U90Ladder Special Functions via SIs& SBs

5/24/2004

Table Of Contents

Special Functions: without Elements 1
  Functions activated by SI 140 4
    A*B/C 4
    Communication Utilities 5
    Copy Vector 10
    Fill Vector 11
    Find Mean, Maximum, and Minimum Values 13
    GSM PIN Code via MI 14
    Interrupt 15
    Load Indirect 16
    Load Timer Preset/Current Value 17
    MODBUS 18
    SMS Phone Number: via MI Pointer 25
    Store Timer's Preset/Current Value 27
    Square Root 28
  Functions activated by SBs 29
    Convert MB to MI, MI to MB 29
    Copy MI to Output vector, Input vector to MI 30
    Database: Access indirectly addressed registers 32
    Deleting SMS messages 35
    Immediate: Read Inputs & HSC, Set/Reset Outputs 35
    'Long' Integer functions 36
    Linearization 38
    Shift Register 41
    Display Integer values as ASCII or Hexadecimal 42
Index 45

Special Functions: without Elements

U90Ladder contains special functions that are not represented by Ladder Elements. You can perform these functions by storing values into the System Integers listed here.

To implement a special function, first store the parameters of the function in the relevant SI function operands, as described in the Help topic of the function, then store the command number into SI 140.

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Click on the function name in the following tables to view a topic containing specific instructions on how to implement the function.

SI	Description
140	Function Number
141	Function Operand #1
142	Function Operand #2
143	Function Operand #3
144	Function Operand #4
145	Function Operand #5
146	Function Operand #6

Functions activated by SI 140

Function Name	Description	Parameters	Execute Function, Store into SI140
A*B/C	Enables PLC to multiply 2 operand values & divide the product by a third operand.	➤ SI 141 Operand A (multiplicand). ➤ SI 142 Operand B (multiplicand), ➤ SI 143 Operand C (divisor).	➤ 100
Communication Utility	Enables PLC to receive data from external devices, such as bar-code readers, via an RS232 port.	➤ SI 141 STX ➤ SI 142 ETX ➤ SI 143 ETX Length or Silent ➤ SI 144 Maximum Length ➤ SI 145 Start Address: Receive Buffer ➤ SI 60 # of Bytes currently in Receive Buffer ➤ SI 61 # of Bytes in Receive Buffer when SB 60=1 ➤ SI 146 Copy Data: Format ➤ ➤ SB 60 Data Successfully Received	➤ 300 ➤ Additional Functions: ➤ Set SB 61 to Copy Data in Receive Buffer to Vector ➤ Set SB 62 to Clear Receive Buffer, Clear SI 60, Clear SI 61,& Reset SB 60
Copy Vector	Set a vector, copy source values, then write those values into a corresponding target vector.	➤ SI 141 Source Vector ➤ SI 142 Vector Length ➤ SI 143 Target Vector	➤ Copy MIs to MIs: 20 ➤ Copy MIs to DBs: 21 ➤ Copy DBs to MIs: 22 ➤ Copy DB to DB: 23
Fill Vector	Copies a source value, then write that value into every operand within the target vector.	➤ SI 141 Start of Target vector, ➤ SI 142 Length of Target vector, ➤ SI 143 Fill Value; register whose value will be written into each register within the target vector	➤ Fill MI vector: 30 ➤ Fill DB vector: 31
Find Mean, Maximum, and Minimum Values	Find within vector: Mean, Minimum, & Maximum.	➤ SI 141 Start of vector, ➤ SI 142 Length of vector	➤ Find in MI vector: 40 ➤ Find in DB vector: 41

GSM PIN Code via MI	Uses an MI vector to supply a GSM modem PIN code	❍ SI 141 Start of vector	❍ 410
Interrupt	Causes program to stop immediately without regard to program scan	See Interrupt for details	❍ 500 ❍
Load Indirect	Takes value contained in a **source** operand and loads that value into a **target** operand using indirect addressing.	❍ SI 141 Data source ❍ SI 142 Load target	❍ Load MI to MI: 10 ❍ Load SI to MI: 11 ❍ Load MI to SI: 12 ❍ Load SI to SI: 13 ❍
MODBUS	Enables MODBUS Master/Slave communications	See MODBUS for details	❍ Configure: 600 ❍ Read Coils: 601 ❍ Force Coil: 602 ❍ Force Coils: 603 ❍ Read Output Registers: 604 ❍ Preset Register: 605 ❍ Preset Registers:606 ❍ Read Output Registers in Float Format: 607 ❍ Preset Float Registers: 608 ❍ Read Input Registers: 609 ❍ Read Input Registers in Float Format: 610 ❍ Read Inputs: 611 ❍ Loopback Test: 612
SMS Phone Number: via MI Pointer	Uses an MI vector to supply a phone number in the SMS phone book	SI 141 Start address of the MI vector containing the phone number	❍ Store 400 into SI 140
Square Root	Finds the square root of a number	SI 141 Store the number	❍ Store 110 into SI 140

Functions activated by SBs

Function Name	Description	Parameters	Activating SB-SI
Convert MB to MI, MI to MB	Converts 16 bits or more into a integer value, or an integer value into 16 bits	❍ SI 170 Address of MI containing integer value ❍ SI 171 Start address of MB array (vector) ❍ SI 172 Amount of MBs	❍ Set SB 170 to activate MB to MI ❍ Set SB 171 to activate MI to MB
Copy MI to Output vector, Input vector to MI	❍ Copy a vector of Inputs (**I**) to a register. ❍ Copy a register value to a vector of Outputs (**O**)	❍ SI 170 Address of MI containing integer value ❍ SI 171 Start address of bit array (vector) ❍ SI 172 Amount of bits	❍ Set SB 170 to activate I to MI ❍ Set SB 171 to activate MI to O

Database	The M90/91 has a special memory area containing integers that are function as a database.	Within the database, you can access and use integers 0 through 1023 via SI 40 and SI 41. See Using the Database for details.	
Immediate: Read Inputs & HSC, Set/Reset Outputs	Perform immediate actions, without regard to the program scan.	Model dependent; to learn what is relevant to a particular controller model, see Help topic Immediate: Read Inputs & HSC, Set/Reset Outputs.	
Long Integer Functions	❯ Uses adjacent MIs in performing calculations and storing results. ❯ M91 Only.		❯ Set SB 82 to treat 2 registers as 'long integer' ❯
Shift Register	Load SI 87 with a value, use SBs to shift register bits left/right	❯ SI 87 Contains the **number** to be shifted ❯ SI 88 contains the number of **bits** to be shifted (Default is 1 bit)	❯ Set SB 87 to shift left ❯ Set SB 88 to shift right

Functions activated by SI 140

A*B/C

This function enables you to :

- Multiply 2 operand values,
- Divide the product by a third operand.

The product of the multiplication operation is temporarily stored in a long integer to avoid overflow problems.

Since there is no Ladder element for this function; you perform it by storing values into:

- SI 141 to provide Operand A (multiplicand),
- SI 142 to provide Operand B (multiplicand),
- SI 143 to provide Operand C (divisor),

Store 100 into SI 140 to call the function. In your application, call the function **after** you have entered all of the other parameters.

The results will be placed in:

- SI 144,
- SI 4: Divide Remainder.

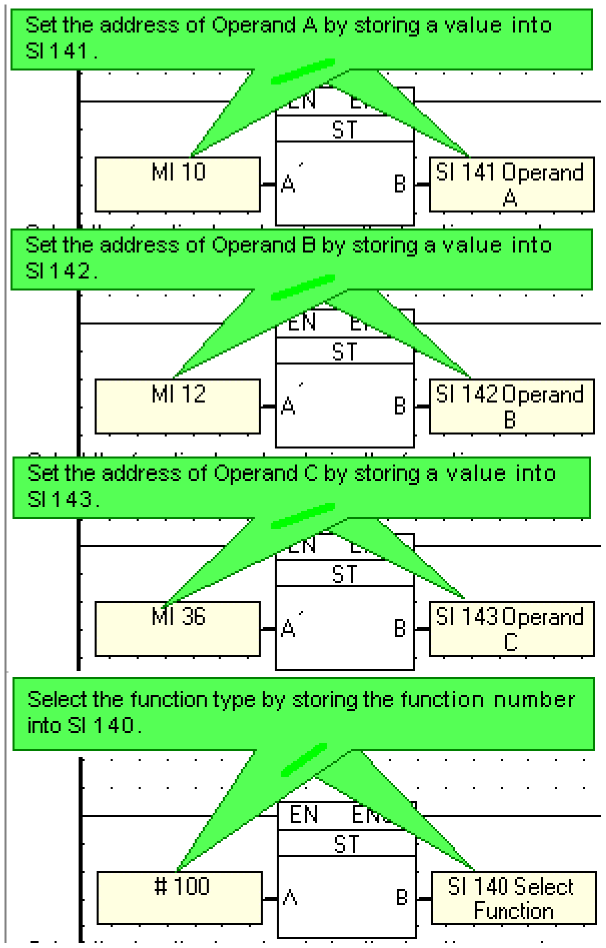
If the result is out of the integer range:

- SB 141 will turn ON.

If the value contained in Operand C (divisor) is 0:

- SB 4: Divide by 0, will turn ON.

To use this function:


Set the address of Operand A by storing a value into SI 141.
EN
ST
MI 10
A
B
SI 141 Operand A
Set the address of Operand B by storing a value into SI 142.
MI 12
SI 142 Operand B
Set the address of Operand C by storing a value into SI 143.
MI 36
SI 143 Operand C
Select the function type by storing the function number into SI 140.
# 100
SI 140 Select Function

Function Number (SI 140)	Description
100	Multiply A x B, Divide by C

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Communication Utilities

Use this utility to enable your controller to receive data from external devices, such as bar-code readers, via an RS232 port. Since there is no Ladder element for this function; you perform it by storing values into SIs.

Note that the communication settings stored into these SIs only take effect at power-up.

SI	Parameter	Value to Store	Notes
141	STX (Start of Text)	0-255(ASCII) -1: No Start of Text (not recommended)	The STX parameter indicates where the data block begins. • Note that the ASCII character '/' (backslash) cannot be used to indicate the start of the data block.
142	ETX (End of Text)	0-255(ASCII) -1: ETX marked by Length -2: ETX marked by 'Silence'	The ETX parameter indicates where the data block ends. When the ETX is registered by the function, SB 60 turns ON. • If you use an ASCII character (0-255), note that if this character occurs **after** the Length parameter defined in SI 143, SB 60 turns ON. • Selecting -1 causes the function to use the **length** of a data

			block alone to determine its end. • Selecting -2 causes the function to use the **duration** of silent time following the STX to determine the end of a data block.
143	ETX Length or Silent	Length: up to 128 Silent: up to 24000	• This defines both the length of text, or silence, that signal the end of text. • Note that the duration of a silent 'counter' unit is approximately 2.509 mS. The 'silent' value should be lower than the M90 TimeOut value. • When defined as length, SI 143 cannot exceed SI 144.
144	Maximum Length	Up to 128	• This is the maximum legal length for received text. • When the maximum length is exceeded, the Receive Buffer is automatically cleared, and SB 60 is turned OFF, enabling new data to be received. • This can be used to detect buffer overflow.
145	Start Address: Receive Buffer	MI Address	This MI contains the start address for the vector of registers that serves as the Receive Buffer.
60	Number of Bytes currently in Receive Buffer	Read only	SI 60 indicates how many bytes of data are currently in the Receive Buffer.
61	Number of Bytes in Receive Buffer when SB 60=1	Read only	SI 61 indicates how many bytes of data are in the Receive Buffer when SB 60 turns ON.
146	Copy Data: Format	0: copy each received byte 1: copy in groups of 4 received bytes.	• **0** causes each separate byte to be copied to a separate register including STX and ETX. • **1** causes every **4** bytes to be copied to a single register, without the STX and ETX. This is used when the received data is in numeric format. For example 12345 would be copied to 2 consecutive MIs. The first MI would contain 1234, the second would contain 5.
140	Start receiving	300	In your application, use this to call the function after you have entered all of the other parameters. Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

SB	Description	Notes
60	Data Successfully Received	Read only. Turns ON when the ETX condition is registered by the system.
61	Copy Data in Receive Buffer to MI Vector	Write only. • Turning this SB ON causes the buffer contents to be copied to the MI vector defined in SI 145. The data will

		be copied according to the format defined in SI 146. • If SI 146 is set to 0, this SB can be set at any time. If SI 146 is set to 1, this SB can be set after SB 60 turns ON.
62	Clear Receive Buffer, Clear SI 60, Clear SI 61, Reset SB 60	• This SB must be turned ON to enable a new message, or data block, to be received. • Turn this SB ON to enable data to be received **before** the maximum length, defined in SI 144, is exceeded.

Note that if no data is received for a period exceeding the M90 TimeOut, you will lose the data in the buffer.

To see how to use the Communications Utility, check the sample application **Read Card - Display Number Value.U90**. This may be found by accessing Sample U90 Projects from the Help menu.

This application demonstrates how to read a magnetic card number using an "IDTECH" card reader, then display that number on the M90's screen. The card reader transmits the number in ASCII characters in this format:

< %?[CR];xxxxx?[CR] > where **xxxxx** is the card number.

The ASCII character used to mark the Start Of Text (STX) is < **;** > (semicolon). End Of Text (ETX) is marked with the character < **?** > .

Since the card number is 5 digits long, the card number is copied to 2 separate MIs. The MIs are linked to 2 variables that are shown on the M90's screen in 2 separate Displays.

The parameters must be written into their respective operands using one scan condition. For this purpose, it is recommended to use SB 2 Power-up bit, as shown in the sample application.

ASCII character table

Value	Hex	Char
32	20	<SPACE>
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6

55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
94	5E	^
95	5F	_
96	60	<Degree sign>
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r

115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
124	7C	|
160	A0	א
161	A1	ב
162	A2	ג
163	A3	ד
164	A4	ה
165	A5	ו
166	A6	ז
167	A7	ח
168	A8	ט
169	A9	י
170	AA	ך
171	AB	כ
172	AC	ל
173	AD	ם
174	AE	מ
175	AF	ן
176	B0	נ
177	B1	ס
178	B2	ע
179	B3	ף
180	B4	פ
181	B5	ץ
182	B6	צ
183	B7	ק
184	B8	ר
185	B9	ש
186	BA	ת
255	FF	<Black box>

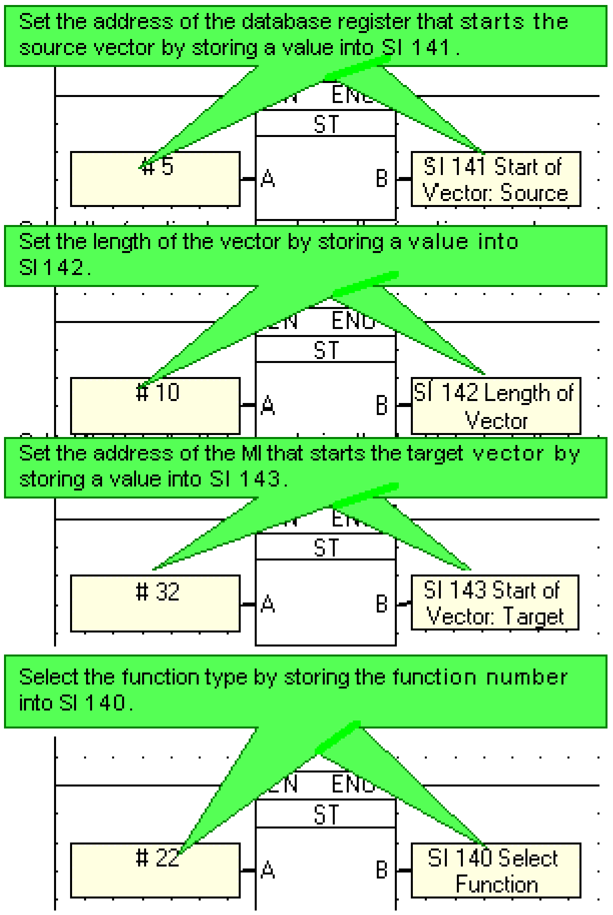
Copy Vector

Vector Copy enables you to set a range of operands, copy the values of each operand within that range **(source)**, then write those values into a corresponding range of operands of the same length **(target)**. You can copy from/to a vector of MI registers or Database registers by selecting the appropriate function.

Note that since there is no Ladder element for this function; you perform it by storing values into:

- SI 141 to determine the source vector,
- SI 142 to determine the length of the vector,
- SI 143 to determine the target vector,
- SI 140 to select the type of function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters.

To use Copy Vector:



Set the address of the database register that starts the source vector by storing a value into SI 141.

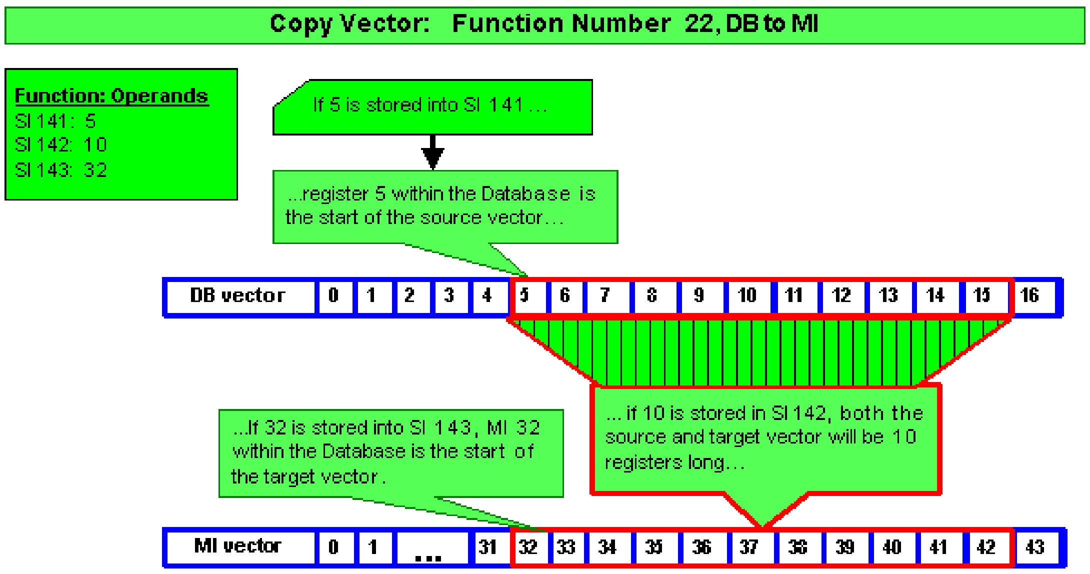
Set the length of the vector by storing a value into SI 142.

Set the address of the MI that starts the target vector by storing a value into SI 143.

Select the function type by storing the function number into SI 140.

Function Number (SI 140)	Source Vector, (SI 141)	Target Vector, (SI 142)
20	MI	MI
21	MI	DB
22	DB	MI
23	DB	DB

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.


Copy Vector: Function Number 22, DB to MI
Function: Operands
SI 141: 5
SI 142: 10
SI 143: 32
If 5 is stored into SI 141…
...register 5 within the Database is the start of the source vector…
DB vector 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
...If 32 is stored into SI 143, MI 32 within the Database is the start of the target vector.
... if 10 is stored in SI 142, both the source and target vector will be 10 registers long…
MI vector 0 1 ... 31 32 33 34 35 36 37 38 39 40 41 42 43

Fill Vector

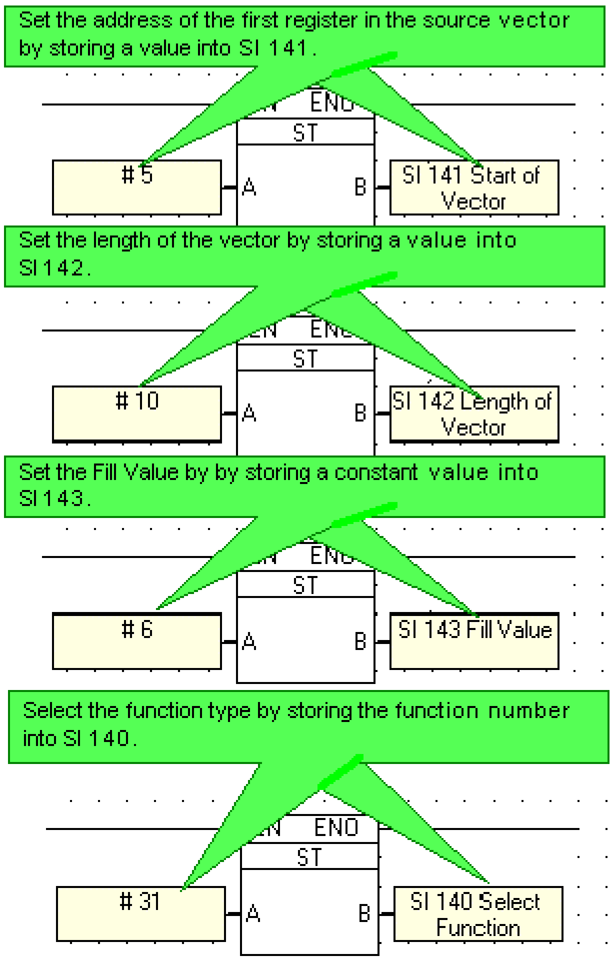
Fill Vector enables you to set a range of registers. The function copies a value from a desired operand or constant value **(source)**, then writes that value into every operand within the range **(target vector)**.

You can fill a vector of MI registers or Database registers by selecting the appropriate function.

Note that since there is no Ladder element for this function; you perform it by storing values into:

- SI 141 to determine the start of the target vector,
- SI 142 to determine the length of the target vector,
- SI 143 to select the Fill Value; the register whose value will be written into each register within the target vector,
- SI 140 to select the type of function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters.

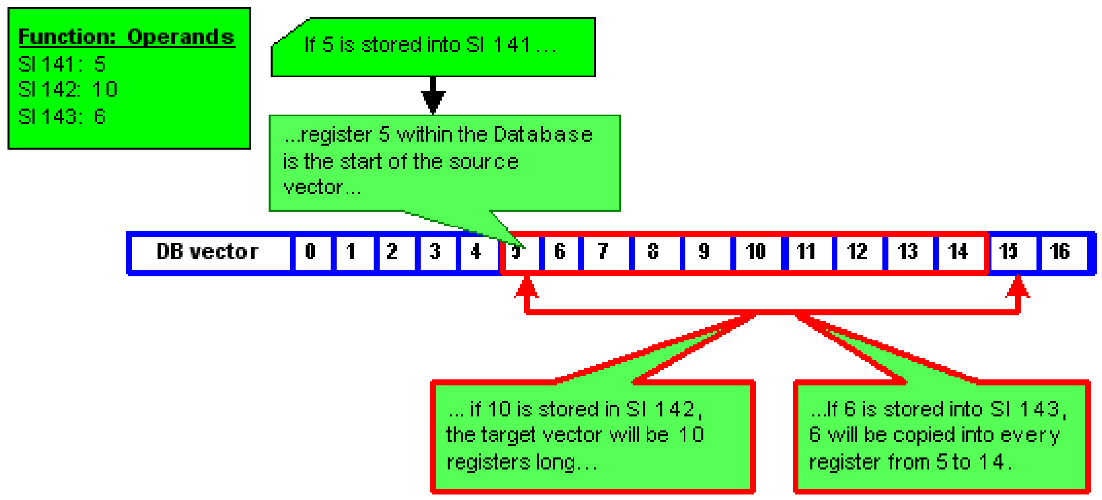
To use Fill Vector:


Set the address of the first register in the source vector by storing a value into SI 141.
EN ENO
ST
# 5
A
B
SI 141 Start of Vector
Set the length of the vector by storing a value into SI 142.
EN ENO
ST
# 10
A
B
SI 142 Length of Vector
Set the Fill Value by by storing a constant value into SI 143.
EN ENO
ST
# 6
A
B
SI 143 Fill Value
Select the function type by storing the function number into SI 140.
EN ENO
ST
# 31
A
B
SI 140 Select Function

Function Number (SI 140)	Description
30	Fill MI Vector
31	Fill DB Vector

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Fill Vector: Function Number 31, Fill DB vector


Function: Operands
SI 141: 5
SI 142: 10
SI 143: 6
If 5 is stored into SI 141...
...register 5 within the Database is the start of the source vector...
DB vector
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
... if 10 is stored in SI 142, the target vector will be 10 registers long...
...If 6 is stored into SI 143, 6 will be copied into every register from 5 to 14.

Find Mean, Maximum, and Minimum Values

This function enables you to take a vector of registers and find the:

- Mean of all the values in the vector,
- Minimum value in the vector,
- Maximum value in the vector.

You can base the function on a vector of MI registers or Database registers by selecting the appropriate function.

Note that since there is no Ladder element for this function; you perform it by storing values into:

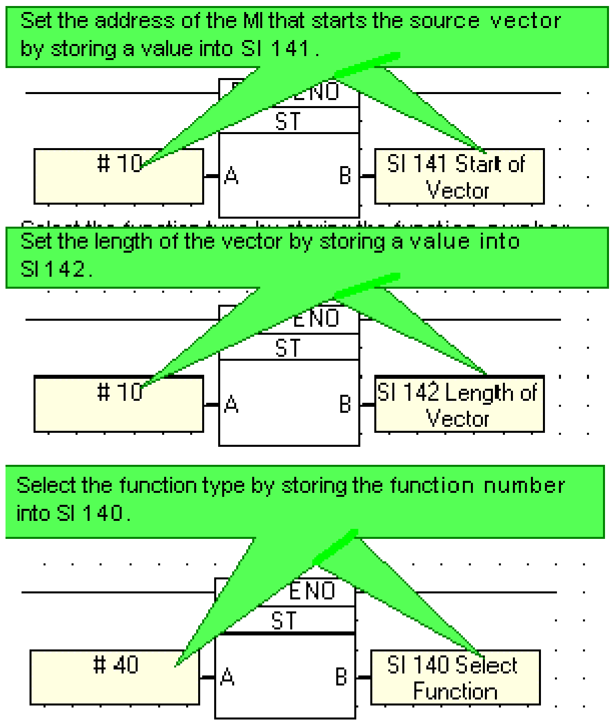
- SI 141 to determine the start of the vector,
- SI 142 to determine the length of the vector,
- SI 140 to select the type of function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters.

The results will be placed in:

- SI 143: Mean
- SI 144: Minimum
- SI 145: Maximum

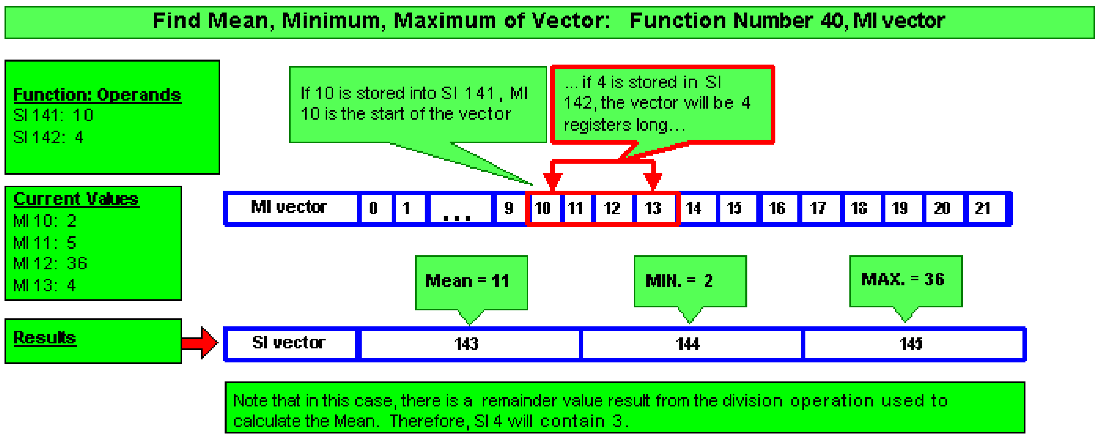
Note that if a remainder value results from the division operation used to calculate the Mean, that remainder value will be place in SI 4, Divide Remainder.

To use this function:


Set the address of the MI that starts the source vector by storing a value into SI 141.
ENO
ST
# 10
A
B
SI 141 Start of Vector
Set the length of the vector by storing a value into SI 142.
ENO
ST
# 10
A
B
SI 142 Length of Vector
Select the function type by storing the function number into SI 140.
ENO
ST
# 40
A
B
SI 140 Select Function

Function Number (SI 140)	Description
40	Find Mean, Minimum, Maximum in MI vector
41	Find Mean, Minimum, Maximum in DB vector

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.


Find Mean, Minimum, Maximum of Vector: Function Number 40, MI vector
Function: Operands
SI 141: 10
SI 142: 4
Current Values
MI 10: 2
MI 11: 5
MI 12: 36
MI 13: 4
Results
If 10 is stored into SI 141, MI 10 is the start of the vector
... if 4 is stored in SI 142, the vector will be 4 registers long...
MI vector
0
1
...
9
10
11
12
13
14
15
16
17
18
19
20
21
Mean = 11
MIN. = 2
MAX. = 36
SI vector
143
144
145
Note that in this case, there is a remainder value result from the division operation used to calculate the Mean. Therefore, SI 4 will contain 3.

GSM PIN Code via MI

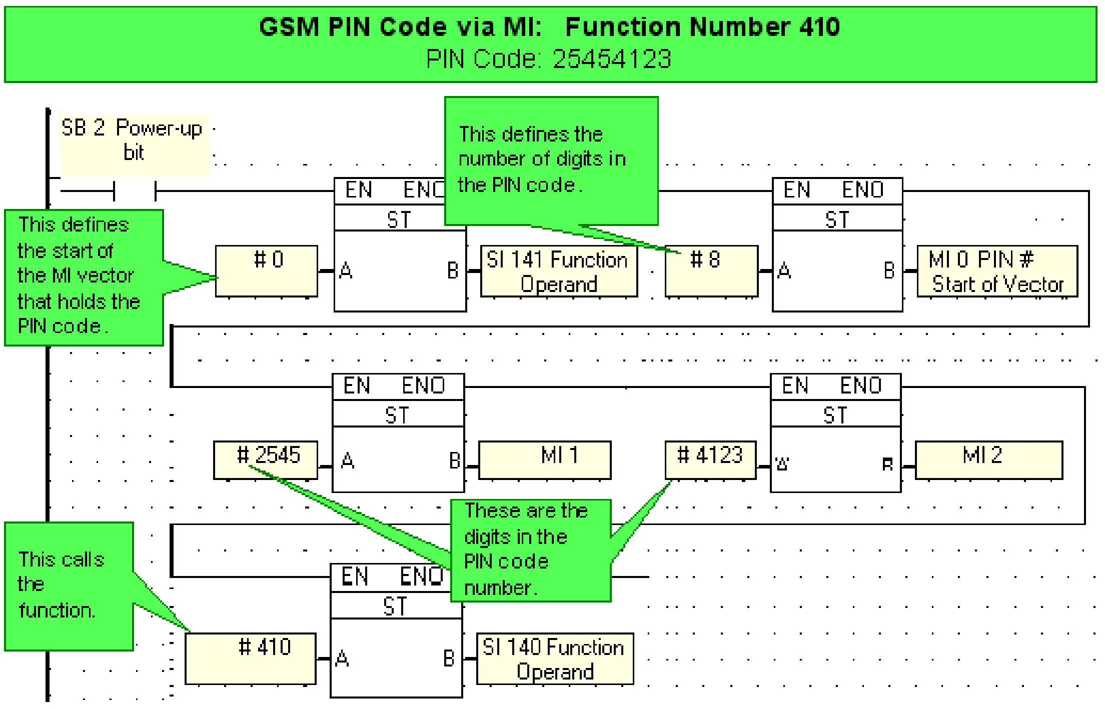
Use this utility to use an MI vector to supply a GSM modem PIN code. When you use this function, the controller will look for the number in the MIs, bypassing the PIN code in the SMS message dialog box.

Note that since there is no Ladder element for this function; you perform it by:

- Storing the start address of the MI vector needed to contain the PIN into SI 141,
- Storing 410 into SI 140 to select the function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters. Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

The PIN code should be called before the modem is initialized; the function should therefore be called as a power-up task.

Note that if the MIs contain an incorrect PIN code format, the error will be indicated by Error message #18 in SI 180--Illegal PIN Format.


GSM PIN Code via MI: Function Number 410
PIN Code: 25454123
SB 2 Power-up bit
This defines the start of the MI vector that holds the PIN code.
This defines the number of digits in the PIN code.
EN ENO
ST
A B
# 0
SI 141 Function Operand
# 8
MI 0 PIN # Start of Vector
# 2545
MI 1
# 4123
MI 2
These are the digits in the PIN code number.
This calls the function.
# 410
SI 140 Function Operand

Interrupt

This function is time-based. You call an interrupt routine by storing 500 into SI 140. The interrupt function causes:

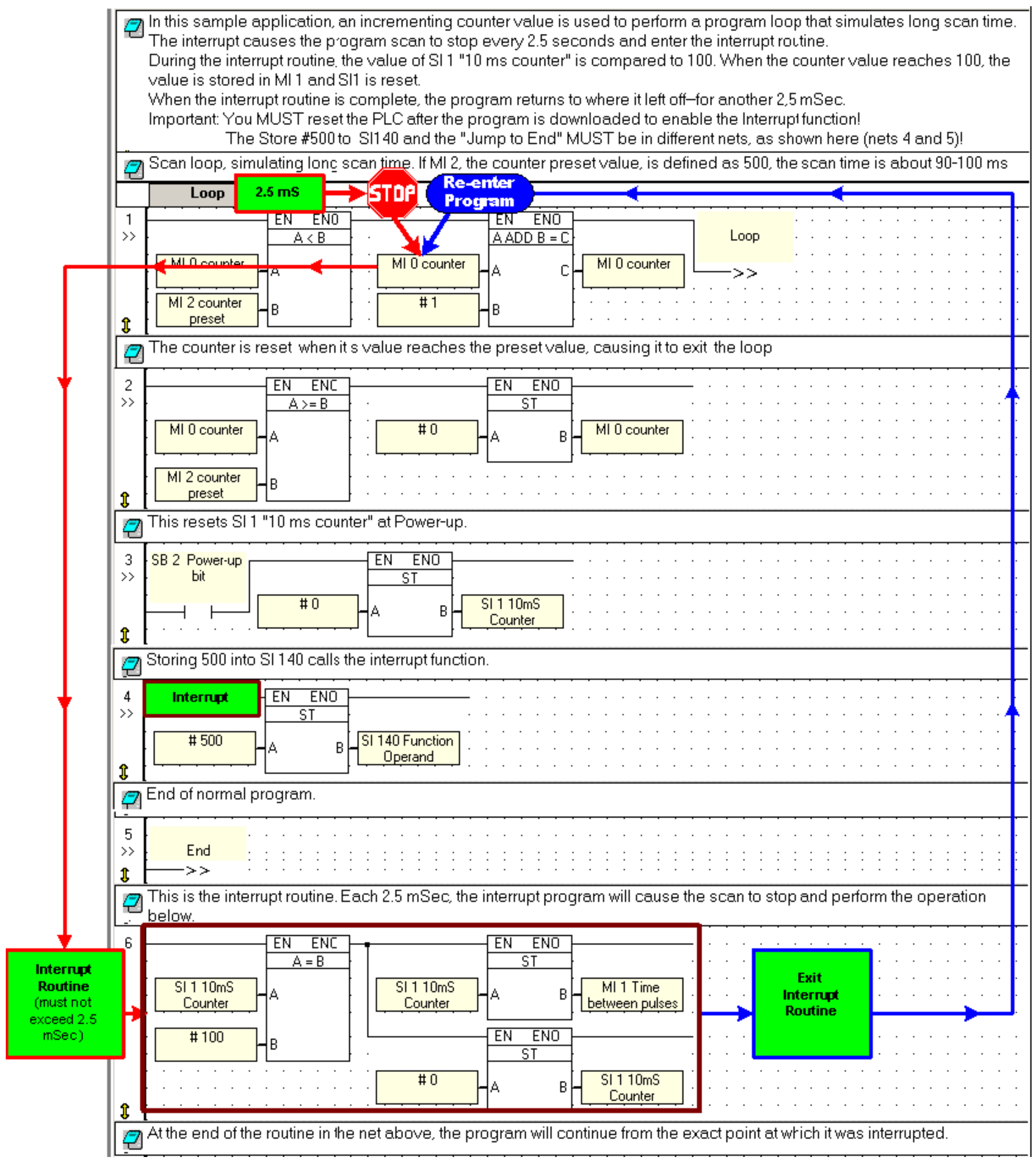
- The program scan to pause every 2.509 mSec. The interrupt causes the program to stop immediately without regard to the program scan, even if it occurs in the middle of a net.
- A jump to the net which follows the interrupt. The nets following the interrupt comprise the interrupt routine. Note that 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 continues from where it left off.

Note that the nets containing the Interrupt routine must be the last ones in the program. The format must be as shown in the example below:

- Store 500 into SI140 to call the function
- Jump to End
- The nets containing the actual interrupt routine.

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Example



In this sample application, an incrementing counter value is used to perform a program loop that simulates long scan time.
The interrupt causes the program scan to stop every 2.5 seconds and enter the interrupt routine.
During the interrupt routine, the value of SI 1 "10 ms counter" is compared to 100. When the counter value reaches 100, the value is stored in MI 1 and SI1 is reset.
When the interrupt routine is complete, the program returns to where it left off—for another 2,5 mSec.
Important: You MUST reset the PLC after the program is downloaded to enable the Interrupt function!
The Store #500 to SI140 and the "Jump to End" MUST be in different nets, as shown here (nets 4 and 5)!

Scan loop, simulating long scan time. If MI 2, the counter preset value, is defined as 500, the scan time is about 90-100 ms

The counter is reset when it s value reaches the preset value, causing it to exit the loop

This resets SI 1 "10 ms counter" at Power-up.

Storing 500 into SI 140 calls the interrupt function.

End of normal program.

This is the interrupt routine. Each 2.5 mSec, the interrupt program will cause the scan to stop and perform the operation below.

At the end of the routine in the net above, the program will continue from the exact point at which it was interrupted.

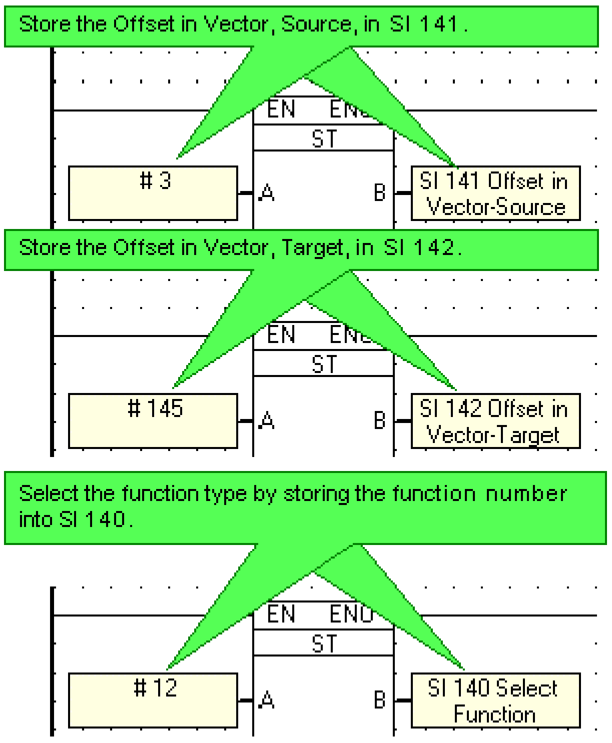
Load Indirect

Load Indirect allows you to take a value contained in a **source** operand and load that value into a **target** operand using indirect addressing. Note that since there is no Ladder element for this function; you perform it by storing values into:

- SI 141 to determine the data source,
- SI 142 to determine the load target,
- SI 140 to select the type of function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters.

To use Load Indirect:

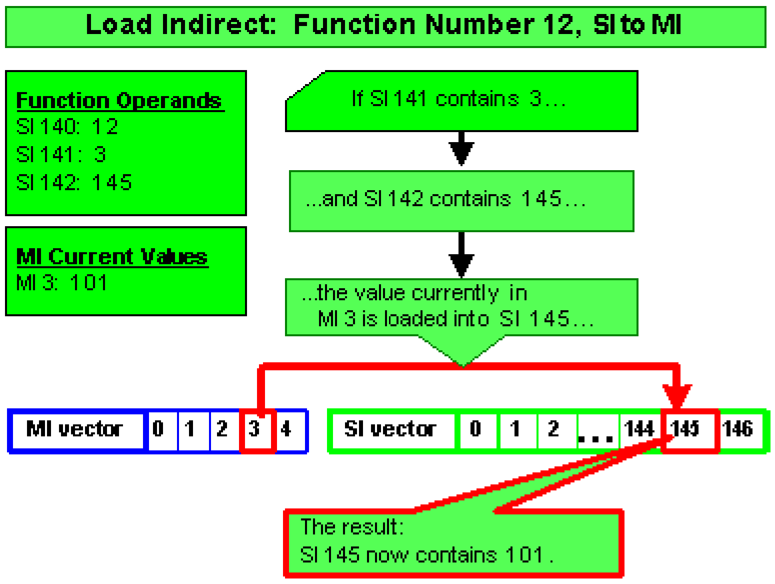
•


Store the Offset in Vector, Source, in SI 141.
EN
ENO
ST
# 3
A
B
SI 141 Offset in Vector-Source
Store the Offset in Vector, Target, in SI 142.
EN
ENO
ST
# 145
A
B
SI 142 Offset in Vector-Target
Select the function type by storing the function number into SI 140.
EN
ENO
ST
# 12
A
B
SI 140 Select Function

Function Number (SI 140)	Offset in Vector, Source (SI 141)	Offset in Vector, Target (SI 142)
10	MI	MI
11	SI	MI
12	MI	S
13	SI	S

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

•


Load Indirect: Function Number 12, SI to MI
Function Operands
SI 140: 12
SI 141: 3
SI 142: 145
MI Current Values
MI 3: 101
If SI 141 contains 3...
...and SI 142 contains 145...
...the value currently in MI 3 is loaded into SI 145...
MI vector
0
1
2
3
4
SI vector
0
1
2
...
144
145
146
The result: SI 145 now contains 101.

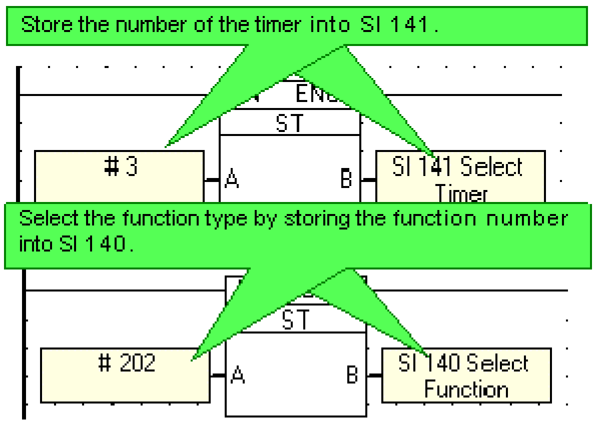
Load Timer Preset/Current Value

This function allows you to take a preset or current timer value and load it into another operand. Note that since there is no Ladder element for this function; you perform it by storing values into:

- SI 141 to select the timer; 0-63,

- SI 140 to select the type of function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters.

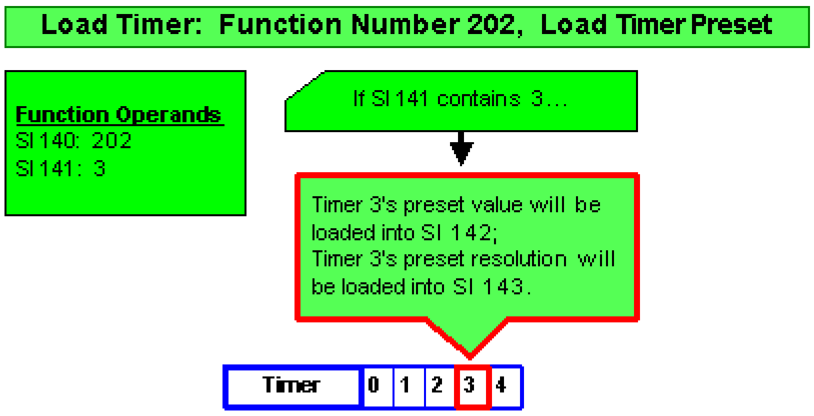
To use this function:


Store the number of the timer into SI 141.
EN
ST
#3
A
B
SI 141 Select Timer
Select the function type by storing the function number into SI 140.
ST
# 202
A
B
SI 140 Select Function

Function Number (SI 140)	Description
202	Load Timer Preset
203	Load Timer Current

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Timer Resolution (stored into SI 143)	
Value	**Resolution**
1	10mS (0.01S)
10	100mS (001S)
100	1000mS (1S)
1000	10000mS (10S)


Load Timer: Function Number 202, Load Timer Preset
Function Operands
SI 140: 202
SI 141: 3
If SI 141 contains 3...
Timer 3's preset value will be loaded into SI 142;
Timer 3's preset resolution will be loaded into SI 143.
Timer
0
1
2
3
4

MODBUS

MODBUS enables you to establish master-slave communications with any connected device that supports the MODBUS protocol. Any controller in the network may function as either master or slave using any of the controller's existing COM Ports.

Unitronics currently supports RTU (binary) transmission mode. Note that M91 models support MODBUS, M90 models do not.

Since there are no Ladder element for these functions; you perform them by storing values into SIs in accordance with the tables and figures shown below.

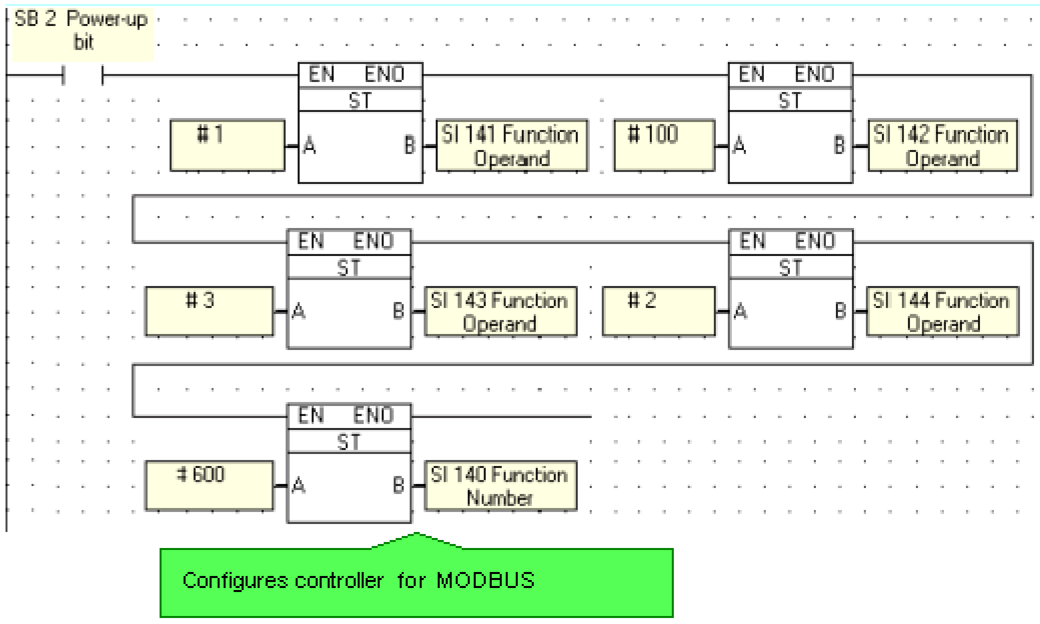
MODBUS Configuration

Before you can run a MODBUS command, you must configure MODBUS parameters for both Master and Slave devices.

Configuration Parameters

These parameters configure a controller for MODBUS communications. A device is configured for MODBUS by storing the value 600 into SI 140.

To configure a slave device, build a Ladder net that stores the appropriate values into the SIs according to the following table, and that ends by storing the value 600 into SI 140.

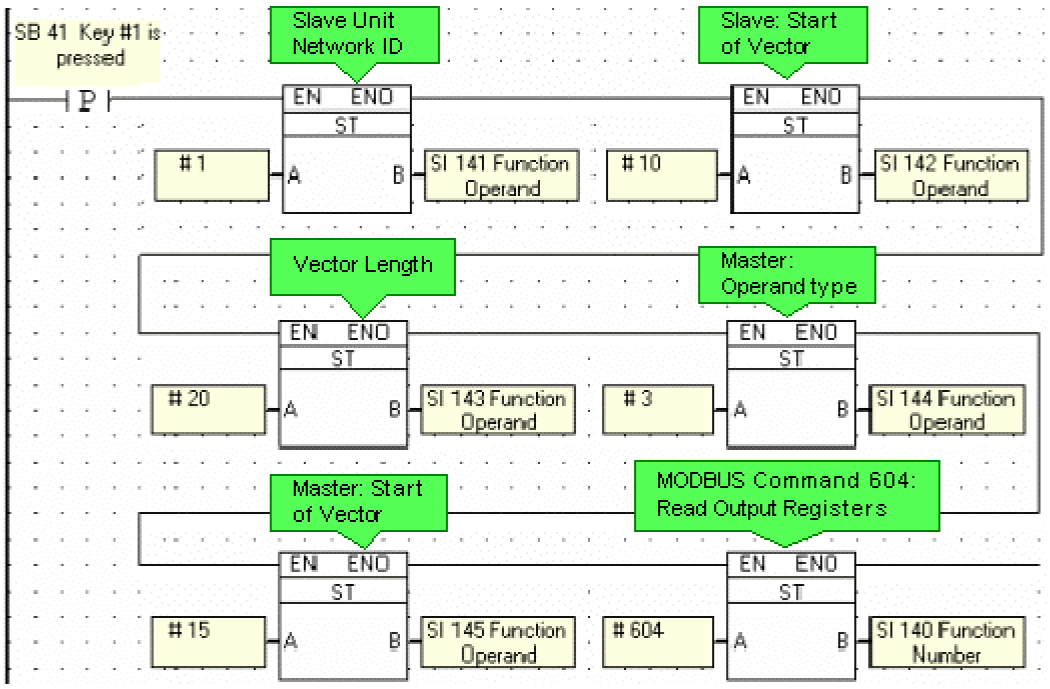


Parameter	Store into SI	Function
Network ID	141	This number identifies the device on the network. You can either assign an ID via an MI, or directly via a constant number. The unit ID range is from 0-255. Do not assign the same ID number to more than one device.
Time out	142	This is the amount of time a master device will wait for an answer from a slave. Time out units are defined in 10 msecs; a Time out value of 100 is equal to 1 second.
Retries	143	This is the number of times a device will try to send a message.
Maximum Time Delay	144	This is the maximum time interval permitted between messages. The time units are 2.5 msec. This should be set to 2, setting the permitted interval to 5 msecs (n x 2.5 =interval).
Call MODBUS Configuration	140	Storing the value 600 into SI 140 configures the controller for MODBUS. This must be the final parameter stored.

MODBUS Commands

Before you can call a MODBUS command, you store the appropriate parameter values into the correct SIs in accordance with the Command Parameters table. After this is done, call the command by storing the command number into SI 140.

The figure below shows how to implement the MODBUS command Read Output Registers.



Command Parameters

Parameter	Store into SI	Function
Slave Unit Network ID	141	The ID of the slave device containing the data to be read (data source).
Slave: Start of Vector	142	The start of the vector of operands in the slave. Check the Slave Address Tables below.
Vector Length	143	The vector length. **Note** ♦ A MODBUS command cannot read/write more than 1900 bit operands at one time. In addition, 0 is not a legal length.
Master: Operand Type	144	Store the number that relates to the type of operand you wish to write to in the master device. **MB** 1 **SB** 2 **MI** 3 **SI** 4 **I** 9 **O** 10

T (current)	**122**
T (preset	**128**

Master start of Vector	145
MODBUS Command	140

Note ♦ While a master attempts to send a command, SB 63 Function In Progress is ON. The number of attempts that the master will make is the number in Retries +1, where '1' is the initial access attempt.

♦ When a master attempts to access a slave device, and the slave does not answer, SB 66 Function In Progress will turn ON. This bit will remain on according to the following:
(the number of retries + 1) x (Time Out), where '1' is the initial access attempt.
Note that the Time Out parameter is in units of 10 msec.

MODBUS Command Number

MODBUS Commands	**U90 Command # (Value to store into SI 140**
Read Coils	601
Force Coil	602
Force Coils	603
Read Output Registers	604
Preset Register	605
Preset Registers	606
Read Output Registers in Float Format	607
Preset Float Registers	608
Read Input Registers	609
Read Input Registers in Float Format	610
Read Inputs	611
Loopback Test	612

MODBUS Indications: SBs and SIs

SB 66 **Function in Progress** Shows status of master's **MODBUS Configuration**	**Turns ON when:** • A master Vision initiates MODBUS communication. • Remains ON during the MODBUS session.	**Turns OFF when** • The **MODBUS: Configuration** is activated. • An answer is received from a slave. • The TimeOut defined in the **Configuration** is exceeded. • Certain Status Messages are given
SI 66 **Status Messages** Shows status of master's data requests and the replies the master	• Automatically initialized to 0 when MODBUS operation is activated. • Updated at the end of each attempt to communicate via MODBUS. • Indicates status of **MODBUS** communications, according to the table below. Note that the current value always shows the most	

receives from the slaves	**recent** status.

#	Status Message
0	Status OK
1	Unknown Command Number This is received from the slave device**.**
2	Illegal Data Address ● Master: an invalid address is found by the master before a data request is sent to a slave. This may result, for example, when an MI is used to provide vector length. ● Slave: The slave notifies the master that the data request command includes invalid addresses.
3	Slave to Master: Illegal Data Type Quantity Number of operands requested by user exceeds the maximum Note ♦ A MODBUS command cannot read more than 124 16-bit integers, 62 double registers, 62 float registers, or 1900 bit operands at one time. In addition, 0 is not a legal vector length.
4	Master--Time Out The amount of time the master will attempt to establish a MODBUS session
5	No Communication The MODBUS session cannot be established.
Note♦ Messages 4 & 5. **TimeOut** and **Number of Retries** are defined as **Configuration Parameters**. A Retry is an attempt to establish a MODBUS session. If, for example, TimeOut is defined as 2 seconds, and number of Retries as 3: - the controller will try to establish the session once, and will continue to try for 2 seconds. - If the first attempt fails, the **Status Message value will be 4**, Master TimeOut. -The controller will try twice more, for a total of 3 retries over 6 seconds. - If all attempts fail, the **Status Message value will be 5**. -If any attempt succeeds, the Status Message will be 0.	
*6	Master-slave data incorrectly synchronized
*7	Master-slave data incorrectly synchronized
8	Master to application: Illegal Data Type Quantity Number of operands requested by user exceeds the maximum permitted for that FB operation in the master. Note ♦ A MODBUS command cannot read more than 124 16-bit integers, 62 double registers, 62 float registers, or 1900 bit operands at one time. In addition, 0 is not a legal vector length.
9	Slave ID =0 An attempt does to communicate with Slave ID 0.
*11	Master-slave data incorrectly synchronized

***** Messages 6, 7, and 11mean that the master has found incompatible elements in the data sent between master and slave.

Slave Address Tables

Coils		MODBUS Command Number	
Pointer Value From:	Operand type	Read	Write
0000	MB	#01 Read Coils	#15 Force Coils
3000	SB		#15 Force Coils
4000	I (read-only)		Read-only
5000	O		#15 Force Coils
6000	T(read-only)		Read-only
7000	C(read-only)		Read-only

Registers			**MODBUS Command Number**	
Pointer Value From:	Operand type	Register size	Read	Write
0000	MI	16 bit	# 03 Read Holding Registers	# 16 Preset Holding Registers
4000	SI	16 bit		
5100	ML	32 bit		
6100	SL	32 bit		
6300	MDW	32 bit		
6700	SDW	32 bit		
6900	Timer preset	32 bit		
7200	Timer current	32 bit		
7500	Counter	16 bit		
7700	MF 0	32 bit		

Examples

The examples below show that:

- MODBUS addressing systems start at 1.
- Unitronics PLC addressing starts at 0.

Bit Operands

Read a 10-bit vector of inputs in a slave Unitronics PLC, starting at Input 20, via Read Coils (Command 601)

- Unitronics PLC as the MODBUS master

Store 4020 into SI 142 (Slave: Start of Vector parameter), 10 into SI 143 (Read: Vector Length parameter), 9 into SI 144 (Master: Operand Type), and 601 into SI 140. Within the slave PLC, the master PLC will read I 20 - I 29.

- SCADA as the MODBUS master
  In the SCADA application, set the Slave: Start of Vector parameter to 34021(30001 + 4000 + 20), and the Read: Vector Length to 10, enabling the master device to read I 20 - I 29 within the slave PLC.

Write a 3-bit vector of outputs in a slave Unitronics PLC, starting at Output 8, via Force Coils (Command 603)

- Unitronics PLC as the MODBUS master
  Store 5008 into SI 142 (Slave: Start of Vector parameter), 3 into SI 143 (Read: Vector Length parameter), 10 into SI 144 (Master: Operand Type), and 603 into SI 140. Within the slave PLC, the master will write to O 8 - O 10.
- SCADA as the MODBUS master
  In the SCADA application, set the Slave: Start of Vector parameter to 35009 (30001 + 5000 + 8) and the Read: Vector Length parameter to 3, enabling the master device to write to O 8 - O 10 within the slave controller.

Registers

Read a 9-register long vector of **16-bit integers** in a slave Unitronics controller, starting at MI 32, via Read Holding Registers (Command 604)

- Unitronics PLC as the MODBUS master
  Store 32 into SI 142 (Slave: Start of Vector parameter), 9 into SI 143 (Read: Vector Length parameter), 3 into SI 144 (Master: Operand Type),and 604 into SI 140. Within the slave PLC, the master PLC will read MI 32 - MI 40.
- SCADA as the MODBUS master
  In the SCADA application, set the Slave: Start of Vector parameter to 40033 (40001 + 0000 + 3), and the Read: Vector Length parameter to 9, enabling the master device to read MI 32 - MI 41 within the slave controller.

Note ♦ M91 does not support 32-bit registers.

Write a 6-register long vector of **16-bit integers** in a slave Unitronics controller, starting at MI 32, via Preset Registers (Command 606)

- Unitronics PLC as the MODBUS master
  Store 32 into SI 142 (Slave: Start of Vector parameter), 6 into SI 143 (Read: Vector Length parameter), 3 into SI 144 (Master: Operand Type),and 606 into SI 140. Within the slave PLC, the master PLC will write to MI 32 - MI 37.
- SCADA as the MODBUS master
  In the SCADA application, set the Slave: Start of Vector parameter to 40033, and the Read: Vector Length parameter to 6, enabling the master device to write to MI 32 - MI 37 within the slave controller.

SMS Phone Number: via MI Pointer

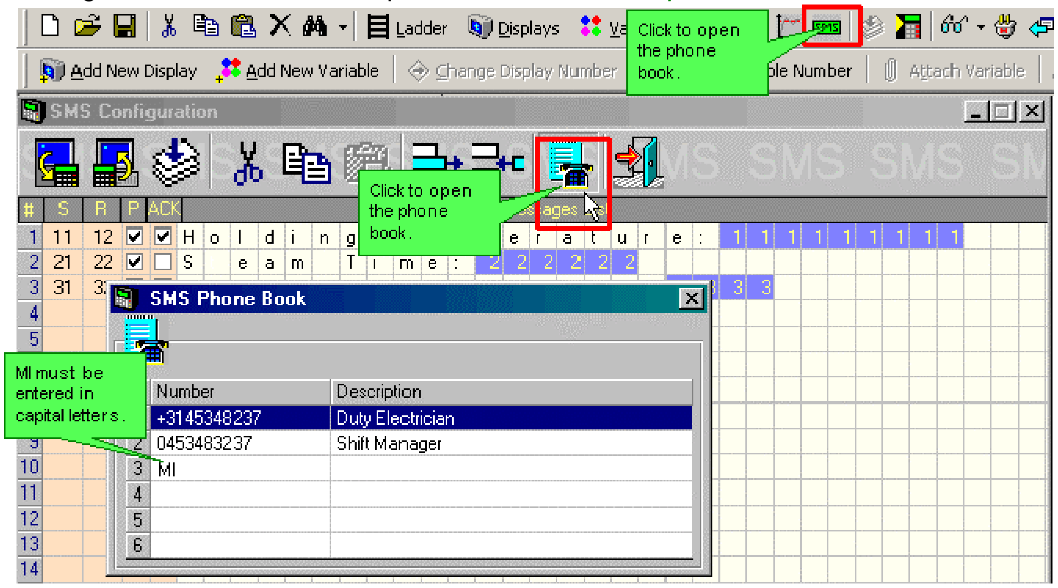
Use this utility to use an MI vector as one of the phone numbers in the SMS phone book. This allows you to:

- Enable a number to be dialed via the M90's keypad.

- Exceed the 6 number limit of the SMS phone book.

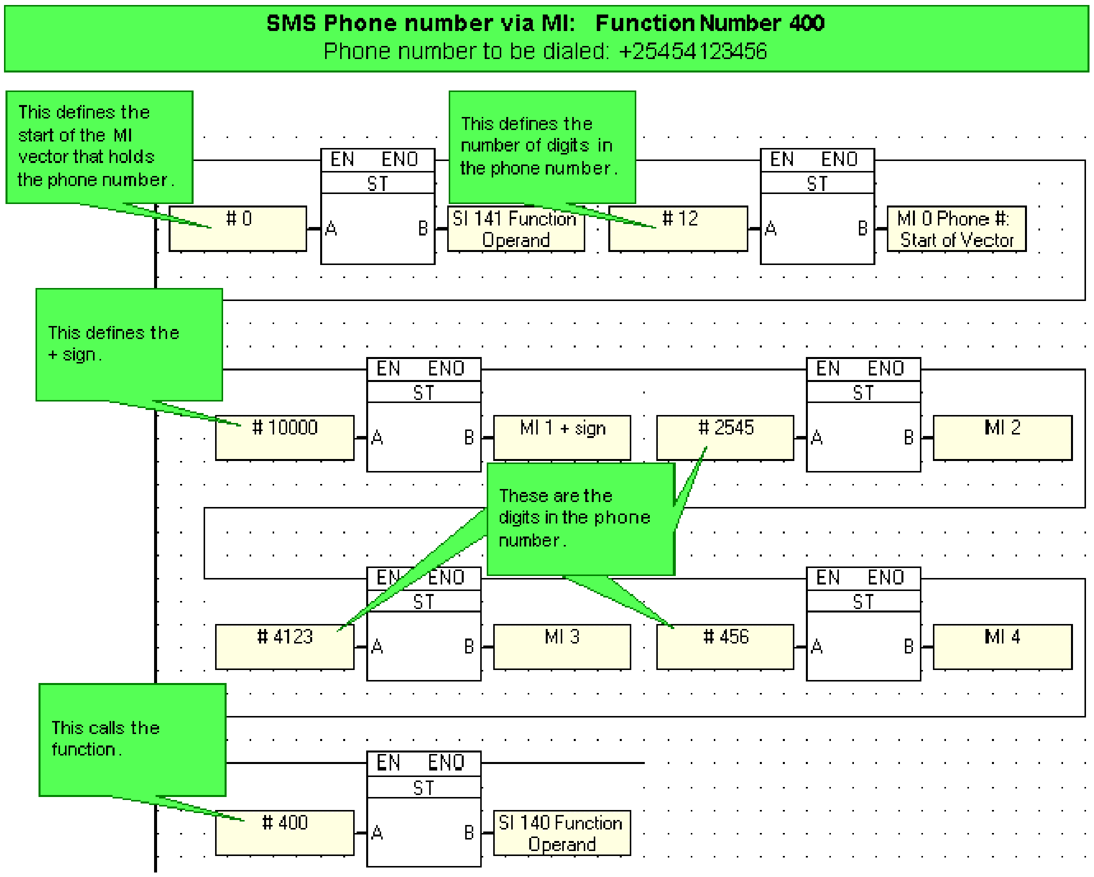
Note that since there is no Ladder element for this function; you perform it by:

- Storing the start address of the MI vector needed to contain the phone number into SI 141,
- Entering the character's MI, in capital letters, in the *SMS phone book*,



	Number	Description
1	+3145348237	Duty Electrician
2	0453483237	Shift Manager
3	MI	
4		
5		
6		

- Using the index number of that line to call the number, which enables the number in the MI vector to be called,
- Storing 400 into SI 140 to select the function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters. Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.


SMS Phone number via MI: Function Number 400
Phone number to be dialed: +25454123456
This defines the start of the MI vector that holds the phone number.
This defines the number of digits in the phone number.
EN ENO ST A B
# 0
SI 141 Function Operand
# 12
MI 0 Phone #: Start of Vector
This defines the + sign.
# 10000
MI 1 + sign
# 2545
MI 2
These are the digits in the phone number.
# 4123
MI 3
# 456
MI 4
This calls the function.
# 400
SI 140 Function Operand

Store Timer's Preset/Current Value

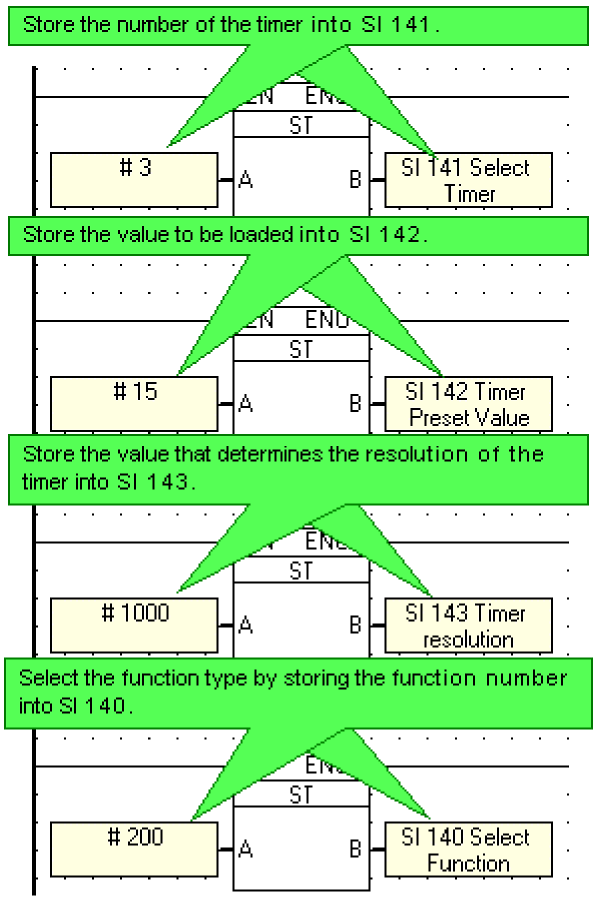
This function allows you to take a value and store it into a timer to change the preset or current timer value. Since there is no Ladder element for this function; you perform it by storing values into :

- SI 141 to select the timer; 0-63,
- SI 142 to determine the timer value,
- SI 143 to select the timer's resolution (timer units, or 'ticks'),
- SI 140 to select the type of function. Storing the function number calls the function. In your application, call the function **after** you have entered all of the other parameters.

Take into account that:

- Since you cannot change the resolution of a timer when the application is running, SI 143 is not used in a Store Timer's Current Value function.
- A timer's current value can be changed at any time, including when the timer is active. The new value can be either greater or smaller than the previous value; storing 0 into a timer's current value stops it immediately.
- A change of Timer Preset value without changing the resolution will take effect when the timer restarts.
- Changing the resolution of the timer's preset value does not affect the current resolution; it is therefore recommended that the resolution not be changed while the timer is active.
- The timer value is 14 bits.

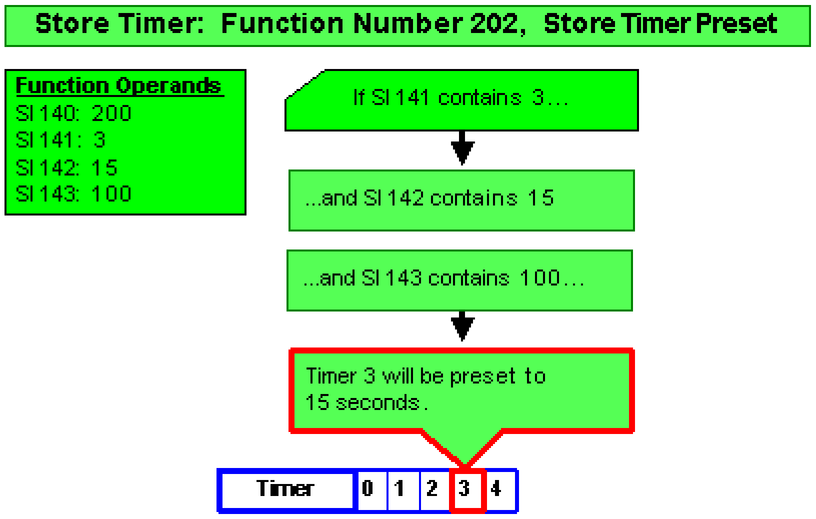
To use this function:


Store the number of the timer into SI 141.
EN ENO
ST
# 3
A
B
SI 141 Select Timer
Store the value to be loaded into SI 142.
EN ENO
ST
# 15
A
B
SI 142 Timer Preset Value
Store the value that determines the resolution of the timer into SI 143.
ENO
ST
# 1000
A
B
SI 143 Timer resolution
Select the function type by storing the function number into SI 140.
ENO
ST
# 200
A
B
SI 140 Select Function

Function Number (SI 140)	Description
200	Store Timer Preset
201	Store Timer Current

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Timer Resolution (stored into SI 143)	
Value	**Resolution**
0	Maintain Timer Resolution
1	10mS (0.01S)
10	100mS (0.1S)
100	1000mS (1.0S)
1000	10000mS (10.0S)


Store Timer: Function Number 202, Store Timer Preset
Function Operands
SI 140: 200
SI 141: 3
SI 142: 15
SI 143: 100
If SI 141 contains 3...
...and SI 142 contains 15
...and SI 143 contains 100...
Timer 3 will be preset to 15 seconds.
Timer
0
1
2
3
4

Square Root

This function enables you to find the square root of a number.

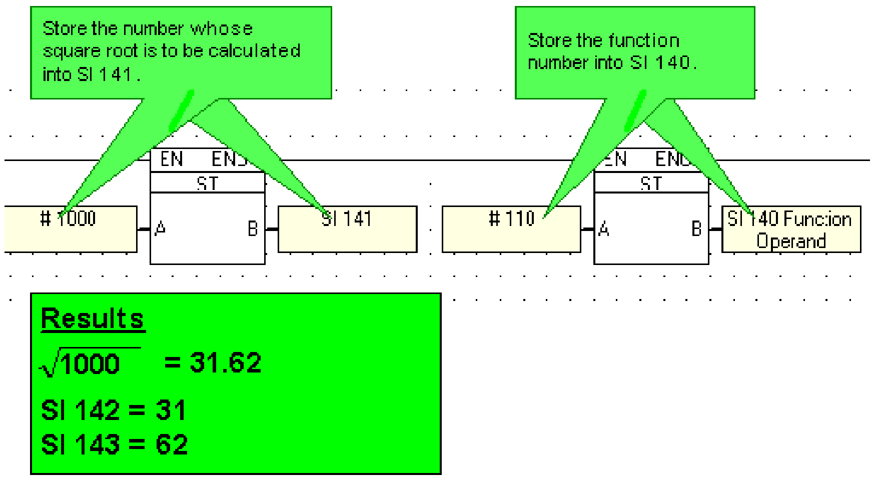
Since there is no Ladder element for this function; you perform it by storing the number whose square root is to be calculated into SI 141.

Store 110 into SI 140 to call the function. In your application, call the function **after** you have entered all of the other parameters.

The results will be placed in:

- SI 142. This contains the whole number result.
- SI 143. If the result is not a whole number, this contains up to 2 digits to the left of the decimal point.

To use this function:


Store the number whose square root is to be calculated into SI 141.
Store the function number into SI 140.
EN ENO
ST
# 1000
A B
SI 141
# 110
SI 140 Function Operand
Results
√1000 = 31.62
SI 142 = 31
SI 143 = 62

Function Number (SI 140)	Description
110	Calculate square root

Note that when you run Test (Debug) Mode, the current value in SI 140 will **not** be displayed.

Functions activated by SBs

Convert MB to MI, MI to MB

An M90 register is built of 16 bits.

Using the MB to MI function, you can convert 16 bits or more into a integer value. Conversely, you can convert an integer value into 16 bits or more using the MI to MB function.

Note that if the converted values exceed 16 bits, the function will write the value to consecutive registers. Any values in those registers will be overwritten.

To apply the functions, use the following System Integers (SI) and System Bits (SB)

SI	Description	SB	
SI170	Address of MI containing integer value	SB170	MB to MI
SI171	Start address of MB array (vector)	SB171	MI to MB
SI172	Amount of MBs		

You can use this function, for example to send an SMS when there is a change in the status of the M90's inputs:

1. Represent the status of the M90's inputs using MBs.

2. Convert these MBs into an MI
3. Perform a XOR operation on the result.

When there is a change in input status, the XOR operation will return a value different than 0, which may then be used to trigger the sending of an SMS.

Examples

Example 1:

1. Store the value 7 into SI 170, 10 into SI 171 and 9 into SI 172.
2. Set SB 170 to ON.

The program will calculate the binary value of a 9 bit array which starts with MB 10. The resulting value will be placed into MI 7.

Example 2:

1. Store the value 7 into SI 170, 10 into SI 171 and 9 into SI 172.
2. Set SB 171 to ON

The program will calculate the binary value of the value contained in MI 7. The result will be scattered on a 9 bit array which starts with MB 10.

Copy MI to Output vector, Input vector to MI

Using this function, you can:

- Copy a vector of Inputs (**`I`**) to a register.
- Copy a register value to a vector of Outputs (**`O`**).

Note that an M90 register contains 16 bits. If the converted values exceed 16 bits, the function will write the value to consecutive registers. Any values in those registers will be overwritten. When a register value is copied to outputs, the function will store the register value in consecutive outputs.

Input to Register

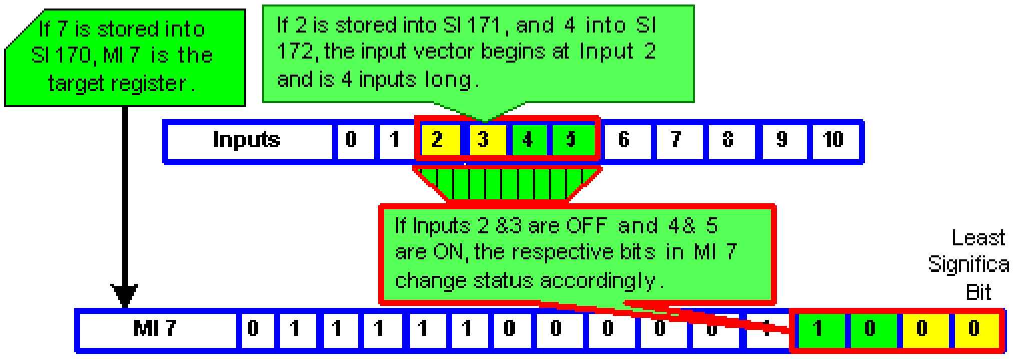
SI	Description	SB	Function
SI170	Address of MI containing integer value	SB172	I to MI
SI171	Start address of bit array (vector)	SB173	MI to O
SI172	Amount of bits		

Example: Input to MI, SB 172

1. Store the value 7 into SI 170, 2 into SI 171 and 4 into SI 172.
2. Set SB 172 to ON.

The program takes the status of I2 to I5, and changes the status of the respective bits in MI 7.

Bits in the target register that are outside of the defined range are not affected.


If 7 is stored into SI 170, MI 7 is the target register.
If 2 is stored into SI 171, and 4 into SI 172, the input vector begins at Input 2 and is 4 inputs long.
Inputs
0
1
2
3
4
5
6
7
8
9
10
If Inputs 2 &3 are OFF and 4& 5 are ON, the respective bits in MI 7 change status accordingly.
Least Significant Bit
MI 7
0 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0

Example: MI to Output, SB 173

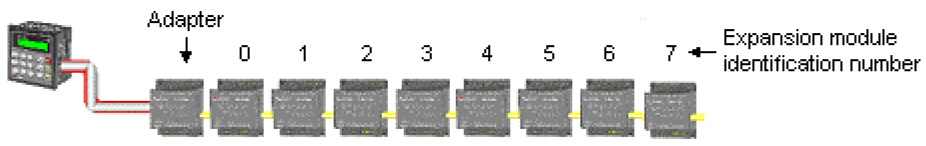
1. Store the value 7 into SI 170, 3 into SI 171 and 7into SI 172.
2. Set SB 173 to ON.

The program will take the binary value of the MI 7, and change the status of the respective outputs in the defined vector, O3 to O7.

Addressing: I/O Expansion Modules

Inputs and outputs located on I/O expansion modules that are connected into an M90 OPLC are assigned addresses that comprise a letter and a number. The letter indicates whether the I/O is an input (I) or an output (O). The number indicates the I/O's location in the system. This number relates to both the expansion module's position in the system, and to the position of the I/O on that module.

Expansion modules are numbered from 0-7 as shown in the figure below.



The formula below is used to assign addresses for I/O modules used in conjunction with the M90 OPLC.

X is the number representing a specific module's location (0-7). Y is the number of the input or output on that specific module (0-15).

The number that represents the I/O's location is equal to: **32 + x • 16 + y**

Example

- Input #3, located on expansion module #2 in the system, will be addressed as I 67, 67 = 32 + 2 • 16 + 3
- Output #4, located on expansion module #3 in the system, will be addressed as O 84, 84 = 32 + 3 • 16 + 4.

EX90-DI8-RO8 is a stand-alone I/O module. Even if it is the only module in the configuration, the EX90-DI8-RO8 is always assigned the number 7. Its I/Os are addressed accordingly.

Example

- Input #5, located on an EX90-DI8-RO8 connected to an M90 OPLC will be addressed as I 149, 149 = 32 + 7 • 16 + 5

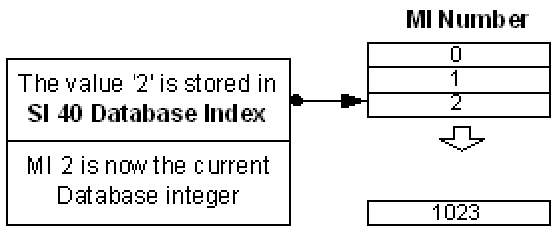
Database: Access indirectly addressed registers

The M90 OPLC has a special memory area containing integers that are function as a database. These integers are not related in any way to system or memory integers. Within the database, you can access and use integers 0 through 1023 via SI 40 and SI 41.

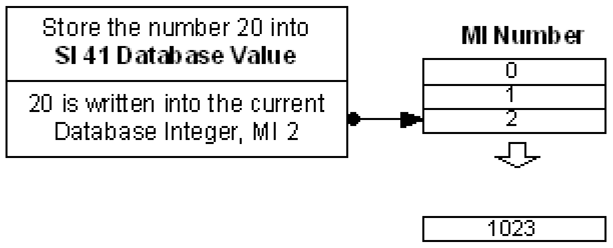
Note that when you run Test (Debug) Mode, the current value in SI 140 (Function Number) will **not** be displayed.

Writing Values

1. Use SI 40 Database Index to access a particular MI.
   For example, to access MI 2 you store the number 2 into SI 40.

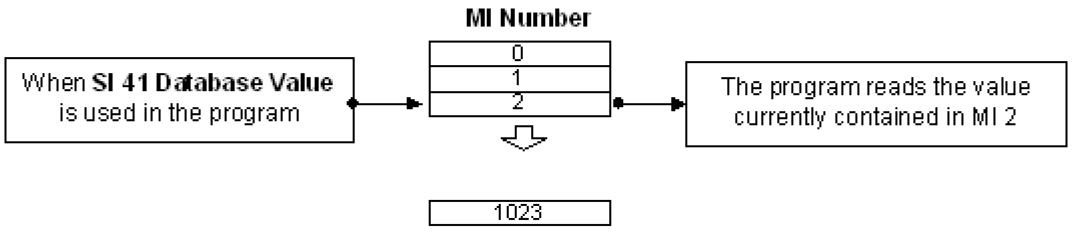

MI Number
The value '2' is stored in SI 40 Database Index
MI 2 is now the current Database integer
0
1
2
1023

2. Use SI 41 Database Value to write a value into MI 2.
   For example, you can store a number value into SI 41.



Reading Values

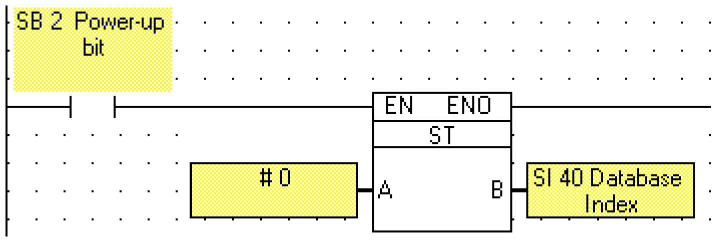
When you use SI 41 Database Value in your program, the program actually reads the MI that is referenced by SI 40 Database Index.


MI Number
When **SI 41 Database Value** is used in the program
0
1
2
1023
The program reads the value currently contained in MI 2

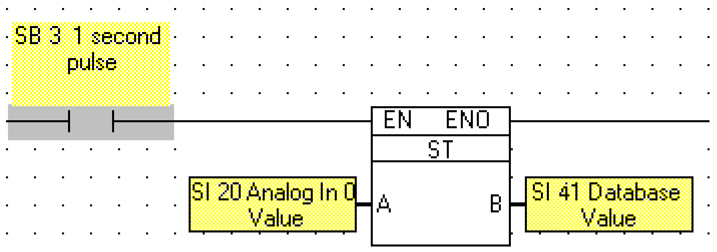
Examples

Example 1: Write

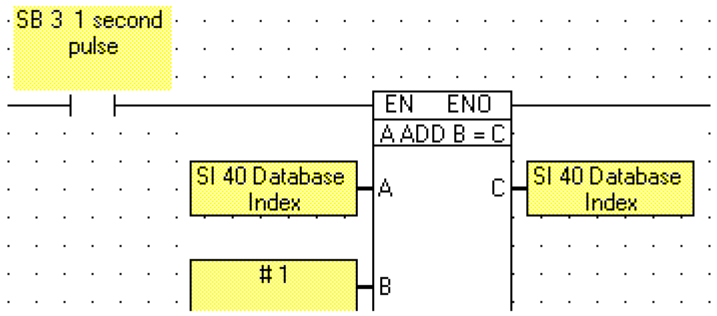
In the net below, 0 is stored in SI 40 when the M90 OPLC is powered up. This means that integer 0 is now the current 'database' integer.



In the net below, the analog value contained in SI 20 is stored in SI 41 every second. According to the net above, the current 'database' integer is 0. The analog value is therefore stored in integer 0.



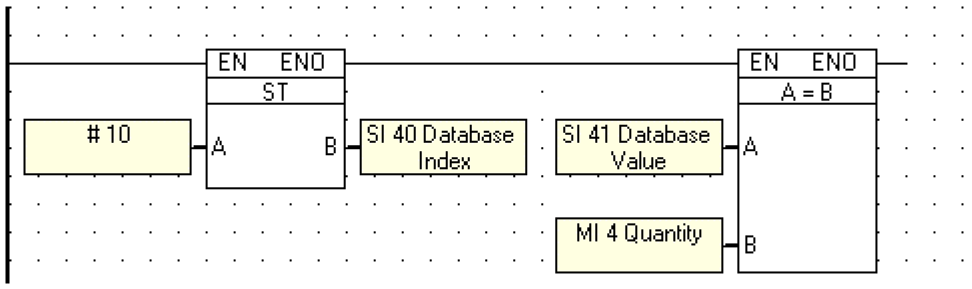
In the next net, the value in SI 40 is incremented by 1every second, changing the current database integer. This means that the first analog value will be stored in integer 0, the second analog value in integer 1, and so on.



Example 2: Read

In the first part of the net below, 10 is stored into SI 40. Integer 10 is the ‘database’ integer. In the second part of the net, the value in SI 41 is compared to the value in integer 4.

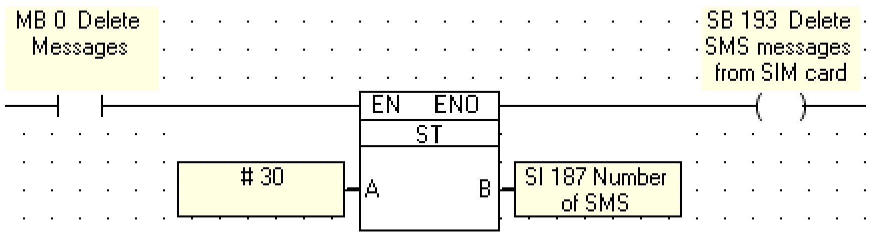
The value in SI 41 is the value actually in integer 10—the current database integer.



Deleting SMS messages

In order to delete SMS messages from a SIM card, turn SB 193, Delete SMS Messages, ON. When used alone, SB 193 will delete 20 messages from the SIM card.

Using SB 193 in conjunction with SI 187, Number of SMS messages to be deleted, enables you to delete up to 30 SMS messages.



Immediate: Read Inputs & HSC, Set/Reset Outputs

You can perform the following immediate actions, without regard to the program scan.

- Set SB 116 to immediately read the status of specific inputs and high-speed counter values. When SB 116 turns ON, the current input value written into linked SBs, current high-speed counter values are written into linked SIs.
- Set the appropriate SBs to immediately clear high-speed counter values.
- Set the appropriate SBs to immediately Set/Reset Outputs.

Note that:

- Values are stored in linked SBs and SIs according to your controller model.
- In the Ladder, inputs and high-speed counters retain the values updated at the beginning of the scan. Only the linked operands listed below are immediately updated. However, immediate changes in output status are immediately updated in the Ladder.

Use the table below to determine which actions, SBs, and SIs are relevant to your model controller.

M90 Model	Input #	Value stored in:	HSC #	Value stored in	HSC #	Immediate Clear	Output #	Set/ Reset via:
M90-T	I 6 I 7	SB 112 SB 113	HSC 0	SI 44	HSC 0	SB 117	None	
M90-T1 M90-T1-CAN	I 8 I 9 I 10 I 11	SB 110 SB 111 SB 112 SB 113	HSC 0	SI 44	HSC 0	SB 117	O 8 O 9 O 10 O 11	SB 120 SB 121 SB 122 SB 123
M90-19-B1A M90-R1 M90-R1-CAN M90-R2-CAN M90-TA2-CAN	I 8 I 9	SB 112 SB 113	HSC 0	SI 44	HSC 0	SB 117	None	

M91-19-TC2 **M91-19-UN2** **M91-19-T1**	**I 0** **I 1** **I 2** **I 3**	SB 110 SB 111 SB 112 SB 113	HSC 0 HSC 1	SI 44 SI 45	HSC 0 HSC 1	SB 117 SB 118	**O 0** **O 1** **O 10** **O 11**	SB 120 SB 121 SB 122 SB 123
M91-19-R1 **M91-19-R2** **M91-19-R2-CAN**	**I 0** **I 1** **I 2** **I 3** **I 4** **I 5**	SB 110 SB 111 SB 112 SB 113 SB 114 SB 115	HSC 0 HSC 1 HSC 2	SI 44 SI 45 SI 46	HSC 0 HSC 1 HSC 3	SB 117 SB 118 SB 119	**O 0** **O 1** **O 2**	SB 120 SB 121 SB 122
M91-19-T38	**I 0** **I 1** **I 2** **I 3**	SB 110 SB 111 SB 114 SB 115	HSC 0 HSC 1	SI 44 SI 46	HSC 0 HSC 1	SB 117 SB 119	**O 0** **O 1** **O 10** **O 11**	SB 120 SB 121 SB 122 SB 123
M91-19-UA2	**I 0** **I 1**	SB 110 SB 111	HSC 0	SI 44	HSC 0	SB 117	**O 0** **O 1**	SB 120 SB 121
M91 19 T2C	**I 0** **I 1** **I 2** **I 3** **I 4** **I 5**	SB 110 SB 111 SB 112 SB 113 SB 114 SB 115	HSC 0 HSC 1 HSC 2	SI 44 SI 45 SI 46	HSC 0 HSC 1 HSC 2	SB 117 SB 118 SB 119	**O 0** **O 1** **O 10** **O 11**	SB 120 SB 121 SB 122 SB 123
M91_19_R6C	**I 0** **I 1**	SB 112 SB 113	HSC 0	SI 45	HSC 0	SB 118	**O 0** **O 1** **O 2**	SB 120 SB 121 SB 123

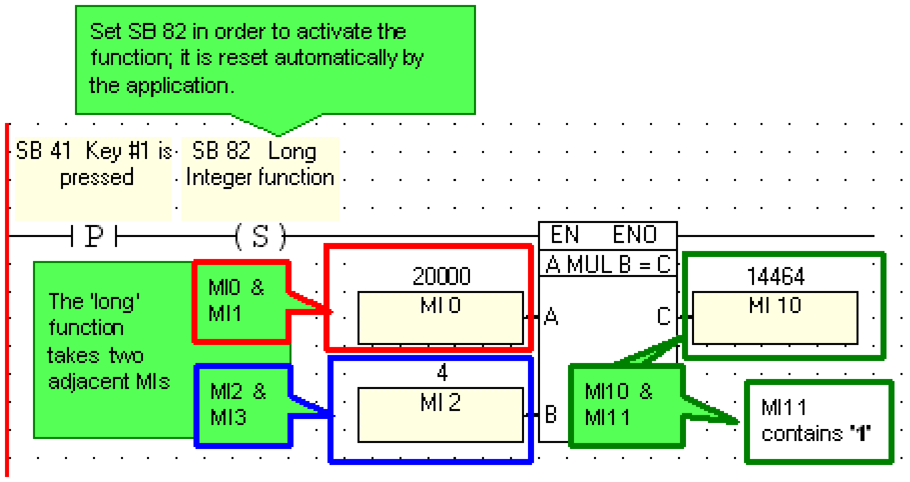
'Long' Integer functions

This special function is supported by M91 controllers alone(OS 91). Note that constant values are not supported; only MI value may be used.

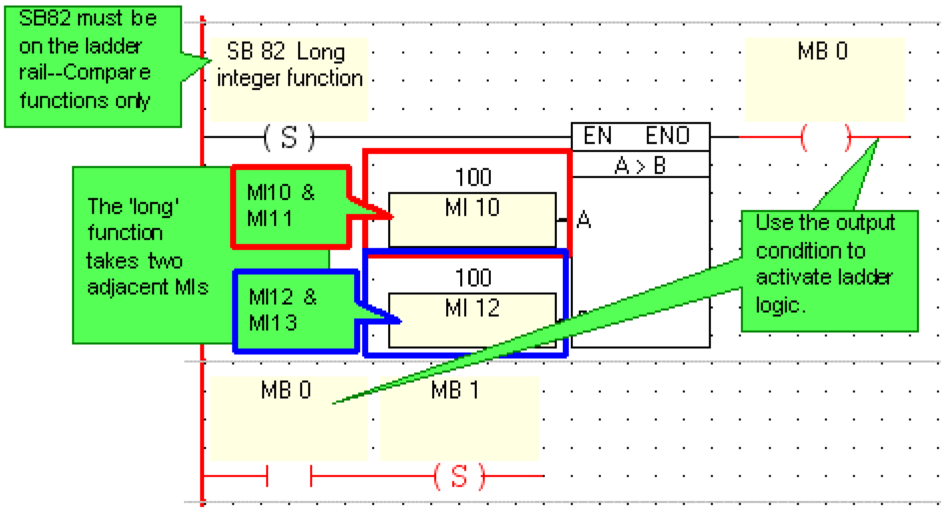
Long integer functions are activated via SB82. A long integer function uses adjacent MIs in performing calculations and storing results. When SB82 is used as the activating condition for a Math, Compare, or Store function, selecting a single MI as an input value causes the following MI to be included with the input. The selected MI serves as the 'low byte' of the long register, and the following MI serves as the 'high byte'. The same logic holds for the output value.

In the example below, the values in MI0 and MI1 provide the 'A' input, MI2 and MI3 provide the 'B' input. Note that MI0 is the 'low byte' of input 'A' and MI1 is the 'high byte'

The result is stored in MI 10 (low) and 11 (high).

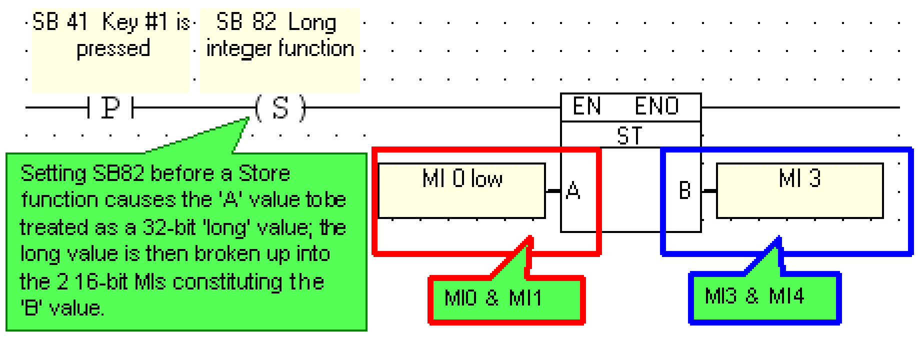

Set SB 82 in order to activate the function; it is reset automatically by the application.
SB 41 Key #1 is pressed
SB 82 Long Integer function
P
S
EN ENO
A MUL B = C
20000
MI 0
4
MI 2
A
B
C
14464
MI 10
The 'long' function takes two adjacent MIs
MI0 & MI1
MI2 & MI3
MI10 & MI11
MI11 contains '1'

In the Compare function below, MI 10 contains 100, MI 11 contains 3, MI 12 contains 100, and MI13 contains 0, making the comparison true. Note that to activate Compare functions, SB 82 must be on the left ladder rail. This is **not** so for Math and Store functions.


SB82 must be on the ladder rail--Compare functions only
SB 82 Long integer function
S
EN ENO
A > B
MB 0
100
MI 10
100
MI 12
A
The 'long' function takes two adjacent MIs
MI10 & MI11
MI12 & MI13
Use the output condition to activate ladder logic.
MB 0
MB 1
S

You can use the Store function in two ways; these can enable you to display long values on the LCD. Note that in order to display long values, the variable used to represent the 'low' byte should be configured to show leading zeros. Display is restricted to positive values within the range of 0-99,999,999.

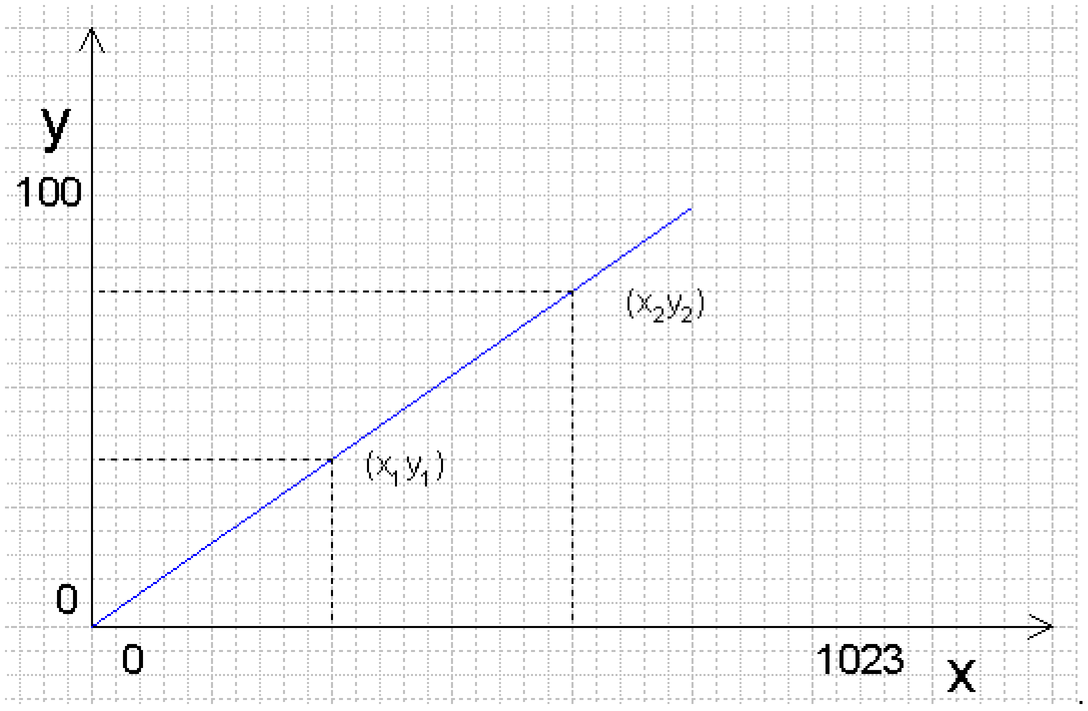
Setting SB82 before a Store function causes the 'A' value to be treated as a 32-bit 'long' value; the long value is then broken up into the 2 16-bit MIs constituting the 'B' value.


SB 41 Key #1 is pressed
SB 82 Long integer function
P
S
EN ENO
ST
MI 0 low
A
B
MI 3
Setting SB82 before a Store function causes the 'A' value to be treated as a 32-bit 'long' value; the long value is then broken up into the 2 16-bit MIs constituting the 'B' value.
MI0 & MI1
MI3 & MI4

Resetting SB82 before a Store function causes the 'A' value to be treated as a 2 16-bit values; the values are then stored as a long 32-bit 'B' value.

Linearization

Linearization can be used to convert analog values from I/Os into decimal or other integer values. An analog value from a temperature probe, for example can be converted to degrees Celsius and displayed on the controller's display screen.



Linearize values for Display

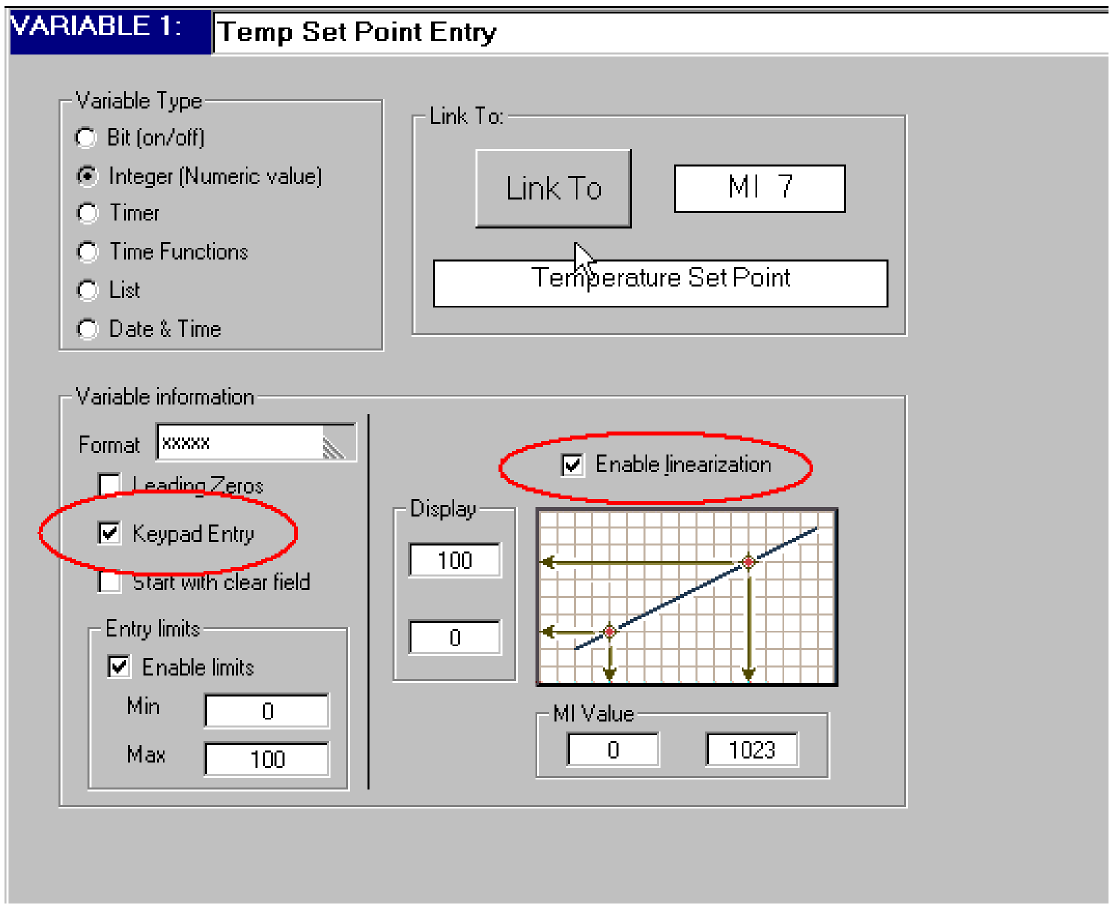
Note that the linearized value created in this way may be displayed-- **but** the value **cannot** be used anywhere else within the project for further calculations or operations.

You can enter an Analog value, such as temperature, via the M90 keypad, then convert that value into a Digital value for comparison with a digital value from a temperature probe by selecting **Enable Linearization** in the linked Variable.

This conversion process is Reverse Linearization.

To enable Analog to Digital conversion:

1. Create a Display for entering the analog value.
2. Create an Integer Variable.
3. Select **keypad entry** and **enable linearization**.
4. Enter the linearization values for the x and y axes.



VARIABLE 1: Temp Set Point Entry

Variable Type: Bit (on/off) [✓ Integer (Numeric value)] Timer Time Functions List Date & Time

Link To: Link To | MI 7 | Temperature Set Point

Variable information

Format: xxxxx

☐ Leading Zeros
☑ Keypad Entry
☐ Start with clear field

Entry limits: ☑ Enable limits

Min	0
Max	100

☑ Enable linearization

Display	MI Value
100	1023
0	0

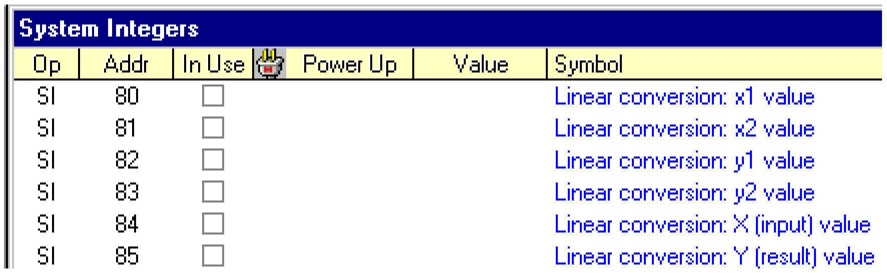
According to the above example:

- A temperature entry of 100^{0} C will be converted to 1023 Digital value.
- A temperature entry of 50^{0} C will be converted to 512 Digital value.

Linearize values in the Ladder

You can also linearize values in your Ladder and display them on the M90's LCD.

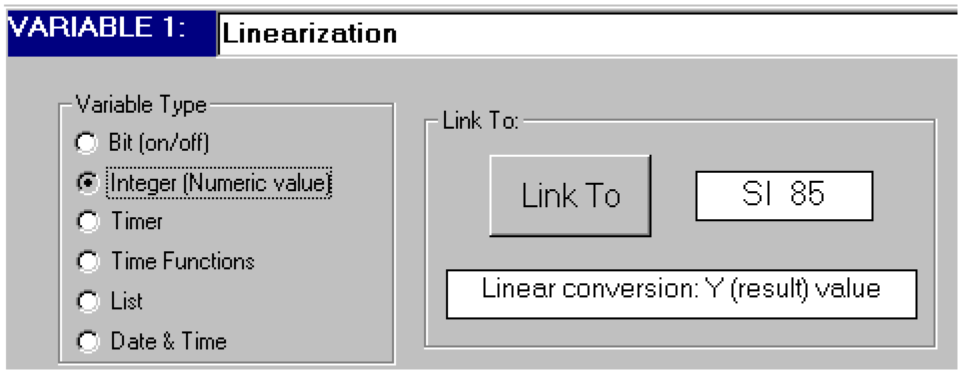
1. In your Ladder project, use SI 80 - 85 to set the (x,y) variable ranges. Use SB 80 to activate the **Linearization** function.



System Integers

Op	Addr	In Use	Power Up	Value	Symbol
SI	80	☐			Linear conversion: x1 value
SI	81	☐			Linear conversion: x2 value
SI	82	☐			Linear conversion: y1 value
SI	83	☐			Linear conversion: y2 value
SI	84	☐			Linear conversion: X (input) value
SI	85	☐			Linear conversion: Y (result) value

The linearization values created here can be displayed by linking SI 85 to a Display; the value **can** be used elsewhere within the project for further calculations or operations.



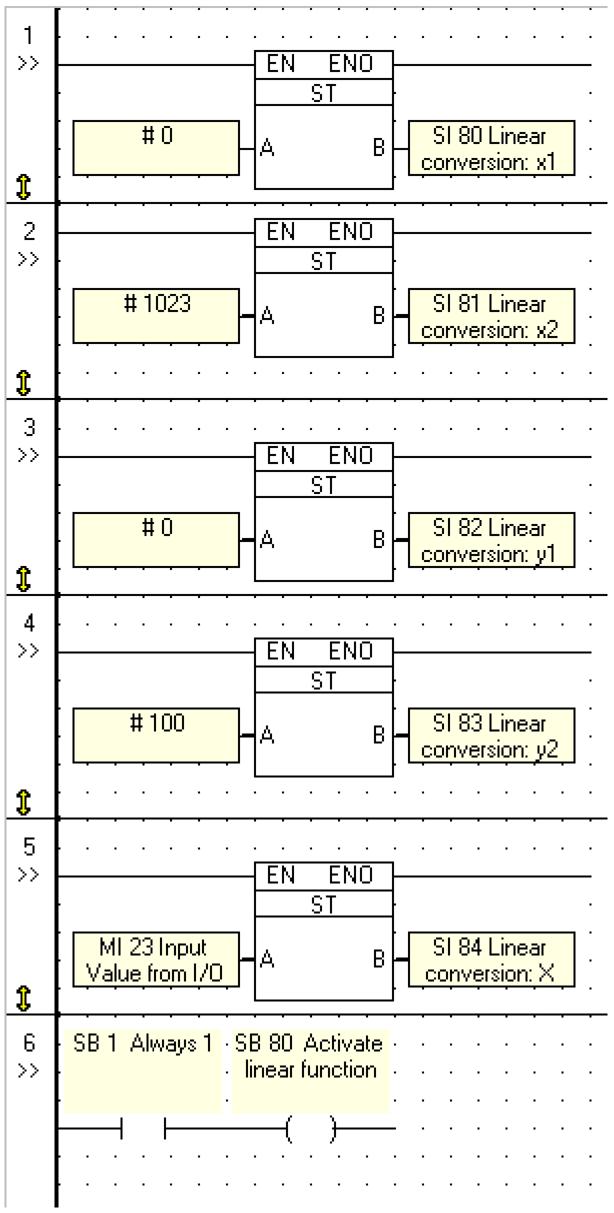
VARIABLE 1: Linearization

Variable Type: Integer (Numeric value)

Link To: SI 85

Linear conversion: Y (result) value

Example: write the variable ranges into SI 80 - 83, then writing an analog input into SI 84:


1
EN ENO
ST
# 0
A
B
SI 80 Linear conversion: x1
2
# 1023
SI 81 Linear conversion: x2
3
# 0
SI 82 Linear conversion: y1
4
# 100
SI 83 Linear conversion: y2
5
MI 23 Input Value from I/O
SI 84 Linear conversion: X
6
SB 1 Always 1
SB 80 Activate linear function

Shift Register

You can use the following SIs and SBs to perform Shift Left and Shift Right Functions.

SI	Symbol	Description
87	Shift Value	This register contains the number to be shifted.
88	Shift By	This register contains the number of bits to be shifted (Default is 1 bit).

SB	Symbol
87	Shift Left
88	Shift Right

Example : Shift Left

To shift the number 64 **left** by 1 bit:

1. Use a Store function to write the number 64 into SI 87.
2. Use a Store function to write the number 1 into SI 88.
3. Turn SB 87 ON.

Once the function is performed SI 87 will contain 128.

```
In binary:
Start value:        0000000001000000  = 64
After Shift Left :   0000000010000000  =128
```

Example : Shift Right

To shift the number 64 **right** by 1 bit:

1. Use a Store function to write the number 64 into SI 87.
2. Use a Store function to write the number 1 into SI 88.
3. Turn SB 88 ON.

Once the function is performed SI 87 will contain 32.

```
In binary:
Start value:        0000000001000000  = 64
After Shift Right:    0000000000100000   =32
```

Display Integer values as ASCII or Hexadecimal

You can:

- Display the values in an MI vector as ASCII characters.
- Display a register value in hexadecimal format.

To do this, attach a numeric Variable to a Display. The variable uses linearization to display the value(s) in the desired format.

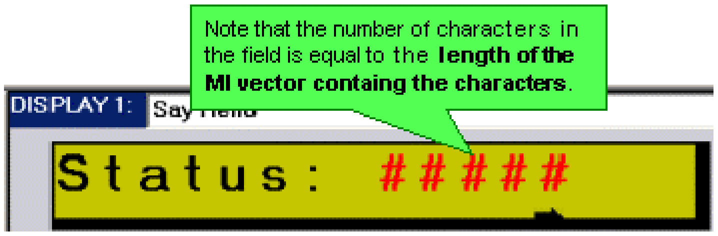
Note that non-supported ASCII characters will be shown as <space> characters.

ASCII -Hexadecimal character table

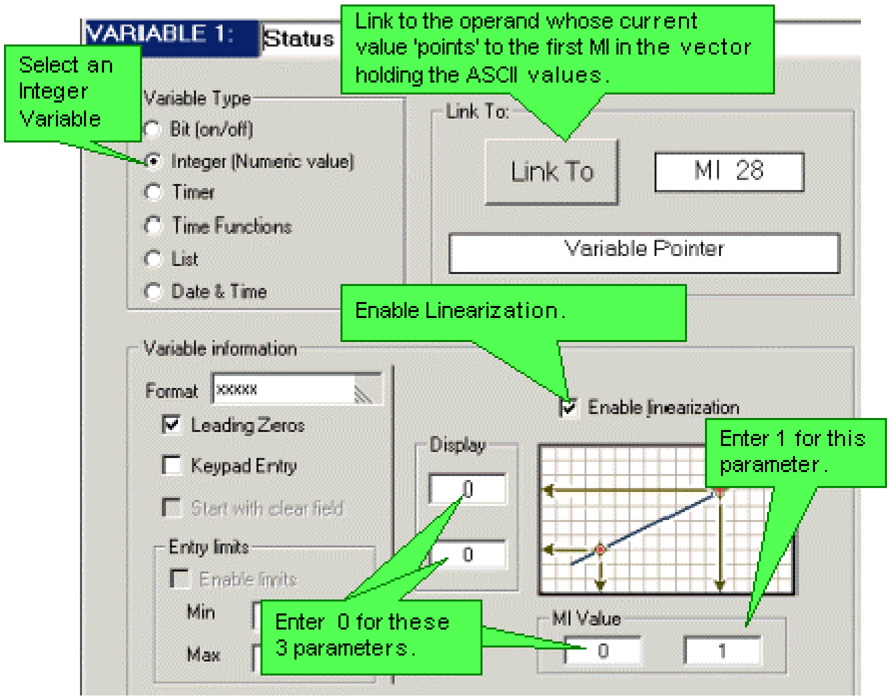
Vector as ASCII

When the application shown in the example below is downloaded, the ASCII characters 'Hello' will be displayed on the M90 screen when Key #3 is pressed.

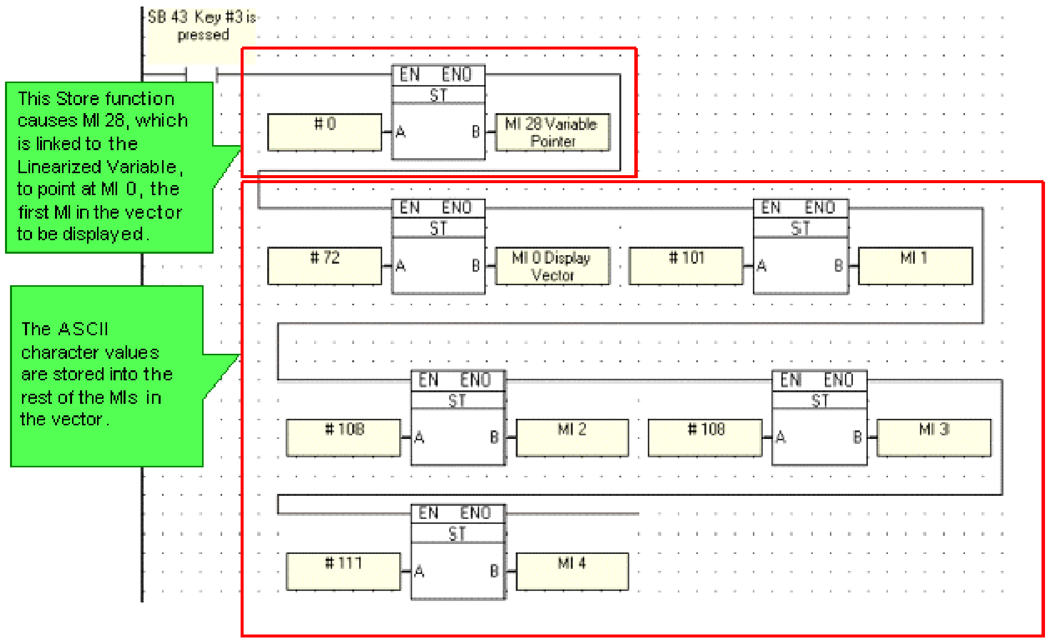
1. Create a Variable Field in a Display, then attach a Variable.


Note that the number of characters in the field is equal to the **length of the MI vector containing the characters**.
DISPLAY 1:
Say Hello
Status: #####

2. Define the Variable as shown below.



3. The Ladder net below sets the Variable pointer and stores ASCII values into the MI vector.

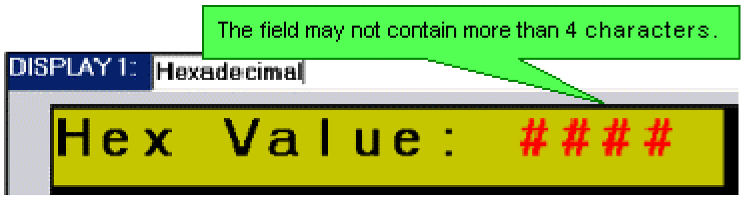

SB 43 Key #3 is pressed
This Store function causes MI 28, which is linked to the Linearized Variable, to point at MI 0, the first MI in the vector to be displayed.
The ASCII character values are stored into the rest of the MIs in the vector.
EN ENO ST A B
# 0
MI 28 Variable Pointer
# 72
MI 0 Display Vector
# 101
MI 1
# 108
MI 2
# 108
MI 3
# 111
MI 4

When the application used in this example is downloaded, the ASCII characters 'Hello' will be displayed.
M-90
Status: Hello

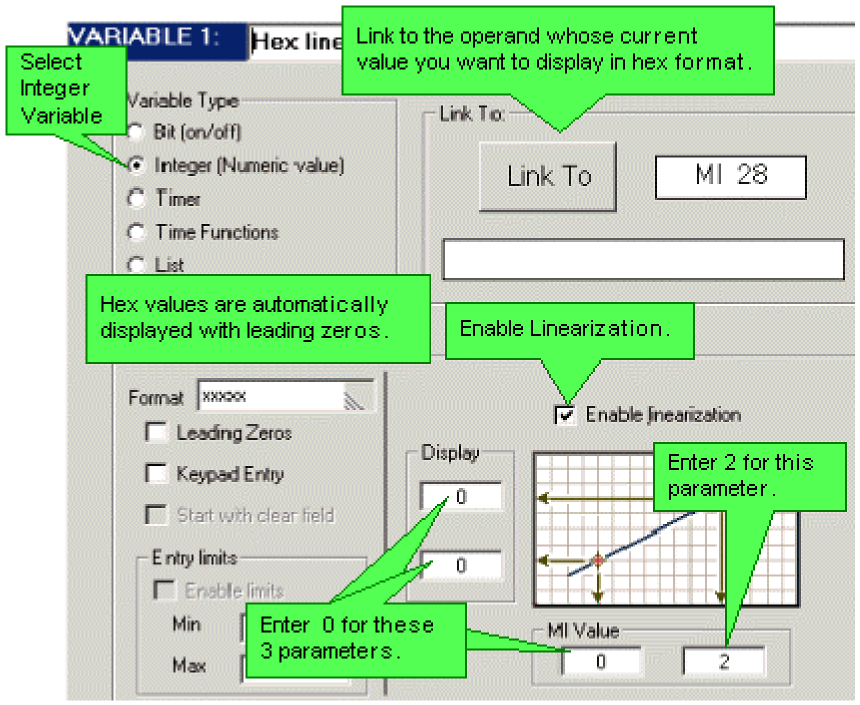
Register Value in Hexadecimal

When the application shown in the example below is downloaded, the hexadecimal value of 63 will be displayed on the M90 screen.

1. Create a Variable Field in a Display, then attach a Variable. Note that if the field is too short, only the right-most characters are displayed. For example, the hex value 63(3F) cannot be shown in a field one character long.



2. Define the Variable as shown below.



Select Integer Variable

Link to the operand whose current value you want to display in hex format.

MI 28

Hex values are automatically displayed with leading zeros.

Format xxxxx

Enable Linearization.

[✓] Enable linearization

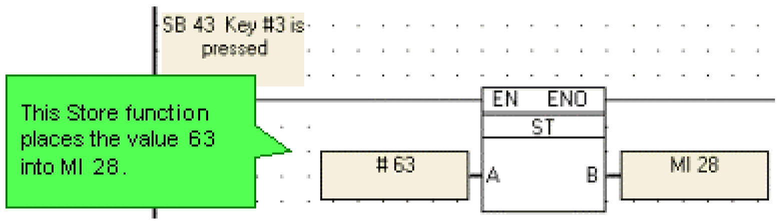
Enter 2 for this parameter.

Enter 0 for these 3 parameters.

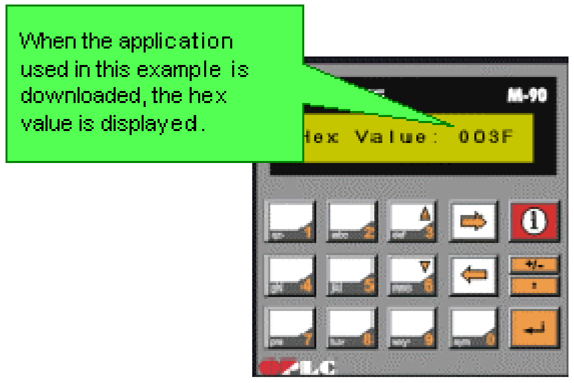
Display: 0, 0

MI Value: 0, 2

3. The Ladder net below stores the value into the MI.



This Store function places the value 63 into MI 28.



When the application used in this example is downloaded, the hex value is displayed.

Index

3
32-bit long values 36
A
ASCII 7, 42
Average 13
C
communications 5
Compare 36
convert MB-MI.. 29
Copy vector...................................... 10, 30

D
Database 32
Display 42
Display variable data 42
F
factor 4
Fill Vector 11
Find 13
Functions 4, 5, 10, 11, 13, 15, 27, 28, 30, 32, 36, 41
G
GSM 14
H
Hexadecimal 42
HMI 25
hsc 35
I
I/O 30, 35
indirectly addressed registers 32
Input 30, 35
K
Keypad 25
L
Linearization 38
Load 16, 17
Long 36
M
Math 36
Math Functions 4, 36
Maximum 13
Mean 13
Memory Integer 36
Miniumum 13
modem 14, 35
O
Outputs 30
P
PIN code 14
S
Shift Register 41
SMS 5, 25, 35
Store 27, 36
Store Direct 27
T
Timers 27