num to BDC

Fill Direct

Vector Copy

Step in Range

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

Num to ASCII, ASCII to NumI
Display RTC (ASCII
Time to ASCII
Timer
IP to ASCII
Mac Address to ASCII
Transpose
Strings: Section Operations
Set String Library
Strings:Text Library to ASCII

COM

Set PLC Name

Set PLC ID Number

COM Port: Init

Dial and Hang-up

Last Call (CLIP): Identifying Callers

Ethernet TCP/IP

Send e-mail

CANopen

CANbus UniCAN

CANbus, Layer 2

DF1 (Slave, AB Protocol)

HMI

Load HMI Display Function

HMI-Ladder: Draw Pixel/Line

HMI-Ladder: Clear Rectangle

HMI-Ladder: Previous Var

Inverse Var/Hide Var

Data Tables

Read/Write 

Direct Read/Write 

Data Tables: Clear Table

Data Tables: Find Row

SD Card functions

Set SD Card Password

SD Card: Folder Report Function

SD Card and Data Table Functions (Ladder)

SD Card: Data to Excel

SD File Functions

SD Block Functions

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.

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 not** flow through the Inverted Contact. The emergency light will stay off.

If the Inverted Contact address (power supply) is OFF (logic 0): power **will** flow through the Inverted Contact. The emergency light turns on.

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

Positive Transition Contact (Rise)

A Positive Transition Contact gives a single one-shot pulse when the bit operand it is linked to **rises** from OFF (logic 0) to ON (logic 1).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). You can link them to any of the following bit operands:

- Memory Bit
- System Bit
- Output
- Timer
- Counter

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.

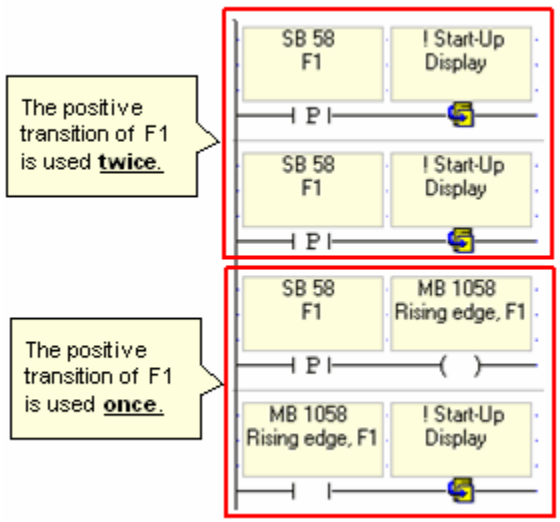
At the end of a scan, the Positive Transition Contact is reset to ON (logic 1). The Positive Transition Contact is re-activated when the linked signal turns from OFF to ON.

Note •

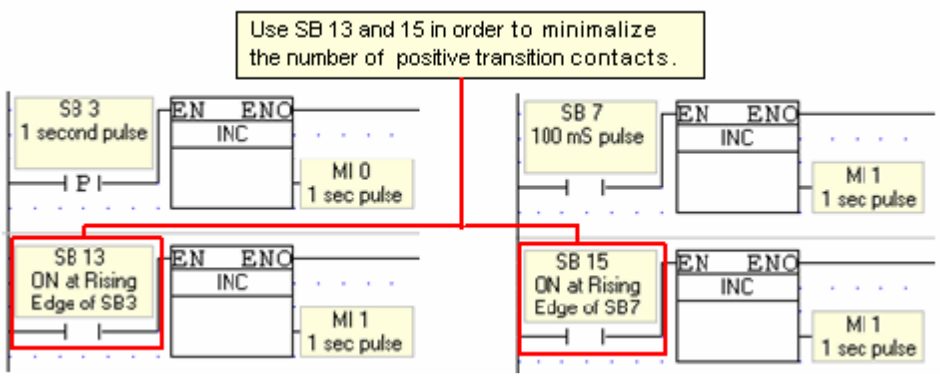
Execution time for Positive and Negative Transition contacts is considerably greater than the execution time for direct and indirect contacts. However, you can decrease the amount of transitional contacts in your program.

Decreasing Number of Transitional Contacts

You can use the coil of a bit operand to save the positive transition of a contact, and then use the direct contact of the operand in your program.



You can use the Direct Contact of SB 13 instead of using the Positive Transition Contact of SB 3, and the Direct Contact of SB 15 instead of using the Positive Transition Contact of SB 7.
SB 3 is a pulse generator, with a cycle time of 1 second and a duty cycle of 50% (0.5 seconds ON, 0.5 seconds OFF).
SB 13 is the Positive Transition (rising edge) contact of SB 3.
SB 7 is a also a pulse generator, with a cycle time of 0.1 second. SB 15 is the Positive Transition (rising edge) contact of SB 7.



Rise/Fall Usage Summary

The **maximum number of** Rise/Fall elements that is allowed in a project depends on the controller model. To ascertain how many elements of each type are in the project, use the Rise/Fall utility on the View menu.

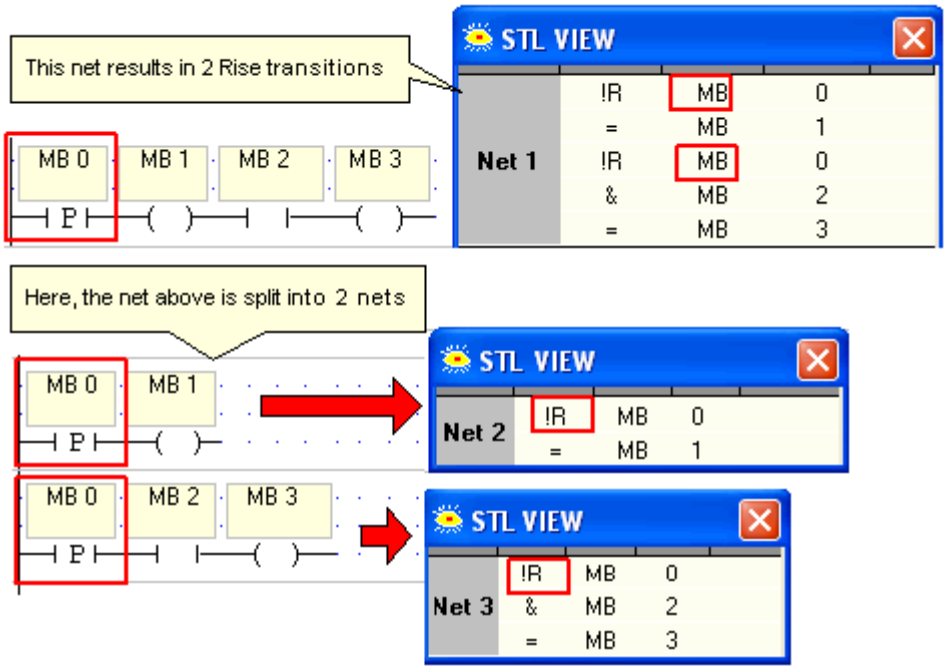
The **sum** of the results must not exceed:

- V570 – 1024 (0...1023)
- V350 – 1024 (0…1023)
- V130 – 512 (0…511)
- V2xx – 256 (0…255)

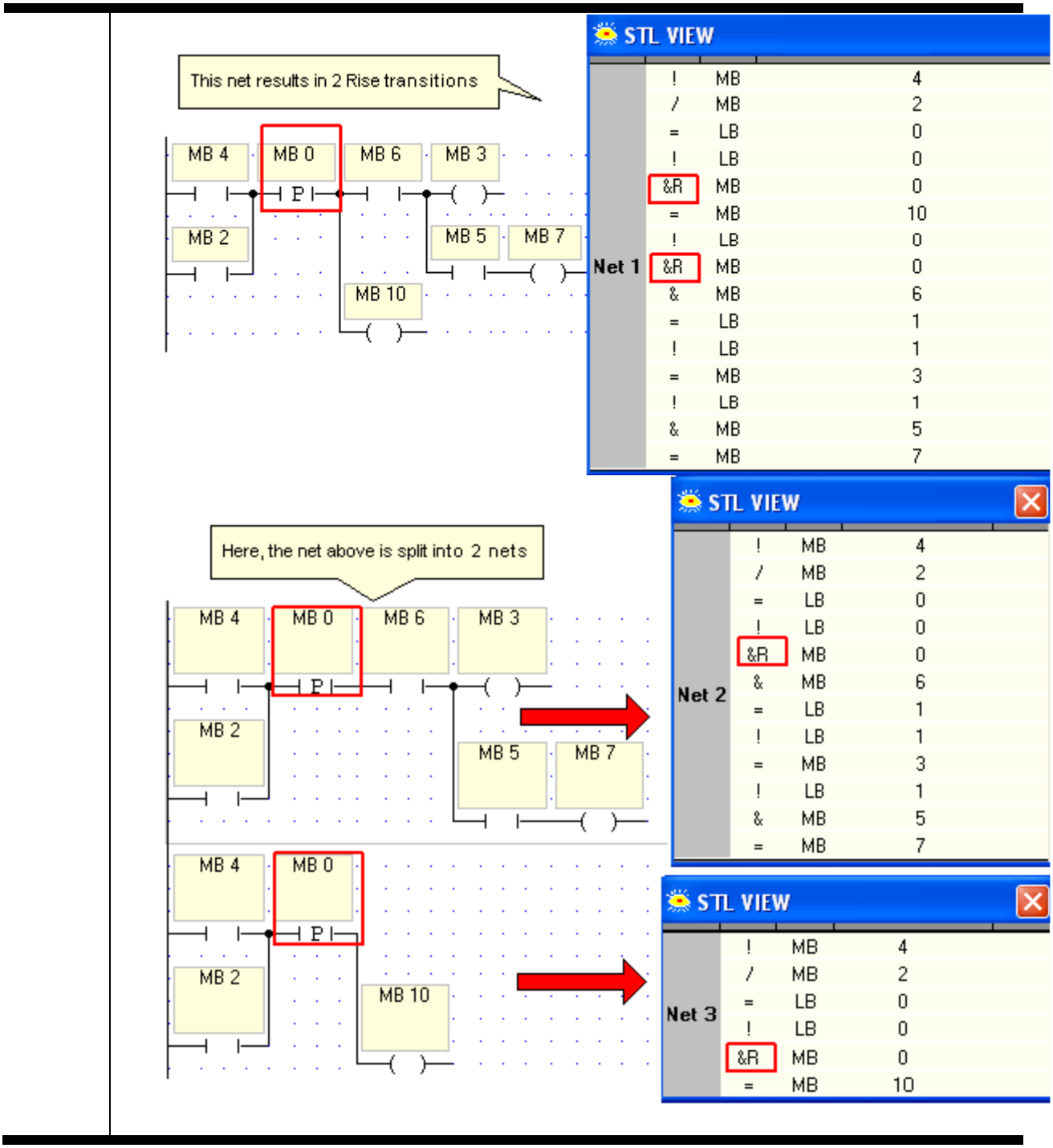
If a program exceeds this number, Error 1017 results.

However, in certain cases, the **actual** compiled number of Rise/Fall elements is **greater than** the total that is shown in the Summary. Examples are shown below.

Example 1



Example 2



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 Positive Transition Contact gives a single one-shot pulse when the bit operand it is linked to **rises** from OFF (logic 0) to ON (logic 1). You can link them 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 •

Execution time for Positive and Negative Transition contacts is considerably greater than the execution time for direct and indirect contacts. However, you can decrease the amount of transitional contacts in your program.

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

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 that 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.

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.

Note that there are differences between Standard and Enhanced Vision Divisions.

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	Q'ty Standard+ V130	Q'ty Enhanced	Value	Address Range Standard	Address Range Enhanced
Input	**I**	544		Bit	**I0-I543**	
Output	**O**	544		Bit	**O0-O543**	
Timer	**T**	192	384	32-bit	**T0-T191**	**T0-T383**
Counters (C)	**C**	24	32	16-bit	**C0-C24**	**C0-C31**
Memory Bit	**MB**	4096	8192	Bit	**MB0-MB4095**	**MB0-MB8191**
Memory Integer	**MI**	2048	4095	16-bit	**MI0-MI2047**	**MI0-MI4094**

Memory Long Integer	**ML**	256	512	32-bit	**ML0-ML255**	**ML0-ML511**
Double Word (unsigned)	**DW**	64	256	32-bit	**DW0-DW63**	**DW0-DW255**
Memory Floating Point Integer	**MF**	24	64	32	**MF0-MF24**	**MF0-MF63**
Constant Value	#	Dynamic			Dynamic	

X Operands (Enhanced only)

X Operands are processed within the CPU's RAM memory. Use them in subroutines where scan time is critical, as for example during Interrupt Routines.

Note that X Operand values are not retained, meaning that they are **not** backed up by battery.

Type	Symbol	Quantity	Value	Address Range
X Bit	**XB**	1024	Bit	**XB0-XB1023**
X Integer	**XI**	512	16-bit	**XI0-XI511**
X Long Integer	**XL**	256	32-bit	**XL0-XL255**
X Double Word (unsigned)	**XDW**	64	32-bit	**XDW0-XDW63**

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

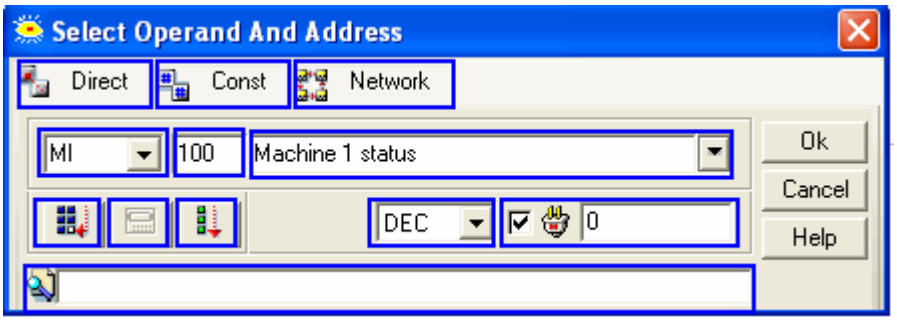
If a controller is networked, the following operands are accessible to other controllers:

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

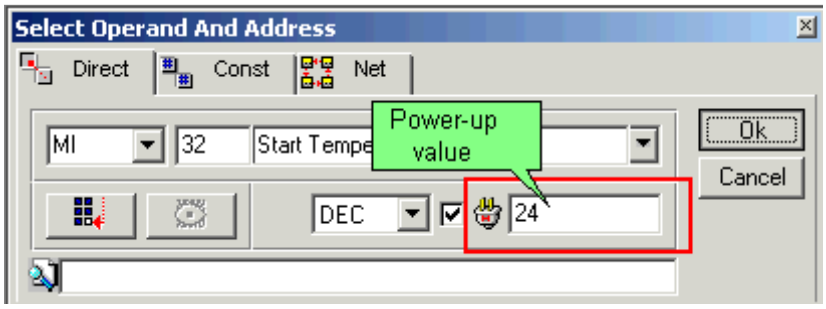
Power-up values can be assigned to most operands. These values are written into the operands when the controller is turned on.

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:

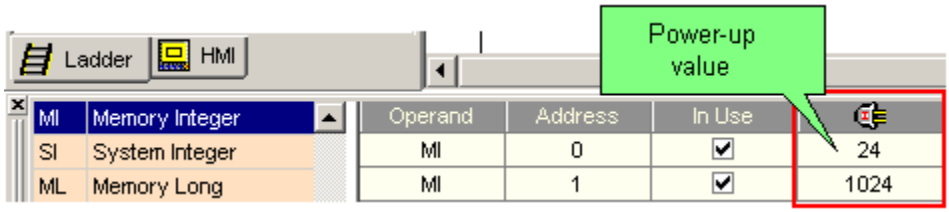
- Select Operand and Address Dialog Box

  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.



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.

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).

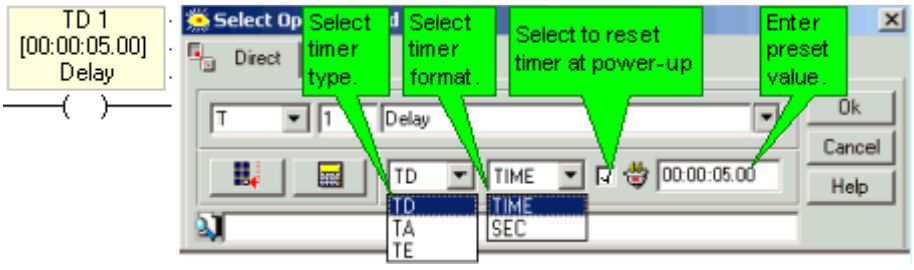
The number of Outputs varies according to the Snap-in I/O Modules and I/O Expansion Modules you integrate into your system.

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.'

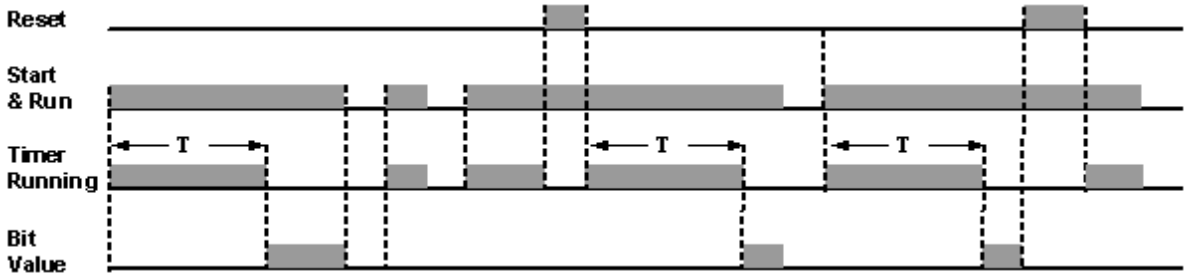


There are 3 types of timers. Each timer type has 3 variables:

- 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.

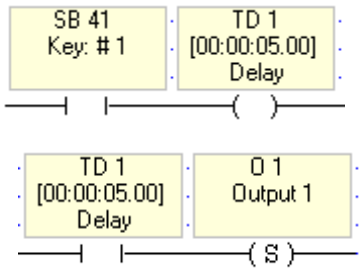
If the timer's Start & Run Condition falls while the timer is decrementing, the timer stops running. The current value of the timer remains.

Timer Reset takes precedence over the timer's Start & Run Condition. When the timer' 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..

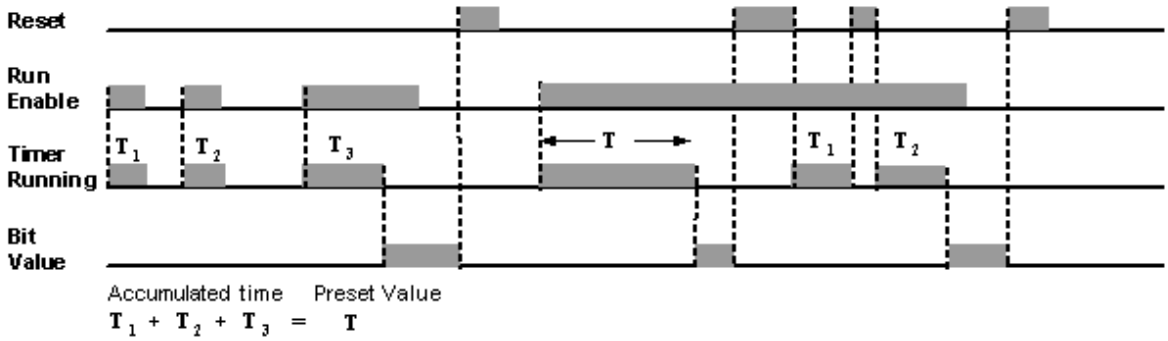
When the timer's Reset Condition falls while the timer's Start & Run Condition is ON, the timer begins to run, exactly the same as when the timer's Start & Run Condition rises.

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' 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.

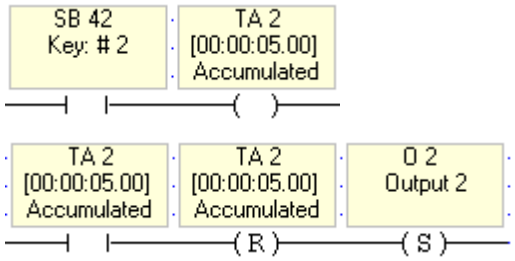
When the timer's Reset Condition falls while the timer's Start & Run Condition is ON, the timer begins to run, exactly the same as when the timer's Run Enable Condition rises.

Note •

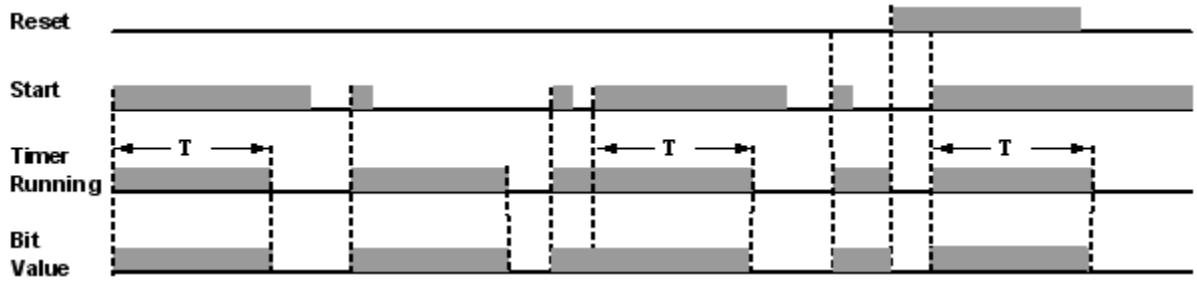
Once a TA Timer has reached its preset value, its Bit Value remains ON until the timer is reset in the program. The timer cannot be activated by Run Enable until it has been reset.

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.

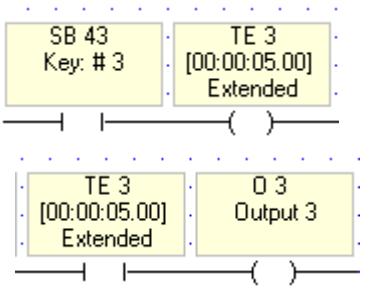
Timer Reset takes precedence over the timer's Start Condition. When the timer' Reset Condition rises, the timer's Bit Value turns OFF. The timer's Preset Value is loaded into the Current Value, and the timer's Start Condition cannot activate the timer as long as Reset is ON..

When the timer's Reset Condition falls while the timer's Start Condition is ON, the timer stops. When the Start condition rises, the timer begins to run, counting down from the Preset Value, 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.

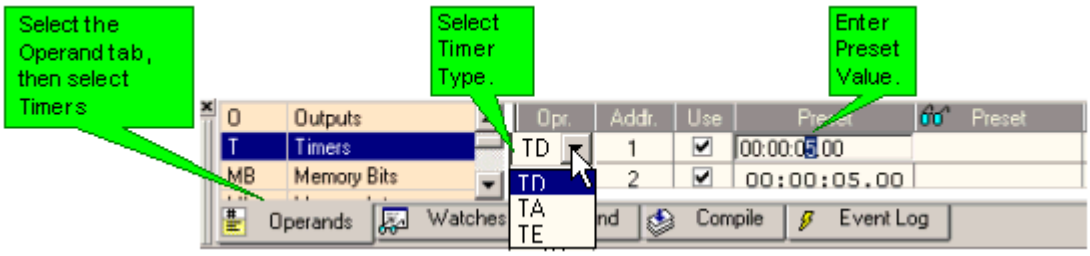
In the nets below, pressing Key #3 on the Vision keypad activates TE3, which is preset to 5 seconds. Once Key #3 is pressed, TE3 decrements to zero. O3 switches on.



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.

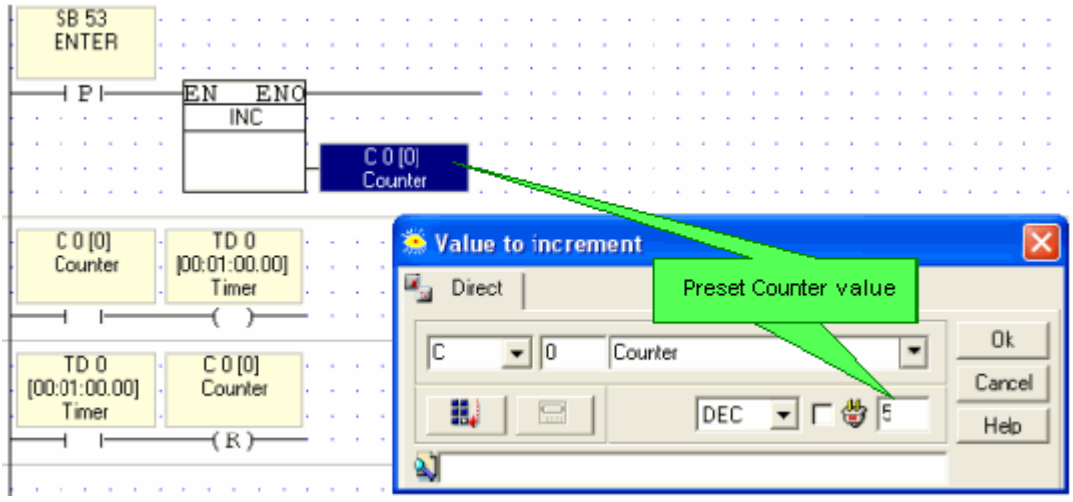
You can also use Information Mode to edit or enter a timer value via the controller keyboard while the controller is running its control program.

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.

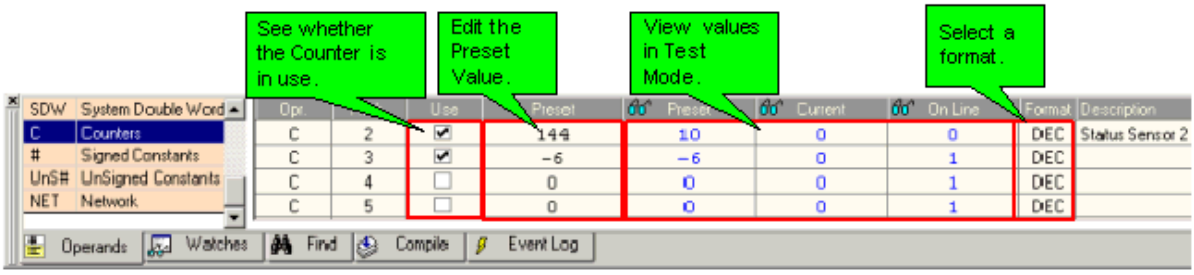
When the accumulated number of pulses equals the counter's preset value, power flows through the function and the counter bit turns ON. Once the preset value is reached, the counter bit stays ON until it is reset via a Reset Coil. This also initializes the counter value.



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.



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 4,294,967,295.

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 -3.402E37 to -1.176E-35 for negative numbers, and +1.176E-35 to +3.402E37 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

X Operands (Enhanced only)

X Operands are processed within the CPU's RAM memory. Use them in subroutines where scan time is critical, as for example during Interrupt Routines.

Note that X Operand values are not retained, meaning that they are **not** backed up by battery.

Type	Symbol	Quantity	Value	Address Range
X Bit	**XB**	1024	Bit	**XB0-XB1023**
X Integer	**XI**	512	16-bit	**XI0-XI511**
X Long Integer	**XL**	256	32-bit	**XL0-XL255**
X Double Word (unsigned)	**XDW**	64	32-bit	**XDW0-XDW63**

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

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

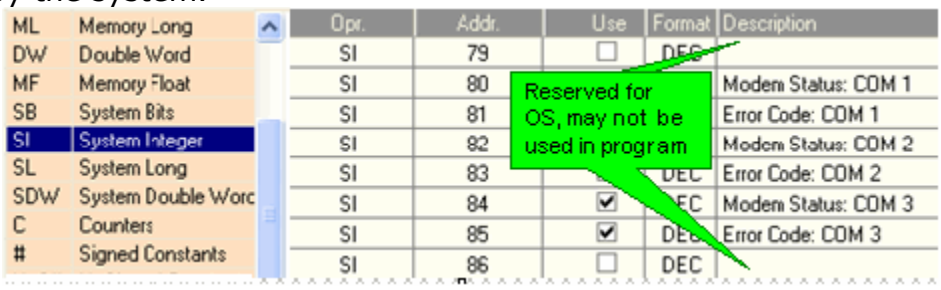
System Operands are used by the controller's operating system to manage certain functions and values. Many System Operands are linked to fixed parameters and are read-only, such as SB 2 Power-up bit, which turns ON for a single cycle whenever the controller powers up.

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 0-15

#	Description	Turns ON when:	Turns OFF when:	Reset by:

SB 0	Always 0	Never	Always	
SB 1	Always 1	Always	Never	
SB 2	Power-up bit	Power-up occurs, for 1 scan		
Note that SB2 is limited to 800 instances per program. You can use SB2 to drive MBs and use those in your program. If you are using Enhanced Vision, note that XBs are initialized at powerup.				
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	Battery needs to be replaced, or RAM has failed	Battery and RAM are functioning	Reset by user: via info, or Communication
SB 10	Float Error	By OS when the result of a float operation is an illegal float value. Error code is in SI440.		By user, or at power-up.
SB 11	User Stack Overflow			
SB 13	ON at Rising Edge of SB3 (1sec pulse)	Turns ON when SB3, 1 second pulse, rises		OS
SB 14	Calculate current controller temperature (not supported by V120/130/350)	By user. When SB 14 turns ON, the value in SI 14, Current Controller Temperature updates.	By OS	OS
SB 15	ON at Rising Edge of SB7 (100 mS pulse)	Turns ON when SB7, 100mS second pulse, rises		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 Note that the touch property must be assigned to a variable. If this property is assigned, touching the variable activates it, causing it to be marked by the blinking cursor.	The screen is not being touched.	OS
SB 17	Enable/Disable Touch-screen 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 (Relevant only to Standard Vision + Touchscreen, not Enhanced))	ON by default in Touchscreen-only models (V290). Causes a Virtual Touchscreen to be shown on screen when the user touches a display entry variable. . In Touchscreen + HMI keypad models (V280), user turns ON to enable Virtual keypad. When ON, the normal alphanumeric keypad is suspended.	Off by default in all models with physical; keypad May be turned OFF by user.	User/ OS

Enable all HMI keys during Keypad Entry, SB 23

#	Description	Turned ON	Turned Off	Comments
SB 23	Enable all HMI keys during	By program or user	Off by default. Once turned ON, must be	By default, an active Keypad Entry variable suspends the

	Keypad Entry		turned OFF by program or user.	normal activity of keypad keys. This means that the following SBs do not rise during keypad entry: SB 40-49, 51, 52, 55, & 56. Turning SB 23 enables the SBs to rise during keypad entry.

#	Description	Turned ON	Turned Off	Comments
SB 23	Enable all HMI keys during Keypad Entry	By program or user	Off by default. Once turned ON, must be turned OFF by program or user.	By default, an active Keypad Entry variable suspends the normal activity of keypad keys. This means that the following SBs do not rise during keypad entry: SB 40-49, 51, 52, 55, & 56. Turning SB 23 enables the SBs to rise during keypad entry.

Initialize and Reset PLC, SB 24

#	Description	Turned ON	Turned Off	Comments
SB 23	Enable all HMI keys during Keypad Entry	By program or user	Off by default. Once turned ON, must be turned OFF by program or user.	By default, an active Keypad Entry variable suspends the normal activity of keypad keys. This means that the following SBs do not rise during keypad entry: SB 40-49, 51, 52, 55, & 56. Turning SB 23 enables the SBs to rise during keypad entry.

#	Description	Turned ON	Turned Off	Comments
SB 24	Initialize and Reset PLC	By program or user	Off by default. Once turned ON, must be turned OFF by program or user.	Set this to cause PLC to reset, and to initialize all operands to 0. Note that SB 300 performs Reset only.

HMI Display tasks, SBs 26-34

SB25	Use operand value as Index of HMI variable	User	User	Enables a variable to be indirectly addressed
SB 26	Exiting OS Draw Mode (ON for 1 cycle after OS draw)	By OS Turns ON for a single cycle when SB 28 turns OF. This happens at the following times: When the PLC exits Info Mode Rises the cycle after a Display is entered. When Virtual Keypad mode exits.	By OS	OS Draw Mode means that the controller's Operating System takes control of the LCD screen: During Info Mode When a Display is entered When the Virtual Keypad (touch-screen models) is displayed
SB 27	Disable all keypad automation (touch-screen + keyboard models only, V280)	By program If SB 22 is ON, SB 27 turns ON automatically	By program	If SB 27 is ON when a Display is shown: 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: ▪ 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.
SB 28	LCD: controlled by OS (OS drawing on LCD)	By OS at entry to drawing mode, remains ON during the drawing task: Info Mode. Rises when a Display is entered. When the V290 enters Virtual Keypad mode and displays the virtual keypad on the LCD	By OS when the OS exits the drawing mode: PLC exits Info Mode. After a Display is entered. When Virtual Keypad mode exits.	Any Ladder- drawn elements (ex. Draw Axis, Trends, Draw Pixel/Line), are cleared when SB 28 turns ON; the programmer may use the Negative Transition of SB 28 to refresh these elements on the LCD.
SB 29	HMI keypad entries complete, reload vars (Relevant for non touch-screen models: V120, V230,V260)	By program	By OS	Turn SB 29 ON after data is keyed into any variable, to enable the user to **skip** keying in data for the **remaining** variables in the current display. When SB 29 is ON: No cursor blinks on screen. The current values of all variables is loaded on screen.
SB 30	Keypad Vars Locked (Standard: OS turns ON after entries complete (Enhanced: User turns ON/OFF) (Relevant for non touch-screen models: V120, V230,V260)	Standard Vision By OS, after all HMI keypad entries are complete By SB29 turning ON By program Enhanced Vision (non touch-screen) Turned ON by User	When a Display is entered By turning SB31 ON When SB 27 and 29 turn OFF When keypad entry variable is touched	Use SB 30 to run ladder tasks that require data entered via keypad. When a variable is active, pressing the Enter button on the keypad signals that the user has finished entering the value. When the Enter button has been pressed for each variable in the current display, SB 30 turns ON. **Note** • To immediately re-enable data entry (restore cursor) turn SB31 ON
SB 31	Refresh current LCD screen display variables (Relevant for non touch-screen models: V120, V230,V260)	By program	By OS	Restores the Display cursor, re-activates all keypad entry variables in the current Display.
SB 32	HMI keypad entry in progress	By OS	By OS	This turns ON automatically when the blinking cursor is on an active variable.

SBs 33 and 34 function when an HMI Display **that calls a subroutine** is loaded/unloaded from the display screen.

Note • ASCII String Display: In cases where a Display contains a Display ASCII String Variable, and the linked subroutine contains the Display String 'trigger' MB, reset this MB when the Display unloads by using the falling edge of SB 34.

SB 33	Load Display with linked Call Subroutine	By OS	By OS	When a Display containing a Call Subroutine **starts loading**, SB 33 turns ON for a single scan cycle the

				first time the linked subroutine runs. Use this SB to initialize operands in the HMI subroutine.
SB 34	UnLoad Display with linked Call Subroutine	By OS	By OS	When a Display containing a Call Subroutine **starts unloading**, SB 34 turns ON for a single scan cycle the last time the linked subroutine runs.
SB 36	INFO mode	By OS, Remote Access, or program	Turns OFF when user exits Info Mode	
SB 37	Exclude last Display from FIFO	By Remote Access, or program	OS	Enables user to skip going back to certain Displays.
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.
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
SB 250	Keypad entry within limits	By OS	By OS " The current Display is either re-called or changed, or " At the beginning of the next program cycle.	Turns ON for one scan when the entered value is within the Min/Max limits set in the variable's parameters.
SB 251	Keypad entry exceeds limits	By OS	By OS " The current Display is either re-called or changed, or " At the beginning of the next program cycle.	Is ON when the entered value is within the Min/Max limits. **Note** • When this SB is ON, the blinking cursor remains on the active variable even after the user presses Enter.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 26	Exiting OS Draw Mode (ON for 1 cycle after OS draw) OS Draw Mode means that the controller's Operating System takes control of the LCD screen: During Info Mode When a Display is entered When the Virtual Keypad (touch-screen models) is displayed When 'Symbols' are displayed during Keypad Entry.	Turns ON for a single cycle when SB 28 turns OFF. This happens at the following times: When the PLC exits Info Mode. Rises the cycle after a Display is entered. When Virtual Keypad mode exits. After 'Symbols' are displayed during Keypad Entry.	At all other times	OS
SB 27	Enter Display without active Keypad Entry Variables If SB 27 is ON when a Display is	By program	By program	

	shown: 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: ▪ 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.			
SB 28	LCD controlled by OS (OS drawing) OS Draw Mode means that the controller's Operating System takes control of the LCD screen: During Info Mode When a Display is entered When the Virtual Keypad (touch-screen models) is displayed When 'Symbols' are displayed during Keypad Entry.	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.	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	By program	By program	OS
SB 30	HMI keypad entries completed When a variable is active, pressing the Enter button on the keypad signals that the user has finished entering the value. **Note** • Turning this SB OFF, via SB31, enables the variables to be reactivated.	Turns ON when: Turns ON when: When the Enter button has been pressed for each Variable, SB 30 turns ON. By program By SB29, by program	Turns OFF when: SB31 turns ON When PLC is initialized	OS
SB 31	Refresh current LCD screen display variables	Turning this ON reloads the display, initializing all Keypad Entry variables.		OS
SB 32	HMI keypad entry in progress			OS

SBs 33 and 34 function when an HMI Display **that calls a subroutine** is loaded/unloaded from the display screen.

Note • ASCII String Display: In cases where a Display contains a Display ASCII String Variable, and the linked subroutine contains the Display String 'trigger' MB, you can reset this MB when the Display unloads by using the falling edge of SB 34.

SB 33	Load Display with linked Call Subroutine - Use this SB to initialize operands in the HMI subroutine. - Do not link this SB to a positive or negative transitional contact.	When a Display containing a Call Subroutine **starts loading**, SB 33 turns ON for a single scan cycle the first time the linked subroutine runs.		OS
SB 34	UnLoad Display with linked Call Subroutine	When a Display containing a Call Subroutine **starts**		OS

		unloading, SB 34 turns ON for a single scan cycle the last time the linked subroutine runs.		

OnLine Test SB 35

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 35	OnLine Test Point	During OnLine mode, Single Scan, when more than 1 instance of OnLine Test Point is activated (receives RLO).	One or none instances are 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

Exclude last Display from FIFO,SB 37

#	Description	Turned ON	Turned Off	Comments
SB 37	Exclude last Display from FIFO	By program	By program	Enables user to skip going back to certain Displays

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.

FLASH Memory Access, SB 39

#	Description	Turned ON	Turned Off	Comments
SB 39	FLASH on LCD, Display not Refreshed (V570, 290-C,530)	FLASH memory is on screen	By OS	

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	Key is pressed/held down	Key is released	OS
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

Disable HMI cursor blinking SB 73

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

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 73	Disable HMI cursor blinking (Turn ON to disable blinking)	By user program	By user program	By user program

Download Complete, SB 75

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 75	Download Complete, PLC and HMI applications	When Download is finished, Turns ON for single scan ; PLC can then run application	The scan after Download ends	OS

Keypad Entry: Focus (V130), SB 76

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 76	Keypad Entry: Focus (V130) If SB 76 is OFF after Keypad Entry, the user must use the arrow keys to move to the next variable in the Variable Tab Order If SB 76 is ON, the user moves to the next variable by pressing the Enter button twice: - Once to 'approve' the value - Once to jump to and open the next variable for data entry. To enable the user to press Enter once, to both jump to **and** to automatically open the next variable for data entry, turn ON both SB76 and SB108 (Press "Enter" 1x) Note that the user can press ESC to exit keypad entry mode.	By user	By user	By user, or at Power-up

Modem 'busy' status, (Color only) SB 77-79

Each port is linked to an SB indicating modem communication status. These can be used as a condition to sending new messages.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 77	Modem 'Busy': COM Port 1	Port is busy transmitting or receiving	Port is free	OS
SB 78	Modem 'Busy': COM Port 2			
SB 79	Modem 'Busy': COM Port 3			

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	Example: COM Port 1		
SB 80	Modem Initialized: 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	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 91

See Help topic Detecting short-circuited end devices

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 91	I/O Exp. Module--Command buffer is full	ON when commands **cannot** be sent to the I/O module.	OFF when commands **can** be sent to the I/O module.	

GPRS modem connected, SB 100

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 100	GPRS modem connected	Call Remote device begins. GPRS incoming call is answered.	End Session succeeds. Disconnect from Network succeeds.	OS

MODBUS Read Long: SB 102

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 102	MODBUS Read long: Transpose 16 bits of 32-bit long	By User program	• By default • By User Program	User

Press "Enter" 1x (V130) SB 108

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 108	Press "Enter" 1x (V130) If BOTH SB76 and SB108 are ON, after entering a keypad value, the user needs to press Enter only once, to approve the value, and to automatically jump to **and** open the next variable for data entry. Note that the user can press ESC to exit keypad entry mode.	By user	By user	By user

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

Keypad keys, letter/number order, V130 SB115

Each one of the V130 keypad keys 2 to 9 are marked with both letters and numbers. For example, Key 2 is marked with the numeral 2, and by the letters abc. By default, at keypad entry, the a single keypress enters '2', two key presses enter 'a', three enter 'b', and so on.

In order to cause the letter 'a' to be selected by a single keypress, turn SB 115 ON. This reverses the number-letter order to letter number, and in the case of Key 1, reverses number-symbol to symbol-number.

#	Description	Turns ON :	Turns OFF :	Reset by:
SB 115	V130 only. Reverse key letter/number order	User Program	Default. User Program	User

Save Trends to SD Card, SB 116-119

When you save a Trend to an SD card, each time you start and stop the save, another segment is added to the .utr file.

#	Description	Turns ON :	Turns OFF :	Reset by:
SB 116	SD Trends to SD: Set to Overwrite .utr	User application	User application	User
Use these to control the display of Trend segments on the HMI screen. Use the inverted contact of SB119 as a condition.				
SB 117	SD Trends: Jump to next segment	User application	User application	User
SB 118	SD Trends: Jump to previous segment	User application	User application	User
SB 119	SD Trends: System busy - Draw Trend is gathering data	User application	User application	User

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			

Remote Access - Read Only, SB140

#	Description	Turns ON :	Turns OFF :	Reset by:
SB 140	Remote Access - Read Only	User application When ON, PLC ignores requests from Remote Access and Remote Operator	User application	User

Ethernet-enabled controllers only, SBs 141-176

#	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	Ethernet: Socket	Socket 2	Socket 2		

145	2 Initialized	initialization succeeds	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		SBs 147-150 turn ON when: • Link exists • Ethernet Card initialization complete • Socket initialization complete • Hardware TCP/IP Socket state is Connection Established
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 (cable plugged in)	No link exists (cable disconnected)		This refers to the physical Ethernet cable
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		
SB 159	Enable Unicast, Socket 0	Turn ON (ON by default) to enable Socket 0 for Unicast	Turn OFF to disable Socket 0 for Unicast		
SB 162	Ethernet Reconnect parameters saved				

SB 163	Connection is Closed (Socket 0)	By OS, when connection is closed			SB turns ON when Close Connection is performed. This is after Transmit / Receive buffers are empty or 1-second timeout has passed. Socket is initialized.
SB 164	Connection Closed (Socket 1)	By OS, when connection is closed			
SB 165	Connection Closed (Socket 2)	By OS, when connection is closed			
SB 166	Connection Closed (Socket 3)	By OS, when connection is closed			
SB 167	Ethernet Critical error	Turns ON at critical error			OS reads Ethernet card registers to SI 300 - 427, then re-initializes the card. User must reset PLC
SB 168	Enable "Link lost" auto recover	Automatically retry link	Do not automatically retry (default)		SB 168 should be turned ON at power-up. It is OFF by default to preserve backwards compatibility with applications created previous to OS 4.70 B14. If SB 168 is ON, when the Ethernet link fails, the OS saves the Ethernet parameters and resets: • SB 142 Card Initialized • SBs 143-146 Socket initialized (Sockets 0-3) • SBs 147-150 Socket connected (Sockets 0-3) When the Ethernet link is reestablished, the O/S performs CARD INIT and SOCKET INIT for all 4 sockets according to the saved parameters.
SB 169	Automatic reconnect requested, in progress (Socket 0)	At Ladder			TCP - Used as internal flags by O/S in order to perform auto reconnect (user parameters SI 107 – 110)
SB 170	Automatic reconnect requested, in progress (Socket 1)				
SB 171	Automatic reconnect requested, in progress (Socket 2)				
SB 172	Automatic reconnect requested, in progress (Socket 3)				
SB 173	Automatic reconnect				

	requested (Socket 0)			
SB 174	Automatic reconnect requested (Socket 1)			
SB 175	Automatic reconnect requested (Socket 2)			
SB 176	Automatic reconnect requested (Socket 3)			

Received message is invalid, SBs 180-182

When SBs180-182 turns ON, the STX, ETX, or checksum of the received message was incorrect.

SB	Description	Turns ON when:	Turns OFF when:
180	COM1 Received Message Invalid, STX/ETX/Checksum (V570, 290-C,130)	Received message is invalid	Another message is received
181	COM2 Received Message Invalid, STX/ETX/Checksum (V570, 290-C,130)		
182	COM3 Received Message Invalid, STX/ETX/Checksum (V570, 290-C,130)		

SMS message transmission status, SBs 184-185

Standard Vision Division

Controllers in this division can only support a single modem. You can connect a modem to any COM port. However, note that SB 184 TX Success and SB 185 TX Failed indicate message transmission status regardless of the actual COM port connected to the modem.

Enhanced Vision Division

Controllers in this division can support a modem on each COM port. Each port is linked to a Succeed and Fail SB:

COM1: SB 184 and SB 185, COM2: SB186 and SB 187, COM3: SB 188 and SB 189.

Each port has a Succeed and Fail SB. When the Send process begins from a particular COM port, for each and every message, both the Succeed and Fail SB turn OFF.

If the message is sent successfully, the bit turns ON, indicating the success or failure of that message.

If the message fails, when TimeOut is exceeded or because the modem reports an error, the bit remains OFF.

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, COM1 (ACK)	Transmission succeeds	Transmission begins
185	SMS: Transmission Failed, COM1 (NACK)	Transmission fails	
186	SMS: Transmission Succeeded, COM2 (ACK)	Transmission succeeds	
187	SMS: Transmission Failed, COM2 (NACK)	Transmission fails	
188	SMS: Transmission Succeeded, COM3 (ACK)	Transmission succeeds	
189	SMS: Transmission Failed, COM3 (NACK)	Transmission fails	

SMS messages: Write to Vector SBs 198-199

Use these together with SI 198 and 199 to write incoming SMS messages to a vector of operands. This does not affect the function of the SMS message function blocks.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 198	SMS Arrived, Record the Received SMS Message Length in SI 198	If SB 199 is set, , SB 198 is set when a message is received	User Program	User Program
SB 199	Save SMS to Memory Vector	User Program		

CANbus, SBs 200-215, 236-237

The function of some operands depends on whether the CANbus network is defined as CANbus ISC or UniCAN.

When using CANbus ISC

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			

When using UniCAN

SB#	Description	Turned ON	Turned Off	Comments
200	Broadcast bit	When UniCAN broadcast MB is received whose status is ON.	When UniCAN broadcast MB is received whose status is OFF.	The user must initialize this SB
201	High Priority Send Buffer Status	When full	When not full	Use the negative transition of this SB as a Send UniCAN condition for High Priority messages
202	Low Priority Send Buffer Status	When full	When not full	Use the negative transition of this SB as a Send UniCAN condition for Low Priority messages
203	UniCAN Broadcast in Progress	When data is being sent	When data is not being sent	Use the negative transition of SB 203 as a Send Broadcast condition,

SD Card, SBs 217-219

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 217	SD Card Present	An SD Card is in the slot, and is formatted to FAT32	SD Card is not found, or is incorrectly formatted	OS
SB 218	SD Card Write Enabled	Write is enabled: the card's write-protect lock is off	Write is disabled: the card's write-protect lock is on	OS
SB 219	SD FIFO Empty (SD Card may be Ejected)	Power-up No SD Card is in Slot No SD requests exist	There are no SD requests pending, such as Data Table Copy/Log, Alarms, or from Info Mode	OS

CANopen, SBs 240-243

SB#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 240	CANopen: Configuration downloaded	The CANopen Configuration FB is downloaded to the PLC	No CANopen Configuration is present	PLC
SB 241	CANopen: Configured	CANopen Configuration was successful	CANopen Configuration failed	PLC
SB 242	CANopen: SDO in Progress	SDO is busy transferring data	SDO is not in progress	PLC
SB 243	CANopen: SDO transfer failed	SDO data transfer fails	SDO transfer begins	PLC

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 (Relevant for Standard Vision and non touch-screen models: V120, V230,V260)	Turns ON for one scan when the entered value is within the Min/Max limits set in the variable's parameters.	• The current Display is either re-called or changed, or • At the beginning of the next program cycle.	
SB 251	Keypad entry exceeds limits (Relevant for Standard Vision and non touch-screen models: V120, V230,V260)	Is ON when the entered value is within the Min/Max limits. **Note** • When this SB is ON, the blinking cursor remains on the active variable even after the user presses Enter..	• The current Display is either re-called or changed, or • At the beginning of the next program cycle.	

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	User Program

SMS Force SMS Display, SB 280

#	Description	Turns ON when:	Turns OFF	Reset by:

			when:	
SB 280	Force Message Display on Cell Phone	User Program Should be turned ON at power-up, before Com Init.	User Program	User Program

CANopen Buffer Management SB 284-293

Use a Negative Transition contact of the appropriate SB as a Send condition.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 284	Send PDO1	Buffer is full: SI 212 =8	Number of messages in buffer is less than maximum	PLC
SB 285	Send PDO2	Buffer is full: SI 213 =8		
SB 286	Send PDO3	Buffer is full: SI 214 =8		
SB 287	Send PDO4	Buffer is full: SI 215 =8		
SB 288	RTR PDO1	Buffer is full: SI 216 =12		
SB 289	RTR PDO2	Buffer is full: SI 217 =12		
SB 290	RTR PDO3	Buffer is full: SI 218 =12		
SB 291	RTR PDO4	Buffer is full: SI 219 =12		
SB 292	Send NMT MC	Buffer is full: SI 221 =8		
SB 293	Send RTR NMT	Buffer is full: SI 222 =12		

Reset PLC, SB 300

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 300	Reset PLC Note that SB24 performs Reset + Init.	By program or Remote Access	Reset is run	OS

Backup Security PLC, SB 303

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 303	Backup Security in Memory (values stored for SB 314 Block PC access to PLC, SI[253] Info Password value, SI [50] Info Mode press time)	By program or Remote Access	Reset is run	OS

Buzzer, SBs 310, 311

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 310	Buzzer Turn this ON to sound a buzzer Buzzer included only in Touch-screen only models	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 in relevant models	By user, ON by default in V290/280	By user	User

Backup security. SB 314

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 314	Blocks PC access to PLC	By user	By user	User

(VisiLogic, RA, etc.)	(Default)

SD File Utilities, SBs 324-29

SB 324	SD: Open File (Read to SD) (Status messages in SI 67)	When Ladder function SD File: Open successfully activates a file for Read	When Ladder function SD File: Close finishes closing an open file and SB 327 (EOF) turns ON	OS. At Power-up and at SD File: Close
SB 325	SD File: Read Chunk in Progress (a Chunk is 512 bytes long)	When the Ladder function SD: Get Next File Chunk is reading a chunk into a vector	When the Ladder function SD: Get Next File Chunk has finished reading the chunk	OS. At Power-up
SB 326	SD Read File: End Of File (EOF, entire file has been read)	When the When the Ladder function SD: Get Next File Chunk reads the final Chunk	When the last chunk has been read, and when Ladder function SD File: Close start	OS. At Power-up and at SD File: Close
SB 327	SD: Open File (Write to SD) (Status messages in SI 67)	When Ladder function SD File: Open successfully activates a file for Write on a SD card		
SB 328	SD File: Write Chunk in Progress (a Chunk is 512 bytes long)	When the Ladder function SD: Get Next File Chunk is writing a chunk into a vector		
SB329	SD Write File: End Of File (EOF, entire file has been read)	When the When the Ladder function SD: Get Next File Chunk writes the final Chunk		

Break from External Device, SBs 330-332

Each COM port is linked to an SB that monitors communication signal breaks. Note

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 330	Break from External Device, COM Port 1	PLC receives break from external device	PLC has finished processing tasks related to break. If no messages are received after the break, the PLC resets the SB after 5 seconds. Each message received causes the PLC to wait for 40 seconds before resetting the SB.	OS
SB 331	Break from External Device, COM Port 1			
SB 332	Break from External Device, COM Port 1			

GPRS status, SBs 334-336

Each COM port is linked to an SB indicating GPRS communication status. These can be used as a condition to sending new messages.

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 334	GPRS 'Active': COM Port 1	Port is transmitting or receiving GPRS signals	Port is free	OS
SB 335	GPRS 'Active': COM Port 2			
SB 336	GPRS 'Active': COM Port 3			

Break from Modem, SBs 337-339

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 340	Log to SD in Progress	Row is being copied from DT to SD Card	When copy is complete	OS
SB	Copy Data Table from	Entire Data Table is being	When the Write process	OS

341	**PLC to SD** in Progress	copied from DT to SD Card	is complete	
SB 342	Copy Data Table from **SD to PLC** in Progress	Entire Data Table is being copied from SD Card to DT	When the Write process is complete	

SD Card DT and Log Functions, SBs 340-342

SB 340	Log to SD in Progress	Row is being copied from DT to SD Card	When copy is complete	OS
SB 341	Write Data Table from **PLC to SD** in Progress	Entire Data Table is being copied from DT to SD Card	When the Write process is complete	OS
SB 342	Read Data Table from **SD to PLC** in Progress	Entire Data Table is being copied from SD Card to DT	When the Write process is complete	OS

SD Card DT and Log Functions, SBs 343-345

SB 343	File Report in Progress	While Report process is in progress	When the Report is complete	OS
SB 344	Write delimited line to SD in Progress	While line is being written	When the Write process is complete	OS
SB 345	Read Data Table from **SD to PLC** in Progress	Entire Data Table is being copied from SD Card to DT	When the Write process is complete	

SD Card Data Block Functions, SBs 346-49

SB 346	SD Data Block 0 Busy	When a Write or Read utility is being run on a Data Block	When no utility is running	OS
SB 347	SD Data Block 1 Busy			
SB 348	SD Data Block 2 Busy			
SB 349	SD Data Block 3 Busy			

SD Card Alarms to SD, SB 352

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 352	SD: Enable writing Alarm History to SD	Turned ON by user. Causes the PLC to write entire Alarm History to the SD Card	Off by default. PLC stores Alarm History to memory.	At Power-up, or by user

SD Card Delete File, SB 358

SB 358	SD: Delete File in Progress	ON when function is busy	OFF when function is not busy	OS

SD Card File Info, SB 359

SB 359	SD: File Info function in Progress	ON when function is busy	OFF when function is not busy	OS

SD Card Clone in Progress, SB 366

SB 366	SD Clone in Progress Note that the process can take from several seconds to several minutes.	Turns ON when a Clone file is being created OR installed	Turns OFF : • When Cloning process is complete • Power up • SB 217 SD Card Present turns ON	OS

Data Tables, SB 500

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 500	User RAM overlap warning	When the application requests more memory than the PLC currently has free: Function Blocks memory requirements may have exceeded free memory. Data Table requirements exceed free memory. During DT Write: If the value of the pointer to DT is greater than the number of DT lines	Requirements fall with memory capacity	PLC initialization When a password is assigned to a VisiLogic project

Utility Error SB 399

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 399	Ladder utility failure	A Ladder utility fails to function. When SB399 is ON, the number of the failed utility is found in SI 26	At Power-up	User Program

Retain Inputs Forced Value, SB 501

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 501	Retain Inputs Forced Value after power failure Set SB 501at power-up to retain the state of inputs that are forced to 0 or 1 when the power is turned off. Reset SB 501at power-up to initialize forced inputs	By User	By User	User, or when PLC is initialized

System Integers

General, SIs 0-14

#	Description	Value	Comments
SI 0	Scan Time, Resolution: Units of 1 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	Relevant for V120, V130, V280, V290. LCD contrast is set for V230, V260 via potentiometer.
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 - 0ff	Note that this is relevant for CSTN

		1 - On (low intensity) (V230 only) 2 - On (max. intensity) - Default	displays only.
SI 14	Current controller temperature (not supported by V120/130/350)		Includes decimal point. For example, if the value is 245, the actual value is 24.5. The value in SI14 is only updated when SB 14 is turned ON by the user.

Ladder Utility Failure Indication, SI 26

#	Description	Value	Comments
26	Ladder Utility Failure Indication	Check table below	When SB 399 is ON, and any Ladder utility fails, SI 26 contains the utility number. Any time a utility fails, SI 26 is overwritten. Note that resetting SB 399 initializes SI 26.

Function	Value
Math: Linearization	1
Math: Factor	2
Vector: Fill	3
Logic: Shift Right	4
Logic: Shift Left	5
Logic: Rotate Right	6
Logic: Rotate Left	7
Vector: Bit to Numeric	8
Vector: Numeric to Bit	9
the function Config	10
Math: Power	11
Math: Square Root	12
Vector: Get Minimum (value in vector)	13
Vector: Get Maximum (value in vector)	14
Vector: Find Bit (in vector)	15
Vector: Copy	16
COM: Init function	17
Logic: Test Bit (in vector)	18
Logic: Set Bit (in vector)	19
Logic: Reset Bit (in vector)	20
Vector: Load	21
Vector: Store	22
Vector: Compare	23
Vector: Compare (Offset)	24
Vector: Copy (Offset)	25
Vector: Fill (Offset)	26
Data Tables: Read	27
Data Tables: Write	28
Data Tables: Read Line	29
Data Tables: Write Line	30
HMI: Display Loaded	31
HMI: Load Last HMI Display	32
/*EthernetUtils.OnlyfortheV200Series*/	
TCP/IP COM Init	33
TCP/IP Socket Init	34
Connect: TCP	35
Close: TCP	36
Math: Float Trig SIN	37
Math: Float Trig COS	38
Math: Float Trig TAN	39
Math: Float Trig ASIN	40
Math: Float Trig ACOS	41

Math: Float Trig ATAN	42
Math: Float Extended EXP	43
Math: Float Extended LN	44
Math: Float Extended LOG10	45
Math: Float Extended Power	46
Math: Float Extended Square root	47
Math: Float Basic Add	48
Math: Float Basic Subtract	49
Math: Float Basic Multiply	50
Math: Float Basic Divide	51
Math: Float Convert A+B/n	52
Math: Float Convert A+B/n (decimal)	53
Math: Float Convert Inverse (A+B/n)	54
Math: Float Convert Inverse (A+B/n) (decimal)	55
Math: Float Basic ABS	56
Math: Float Trig Degrees	57
Math: Float Trig Radians	58
Math: Float Compare Greater Than	59
Math: Float Compare Greater Equal	60
Math: Float Compare Equal	61
Math: Float Compare Not Equal	62
Math: Float Compare Less Than	63
Math: Float Compare Less than Equal	64
Float Store	65
COM: Dial	66
COM: Hangup	67
Strings: Transpose	68
Vector: Copy Memory	70
Strings: Num to ASCII	71
HMI: Draw Pixel	73
HMI: Draw Line	74
Logic: RS FlipFlop	75
Logic: SR FlipFlop	76
Time To ASCII	77
Clear Table Row column, Clear All DB	78
Online Point	79
Vector: Shift	80
Data Table: Find	81
Math: Float Compare In Range	82
Vector:Load Timer Scan Bit	83
More: Immediate Read Physical Input	84
Immediate: Write to Physical Output Digital or Analog	85
More: Immediate Update Physical High-speed Input	86
Strings: IP to ASCII	87
COM: Set PLC Name	88
Data Table: Find Extended	89
HMI: Clear Rectangle	90
More: Debug Interval Start	91
More: Debug Interval End	92
Strings: Time to ASCII	93
Store: Num to BCD	94
Store: BDC to Num	95
Clock: RTC to UTC	96
Clock: UTC to RTC	97
Math: Linearize Vector	98
Data Table: Read Line	99

Data Table: Write Line	100
Immediate: Write to Physical Analog Output	101
Com TCP/IP RFC 1305	102
Math: Formula	103
String: ASCII to Num	104
Data Table: Copy Row	105
Data Table: Copy Column	106
Strings: MAC Address to ASCII	107
Vector: Struct	108
Com: Send UniCAN	109
Data Table: Clear Row	110
Data Table: Clear Column	111
Com: UniCAN Check Alive	112
Com: UniCAN Broadcast	113
Vector: Swap Bytes	114
Com: UniCAN Message Arrived	115
Logic: RLO to Bit	117
Vector: Sort	118
Com: CANopen Configuration	119
Com: CANopen Send PDO	120
Com: CANopen Send RTR PDO	121
Com: CANopen Send SDO	122
Com: CANopen Send NMT Control	123
Com: CANopen Send NMT Node Guard	124
Com: CANopen Send Synch	125
Com: CANopen	126
Com: CANopen Send Download STR	127
Com: CANopen Send Upload STR	128
Vector: String Length	129
HMI: Is HMI Last Displayed	130
Com: TCP Send RAW UDP	131
Com: TCP Receive RAW UDP	132
Com: CAN Layer 2 Send	133
Com: CAN Layer 2 Receive	134
Last Call Received	I o135
More: Idle	136
Alarms: Show Groups	137
Alarms: Show Alarms	138
Strings: Set String Library	139
Strings: Timer to ASCII	140
Strings: Lib.Str. to ASCII	141
Vector: Toggle Bit in Vector	142
Vector: Toggle Bit in Vector	143
Step In Range	144
Strings to ASCII	145
Strings: to NUM	145
Strings: remove	146
Strings: insert	147
Strings: left	148
Strings: right	149
Strings: middle	150
Strings: find_in_str	151
COM: Set PLC Network ID	152
Strings: Timer to ASCII	153
HMI: Refresh HMI Display	154
Data Tables: Read/Write Column	155

Reserved	156
SD: Write DT to SD .udt	158
SD: Log DT row to SD	159
Email	160
Alarms: Clear History Buffer	161
COM: Send TCP RAW	162
COM: Scan TCP RAW	163
SD: Set SD Password	164
String: Replace STR A with STR B	165
SD: Save Trend to SD	166
SD: Stop Trend to SD	167
SD: Read from .udt SD to DT	168
Immediate: Frequency Measurement based on HSC	169
Reserved	170
Immediate: Stop Frequency Measurer	171
Immediate: Reset HSC	172
COM: CANopen: Map register bytes	173
SD: Read SD.udt to Data Table	174
CAN J1939: Config	175
CAN J1939: Activate cyclic send	177
CAN J1939: Send pgn	178
Data Tables: Read row	179
Data Tables: Write row	180
COM: TCP/IP: Ping	181
SD: Read SD Block (udb) to Operand Vector	182
SD: Write from Operand Vector to SD Block (udb)	183
SD: Create Block	184
SD: File Read: Open	185
S SD: File Read: Next Chunk	186
SD: File Read: Close	187
CAN J1939: Request PGN Data	188
COM: Df1 Scan	189
COM: SNMP Trap	190
SD: Create Delimited Line	191
SD Write: open file	192
SD Write: read file	193
SD Write: close file	194
SD: Write Delimited Line	195
SD Folder Report: Number of Files	196
SD File: Delete file	197
SD: Write DT to SD .udt	198
SD: Log DT row to SD	199
SD Files: File Info	200
SD Data Tables: Search .udt for key	201
COM: Telegram parser	202
COM: CANLayer2 ScanEx	203
COM: Set SNMP community name	204
Backup security operands	205
SD: Safely Remove	206
SD: Clone	207
COM: DNS Resolver	208
Data Tables: Delete Rows	209
SD: Rename SD File	210

Real Time Clock, SIs 30-37

#	Description	Value	Comments
SI 30	Current second	0-59	According to RTC
SI 31	Current time	24 hour clock: 14:59 = 1459	
SI 32	Current date	12/07 = 12th of July	

SI 33	Current year	2007=2007
SI 34	Current day of week	1-7
SI 35	Current hour	0-23
SI 36	Current minute	0-59
SI 37	Current day of month	30 = 30th day of month

Touch Coordinates, SIs 40, 41

#	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

Keypad Entry Out of Limits, SI 45

#	Description	Value	Comments
SI 45	Numeric Key Entry Out of Limit - Counter of Attempts	Counts the number of failed attempts to enter a value, such as a password	Enhanced Vision only If a Legal Entry bit is defined, SB 94 does not turn ON if the entered value is out of range. The keypad stays on screen until a legal value is entered. You can use SI 45 in conjunction with a Compare function to exit the variable.

Refresh HMI, Buttons, Frame, Text , SI 46

#	Description	Value	Comments
SI 46	Refresh HMI Buttons, Frame, Text	Units 10 msec, redraws these items in current display to reflect changes	

Select Touch Keyboard Type (enhanced only), SI 49

#	Description	Value	Comments
SI 49	Select Touch Keyboard Type	0, 1, 257	See the topic Vision Controller Divisions, Special Issues, Virtual Keypads: Enhanced Touchscreen Models

INFO delay time, SI 50

#	Description	Value	Comments
SI 50	INFO delay time	Default by O/S (every power up) = 4 seconds	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
SI 51	Info Mode, Serial COM Monitor: # of messages not displayed	Number of messages not displayed. Initialized every time the Monitor is entered	When entering the monitor, the display must synchronize with the actual messages in real time. This SI contains the number of messages that are not displayed before synchronization is complete.

email Limit File Attachment Size, SI 58

#	Description	Value	Comments
SI 58	email: Limit File Attachment Size	1=1024 bytes	Power-up default is 1 Maximum per attachment = 10 (10 MB)

			Note that the file size must not be changed while the Send is in Progress.

Max number of *.udt files saved to SD, SI 63-64

#	Description	Value	Comments
SI 63	Maximum number of Trend files that can be saved (read-only)	0-64 The maximum amount of Trend files (*.udt files) in a single folder is 64. The value in SI 63 shows the number of remaining *.utr files; if 5 *.utrt files exist, SI 64 = 59	Initialized at Power-up Updated when:SB 217 is ON and SB 341 turns ON
SI 64	Maximum number of DT files that can be saved (read-only)	0-64 The maximum amount of Trend files (*.udt files) in a single folder is 64. The value in SI 64 shows the number of remaining *.udt files; if 5 *udt files exist, SI 64 = 59	Initialized at Power-up Updated when:SB 217 is ON and SB 341 turns ON

SD Card Status Messages, SI 66

#	Description	Value	Comments
SI 66	SD Card Status Messages	This SI is a bitmap; a bit turns ON to indicate status. All bits OFF – No errors Bit 1 – Read: End Of File indication Bit 2 – Can't open file Bit 3– Error while writing to a file Bit 4 – Error while reading from a file Bit 5 – Failed to close a file Bit 6 – SD is full Bit 7 – Path not found Bit 14 - Turns ON when SD is inserted into slot and PLC runs checks, turns OFF when SB 217 turns ON	Initialized at Power-up. While the application is running, the user application must reset the bits.

SD Card Read/Write Files, SIs 67, 68

#	Description	Value	Comments
SI 67	SD Card, Read Files: Status	Value 0= No error 1= No SD card in Slot 2= Vector is not long enough to contain data (may be at upper address limit of that data type) 3= Path to SD file not found 4=Another file is currently open 5 = File is closed 6 = Busy: previous request in progress 7 = File Open Error	SI 67 reports status for the following SD File utilities: • Read SD File: Open • Read Next File Chunk • Read SD File: Close

		8 = Read Error 9 = File Close error	
SI 68	SD Card, Write Files: Status	Value 0 = No error 1 = No SD card in Slot 2 = Vector is not long enough to contain data (may be at upper address limit of that data type) 3 = Path to SD file not found 4 = Another file is currently open 5 = File is closed 6 = File Open error 7 = Write Error 14 = File Close error	SI 68 reports status for the following SD File utilities: • Write SD File: Open • Write Next File Chunk • Write SD File: Close

SD Card File Open Time, SI 69

#	Description	Value	Comments
SI 69	SD Card: File Open Time (may signal file fragmentation)	Time required to open SD files, in units of 10mSec.	Each time a file is opened, the OS updates this value. A typical first write (open + write) =approx. 500mSec, typical first read (open + read)= approx. 60mSec Over time, this may increase due to file fragmentation. If the time becomes to great, the card should be reformatted Reset at Power-up and when SD card is removed.

SD Trend Status, SIs 160-167

#	Description	Value	Comments
SI 160	SD Trend 1 status	This SI is a bitmap; a bit turns ON to indicate status when the function Start Saving Trend to SD runs. All bits OFF – No errors Bit 4 – Start Saving Trend is in progress for another Trend Bit 7 – This Trend does not exist (may result when an MI is used to provide the Trend number, and the value points to a non-existent Trend) Bit 8 – Start Saving Trend is in progress for this Trend Bit 9 – Start Saving Trend failed	
SI 161	SD Trend 2 status		
SI 162	SD Trend 3 status		
SI 163	SD Trend 4 status		
SI 164	SD Trend 5 status		
SI 165	SD Trend 6 status		
SI 166	SD Trend 7 status		
SI 167	SD Trend 8 status		

SD DT blocks to/From SD, SIs 330 -333

#	Description	Value	Comments
SI	SD: Write DT from PLC	When the application runs	Initialized at Power-up

330	to SD - Total Amount of Data to be Copied (blocks of 512 bytes)	the function Copy Data Table to SD, SI 330 shows the total number of blocks of data **to be copied** from the PLC.	
SI 331	SD: Write DT from PLC to SD - Remaining Amount (blocks not yet copied)	Shows how many blocks of data **remain** to be copied. The value increases by 1 each time a block is copied.	Initialized: When the PLC begins to copy a new block of data to the SD card At Power-up.
SI 332	SD: Read DT SD to PLC - Total Amount of Data to be Copied (blocks of 512 bytes)	When the application runs the function Copy Data Table to PLC, SI 332 shows the total number of blocks of data **to be copied** from the SD.	Initialized at Power-up
SI 333	SD: Read DT from SD to PLC - Remaining Amount (blocks not yet copied)	Shows how many blocks of data **remain** to be copied. The value increases by 1 each time a block is copied.	Initialized: When the PLC begins to copy a new block of data from the SD card At Power-up.

FLASH Storage, SI 72

#	Description	Value	Comments
SI 72	FLASH Storage Bitmap (V570, 290-C)	A bit is on when data is present Bit 0 = Data backup from RAM to FLASH Bit 1 = Upload data (.vlp in PLC can be uploaded) Bit 2-7 = Internal Bits 8-15 = String storage The SI is updated by the OS after every download.	FLASH informations is divided into sections. The status the bits shows if data is stored in these sections. Relevant to V570, V290-C.

Alarms: Status, SI 74

#	Description	Value	Comments
SI 74	Alarms Utility, General Status	A bit is on when data is present Initialized at power-up Bit 0 = Alarm Version in OS does not match VisiLogic version Bit 1 = No Alarms defined Bit 2 = Internal error Bit 8 = History buffer full	

Operand assignment error, SI 75

#	Description	Value	Comments
SI 75	Operand assignment error	0 = no error Any other value = Error	Number of functions assigned operands that are of illegal address or type. Relevant to V570, V290-C.

Number of Alarms in History, SI 76

#	Description	Value	Comments
SI 76	Number of Alarms currently in History Buffer	Shows the number of Alarms in the PLC memory buffer	

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,**

SI 81	Error Code: COM 1
SI 82	Modem Status: COM 2
SI 83	Error Code: COM 2
SI 84	Modem Status: COM 3
SI 85	Error Code: COM 3

82, 84)

Value	Message	Value	Message
0	No error	0	Modem Idle
1	TimeOut exceeded: no reply	1	Initialization in Progress
2	Reply Error	2	Initialization OK
3	Wrong PIN number	3	Initialization Failed
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		
16	Error in reply to PIN number		
17	Check: CREG failed		
18	Check: CREG timeout		
19	Check: slots timeout		
20	Check: Format timeout		

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

SI	Description	Value
100	Maximum Time Delay between characters (units 2.5ms) MODBUS + Modem	When MODBUS (Serial) is configured, 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). Note that: - The power-up value is 1, - the application must update SI 100 before the MODBUS configuration is activated.

Ethernet-enabled controllers only, SIs 101-148

#	Description	Value	Comments
SI 101	TCP/IP retries base time out	Legal values are 1 to 10, units of 100 msec (1 stands for 100 msec etc.) Default value is 200 msec.	Same value is for ALL 4 sockets Requires CARD INIT Illegal value request will be rejected (no change)
SI 102	Retries count	Legal values are from 1 to 50 Default value is 6	Illegal value request will be rejected (no change)
SI 103	TCP/IP Connection Keep Alive (Socket 0)	Units of 100 msec Note- When value is '0', the function is disabled	Enables the PLC to disconnect if there is no communication from the connected device. When TCP/IP connection is established (SI 145-148 = 6) check data transport (SDW 14 – 21). If no data transport occurred during the defined time – perform 'Socket Init'.
SI 104	TCP/IP Connection Keep Alive (Socket 1)		
SI 105	TCP/IP Connection Keep Alive (Socket 2)		
SI 106	TCP/IP Connection Keep Alive (Socket 3)		

			Data Transport counter is SDW 38
SI 107	TCP/IP Keep Master Connection (Socket 0)	Units of 100 msec Note- When value is '0', the function is disabled	Enables the PLC to reconnect the connect when there is no communication from the connected device for the defined time. Note that the value per socket should be higher the regular "keep alive" (SI 103 – 106) Counter of the operation in SDW 45
SI 108	TCP/IP Keep Master Connection (Socket 1)		
SI 109	TCP/IP Keep Master Connection (Socket 2)		
SI 110	TCP/IP Keep Master Connection (Socket 3)		
SI 140	Ethernet Send has failed, per socket (bitmap)	Bit is ON when Send is not successful	Bitmap: | UDP S3 | UDP S2 | UDP S1 | UDP S0 | TCP S3 | TCP S2 | TCP S1 | TCP S0 |
SI 141	Ethernet Socket 0: Protocol Type	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
S1 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, 188, 191

SI	Description	Value
185	GSM Signal Quality (V120/230/260/280/290-BW)	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.
188	GSM Signal Quality COM2 (V570, 290-C)	
191	GSM Signal Quality COM3 (V570, 290-C)	

SMS messages: Write to Vector SIs 198-199

Use these together with SB 198 and 199 to write incoming SMS messages to a vector of operands. This does not affect the function of the SMS message function blocks.

SI#	Description	Value	Comments
SI 198	Received SMS Message Length	Shows the length of the message in bytes	The data remains until the vector is overwritten
SI 199	SMS to Memory Vector - start of vector	The SMS message data is written starting from this address	To write to a vector of XIs, enter a negative value)

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

The function of some operands depends on whether the CANbus network is defined as CANbus ISC, CANopen or UniCAN.

When using CANopen

SI#	Description	Value	Comments
SI 211	CANopen: Number of received messages	Shows the number of received messages in the Receive buffer	Maximum number of messages=128

		(except for SDOs)	
SI 212	CANopen: Number of Send PDO1	Shows the number of PDO1 messages currently in the PDO1 Send buffer	
SI 213	CANopen: Number of Send PDO2	Shows the number of PDO2 messages currently in the PDO2 Send buffer	
SI 214	CANopen: Number of Send PDO3	Shows the number of PDO3 messages currently in the PDO3 Send buffer	
SI 215	CANopen: Number of Send PDO4	Shows the number of PDO4 messages currently in the PDO4 Send buffer	
SI 216	CANopen: Number of Send RTR PDO1	Shows the number of RTR PDO1 messages currently in the PDO1 Send buffer	
SI 217	CANopen: Number of Send RTR PDO2	Shows the number of RTR PDO2 messages currently in the PDO2 Send buffer	
SI 218	CANopen: Number of Send RTR PDO3	Shows the number of RTR PDO3 messages currently in the PDO3 Send buffer	
SI 219	CANopen: Number of Send RTR PDO4	Shows the number of RTR PDO4 messages currently in the PDO4 Send buffer	
SI 220	CANopen: Number of Send SDOs	Shows the number of SDO messages currently in the Send buffer	
SI 221	CANopen: Number of Send NMTs	Shows the number of NMT module control messages currently in the NMT Send buffer	
SI 222	CANopen: Number of Send RTR NMTs	Shows the number of RTR NMT messages currently in the Send buffer	
SI 223	CANopen: Send Buffer full (per type)	The bits in this register represent the different Send buffers (except for SDOs)	Maximum number of messages per buffer=8

When a bit is ON, the corresponding buffer is full.
High byte:| - | - | - | - | - | - | -| NMT mc |
Low byte: | PDO | -| RCV NMT | RCV PDO4| RCV PDO3 | RCV PDO2 | RCV PDO1 | RCV Emergency |

SI 224	CANopen: Number of received SDO messages	Shows the number of received SDOs currently in the Receive buffer	
SI 225	CANopen: SDO status	The status codes are given below.	

Value	**Message**
0	No error
1	PLC in STOP mode
2	CANopen not configured, SB 241 is not set (after configuration)
3	Remote ID is 0
4	Maximum SDO Upload length set to 0
5	SDO in Progress; Download/Upload started while SB242 is ON
6	SDO in Progress Error; SB242 turned OFF during data transfer (system problem)
7	Illegal Operands used in SDO data transfer
8	Number of operands in data type exceeded
9	Process buffer not cleared before SDO Send (system problem)
10	Response Timeout exceeded
11	Receive Error
12	Reserved by CIA
13	Receive Buffer full; more than 127 segments in a block (system problem)

14	Receive Error Toggle bit ON (error in domain segment)
15	Receive domain segment Abort; error code given in SDW 34
16	Byte number error
17	Number of bytes is zero
18	Number of bytes exceeds the maximum upload length
19	Machine State error (system problem)
20	Receive Error in block size transferred from the remote device
21	Send Timeout exceeded
22	Sequence error in the number of segments in block transfer
23	CRC error, block transfer

When using CANbus ISC

SI 200	**CANbus Network operand**	
SI 201	CANbus Network operand	
SI 236	CANbus Network communication error code	**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 SI 241 SI 242 SI 243	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	

When using UniCAN

SI#	**Description**	**Comments**
200	When a UniCAN Broadcast message is received, SI 200 contains the ID number of the sending unit.	The user must initialize these SIs
201	When a UniCAN Broadcast message is received, SI 201 contains the value of the MI that is broadcast.	
202	Number of Send messages waiting in High Priority buffer	Automatically updates
203	Number of Send messages waiting in Low Priority buffer	
204	Number of Received messages waiting in buffer	
240 241 242 243	These provide a bitmap of controllers 1-60 in the UniCAN network.	When the controller receives a message, the appropriate bit turns ON. These bits are reset by the Answer Received function.

X,YCoordinates, Num Keypad/Alarm screen, SI 244, 245

#	**Description**	**Value**	**Comments**
SI 244	V1040, X Coordinates, Num Keypad/Alarm screen	Enter the X value to alter the location of these elements on the screen.	Values remain until changed by user
SI 445	V1040, Y Coordinates, Num Keypad/Alarm screen	Enter the Y value to alter the location of these elements on the screen.	Values remain until changed by user

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	Currently active keypad entry, read/write.

	entry, read/write	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. **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.
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	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. 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.

Messages Received Counters, SI 274-276

SI 274	COM1, Received Message Counter	Counts received messages, increments after message is validated. Initialized by OS at power-up
SI 275	COM2, Received Message Counter	
SI276	COM3, Received Message Counter	

Float Errors SI 440

SI 440	General Error SB 10 turns on when a Float Erro occurs.	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

OS Information SI 497, 498

SI 497	**Firmware Build Number**	**Contains the build number of the OS currently in the controller. (V120, V230, V260, V280, V290 BW)**
SI 498	Firmware Version Number	Contains the version number of the OS currently in the controller.(V120, V230, V260, V280, V290 BW)

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 9	Unique PLC ID number (All Visions)	Each PLC has its own unique ID number	Use SDW9 (unique PLC number) to restrict a program to a particular PLC.
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 transmissions	Updated after each data transmission via 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	Socket 3: Number of received transmissions	Updated after each data packet received via Socket 3	
SDWs that are common to UNICAN and CANopen change function, according to the CANbus type selected in the COM Init function.			
SDW 7	UniCAN, CANbus ISC Error	If not 0, contact technical support	
	CANopen: Number of failed Send attempts	Number of times that data send failed	
SDW 8	CANopen: Number of failed Sync attempts	Number of times that send SYNC failed	
SDW 56	UniCAN Send message counter	Is initialized when CANbus Port Init runs, then increments at every UniCAN Send.	Note that only messages sent from a UniCAN Send are counted
	CANopen: PDO Send Counter	Byte structure: PDO4 | PDO3 | PDO2 | PDO1	
SDW 57	UniCAN Receive message counter	Is initialized when CANbus Port Init runs, then increments at every UniCAN Receive.	Note that only messages received from a UniCAN Send are counted, not Broadcast messages or Check if Alive responses.
	CANopen: NMT/SDO Send Counter	High bits: NMT Low bits: SDO	
SDW 29	CANopen: Bus is OFF Counter	Number of times bus was OFF	
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 33	CANopen: SDO Number of Bytes	SDO upload: number of bytes received	

		SDO download: number of bytes sent	
SDW 34	CANopen: Abort Code in SDO Abort		

Value	Message
0503 0000h	Toggle bit not alternated
0504 0000h	SDO protocol timed out
0504 0001h	Client/server command specifier not valid or unknown
0504 0002h	Invalid block size (block mode only)
0504 0003h	Invalid sequence number (block mode only)
0504 0004h	CRC error (block mode only)
0504 0005h	Out of memory
0601 0000h	Unsupported access to an object
0601 0001h	Attempt to read a write only object
0601 0002h	Attempt to write a read only object
0602 0000h	Object does not exist in the object dictionary
0604 0041h	Object cannot be mapped to the PDO
0604 0042h	The number and length of the objects to be mapped would exceed PDO length
0604 0043h	General parameter incompatibility reason
0604 0047h	General internal incompatibility in the device
0606 0000h	Access failed due to a hardware error
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist
0609 0030h	Invalid value for parameter (upload only)
0609 0031h	Value of parameter written too high (upload only)
0609 0032h	Value of parameter written too low (upload only)
0609 0036h	Maximum value is less than minimum value
060A 0023h	Resource not available: SDO connection
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application
0800 0021h	Data cannot be transferred or stored to the application because of local control
0800 0022h	Data cannot be transferred or stored to the application because of the present device state
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error)
0800 0024h	No data available

SDW 36 CANopen: Bus OFF error

Value	Message
0	No error
1	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed
2	Form Error: Wrong format in fixed format part of a received frame
3	AckError: The message this CAN controller transmitted was not acknowledged by another node
4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level ("1"), but the monitored bus value was dominant
5	During busoff recovery this is set each time a sequence of 11 recessive bits is monitored. This enables the CPU to monitor the proceeding of the busoff

		recovery sequence (indicates that the bus is not stuck at dominant or continuously disturbed	
	6	CRC Error: The received CRC check sum is incorrect	
	7	Unused code: may be written by the CPU to check for updates	
SDW 37	MODBUS Slave: Receive Counter (Bitmap)	Increments a 4-bit field each time a slave receives data	High - >low: | Eth port 3| Eth port 2| Eth port 1| Eth port 0| |spare| | com 3| com 2| com 1|
SDW 38	TCP/IP Keep Alive counter	Increments a 8-bit field each time the O/S initializes the socket due to 'Keep Alive' (SI 103-106)	| Eth port 3| Eth port 2| Eth port 1| Eth port 0|
SDW 39	Ethernet general critical error	8-bit counters	Bits 24-31: "Ethernet card init" – if the MS Byte (xxx.yyy.zzz.kkk, MS means the xxx part) of the IP/ SUBNET/ GATEWAY is zero – do not init the Ethernet. Bits 16-23: Check once in second if local IP SUB and GATEWAY are ok. Cause set of SB 167. Bits 8-15: TCP (connect) & UDP (send) IP is defined – Read HW remote IP to verify. Case verifies failed: Ignore connect or send. Bits 0-7: While getting message from socket – if the high part of the remote IP high is zero. Cause set of SB 167
SDW 42	100mS Timer Counter, Stable	Counts number of pulses	Updates at beginning of program scan only
SDW 43	10mS Timer Counter, Stable	Counts number of pulses	Updates at beginning of program scan only
SDW 44	2.5mS Timer Counter, Stable	Counts number of pulses	Updates at beginning of program scan only
SDW 45	TCP/IP Keep Master Connection	Increments a 8-bit field each time the O/S initializes the socket due to 'Keep Alive' (SI 107-110)	| Eth port 3| Eth port 2| Eth port 1| Eth port 0|
SDW 59	SD Card: Free space (bytes)	Capacity given in 512-byte chunks. The value is written when SB 217 turns ON, and is updated at each write operation. The operand is reset when SB 217 turns OFF.	Initialized at Power-up.
SDW 60	Info Error Status	Error Indication	
SDW 63	Firmware version and Build number	Contains the version number of the OS currently in the controller	Relevant for V570, V290 Color

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

Logic Functions

Function blocks are provided for:

- Bit Test
- Set/Reset Bit
- AND
- OR
- XOR
- Shift
- Rotate
- Convert
- Test Bit
- RS-SR Flip-Flop
- RLO to Bit

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.

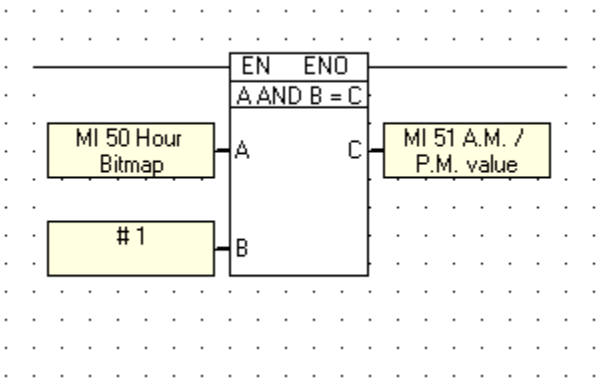
AND can be used to mask out certain bits of an input integer not relevant to a given function.

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).

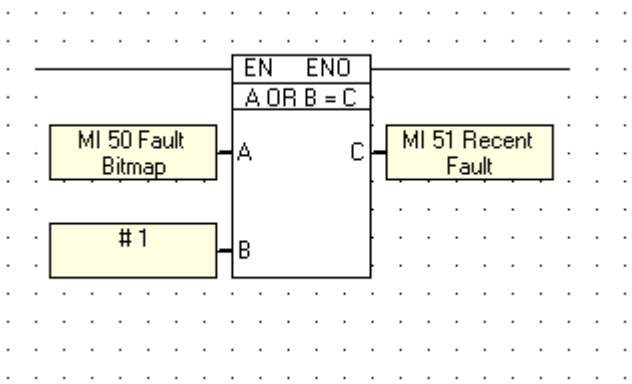
OR Truth Table		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

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

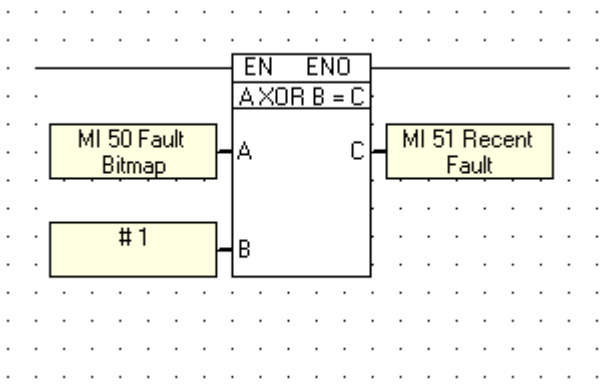
The input values in a XOR function may be:

- 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

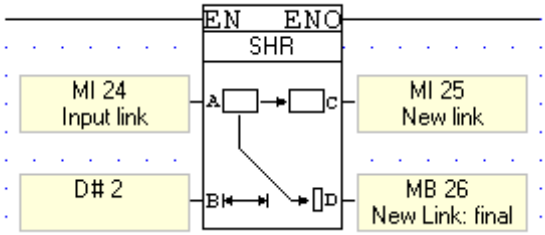


The function is located under the Logic menu on the Ladder toolbar.

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.

- 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 (last) bit in the integer after the operation.



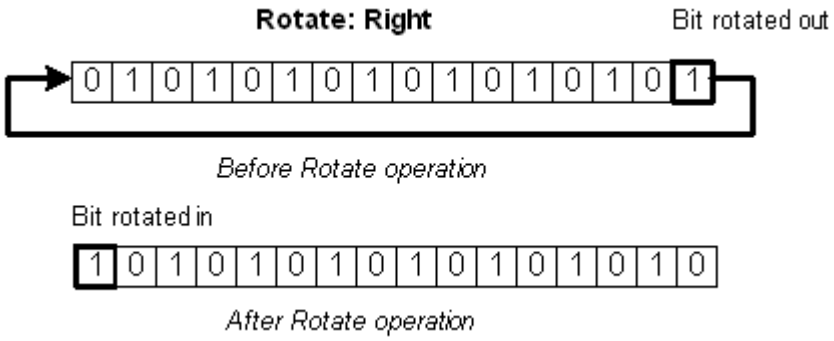
The Shift function may be performed on values contained in the following operands:

- 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.

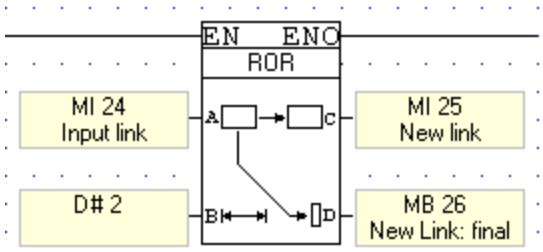
Rotate

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.

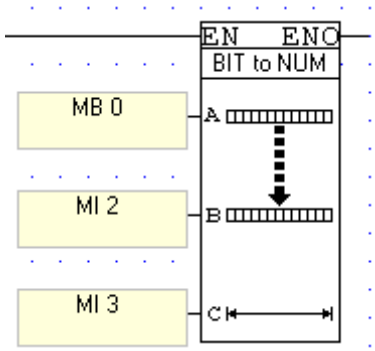
Vector: Bit to Numeric, Numeric to Bit

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

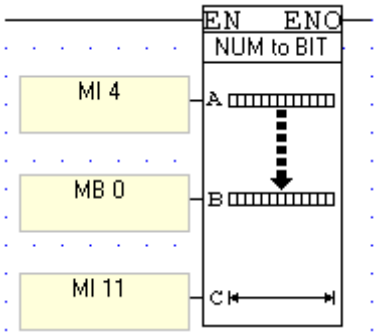
- Operand A: contains the Start Address for the array of bits to be converted.
- Operand B: is the start of the vector that will contain the converted value. Take care in addressing operands, since the converted value may not fit into a single register; the function will overwrite as many consecutive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

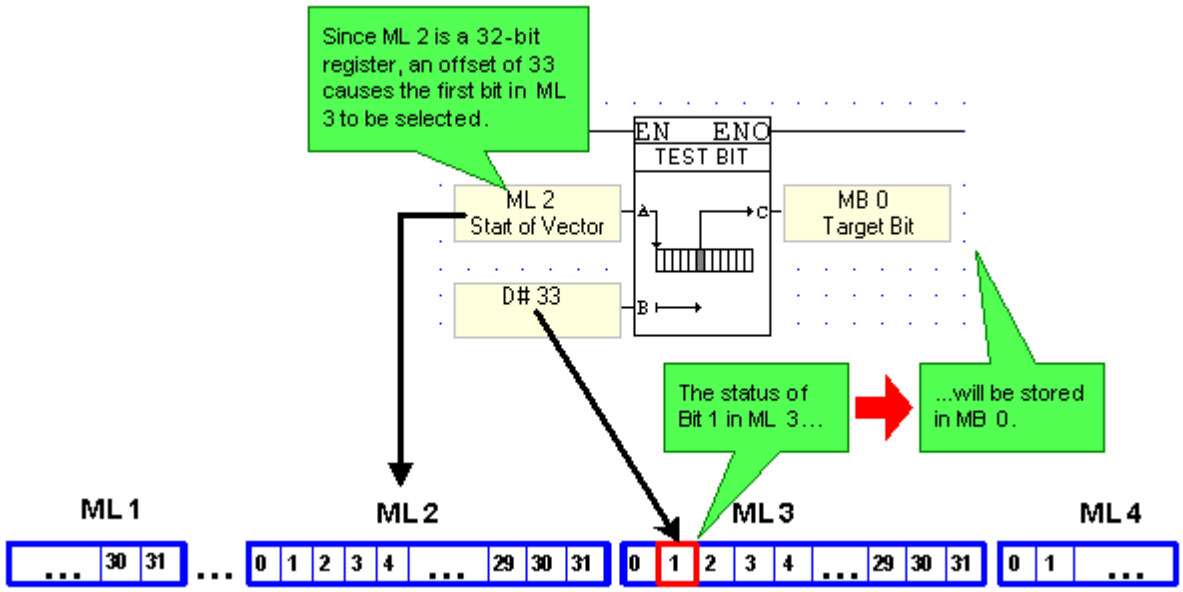
- Operand B: contains the Start Address of the bit array that will contain the converted value.
- Operand C: contains the Length of the bit array that will contain the converted value.



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.



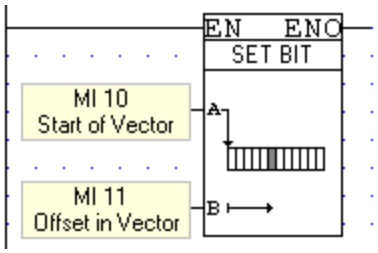
Note that the maximum number of bits in the vector is 255, 16 MIs or 8 double registers. The function is located under the Logic menu on the Ladder toolbar.

Set/Reset Bit

Set Bit enables you to select a bit within a register, and set it.

Reset Bit enables you to select a bit within a register, and reset it.

- Operand A, **Start of Vector**, is the register in which the function will set/rest the bit.
- Operand B, **Offset in Vector**, selects the bit within that vector.



Note that the maximum number of bits in the vector is 255, 16 MIs or 8 double registers. The functions are located under the Logic menu on the Ladder toolbar.

RS-SR Flip-Flop

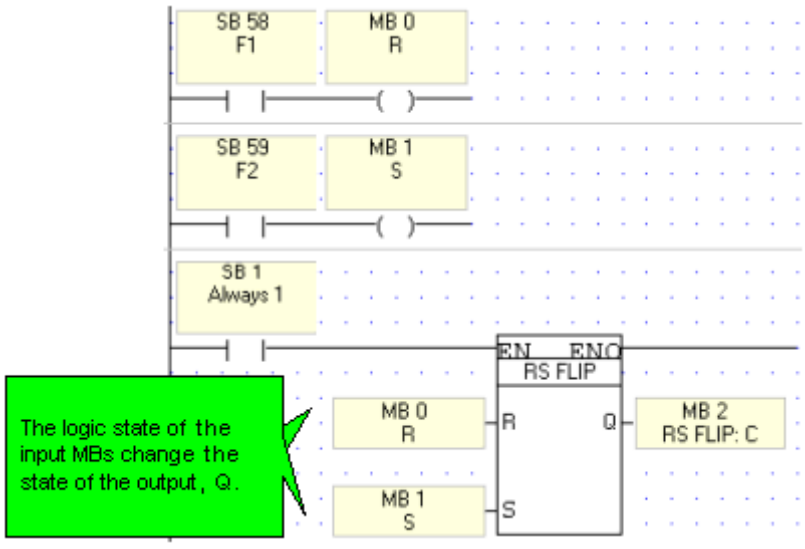
The RS and SR Flip-Flop functions are located on the Logic menu. These functions compare the logic state of two inputs, and use the result to determine an output result in accordance with the tables shown below.

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

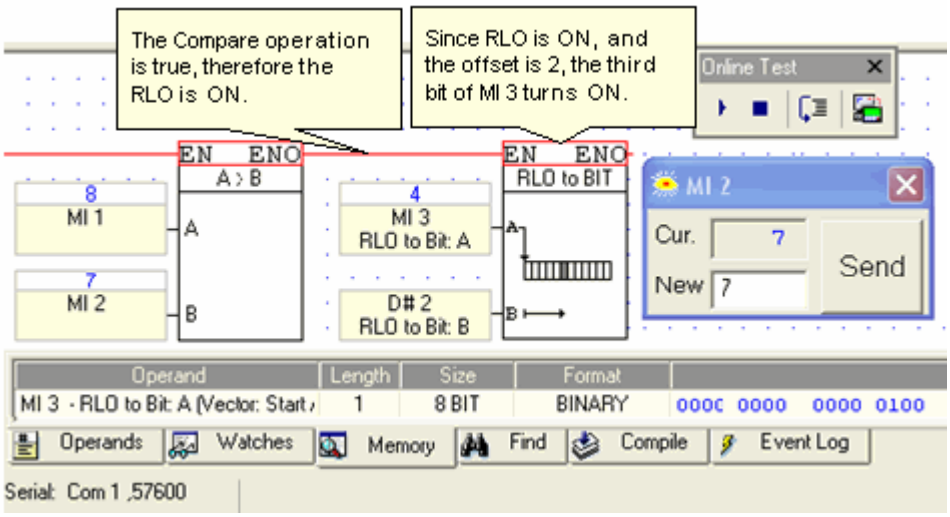


RLO to Bit

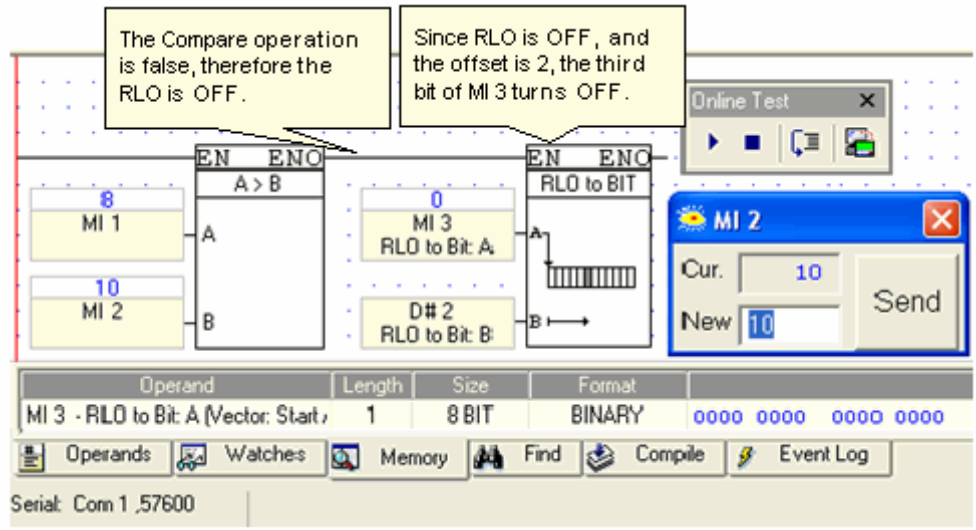
The PLC Ladder program is based on whether or not there is power flow through the logic rung. When there is power flow, the RLO, Result of Logical Operation, is positive, or ON. At the left-hand ladder rail, there is always power flow; therefore the RLO at the rail may be considered as ON. When there is no power flow, the RLO is negative, or OFF.

RLO to Bit takes the status of the RLO and stores it in a register bit according to the desired offset.

The rung in the following figure is shown in Online Test mode, showing the power flow in red. The Compare operation (MI1>MI2) in the rung is true. Therefore power flows through the rung, and the RLO is positive (ON). The state of the RLO is stored in MI 3, at an offset of 2 bits, in the third bit of MI 3. The bit turns ON, and MI 3 contains 4.



In the following figure, the Compare operation is false. Therefore power does not flow through the rung, and the RLO is negative (OFF). The state of the RLO is stored in MI 3, at an offset of 2 bits, in the third bit of MI 3. The bit turns OFF, and MI 3 contains 0.



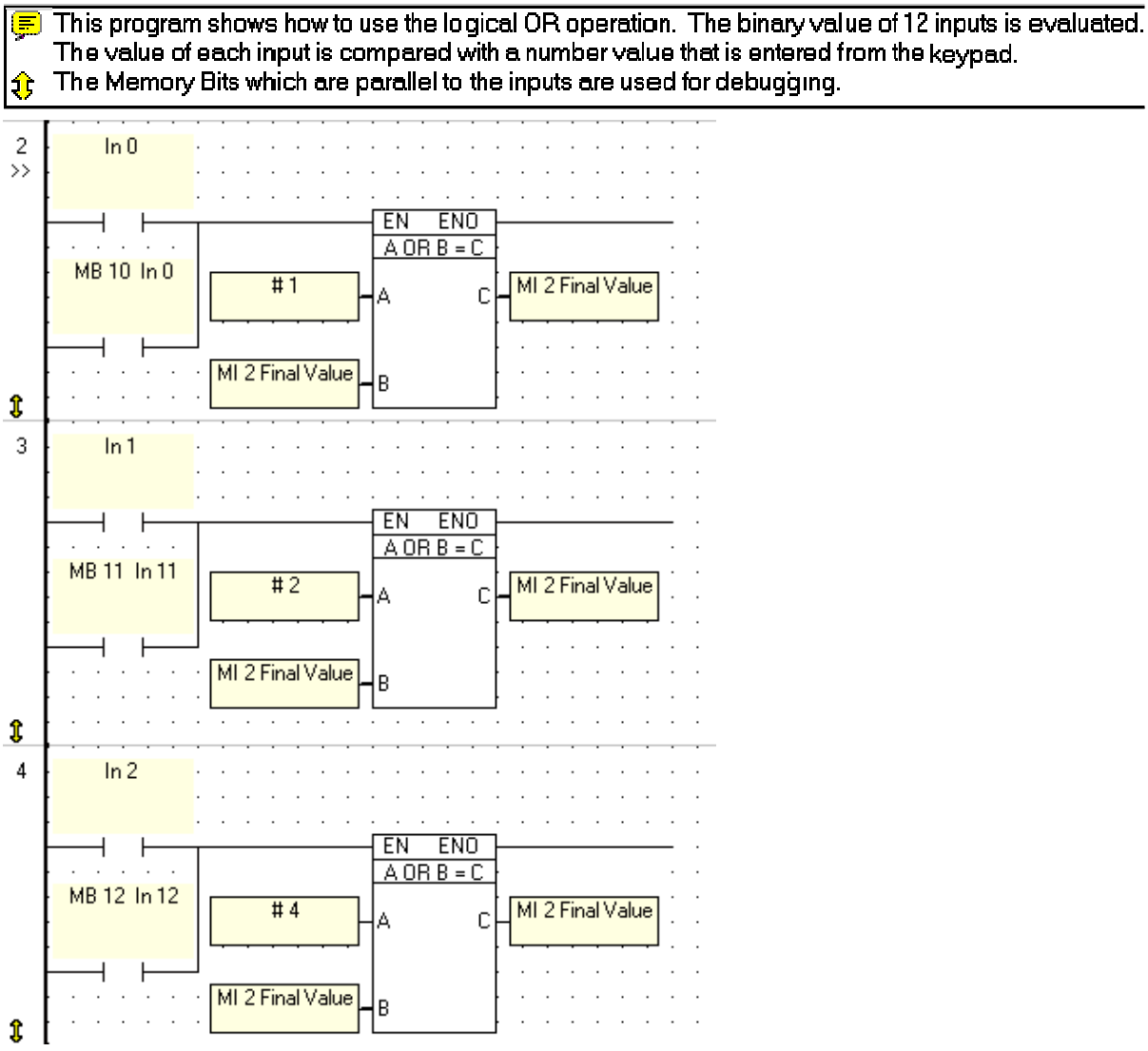
Note that the maximum number of bits in the vector is 255, 16 MIs or 8 double registers.

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.



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	0	1	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	0	1	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	0	1	0	0	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	0	1	**29**
0	0	0	0	0	0	0	0	1	1	1	1	0	**30**
0	0	0	0	0	0	0	0	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	0	0	1	0	**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	1	0	0	0	0	0	0	**64**

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 7 types of Compare Functions:

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

Note

- The Vector menu includes a Compare Vector function.

These values may be compared:

- Memory Integer **(MI)**

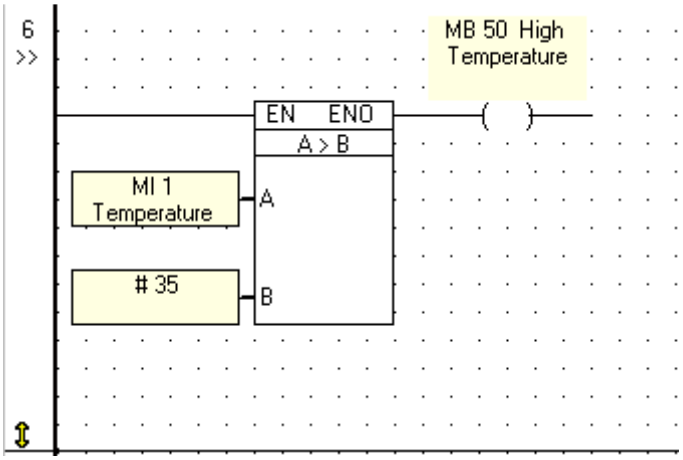
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**
- Network System Integer **(NSI)**
- Constant Value **#**
- Counter

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:

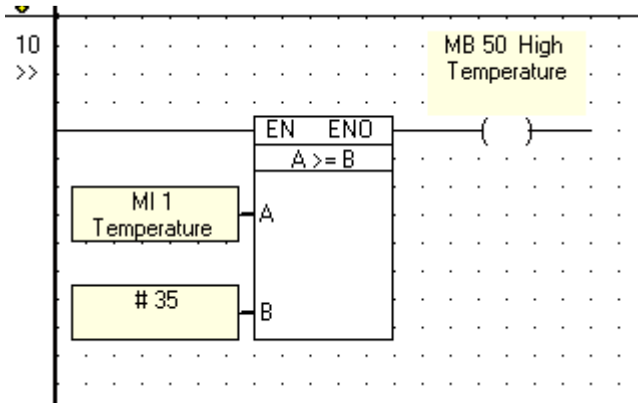
- Memory Integer **(MI)**
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**
- Network System Integer **(NSI)**
- Constant Value **#**

Greater or Equal to

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:

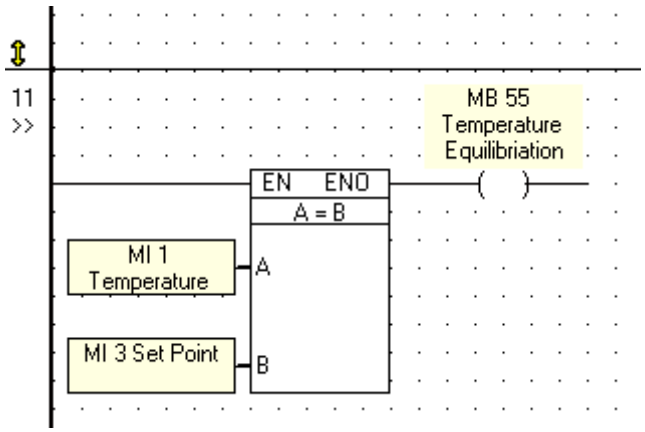
- Memory Integer **(MI)**
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**
- Network System Integer **(NSI)**
- Constant Value **#**

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:

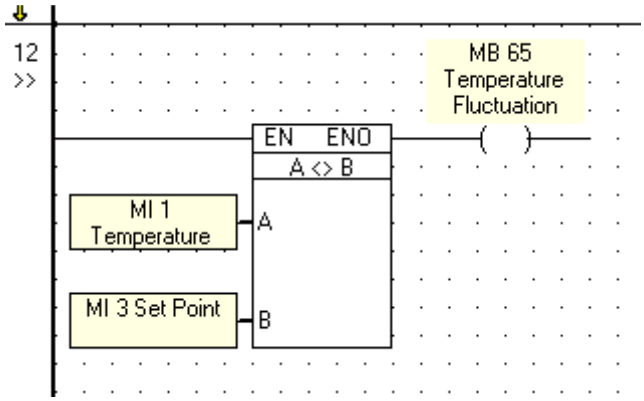
- Memory Integer **(MI)**
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**
- Network System Integer **(NSI)**
- Constant Value **#**

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:

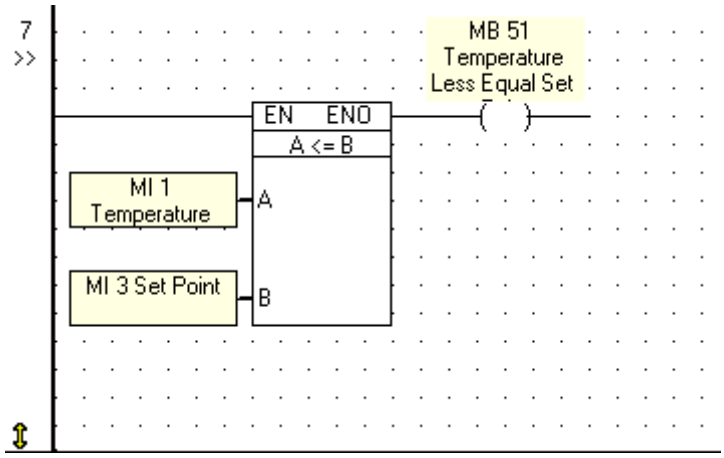
- Memory Integer **(MI)**
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**
- Network System Integer **(NSI)**
- Constant Value **#**

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 greater than or equal to MI3's value, then MB 51 will go to logic "0" (OFF).

These values may be compared:

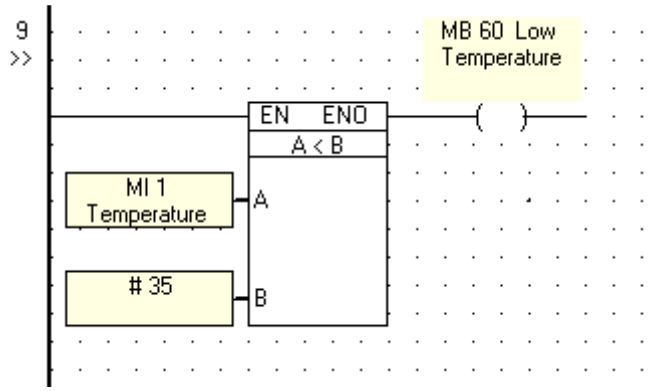
- Memory Integer **(MI)**
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**
- Network System Integer **(NSI)**
- Constant Value **#**

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 **#**

Within Range

The Within Range function 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.

Math Functions

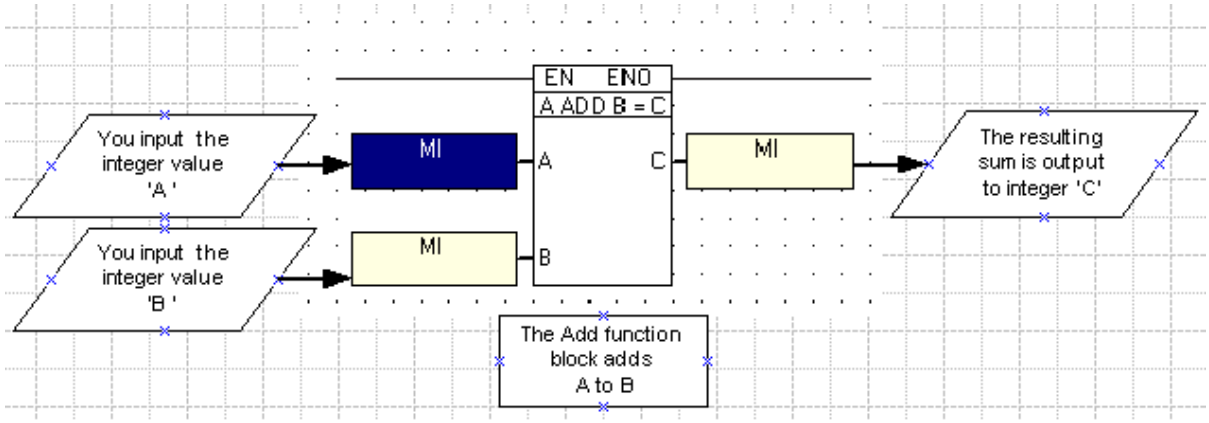
You perform mathematical functions by placing math functions in a net. Math functions, located on the Math menu are provided for:

- 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.



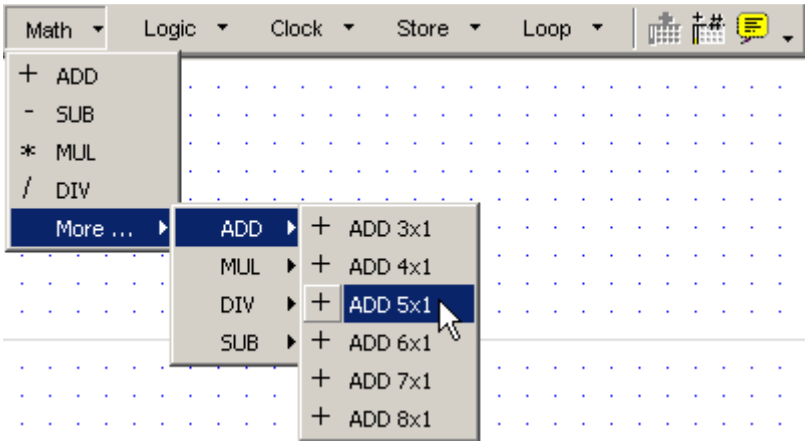
The operands listed below can be used to provide both input and output values, with exception of Constant Values. Constant values can provide input values, but can not contain output values.

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

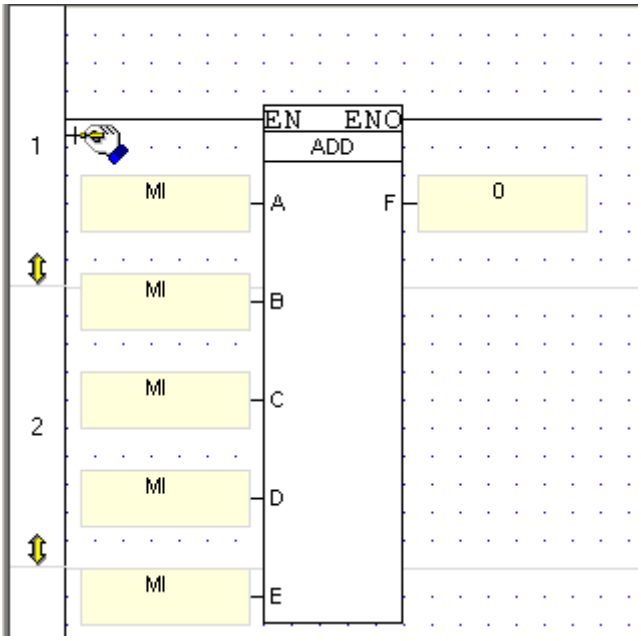
Multiple Input Values in Math Functions

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.

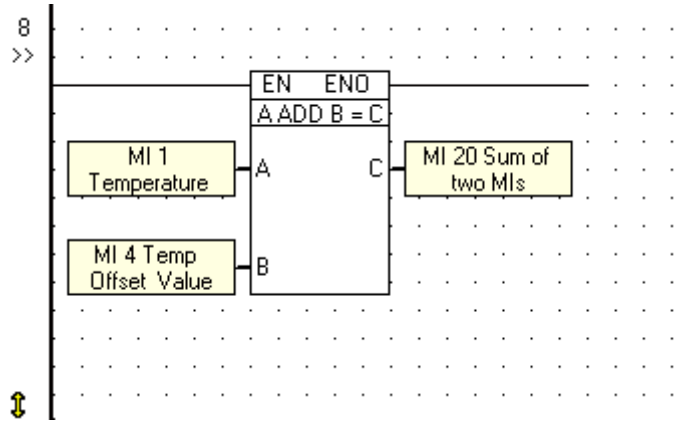
Add

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.

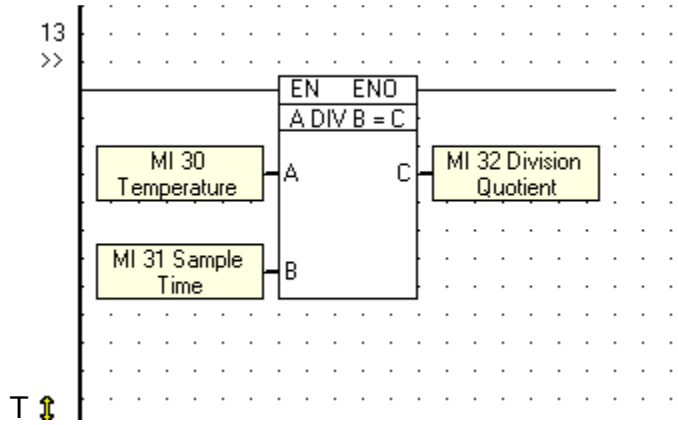


Divide

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.

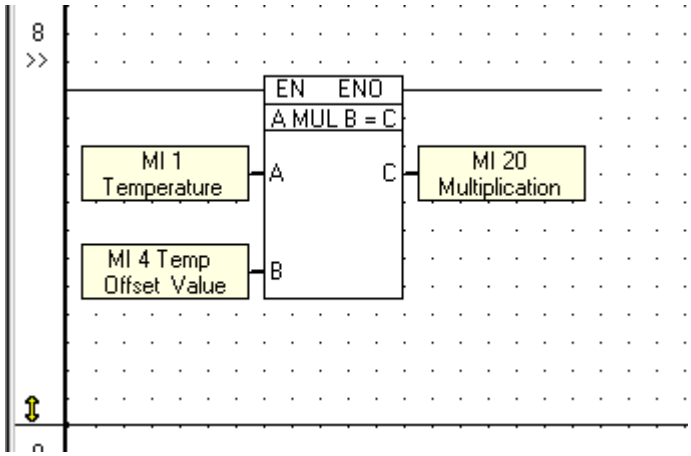
Multiply

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

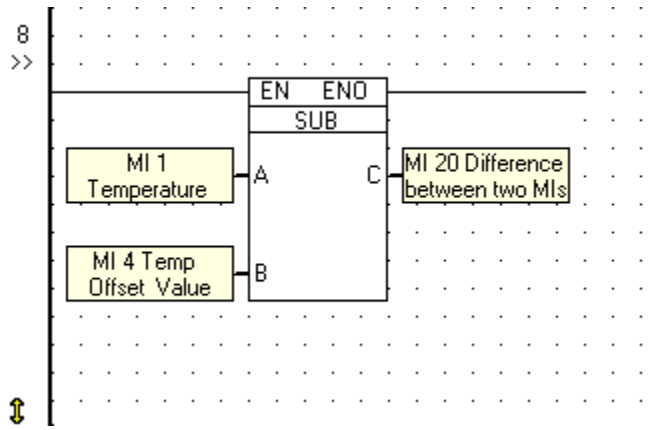
The math function Subtract is executed by the Subtract function block shown below. The function is located on the Math menu.

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.

The function performs A-B=C.



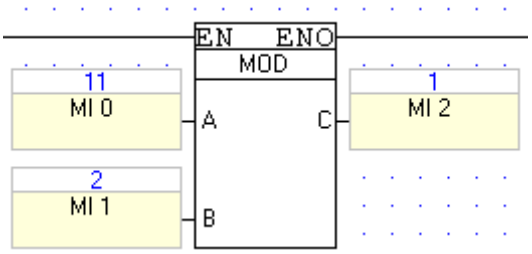
Modulo

The math function Modulo is executed by the Mod function block shown below. The input values in a Modulo 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 Modulo function divides the A input by the B input, and then store the remainder in the C output.

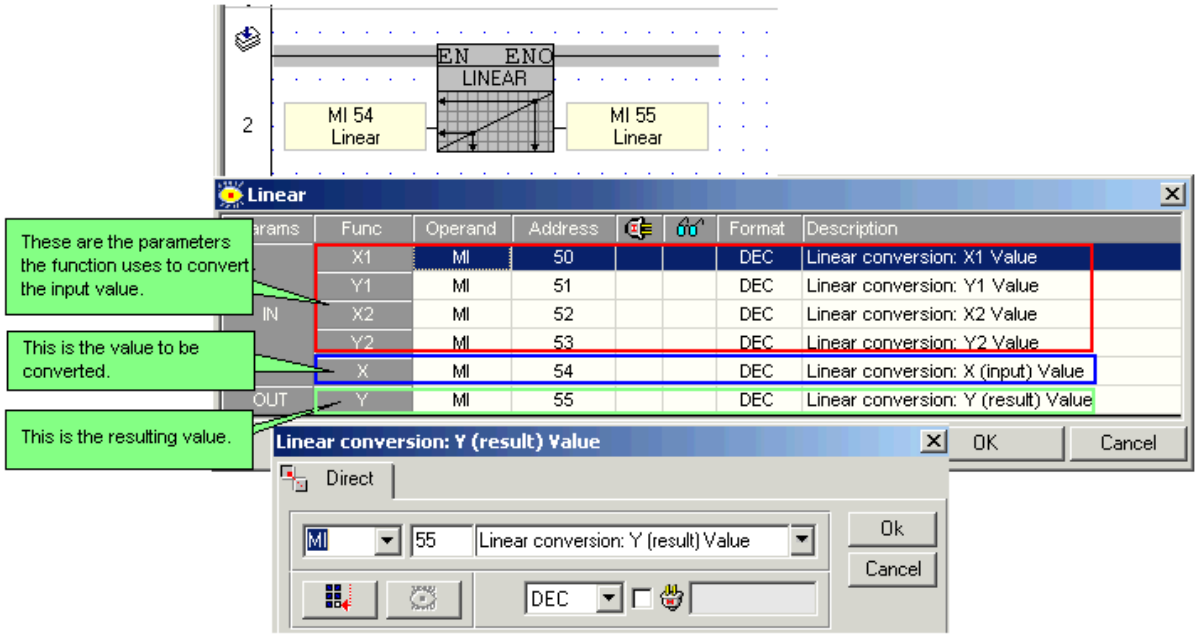
Values may not be divided by zero. In the event that this occurs, System Bit 4 (**SB 4** - Divide by Zero) turns ON.

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.

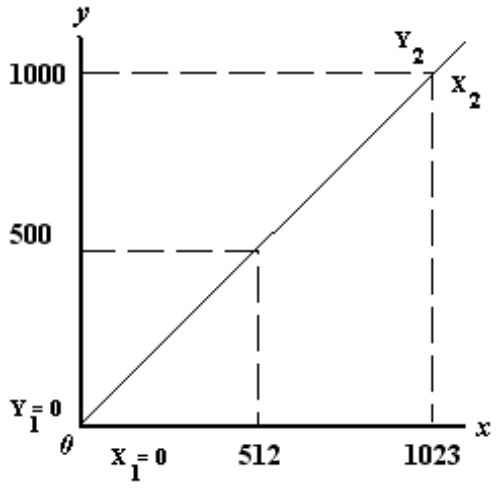
Linearize a Single Value

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



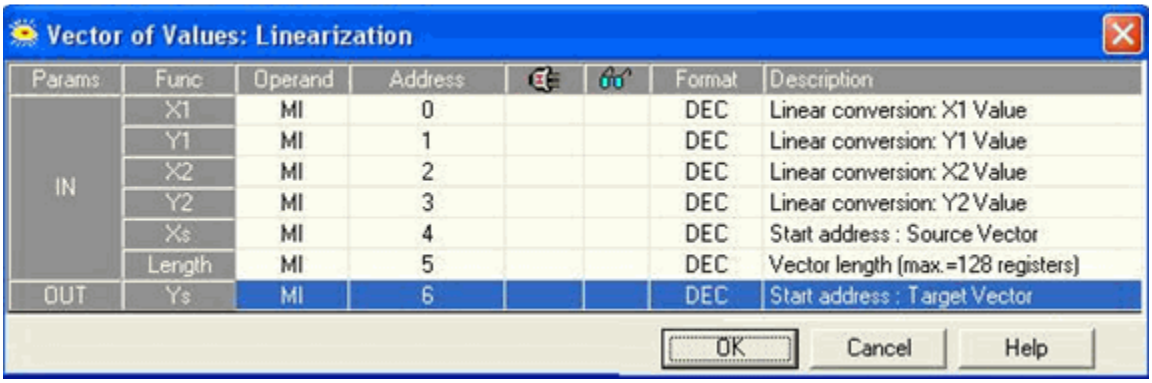
If, for example, X1 and Y1 are 0, and X2=1023 while Y2=1000, the output value will be linearized as graphed. These values would cause:

- An input of 5.0V to be converted to a digital value of 500.
- A input of 10.0Vto be converted to a digital value of 1000.



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.

Note • The X and Y values must not exceed the range of -2147483648 to +2147483647.

Known Issue

Note that the Linearization function cannot be used in cases where one of the following is close to the value FFFFFFFF:

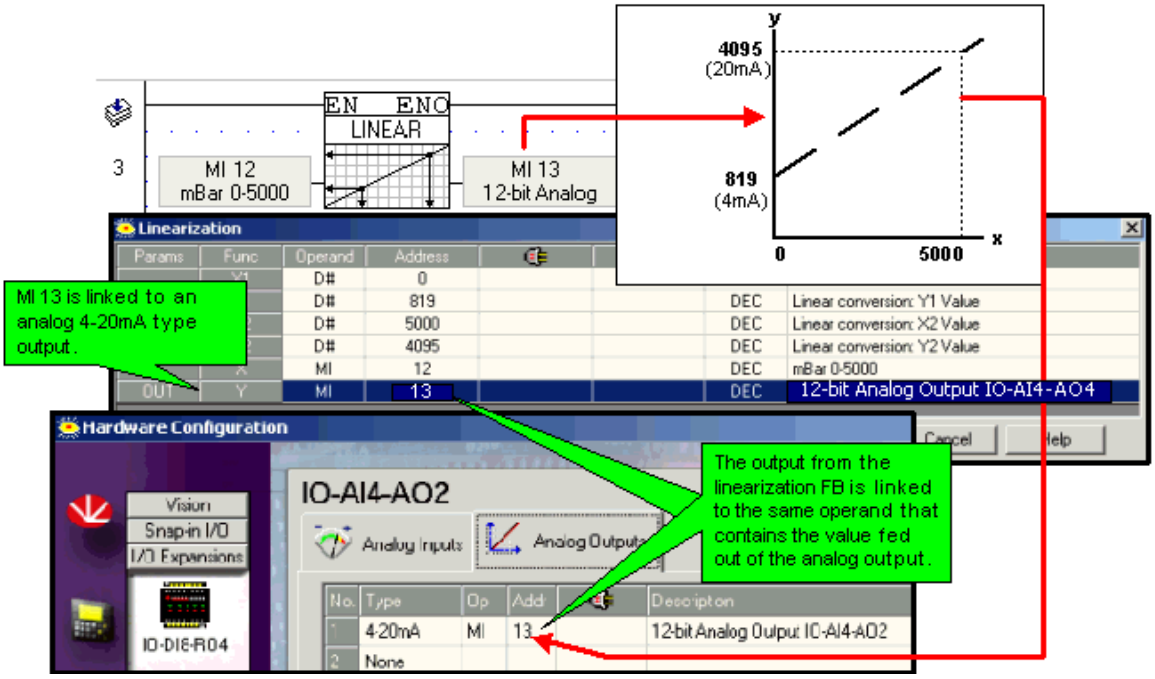
- One of the inputs
- An intermediate calculation

In these cases, use the Formula function with the following equation:

$$Y = \frac{(Y_{max} - Y_{min})}{(X_{max} - X_{min})} \times (X - X_{min}) + Y_{min}$$

Linearizing Analog I/O values

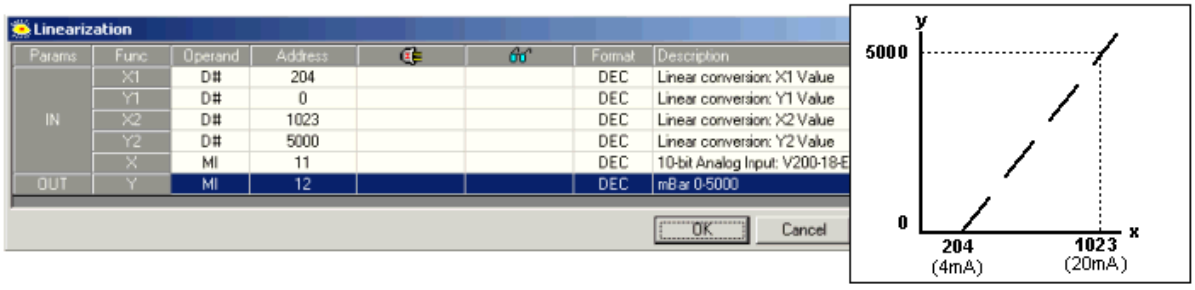
Note • Analog output values are contained in the register that you link to the output in Hardware Configuration.



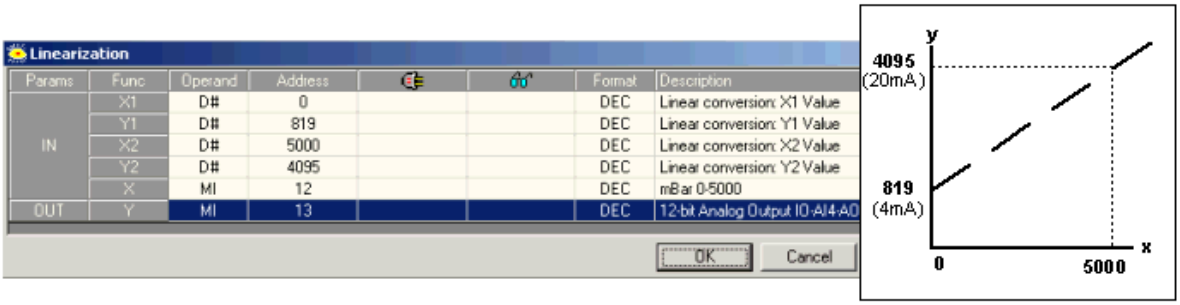
Working within the 4-20mA range

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.

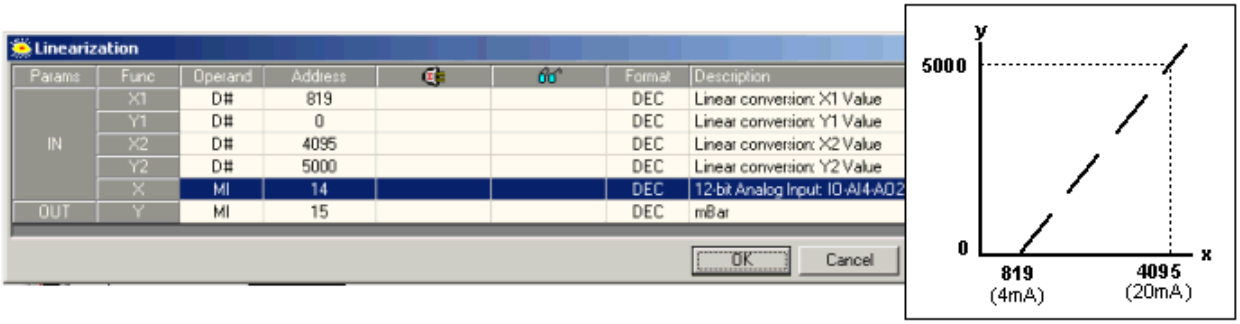
10-bit Analog Input, V200-18-E1

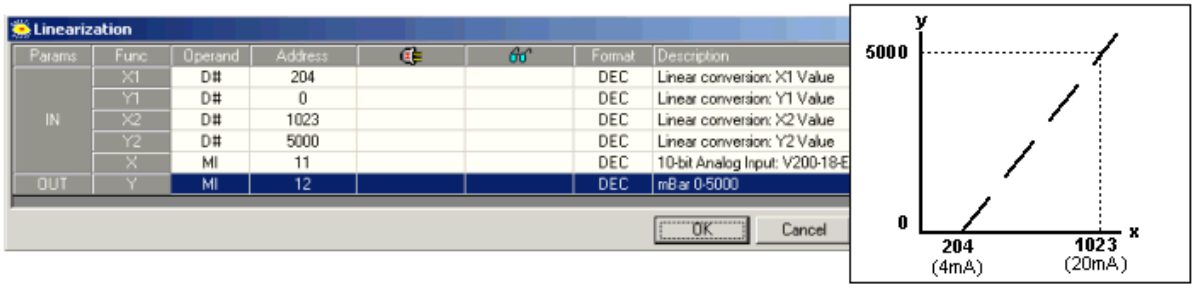


12-bit Analog Output,IO-A14-AO2

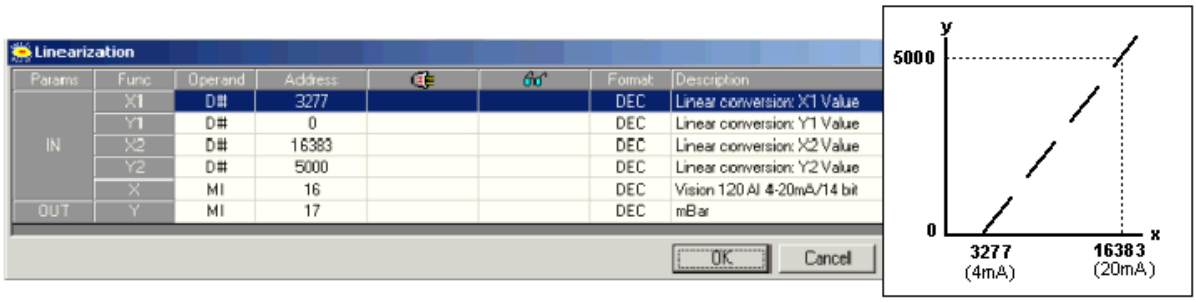


12-bit Analog Input, IO-A14-AO2





14-bit Analog Input, V120-12-UN2

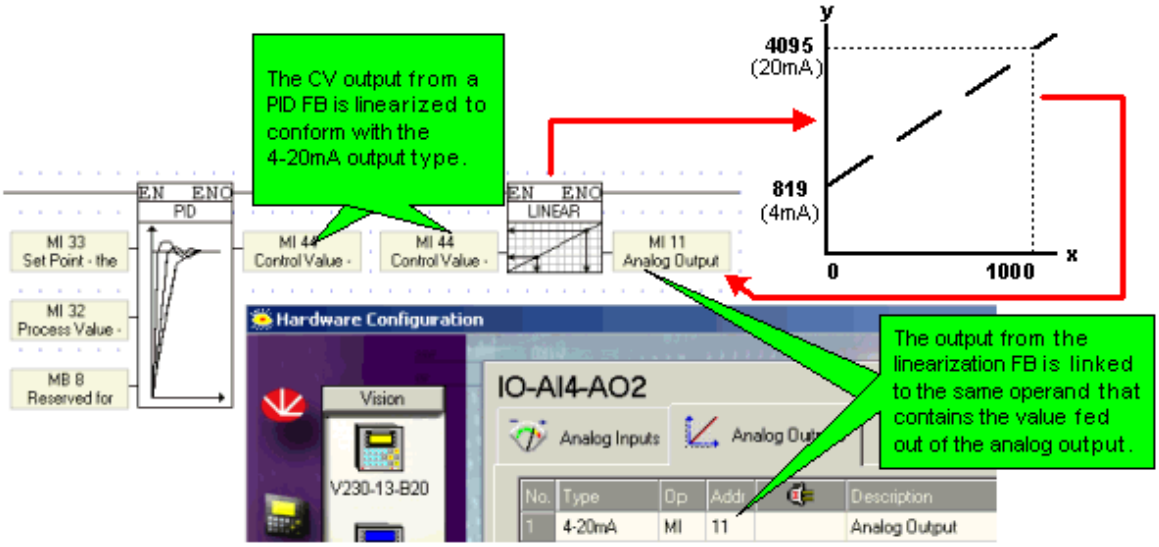


Linearizing a PID Analog Output Value

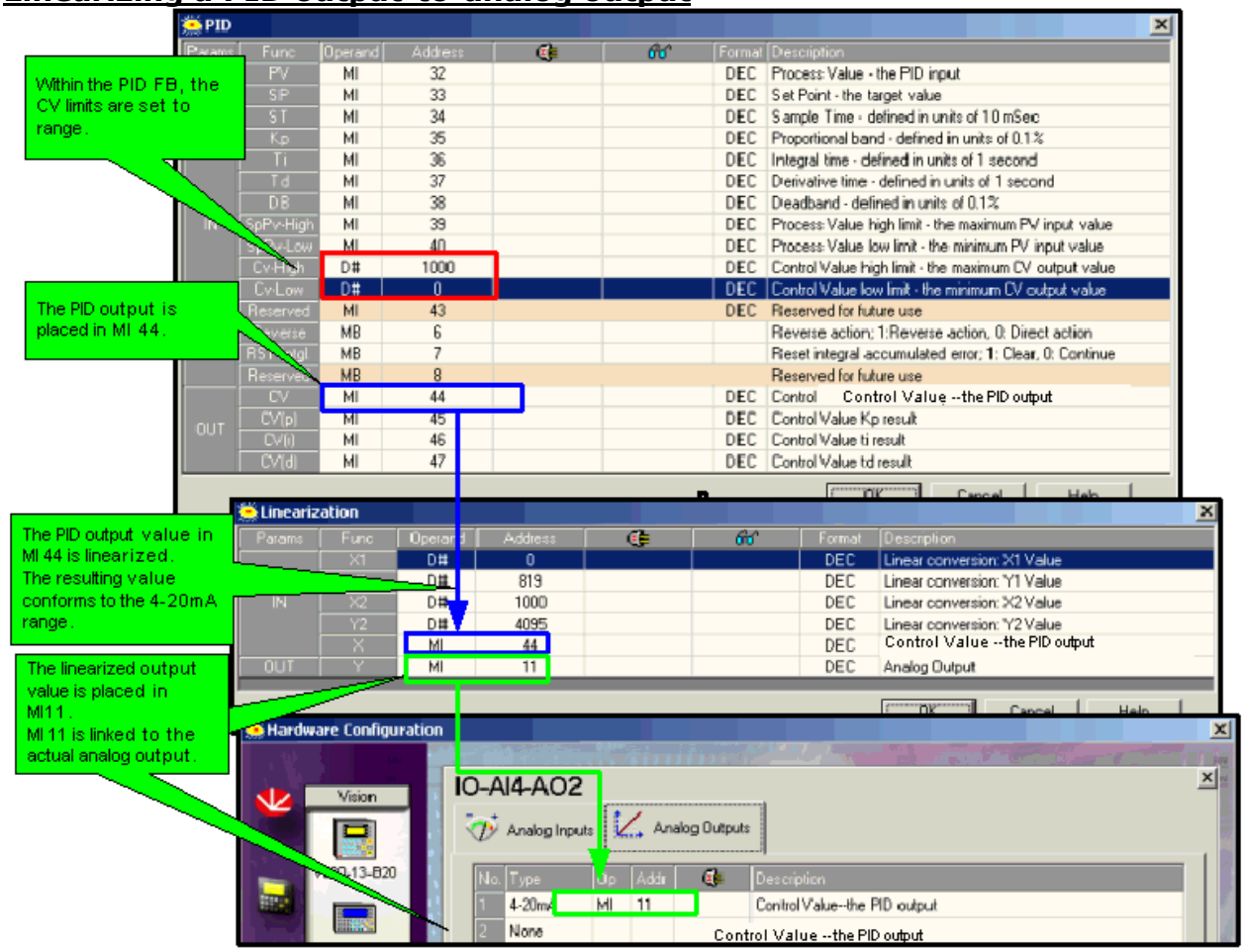
Analog values can be converted to physical values, for example Engineering Units (EU) such as degrees Celsius, by using the Linearization FB.

Note

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



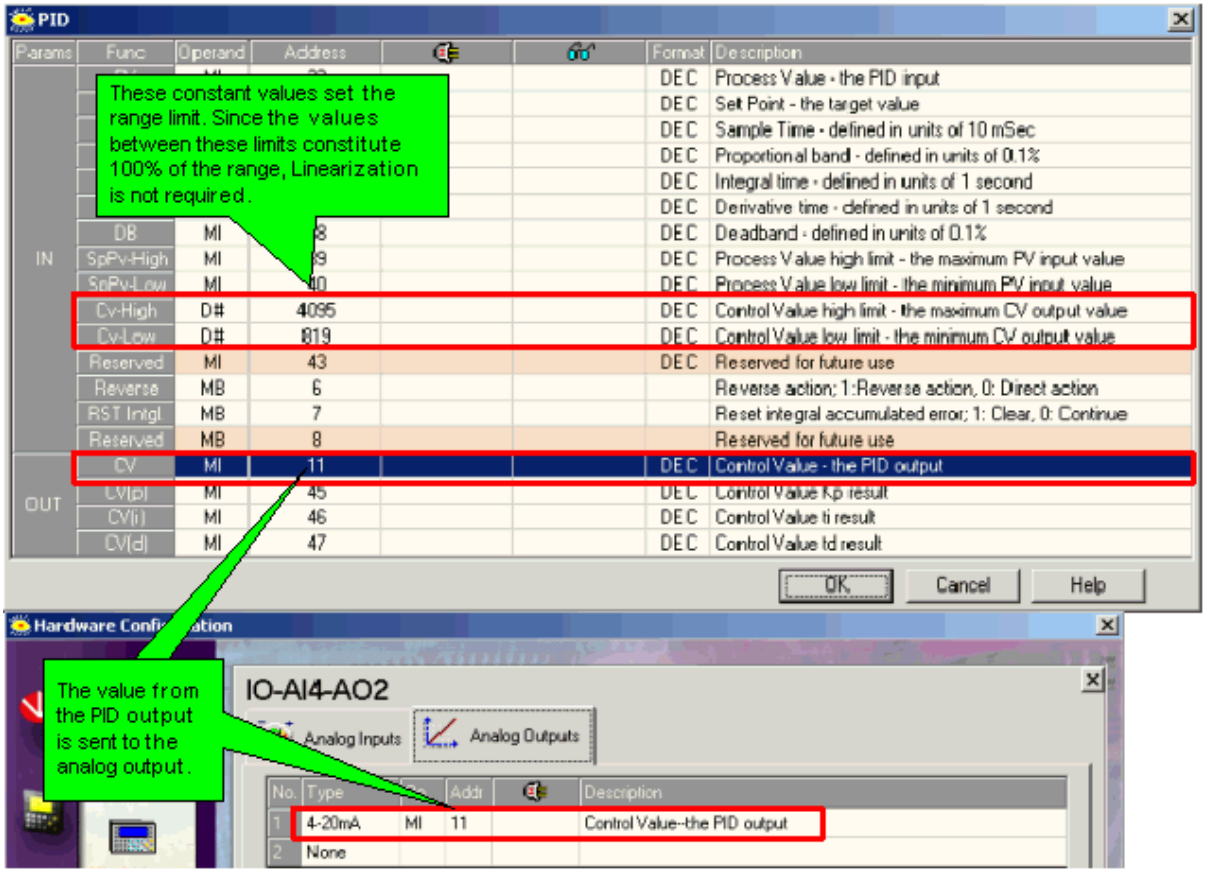
Linearizing a PID output-to-analog output



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

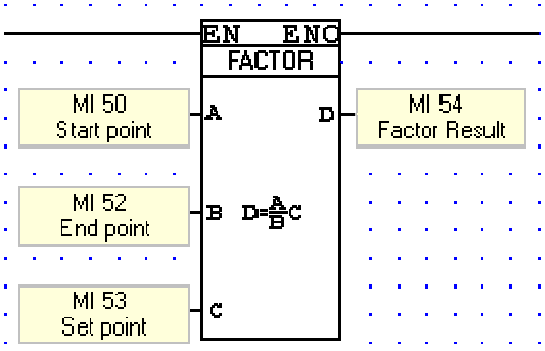
The math function Factor uses 3 input values. Factor divides an A input value by a B input value and then multiplies the result by a C input value. The result is stored in an output operand, D.

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.

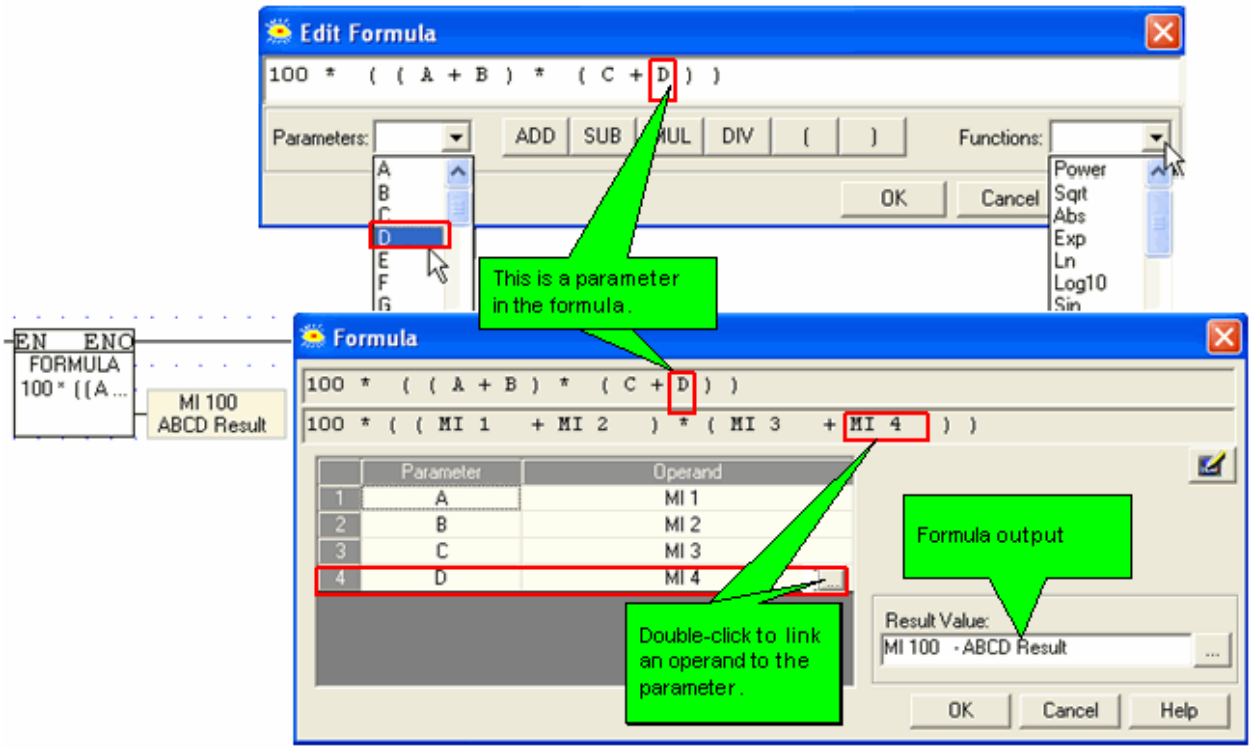


Formula: Build Your Own

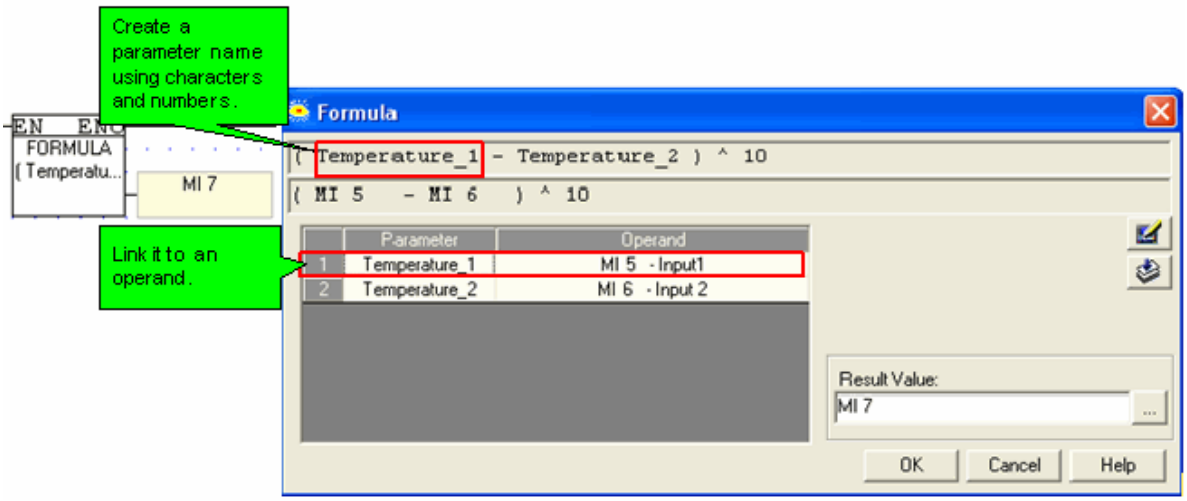
The Formula function, located on the Math menu, enables you to apply mathematical operators to operand values, and then output the result to a register.

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

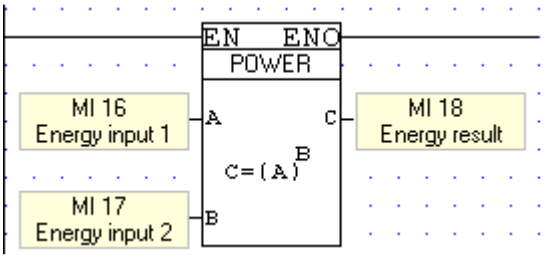
The math function Power uses 2 input values. Power raises an A input value by the power of a B (exponent) input value. The result is stored in an output operand, C. The function is located on the Math menu.

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.



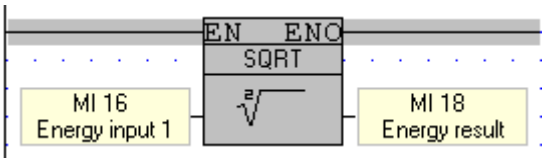
Square Root

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:

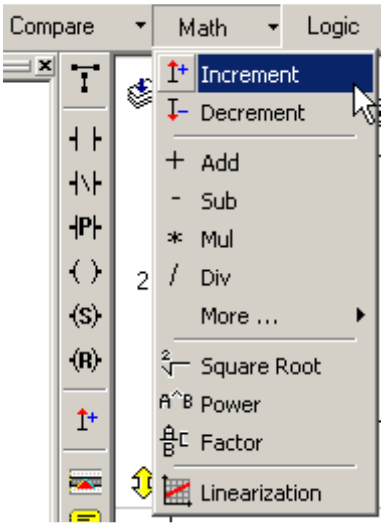
- Memory Integer **(MI)**
- Memory Long Integer **(ML)**
- Double Word **(DW)**
- System Operands:**(SI) (SL) (SDW)**

The example below shows a Square Root function.



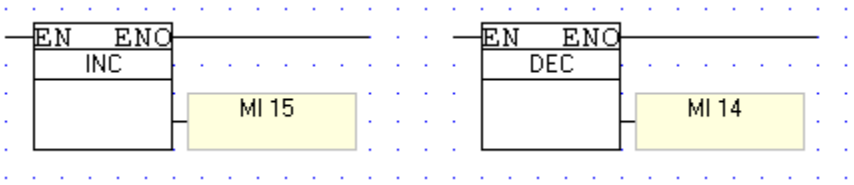
Increment/Decrement

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.

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 60 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(10^2). 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(10^2). If A is 3 and B is 9, the result will be 3,000,000,000.

Float: Trig Functions

These are the available 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.

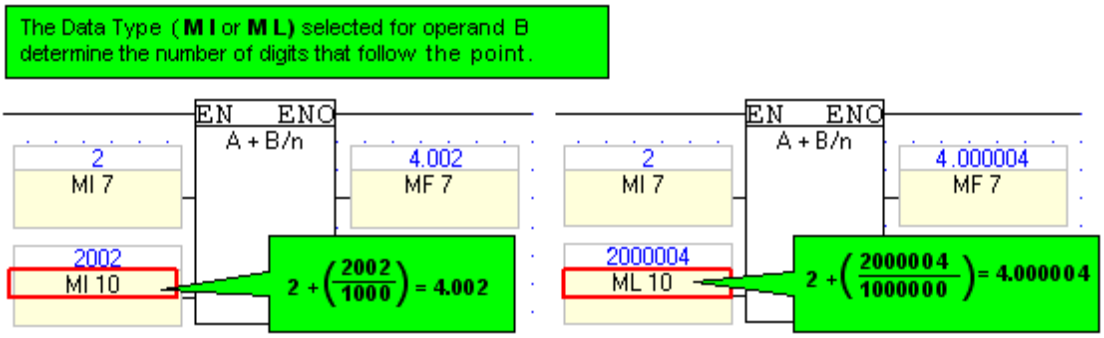
Float: Convert

These are the Convert Float functions:

- A+B/n

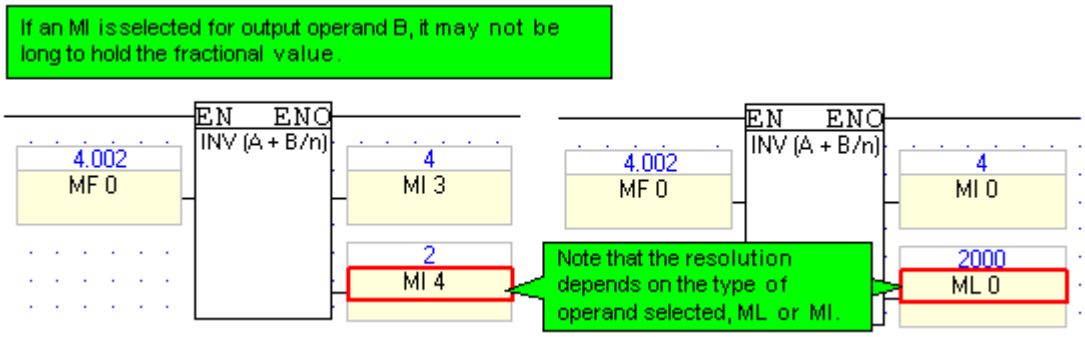
  This function takes 2 non-float values (whole numbers) and creates a single floating value.
  The two non-float values are added together; the A input, a whole number, is added to the B input, which is the fractional part of the number following the decimal point.

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.



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 440 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)

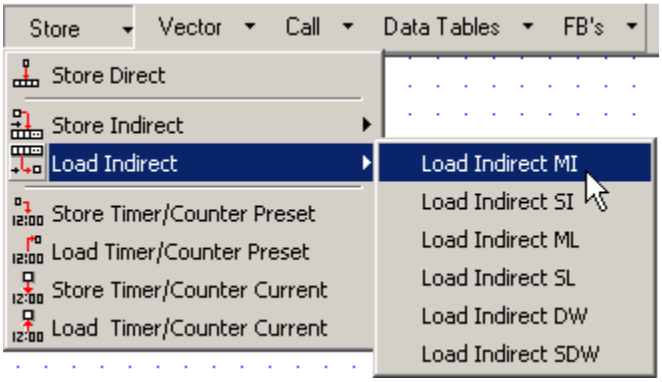
9	Undefined Float	NAN (float result)
10	Conversion Error	0 (integer result)
11	Floating point Stack Overflow	Floating point stack underflow
12	Floating point Stack Underflow	Floating point stack overflow

INF Infinite which is the largest absolute floating point number.

NAN Not a Number, special notation for undefined floating point number.

Store and 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.

- Reset Numeric
- 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
- Step in Range

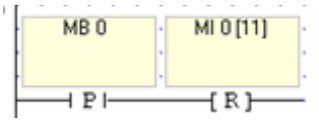
Reset Numeric

Reset Numeric allows you to initialize a register value to zero.

To use the function:

1. Click **Store** on the Ladder Toolbar.

2. Select **Reset Numeric**, then place the function in the desired net. In the following picture, when MB 0 turns ON, MI 0 = 0.



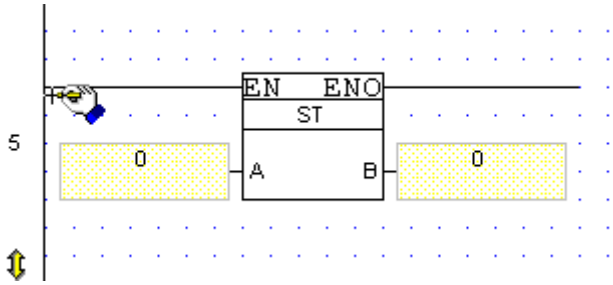
Store Direct Function

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

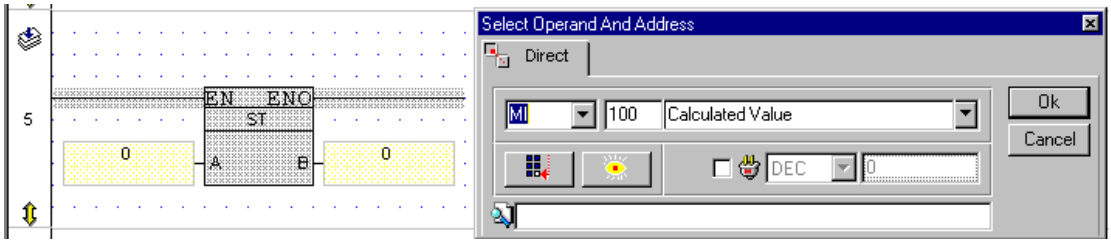
To use the Store Direct function:

1. Click **Store** on the Ladder Toolbar.

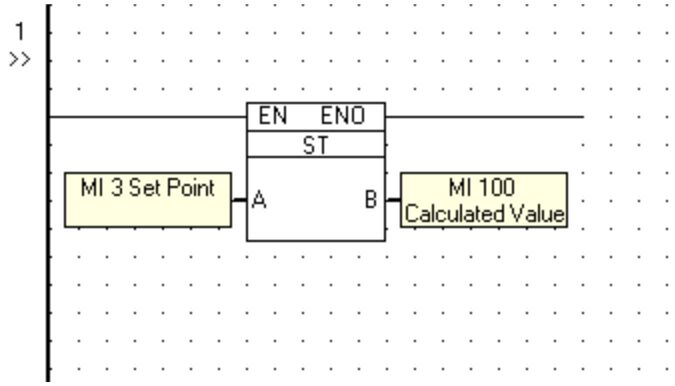
2. Select **Store Direct**, then place the Store Direct function in the desired net.



3. Enter the desired Operands and Addresses.



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

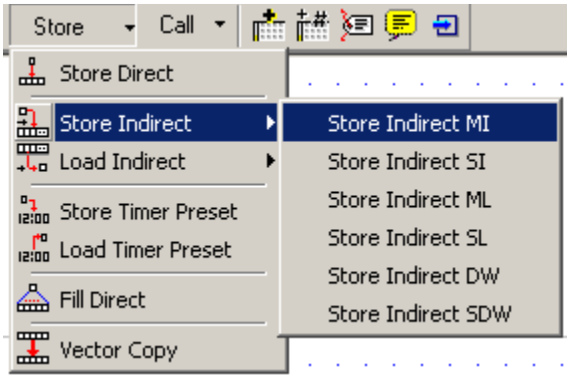


According to the above example, the value in MI 3 will be stored in MI 100. The previous value in MI 100 is **overwritten**. The current value in MI 3 remains **unchanged**.

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.

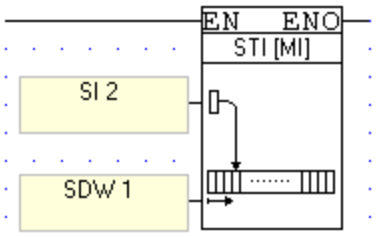
When you select the function type from the Store menu, the program writes the input A value into the address referenced by the output B value--according to the type of function you select.



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.

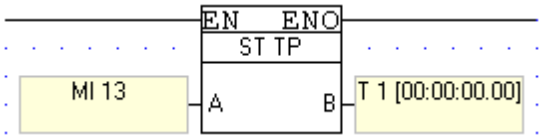
The value 5 will therefore be stored in MI 10.



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.



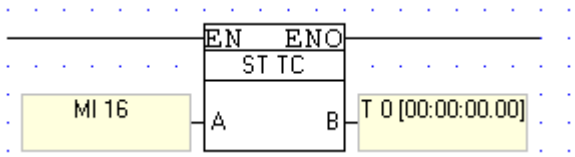
Note

- 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 10120, the value stored into T1 will be 00:01:41.20.

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

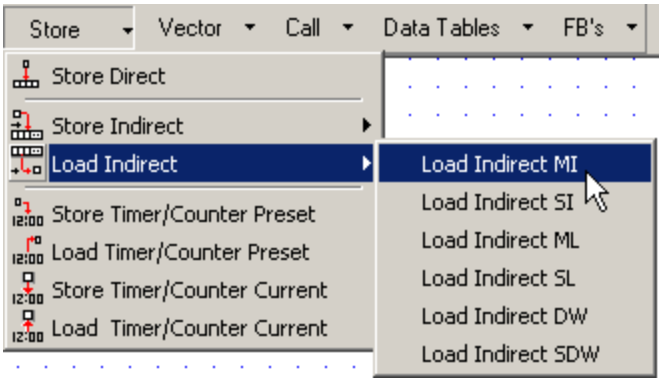
- The value that is stored in the Timer is broken down into units of 10 milliseconds. In the above example, if MI 16 is equal to 10120, the value stored into T1 will be 00:01:41.20.

Load Indirect Functions

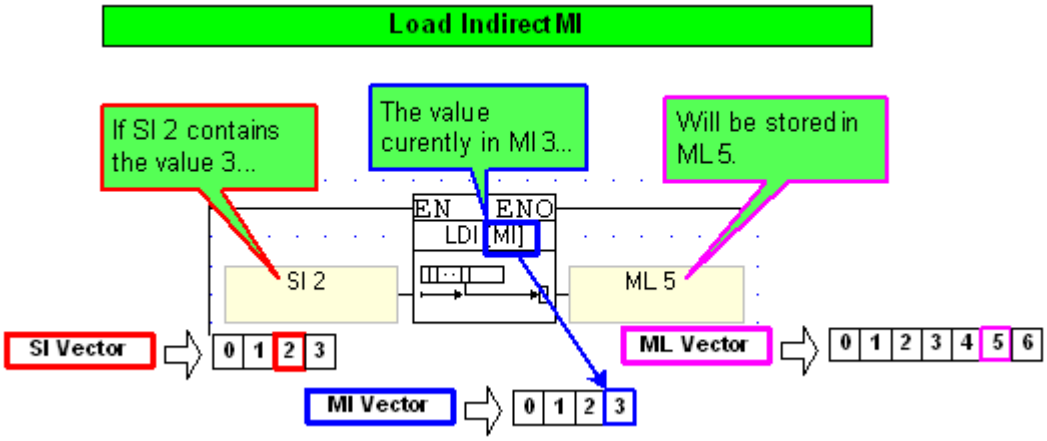
Load Indirect allows you to take a value contained in a source operand and load it into a destination operand using indirect addressing.

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.

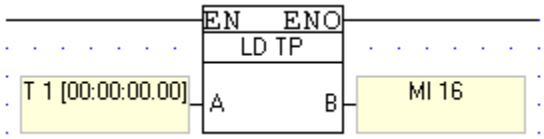


According to the above example, if the value in MI 3 is 986, 986 will be stored in ML 5. The previous value in ML 5 is **overwritten**. The current value in MI 3 remains **unchanged**.

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.

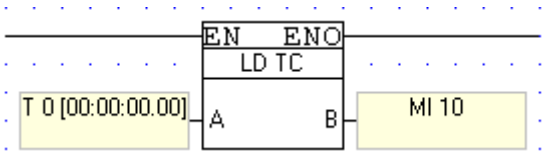


Note • Timer value units are 10 milliseconds. In the above example, if TI is equal to 1 minute, 41 seconds, and 20 deciseconds (00:01:41.20.), the value 10120 will be stored into MI 16.

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.

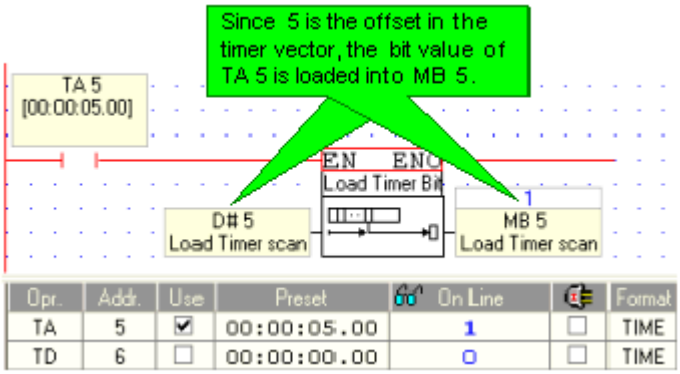


Note •

Timer value units are 10 milliseconds. In the above example, if T0 is equal to 1 minute, 41 seconds, and 20 deciseconds (00:01:41.20.), the value 10120 will be stored into MI 10.

Load Timer Bit Value

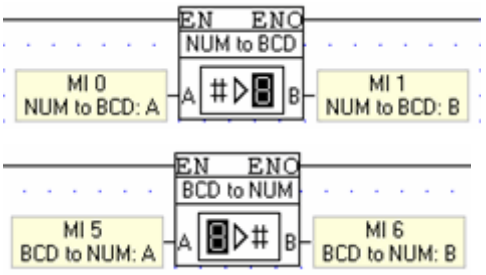
You can use a Ladder condition to load the current bit value of a Timer into an MB. The input to the Load Timer Scan Bit function is the address of the timer within the Timer vector, and may be a constant or a value provided by a register.



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 •

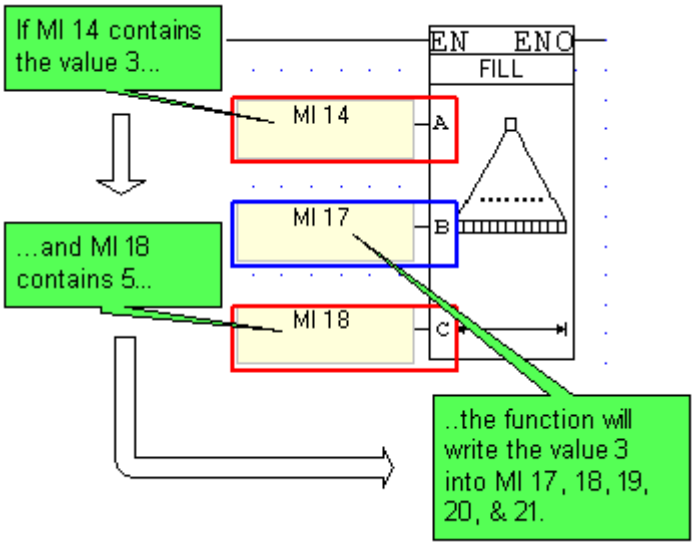
This type of BCD may be used in seven-segment displays, composed of seven elements.

- The function does not support negative values.
- Use ML or DW for large values.

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.



Step in Range

Step in Range enables you to increment a value by a desired amount, to keep the incremented value within a desired range, and notifies you when the incremented value reaches the limit.

To use the function:

1. Click Compare on the Ladder Toolbar, then select Step in Range.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.
   Operand A: Value to Increment/Decrement. This is the value that is incremented.
   Operand B: Result - Minimal Value. This is the lower value of the range.
   Operand C: Result - Maximal Value. This is the higher value of the range
   Operand D: Step. This is the value used for the step size
   Operand E: Roll. Turn ON to cause the function to continue to work once the function has been reached.
   Operand F: Count Up. Turn ON to increment
   Operand G: Count Down. Turn ON to decrement
   Operand H: Output, Limit Reached Notifier. This turns ON for one scan when:
   - Count UP is active, and the incremented value equals the Maximal value or
   - Count Down is active, and the decremented value equals the Minimal

value
The MB resets automatically when the value is not equal.

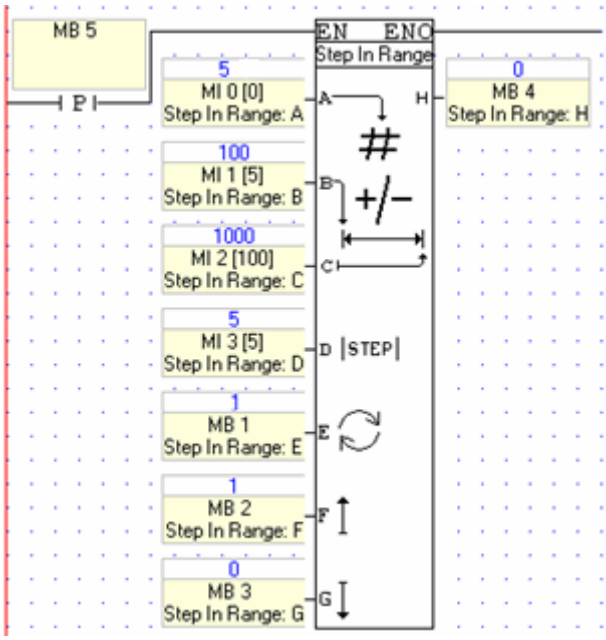
Notes •	The step size is limited to the range of 1-1000.
•	Value to Increment/Decrement, Minimal Value, and Maximal Value must be the **same** type of operand, signed or unsigned.
•	If the initial value of the MI linked to Value to Increment fall outside of the range, the first time it is activated, the value will change to either: - match the Minimal Value — assuming the function is set to Count Up, or -match the Maximal Value, assuming the function is set to Count Down.

Example:

In the following picture:

- MI 0 is the value to be incremented
- The range is 100 to 1000
- The Step Size is 5
- MI 0 is set to Count Up

The first time MB 5 rises, the value in MI 0 will jump to 100, which is the Minimal value. After this, each time the MB 5 rises, MI 0 will increment by 5, until it reaches 1000, turning MB 4 ON. Since Roll is ON, the function will begin to count again, this time from 99, MB 4 will then reset.



Vector Operations

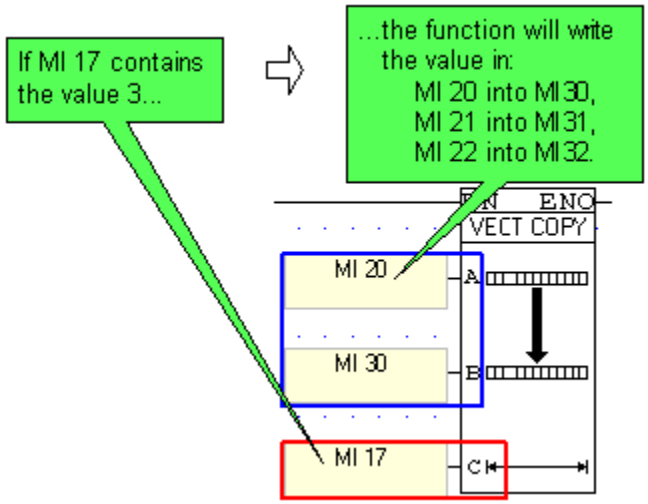
Vector operations enable you to select an operand type, define a vector within that type, and to perform different actions within the defined vector.

- 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 Copy

Vector Copy enables you to set a range of operands, copy the values of each operand within that range, then write those values into a corresponding range of operands of the same length. The function is located on the Vector menu.

- 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.



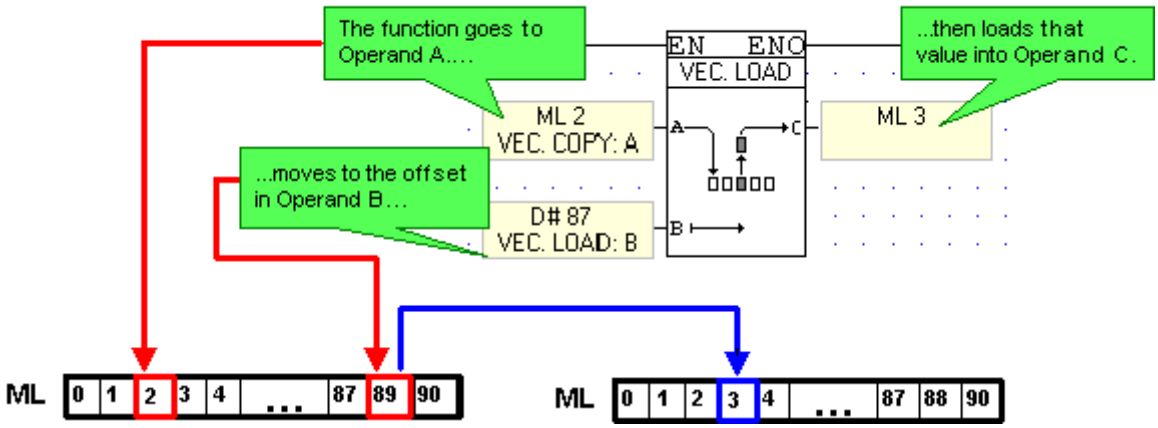
Vector: Load

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.

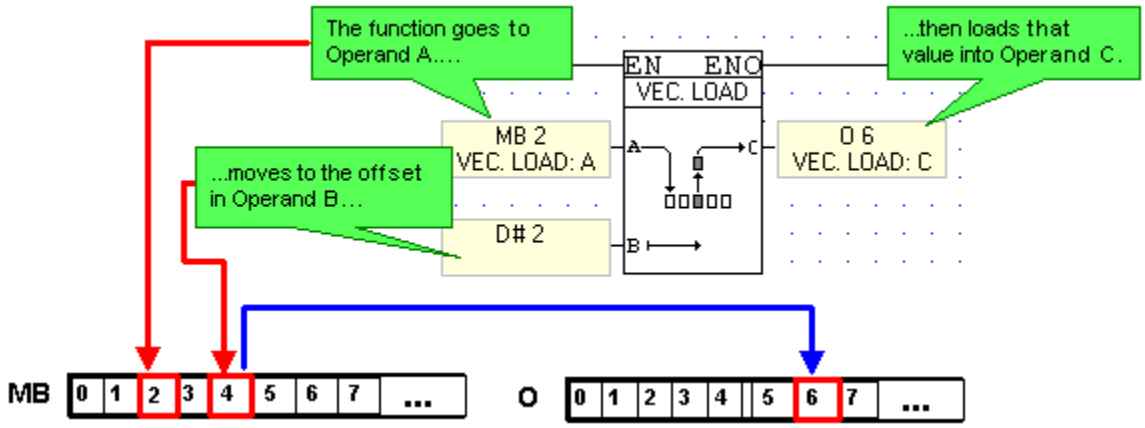
Example: Registers

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.

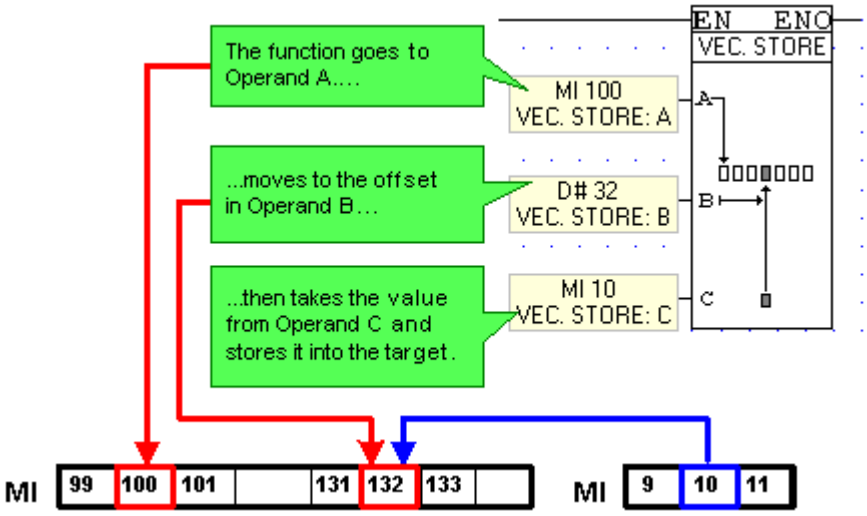
Vector: Store

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.

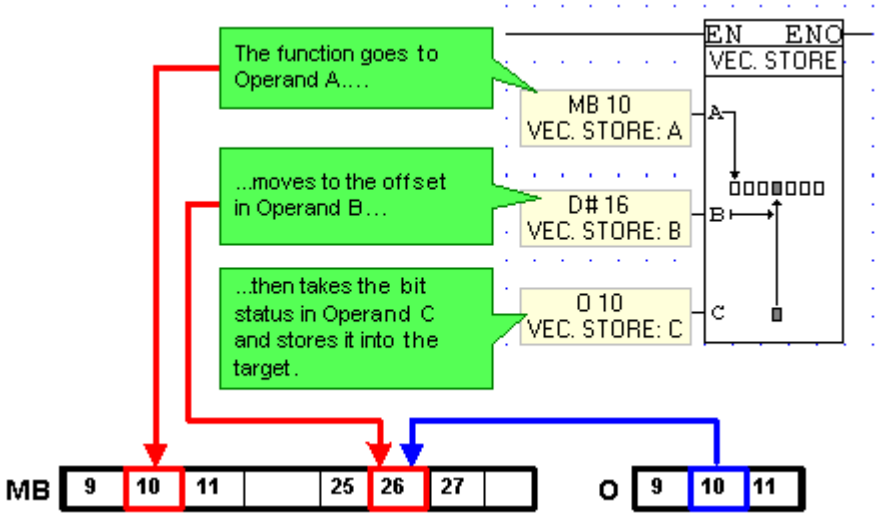
Example: Registers

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:

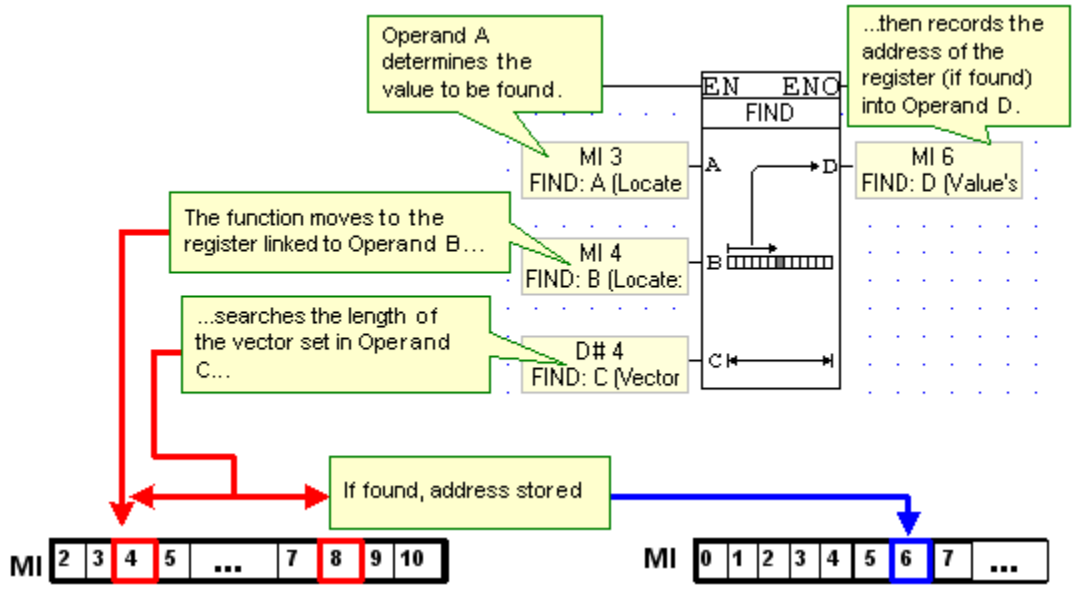
- searches through a vector,
- locates either an integer value or the first bit of a desired status within that vector,
- records the location of the operand containing the desired value.

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

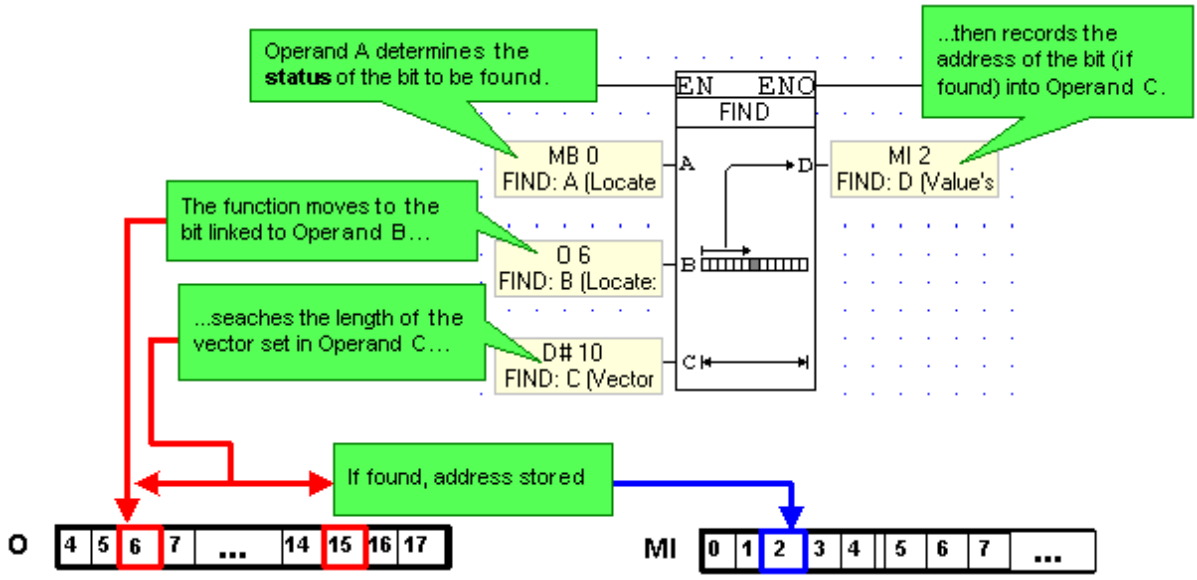
Below, if MI 3 contains the value 16, the function searches for 16 from MI 4 to MI 8. MI 3. If the value 16 is found in the vector, the address of the operand

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

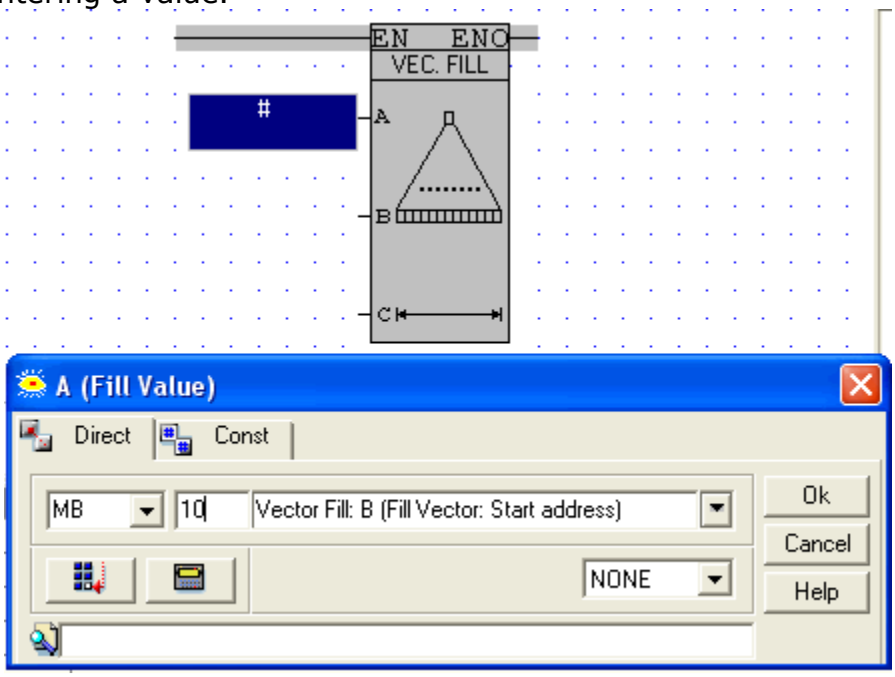
Fill enables you to:

- 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.

Note • The first operand you select, Fill Value, determines the type of operands you can fill. For example, if you wish to initialize a vector of MBs, you must select the Direct tab, and then select an MB or an Output as the Fill Value. Selecting a register will enable you to write to registers, as will selecting the Const tab and entering a value.

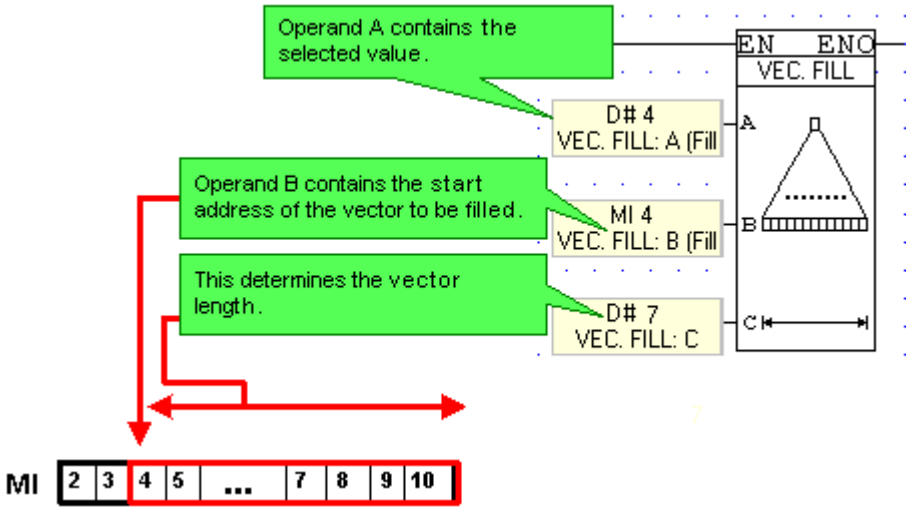


Vector Fill

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)

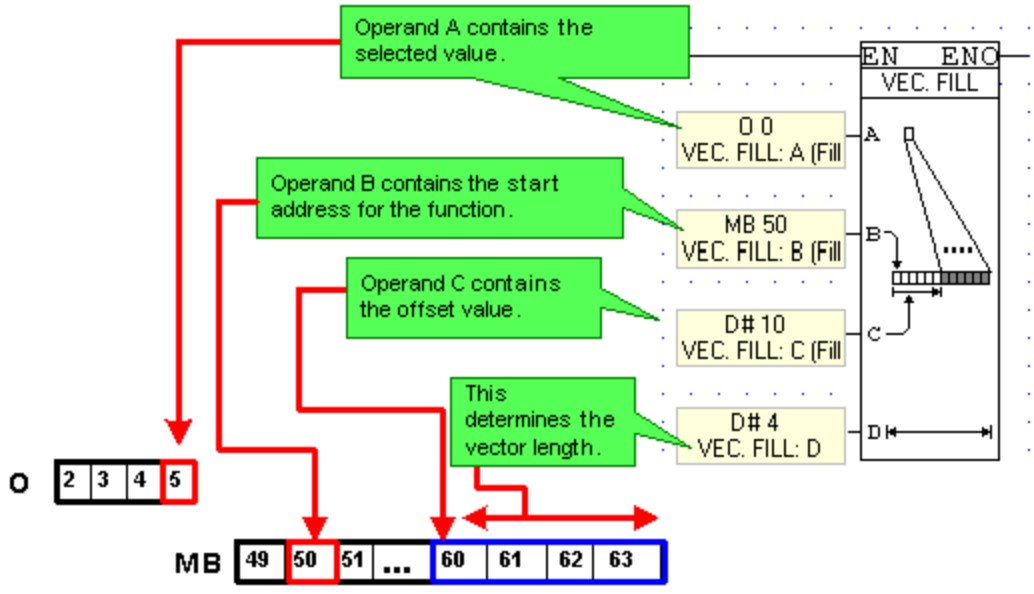
Fill (Offset) enables you to:

- 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.

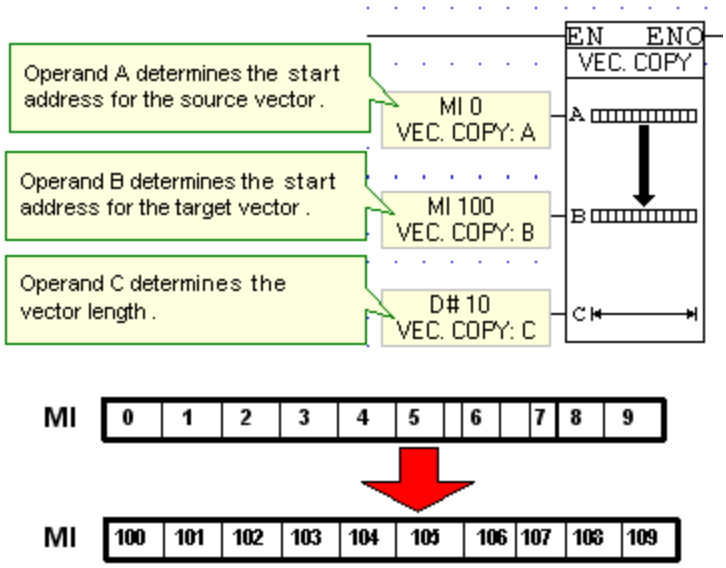
The function is located on the Vector menu.

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 9 will be copied to MI 100 to 109.



Copy (Offset)

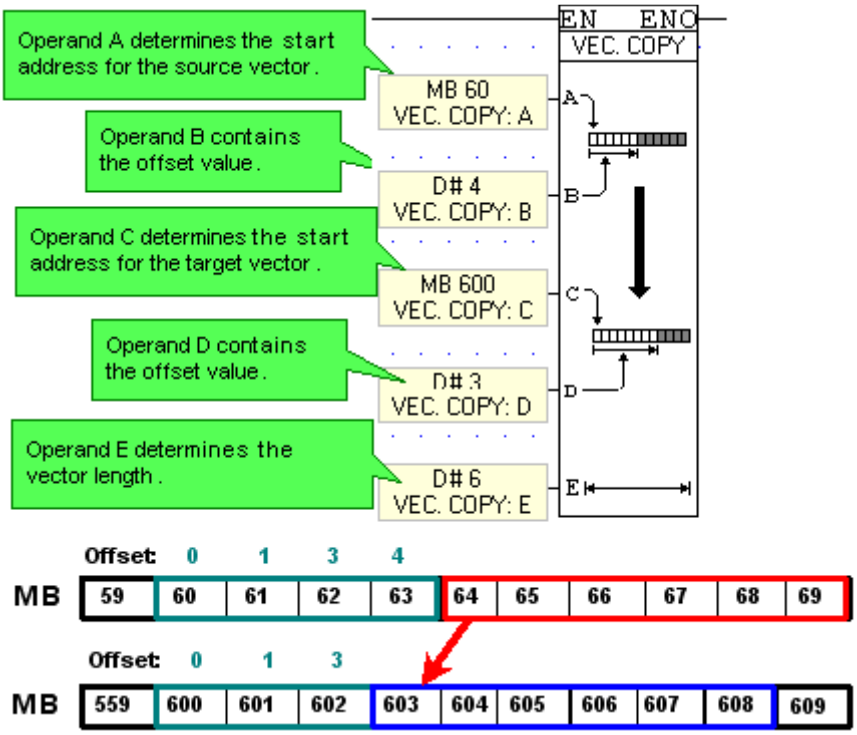
Copy (Offset) enables you to:

- 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.



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.

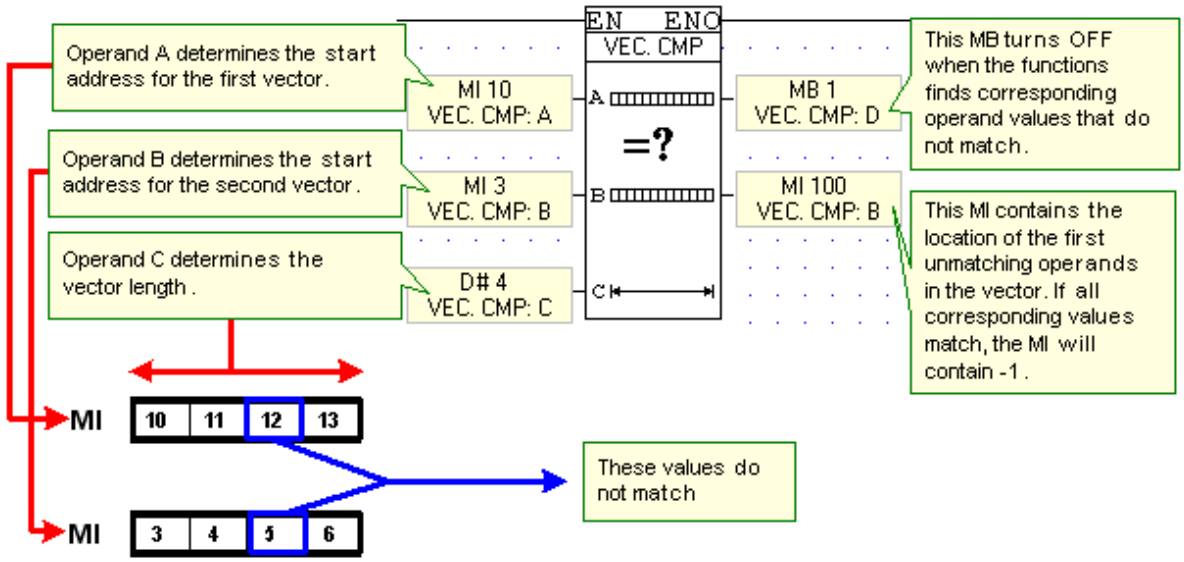
Compare

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:

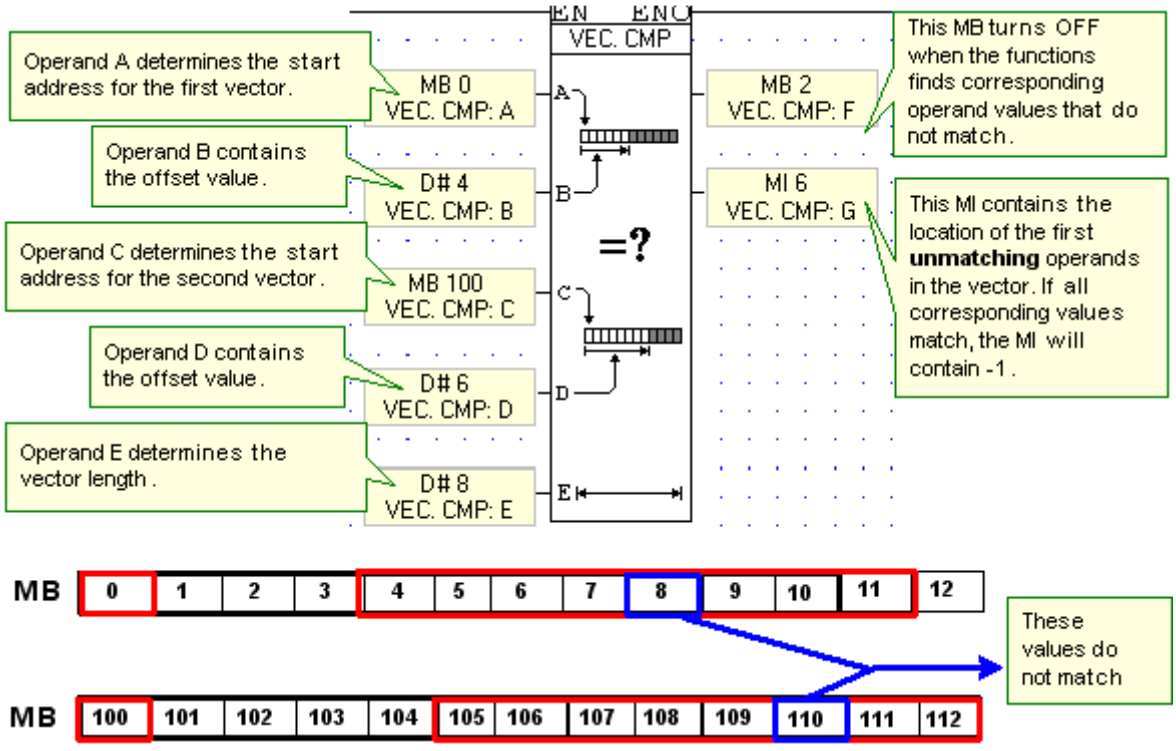
- Define a source vector of operands that is offset from a selected start address,
- define a target vector of operands that is offset from a selected start address,
- compare the values or bit status of each corresponding operand within that range,
- record the location of the first set of unmatched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

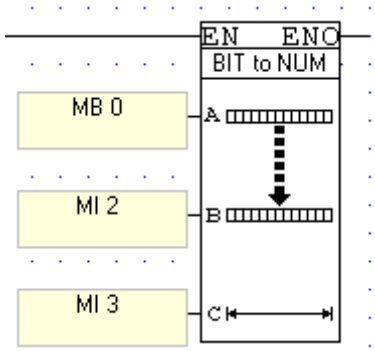
Use these functions to convert an array of bit values to a numeric value, or a numeric value to an array of bits.

The functions are located on the Vector menu.

Bit to Numeric

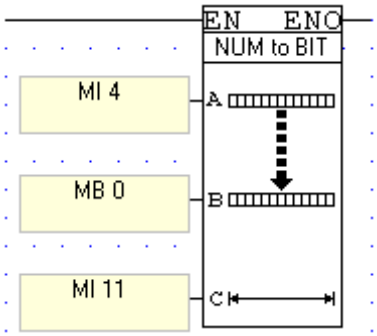
- Operand A: contains the Start Address for the array of bits to be converted.
- Operand B: is the start of the vector that will contain the converted value. Take care in addressing operands, since the converted value may not fit into a single register; the function will overwrite as many consecutive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

- Operand C: contains the length of the bit array that will be converted.



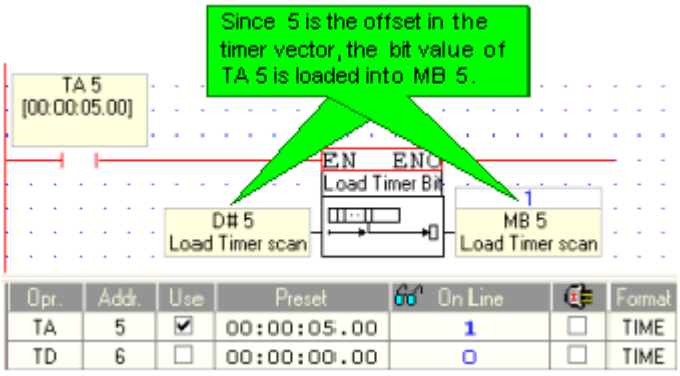
Numeric to Bit

- 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.

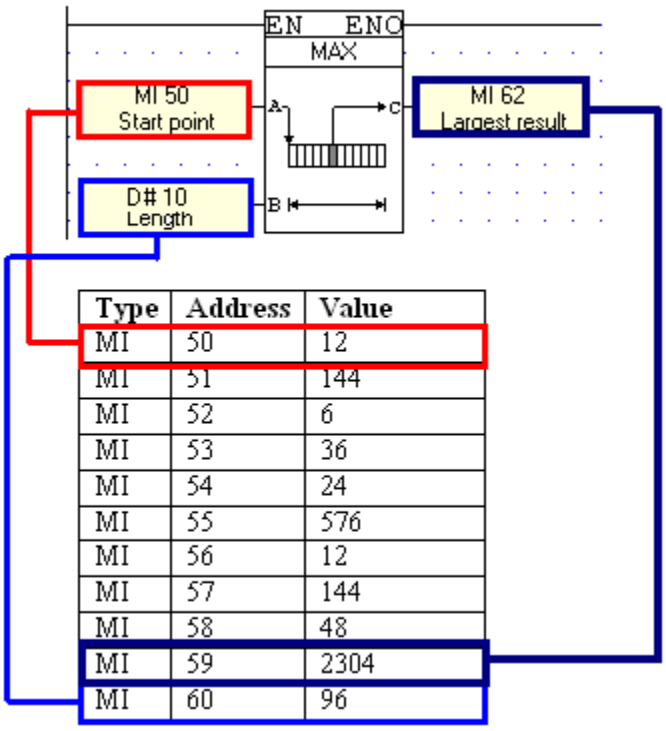


Vector: Get Max

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.



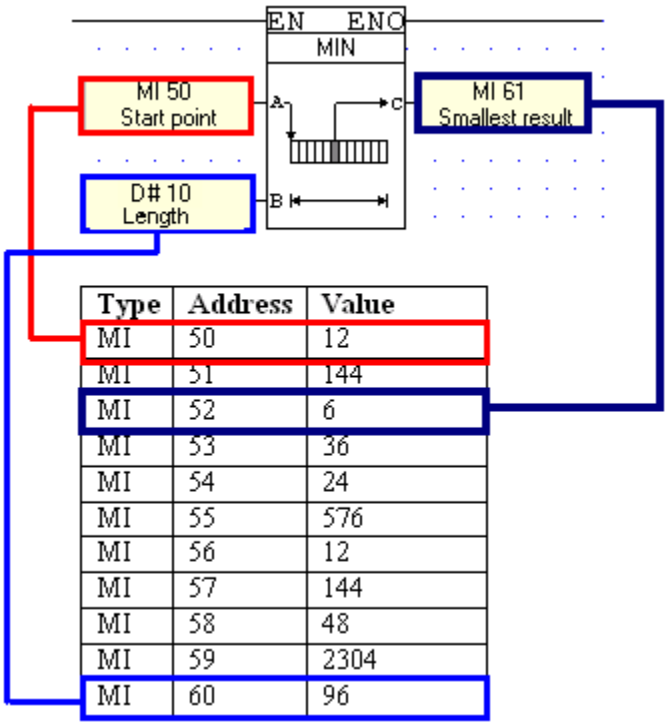
Type	Address	Value
MI	50	12
MI	51	144
MI	52	6
MI	53	36
MI	54	24
MI	55	576
MI	56	12
MI	57	144
MI	58	48
MI	59	2304
MI	60	96

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.



Type	Address	Value
MI	50	12
MI	51	144
MI	52	6
MI	53	36
MI	54	24
MI	55	576
MI	56	12
MI	57	144
MI	58	48
MI	59	2304
MI	60	96

Vector: Copy Memory

Copy Memory enables you to copy a vector of bytes from a vector of registers.

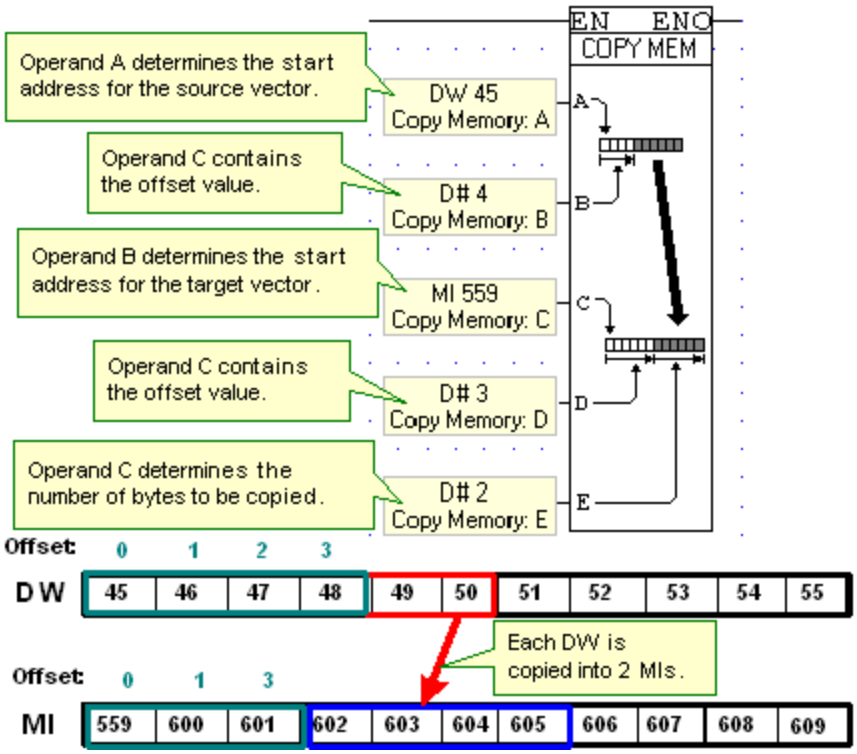
To use this function

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: Start address for the source vector of registers.
   Operand B: Offset from the start address.
   Operand C: Start address for the target vector.
   Operand D: Offset from the start address.
   Operand E: The number of bytes to be copied from the source into the target vector.

Example:

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

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 Left

Shift enables you to:

- define a vector of operands,
- shift the bits or bytes within that vector left

To use this function

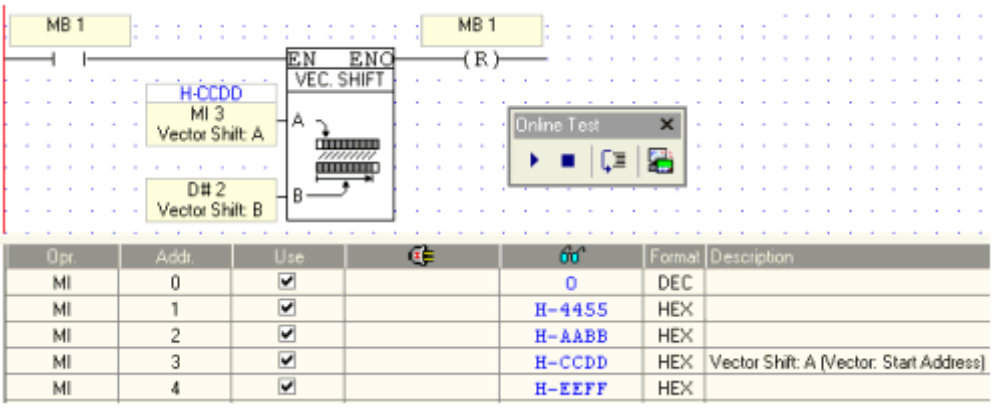
1. Click the Vector menu on the Ladder Toolbar, then select Shift.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.

   Operand A: this is the start address for the source vector.
   If you select MB or XB, the function will shift bits in the vector, if you select a register type, the function shifts bytes.

   Operand B: this is the number of bytes to shift.

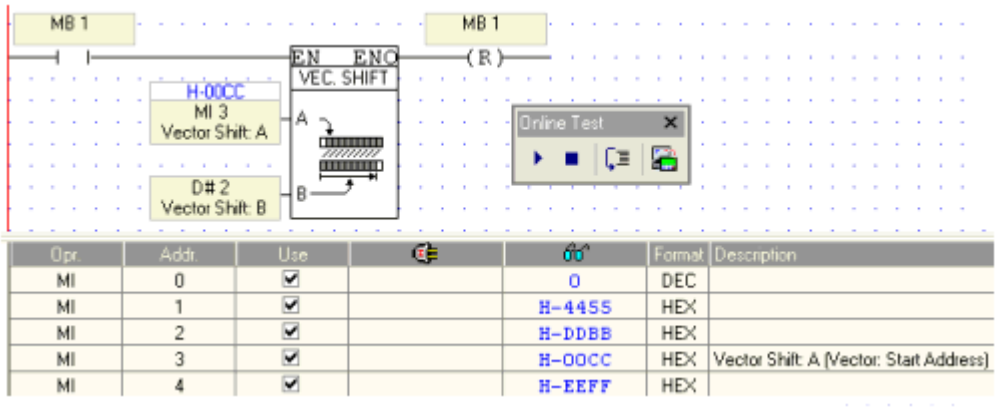
Example:

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



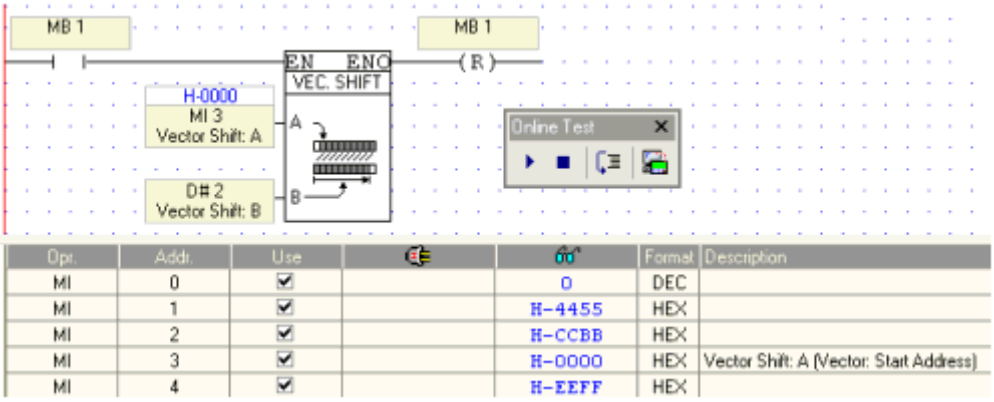
Opr.	Addr.	Use			Format	Description
MI	0	☑		0	DEC	
MI	1	☑		H-4455	HEX	
MI	2	☑		H-AABB	HEX	
MI	3	☑		H-CCDD	HEX	Vector Shift: A (Vector: Start Address)
MI	4	☑		H-EEFF	HEX	

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



Opr.	Addr.	Use			Format	Description
MI	0	☑		0	DEC	
MI	1	☑		H-4455	HEX	
MI	2	☑		H-DDBB	HEX	
MI	3	☑		H-00CC	HEX	Vector Shift: A (Vector: Start Address)
MI	4	☑		H-EEFF	HEX	

Repeating Shift leaves MI 3 empty.



Opr.	Addr.	Use			Format	Description
MI	0	☑		0	DEC	
MI	1	☑		H-4455	HEX	
MI	2	☑		H-CCBB	HEX	
MI	3	☑		H-0000	HEX	Vector Shift: A (Vector: Start Address)
MI	4	☑		H-EEFF	HEX	

Notes • This function cannot be performed on negative values.

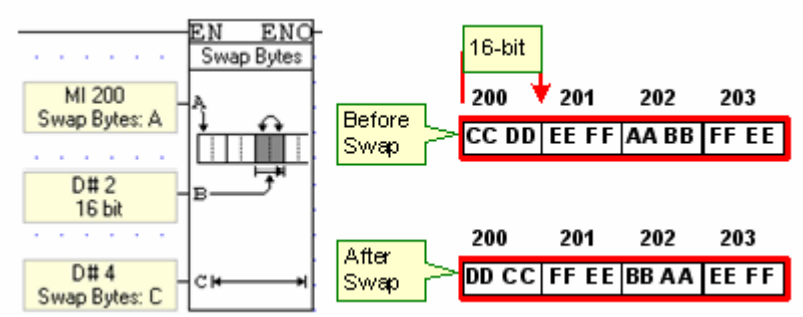
Vector: Swap Bytes

Swap Bytes allows you transpose the bytes within MIs, MLs, and DWs.

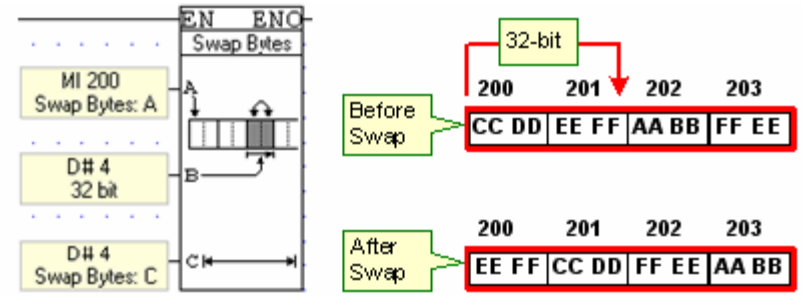
1. Click the Vector menu on the Ladder Toolbar, then select Swap Bytes.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses. Operand A determines the start of the register vector, Operand B whether 16 or 32- bits will be swapped, and Operand C the number of operands that will have their bytes swapped.

The examples below show how the function swaps bytes.

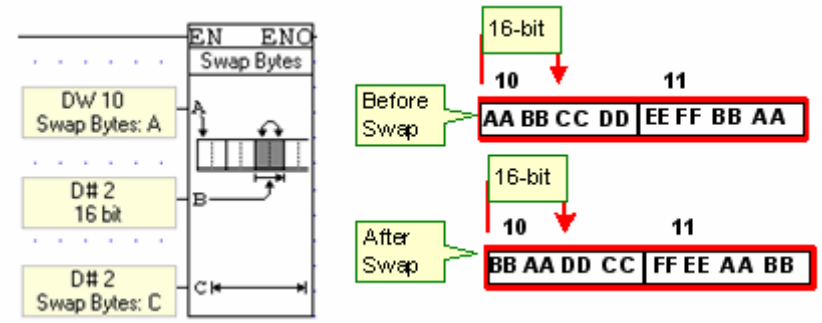
4 MIs, 16-bits



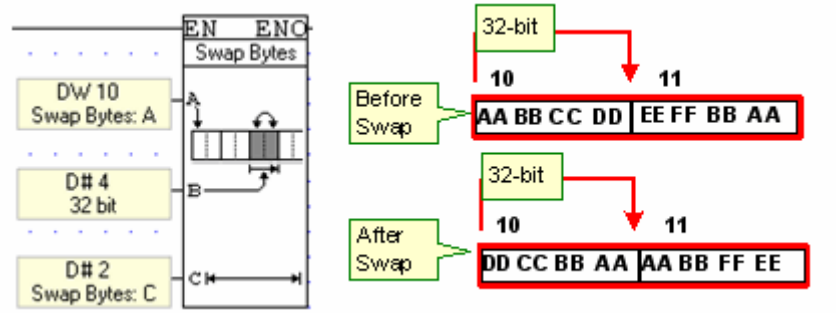
4 MIs, 32-bits



2 DWs, 16-bits



2 DWs,32-bits



Vector: Sort

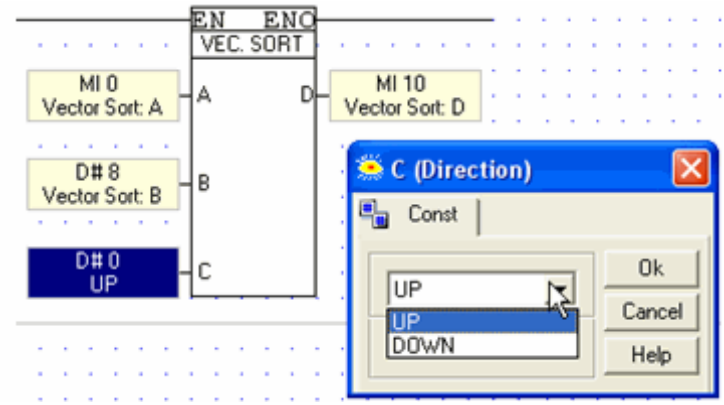
Sort enables you to take a vector of values (MI, ML, or DW) and:

- sort them in ascending or descending order
- either copy the sorted values to a different destination or overwrite them to the original vector.

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

2. Place the function in the desired net.
3. Set Parameter A; link the desired Operand and Address for the MI, ML, or DW vector.
4. Set Parameter B; determine the vector length.
5. Select the Sort direction, Up or Down
6. Link the desired Operands and Addresses for the Vector Sort destination.

The examples below show the function directions.



Vector: Struct

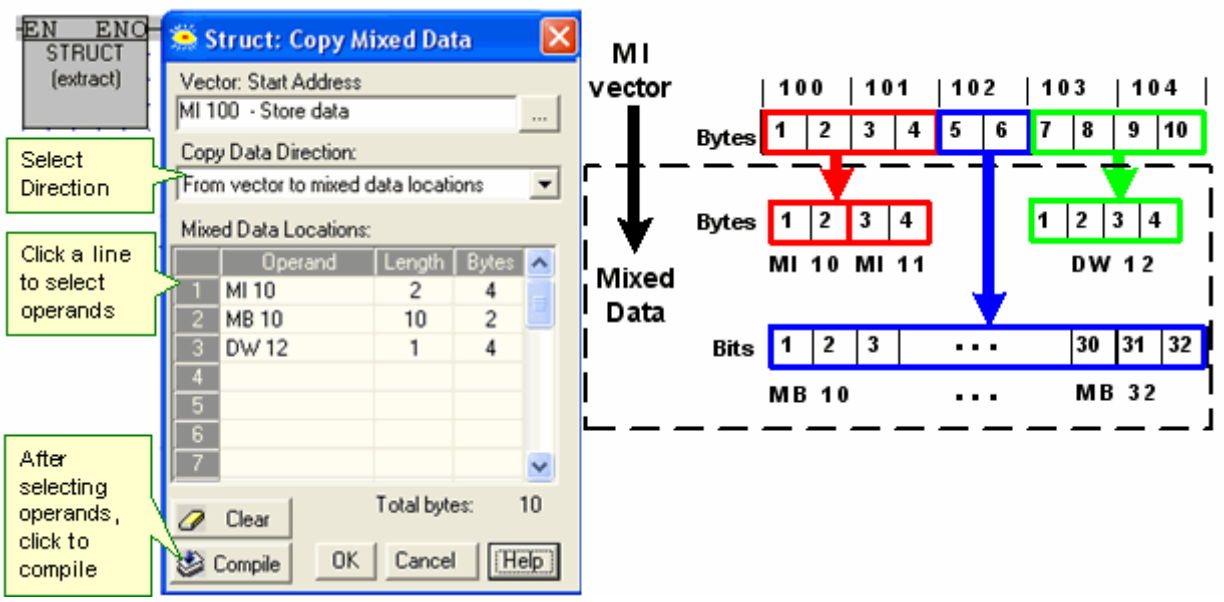
Struct enables you to collect values:

- from a vector of memory operands (MI, ML, or DW) to mixed data locations (MB, MI, ML, MF, I, O, DW), or,
- from mixed data locations to a vector of memory operands.

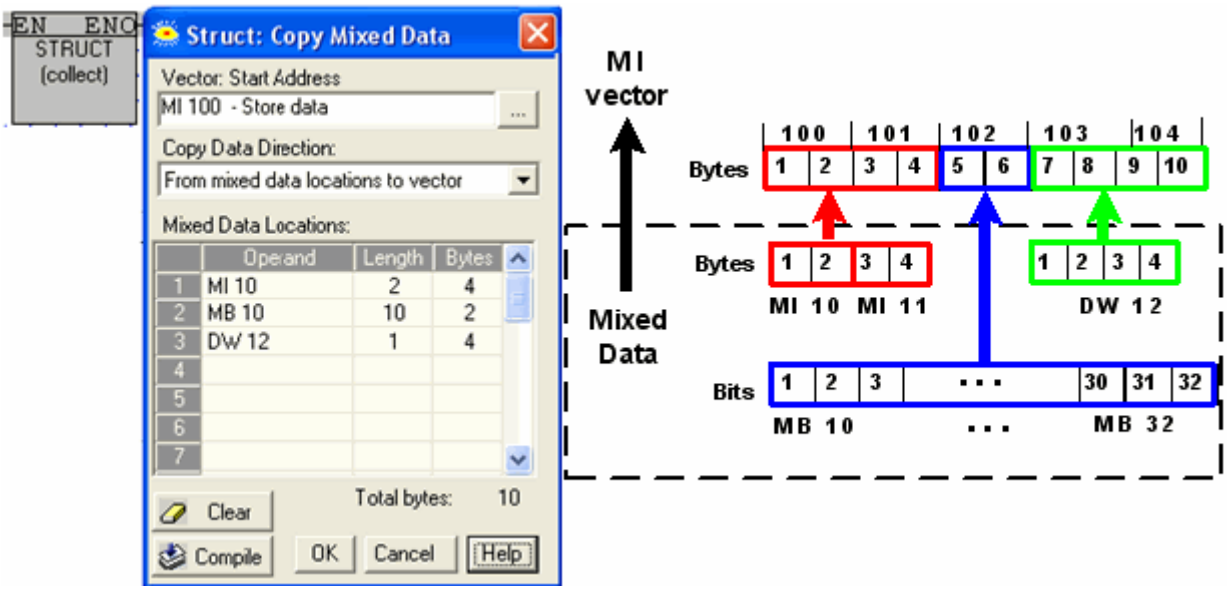
1. Click the Vector menu on the Ladder Toolbar, then select Struct.
2. Place the function in the desired net.
3. Link the desired Operand and Address for the MI, ML, or DW vector.
4. Select the Copy Data Direction:
   -from vector to mixed data locations, or
   -from mixed data locations to vector
5. Link the desired Operands and Addresses for the mixed memory locations.

The examples below show the function directions.

From Vector to Mixed Data Locations



From Mixed Data Locations to Vector



Strings

String operations enable you to manipulate characters.

- Time to ASCII
- Transpose
- Num to ASCII, ASCII to Num
- Display RTC (ASCII)
- IP to ASCII
- Mac Address to ASCII
- Strings: Section Operations
- Set String Library

Strings: Num to ASCII, ASCII to Num

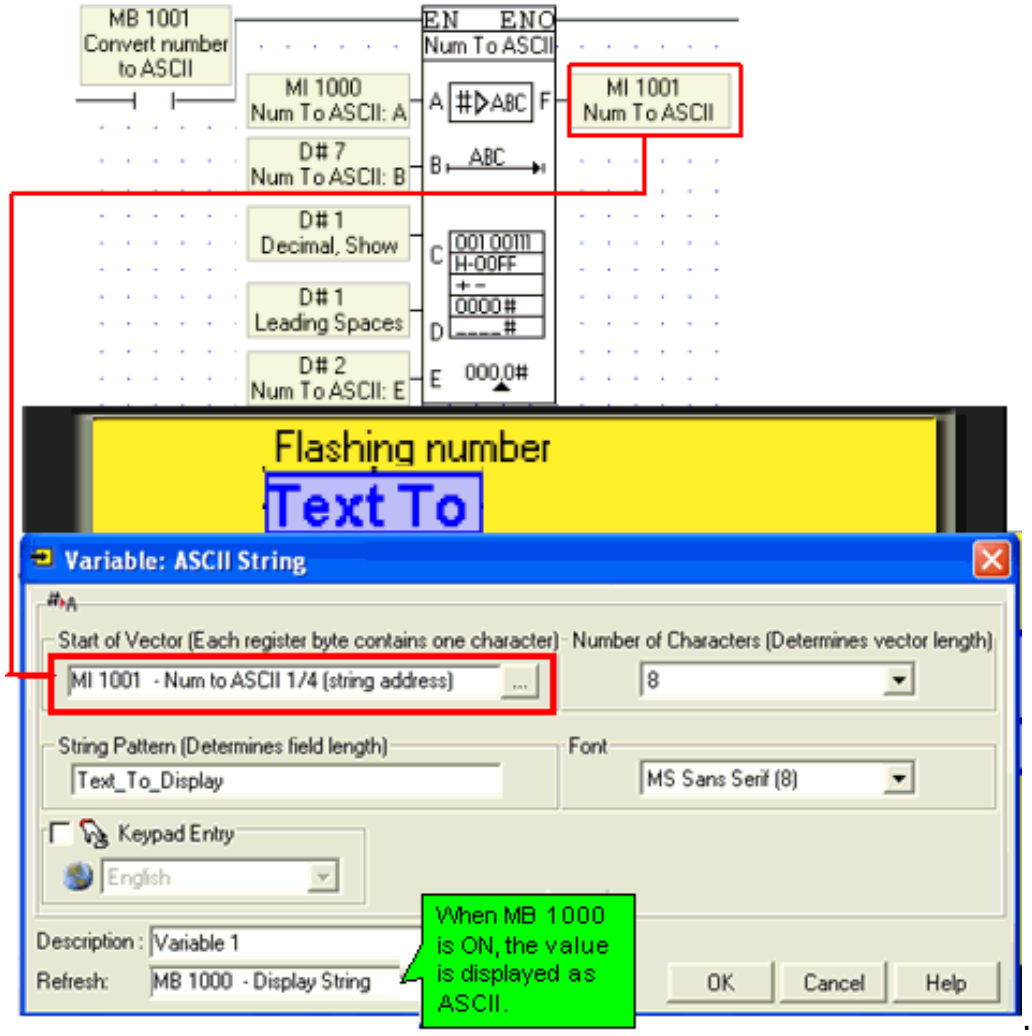
These functions are located on the String menu.

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.

Use this function to convert values, including Float values, to ASCII strings. Note that the vector length you set relates to the number of bytes

- Operand A: Start address for the source vector.

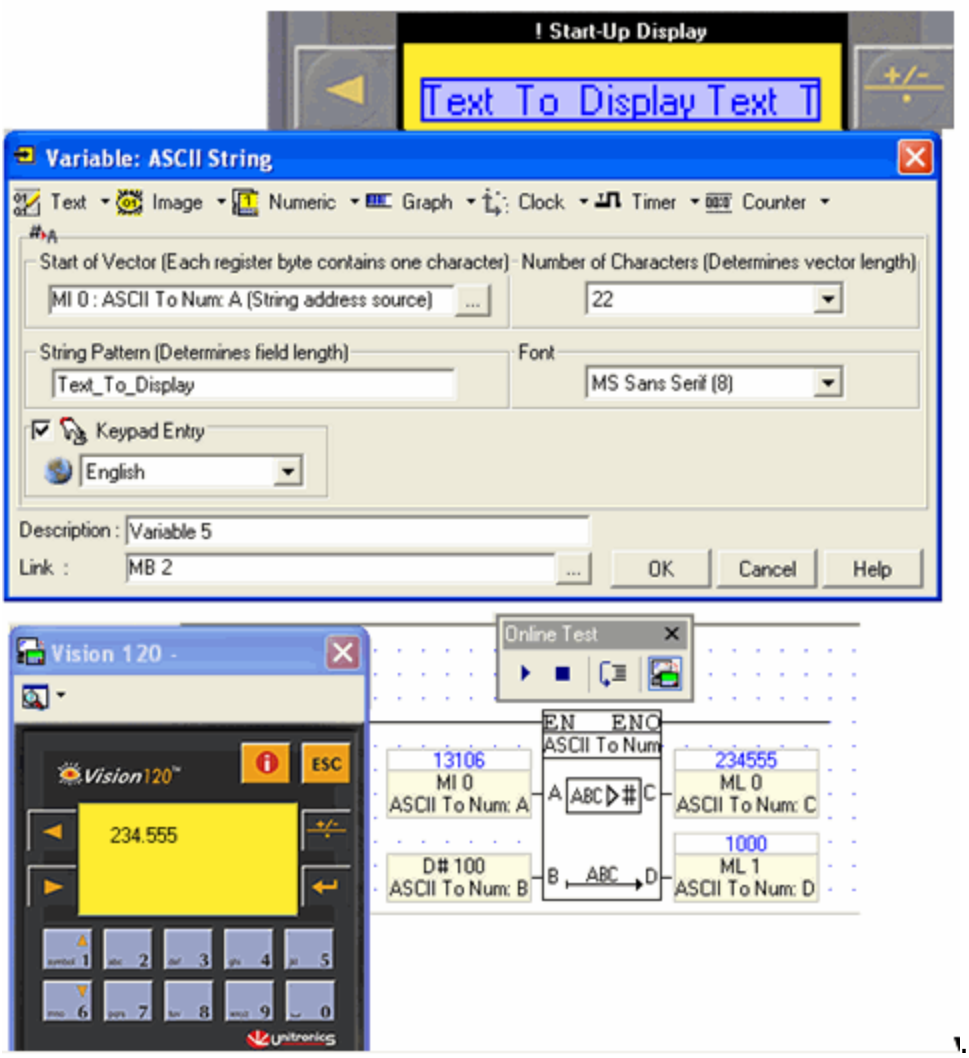
- Operand B: Set the vector length of resulting string (in bytes). Note that the vector must be long enough to contain the value.
- Operand C: Select the format, Decimal, Hex, Binary, or Float.
- Operand D: Select Leading
- Operand B: offset from the start address.
- Operand C: Start address for the target vector.
- Operand D: Leading Zeros
- Operand E: vector length.

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.



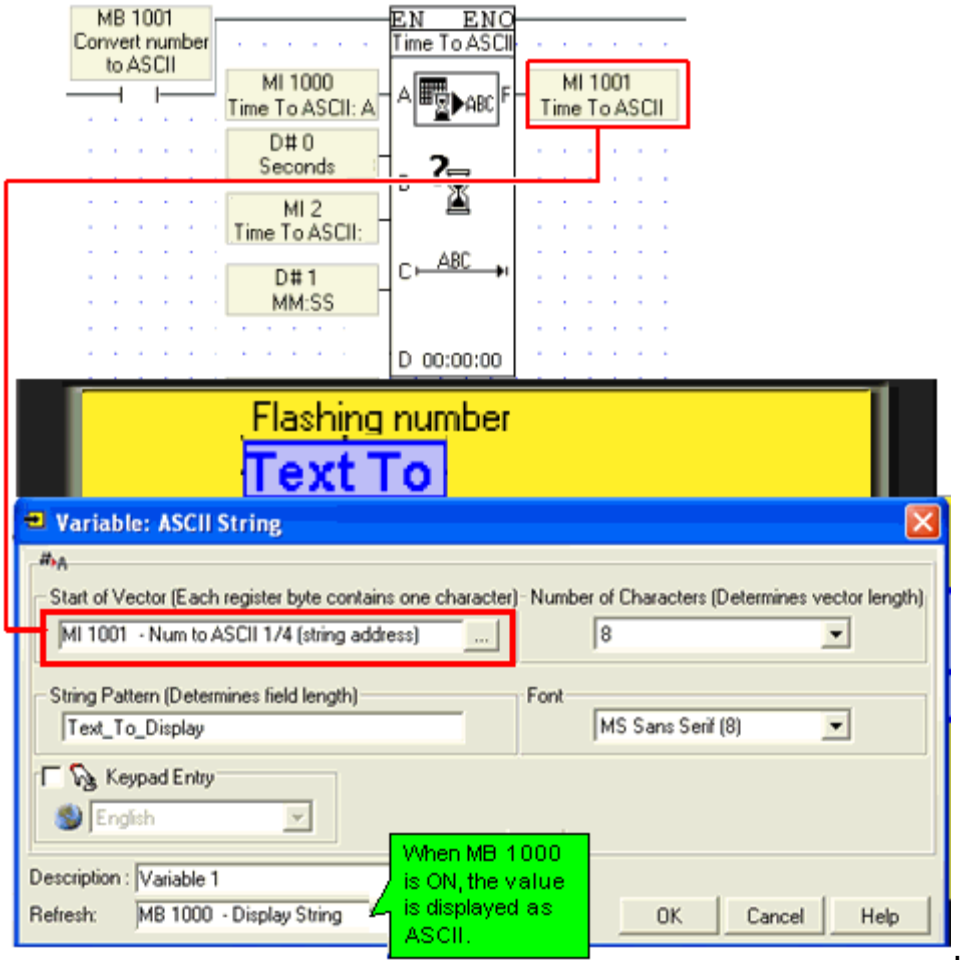
Notes • | ASCII to Num, floating value, is not supported by the V120-12 series.

Time to ASCII

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

1. Select Time 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.



Note • 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 (???????).

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.

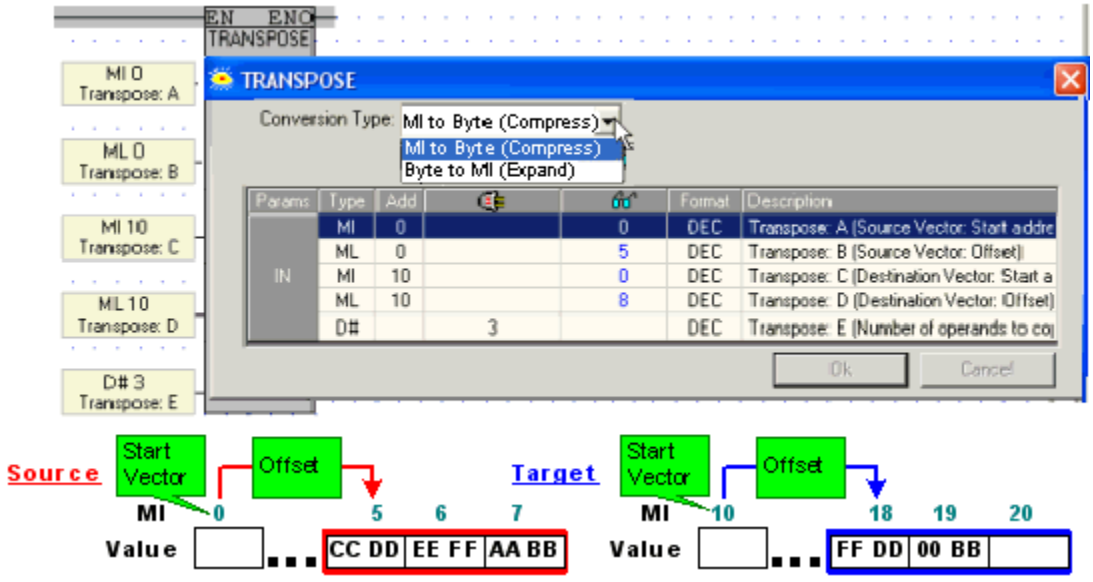
Note • Transpose vector maximum:
MI to Bytes: 128
Bytes to MI 256

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.



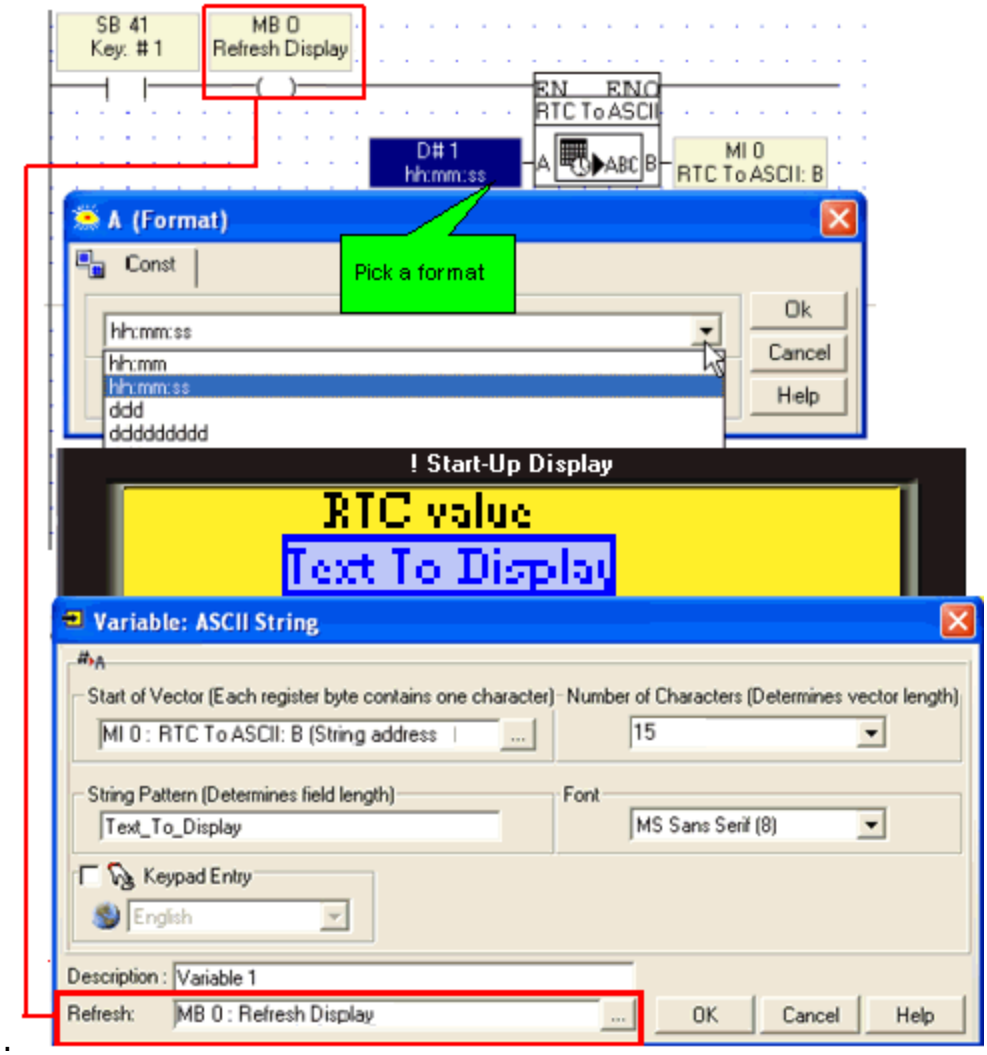
Strings: Display RTC (ASCII)

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

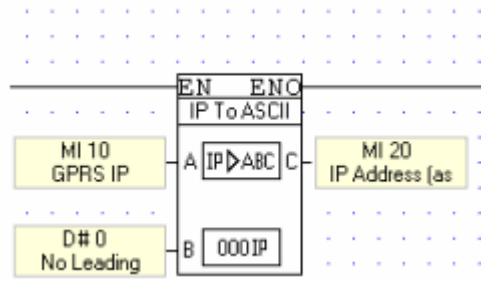
When the program shown below is downloaded, pressing key 1 on the Vision's keypad will display the current time on the Vision's LCD.



Strings: IP to ASCII

You can save a value as an ASCII string by using the Num to ASCII function.

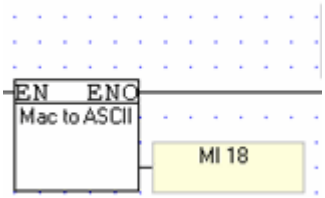
Notes •	If the vector is not long enough, if for example you convert an ML value of "123456" into ASCII and allow only 5 characters, the function returns a string of question marks (???????).
•	This feature is not supported by the V120-12 series.



Mac Address to ASCII

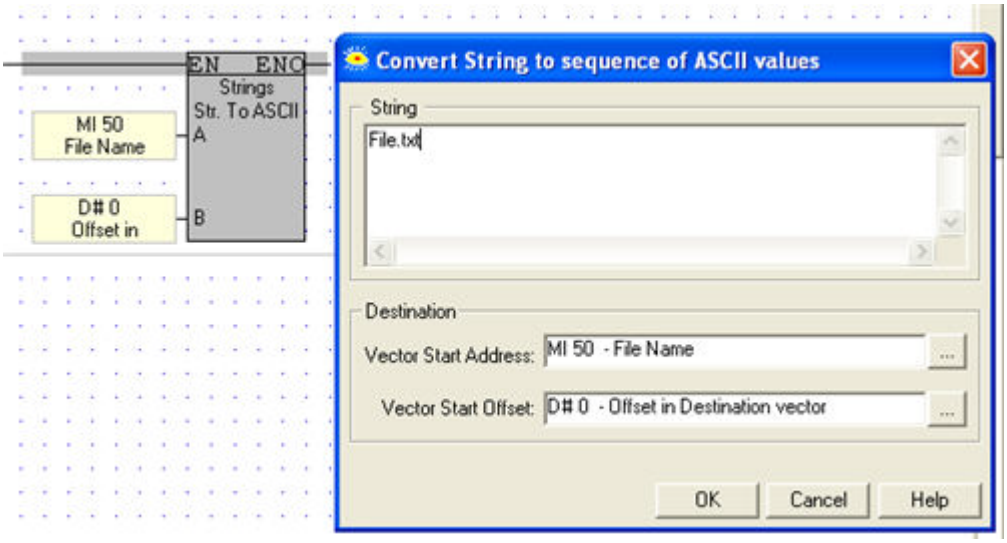
You can store a MAC address as an ASCII string by using the Mac to ASCII function.

Notes •	The MAC address will only be shown if: • The controller contains an Ethernet card • The Ethernet card has been initialized via a TCP/IP function.
•	This feature is not supported by the V120-12 series.



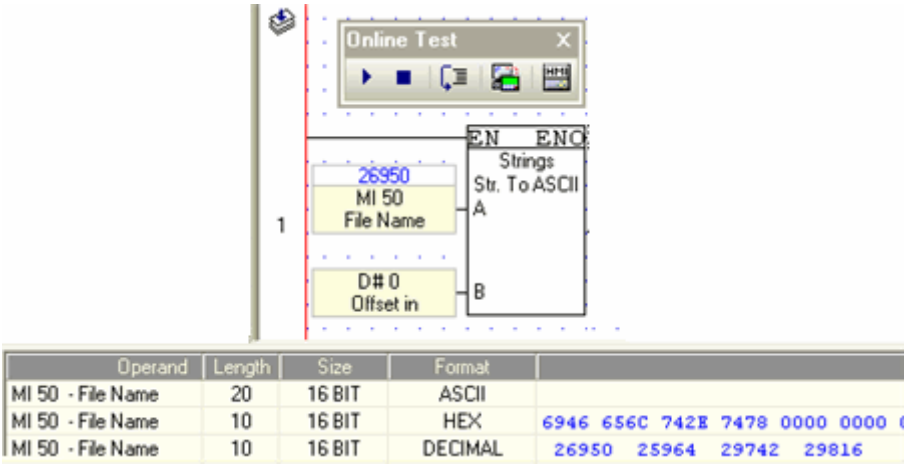
String to ASCII

Use this function to convert a string to an ASCII value.



1. Select String to ASCII from the String menu on the Ladder toolbar, and place the function in the net.
2. Enter the string in the string field
3. Select the register that will contain the ASCII results. You can also use an offset.

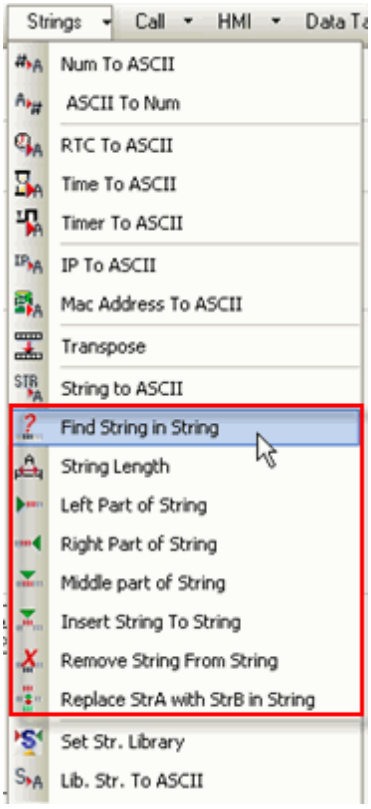
When the function below runs, the values can be seen in the Memory window during Test Mode..



Strings: Section Operations

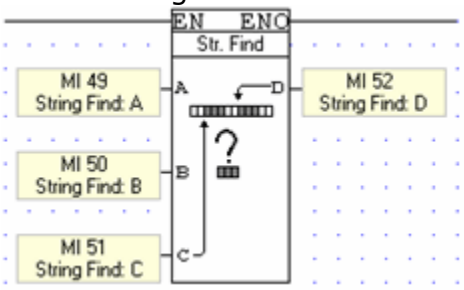
String Section Operations work on parts of strings. To work with these functions, note the following general principles:

- A String always ends with a NULL character.
- The string length (or offset to string) is measured in bytes (characters).
- Parameter A, String Address Source, is the String location. For example, if the string is set to MI 8, the string address is equal to the address of MI 8--not to the content of MI 8.
- Maximum string length is 512 bytes, 120 characters.
- The string cannot extend past the memory type domain (MI, DW, etc.).
- The string offset cannot exceed the string length.



Find String within String

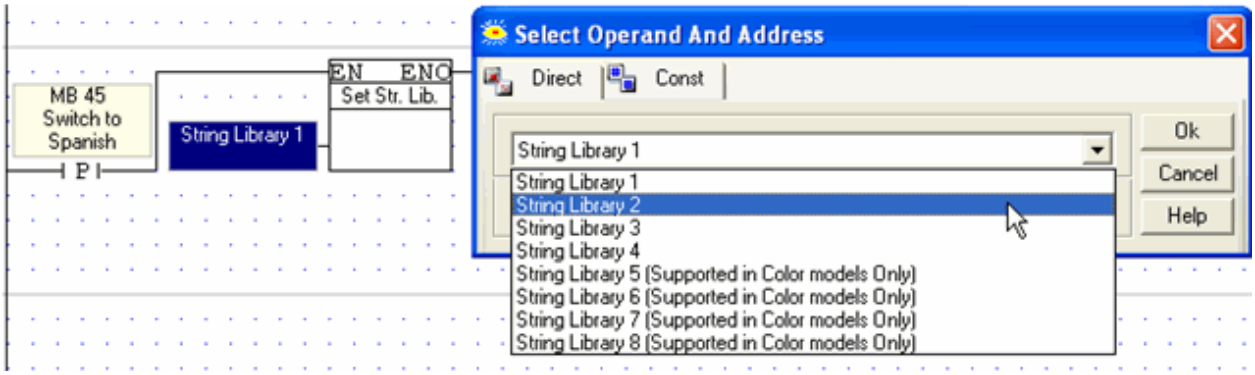
The function shows the location of the first occurrence of the sub string in the source string.



Parameter A:
Str in Str: String to Search in
Parameter B:
Str in Str: String to Search For
Parameter C:
Str in Str: Offset in 'Search IN' vector (bytes)
Parameter D:
Str in Str: Location of found String (-1 = String not found)

Set String Library

Use this function to switch String Libraries.



Utils Menu

This contains a variety of functions:

Calls, Jumps, and Labels

HMI

PTO

Alarms

Clock

Immediate

Debug

Idle

Backup Security

UniVision Licensing

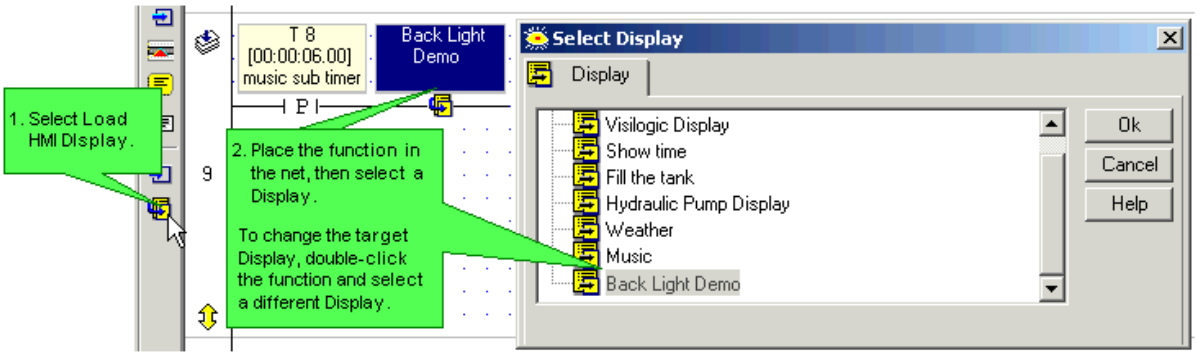
HMI-Ladder: Load HMI Display: Functions

These Ladder functions call HMI Displays. Use these functions to initially load the Display, and then to refresh it when your application requires, as, for example, when you want to update variable display. They are located on the Ladder toolbar, under the HMI menu.

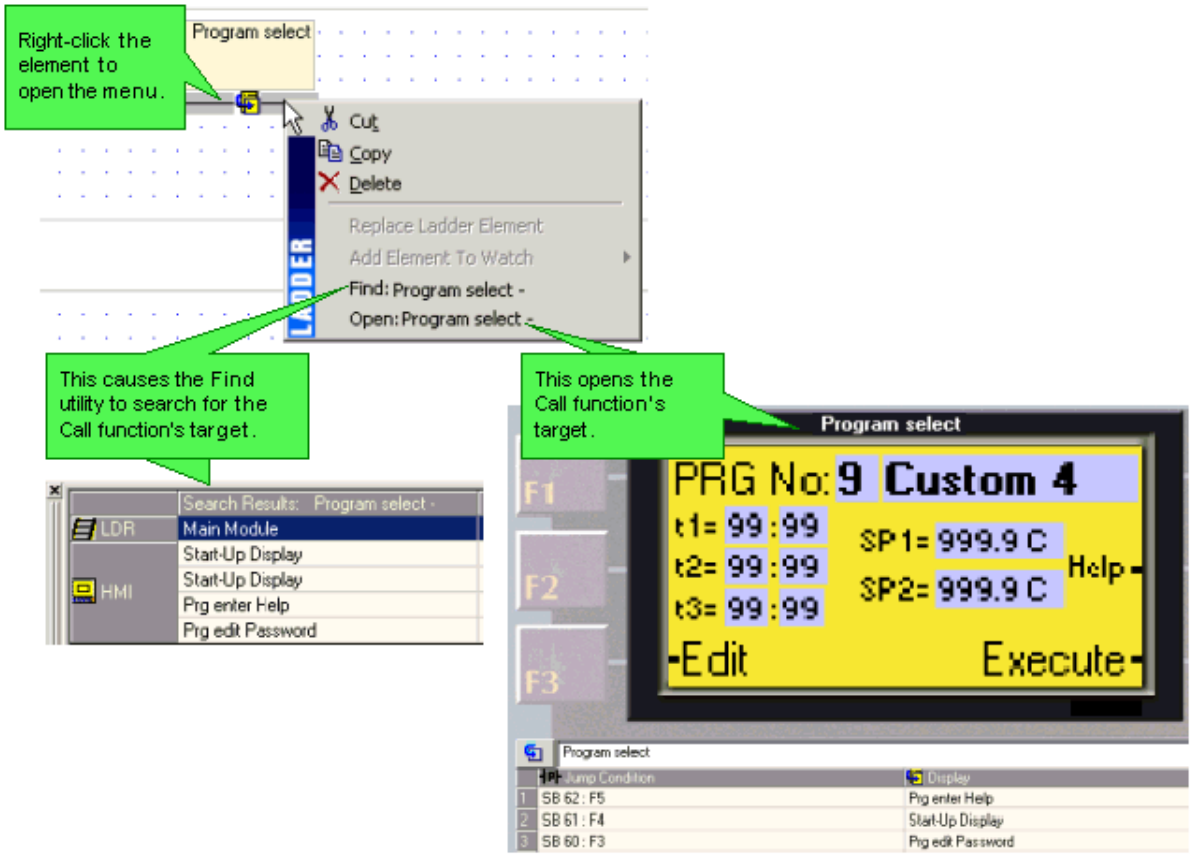
Note •	Load Display functions should not be placed directly on the Ladder rail, or called by conditions that continually call the Display when it is still loaded on the controller screen.
•	You must use a transitional contact to activate a Load HMI Display or Load Last Display function.

Load HMI Display

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

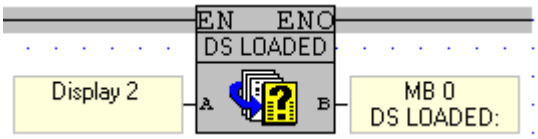


Accessing a Load Display Target



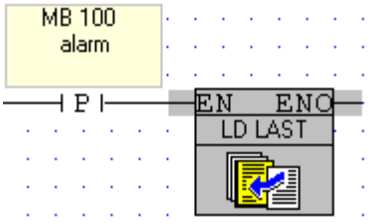
HMI Display Loaded

This turns a linked MB ON when a specific Display begins loading.



Load Last Display

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



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

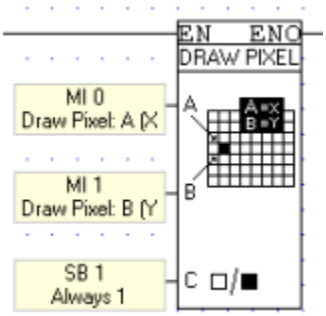
HMI-Ladder: Draw Pixel/Line

These elements allow Ladder events to color pixels or draw lines on the controller's LCD display.

Standard Vision

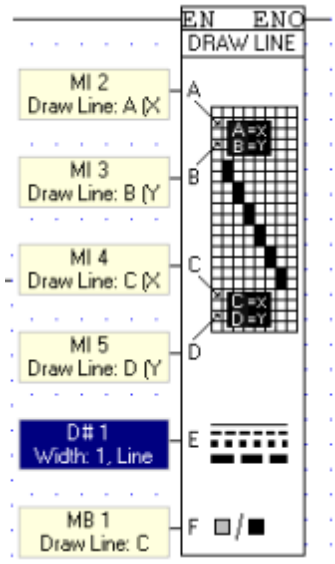
Both Draw Pixel and Draw line are located on the HMI toolbar.

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 used to color the pixel black, SB 0 to color it negative, or an MB may be used.

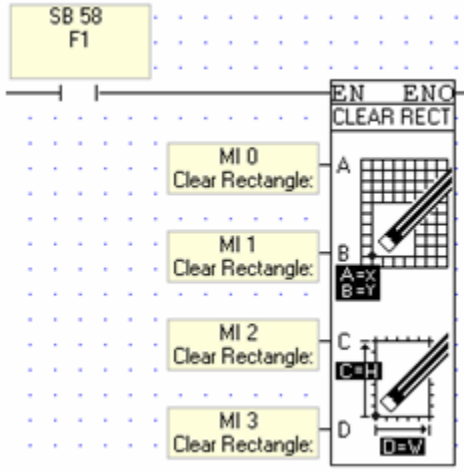
Draw line enables you to draw lines in different widths and formats.



Input	Purpose	Comments
Input A	Start X location	
Input B	Start Y location	
Input C	End X location	
Input D	End y location	
Input e	Format	Select line width: 1 to 4 pixels wide, and line style: solid, dot. or dash. Note that Color Visions support a width of up to 20 pixels
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 used to color the line black, SB 0 to color it negative, or an MB may be used.

HMI-Ladder: Clear Rectangle (Standard Vision only)

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.



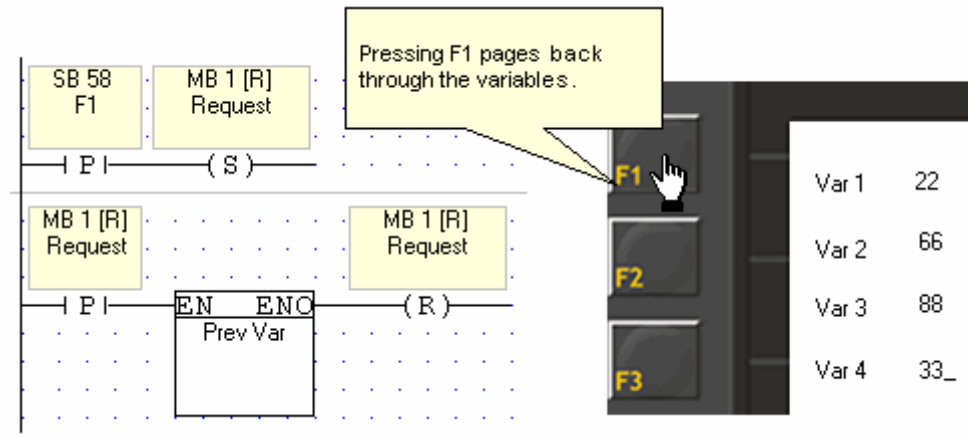
The parameters below set the location and size of the rectangle.

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

HMI-Ladder: Previous Var (Standard Vision only)

This element allows you to use Ladder events to page back through Keypad Entry Variables. Previous Var pages back according to the physical order of the variables on the LCD screen.

In the following figure, if Var 4 is the active variable, pressing F1 once activates Var 3, an additional press activates Var 2, then 1. If Var 1 is active, pressing F1 activates Var 4.



Inverse Var/Hide Var (Standard Vision Only)

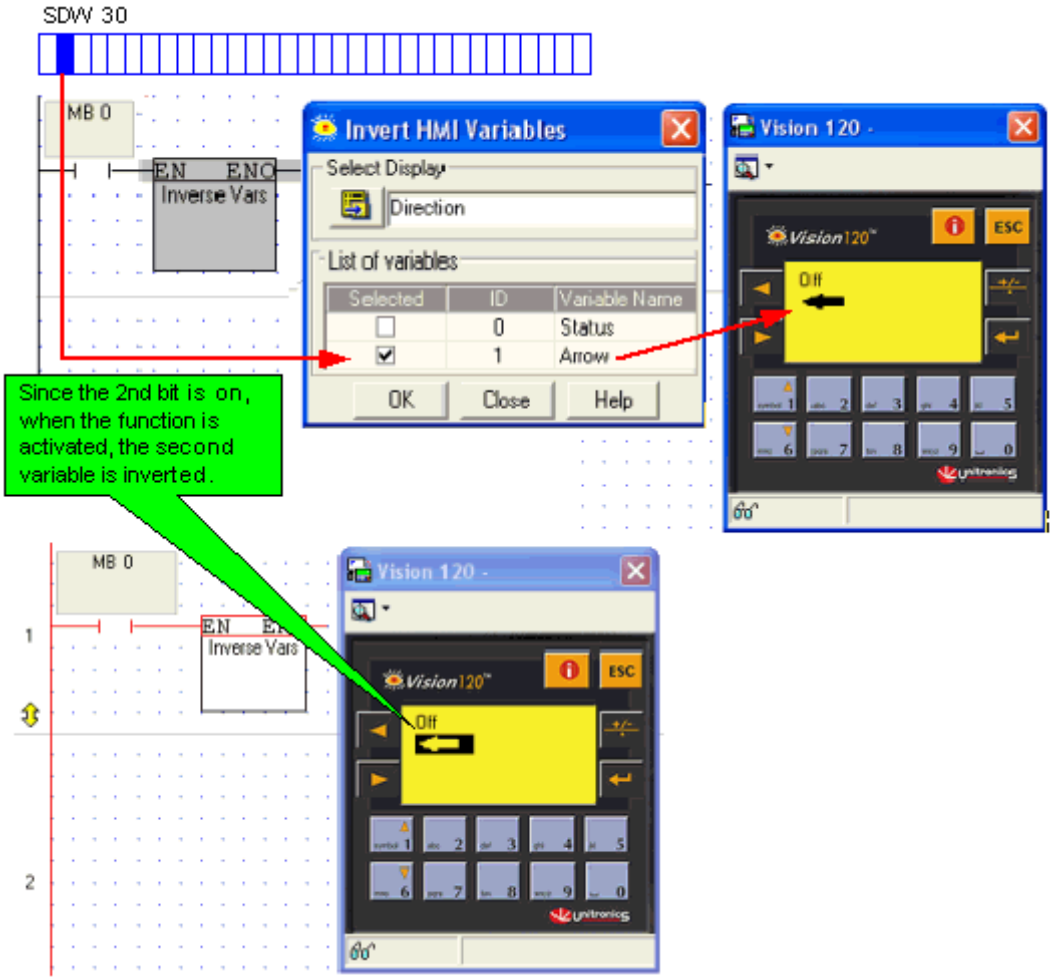
The Inverse Var function 'inverts' the color of a variable, meaning that black pixels are changed to white and white to black.

The Hide Var function hides a variable.

How Inverse/Hide works

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	When a bit is ON, the corresponding variable is hidden.

	Display.

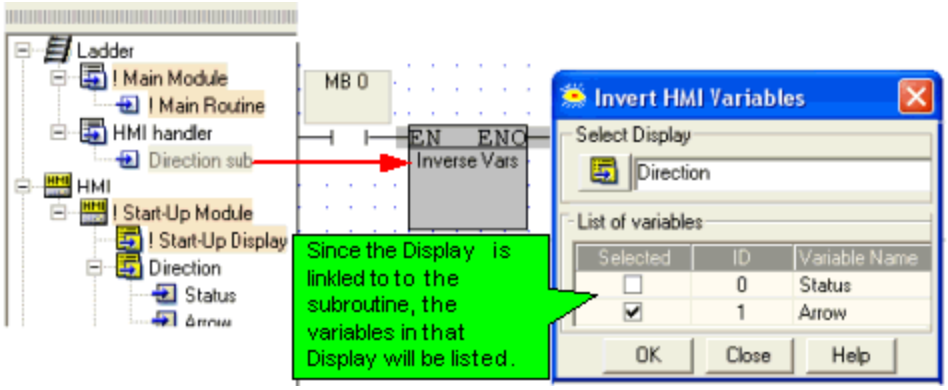
How to use Inverse\Hide

1. Link the Display containing the variables to the desired Subroutine as shown below.

2. 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.

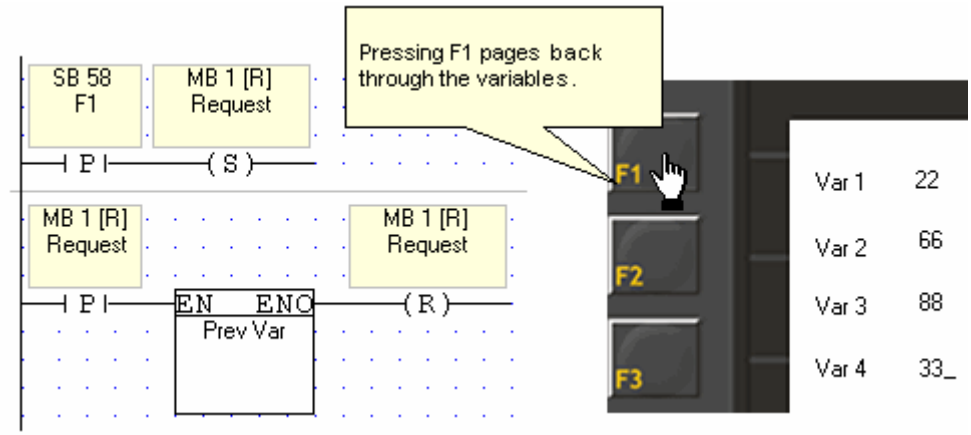


Notes •	The SDW bits are linked to the variable index number, which changes when variables are added or deleted, as well as during copy/paste. If you edit the variables after inserting Inverse/Hide functions, check that the desired variables remain selected.
•	The functions automatically update the variable view of whichever Display is currently on-screen.

HMI-Ladder: Previous Var (Standard Vision only)

This element allows you to use Ladder events to page back through Keypad Entry Variables. Previous Var pages back according to the physical order of the variables on the LCD screen.

In the following figure, if Var 4 is the active variable, pressing F1 once activates Var 3, an additional press activates Var 2, then 1. If Var 1 is active, pressing F1 activates Var 4.


Pressing F1 pages back through the variables.
SB 58
F1
MB 1 [R]
Request
P
S
MB 1 [R]
Request
MB 1 [R]
Request
P
EN
ENO
Prev Var
R
F1
F2
F3
Var 1
22
Var 2
66
Var 3
88
Var 4
33_

Refresh HMI Display

Run this function to redraw the current HMI display.

PTO Functions: Simple Motion Control

You can implement motor control by controlling the high-speed outputs of certain Vision controllers using PTO functions, controlling up to three independent axes.

In this way you can, for example, build speed profiles that are appropriate for stepper motors. Note that the PTO control functions are open-loop, and do not rely on positional feedback.

Supported Modes:

- Pulse
  Uses a single high-speed output
- Pulse + Direction
  Uses 2 high-speed outputs, one for the pulse, and the second to control direction
- Clockwise/Counter Clockwise
  Uses 2 high-speed outputs, one for clockwise, the other for counter-clockwise

Channels and Outputs

A Channel comprises the outputs that are required to implement a Mode.

The number of channels, the possible modes, and the outputs used to implement them vary from model to model. The following tables show the possible combinations, according to Vision model.

Notes •

When an Output is not being used in a channel, it may be used as a general-purpose output (not high-speed)

Caution

These functions are based on programming logic, and therefore do not have the safeguards generally provided by electro-mechanical controls. It is the user's responsibility to implement those safeguards required by his system, such as override and/or emergency stop mechanism.

V130/V350-TR34

Channel	Possible Mode Combinations				
Channel 0	Pulse	Pulse + Direction	Pulse + Direction	Pulse	Clockwise/Counter Clockwise
Channel 1	Pulse	Pulse + Direction	Pulse	Pulse + Direction	Disabled

Channel 2	Pulse	Disabled	Disabled	Pulse	Pulse

Channel	**Output used per Channel**				
Channel 0	Pulse (O0)	Pulse (O0) + Direction (O2)	Pulse (O0) + Direction (O2)	Pulse (O0)	Clockwise (O0) / Counter Clockwise (O1)
Channel 1	Pulse (O1)	Pulse (O1) + Direction (O3)	Pulse (O1)	Pulse (O1) + Direction (O3)	Disabled
Channel 2	Pulse (O2)	Disabled	Disabled	Pulse (O2)	Pulse (O2)

V130/V350-TRA22

The following table shows all of the possible PTO mode combinations for this model.

Channel	**Possible Mode Combinations**				
Channel 0	Pulse	Pulse + Direction	Pulse + Direction	Pulse	Clockwise/Counter Clockwise
Channel 1	Pulse	Pulse + Direction	Pulse	Pulse + Direction	Disabled

Channel	**Output used per Channel**				
Channel 0	Pulse (O0)	Pulse (O0) + Direction (O2)	Pulse (O0) + Direction (O2)	Pulse (O0)	Clockwise (O0) / Counter Clockwise (O1)
Channel 1	Pulse (O1)	Pulse (O1) + Direction (O3)	Pulse (O1)	Pulse (O1) + Direction (O3)	Disabled

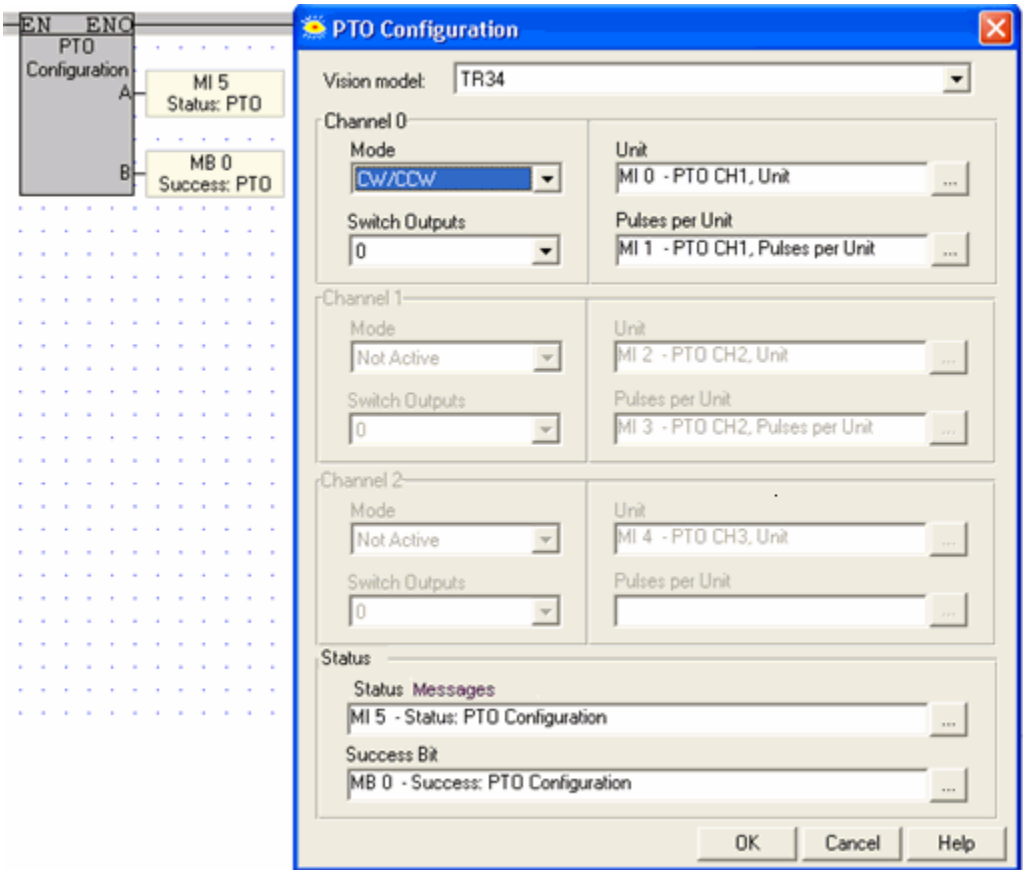
V130/V350-TR20, TR6

Channel	**Possible Mode Combinations**		
Channel 0	Pulse	Pulse + Direction	Clockwise/Counter Clockwise
Channel 1	Pulse	Disabled	Disabled

Channel	**Output used per Channel**		
Channel 0	Pulse (O0)	Pulse (O0) + Direction (O1)	Clockwise (O0) / Counter Clockwise (O1)
Channel 1	Pulse (O1)	Disabled	Disabled

PTO Configuration

In this function you select a Vision model, which determines the available Channels and Modes. Those not available are disabled.

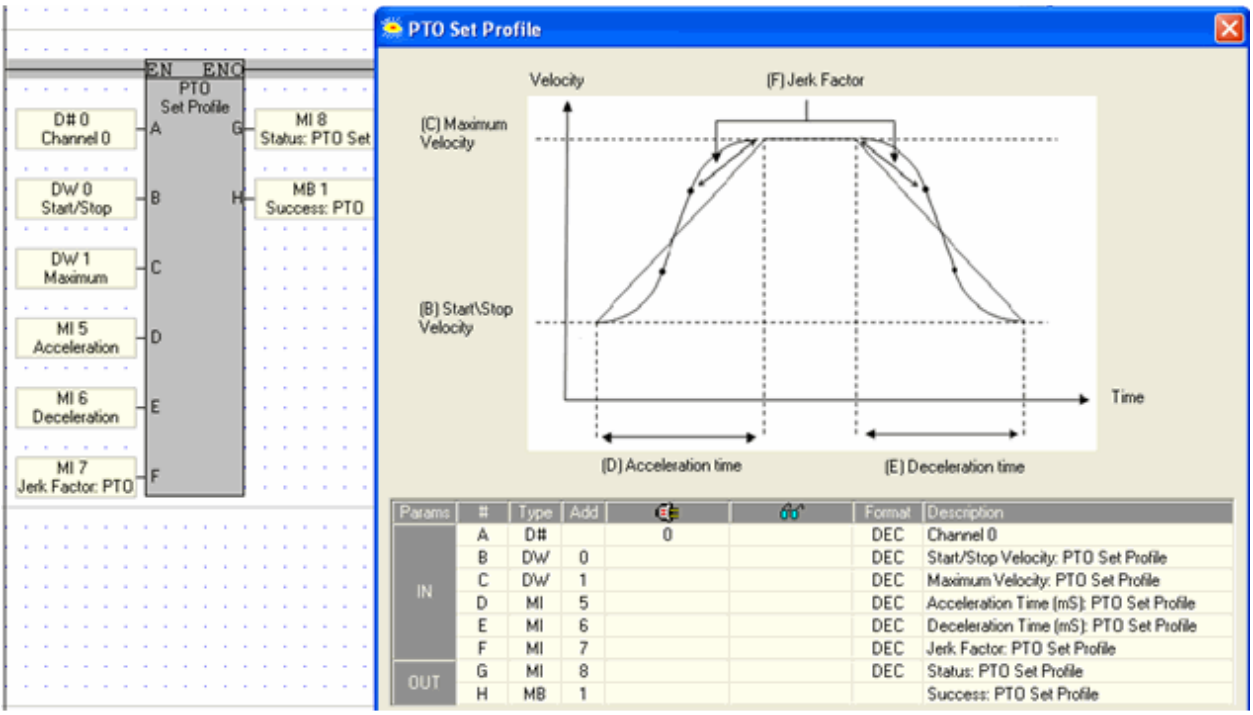


Parameter Name	Purpose
Vision Model	Select the appropriate model
Channel	A Channel comprises the inputs used to carry out the PTO function, and determines their function
Mode	The possible modes are: • Pulse • Pulse + Direction • Clockwise/Counter Clockwise The tables in the previous section give all possible combinations and output assignments, based on model.
Switch	Switch reverses the tasks of the PTO outputs that are assigned to the channel in modes Pulse + Direction or CW/CCW. This can be helpful to fix cases where the output wiring is reversed.
Unit	PTO functions rely on Units. This is where you determine the number of pulses per Unit. Note that neither values for Unit nor Pulses per Unit may exceed 1000 **Note -** To control your output using straight frequency, set 1 pulse = 1 unit. Calculate Frequency to Units according to the following $\text{Units per Second} \times \frac{\text{Pulse}}{\text{Units}} = \text{Pulses per Second}$

	$\text{Units} \times \frac{\text{Pulse}}{\text{Unit}} = \text{Frequency}$
Status Messages	0 - No error 1 - Invalid configuration data 2 - VisiLogic/OS mismatch; this OS version 3 - Vision outputs do not support function 4 - Invalid structure 5 - Invalid configuration channel 6 - Unit or Pulse per Unit exceed limits (1-1000) 7 - Channel already initialized 8 - Currently in motion (function cannot be performed during acceleration or deceleration)
Success Bit	Turns ON when the Status MI =0

Set Profile

Use Set Profile to define the motion profile for a particular Channel in the configuration.

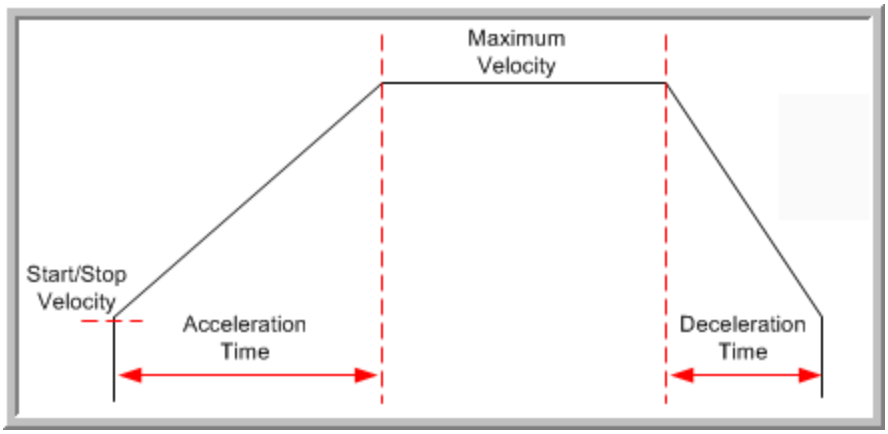


Ranges

Note the minimum and maximum ranges for your motion profile.

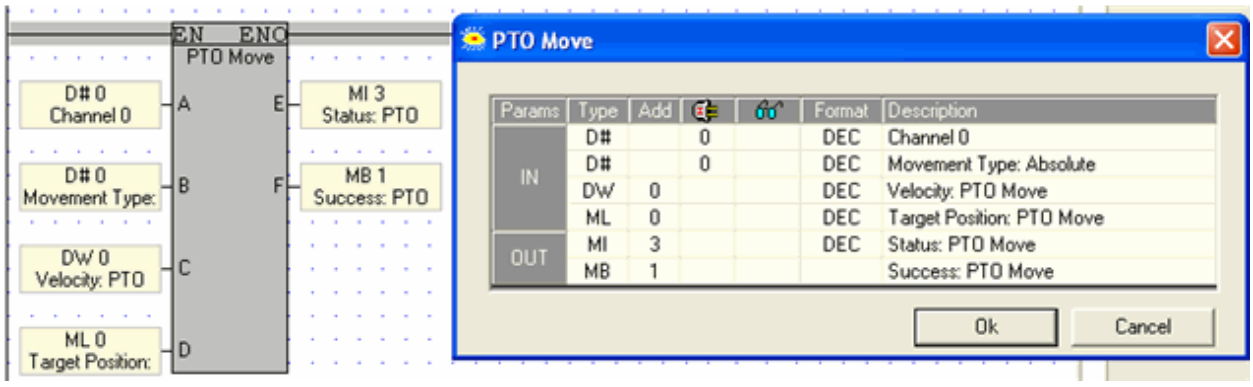
Minimum	Maximum
5 Hz	15 kHz
10 Hz	20 kHz
305 Hz	133 kHz
610 Hz	200 kHz

Parameter Name	Purpose

Channel	Select the relevant channel
Start/ Stop Velocity Maximum Velocity Acceleration Time (mS) Deceleration Time (mS)	These parameters determine the limits of the motion profile for the channel. Note that the resolution of velocity is according to the units set in the PTO Configuration. 
Jerk Factor	Controlling for jerk influences the shape of your motion curve. Legal values are from 1 to 16, where 1=trapezoidal curve and 16 =perfect S.
Status Messages	0 - OK 1 - Invalid configuration data 2 - Currently in motion (function cannot be performed during acceleration or deceleration) 3 - Invalid channel 4 - PTO Configuration block does not exist 5 - Out of range 6 - Maximum value is out of range
Success Bit	Turns ON when the Status MI =0

PTO Move

In this function you determine the parameters of movement.

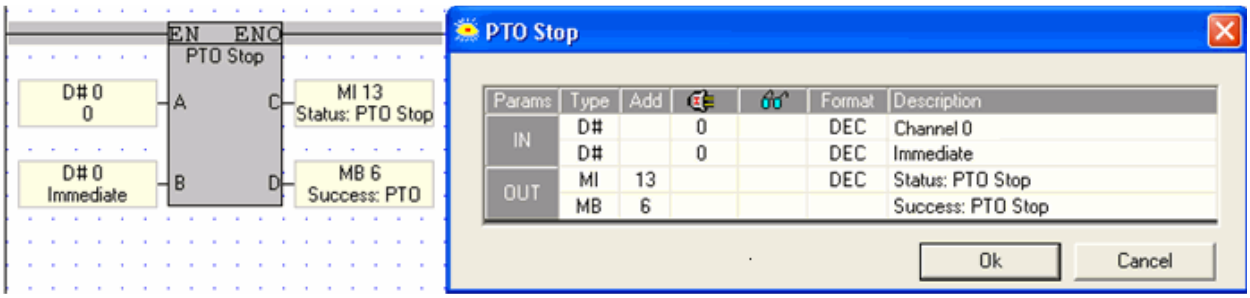


Parameter Name	**Purpose**
Channel	Select the relevant channel

Movement Type	This sets the type of movement: • Absolute Position This causes movement to the exact position requested, without considering the current position. • Relative Position Here the movement is relative to the current position.
Velocity	Note that the resolution of velocity is according to the units set in the PTO Configuration
Target Position	Sets the desired goal
Status Messages	0 - Idle / OK 1 - Configuration data is invalid 2 - Invalid channel 3 - Channel not initialized, or Vision outputs do not support function 4 - Absolute Movement cannot be performed 5 - Currently in motion (function cannot be performed during acceleration or deceleration)
Success Bit	Turns ON when the Status MI =0

PTO Stop

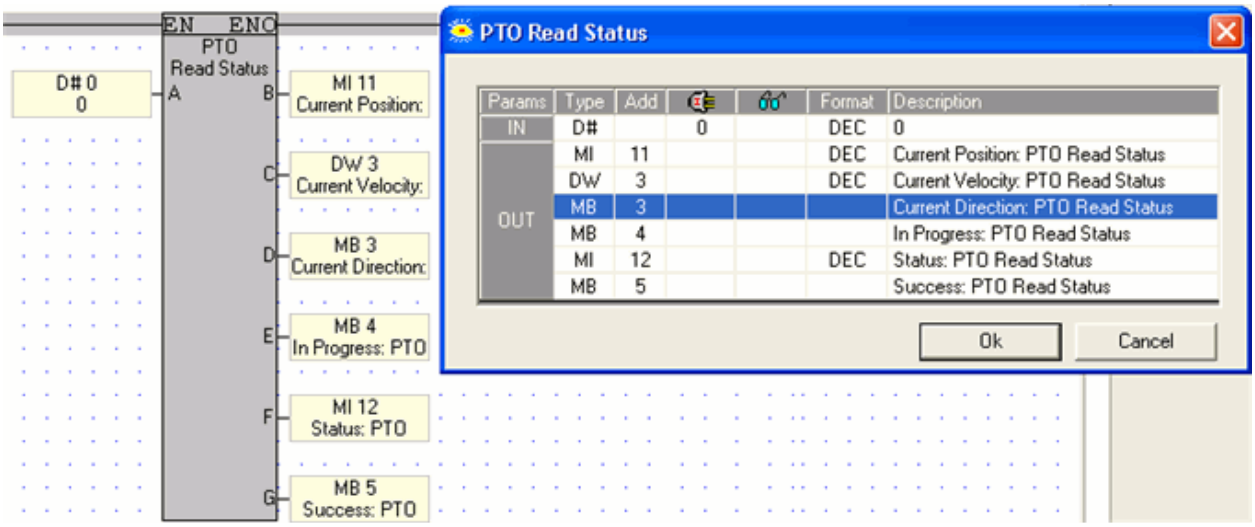
Use this to stop movement



Parameter Name	**Purpose**
Channel	Select the relevant channel
PTO Stop	• Immediate Intended to cause an immediate, emergency stop with no regard for position or any other parameter (requires parameter reset) • Normal Stops motion according to the rate of deceleration set in the PTO Configuration.
Status Messages	0 - Idle / OK 1 - Already stopped 2 - Invalid channel 3 - Channel isn't initialized 4 - unknown command
Success Bit	Turns ON when the Status MI =0

Read Status

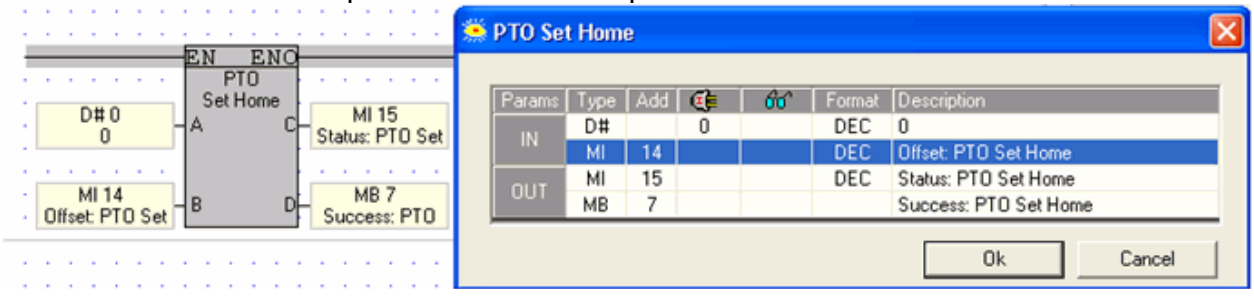
Use this to ascertain the current position.



Parameter Name	Purpose
Channel	Select the relevant channel
Current Position	Use these as a reference for Move functions Note that the resolution of velocity is according to the units set in the PTO Configuration
Velocity	
In Progress	This turns Off after the values have been read.
Status Messages	0 - Idle / OK 1 - Currently in motion (function cannot be performed during acceleration or deceleration) 2 - Channel is not configured 3 - Invalid channel 4 - Read Timeout
Success Bit	Turns ON when the Status MI =0

PTO Set Home

Use this to set a Home position for Move operations set to Absolute Position



Parameter Name	Purpose
Channel	Select the relevant channel
Offset: PTO Set Home	The channel uses this value to set the reference point for the next move operation. If, for example, the Absolute target is set to 600, and the Offset to 200, the channel will move to 400.
Status Messages	0 - OK 1 - invalid channel 2 - precondition error 3 - Channel is currently accelerating or decelerating (Movement can only be performed when system is Idle of in steady state)
Success Bit	Turns ON when the Status MI =0

Alarms: Ladder Functions

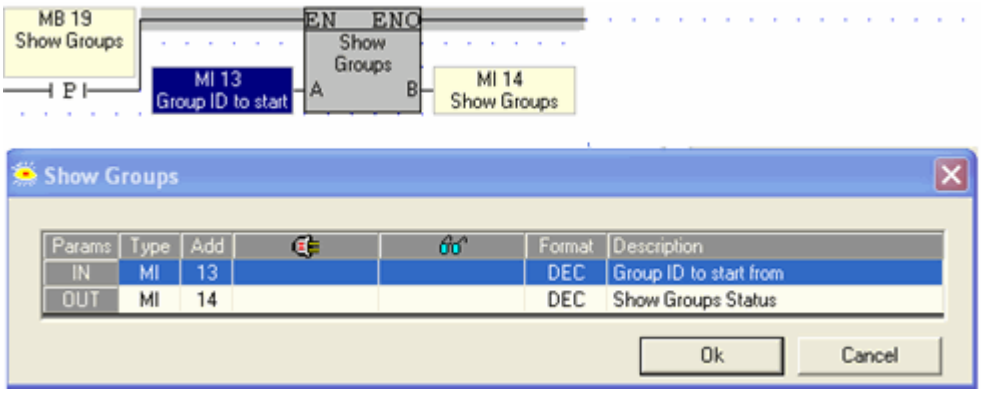
The Alarms displays are shown according to the Ladder application. When the Ladder application calls the Alarms, the displays will only appear if the Alarms are Active.

The functions are located on the Alarms menu in the Ladder toolbar.

Show Groups

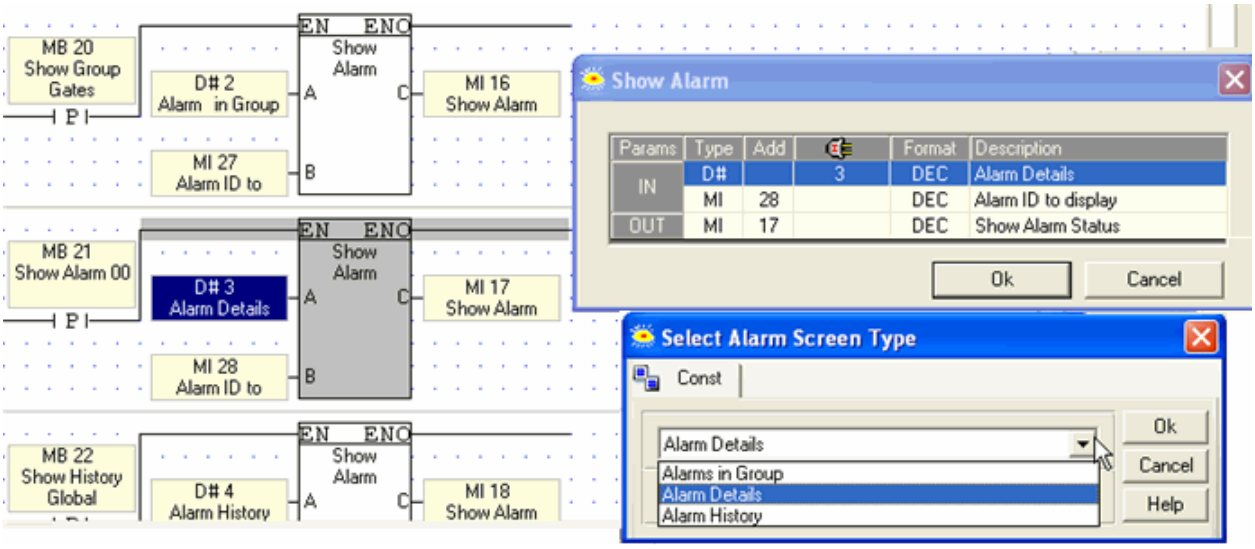
This function shows the Alarms in Group display, according to the number in the MI Group ID to Start From.

Note that the status MI will read 0 if no such group exists.



Show Alarm

This function can show a specific display for a specific Alarm. You can show the Alarm in the Alarms in Group display, or go directly to the Alarm Details or History.



Clear History Buffer

Use this function to erase the Alarm History.

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

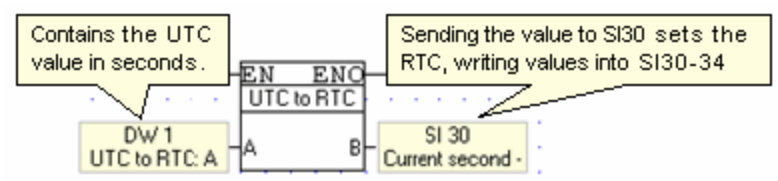
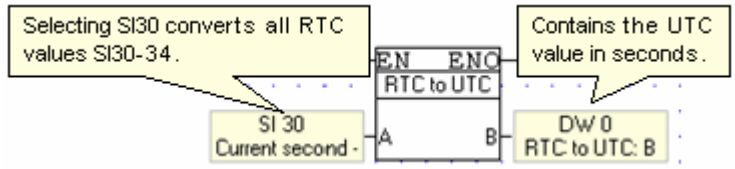
Setting a Clock Function's Time or Date

- **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	UTC to RTC 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.

	
	RTC to UTC Selecting SI 30 will convert the RTC value into a DW. 
Com>TCP/IP menu	RFC-1305 Retrieves, via Ethernet UDP, the current time from a PC UTC server. This may be used to synchronize a Vision RTC with UTC.
HMI Clock Variables	Clock Display Variable, UTC This may be set as read only, or as a Keypad Entry variable used to set the RTC.

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.

About Universal Time (RFC-868, RFC-1305)

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 timestamps 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)

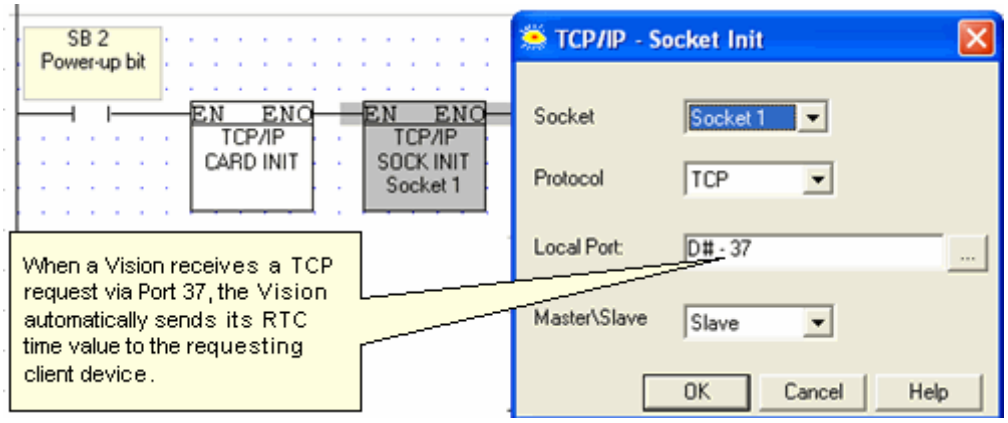
Using RFC-868 to synchronize networked controllers

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.

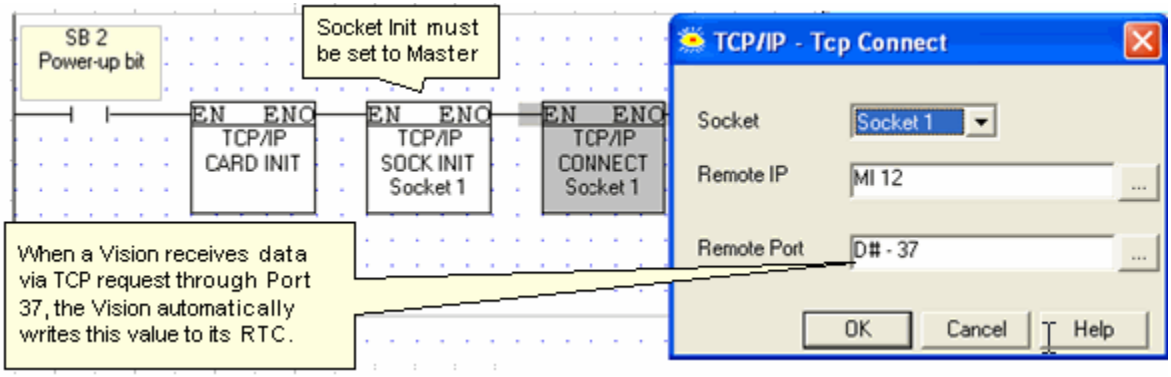
When a Vision receives a TCP request via Port 37, the Vision automatically sends its RTC time value to the requesting client device.



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

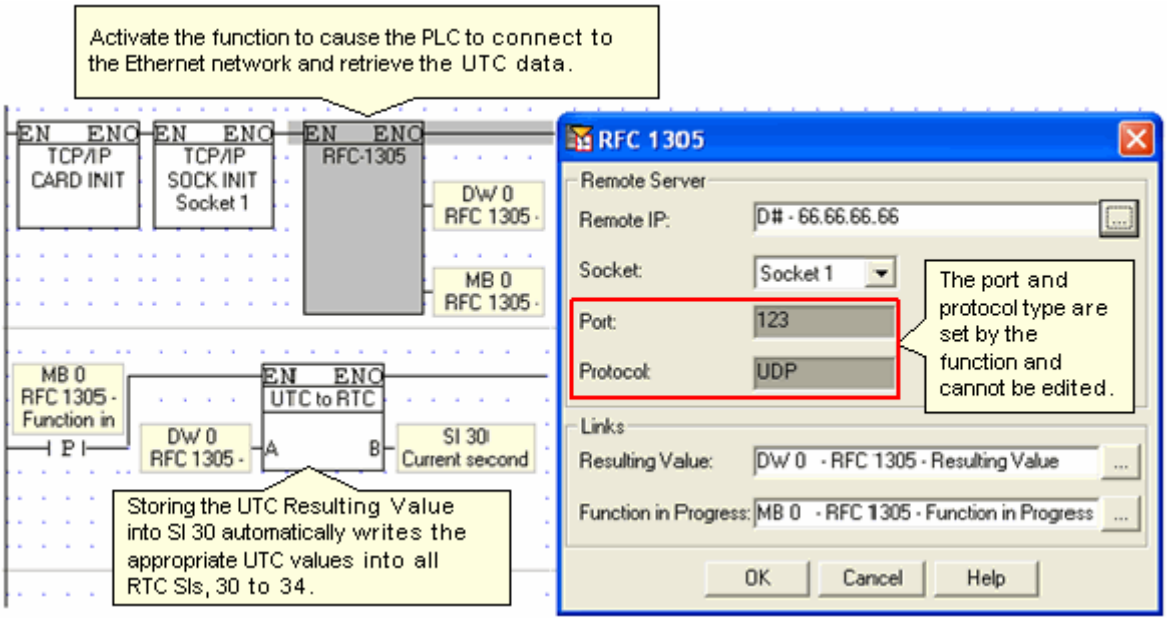
When a UTC PC server receives a UDP request via Port 123, the server automatically sends the time value to the requesting client device.

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

1. Initialize the TCP/IP card 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.



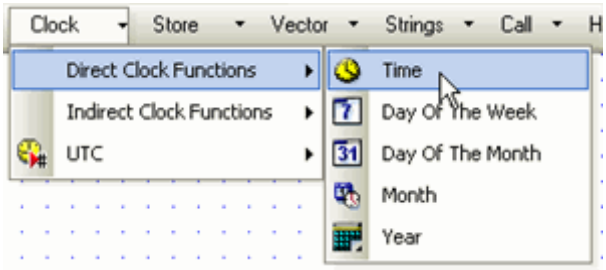
Clock: Direct Function Example

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

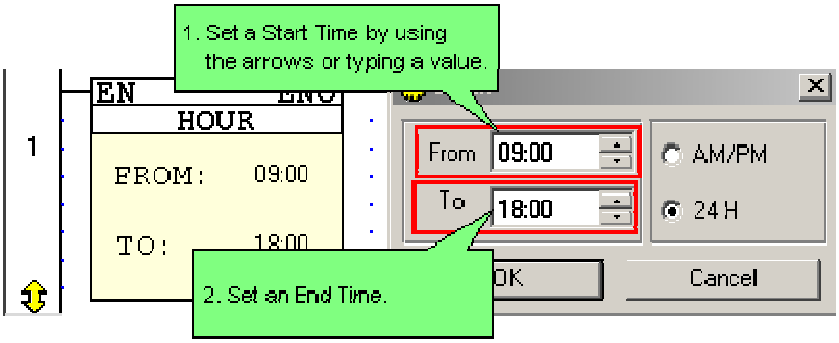
- between the hours 9:00 am and 6:00 PM.
- Monday through Friday
- beginning on the 15th day of a month, until and including the 24th
- in the years 2000 and 2001

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

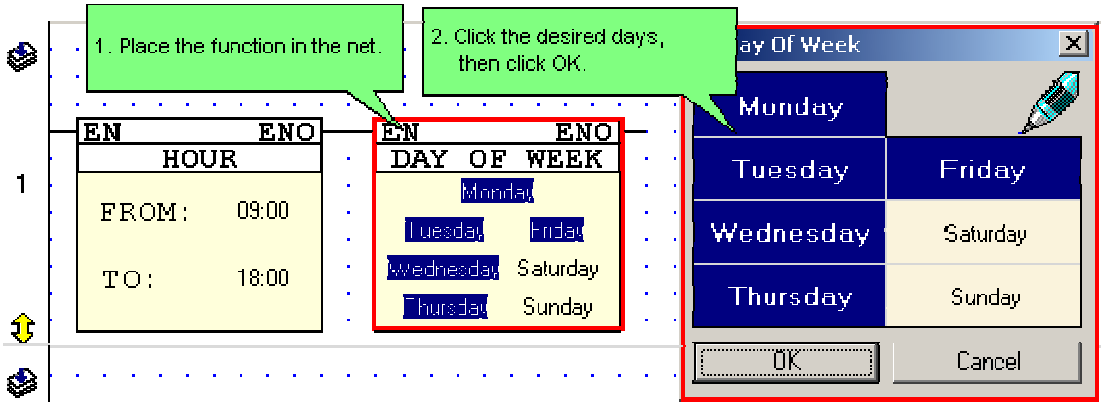
1. Place a Direct Time Function in the net.



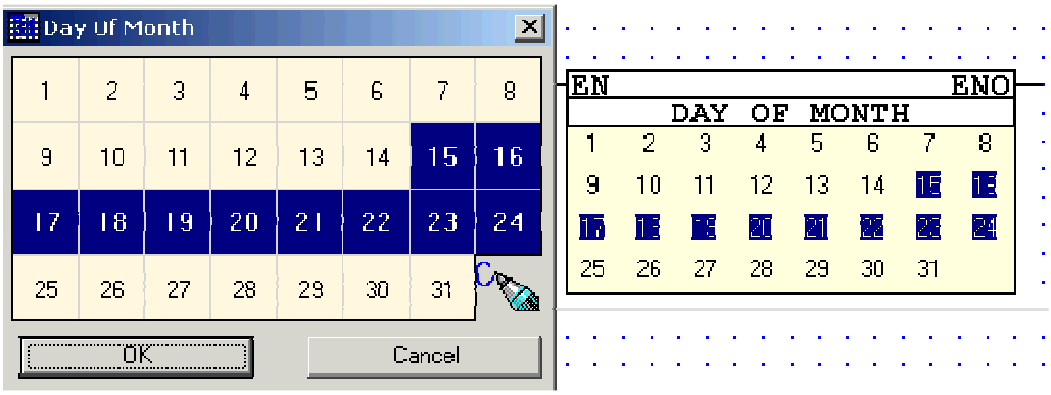
2. Set a Start and End Time. When the RTC is within this range, power flows through the function block.



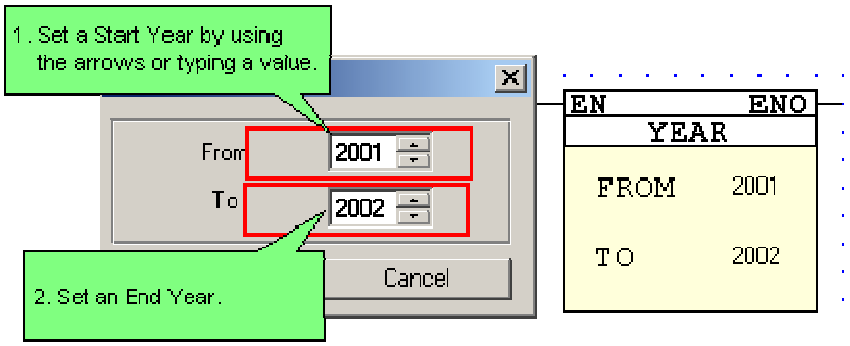
3. Select **Day of the Week** , place it in the net, then select the desired days.



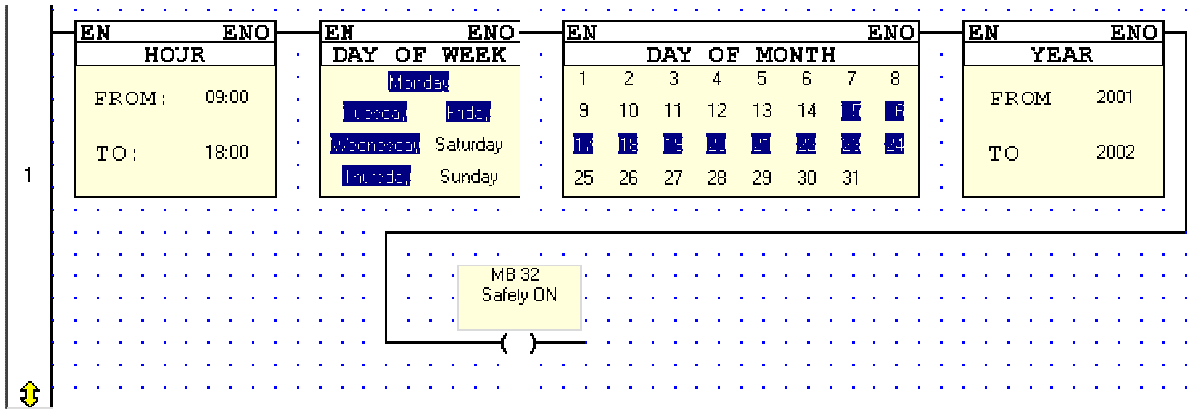
4. Select **Day of Month** , place it in the net, then select the desired dates.



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

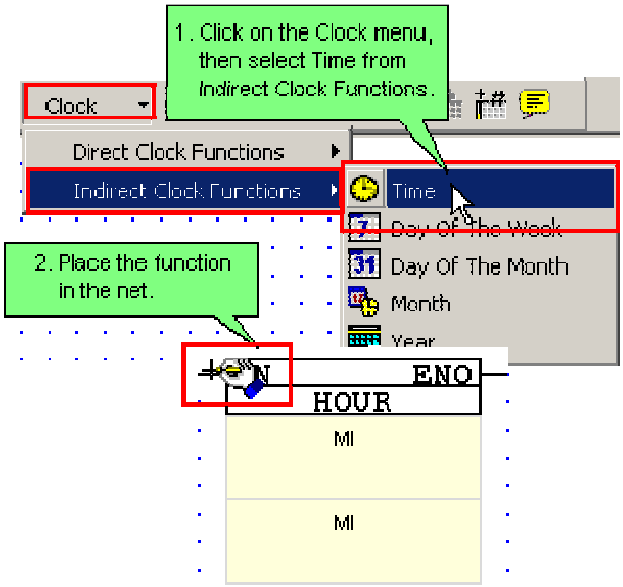
To enable times and dates for tasks or programs to be set from the controller keypad, you:

- 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.

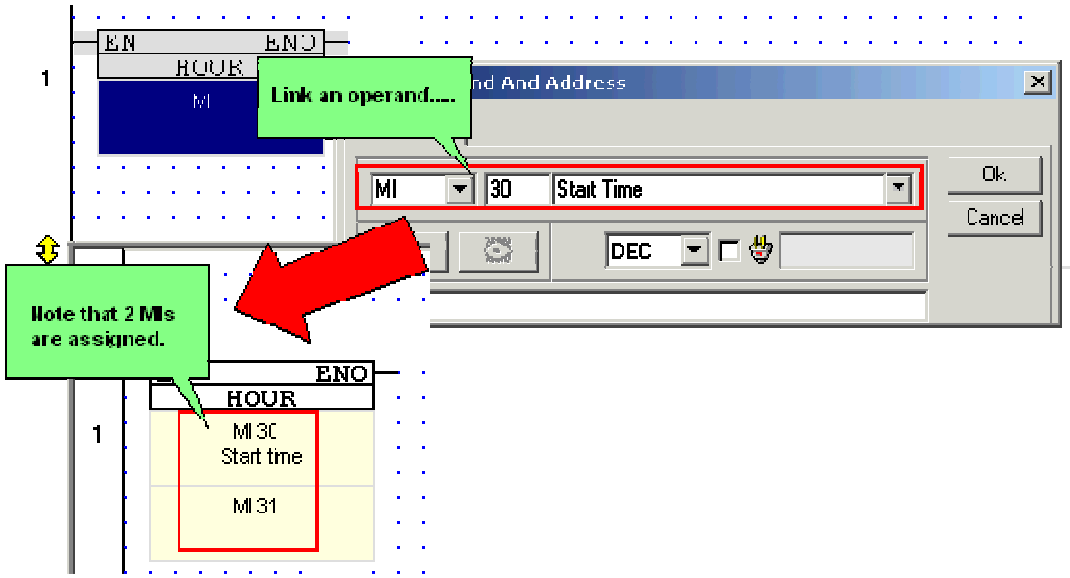
Building the Ladder

1. Place an Indirect Time Function in the net.

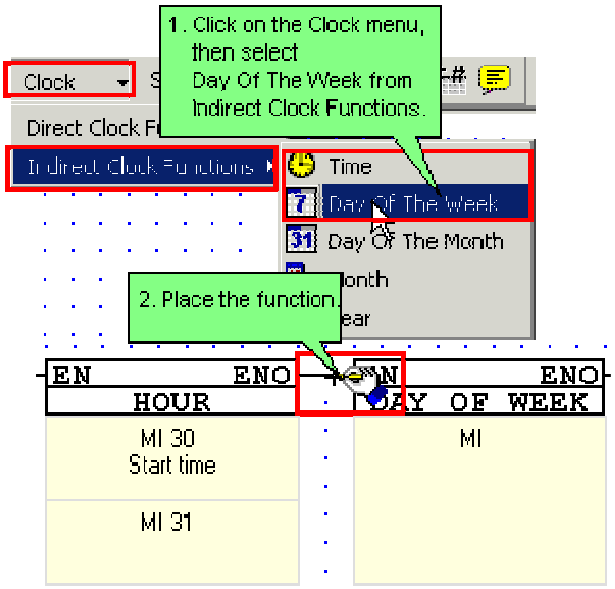


2. Link an operand. The Time function requires two consecutive MIs; the second is automatically assigned by the program. These 2 MIs define a time range. The first MI sets the Start Time for the function, the second

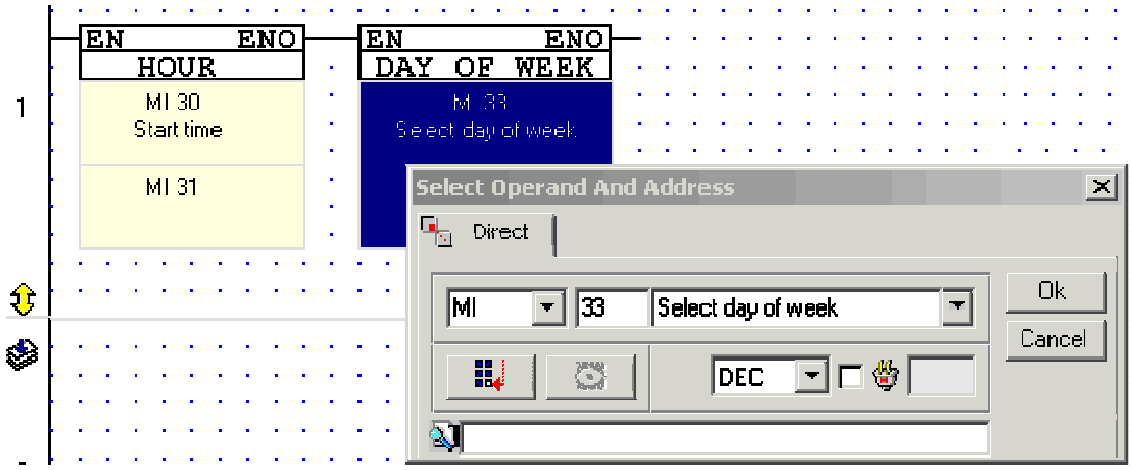
MI marks the End Time. When the RTC is within this range, power flows through the function block.



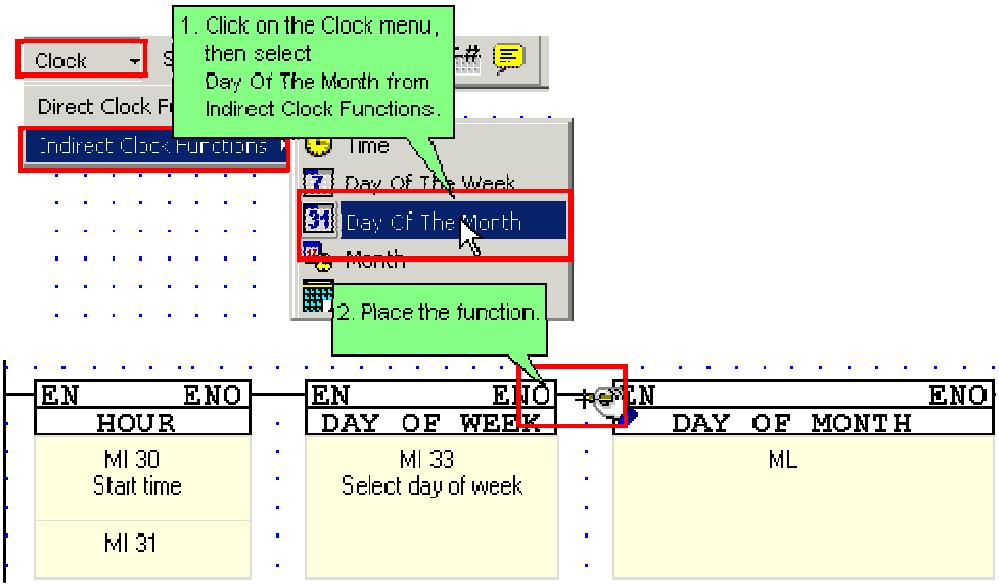
3. Place a Day Of The Week function so that it touches the first function, enabling power flow. This function uses a 16-bit register to contain a 7-bit bitmap representing the days of the week.



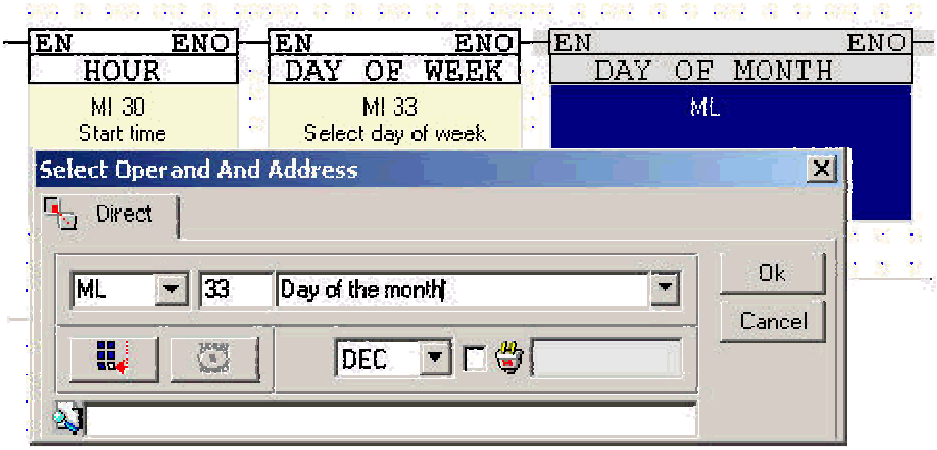
4. Link an operand.



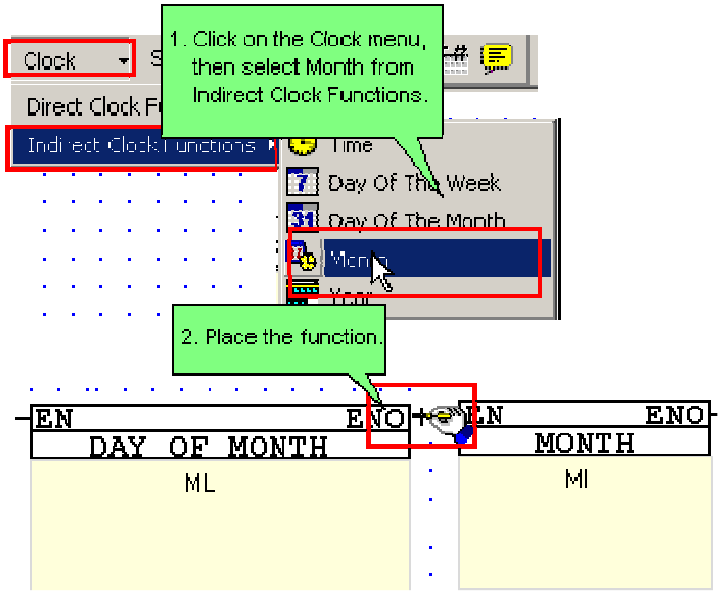
5. Place a Day of The Month function so that it touches the last function.



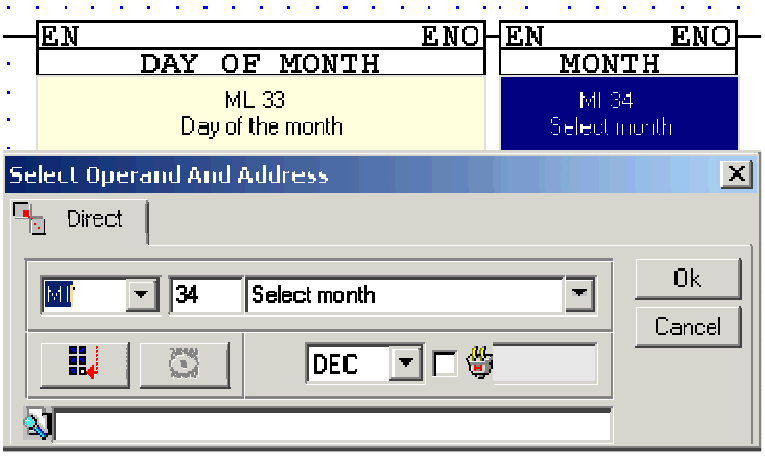
6. Link an operand. This function uses an ML or SL to contain a 32-bit bitmap.



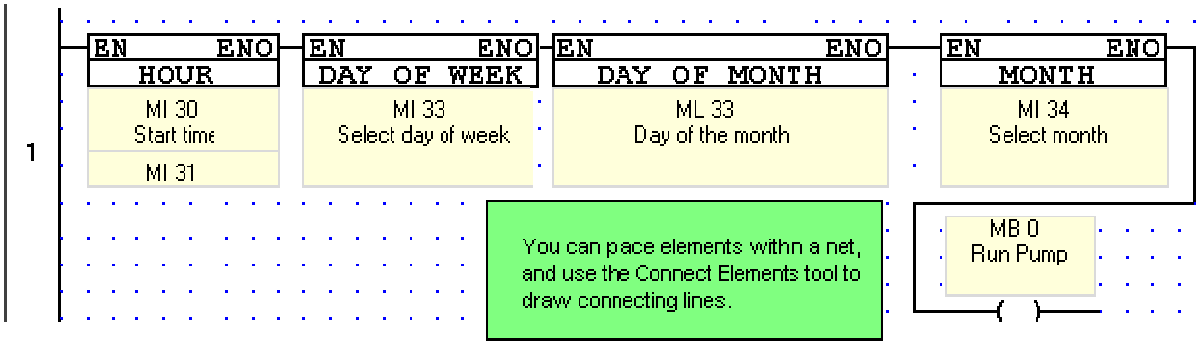
7. Place a Month function so that it touches the last function.



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.



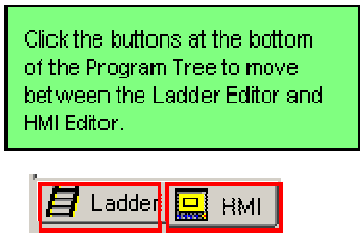
You build the net using Indirect Time functions.

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.

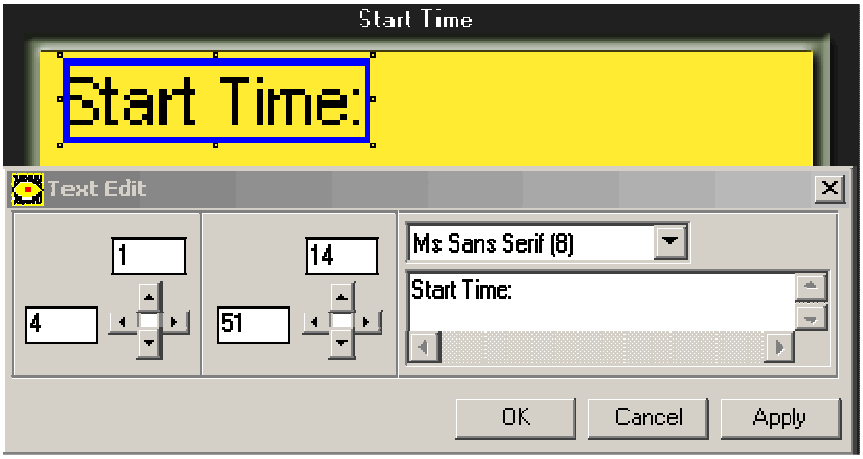
Start & End Time Variables

1. Open the HMI Display editor.



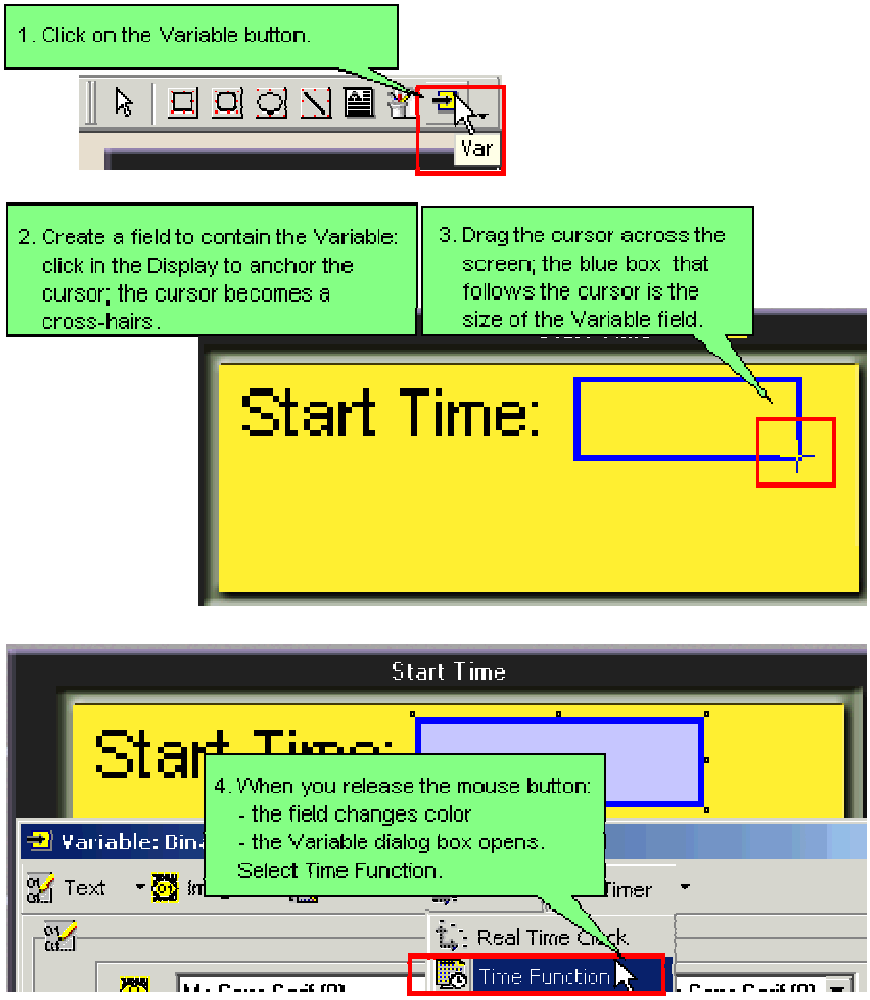
2. Create and name a Display: **Start and End Time**.

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

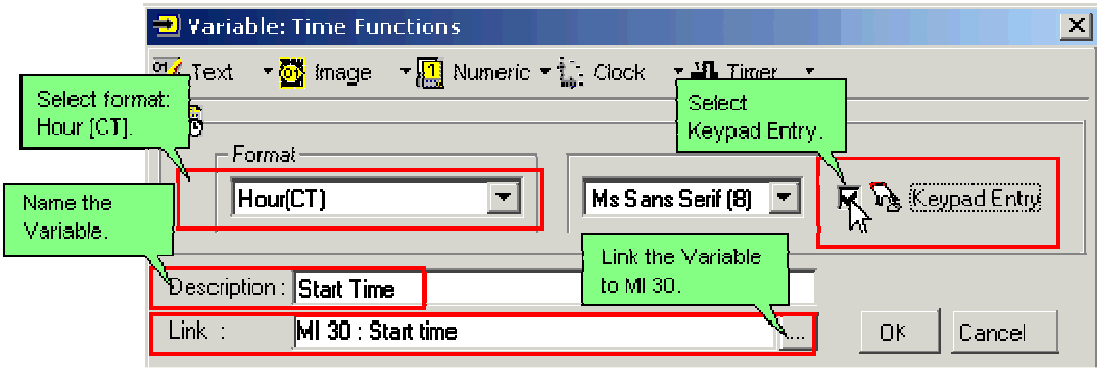


4. Draw another text box, and enter the text: **End Time.**

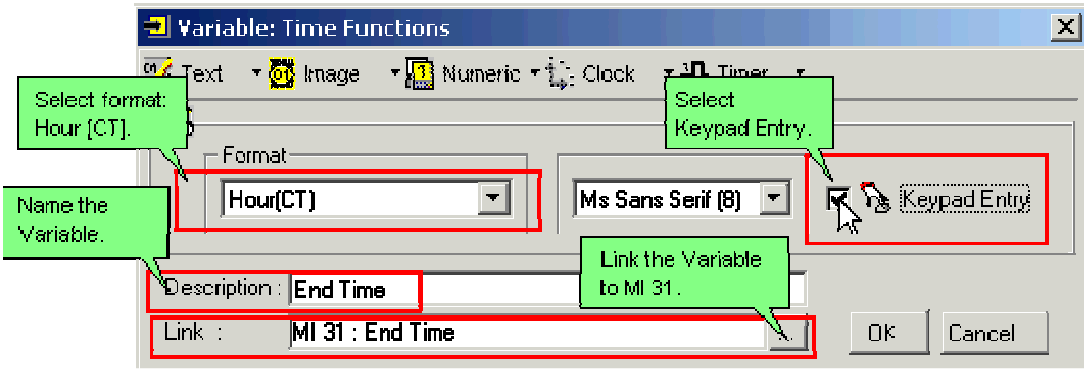
5. Create a field to hold the first Time Function Variable, **Start Time**.



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



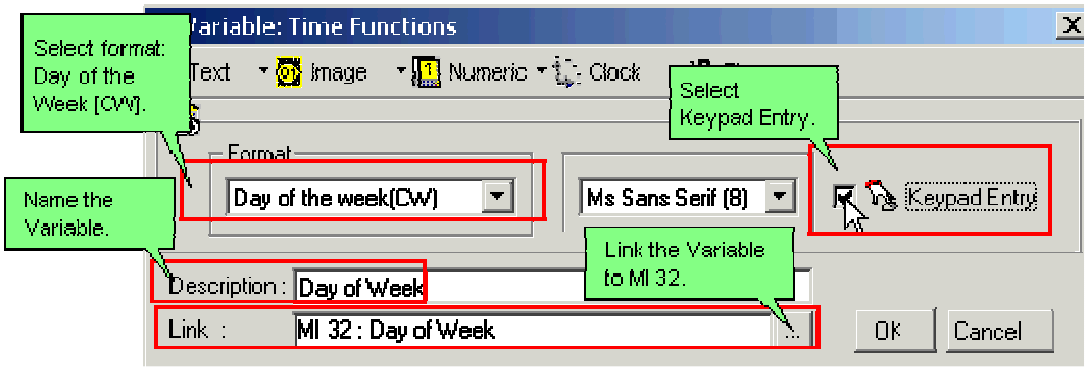
6. Create a field and define the End Time Variable, linking it to MI 31.



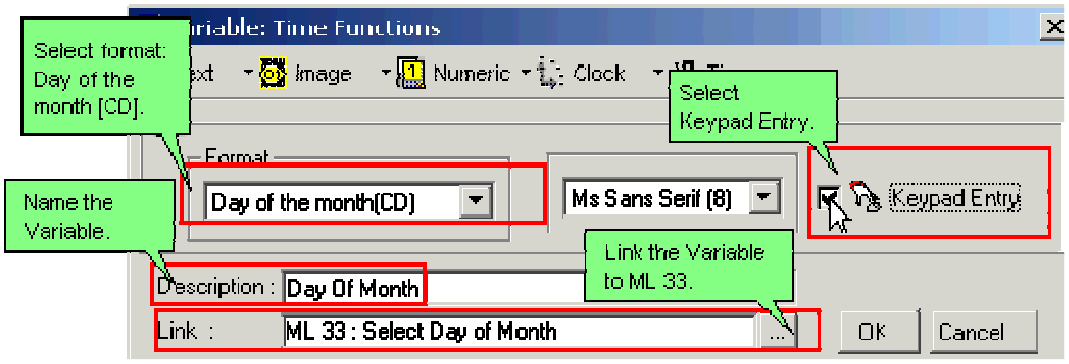
This Display is complete.

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.

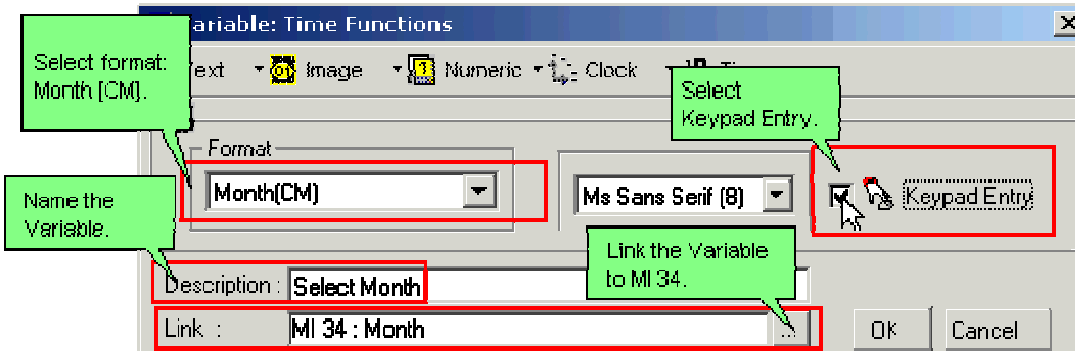


This Display is complete.

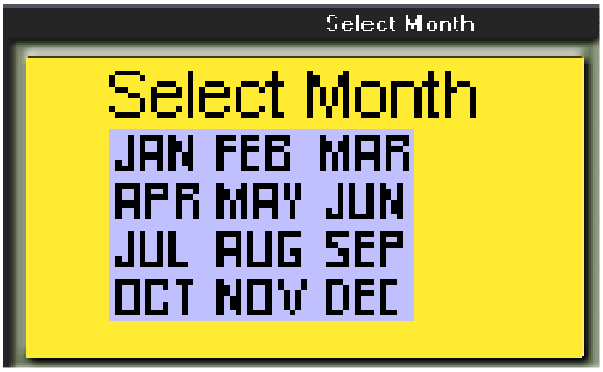
Month Variable

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.

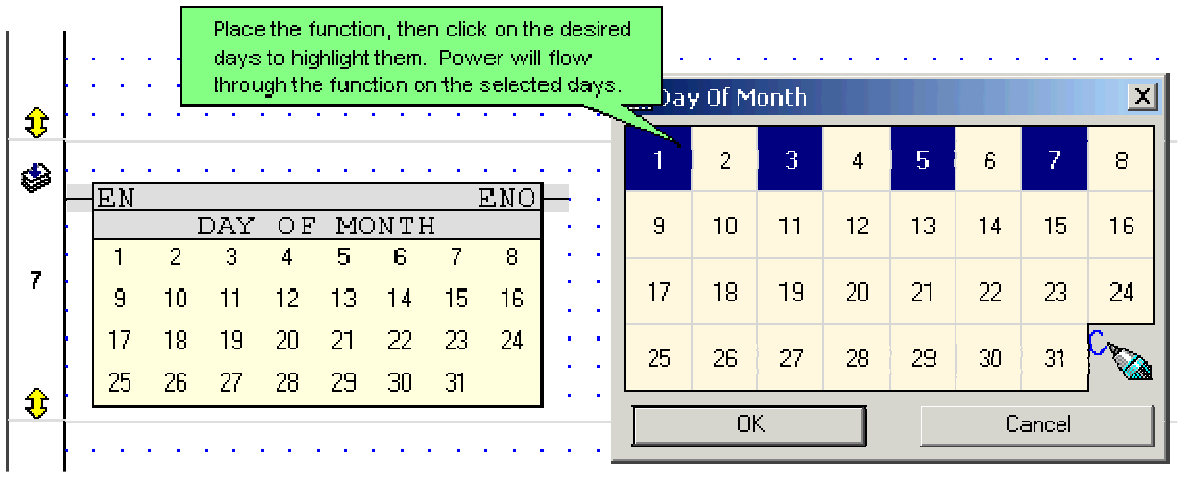
Day of the Month-Direct and Indirect

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.

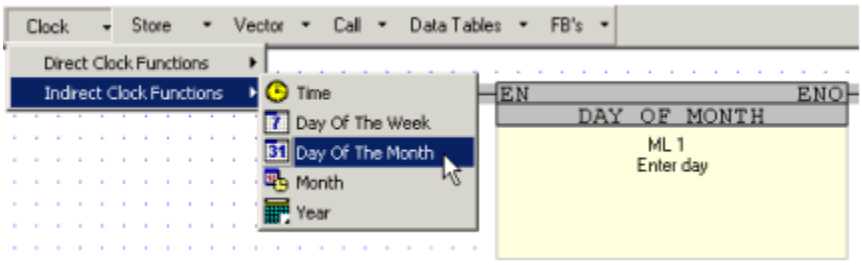
The Indirect Month Time function is linked to a 32-bit ML or SL that provides a bitmap for the functions. The ML value shown below contains the decimal value 271077376 (hexadecimal 10285000). According to this value:

- 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).

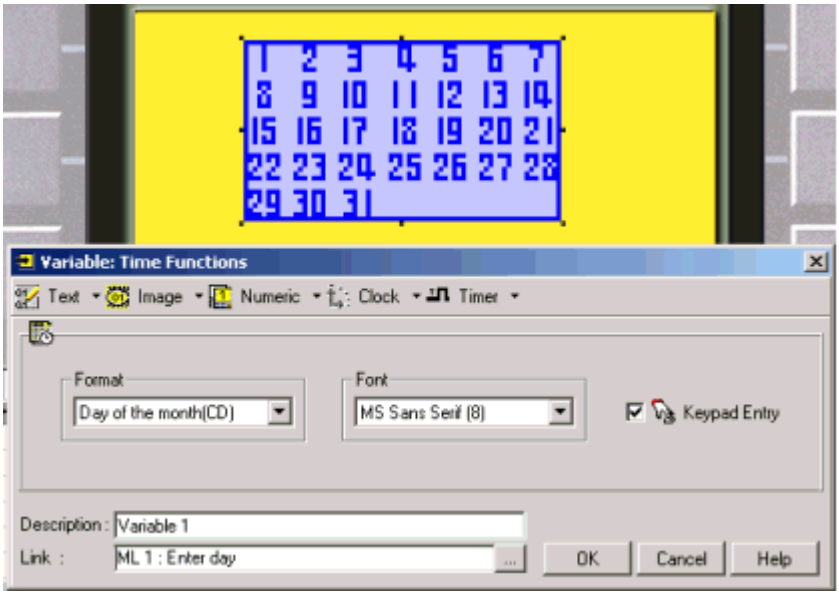
ML	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Status	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

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.



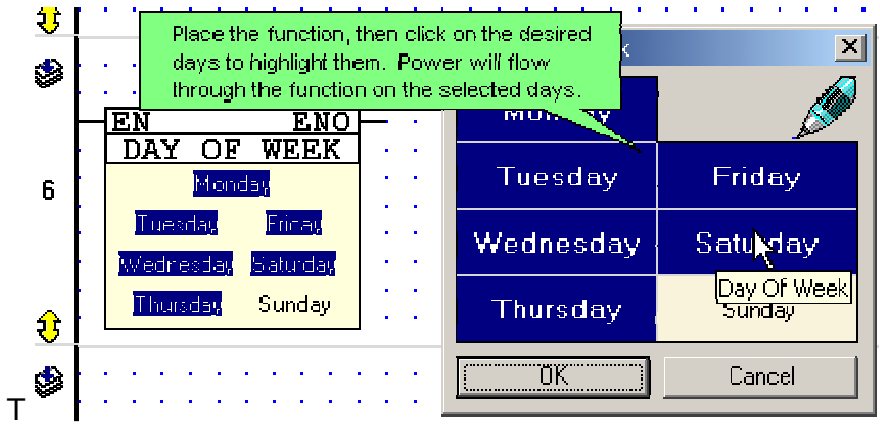
To select the days using the controller's keyboard, the operator uses:

- **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.

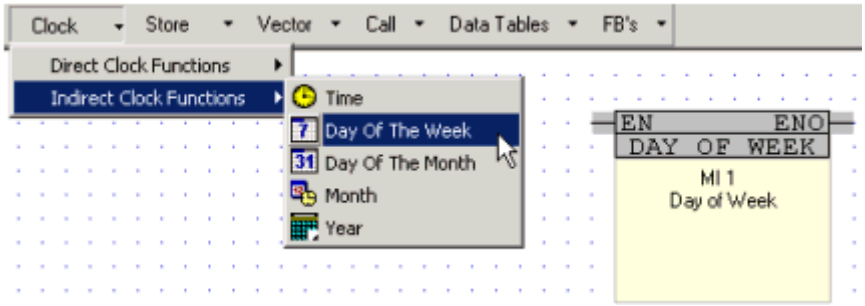
The Indirect Day of Week function is linked to a 16-bit register that provides a 7-bit bitmap in the linked MI. The MI value shown below contains the decimal value 42 (hexadecimal 2A). According to this value:

- 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).

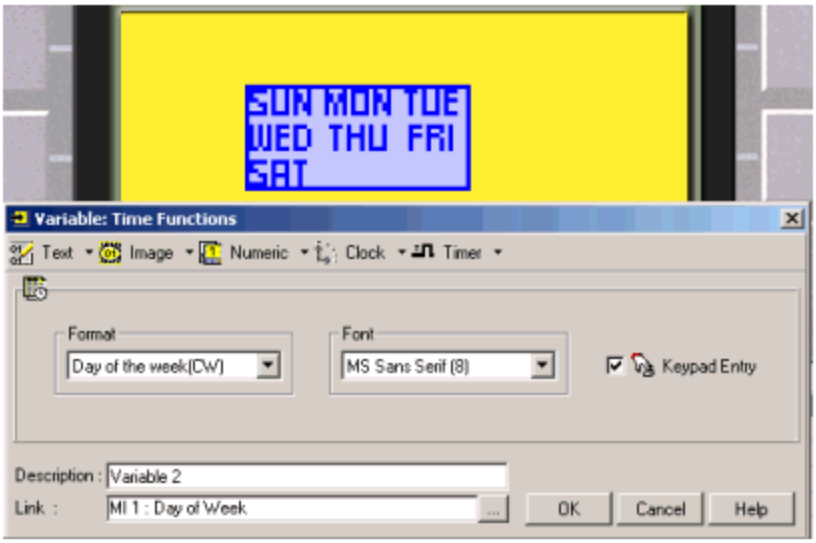
Day	Sat	Fri	Thurs	Wed	Tues	Mon	Sun
MI	6	5	4	3	2	1	0
Bit Status	0	1	0	1	0	1	0

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:

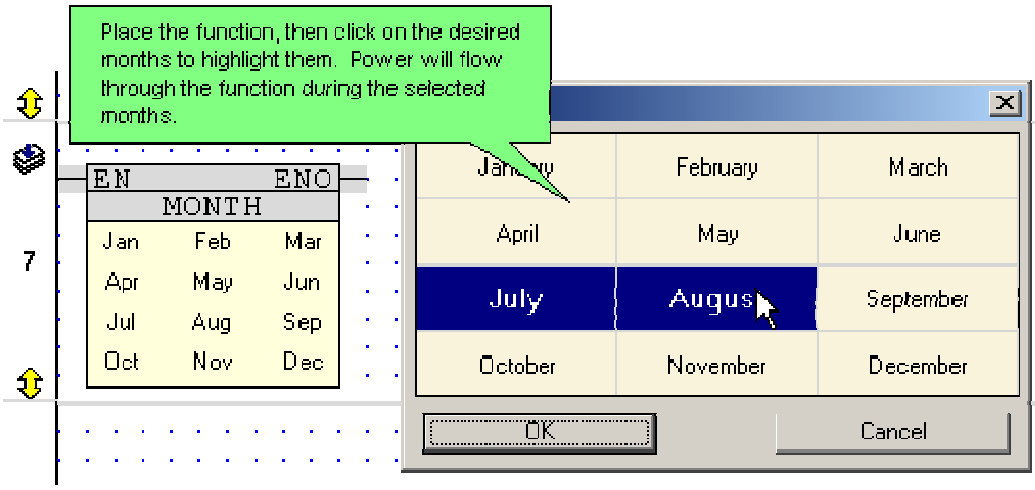
- **Up** and **Down** scroll arrow keys to scroll through the days of the week,
- The **<Enter>**key to select the desired days of the week.

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.

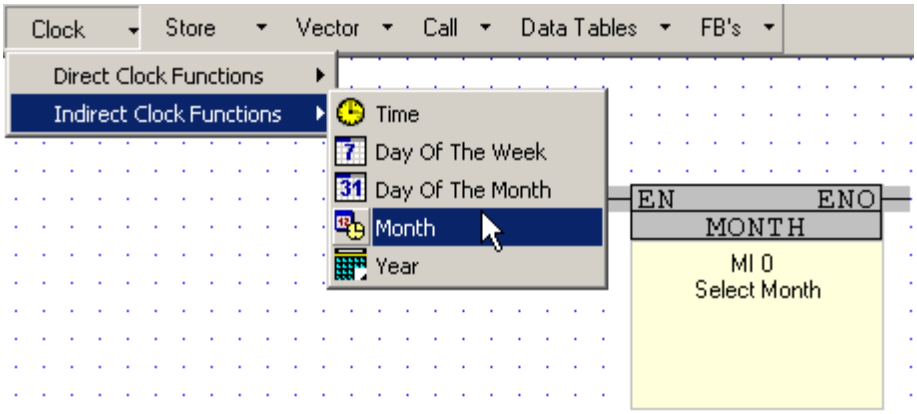
The Indirect Day of Week function is linked to a 16-bit register that provides a 7-bit bitmap in the linked MI. The MI value shown below contains the decimal value 42 (hexadecimal 2A). According to this value:

- 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).

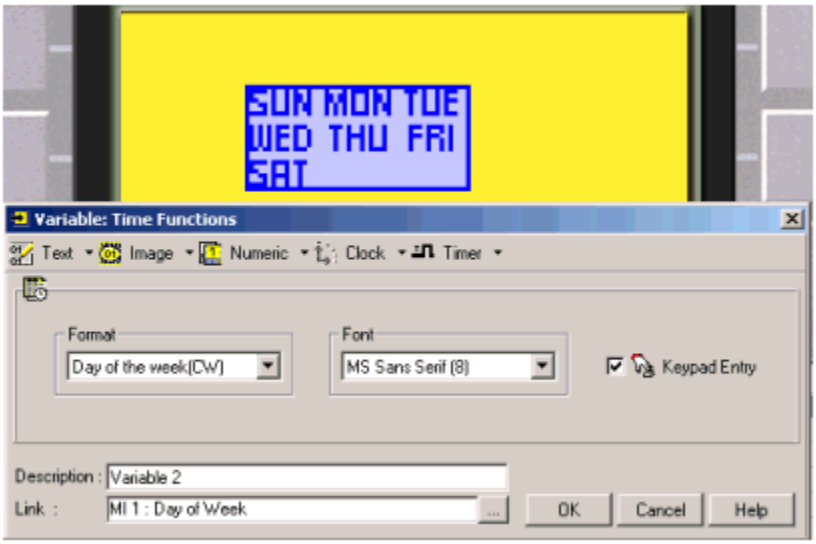
Day	Sat	Fri	Thurs	Wed	Tues	Mon	Sun
MI	6	5	4	3	2	1	0
Bit Status	0	1	0	1	0	1	0

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).

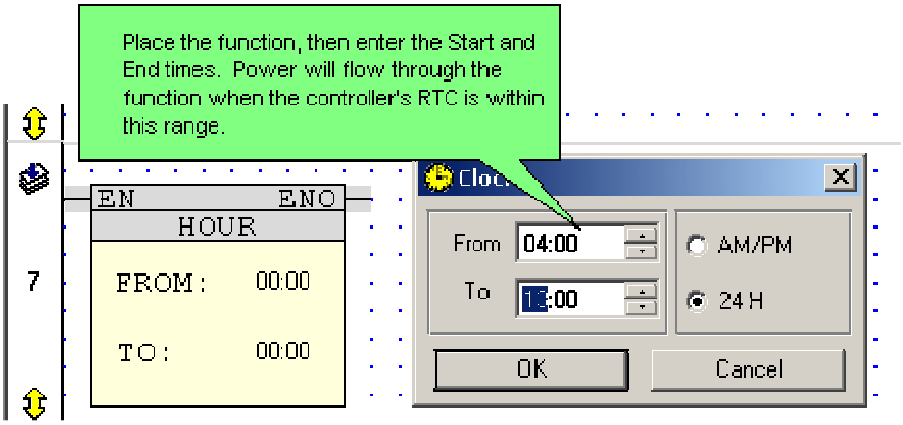
Month	Dec	Nov	Oct	Sep	Aug	Jul	Jun	May	Apr	Mar	Feb	Jan
MI	11	10	9	8	7	6	5	4	3	2	1	0
Bit Status	1	1	1	0	0	0	0	0	0	1	1	1

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.

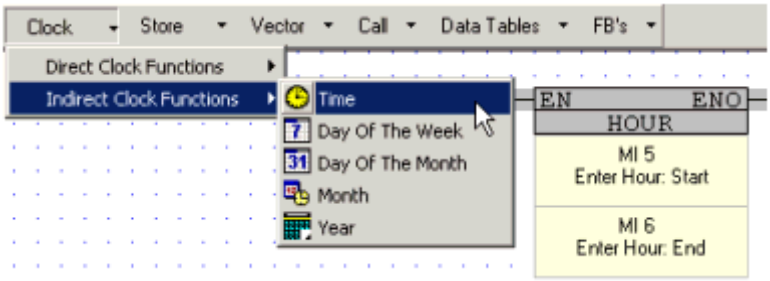
The Indirect Time function is linked to two consecutive registers.
The values are read as hexadecimal (BCD). According to the figure shown below:

- 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).

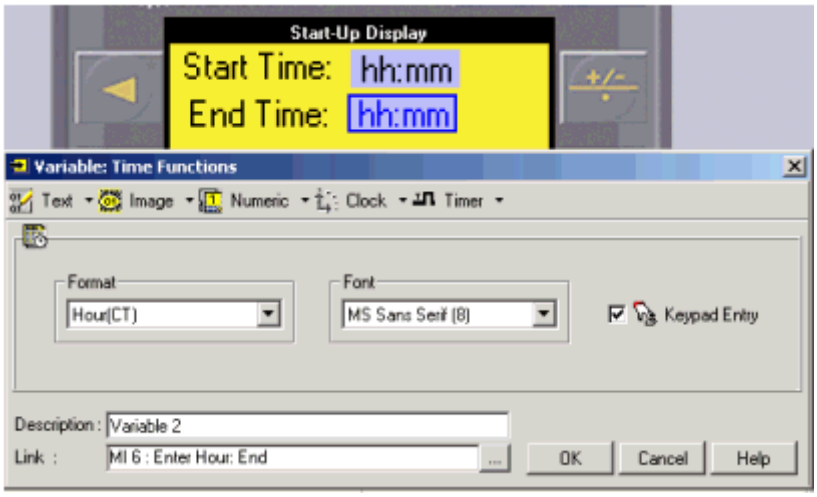
	MI 5: Start Time				MI 6: End Time			
	H	H	M	M	H	H	M	M
Hex	1	9	3	0	2	3	5	9
BCD	0001	1001	0011	0000	0010	0011	0101	1001

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.

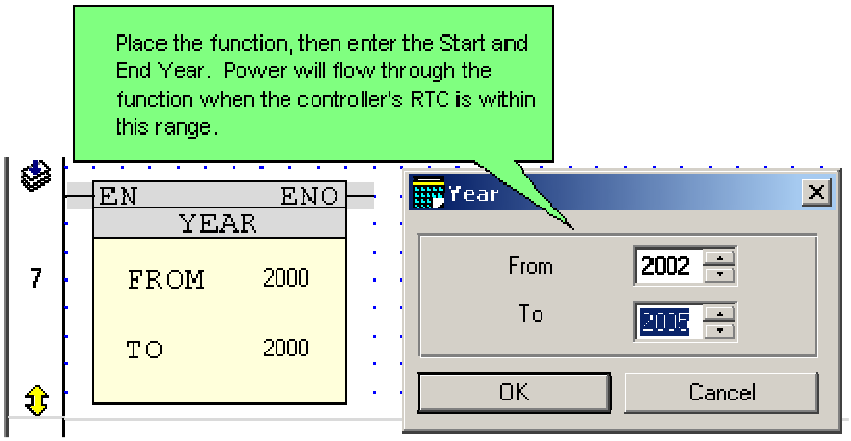
Clock: Year, Direct/Indirect

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.

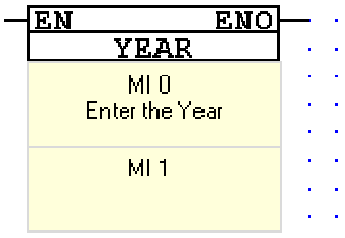
If the RTC is within these two times: power will flow through the function.

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



Immediate Elements

Immediate elements are located on the Utils> Immediate menu. They are supported by Snap-in I/O modules.

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:

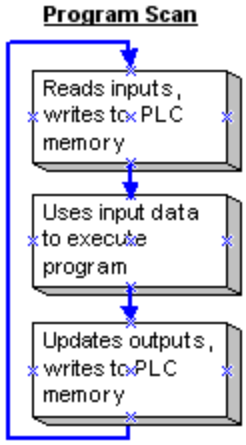
- Write values to inputs, and use the new input value to execute the rest of the PLC program.
- Turn outputs ON, as for example in an emergency routine.

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

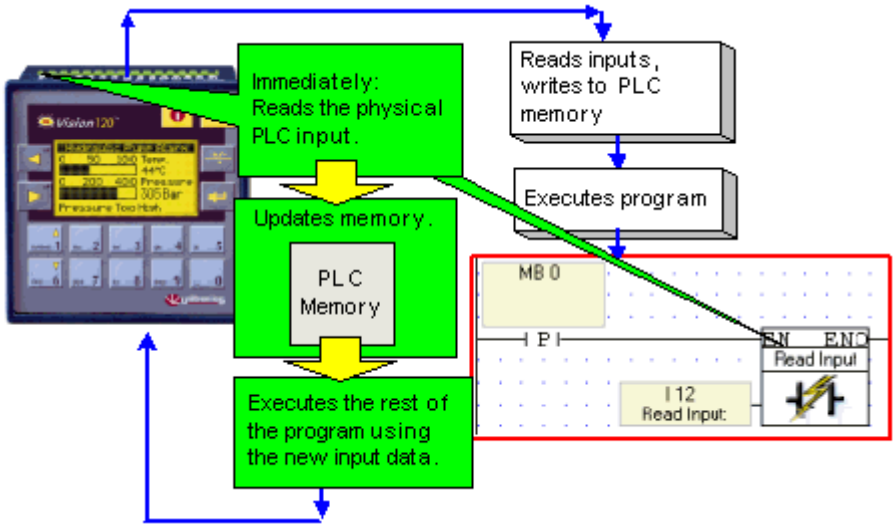
Immediate: Read Physical Input

Read Physical Input is located on the More> Immediate menu. Use this element 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 scan runs like this:



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.

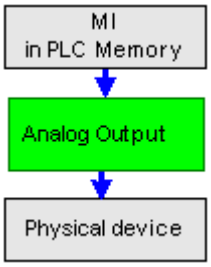
Note • Within a net, Read Physical Input should stand alone except for its

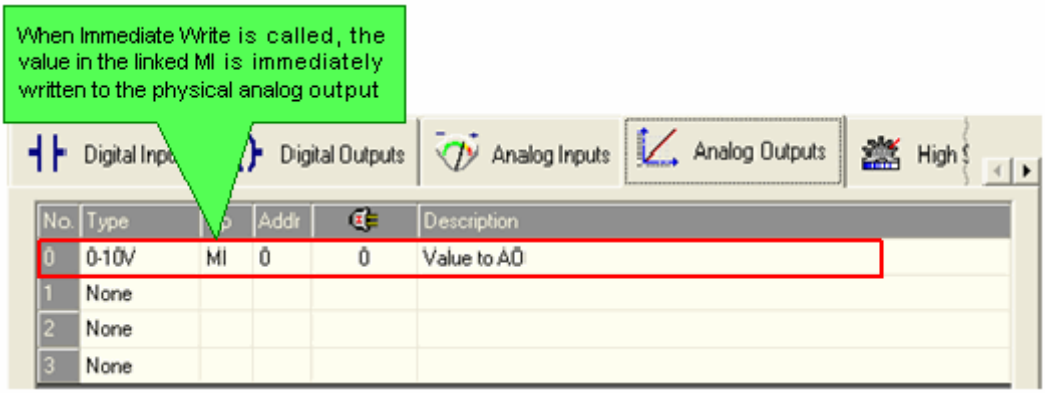
activating condition.

Immediate: Write to Physical Analog Output

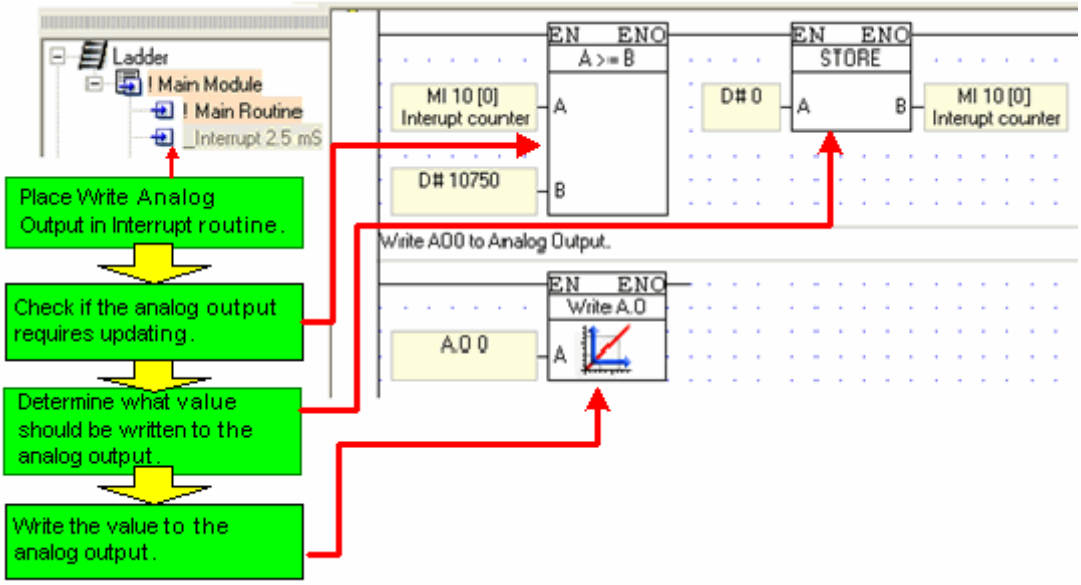
Write to Physical Analog Output is located on the More> Immediate menu. This element can be used to immediately write a value into a physical, hardwired output--without regard to the program scan.

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





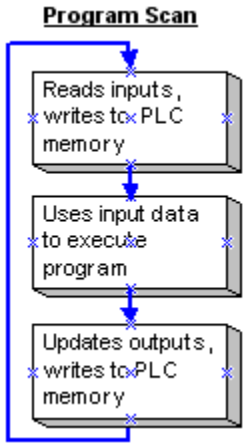
Note • Within a net, Write to Physical Analog Output should stand alone .



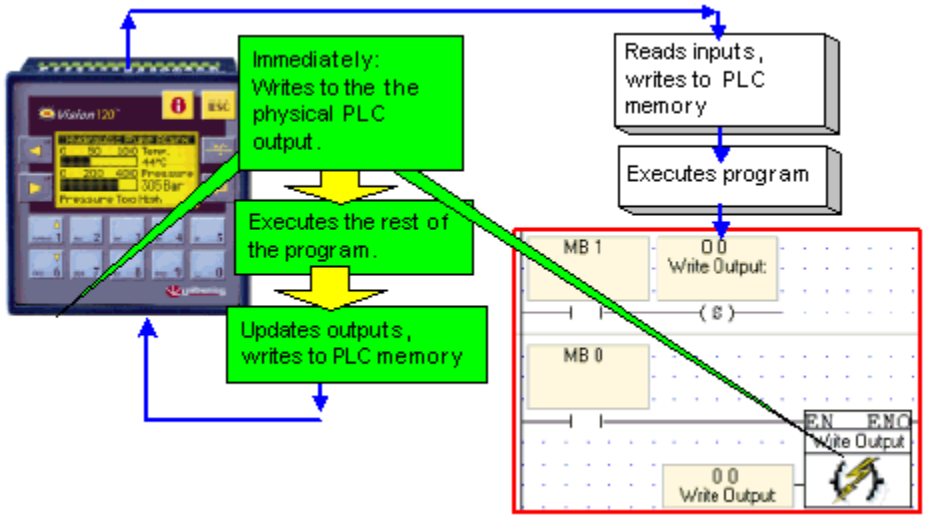
Immediate: Write to Output

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.

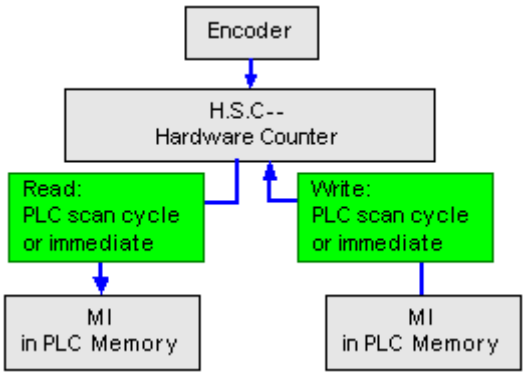


To use Write to Output, place it in a net after an activating condition and select the desired output.

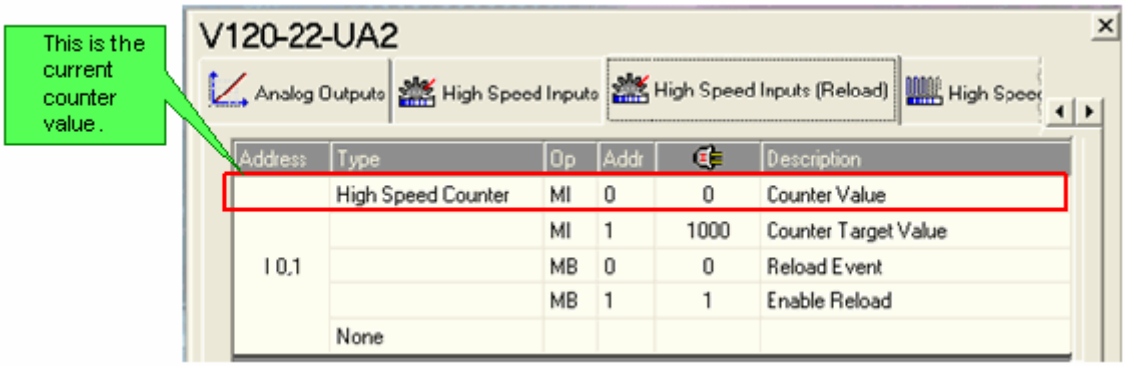
Note •	This function is not supported for outputs located on I/O Expansion modules.
•	Within a net, Write to Output should stand alone except for its activating condition.
•	If, after Write to Output has been executed, the same output is updated as the rest of the program runs, the last update is the one written to the PLC memory at the end of the program scan.

Immediate: Update High-speed Input

Update High-Speed Input is located on the More> Immediate menu. Use this element 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.

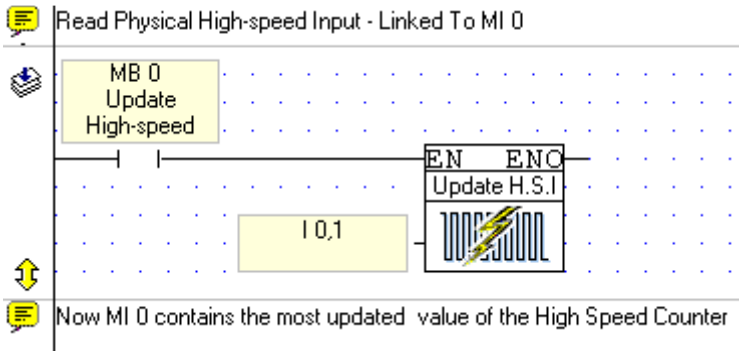


When the program encounters Update High-Speed Input, the program immediately compares the actual, current input value against the value in the MI linked to the input.

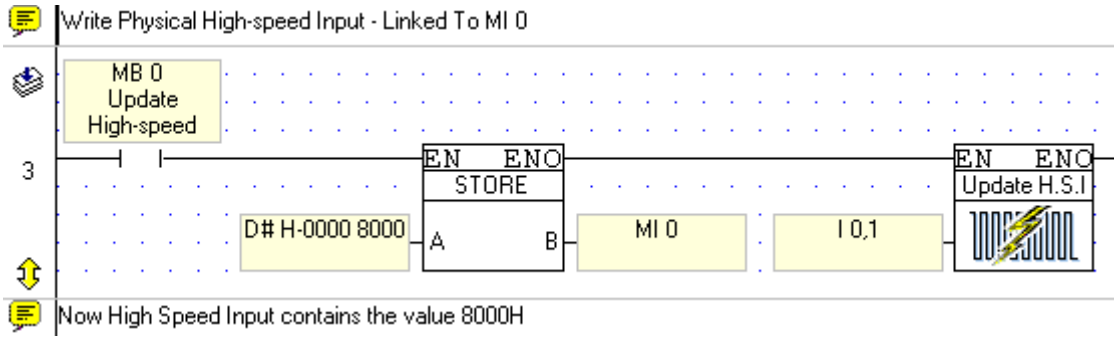


If the values are not equal, the MI is updated with the current input value; the rest of the program executes according to the new input data.

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: HSC Freq. Measurement Utilities

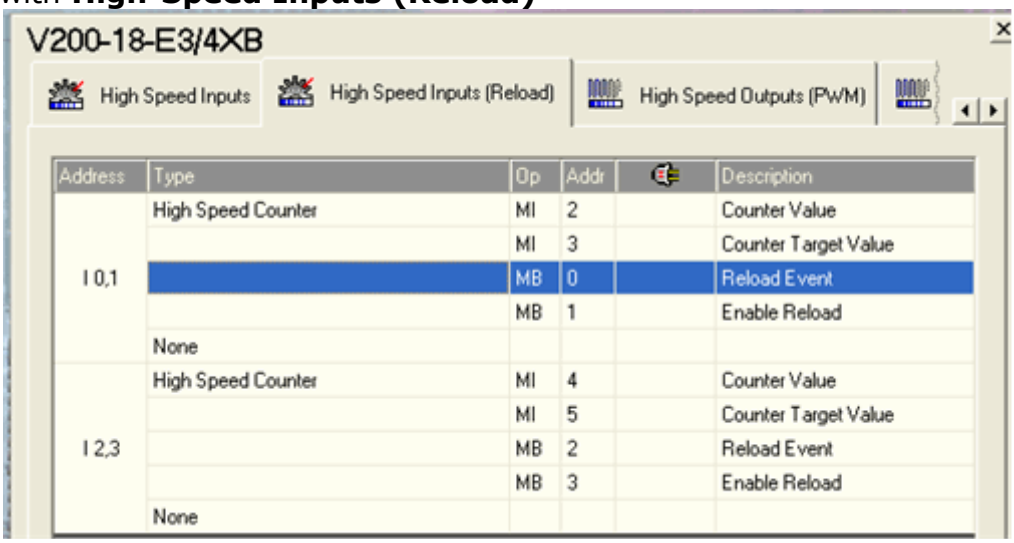
HSC Frequency Measurement Utilities are three functions located on the More> Immediate menu. Together, you can use them to monitor a high-speed counter as a frequency value, and reset it if the changing frequency deviates from a set value.

The frequency can be measured from 0.25 Hz up to the maximum frequency that the PLC can read.

Note •

These utilities are compatible:

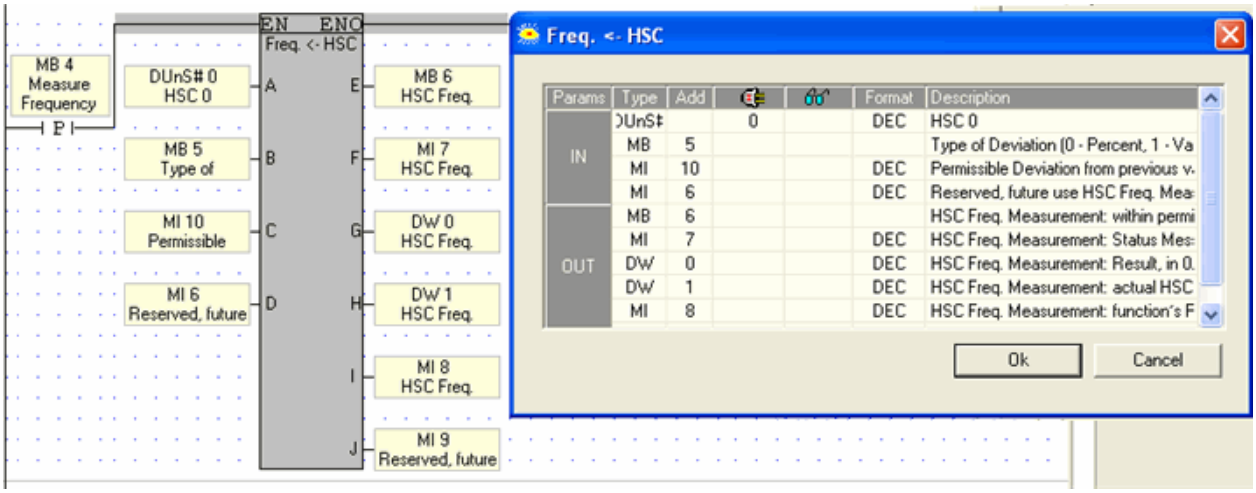
- with Enhanced Vision controllers only.
- with **High-Speed Inputs (Reload)**



Frequency Measurement based on HSC

This element registers the frequency of the high-speed counter at each scan, and checks it against the frequency of the previous scan. If the change (delta) is greater than desired, the function sets an MB.

To use this function, place it in a net after an activating condition and select the desired high-speed counter. Once the function is activated, it continues to run until it is stopped by the application.

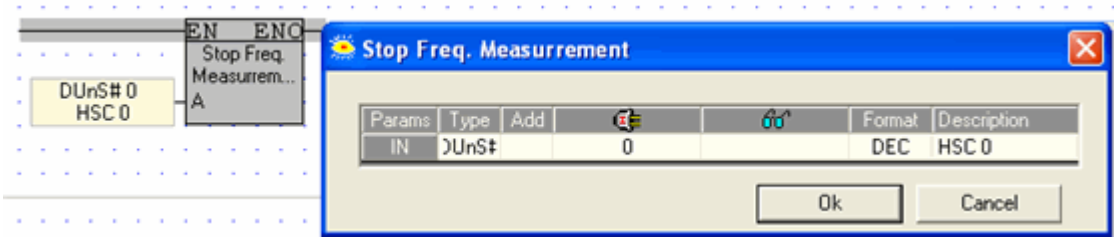


Input	Type	Description	Comments
A	Constant	High-speed counter	This is the actual counter the function monitors
B	MB, XB, SB	Type of Deviation (0 = Percent, 1 = # of pulses)	You can use either a percentage, or in .01Hz resolution by the 'C' input). Percentage: the range is 1 to 1000, where 1000 is 10 times the previous measured value # of Pulses: in 0.01Hz. (100 is equal to 1Hz)
C	MI, XI	Permissible Deviation from previous value (default 20)	This is the legal 'Delta'; the difference between the current value and the previous value that was read.
D	MI, XI	Reserved, future use HSC Freq. Measurement	
Output	**Type**	**Description**	**Comments**
E	MB, XB	HSC Freq. Measurement: within permissible deviation	Turns OFF when the between the current value and the previous value exceeds the permissible deviation.
F	MI, Xi	HSC Freq. Measurement: Status Messages	• 0 - Valid data • 1 - No signal for time at least twice the permissible deviation (value in operand C). • 2 – Signal came after more than 2 sec (Less than 0.5Hz) • 4 – Higher frequency measured than the PLC can run • 8 - The change in frequency is bigger than the value in operand C • 10 - Reserve for system error
G	DW, XDW	HSC Freq. Measurement: Result, in 0.01 Hz	The frequency value

H	DW, XDW	HSC Freq. Measurement: actual HSC value	This is the number of pulses from the counter
I	MI, XI	HSC Freq. Measurement: function's Reload Value	This is the actual high-speed counter's Reload value
J	MI, XI	Reserved, future use HSC Freq. Measurement	

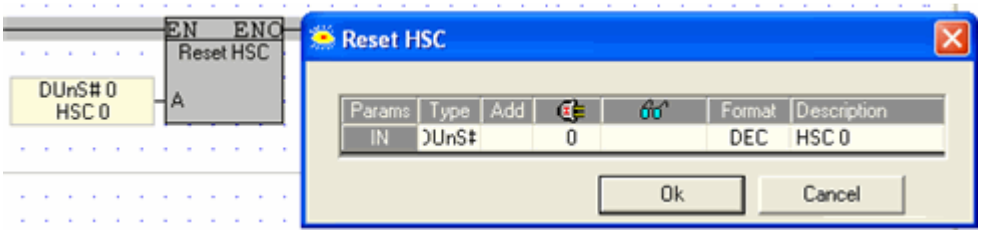
Stop Frequency Measurement

Use this function to stop the **Frequency Measurement based on HSC** function.



Reset HSC

Use this function to initialize the high-speed counter value.



On-Line Test Mode (Debug) functions

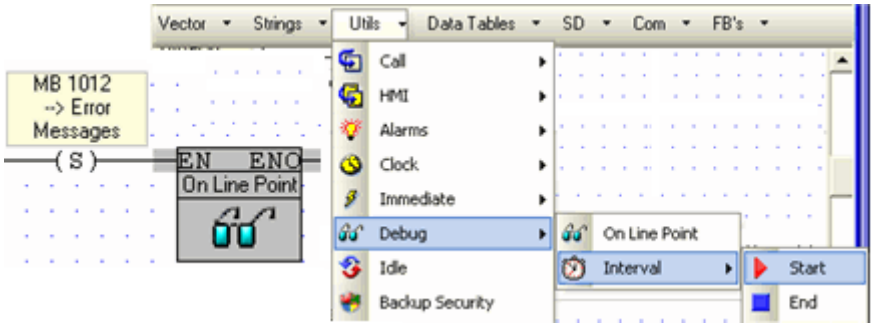
You can use the On-line Point and Interval utilities on the Utils> Debug menus to test your project. For more information, refer to the manual VisiLogic – Utilities.

To test a project, first establish PC-PLC communications by connecting the controller to the PC with the MJ10-22-CS25 programming (communication) cable. Note that the V1040 supports download via a USB (Type A to mini-B) cable. COM port 1 function is suspended when the USB port is physically connected to a PC.

Note • You can also use Remote Access to establish a communication line via modem or network.

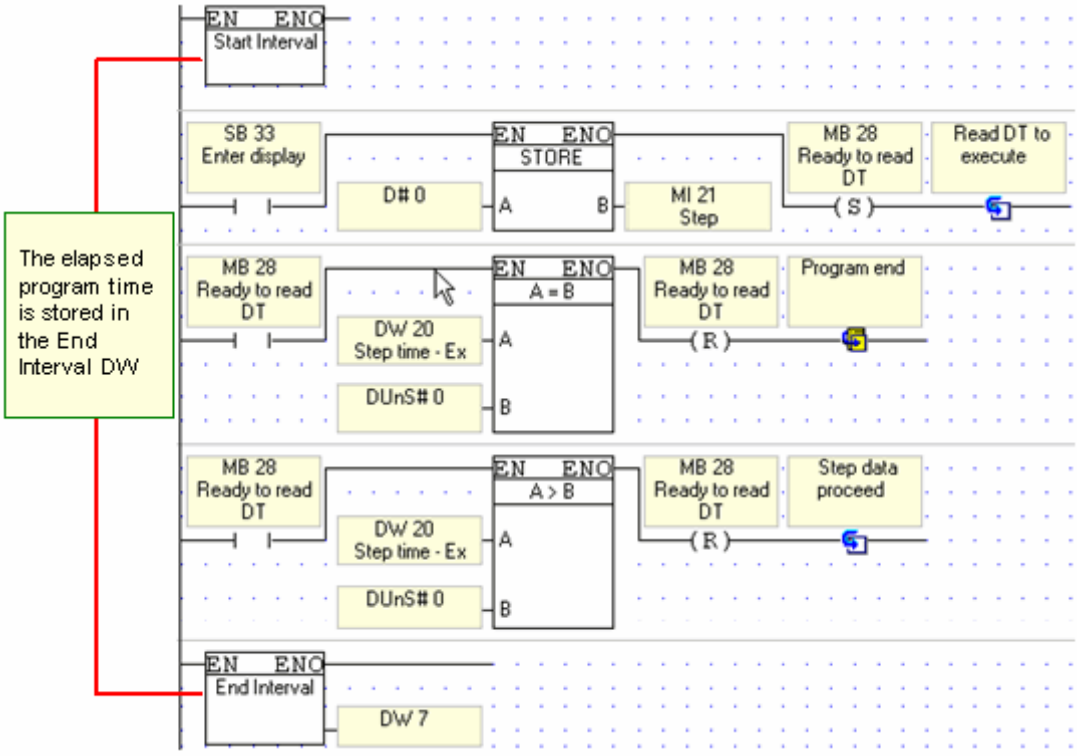
Once you have established communications, download the project and click the On-Line Test button. The Online Test toolbar opens, enabling you to:

- Switch between Run and Stop modes.
- Use Single Scan to run a single cycle of the ladder program for debugging purposes.
  You can stop the scan cycle at any point by placing OnLine Test Points, located on the More menu, in the Ladder.

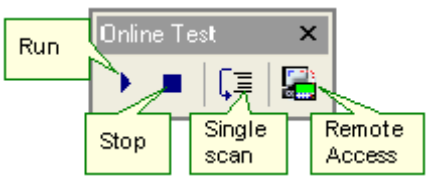


When the scan reaches an OnLine test point that is active (receives RLO), Online Test freezes, enabling you to check element status and values, including Timer values, at that point during Ladder execution. Note that if more than one OnLine test point is activated, SB 35 turns ON.

- Measure the time interval between 2 points in the Ladder application, by placing Start and End Interval elements, located on the More menu, anywhere in the application. The time interval, units of 10 micro-seconds, is stored in the DW linked to the End Interval element. Note that Interval elements should not be placed in Interrupt routines.

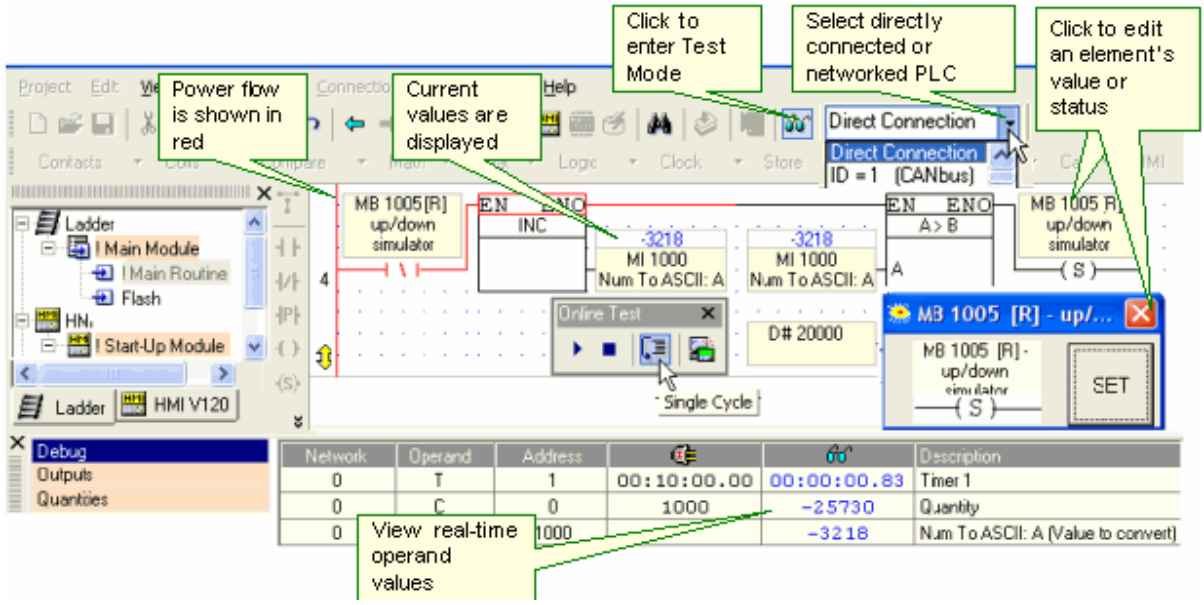


- Open Remote Access to debug remote controllers via network or modem connections.

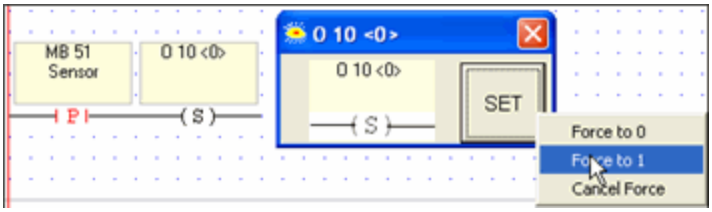


In Online Test mode, you can view the power flow, and view and force operand values and element status. You can also select a controller that is directly

connected to the PC, or a PLC's network ID # if the PC is linked to a CANbus or RS485 network.



- Force I/0, by right-clicking the operand and setting the desired state

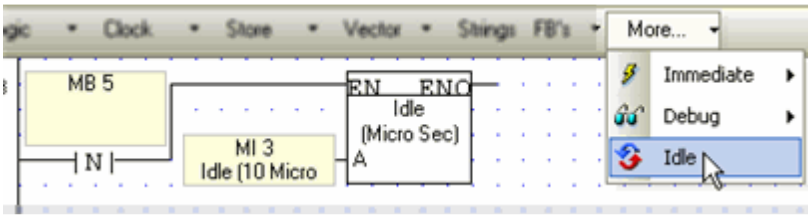


Note • The controller can send and receive SMS messages when the controller is in Test mode.

Idle

Place Idle anywhere in a Ladder program to completely stop the program scan for a specific number of micro-seconds.

All action is suspended, including I/O updates. Idle is located on the More menu,.



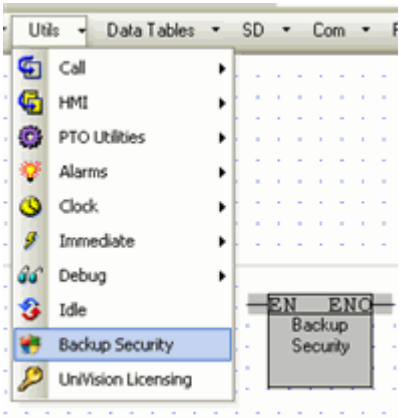
BackUp Security (Enhanced Vision only)

This Ladder function is located on the Utils menu. Use it to store the following values in the controller's memory:

- SB314 "Pcom block" operand
- SI 253 Info Password value

- SI 50 Info press time

Note that if such a backup exists, SB 303 will be ON.

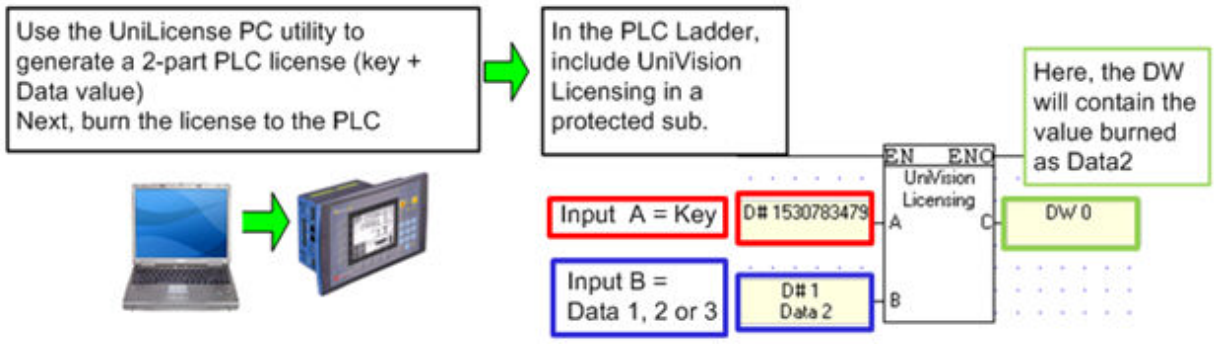


UniVision Licensing

You can create a PLC license number and burn it into a secured, hidden sector in the PLC.

You can then use this license in your Ladder to control how your program functions.

To license the PLC, use the UniVision Licensing function on the Utils menu may be used in conjunction with the UniVision Licensing stand-alone utility, which may be freely downloaded from http://unitronics.com/Content.aspx?page=Downloads



The result of the License operation can then be used to activate or deactivate different sections of your application.

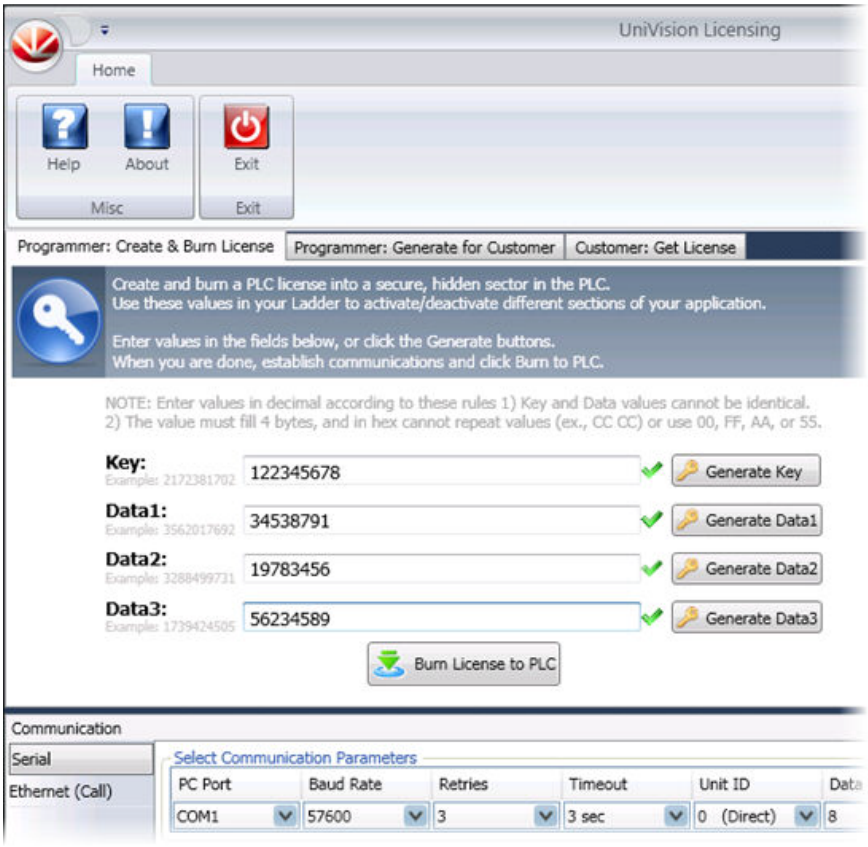
The UniVision Licensing utility enables you to create two kinds of licenses:

- One that licenses the program, but that is not bound to a particular PLC
- One that licenses the program, and incorporates a specific PLC ID number. This type will only license the specific PLC it is generated for. Note that you can use this type to license a remote end user's PLC.

Generate and burn a license, without PLC ID number

1. In the UniVision Licensing utility, click on the first tab, Programmer: Create & Burn License.

2. Enter the key number.
   This is the first part of the license (input A).
3. Enter the values for Data1, Data2, and Data3.
   This is the second part of the license (input B).
4. You can use the Generate buttons to create these values.
   The fact that there are 3 values enables you to create levels of access.
5. Establish a communication connection to the PLC, and press Burn License to PLC.



Generate and burn a license comprising a PLC Unique ID number

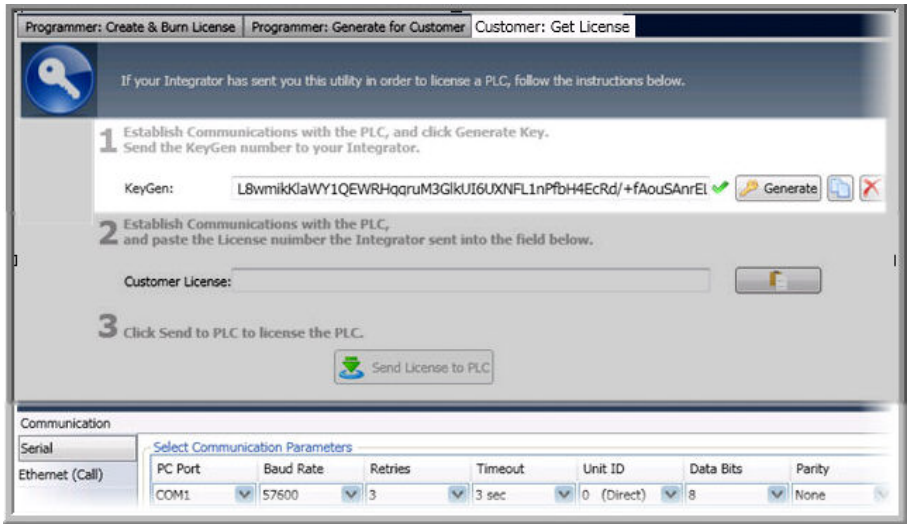
In this case, you must establish a communication link with the specific PLC which is to be licensed, and generate a KeyGen number. The KeyGen number encrypts the PLC's Unique ID Number. You use this number to generate the license, which is **specific to that PLC**. This license number will not work in any other PLC.

Providing a license to a remote customer
Note that you can send the UniVision Licensing utility to a customer. The customer can email you the KeyGen number; you use this to generate the license number and send it back to the customer, who can then license the PLC.

First, get the KeyGen number:

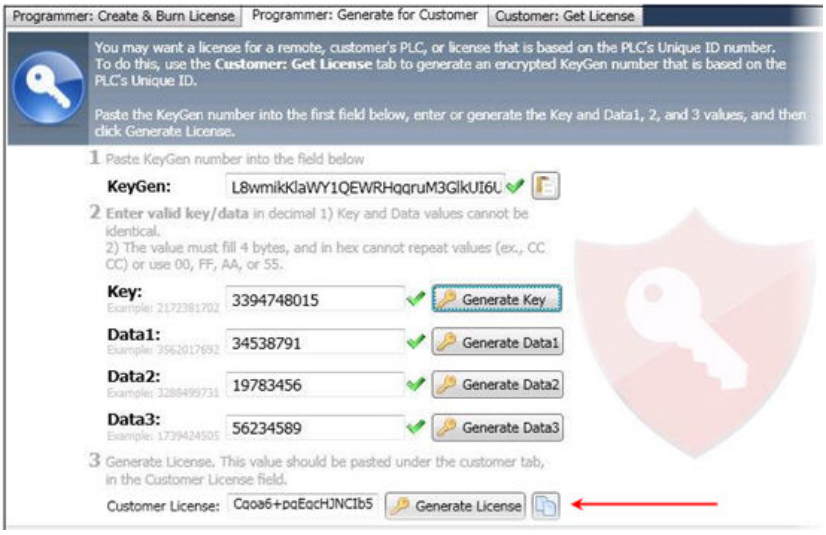
1. In the UniVision Licensing utility, click the Customer:Get License tab.
2. Establish a communication connection to the PLC.

3. Click Generate, and then the Copy button.



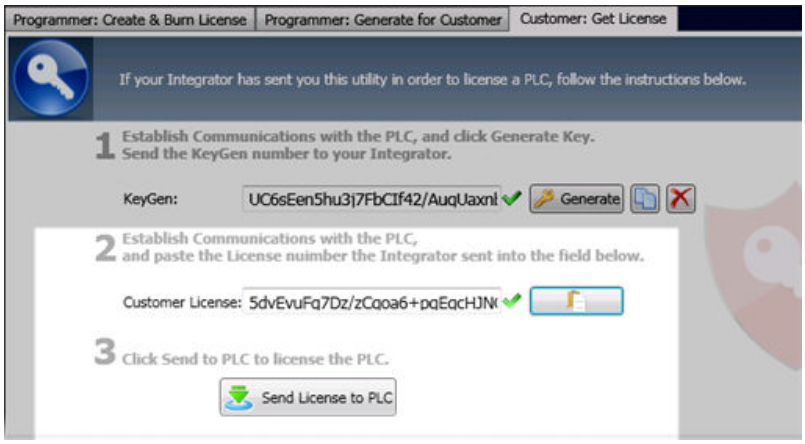
Next, generate the license:

4. Paste the value into the KeyGen field, and fill in the Key and Data values
5. Click Generate License, and then the Copy button..



Now, license the PLC:

6. Paste the number into the Customer License field, and press Send License to PLC to burn it to the PLC.



Data Table Functions

For information regarding Data Tables, refer to the manual VisiLogic – Utilities.

Data Tables, Read/Write

Read enables you to copy values from a Data Table to PLC operands.

Write functions enables you to copy operand values from a PLC to Data Tables.

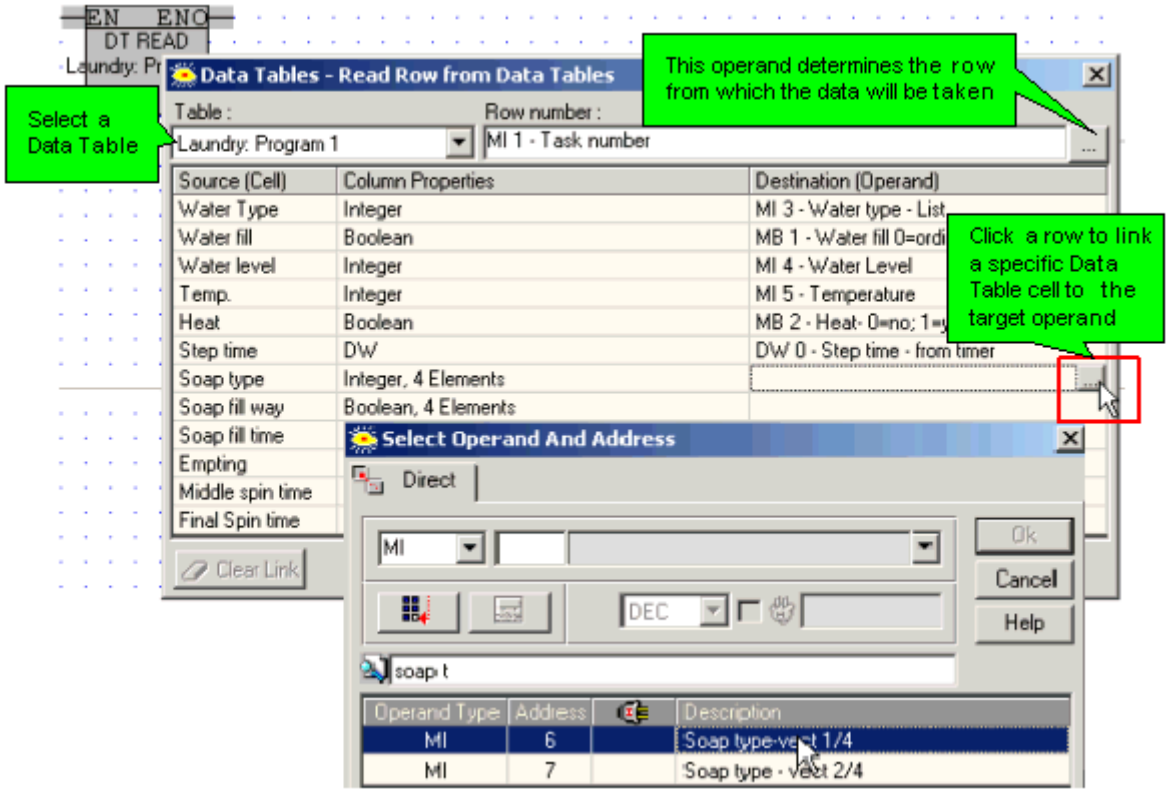
Read/Write functions are located on the Data Tables menu.

Note • The maximum number of operands of **any** type for a Read/Write operation is 128.

Rows

Read Row

Use the Read function to select Data Table rows and read their data into PLC memory operands. Values are read from the Data Table into the operands that are linked to it in the Read function. Note that the number of rows read cannot exceed the number of rows that are in the Data Table.

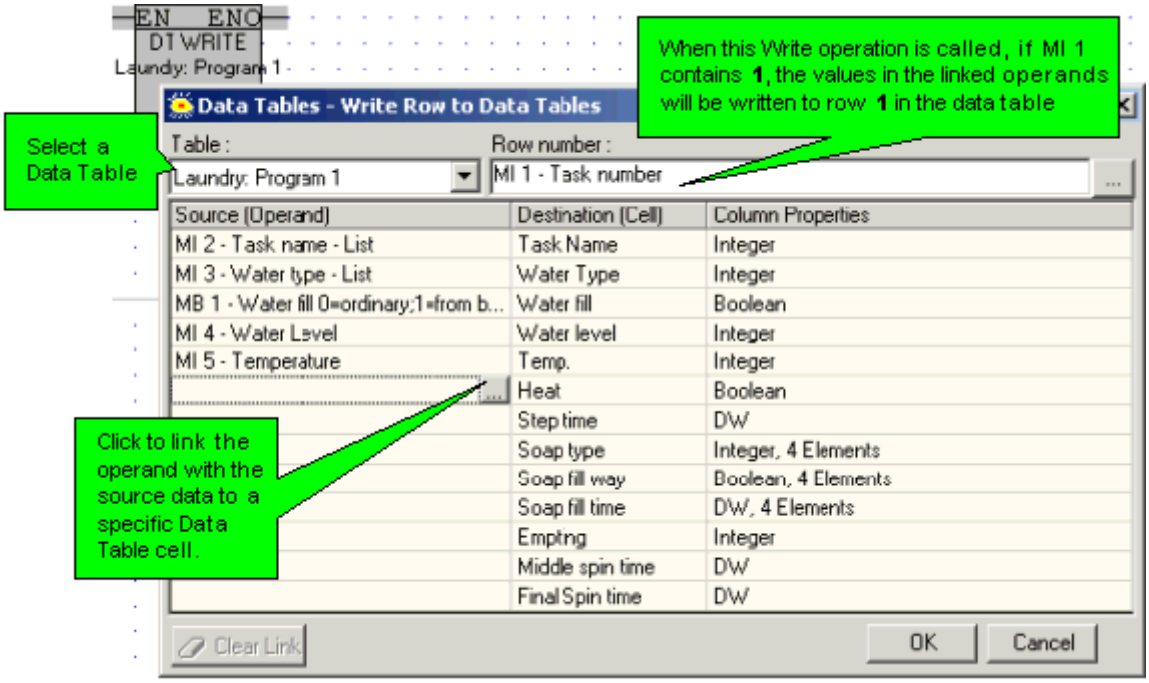


Write Row

Use the Write function to select PLC memory operands and read their data into Data Table rows.

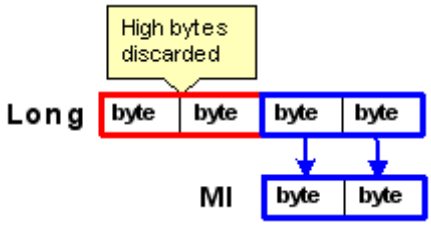
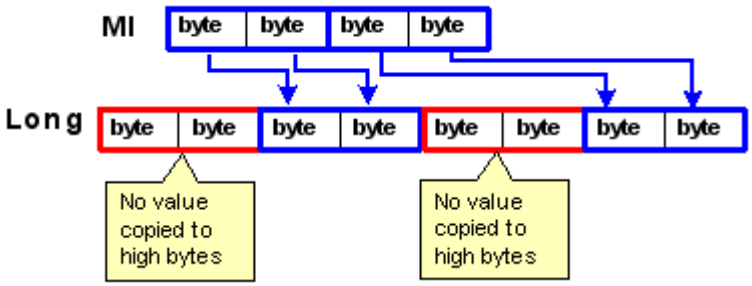
Values are read from the PLC into the Data Table cells that are linked to it in the Write function.

Note that you provide a Start Address for the PLC memory operands; the Write function will take a vector of operands that will fit the number of rows in the Data Table.



Writing to flash via ladder

Columns

Note •	'Part of Project' Data **cannot** be included in **Write Column** functions.
•	Not all Data Types are supported: • Unsupported types: Boolean, Byte, String, all 'Address of' types. • Supported Data Types: Integer (16-bit), Long, Float, Timer (32 bit)
•	When longer data types are copied to shorter data types, the longer values are truncated. 
•	When shorter data types are copied to longer types, each source value is copied to the lower bytes of the destination. 

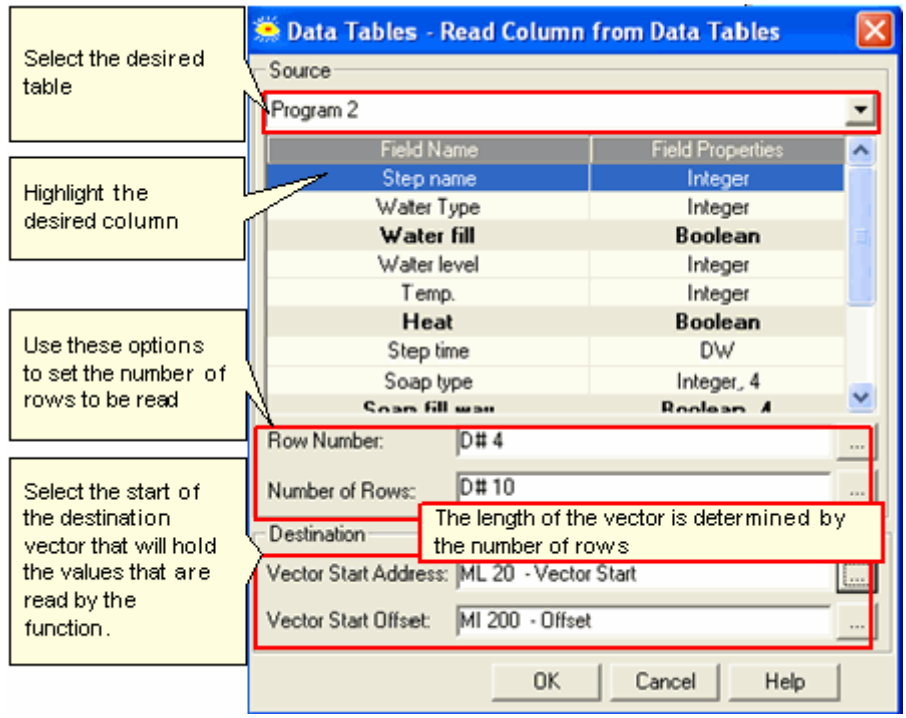
Supported Read Write

Data Table	PLC Operand	Read	Write Result

Byte	Integer	1 Byte is read into the first 8 bits of Integer (LSB).	The first 8 bits of the Integer are written into a Byte. The last 8 bits of the Integer (MSB) are discarded.
Byte	Long Integer		
Integer	Integer		
Integer	Long Integer	1 Integer is read into the first 16 bits of a Long.	The first 16 bits of the Long are written into an Integer. The last 16 bits of the Long are discarded.
Long Integer	Integer	First 16 bits of Long are read into an integer. The last 16 bits of the Long are discarded.	An Integer is written into the first 16 bits of a Long.
Long Integer	Long Integer		
Timer	Timer		
Float	Float		

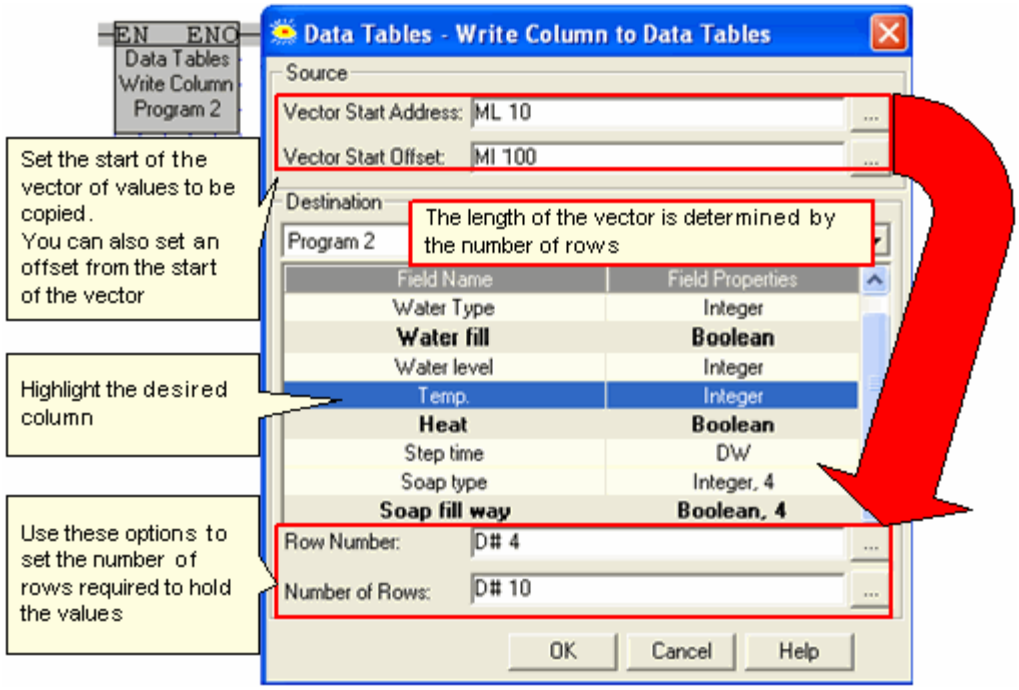
Read Column

A column in a Data Table is the source for the Read function. Values are read from the Data Table into the operands that are linked to it in the Read function, according to FIFO.



Write Column

PLC operands are the source for the Write function. Values are read into the Data Table cells that are linked to it in the Write function. Values are read from the operands into the Data Table according to FIFO.



Read/Write Direct

These operations access the values in the database **without** reference to table structure

Database: Read Direct

The Read Direct operation copies data from the data tables into a vector of registers within the controller.

1. Select Direct: Read from the Data Tables menu.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.

- Operands A & B determine the **data's destination** --**to where** the data from the data table will be copied.

  Operand A: sets the register type for the target vector and the start register.

  Operand B: determines the offset, in registers, from the start register.

- Operands C, D, & E determine the **data's source** --**from where** in the data table the data will be copied.

  Operand C: contains the start byte of the source vector within the data table.

  Operand D: determines the offset, in bytes, from the start register.

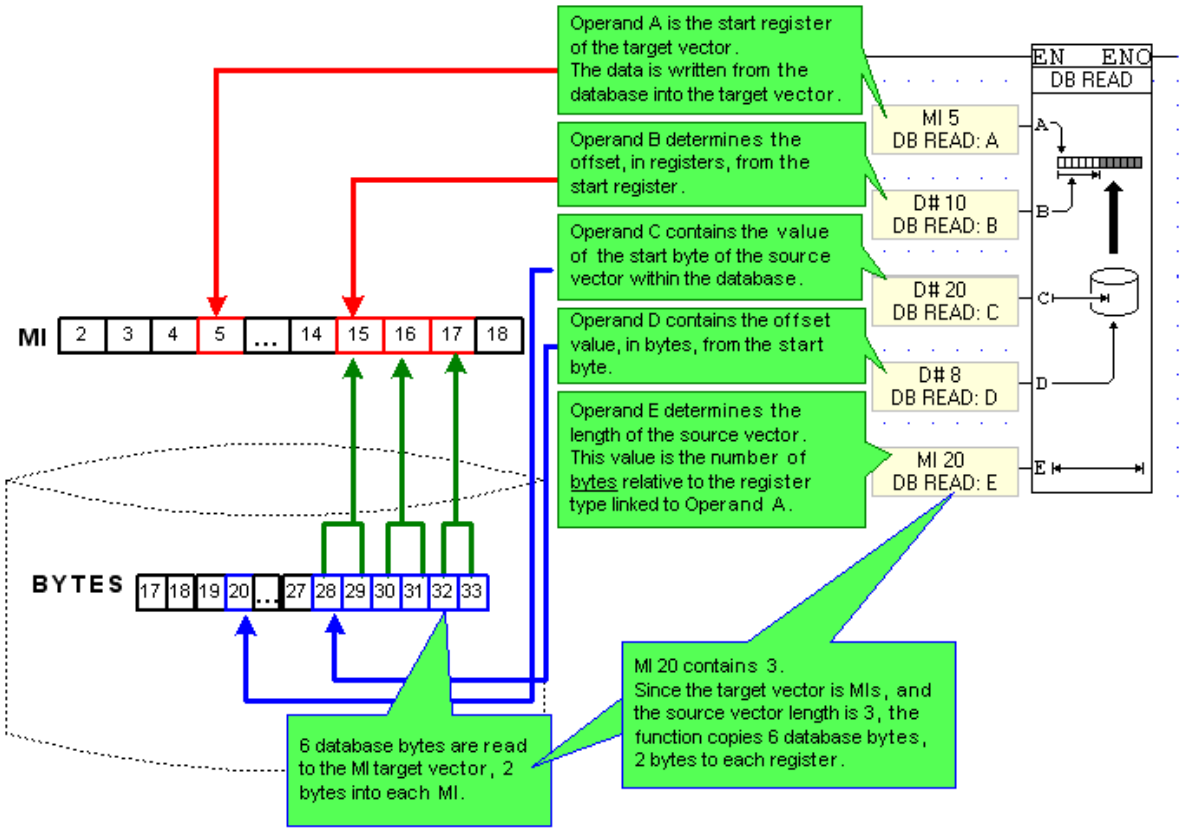
  Operand E: determines the length of the source vector.

Note that the length is relative to the **type** of register linked to Operand A. For example, if Operand A is linked to an MI and Operand E contains 5, 10 bytes of data will be copied from the data table into 5 MIs, 2 bytes into each MI.

If Operand A is linked to a double register; ML or DW; and Operand E contains 2, 8 bytes of data will be copied into 2 double register.

Read Example

Below, database bytes 28, 29, 30, 31, 32, and 33 are read and written into MIs 15, 16, and 17.



Database: Write Direct

The Write operation copies data a vector of registers into the database.

1. Select Data Block Read from the Data Tables menu.
2. Place the function in the desired net.
3. Link the desired Operands and Addresses.

- Operands A & B determine the **data's source** --**from which** registers the data will be copied.

  Operand A: sets the register type for the target vector and the start register.

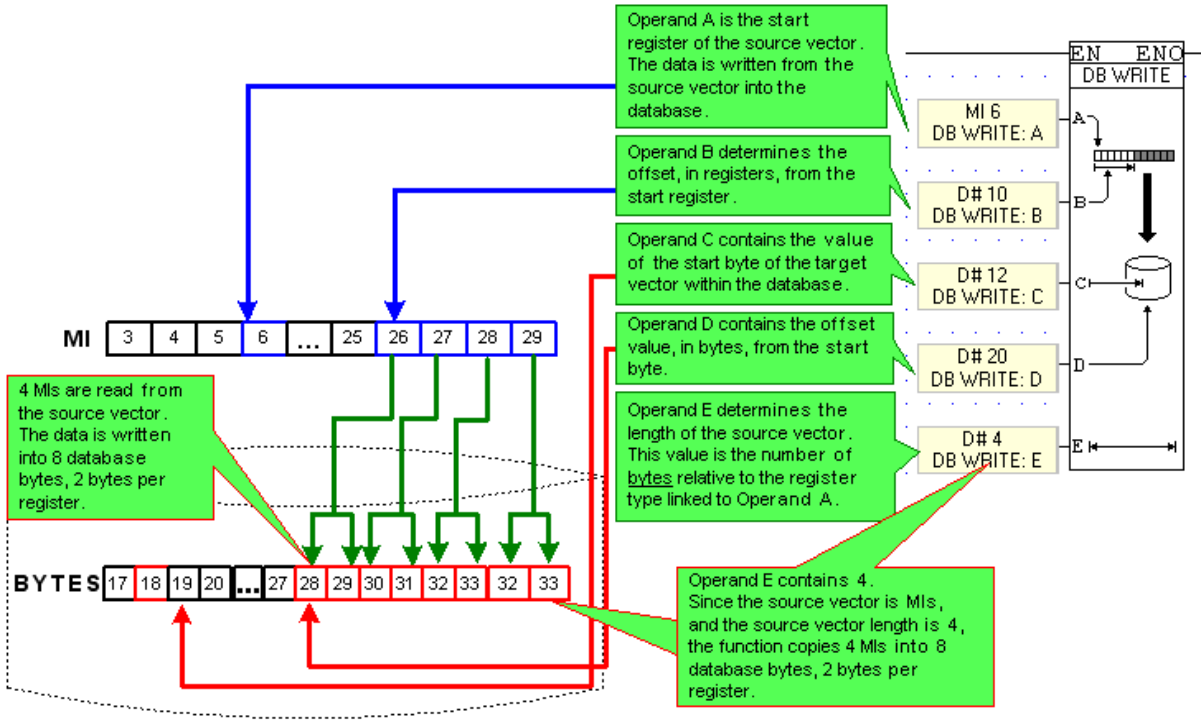
Operand B: determines the offset, in registers, from the start register.

- Operands C, D, & E determine the **data's destination**--**to where** in the database the data will be written.
  Operand C: contains the start byte of the source vector within the database.
  Operand D: determines the offset, in bytes, from the start register.
  Operand E: determines the length of the source vector.

  Note that the length is relative to the **type** of register linked to Operand A. For example, if Operand A is linked to an MI and Operand E contains 5, the data from 5 registers will be copied into 10 database bytes, 2 bytes per MI.
  If Operand A is linked to a double register; ML or DW; and Operand E contains 2, the data from 2 double registers will be copied into 8 database bytes, 4 bytes per ML or DW.

Write Example

Below, MIs 26, 27, 28, 29 are written into database bytes 28 through 33; each register is copied into 2 bytes within the database.

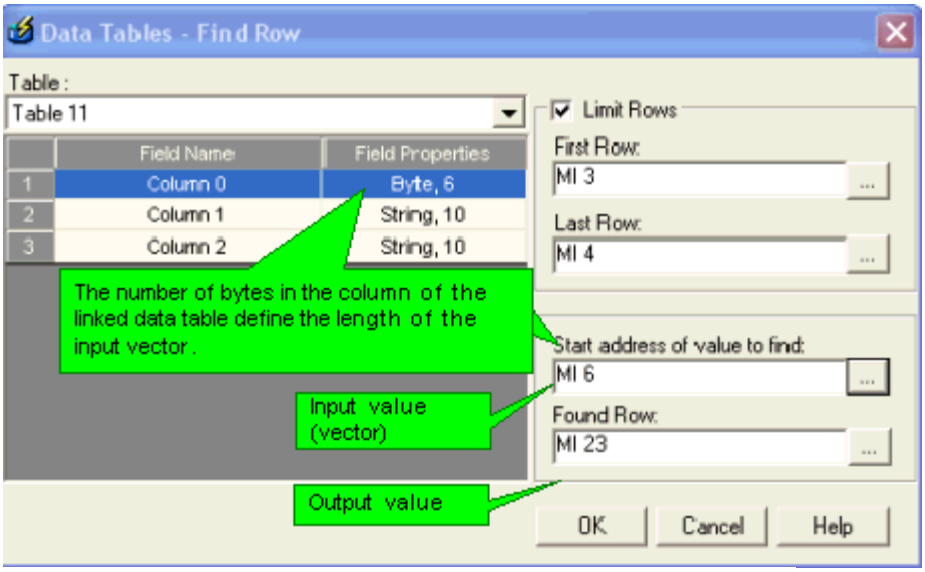


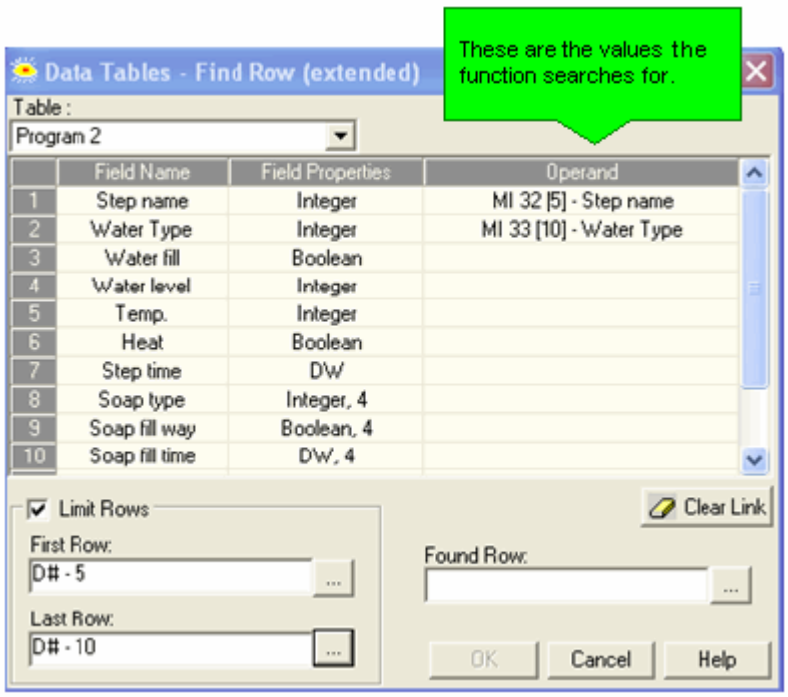
Data Tables: Find Row, Find Row Extended

Find Row and Find Row Extended are located on the Data Tables menu. These functions search through a data table, comparing the input value with the values in the data table.

- Find Row:
  If a matching value is found, the number of the row is stored in the output value.

- Find Row Extended:
  This function enables you to search for more than one value. The number of the row containing all of the values is stored in the output value.





Parameter	Purpose

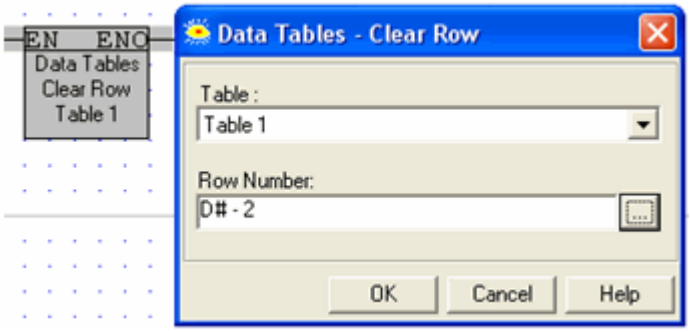
Name	
Table	Click on the drop-down arrow to select a table from the project, then click the desired column. The number of bytes in the column of the linked data table define the length of the input vector.
Limit Rows	Check this option to limit the number of rows the function will search.
Start Address	The length of the input vector is determined by the number of bytes in the selected data table column. If, for example, the column contains 6 bytes, the vector will be 3 MIs long. Note that a string must end with a null (0) character.
Found Row	If a matching value is found, the number of the row is stored in the output value. Note that: - if the value is not found, -1 will be the value returned by the function. - if the row is not found, if, for example, the number given for the first row is higher than the number given for the last row, the value will be -2.

Data Tables: Clear, Row, Column, Table

These functions are located in the Data Tables menu. Clear enables you to use a Ladder condition to delete values in a particular table.

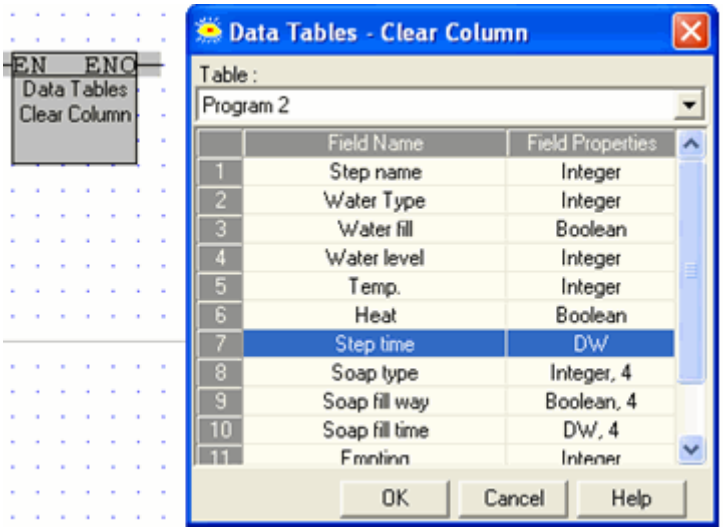
Clear Row

Select the desired Data Table. You can determine which row will be cleared either by entering the row number, or linking to an MI address containing the row number.



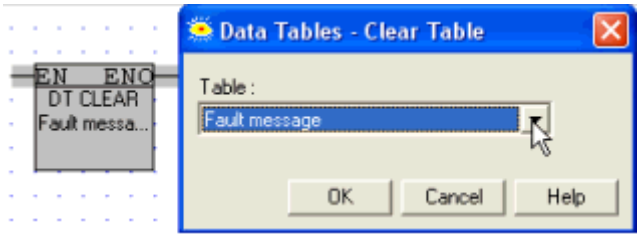
Clear Column

Select the desired Data Table to display its columns. You determine which column will be cleared by clicking it.



Clear Table

Select the desired Data Table. When the function is activated, all of the tables values will be cleared.

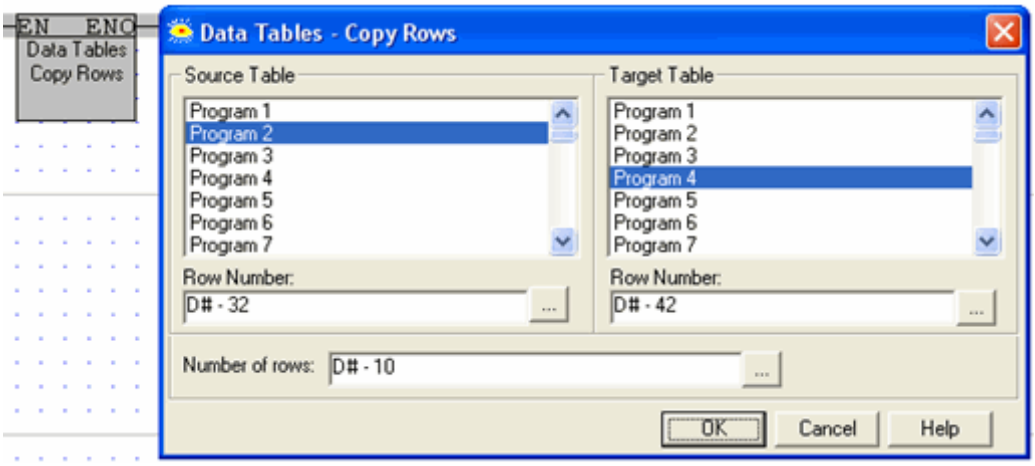


Data Table to Data Table: Copy

These functions enable you to transfer values within the same or between different Data Tables. They are located on the Data Tables menu.

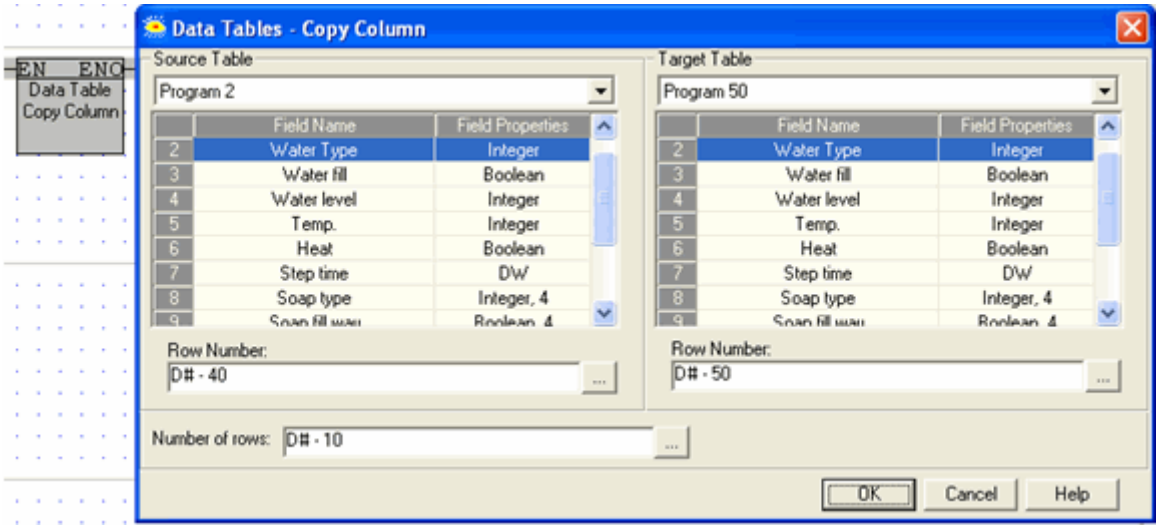
Copy Rows

Select the source table and target table, and make the appropriate selections.



Copy Column

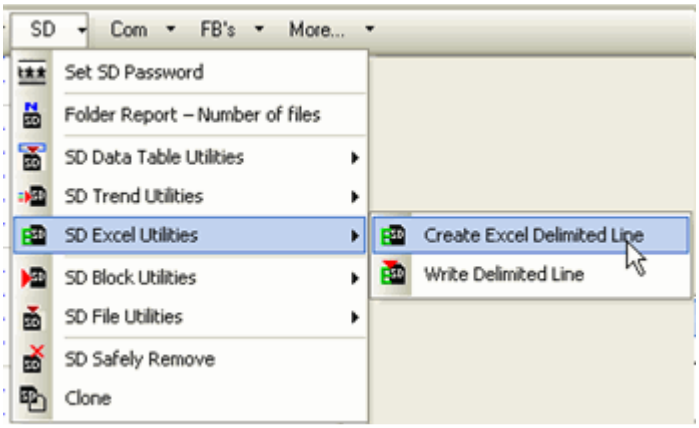
Note that the columns you select must have the same structure.



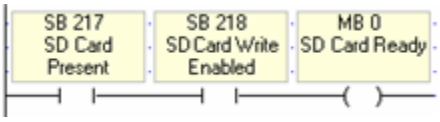
SD Ladder Functions

Use the SD ladder functions to read and write data to and from an SD card.

The functions are located on the SD menu on the Ladder toolbar.



If you include SD functions in your application, build a net that uses SB 217 to check that the SD card is in the PLC and correctly formatted, and SB 218 to check that the card is write-enabled, if it contains a write-protection tab



SD card Functions

Category	Purpose	Functions
SD Password	Use this to guard SD data.	Set SD Card Password
Folder Report	Reports the number of files currently in an SD folder, and	Folder Report Function

	the number of files can still be created in that folder.	
SD Data Table	Use these to read and write data between Data Tables and SD card.	Log Data Table Row to SD Card Copy PLC Data Table to SD Copy SD to PLC Data Table Find Index or Tag in .udt
SD Trend	Record an entire or partial trend to a .utr file in the SD card Trends folder	Save Trend to SD Card Stop Saving Trend to SD
SD Excel	Use Create Excel Line to pull numeric data or text strings from the PLC and use delimiters to structure a line; then use the function Write Delimited Line to create an Excel file on an SD Card. The functions can create .csv and .txt lines	Create Excel Delimited Line Write Excel Delimited Line to SD
Data Blocks	Use these to create data storage areas in the SdBLocks folder on a SD card. SD Data Blocks may reach a total of 4G, or a single Block may be up to 4G. A Data Block comprises Sub-Blocks of 512 Bytes	Create SD Block Read from SD Block to Vector Write from Vector to SD Block
Data Files	Use Windows Explorer to store any type of file onto an SD card, such as .html or .jpg. The SD File Functions enable your Ladder application to read and write these files in 'chunks' of 512 bytes. You can also use these functions to pull data from the PLC and create files on the SD card.	SD File: Open Read/Write next Chunk SD File: Close Delete SD File SD File Info Rename SD File
Safely Remove SD	Use this to ascertain when an SD card may be safely removed from the PLC	Remove SD
Clone to/from SD	Use this to clone a complete PLC and application, Data Tables, or operand values **from** a PLC or install such clone files **to** a PLC of the	Clone to/from SD

	same model	

SD System Operands

#	Description	Turns ON when:	Turns OFF when:	Reset by:
SB 217	SD Card Present	An SD Card is in the slot, and is formatted to FAT32	SD Card is not found, or is incorrectly formatted	OS
SB 218	SD Card Write Enabled	Write is enabled: the card's write-protect lock is off	Write is disabled: the card's write-protect lock is on	OS
SB 219	SD FIFO Empty (SD Card may be Ejected)	Power-up No SD Card is in Slot No SD requests exist	There are no SD requests pending, such as Data Table Copy/Log, Alarms, or from Info Mode	OS
SBs 324-29 are linked to the SD File utilities.				
SB 324	SD: Open File (Read to SD) (Status messages in SI 67)	When Ladder function SD File: Open successfully activates a file for Read	When Ladder function SD File: Close finishes closing an open file and SB 327 (EOF) turns ON	OS. At Power-up and at SD File: Close
SB 325	SD File: Read Chunk in Progress (a Chunk is 512 bytes long)	When the Ladder function SD: Get Next File Chunk is reading a chunk into a vector	When the Ladder function SD: Get Next File Chunk has finished reading the chunk	OS. At Power-up
SB 326	SD Read File: End Of File (EOF, entire file has been read)	When the When the Ladder function SD: Get Next File Chunk reads the final Chunk	When the last chunk has been read, and when Ladder function SD File: Close start	OS. At Power-up and at SD File: Close
SB 327	SD: Open File (Write to SD) (Status messages in SI 67)	When Ladder function SD File: Open successfully activates a file for Write on a SD card		
SB 328	SD File: Write Chunk in Progress (a Chunk is 512 bytes long)	When the Ladder function SD: Get Next File Chunk is writing a chunk into a vector		
SB329	SD Write File: End Of File (EOF, entire file has been read)	When the When the Ladder function SD: Get Next File Chunk writes the final Chunk		
SB 340	Log to SD in Progress	Row is being copied from DT to SD Card	When copy is complete	OS
SB 341	Write Data Table from **PLC to SD** in Progress	Entire Data Table is being copied from DT to SD Card	When the Write process is complete	OS
SB 342	Read Data Table from **SD to PLC** in Progress	Entire Data Table is being copied from SD Card to DT	When the Write process is complete	OS
SB 343	File Report in Progress	While Report process is in progress	When the Report is complete	OS
SB 345	Email Send in Progress	ON when function is busy		
SB 344	Write delimited line to SD in Progress	While line is being written	When the Write process is complete	OS
SB 346	SD Data Block 0 Busy	When a Write or Read utility is being run on a Data Block	When no utility is running	OS
SB 347	SD Data Block 1 Busy			
SB 348	SD Data Block 2 Busy			

SB 349	SD Data Block 3 Busy			
SB 352	SD: Enable writing Alarm History to SD	Turned ON by user to write Alarm History to SD Card	Off by default. Causes the PLC to write Alarm History to the PLC	At Power-up, or by user
SB 358	SD: Delete File in Progress	ON when function is busy	OFF when function is not busy	OS
SB 359	Folder Report Function in Progress	ON when function is busy	OFF when function is not busy	OS
SB 366	Clone in Progress (Process can take from several seconds to several minutes)	ON when function is busy	OFF when function is not busy	OS

#	Description	Value	Comments
SI 63	Maximum number of Trend files that can be saved (read-only)	0-64 The maximum amount of Trend files (*.utt files) in a single folder is 64. The value in SI 634shows the number of remaining *.utr files; if 5 *.utr files exist, SI 64 = 59	Initialized at Power-up Updated when:SB 217 is ON and SB 341 turns ON
SI 64	Maximum number of DT files that can be saved (read-only)	0-64 The maximum amount of Data Table files (*.udt files) in a single folder is 64. The value in SI 634shows the number of remaining *.udt files; if 5 *udt files exist, SI 64 = 59	Initialized at Power-up Updated when:SB 217 is ON and SB 341 turns ON
SI 66	SD Card Status Messages	This SI is a bitmap; a bit turns ON to indicate status. All bits OFF – No errors Bit 1 – Read: End Of File indication Bit 2 – Can't open file Bit 3– Error while writing to a file Bit 4 – Error while reading from a file Bit 5 – Failed to close a file Bit 6 – SD is full Bit 7 – Path not found Bit 14 - Turns ON when SD is inserted into slot and PLC runs checks, turns OFF when SB 217 turns ON	Initialized at Power-up. While the application is running, the user application must reset the bits.
SI 67	SD Card, Read Files: Status	Value 0= No error 1= No SD card in Slot 2= Vector is not long enough to contain data (may be at upper address limit of that data type) 3= Path to SD file not found 4=Another file is currently open 5 = File is closed 6 = Busy: previous request in progress 7 = File Open Error 8 = Read Error 9 = File Close error	SI 67 reports status for the following SD File utilities: • Read SD File: Open • Read Next File Chunk • Read SD File: Close
SI 68	SD Card, Write Files:	Value	SI 68 reports status for the

	Status	0 = No error 1 = No SD card in Slot 2 = Vector is not long enough to contain data (may be at upper address limit of that data type) 3 = Path to SD file not found 4 = Another file is currently open 5 = File is closed 6 = File Open error 7 = Write Error 14 = File Close error	following SD File utilities: • Write SD File: Open • Write Next File Chunk • Write SD File: Close
SI 69	SD Card: File Open Time (may signal file fragmentation)	Time required to open SD files, in units of 10mSec.	Each time a file is opened, the OS updates this value. A typical first write (open + write) = approx. 500mSec, typical first read (open + read)= approx. 60mSec Over time, this may increase due to file fragmentation. If the time becomes to great, the card should be reformatted Reset at Power-up and when SD card is removed.
SI 76	SI 76 Number of Alarms currently in History Buffer	Shows the number of Alarms in the history buffer.	If SB 352 SD: Write Alarm History to SD is ON, the Alarms in the buffer are automatically written to the SD card. Initialized by the user, or when the PLC is initialized.
SI 160	SD Trend 1 status	This SI is a bitmap; a bit turns ON to indicate status when the function Start Saving Trend to SD runs. All bits OFF – No errors Bit 4 – Start Saving Trend is in progress for another Trend Bit 7 – This Trend does not exist (may result when an MI is used to provide the Trend number, and the value points to a non-existent Trend) Bit 8 – Start Saving Trend is in progress for this Trend Bit 9 – Start Saving Trend failed	
SI 161	SD Trend 2 status		
SI 162	SD Trend 3 status		
SI 163	SD Trend 4 status		
SI 164	SD Trend 5 status		
SI 165	SD Trend 6 status		
SI 166	SD Trend 7 status		
SI 167	SD Trend 8 status		
SI 330	SD: Write DT from PLC to SD - Total Amount of Data to be Copied (blocks of 512 bytes)	When the application runs the function Copy Data Table to SD, SI 330 shows the total number of blocks of data **to be copied** from the PLC.	Initialized at Power-up
SI 331	SD: Write DT from PLC to SD - Remaining Amount (blocks not yet copied)	Shows how many blocks of data **remain** to be copied. The value increases by 1 each time a block is copied.	Initialized: When the PLC begins to copy a new block of data to the SD card At Power-up.
SI 332	SD: Read DT SD to PLC - Total Amount of Data to be Copied (blocks of 512 bytes)	When the application runs the function Copy Data Table to PLC, SI 332 shows the total number of blocks of data **to be copied**	Initialized at Power-up

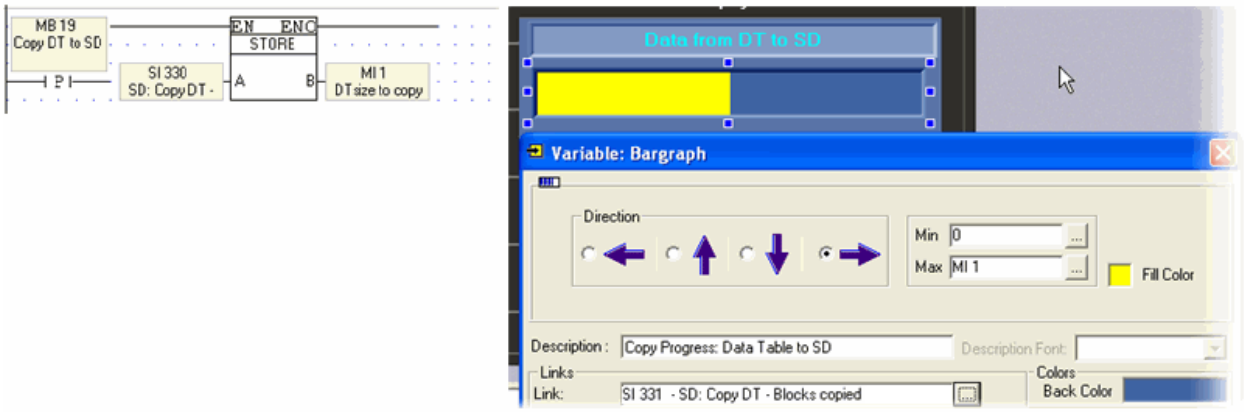
		from the SD.	
SI 333	SD: Read DT from SD to PLC - Remaining Amount (blocks not yet copied)	Shows how many blocks of data **remain** to be copied. The value increases by 1 each time a block is copied.	Initialized: When the PLC begins to copy a new block of data from the SD card At Power-up.
#	**Description**	**Value**	**Comments**
SDW 59	SD Card: Free space (bytes)	Capacity given in 512-byte chunks. The value is written when SB 217 turns ON, and is updated at each write operation. The operand is reset when SB 217 turns OFF.	Initialized at Power-up.

PLC Name

If you apply a PLC name, the PLC writes this name to the files it creates on the SD Card.

HMI Progress Bar

You can use SI 330 and SI 331 to create a progress bar on an HMI display that shows when the PLC is writing data to the SD; and SI 332 and SI333 to show data being written from the SD to the PLC. To create a progress bar, use the elements shown in the following image. Note that the PLC copies data at a rate of .5k per second. This means that a PLC requires approximately 24 seconds to transfer a Data Table comprising 120k to an SD card.



Removing the SD Card

To indicate that the SD card may be safely removed, you can link an HMI element to SB 219 SD FIFO Empty (SD Card may be Ejected).

Set SD Card Password

You can guard the SD card with a password.

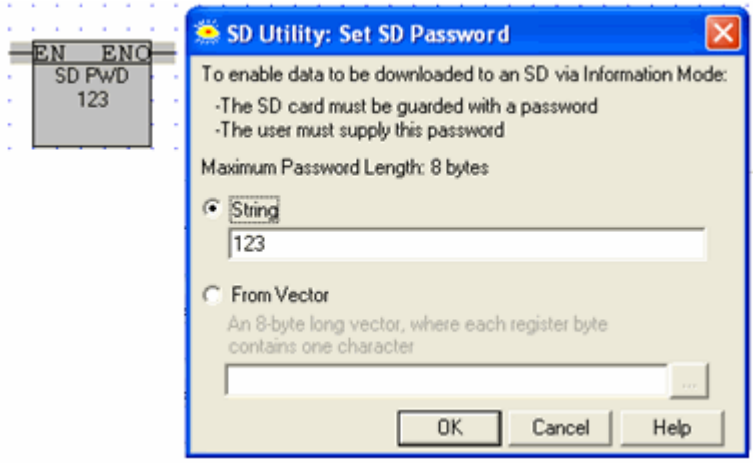
Note that when the PLC is in Information Mode, a user can only download data to an SD card:

- If the SD is guarded with a password.
- If the user can supply the password. The only exception is Firmware, which may be downloaded without password.

Note • The SD Password is case-sensitive

The maximum Password length is 8 bytes. Each register byte contains one character.

1. Place a Set SD Password function in the Ladder; you can either directly assign a text password, or provide it via MI.

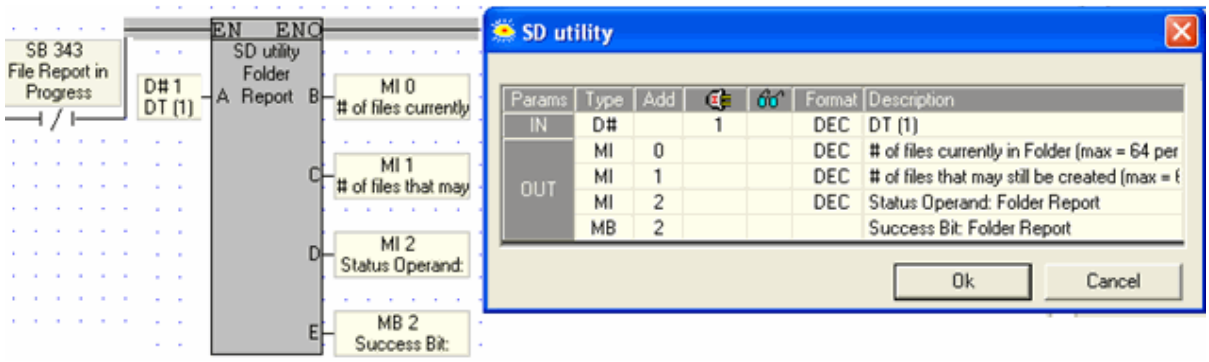


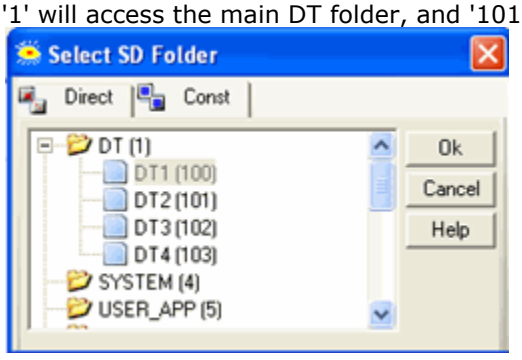
SD Card: Folder Report Function

Use this Function to see:

- The number of files are currently in an SD folder
- The number of files can still be created in that folder

Notes •	The function reports only on the types of files that are linked to a particular folder. For example, each Data Table folder (DT) may contain a maximum of 64 .udt files. If Folder Report is set to DT folders, it only reports the number of .udt file. If there are other file types present in the folder, they are ignored.
•	If there are fewer files than the maximum allowed, but the SD card capacity is exceeded, SI 66 will indicate SD Card Full.
•	Use SB 343,SD: File Report in Progress, as a condition to running the function



Parameter	Name	Purpose
Input	SD Folder: Select SD Folder	Either select a folder, or link an operand. To use an operand value to access folders, use the numbers shown in Select SD Folder; where '1' will access the main DT folder, and '101 will access folder DT2. 
Output	Number of files currently in Folder	The number of files currently in Folder (max = 64 per folder)
	Number of Files that may	The number of files that may still be created (max = 64 per folder)

	still be created	
	Folder Report: Status Operand	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • All bits OFF – No errors, and the SD card is idle • Bit 1 - SD Card internal error • Bit 2 - SD file is incorrect type • Bit 3 - There is no SD card in the slot • Bit 4 - The SD card has failed (Check SI 66) • Bit 5 - Path not found
	Folder Report: Success Bit	Turns ON when the Report is complete. It remains ON until it is reset by the application, or until the application calls the function.

Note SB 359: Folder Report Function in Progress (ON when function is busy)

SD Card and Data Table Functions (Ladder)

These functions enable you to:

- Log a single row of data from a Data Table into a .ulg file located on the SD card
- Write all or part of a Data Table into a .udt file located on the SD card
- Read all or part of an SD card .udt file to a Data Table
- Search for tagged sections in a .udt file

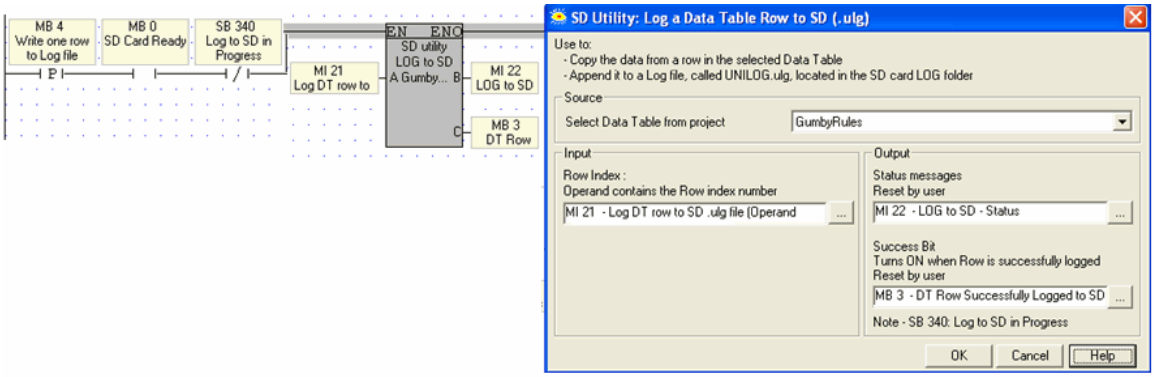
If a Data Table is marked as Part of Project, you **cannot** copy it or log lines from it to an SD card.

#	Description	Value	Comments
SI 64	Maximum number of DT files that can be saved (read-only)	0-64 The maximum amount of Trend files (*.udt files) in a single folder is 64. The value in SI 634shows the number of remaining *.udt files; if 5 *udt files exist, SI 64 = 59	Initialized at Power-up Updated when:SB 217 is ON and SB 341 turns ON

Log Data Table Row to SD Card

1. To log a row from a Data Table, build a net that includes the function SD> Write Log Line to SD.
   Use SB 340 to ensure that the PLC is not currently logging a row to the SD card.

   When the application writes this type of data to the SD card, it creates a single file called UNILOG.ulg in the LOG folder, and then appends each new line from the selected Data Table to this log file.


MB 4
Write one row to Log file
P
MB 0
SD Card Ready
SB 340
Log to SD in Progress
MI 21
Log DT row to
EN
ENO
SD utility
LOG to SD
A Gumby...
B
C
MI 22
LOG to SD
MB 3
DT Row
SD Utility: Log a Data Table Row to SD (.ulg)
Use to:
· Copy the data from a row in the selected Data Table
· Append it to a Log file, called UNILOG.ulg, located in the SD card LOG folder
Source
Select Data Table from project
GumbyRules
Input
Row Index :
Operand contains the Row index number
MI 21 - Log DT row to SD .ulg file (Operand
Output
Status messages
Reset by user
MI 22 - LOG to SD - Status
Success Bit
Turns ON when Row is successfully logged
Reset by user
MB 3 - DT Row Successfully Logged to SD
Note - SB 340: Log to SD in Progress
OK
Cancel
Help

Parameter Name	Purpose
Source	Selects the Data Table you want to log from.
Row index	Determines which row in the table will be logged.
Status messages	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. All bits OFF – No errors, and the SD card is idle Bit 1 – The SD card was formatted in an SD Tools version that is not compatible with the VisiLogic application in the PLC. or VisiLogic version is not compatible with the PLC OS. Check to see if you need to update versions. Bit 2 – The data in the SD is not compatible with the data in the Data Table Bit 3 – Data checksum error Bit 4 – Failed to open file Bit 5 - Failed to write to the SD file Bit 6 - Failed to close file Bit 7 - In progress Bit 8 - No SD card found Bit 9 - SD error, check SI 66 for error message Bit 10 – Requested Data Table does not exist
Success Bit	Turns ON when the data is successfully written to the SD card. It remains ON until it is reset by the application, or until the application calls the function.

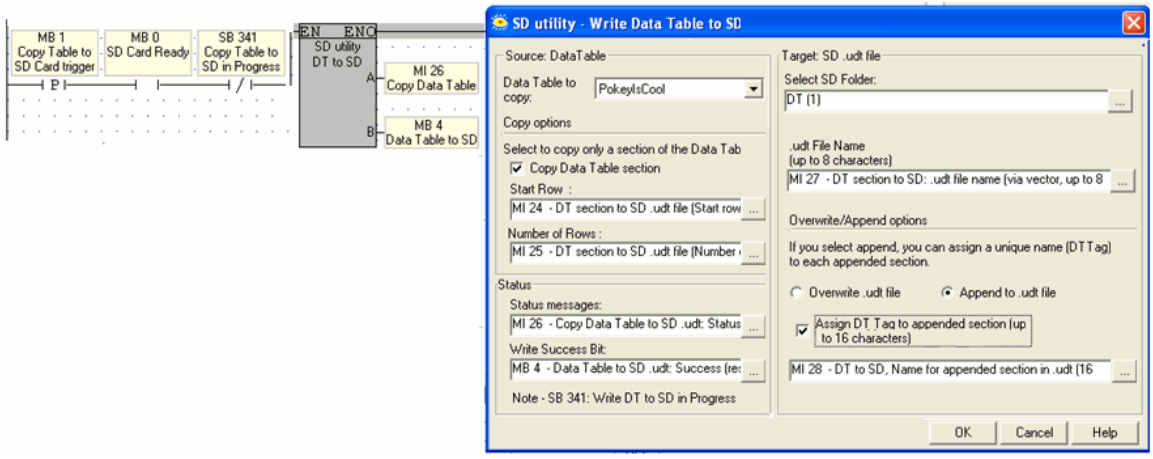
Data Table To / From SD Card

The Ladder function DT to SD creates .udt files and saves them in the main DT folder or in one of four sub-folders. DT1, DT2, DT3, DT4.
Each folder can contain 64 files, for a total of 320 .udt files.

Write Data Table to SD (Copy DT to SD)

1. To copy an entire or partial Data Table, build a net that includes the function SD> Write DataTable to SD.
   Use an inverted contact of SB 341 to ensure that the PLC is not currently writing to the SD card.
2. Set the options to copy all or part of a Data Table.

   When the application writes this type of data to the SD card, it creates a file with the extension .udt in the selected DT folder.



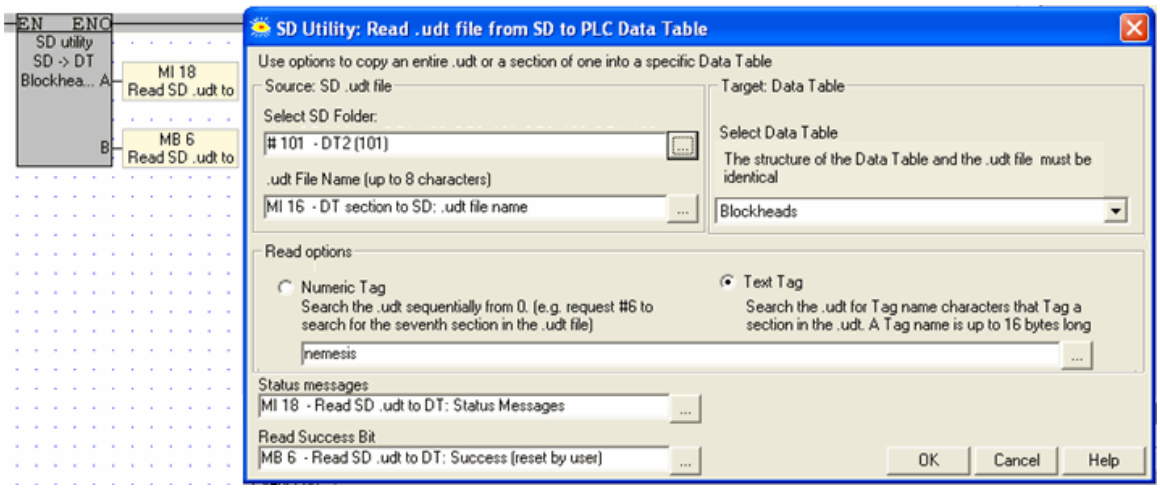
Parameter Name	Purpose
Source: Data Table to copy	Selects the Data Table you want to write from.
Copy options	Select to copy all or part of a Data Table. Selecting Copy enables the Start Row and Number of Rows parameters.
Target: SD Folder	This is where the .udt file will be stored on the SD card. You can select the folder, or provide the Folder number via register. Values point to folders as follows: 1=the main DT folder, 100=DT1, 101=DT2, 102=DT3, and 103=DT4.
.udt File Name	Can be up to 8 characters long, and may be provided by constant text or register. Note that if the name comes from an MI, the function copies a vector 8 bytes long, or until it finds a 'null' character.
Overwrite/Append	If the function finds a .udt file in that folder of the same name, • Selecting Overwrite replaces the file. • Selecting Append adds the new data to the existing .udt file. You can assign a unique name (DT Tag) to each appended section, marking the sections for later use in your program. The Tag may contain up to 16 characters.
Status messages	This MI is a bitmap; a bit turns ON to indicate status.

	The MI is initialized when the function starts. All bits OFF – No errors, and the SD card is idle Bit 1 – The SD card was formatted in an SD Tools version that is not compatible with the VisiLogic application in the PLC. or VisiLogic version is not compatible with the PLC OS. Check to see if you need to update versions. Bit 2 – The structure of the .udt file and the Data Table are not identical Bit 3 – Data checksum error. Please send application and any related information to support@unitronics.com. Bit 4 – Failed to open file Bit 5 - Failed to read from file Bit 6 - Failed to close file Bit 7 - In progress Bit 8 - No SD card found Bit 9 - SD error, check SI 66 for error message Bit 10 – Requested Data Table does not exist
Success Bit	Turns ON when the data is successfully written to the SD Card. It remains ON until it is reset by the application, or until the application calls the function.

Note • The maximum number of Data Table files that can be created in a folder SD card is 64, including the main DT folder.

Read .udt file from SD to PLC Data Table (Copy SD >DT)

1. To copy .udt data from an SD card into a Data Table, build a net that includes the function SD> Copy Data to PLC Data Table.
   Use an inverted contact of SB 342 to ensure that the PLC is not reading writing from the SD card.
   Note that in order to copy data, the Data Table structure in both PLC and SD card must be identical: equal number of rows, equal numbers of columns, and column data types.



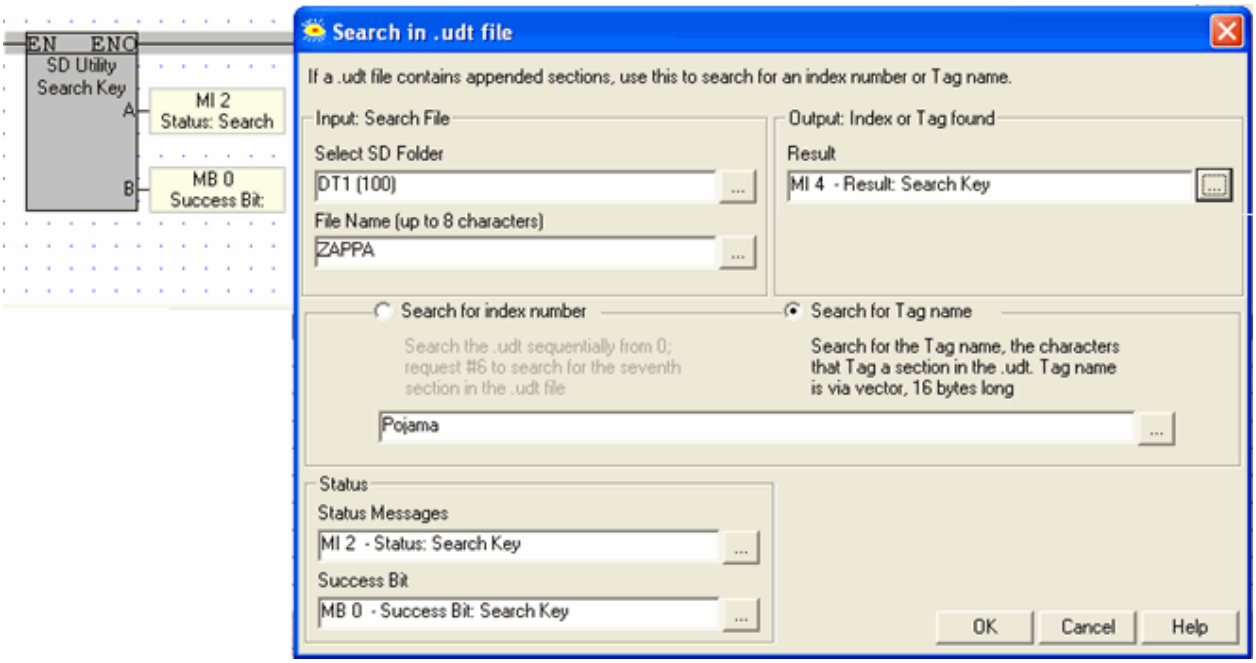
Parameter	Purpose

Name	
Select SD Folder	This is where the source .udt file is on the SD Card. You can select the folder, or provide the Folder number via register. Values point to folders as follows: 1=the main DT folder, 100=DT1, 101=DT2, 102=DT3, and 103=DT4.
File Name	The Table Name can be up to 8 characters long, and may be provided by constant text or register.
Read Options	If the .udt file contains appended sections, you can search for a Numeric or Text Tag.
Target: Data Table	Click on the drop-down arrow to select a Data Table in the project. The Table Name can be up to 8 characters long, and may be provided by constant text or register. Note that if the name comes from an MI, the function copies a vector 8 bytes long, or until it finds a 'null' character.
Status Operand	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • Bit 1 – The SD card was formatted in an SD Tools version that is not compatible with the VisiLogic application in the PLC. or VisiLogic version is not compatible with the PLC OS. Check to see if you need to update versions. • Bit 2 – The data in the SD is not compatible with the data in the Data Table • Bit 3 – Data checksum error • Bit 4 – Failed to open file • Bit 5 - Failed to read from file • Bit 6 - Failed to close file • Bit 7 - In progress (SB 342 ON) • Bit 8 - No SD card found (SB217 (ON) • Bit 9 - SD error, check SI 66 for error message • Bit 10 – Requested Data Table does not exist
Success Bit	Turns ON when the data is successfully read. It remains ON until it is reset by the application, or until the application calls the function.

Search .udt for Tag or Index#

If a .udt file was created using appended sections, you can search it for the index number or tag name.

Use an inverted contact of SB 342 to ensure that the PLC is not reading writing from the SD card.



Parameter Name	Purpose
Select SD Folder	This is where the source .udt file is on the SD Card. You can select the folder, or provide the Folder number via register. Values point to folders as follows: 1=the main DT folder, 100=DT1, 101=DT2, 102=DT3, and 103=DT4.
File Name	The Table Name can be up to 8 characters long, and may be provided by constant text or register.
Tag Type	Search for a Numeric or Text Tag.
Table	Click on the drop-down arrow to select a Data Table in the project. The Table Name can be up to 8 characters long, and may be provided by constant text or register. Note that if the name comes from an MI, the function copies a vector 8 bytes long, or until it finds a 'null' character.
Status Operand	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • Bit 1 – The SD card was formatted in an SD Tools version that is not compatible with the VisiLogic application in the PLC. or VisiLogic version is not compatible with the PLC OS. Check to see if you need to update versions. • Bit 2 – The data in the SD is not compatible with the data in the Data Table • Bit 3 – Data checksum error • Bit 4 – Failed to open file • Bit 5 - Failed to read from file • Bit 6 - Failed to close file • Bit 7 - In progress (SB 342 ON) • Bit 8 - No SD card found (SB217 (ON) • Bit 9 - SD error, check SI 66 for error message • Bit 10 – Requested Data Table does not exist

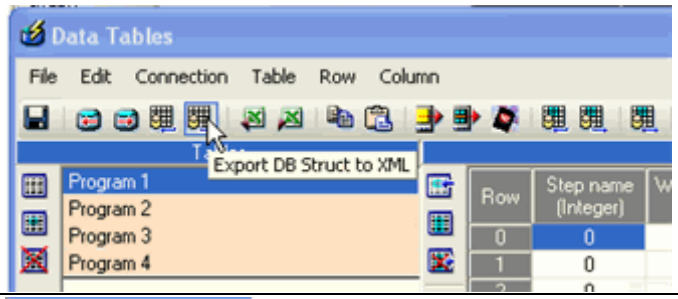
Success Bit	Turns ON when the tag is found. It remains ON until it is reset by the application, or until the application calls the function.

Import data from an SD card into a PLC Data Table

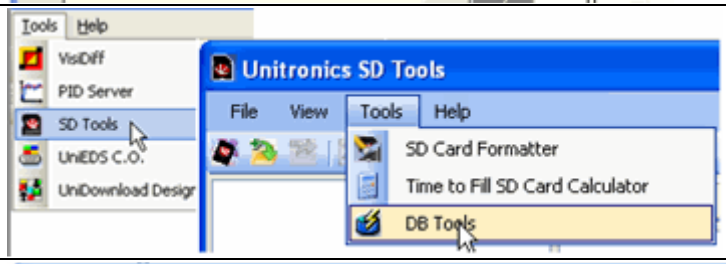
This imports data from a Data Table on the SD card into a Data Table in the PLC.

The Data Tables must be identical. In order to ensure this, follow the recipe below.

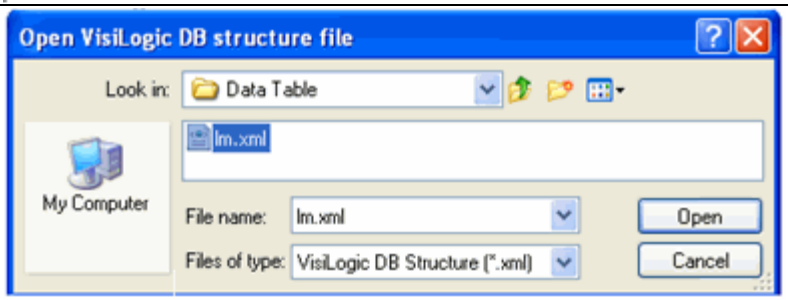
1. Open the Data Table, and click Export the Data Table Structure to convert the table to an .xml file.



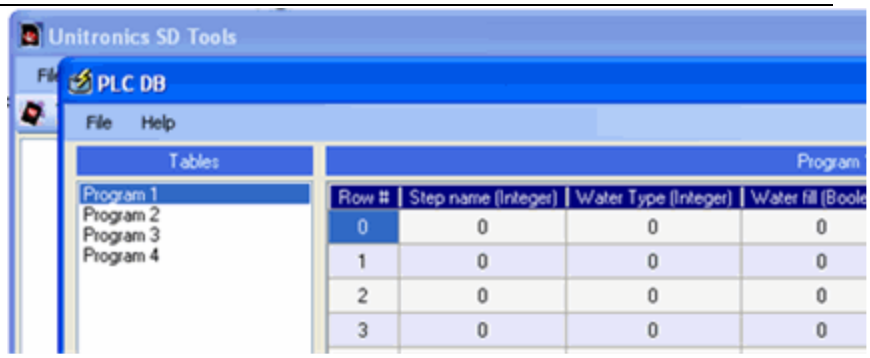
2. From the VisiLogic Tools menu, open SD Tools.
3. From the SD Tools Tools menu, open DB Tools.



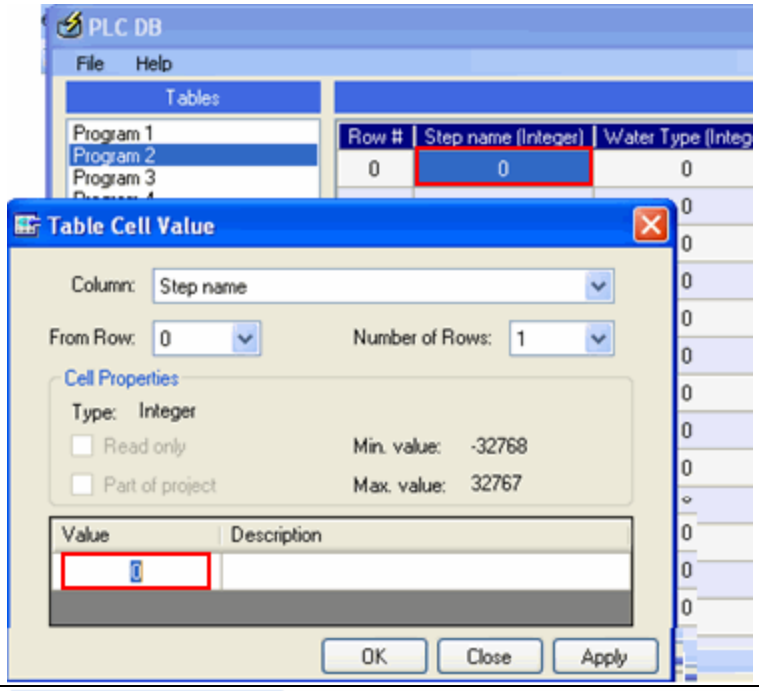
4. Navigate to and select the .xml file.



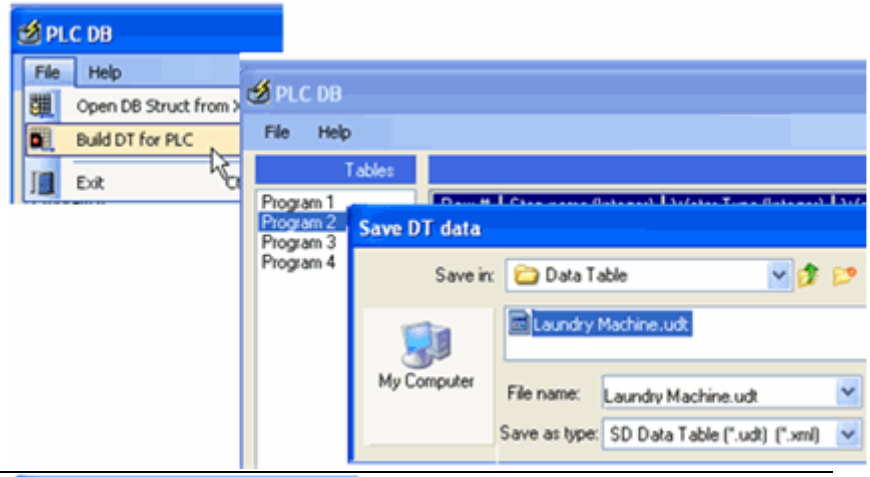
5. SD Tools opens the file for editing.



6. Click on table cell to edit the values.

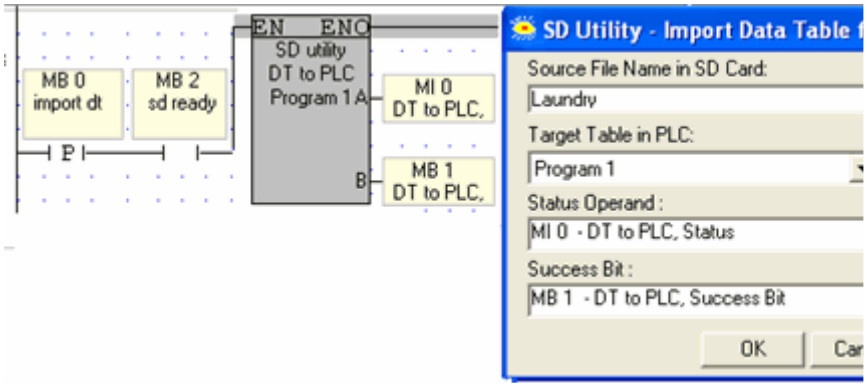


7. From the File menu, select Build DT for PLC.



8. Copy the resulting .udt file to the SD card, and then place it in the PLC.

9. Build a net including the SD utility DT to PLC.

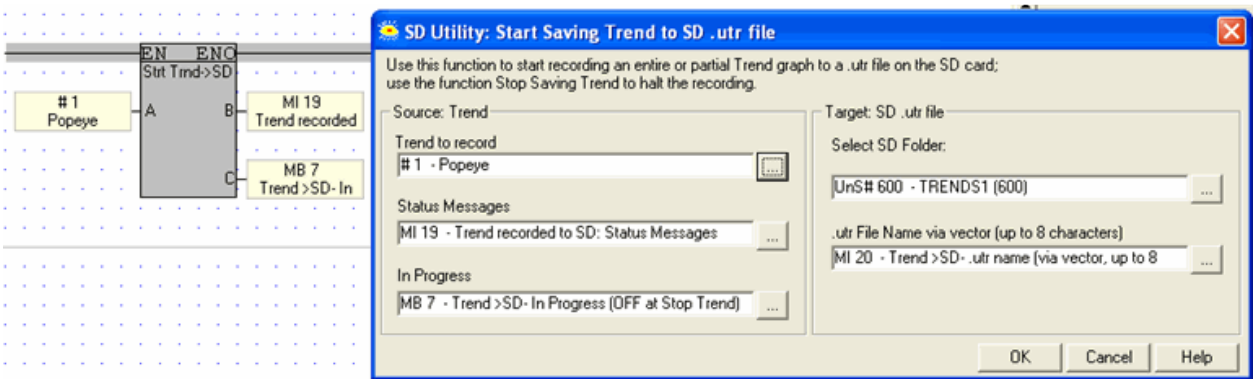


When the Program runs, it will copy the values from the Data Table on the SD card into the PLC Data Table cells.

SD Card and Trends

Use the Start Saving Trend to SD to record an entire or partial trend; and Stop Saving Trend to halt the recording process.

When the application writes this type of data to the SD card, it creates a file with the extension .utr in the Trends folder. Each time you start and stop saving the Trend, the application adds a new segment to the file.



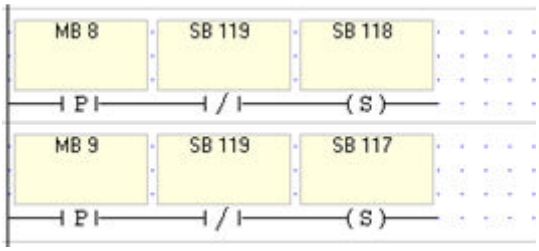
Parameter Name	Purpose
Source Trend Number	Click on the drop-down arrow to select a Trend in the project.
Target SD .utr file	Link an operand to provide a file name. Note that you MUST provide a file name. If the linked register is empty, the Trend will not be recorded to the SD. This is where the .udt file will be stored on the SD card. You can select the folder, or provide the Folder number via register. Values point to folders as follows: 1=the main DT folder, 100=DT1, 101=DT2, 102=DT3, and 103=DT4.
Status Operand	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • Bit 1 – The SD card was formatted in an SD Tools version that is not

	compatible with the VisiLogic application in the PLC. or VisiLogic version is not compatible with the PLC OS. Check to see if you need to update versions. • Bit 2 – The data in the SD is not compatible with the data in the Data Table • Bit 3 .-.Data checksum error • Bit 4 – Failed to open file • Bit 5 - Failed to read from file • Bit 6 - Failed to close file • Bit 7 - In progress • Bit 8 - No SD card found • Bit 9 - SD error, check SI 66 for error message
Success Bit	Turns ON when the data is successfully written to the PLC Data Table. It remains ON until it is reset by the application, or until the application calls the function.

Displaying the saved Trend

You can display Trend curves directly from a .utr file by using the HMI element Trend from SD.

Link SBs 117 and 118 with HMI buttons to enable users to jump between segments. Use the inverted contact of SB119 as a condition as shown below.



When you save a Trend to an SD card, each time you start and stop the save, another segment is added to the .utr file.

#	Description	Turns ON :	Turns OFF :	Reset by:
SB 116	SD Trends to SD: Set to Overwrite .utr	User application	User application	User
Use these to control the display of Trend segments on the HMI screen. Use the inverted contact of SB119 as a condition.				
SB 117	SD Trends: Jump to next segment	User application	User application	User
SB 118	SD Trends: Jump to previous segment	User application	User application	User
SB 119	SD Trends: System busy - Draw Trend is gathering data	User application	User application	User

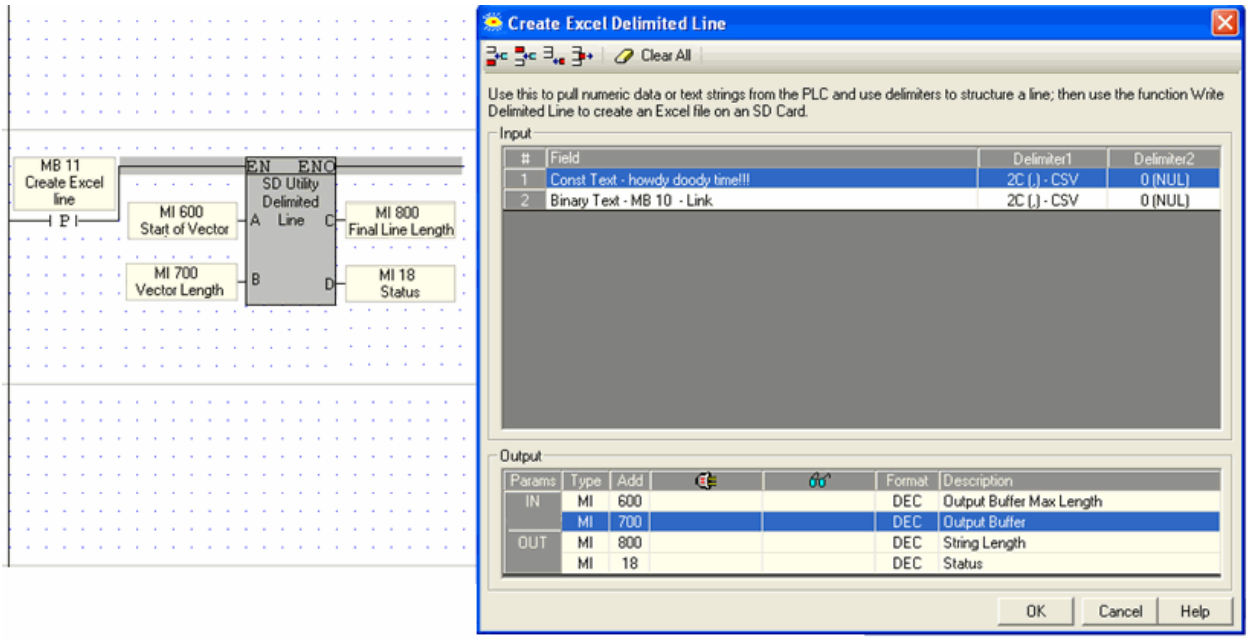
SD Card: Data to Excel

You can write PLC data to Excel files an the SD card using the functions Create Delimited Line to structure a line, and Write Delimited Line to send it to a specified Excel file on an SD Card.

Note that the main EXCEL folder and subfolders EXCEL0, EXCEL1, EXCEL2, EXCEL3 can each contain 64 files, for a total of 320 .csv files.

Create Delimited Line

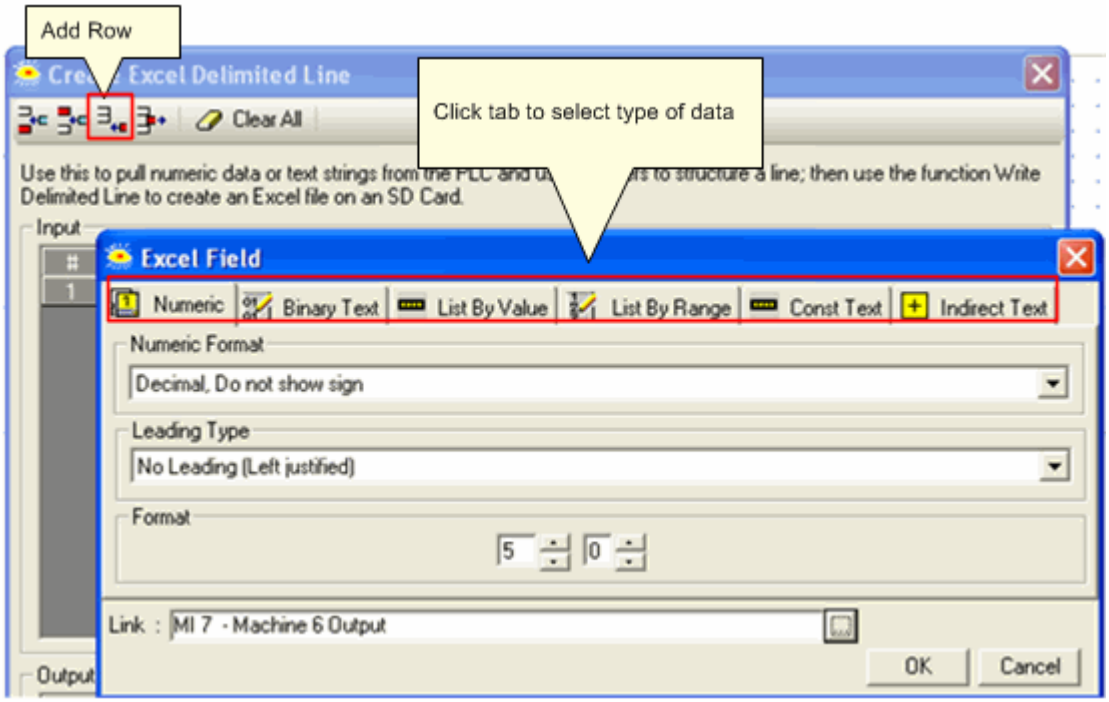
Use this function to select PLC data, including numeric data and text strings, structure it, and save the resulting line to a vector of operands.



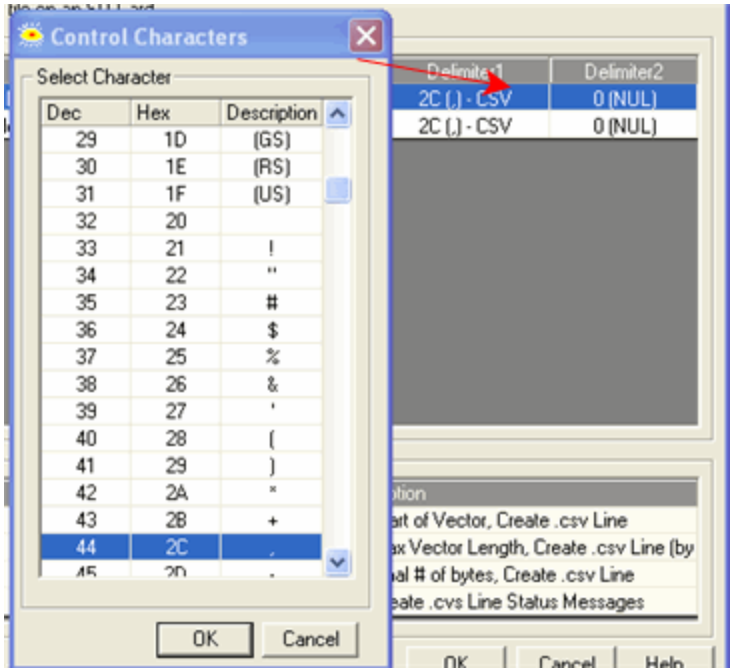
Defining a Line

Each row in the table displayed under Input will be a cell in the Excel line.

1. Click the Add Row icon to open the Excel Field dialog box.
2. Click a tab to select the type of data.



3. Click the Delimiter cells to select a Delimiter character that is different from the default.



4. Add and delete row by using the icons at the top of the function

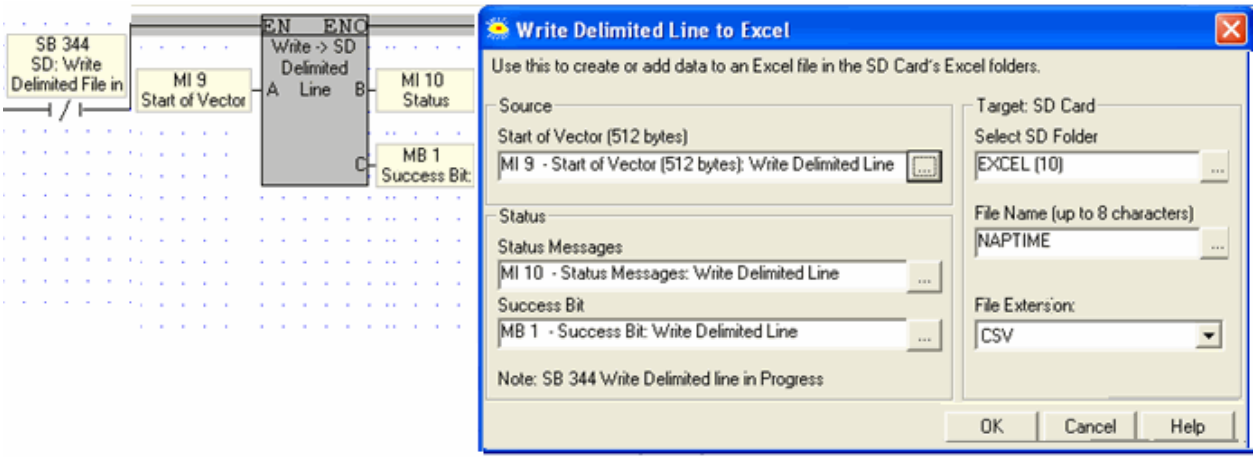
Type	Parameter	Purpose
Source: Define Data	Field	Use this to specify data for a cell in the Excel line.
	Delimiters	Control characters that delimit the data for that cell
Target: Data Buffer	SD: Start of Vector, Create .csv Line	Select the operand that will be start of the vector the function uses to store the data selected for the line, plus its delimiters. Use this operand for the function Write Delimited Line
	SD: Max Vector Length, Create .csv Line (bytes)	Sets the maximum length of the vector in bytes.
	SD: Final # of bytes, Create .csv Line	Reports the actual number of bytes sent to the vector
Status	SD: Create .csv Line Status Messages	This is a bitmap; a bit turns ON to indicate status. It is initialized when the function starts. • Bit 1 - The line is truncated • Bit 2 - Fail to open the file. • Bit 3 - Fail to write the file • Bit 4 - SD full • Bit 5 - No SD card (SB [217]) • Bit 6 - Path not found • Bit 7 - Unknown error - please check SI 66

Write Delimited Line

Use Write Delimited Line to pull the data from the vector used by Create Delimited, and use it to write to (or create) an Excel file in this folder, or in one of four sub-folders.

Notes

- Write Delimited Line **pulls data from the vector in chunks of 512 bytes, and writes this entire 512 bytes to the SD card.** Write Delimited line is not linked in any way to Create Delimited Line. In Create Delimited Line, the parameters SD: Max Vector Length and SD: Final # of bytes, do not influence Write Delimited line.
- Use SB 344,Write delimited line to SD in Progress, as a condition to running the function.



Type	Parameter	Purpose
Source	Start of Vector	Use the operand that is the Start of Vector for the Create Delimited Line function.
Target	Select SD folder	This is where the line will be stored on the SD card. You can select the folder, or provide the Folder number via register. Values point to folders as follows: 10=the main Excel folder, 1000= Excel1, 1001= Excel2, 1002= Excel3, and 1003= Excel4.
	File Name	Either enter a name, or link an operand to provide a file name. Note that you MUST provide a file name. If the linked register is empty, the file will not be created to the SD. If the folder does not contain a file of that name, the function will create one.
	File Extension	Select .txt or csv
Status	SD: Write .csv Line Status Messages	This is a bitmap; a bit turns ON to indicate status. It is initialized when the function starts. • Bit 1 - Wrong data

		• Bit 2 - Fail to open the file. • Bit 3 - Fail to write the file • Bit 4 - SD full • Bit 5 - No SD card (SB [217]) • Bit 6 - Path not found • Bit 7 - Unknown error - please check SI 66
	Success Bit	Turns ON when line is successfully written

SD Block Functions

SD Data Blocks are data storage files in the SdBLocks folder on a SD card.

SD Data Blocks may reach a total of 4G, or a single Block may be up to 4G. A Data Block comprises Sub-Blocks of 512 Bytes. The SD Block functions enable you to read/write blocks of raw data between operands and these files.

SD Data Block Functions

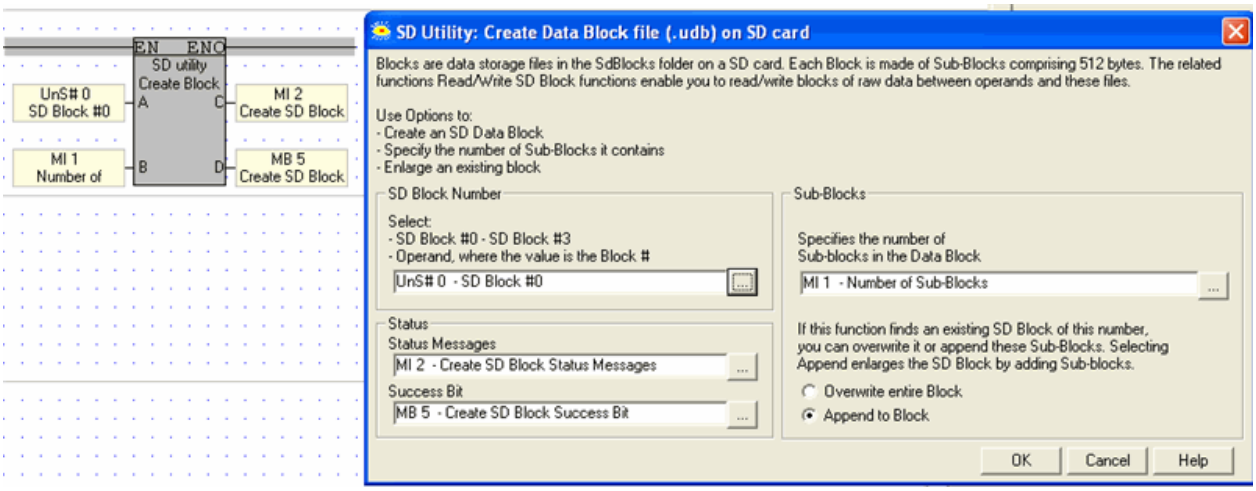
- Create SD Block
  Creates an SD Data Block in the SdBLocks folder.
- Read from SD Block to Vector
  Reads a specified Sub-Block from a specified Data Block to an operand vector that is 512 bytes long.
- Write from Vector to SD Block
  Writes 512 bytes from an operand vector to a specified Sub-Block in a Data Block

Create SD Block

Use this function to:

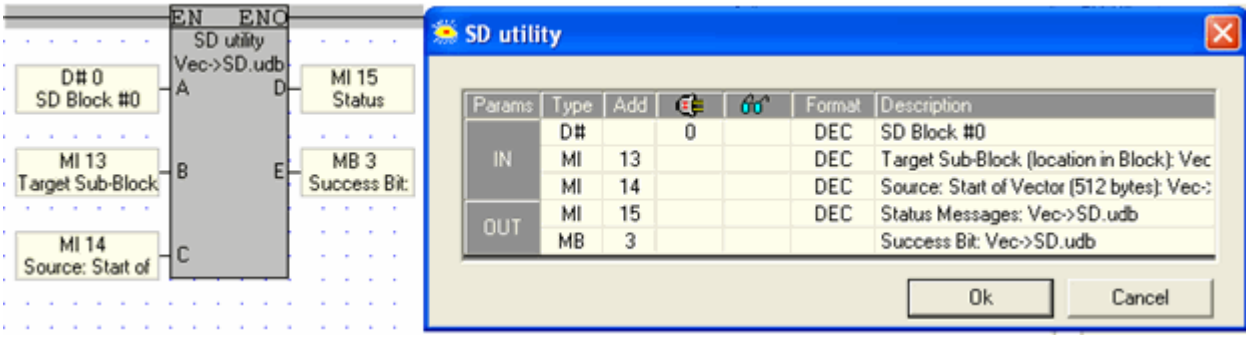
- Create an SD Data Block. You can create up to 4 SD Data Block: Block0.udb, Block1.udb, Block2.udb, and Block3.udb
- Specify the number of Sub-Blocks it contains.
- Enlarge an existing block.

You can also specify the number of Sub-Blocks the block will contain. This may be used to enlarge the Block by appending Sub-Blocks.



Parameter	Purpose
SD Block number	Select the Block number, 0-3, or use an operand to assign a number at run time.
Sub-Blocks	Use this to specify the number of Sub-blocks in the Data Block.
Overwrite/Append	If this function finds an existing SD Block of this number, you can overwrite it or append these Sub-Blocks. Selecting Append enlarges the SD Block by adding Sub-blocks.
Status Operand	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • All bits OFF – No errors, and the SD card is idle • Bit 1 – The SD card is busy. • Bit 2 – No SD card found, or the card is locked (Write-enable OFF) • Bit 6 - Internal error
Success Bit	Turns ON when the Block is created. It remains ON until it is reset by the application, or until the application calls the function.

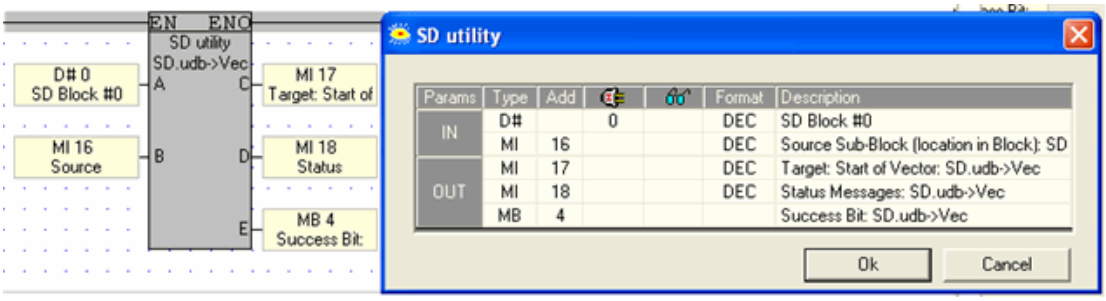
Read from Vector to SD Block



Parameter	Purpose
SD Block number	Select the Block number, 0-3, or use an operand to assign a number at run time.
Target Sub-Block (location in Block): Vec->SD.udb	The data will be written to this sub block. The number is the sequential number of the sub-block in the .udb file.
Source: Start of Vector	Select the operand that is the start of the 512-byte long vector that provides the data that is written to the .udb file.

(512 bytes): Vec->SD	
Status Messages	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • All bits OFF – No errors, and the SD card is idle • Bit 1 – The SD card is busy. • Bit 2 – No SD card found, or the card is locked (Write-enable OFF) • Bit 3 .-.There are less than 512 bytes in this vector (can happen if the start of the vector is too close to the end of the operand address range) • Bit 4 – The SD Data Block number is invalid (valid numbers are 0-3. This error may result when using indirect addressing) • Bit 5 - SD card function was called while the SD is busy • Bit 6 - Internal error • Bit 7 - Data Block size exceeds 4G • Bit 8 - SD card is full • Bits 10-13 can occur because the SD card does not have an SD_Blocks folder, or because a file of that name has not been created in the SD_Blocks folder. • Bit 10 - Can't open file/path not found • Bit 11 - Error while writing to a file/path not found • Bit 13 - Failed to close a file/path not found • Bit 14 - Create SD Block: Do not overwrite is selected, but the number of Sub-blocks is less than the number of sub-blocks already in the Data Block
Success Bit	Turns ON when the data is successfully written to the Block. It remains ON until it is reset by the application, or until the application calls the function.

Write from SD Block to Vector



Parameter	Purpose
SD Block number	Select the Block number, 0-3, or use an operand to assign a number at run time.
Source Sub-Block (location in Block): SD.udb-> Vec	This is the sequential number of the sub-block in the .udb file.

Target: Start of Vector: SD.udb-> Vec	The function will write 512 bytes of data to the PLC, starting with this operand.
Status Messages	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • All bits OFF – No errors, and the SD card is idle • Bit 1 – The SD card is busy. • Bit 2 – No SD card found, or the card is locked (Write-enable OFF) • Bit 3 .-.There are less than 512 bytes in this vector (can happen if the start of the vector is too close to the end of the operand address range) • Bit 4 – The SD Data Block number is invalid (valid numbers are 0-3. This error may result when using indirect addressing) • Bit 5 - SD card function was called while the SD is busy • Bit 6 - Internal error • Bit 8 - SD card is full • Bit 9 - Read: End Of File indication Bits 10-13 can occur because the SD card does not have an SD_Blocks folder, or because a file of that name has not been created in the SD_Blocks folder. • Bit 10 - Can't open file/path not found • Bit 12 - Error while reading from a file/path not found • Bit 13 - Failed to close a file/path not found
Success Bit	Turns ON when the data is successfully written to the Block. It remains ON until it is reset by the application, or until the application calls the function.

SD File Functions

You can use Windows Explorer to store any type of file onto an SD card, such as .html or .jpg. The SD File Functions enable your Ladder application to read and write these files in 'chunks' of 512 bytes.

Note • These functions can only run on files that observe the 8.3 naming convention; the file name cannot exceed 8 characters, and the file extension cannot exceed three.

Using SD File Functions

Each read or write operation requires three functions: Open File, Read or Write, and Close File. The functions are located on SD>SD File Utilities.

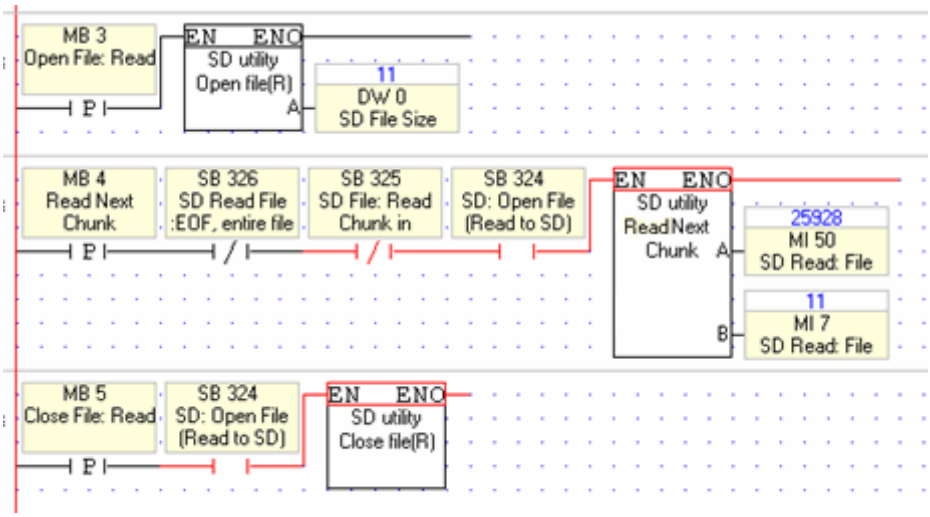
- Open SD File
  Note that there are separate Open File functions for both Read and Write. Open File must be used to activate the correct file in the correct folder before running a read or write function.
- Read Next Chunk
  Reads a specified file from a specified folder to an operand vector in 'chunks' that are 512 bytes long.
- Write Next Chunk
  Writes data chunks 512 bytes from an operand vector to a specified file on the SD card.

- Close File.
  There are separate Close File functions for both Read and Write.

The examples below show the functions and the System operands required to run read and write operations.

Read File: Example

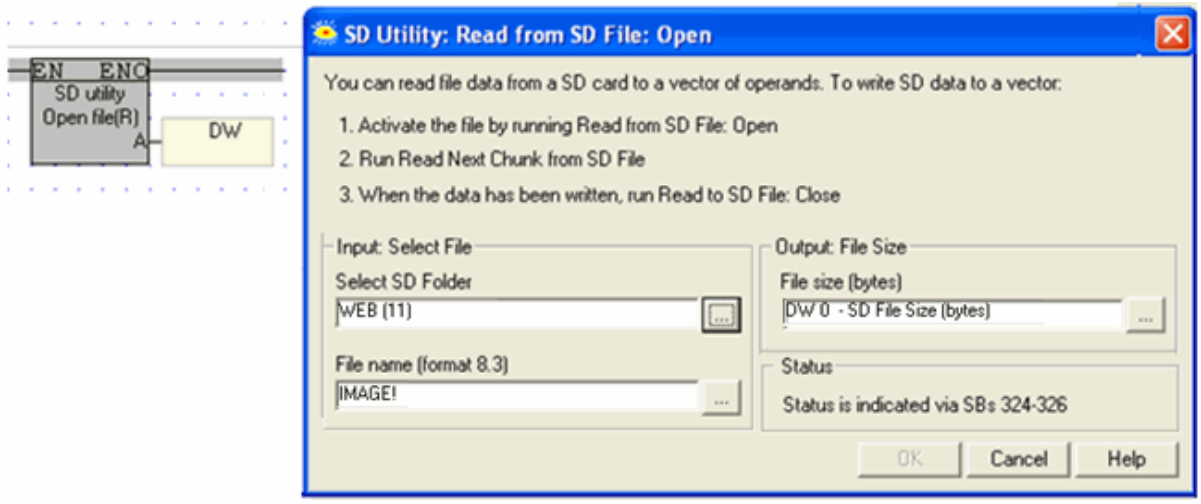
Note the use of SBs 324, 325, and 326. These enable the Read Next Chunk function to continue reading data chunks until it has completed reading the entire file.



Write File: Example

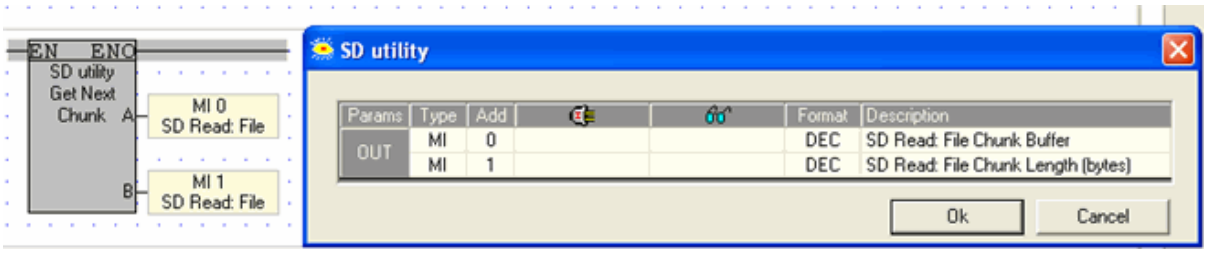
Functions

Read From SD File: Open



Parameter	Purpose
Select Folder	Select the folder, or use an operand to assign a number at run time using the following values: Alarms: 0 • DT main folder: 1 • DT1–4: 101 to 103 • Log: 3 • System: 4 • User_app: 5 • Trends main folder - 600 • Trends1-4: 600 to 603 • SdBLocks: 9 • Excel main folder: 100 • Excel1-4: 1000 to 1003 • Web: 11
File name	Either enter the file name, or provide it via operand.
File size	When the function runs, this reports the size of the file, in bytes.

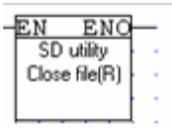
Read File: Get Next Chunk



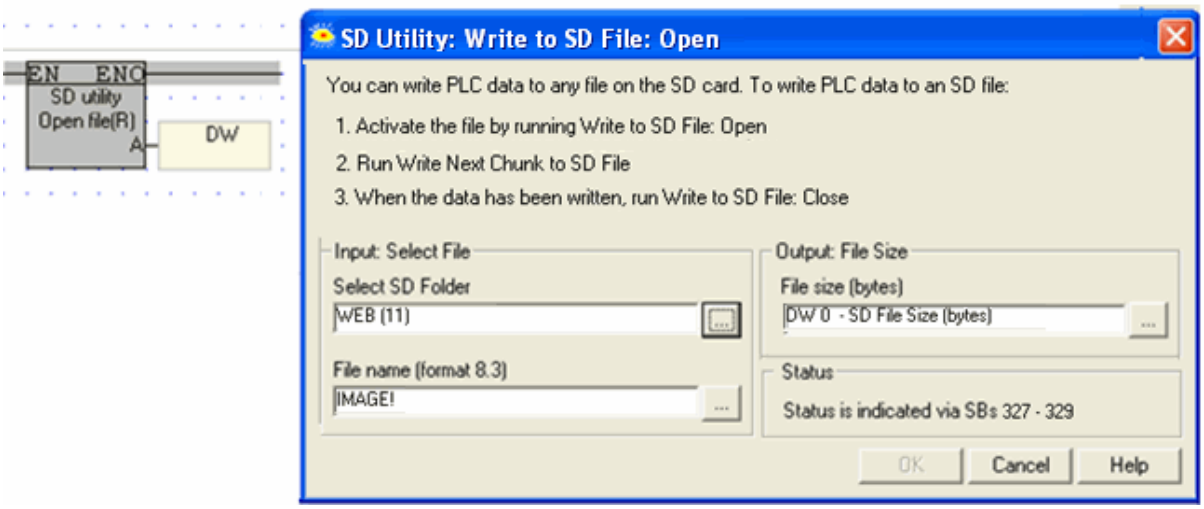
Parameter	Purpose
Read: File Chunk Buffer	This the start of the vector that holds the data read from the file. This vector is 512 bytes long.
Read: File Chunk Length	This shows the length of the chunk that is currently read. Note that the final chunk, containing the last of the file data, will generally be LESS than 512 bytes.

Read File: Close

Run this when entire files has been read.

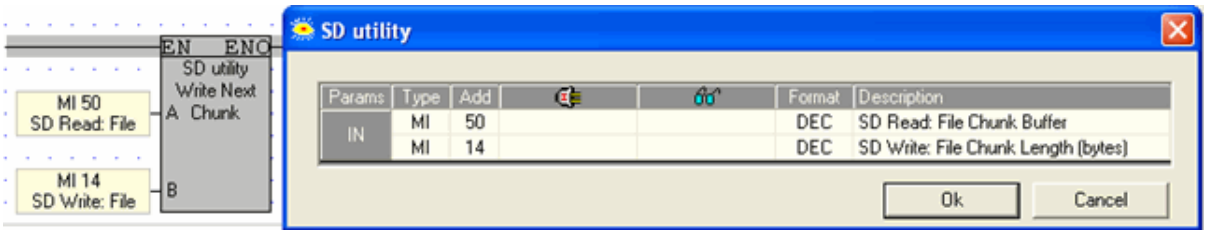


Write From SD File: Open



Parameter	Purpose
Select Folder	Select the folder, or use an operand to assign a number at run time using the following values: Alarms: 0 • DT main folder: 1 • DT1–4: 101 to 103 • Log: 3 • System: 4 • User_app: 5 • Trends main folder - 600 • Trends1-4: 600 to 603 • SdBLocks: 9 • Excel main folder: 100 • Excel1-4: 1000 to 1003 • Web: 11
File name	Either enter the file name, or provide it via operand. Note that: -If the file does not exist on the SD -and the SD card is Write-enabled the function will create the file.
File size	When the function runs, this reports the size of the file, in bytes.

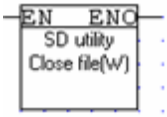
Write File: Get Next Chunk



Parameter	Purpose
Write: File Chunk Buffer	This the start of the vector that holds the data that will be written to the file. The function takes 512 bytes of data.
Write: File Chunk Length	Enter the number of bytes to be written to the SD file.

Write File: Close

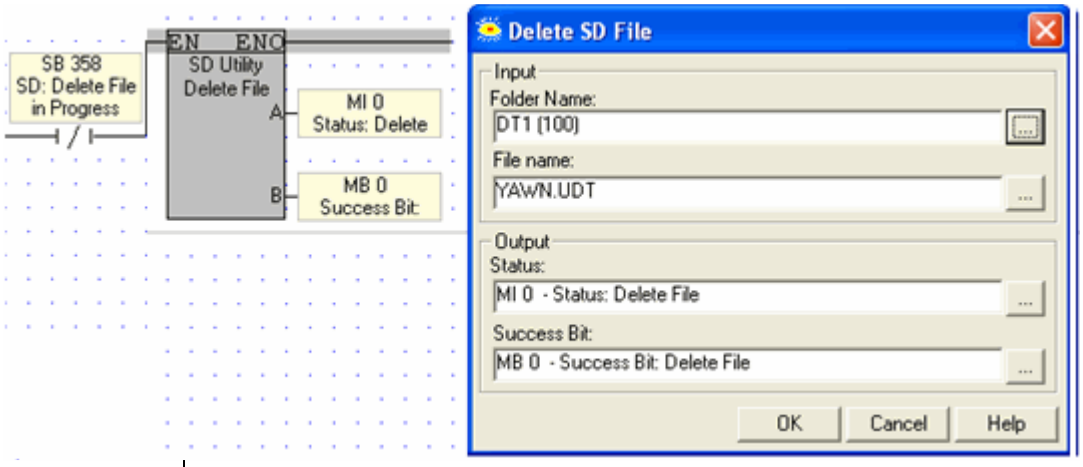
Run this when entire files has been written.



Delete File

Use this to delete any file on the SD card

Note •	These functions can only run on files that observe the 8.3 naming convention; the file name cannot exceed 8 characters, and the file extension cannot exceed three.
•	Use SB 358, Delete File in Progress, as a condition to running the function



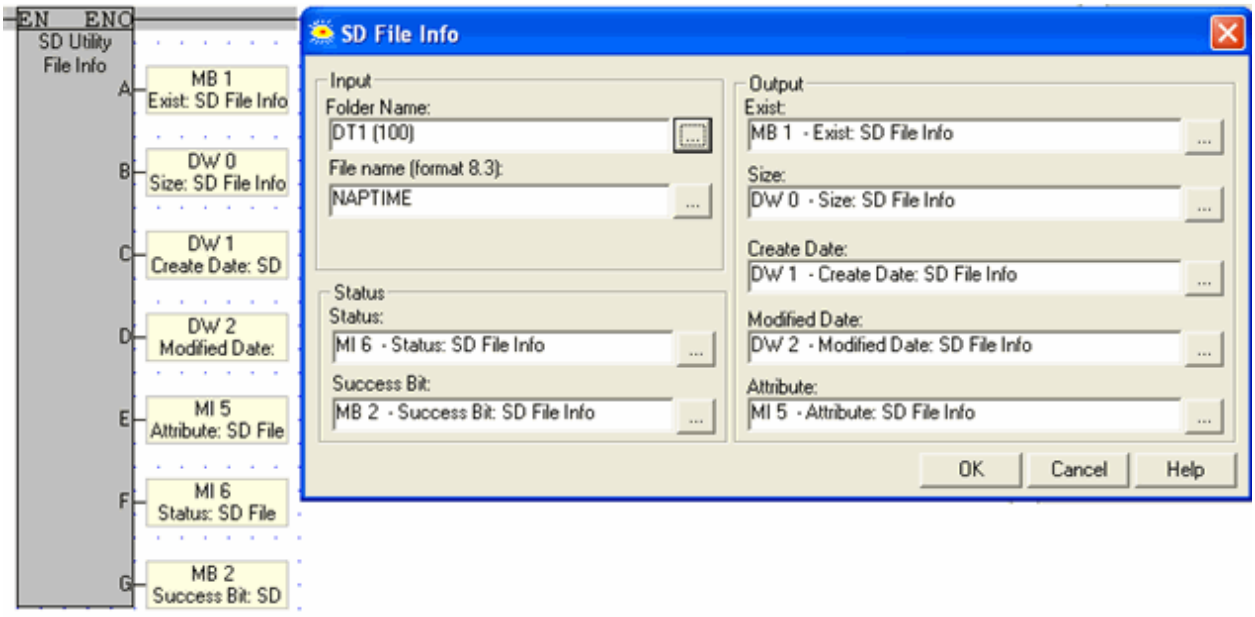
Parameter	Purpose
Select Folder	Select the folder, or use an operand to assign a number at run time using the following values: Alarms: 0 • DT main folder: 1 • DT1–4: 101 to 103 • Log: 3 • System: 4 • User_app: 5 • Trends main folder - 600 • Trends1-4: 600 to 603 • SdBLocks: 9 • Excel main folder: 100 • Excel1-4: 1000 to 1003 • Web: 11
File name	Either enter the file name, or provide it via operand. Note that: -If the file does not exist on the SD

	-and the SD card is Write-enabled the function will create the file.
Status Messages	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • All bits OFF – No errors, and the SD card is idle • Bit 1 – SD Card internal error. • Bit 2 – Delete Failed • Bit 3 - No SD card found, or the card is locked (Write-enable OFF) • Bit 5– Path not found • Bit 7 - The SD card has failed (Check SI 66)
Success Bit	Turns ON when the data is successfully written to the Block. It remains ON until it is reset by the application, or until the application calls the function.

SD File Information

Use this function to check if a specific file is located in a specific SD folder, and get specific file details.

Note • Use SB 359, File Info function in Progress, as a condition to running the function

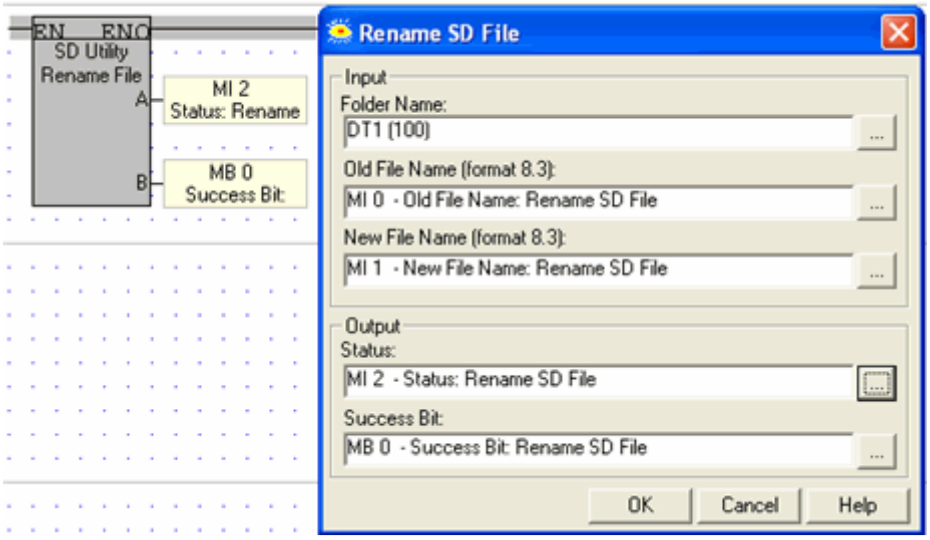


Parameter	Purpose
Select Folder	Select the folder, or use an operand to assign a number at run time using the following values: Alarms: 0 • DT main folder: 1 • DT1–4: 101 to 103 • Log: 3 • System: 4 • User_app: 5 • Trends main folder - 600 • Trends1-4: 600 to 603 • SdBLocks: 9 • Excel main folder: 100 • Excel1-4: 1000 to 1003 • Web: 11
File name	Either enter the file name, or provide it via operand.
Status Messages	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts.

	• All bits OFF – No errors, and the SD card is idle • Bit 1 – SD Card internal error. • Bit 2 – Cannot read file • Bit 3 - No SD card found, or the card is locked (Write-enable OFF) • Bit 4– The SD card has failed (Check SI 66) • Bit 5 - Path not found
Success Bit	Turns ON when the data is successfully written to the Block. It remains ON until it is reset by the application, or until the application calls the function.

Rename SD File

Use this to rename any file on the SD card.



System Operands

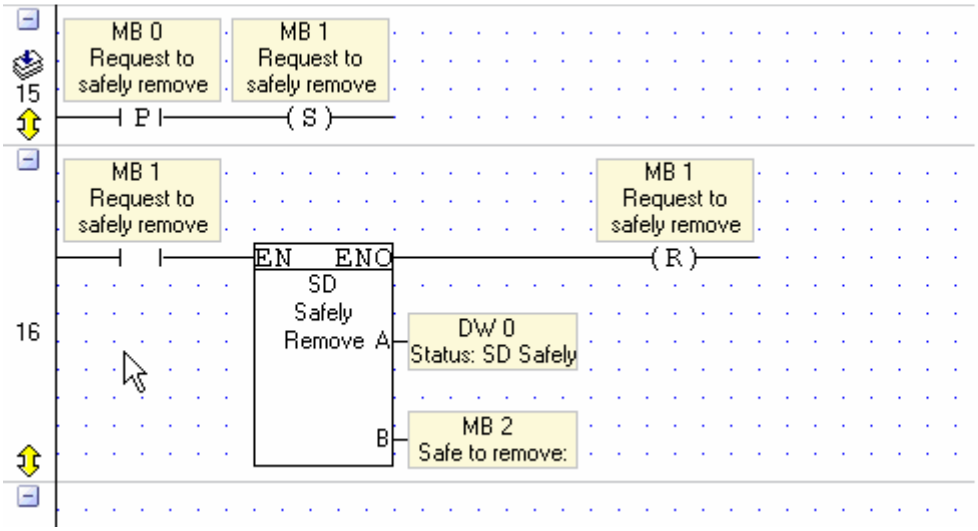
SD: Safely Remove

Use SD: Safely Remove to prevent the card from being physically removed while an SD function is in progress.

When SD: Safely Remove is called, it:

- Checks to see if any SD functions are running.
- If so, Safe to Remove indicates which function is active via the Status DW.
- It allows a current task to be completed, but prevents new ones from starting.

When the SD card is completely free, the Safe to Remove bit turns ON. This must be reset by the user.



The Status DW is a bitmap.
Bits and their indications are shown in the following table. When a bit is ON, the related function is active.

Bit	Function
0-5 (reserved)	
6	Start Saving to SD is running: SD Trend 8
7	Start Saving to SD is running: SD Trend 7
8	Start Saving to SD is running: SD Trend 6
9	Start Saving to SD is running: SD Trend 5
10	Start Saving to SD is running: SD Trend 4
11	Start Saving to SD is running: SD Trend 3
12	Start Saving to SD is running: SD Trend 2
13	Start Saving to SD is running:SD Trend 1
14	HMI function Trend from SD
15	SD File Info
16	Delete SD File
17	Folder Report: Number of Files
18	Create Excel Delimited Line
19	SD File Utilities: SD File Write
20	SD File Utilities: SD File Read
21	SD Block Utilities: Read/Write to Block 3
22	SD Block Utilities: Read/Write to Block 2
23	SD Block Utilities: Read/Write to Block 1
24	SD Block Utilities: Read/Write to Block 0
25	HMI Variable SD Browser
26	PC Utility is communicating with SD
27	Information Mode is accessing SD
28	SD Data Table Utilities: Log DT Row
29	SD Data Table Utilities: Read from DT /Search DT for Tag or Index
30	SD Data Table Utilities: Write to DT
31	Alarm History is being logged to SD

SD: Cloning via Ladder

SD Clone functions can:

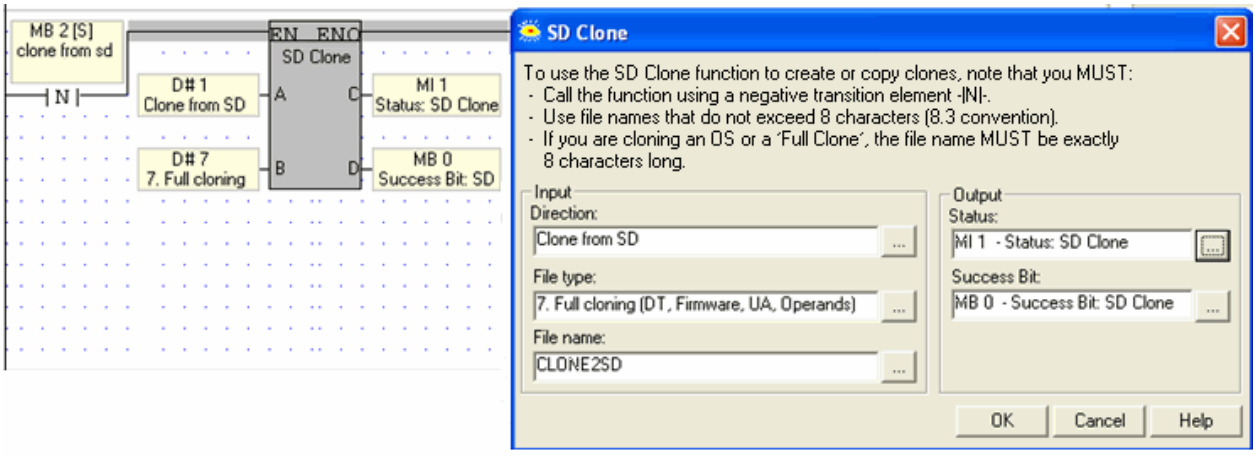
- Create compressed data files and store them on an SD card.
- Upload compressed files from an SD card to a PLC.

You can 'clone' a complete PLC or data using the SD Ladder functions

These Ladder functions are parallel to the actions you can carry out via Information Mode.

Notes •	The SD Card password and the Clone File password must be **identical**.
•	You can use Unitronics' SD Card Explorer, included in the SD Card Suite, to access SD card files and either upload them to a PC for viewing and editing, or transfer them into another PLC's SD card.

This function must be used with a negative transition element.

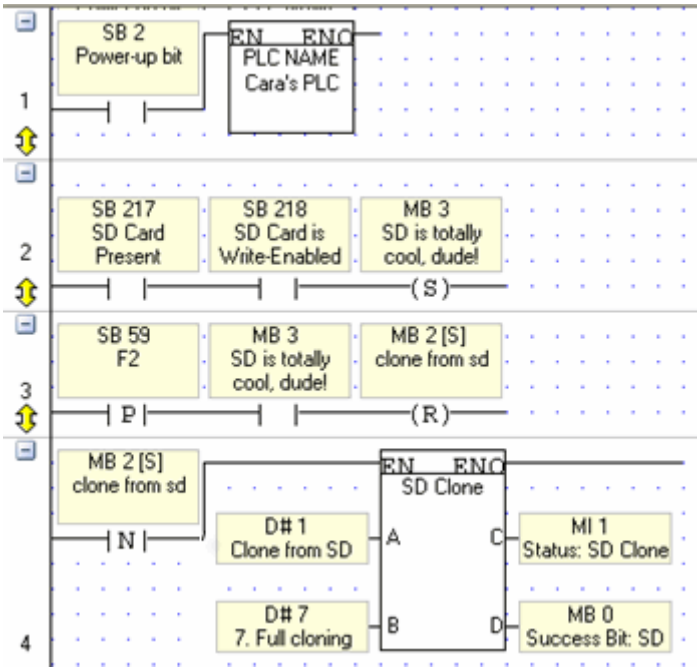


Parameter	Purpose
Direction	Clone To SD: Creates a compressed data file in the correct SD card folder Clone From SD: Installs a compressed data file from an SD card folder into the PLC
File Type	Select Direct or Constant. If you select Direct, the value in the register determines the data file that the function creates/installs according to the following legend: 2 = Full Data Table (*.fdt files) 4 = Firmware (*.Oxx files: .O13, .O35, or .O57) 5 = User Application (+VLP if exists) (*.vxx files: .V13, .V35, .V57) 7 - Full cloning (DT, Firmware, UA, Operands)(*.Cxx) .C13, .C35, or C57) 8 = Operands (*.Dxx files: .D13, .D35, .D57) Note that the file extension numbers relate to the Vision model: .x13 =V130, .x35 =V350, .x57 =V570
File Name	This is limited to 8 characters. The file extension is automatically assigned by the PLC according to the file type. If you are cloning an OS or a 'Full Clone', the file name MUST be exactly 8 characters long.
Status	This MI is a bitmap; a bit turns ON to indicate status. The MI is initialized when the function starts. • All bits OFF – No errors, and the process is idle • Bit 1 – No SD card found, or the card is locked (Write-enable OFF) • Bit 2 – Clone utility busy • Bit 3 -File type not found (*.FDT,*.Oxx,*.Vxx,*.Dxx,*.Cxx) • Bit 4– Incompatible Boot Version/Firmware/Clone file • Bit 5 - Internal use • Bit 6- Timeout exceeded • Bit 7 - - Safe to Remove Bit is ON • Bit 8 - Path not found (Install Clone) • Bit 9 - Password error

Success Bit	Turns ON when the data is successfully written to the SD. It remains ON until it is reset by the application, or until the application calls the function.

SB 366: Clone in Progress. Note that the process can take from several seconds to several minutes.

The following nets show the conditions required to run the function.



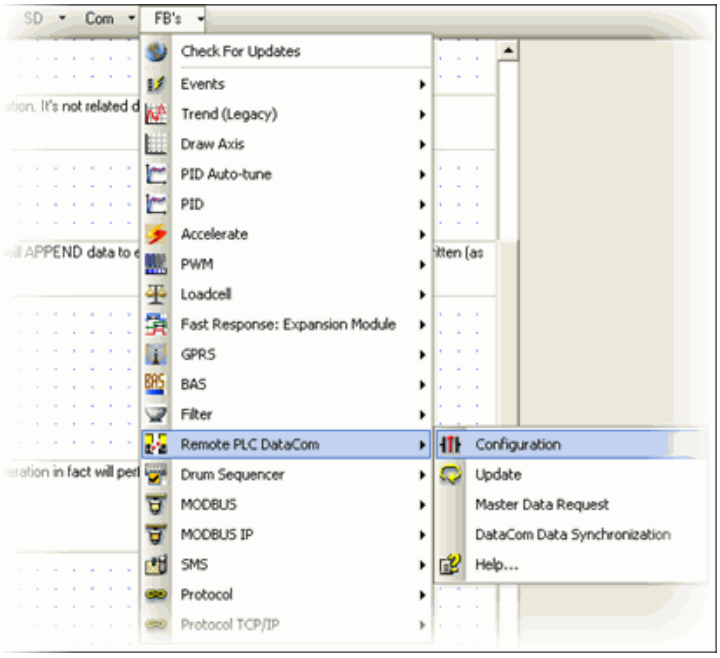
COM Functions

For information regarding COM functions, please refer to the VisiLogic – Communications manual.

FBs Library

When you install VisiLogic, the program also installs a Function Block (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.

For specific information, refer to the manual VisiLogic – Function Blocks.

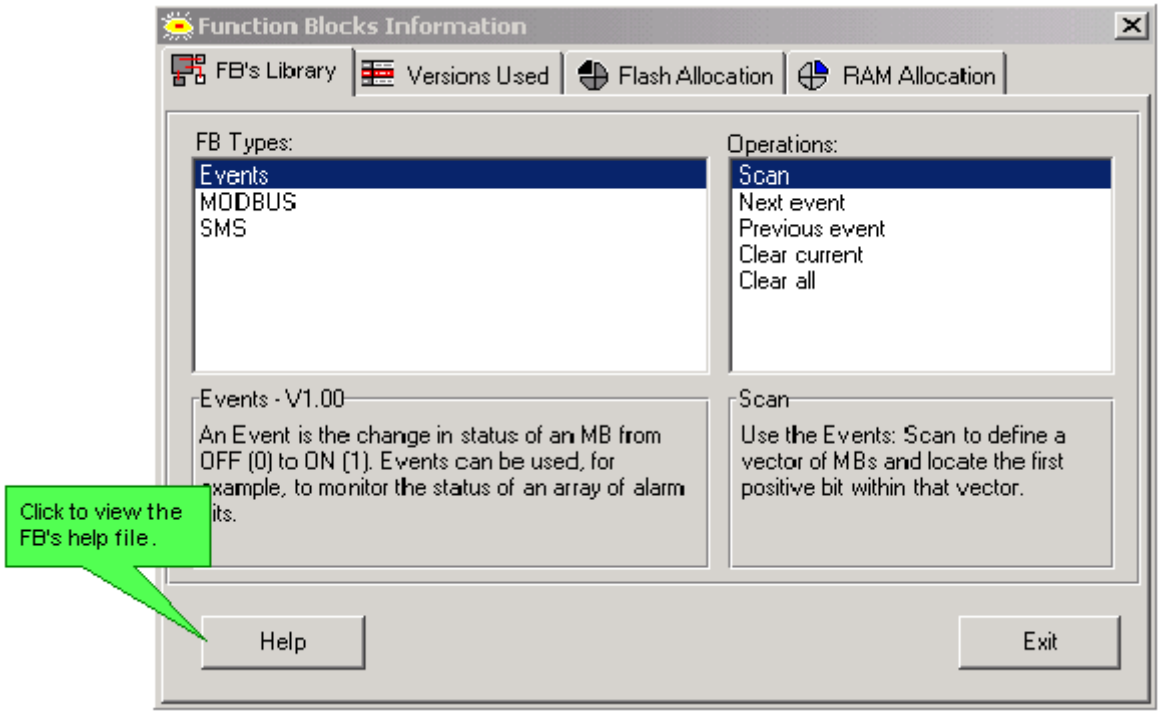


Note • You must use a condition (RLO) to activate any FB that requires Configuration in your application, such as MODBUS or SMS.

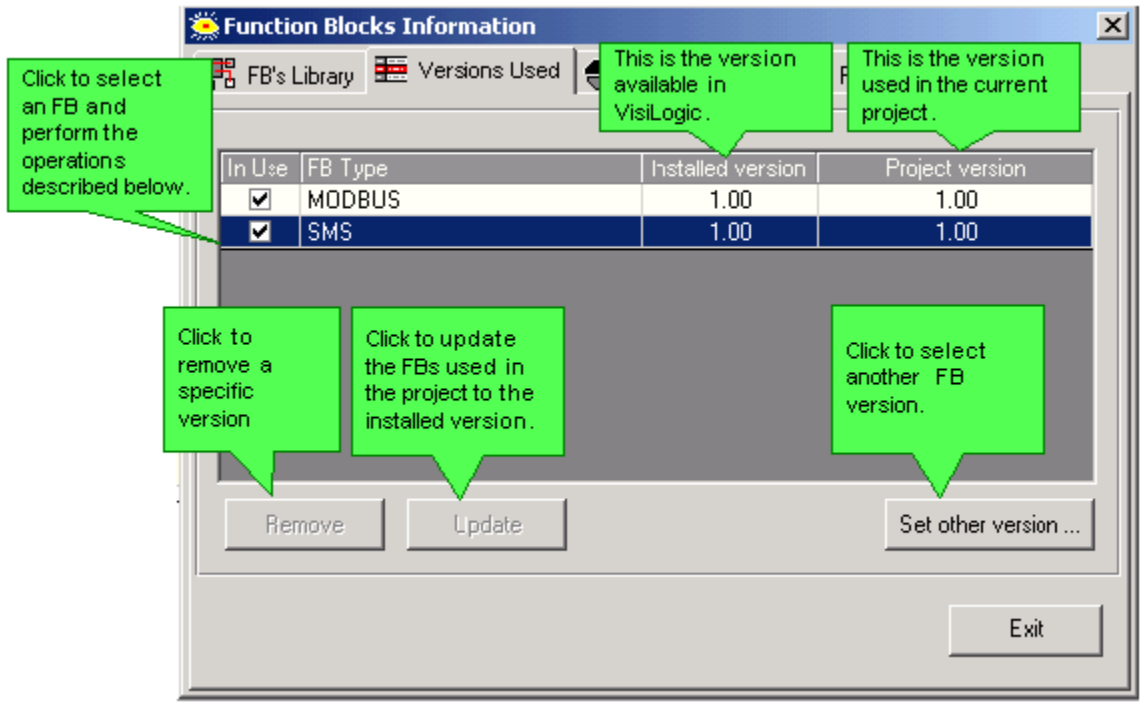
Note • To enable Live Update, select to use a proxy server in Project Properties.

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.


Function Blocks Information
FB's Library
Versions Used
Flash Allocation
RAM Allocation
FB Types:
Events
MODBUS
SMS
Operations:
Scan
Next event
Previous event
Clear current
Clear all
Events - V1.00
An Event is the change in status of an MB from OFF (0) to ON (1). Events can be used, for example, to monitor the status of an array of alarm bits.
Scan
Use the Events: Scan to define a vector of MBs and locate the first positive bit within that vector.
Click to view the FB's help file.
Help
Exit

Versions Used



Updating FB versions

Standard Vision: 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.

Enhanced Vision: FB libraries are updated as part of OS releases. When you update the OS, FBs are automatically updated as well.

FBs List

MODBUS, serial

MODBUS, IP

SMS Messaging

GPRS

Remote PLC DataCom

Communication Protocol

TCP/IP Communication Protocol

PID FB

Drum

Events

MB as PWM

Loadcell

Filter

Accelerate

Fast Response

Draw Axis

BAS

Trends

If your project is configured to Vision controllers that do not support HMI object Trend graphs, the Trend objects will not be displayed in the Project Navigation Window. These controllers include V120/230/260/280/290 (monochrome). In these models, the Trends (Legacy) Function Block may be used.

Trends (Legacy) Function Block

Index