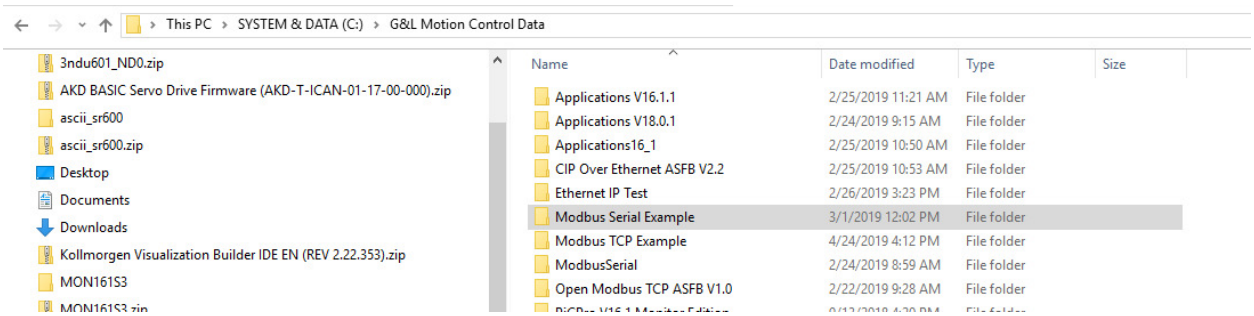
G&L Modbus Serial Example Revision A 6-3-2019

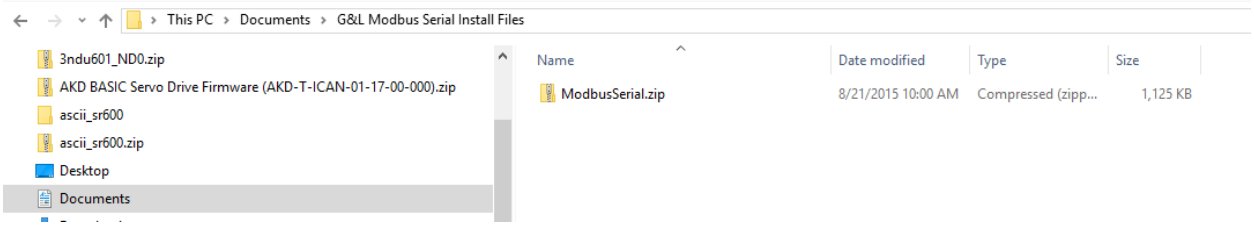
Note!! This example was done with the Digital MMC Smartdrive and Drive Resident Control 16 Axis. The programmer is responsible for any settings and wiring that are different due to differences in hardware.

Also note that the Modbus Serial ASFB library is not free-ware and must be purchased from your local Kollmorgen supplier.

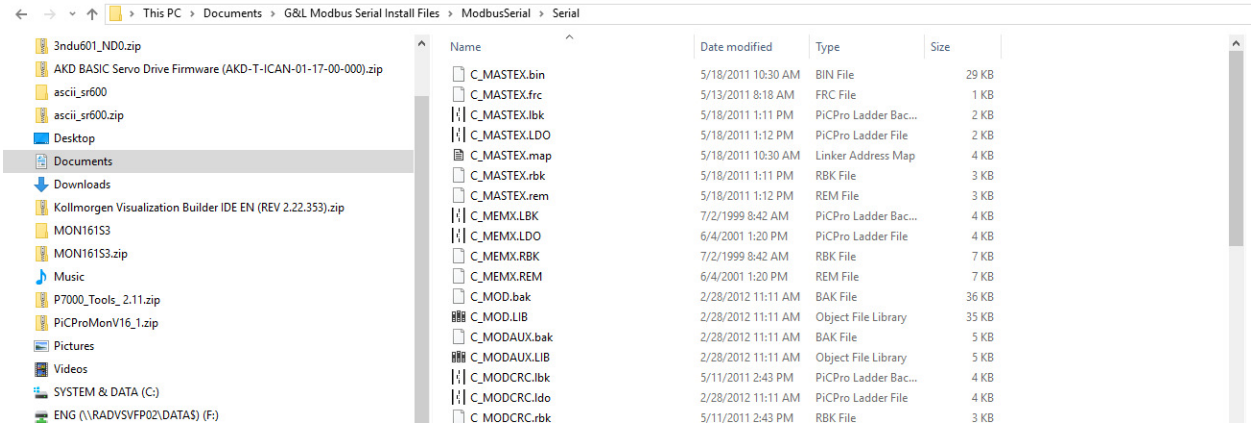
First I created a folder called Modbus Serial Example under the C:\G&L Motion Control Data directory. This will be used later in this application note.



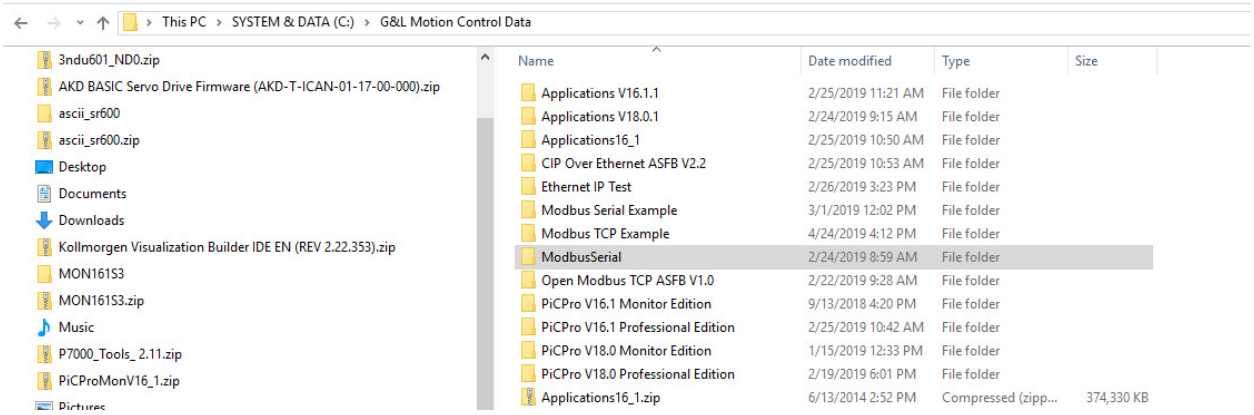
I put the Modbus Serial zip file in a folder I created to allow the files to be unzipped to the new folder.



Unzipping and viewing the contents there is a folder called “ModbusSerial” and a subfolder called “Serial” which contains a large number of files which will be explained shortly.



I copied the entire ModbusSerial folder and pasted it into the C:\G&L Motion Control Data for each of location.



I created a new subfolder called ModbusSerial_ASFB in the C:\G&L Motion Control Data folder to group the required files using the method described in the manual shown below.

Remember the *.LIB files and *.LDO source files must reside in the same folder.

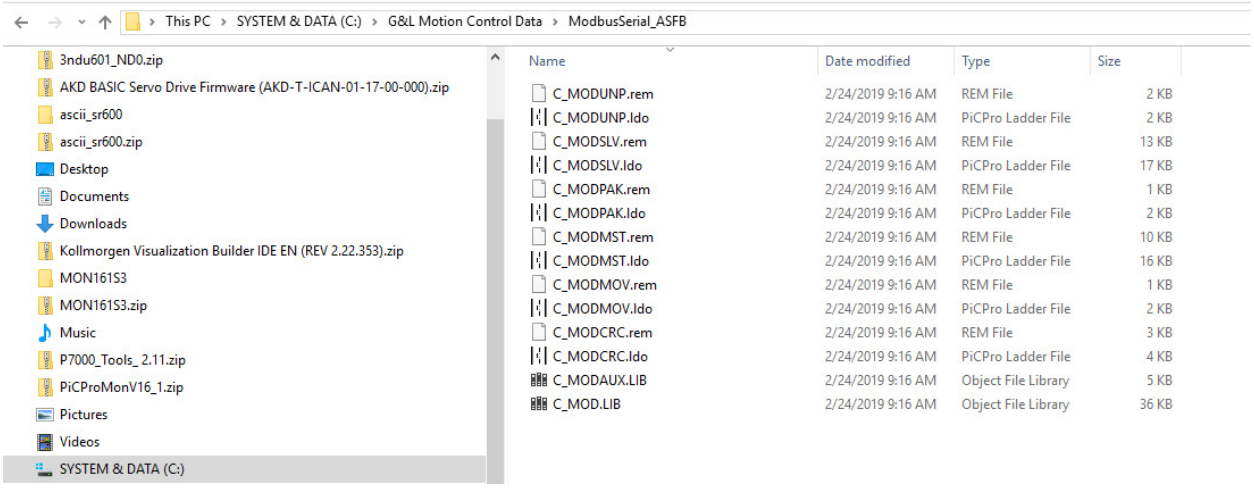
1.3 Installation

The **Modbus** software disk contains the files listed below. The Main group includes the ASFB library (LIB), source ladders for the ASFBs **(LDOs),** and remark files containing the comments in the source ladders **(.REMs).** The Example group includes the example LDO and REM files. The Auxiliary group contains the LIB, **LDOs,** and **REMs** for the **UDFBs** used in the source ladders for the ASFBs. NOTE: It should never be necessary for you to access any of the files in the Auxiliary group. The LIB is required in order for the ASFB to work and the **LDOs** allow you to view the source ladders when troubleshooting if necessary.

Follow the guidelines found at the beginning of the manual. Always make a back up copy of the disk and store the original in a safe place. The recommended destination' directory for each file is listed in the last column.

Group	File	Description	Directory
Main	C_MOD.LIB	The library containing the application specific function block used to perform Modbus communications.	ASFB
	C_MODSLV.LDO	The source ladder for the transceiver function block.	ASFB
	C_MODSLV.REM	The remark file for the source ladder	ASFB
	C_MODMST.LDO	The source ladder for the C_MODMST function block.	ASFB
	C_MODMST.REM	The remark file for the source ladder	ASFB
Example	C_MASTEX.LDO C_MASTEX.REM	The example for Modbus master LDO from which you can build a new application LDO or to which you can merge an existing one.	Working
	C_MODEX.LDO C_MODEX.REM	The example for Modbus slave LDO from which you can build a new application LDO or to which you can merge an existing one.	Working
Auxiliary	C_MODAUX.LIB	The library that holds all the function blocks used in the source ladder for the ASFB.	ASFB
	C_MODCRC.LDO C_MODCRC.REM	Source ladder Remark file	ASFB ASFB
	C_MODMEM.LDO C_MODMEM.REM	Source ladder Remark file	ASFB ASFB
	C_MODMOV.LDO C_MODMOV.REM	Source ladder Remark file	ASFB ASFB
	C_MODPAK.LDO C_MODPAK.REM	Source ladder Remark file	ASFB ASFB
	C_MODUNP.LDO C_MODUNP.REM	Source ladder Remark file	ASFB ASFB

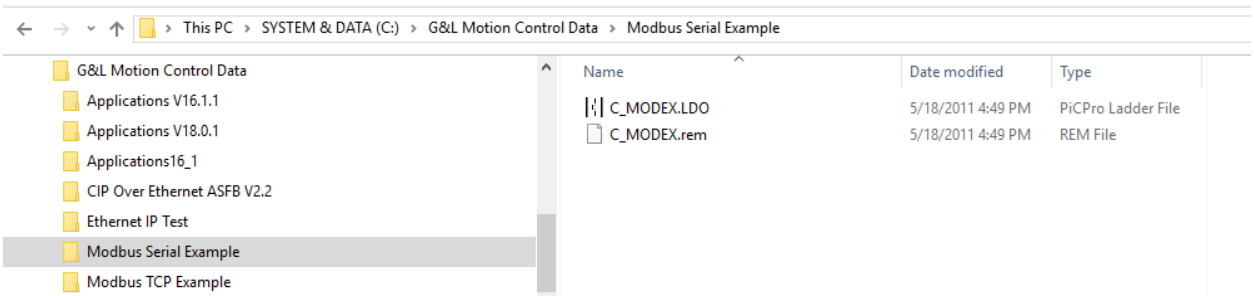
The 14 files for the “ASFB” folder are shown below.



In this application the Digital MMC Drive Resident Control will be the Modbus Slave.

Example	C_MASTEX.LDO C_MASTEX.REM	The example for Modbus master LDO from which you can build a new application LDO or to which you can merge an existing one.	Working
	C_MODEX.LDO C_MODEX.REM	The example for Modbus slave LDO from which you can build a new application LDO or to which you can merge an existing one.	Working

I saved the C_MODEX.LDO and C_MODEX.REM files to the “Working” directory which will be the Modbus Serial Example folder created at the beginning of this application note.



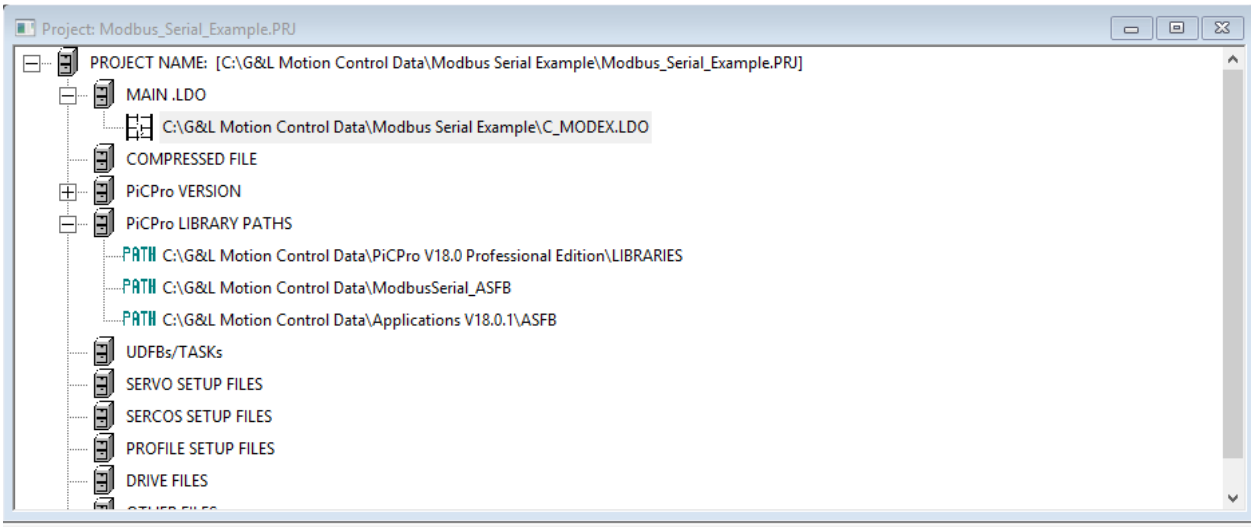
These files can be renamed if desired but I left them as is.

Next I created a project also saved to the Modbus TCP Example folder.

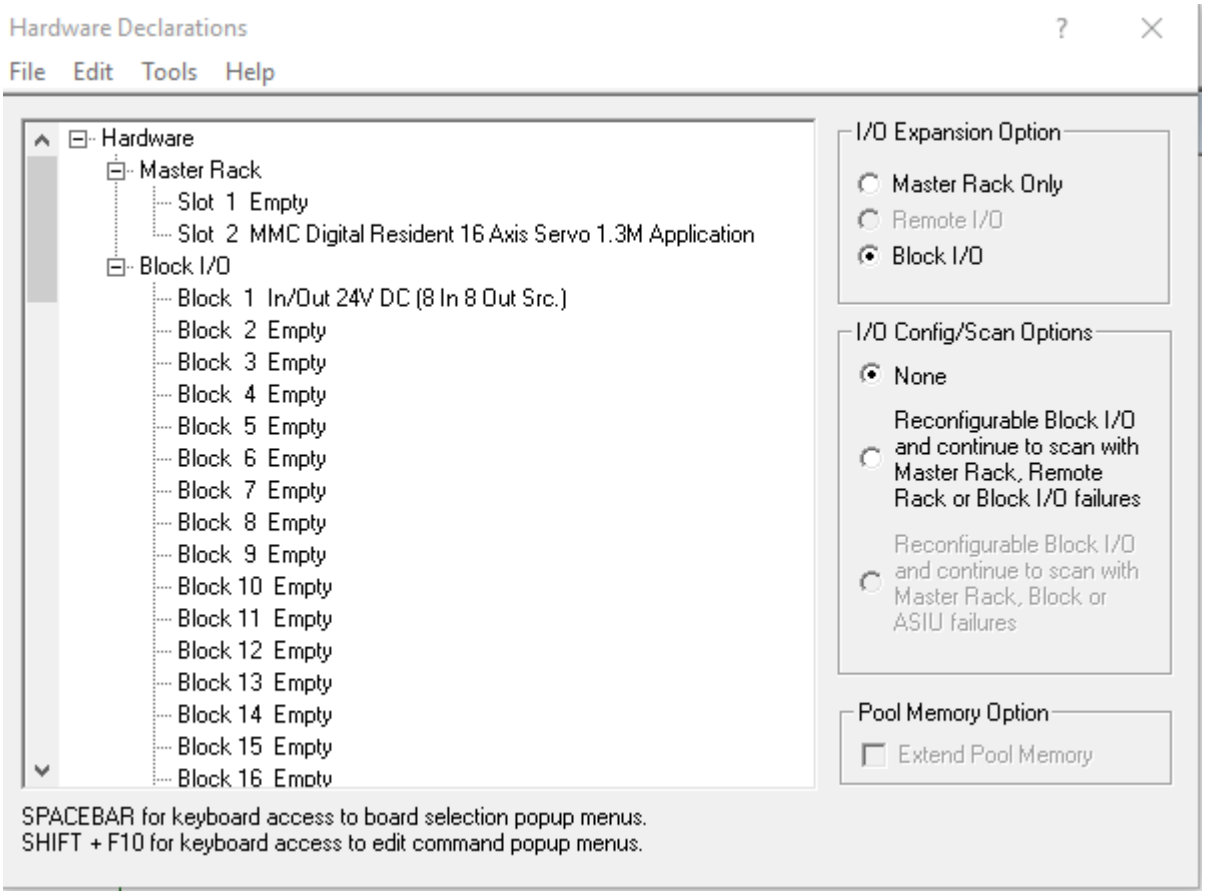
Once the project is created the following paths were setup.

The MAIN.LDO is the C_MODEX.LDO copied into the Modbus Serial Example “working” directory.

The PicPro Library paths point to the standard libraries for this version of PicPro, the ModbusSerial_ASFB folder (created just previously), and the standard ASFB library for this verion of PicPro (part of the Applications disk install).



Under the View pull-down menu->Hardware Declarations I configured this for the hardware of my demo. This will depend on your hardware.



Note there are serial port settings in the Software Declarations.

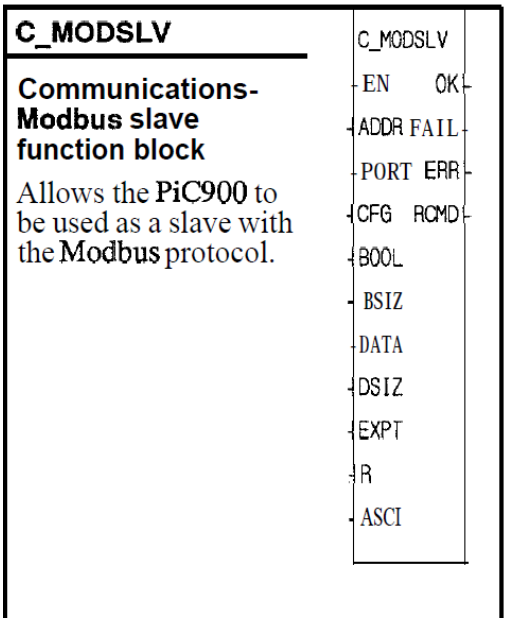
I left them as is. This may vary depending on your hardware.

Software Declarations : Main Ladder - [C_MODEX.LDO]

File Edit Tools Help

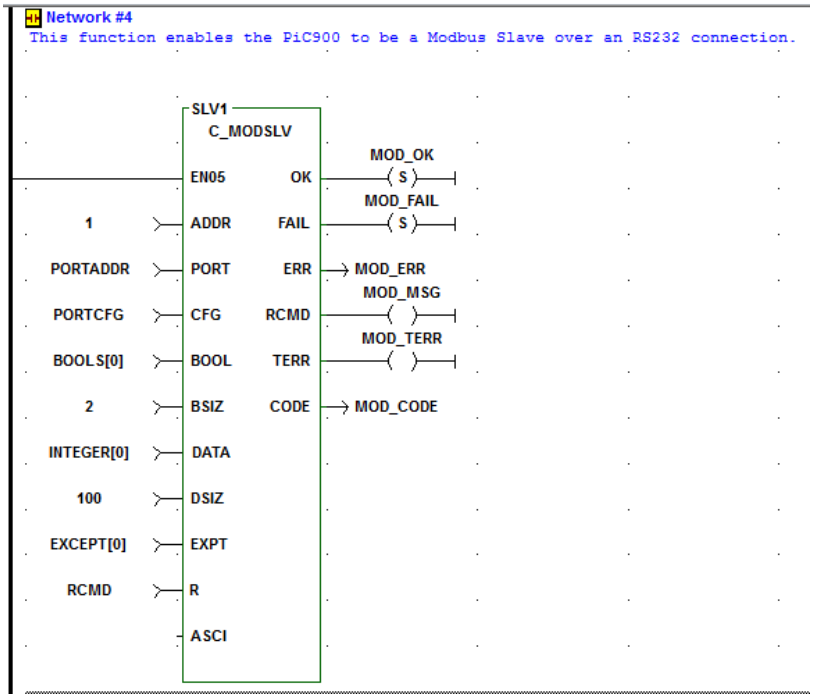
Name	Type	A.	I/O Point	Initial Value	Long Name
REV00	**BOOL**				**Example\.LDO\Revision\Num...**
SLV1	<fb>C_MODSLV				Modbus\Slave\Driver\
BOOLS	BOOL(0..1)				Booleans:\On Modbus:\00001 to...
INTEGER	INT(0..99)				Integers:\On Modbus:\40001 to\...
PORTADDR	STRING[15]			USER:$00	Serial\Port\Name\
PORTCFG	STRING[15]			9600,E,8,1,N$00	Serial\Port\Configure\String
RCMD	STRUCT				Received\Commands\Informatio\...
.ADDRESS	USINT				Address of\Incomming\Comman...
.FUNCTION	USINT				Function\Number of\Incomming\...
	END_STRUCT				
MOD_OK	BOOL				init OK\flag\\
MOD_FAIL	BOOL				init\failure\flag\
MOD_ERR	INT				init\failure\error code\
MOD_MSG	BOOL				message\received\flag\
EXCEPT	BOOL(0..7)				User\Definable\Exception\Status
MOD_TERR	BOOL				trans\action\error flag\

Here is the Modbus Slave function block. Once downloaded we will animate it.



See the Modbus ASFB manual for more details on the I/O of this function block.

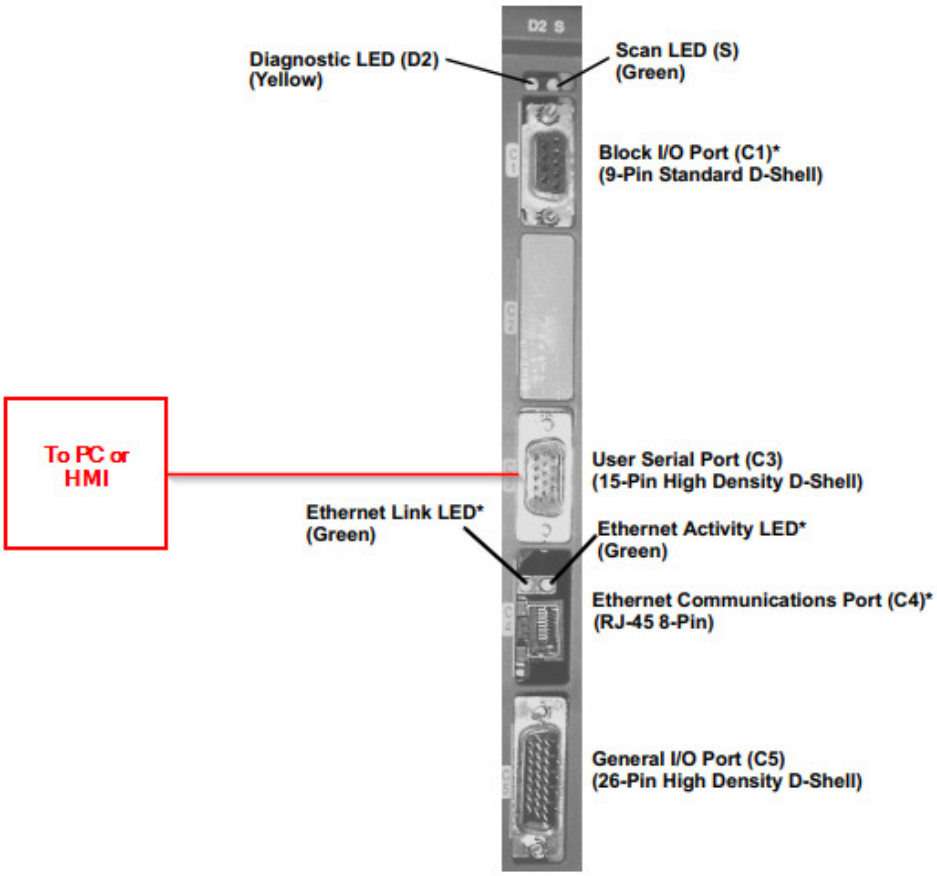
A topic of interest is the input for BOOL (type of data) which starts at BOOLS[0] and the input BSIZ indicates the size of the BOOLS array which is 2. This can be expanded (later).



Prior to downloading the sample ladder here are details on the hardware used in this example and the wiring scheme. Refer to the appropriate hardware/installation manual for your specific hardware.

For serial Modbus the User Serial Port (C3) is used on the Drive Resident Digital MMC Control.

Figure 13-1: The Drive Resident Digital MMC Control



Although the C3 port has both RS232 and RS485 interface capability for this demonstration I used RS232. I used a USB to Serial converter and a DSUB9 connector to DSUBHD15 cable to make the serial connection between my PC and the User Port. Check your hardware manual for your PC or HMI to determine what your connections should be.

PC		User Port	
Pin	Description	Pin	Description
3	TX	9	RX
2	RX	10	TX
5	GND	8	GND

Table 13-9: User Port Pin Assignments

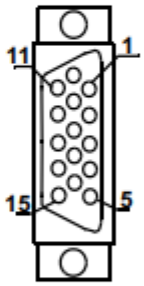
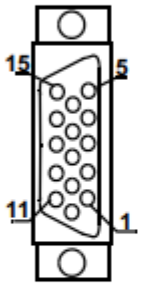
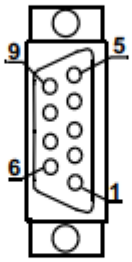
Pin	Signal	In/Out	Connector Pinout
1	NC	N/A	15-pin HD male D-sub 
2	N/C	N/A	
3	N/C	N/A	
4	RS232 Data-terminal-ready (12 Vdc)	Out	
5	RS232 Request-to-Send	Out	
6	N/C	N/A	
7	RS232 Clear- to-Send	In	
8	Signal Ground	In/Out	
9	RS232 Receive Data	In	
10	RS232 Transmit Data	Out	
11	N/C	N/A	
12	RS485 Receive Data +	In	
13	RS485 Receive Data -	In	
14	RS485 Transmit Data +	Out	
15	RS485 Transmit Data -	Out	
Connector Shell	Drain	In	

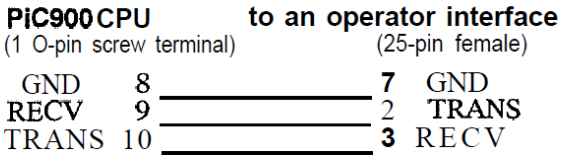
Table 13-11: User Port to RS-232 Exter HMI Cable				
Part Number: M.1302.8453 Length: 4 M (13 ft) Cable type: 24 AWG, shielded, twisted pair, 4 conductor.				
15-Pin HD female D-sub (to User Port, face view) 		9-Pin female D-sub (to Exter HMI COM2 Port, face view) 		
Pin	**Signal**	**Pin**	**Signal**	**Notes**
9	Receive Data	3	Transmit Data	Twisted
10	Transmit Data	2	Receive Data	Pair
8	Signal Ground	5	Signal Ground	
Shell	Drain	Shell	Drain	

This is consistent with what is offered in the Serial Modbus ASFB manual:

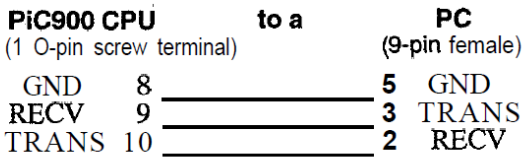
Cable connections

The pinouts for the various Modbus communications connections are shown below. Choose the one for your system.

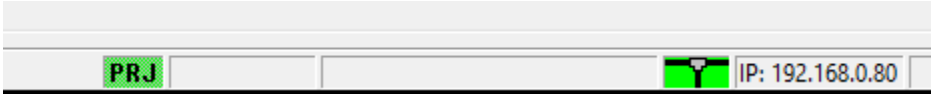
PiC900 to an operator interface



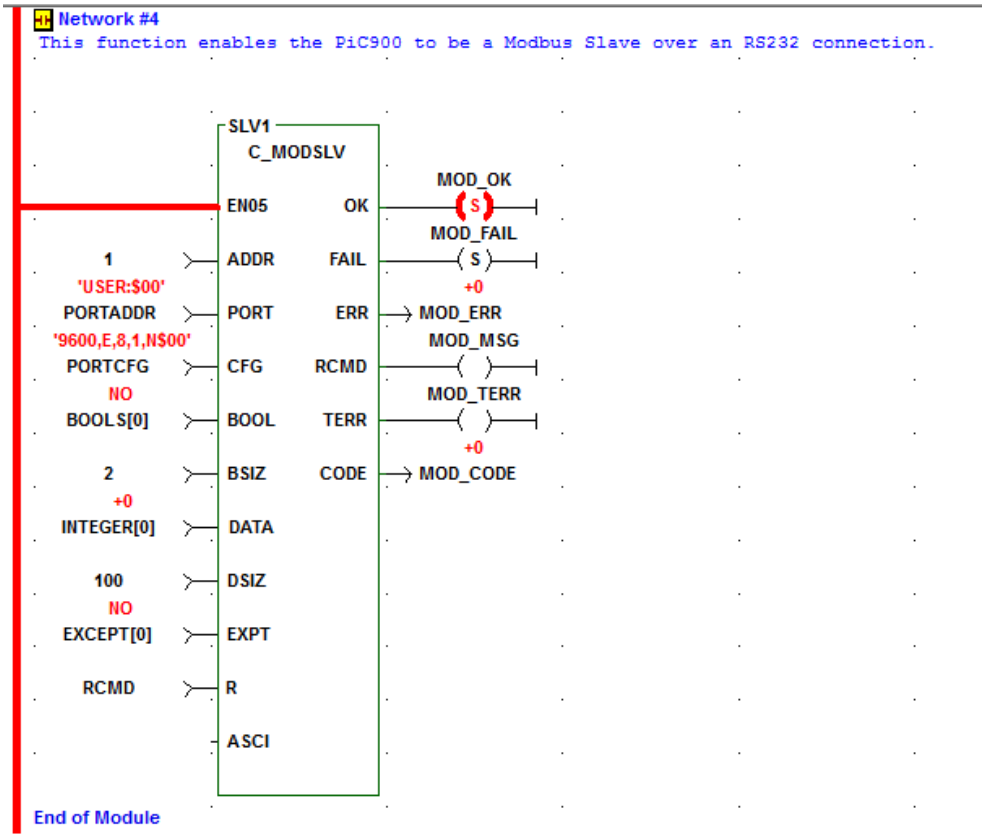
PiC900 to a PC



Finally, I saved the ladder changes and project and then compiled and downloaded it to the control and animated. In my case I used Ethernet to connect to the Digital MMC Drive Resident Control so my serial connection (USB to Serial in my case) would be used for Serial Modbus and to view the PicPro ladder online I used Ethernet.

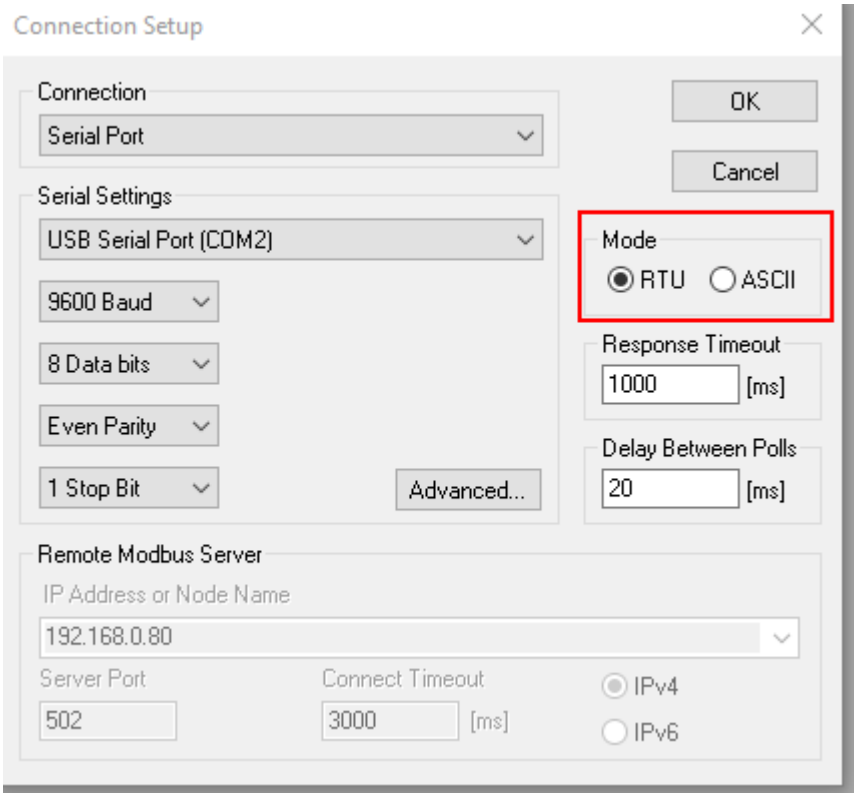


Note the E_MODSVR is enabled, SCON_OK and error outputs are all zero.

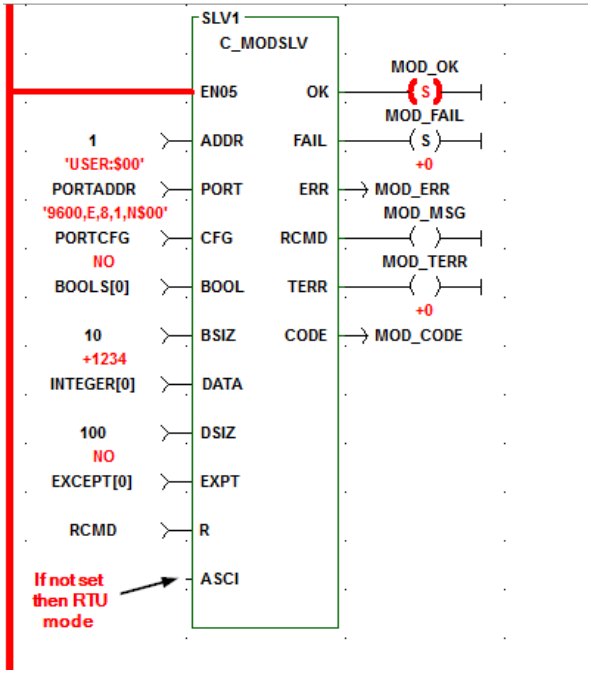


Next using Modbus Poll as the Modbus Serial master and to check if values can be written.

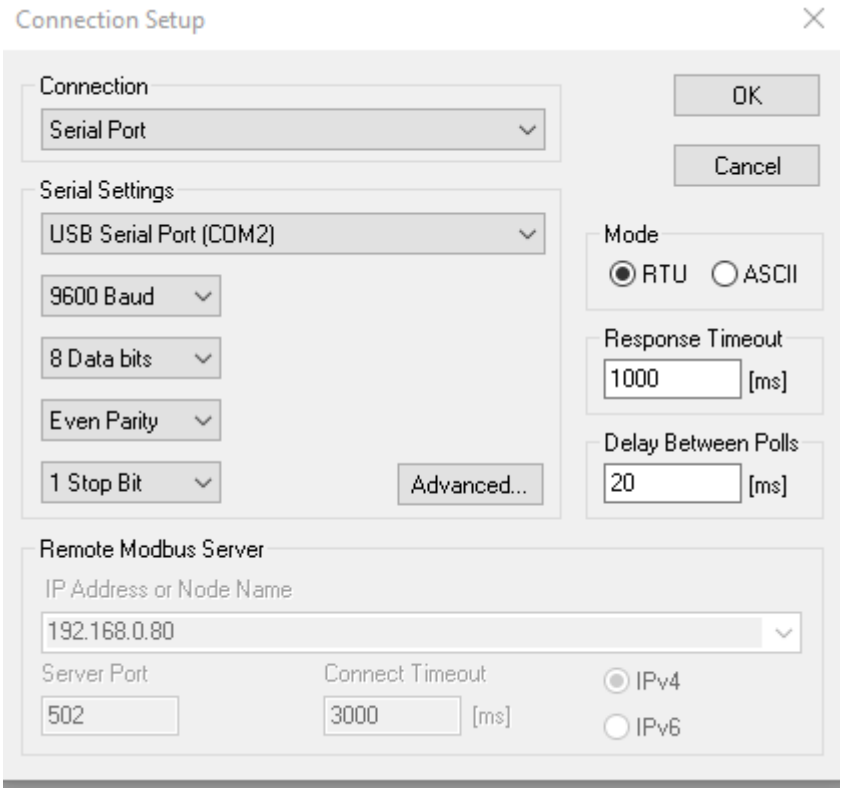
RTU mode is selected as follows:



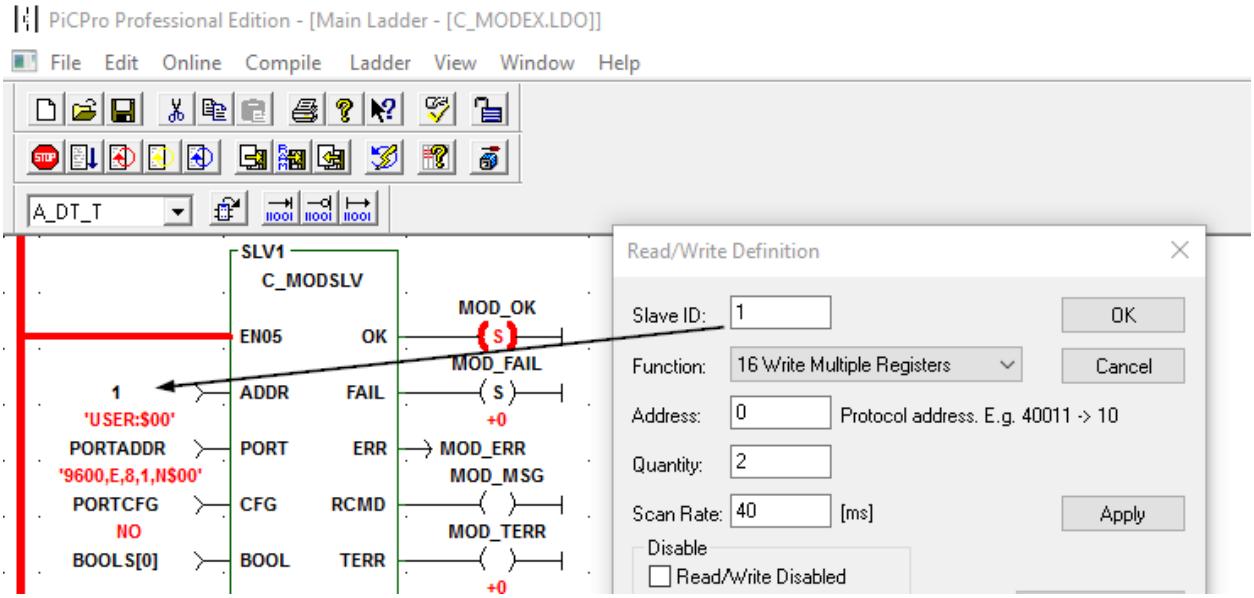
RTU mode is selected by the C_MODSLV block as follows:



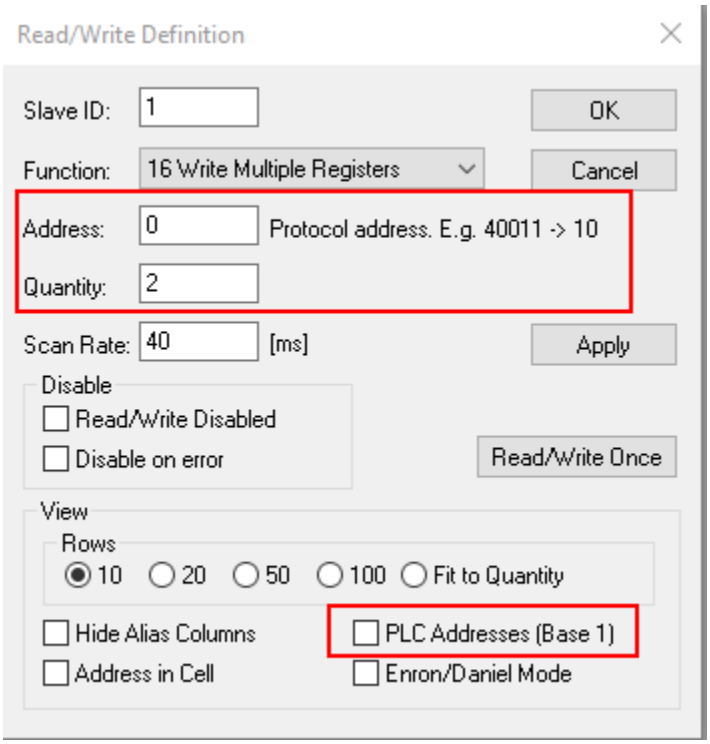
Per the Modbus Slave ASFB the PORTCFG input is for 9600, E, 8, 1 so Modbus Poll is configured with the same settings:



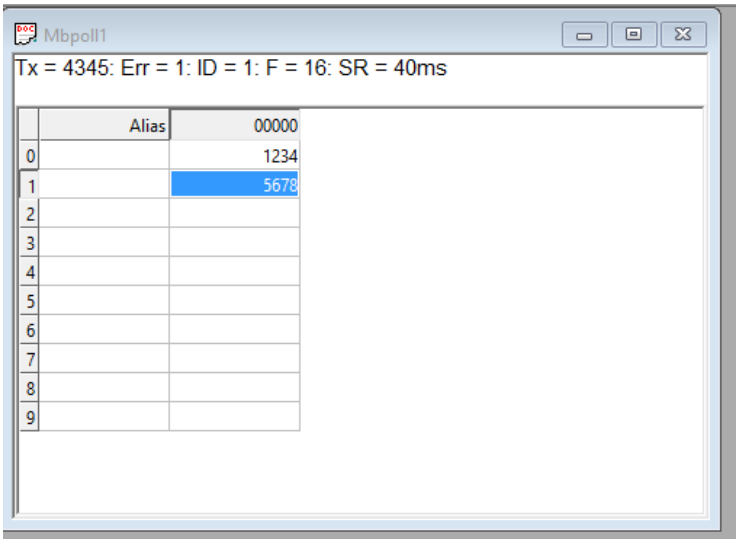
Next In Modbus Poll setup a Read/Write Definition. A very important setting is the Slave ID. This value needs to match the input value of ADDR in the C_MODSLV function block in Picpro. In this example it is 1.



Initially the starting address will be zero and not using Base 1. If you want the convention to follow the chart in the Modbus Serial ASFB manual then the PLC Addresses (Base 1) checkbox needs to be checked. With the first test we will attempt to write to the first to INTEGERS in the G&L.



On connection there are no errors so for demonstration purposes I set the first 2 registers to values 1234 and 5678 respectively.

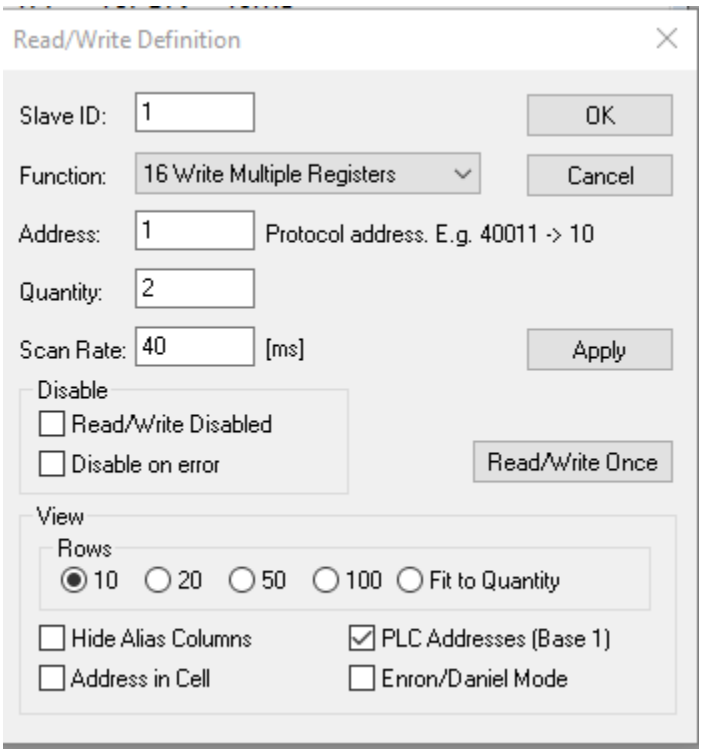


Now switching back to the animated ladder in PicPro I used the View List to monitor the values.

Enable	Name	Type	Value
X	INTEGER[0]	Signed Decimal	+1234
X	INTEGER[1]	Signed Decimal	+5678

Keep in mind some Modbus TCP masters start their address at 1, 40001, 400001, etc. where 1 will likely be equivalent to DATA[0] which is an offset in addressing.

The following yields the same result (Base 1 addressing):



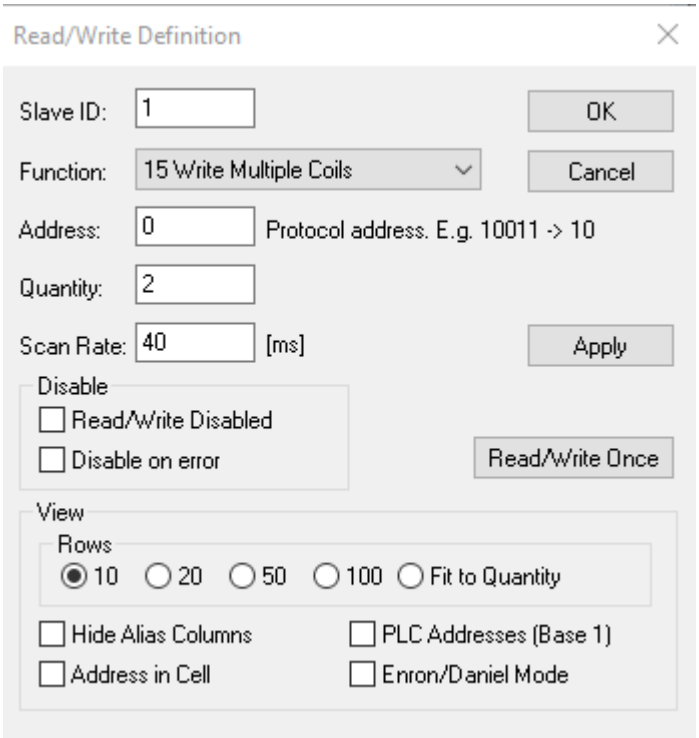
Note the Modbus TCP ASFB manual assumes a base 1 addressing scheme. Also note the array is shown as BOOL(x) in the chart but in the sample project the array is named BOOLS(x). Likewise the integer array is shown as DAT(x) in the chart but INTEGER(x) in the sample project.

Message Addressing

The addressing between the PiC900 and Modbus is as follows:

BOOLEANS		INTEGERS	
Modbus	PiC900	Modbus	PiC900
00001	BOOL(0)	40001	DAT(0)
00002	BOOL(1)	40002	DAT(1)
.		.	.
00999	BOOL(998)	40999	DAT(998)

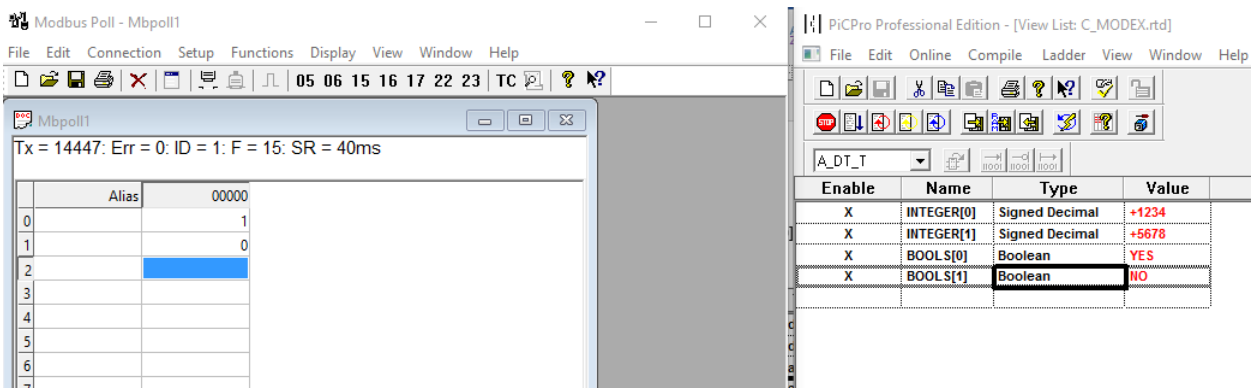
To test the Boolean data I setup Modbus Poll to use function 15-Write Multiple Coils.



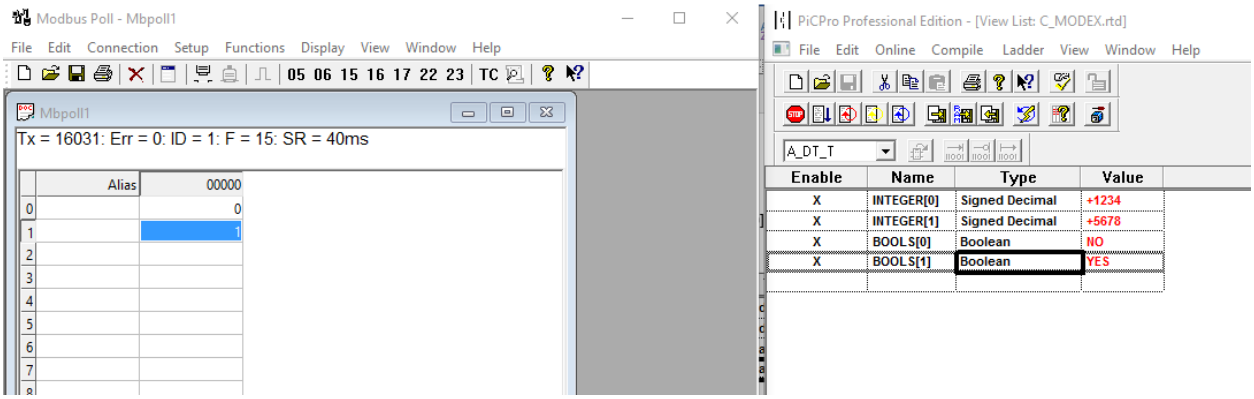
To demonstrate I added BOOLS[0] and BOOLS[1] to the View List in PicPro. NO=0 or OFF and YES=1 or ON.

Enable	Name	Type	Value
X	INTEGER[0]	Signed Decimal	+1234
X	INTEGER[1]	Signed Decimal	+5678
X	BOOLS[0]	Boolean	NO
X	BOOLS[1]	Boolean	NO

Turning on BOOLS[0] in Modbus Poll:



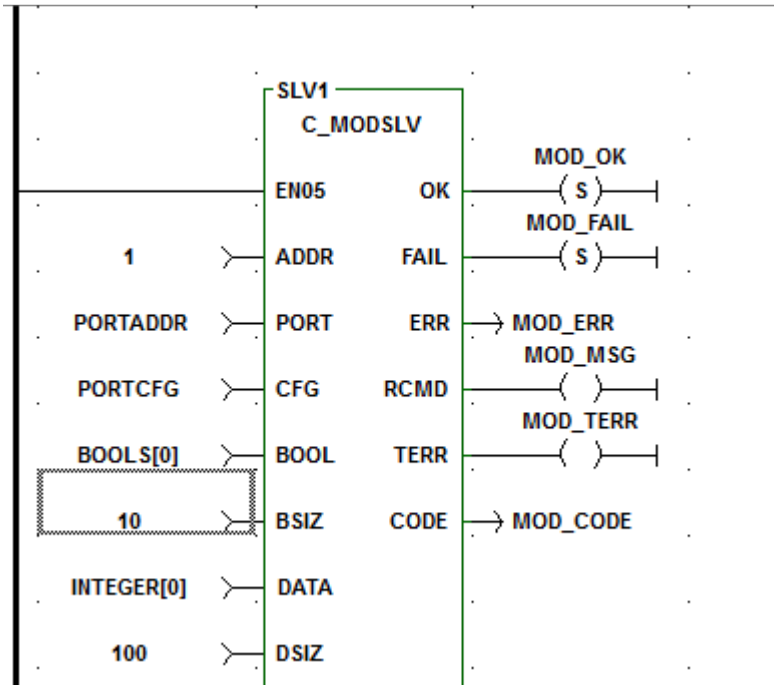
Turning on BOOLS[1] in Modbus Poll:



Note there are 100 INTEGERS in the setup of the sample Modbus Serial ASFB but only 2 BOOLS.

You can expand it by changing the BSIZ from 2 to X within the limit of the size of the BOOLS array (0..998).

For example:



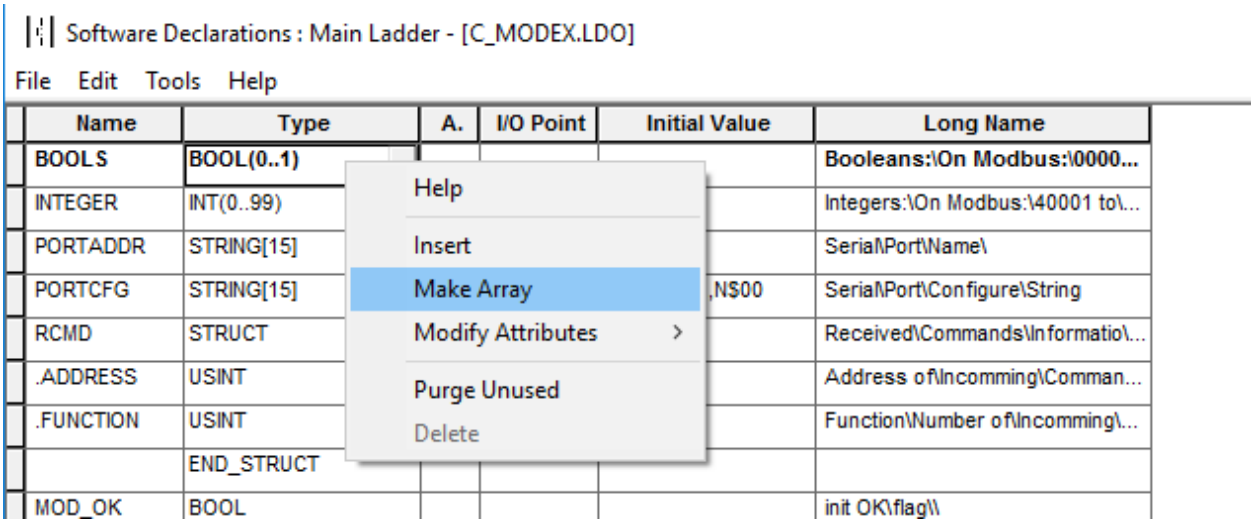
It is important to note that the value of BSIZ must match the size of the array of BOOLS

Software Declarations : Main Ladder - [C_MODEX.LDO]

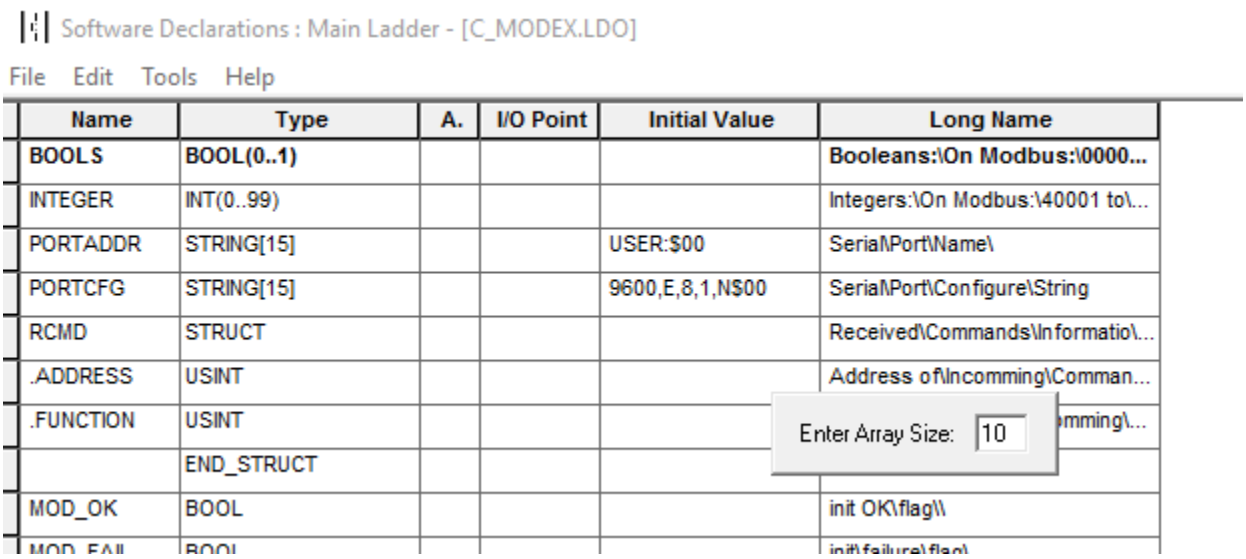
File Edit Tools Help

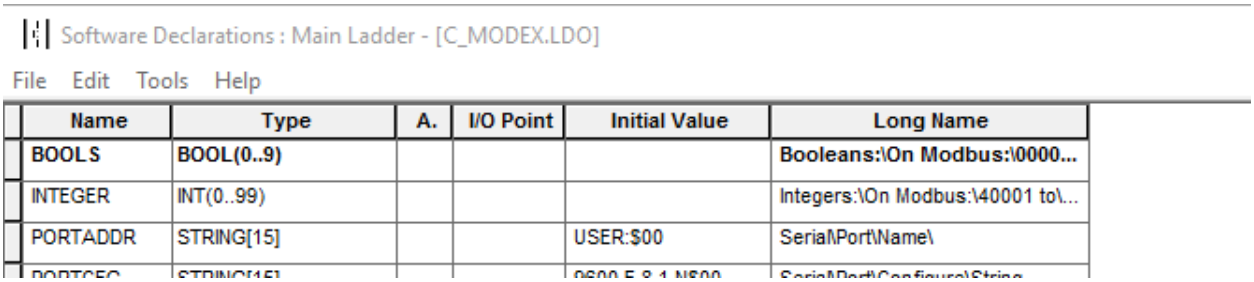
Name	Type	A.	I/O Point	Initial Value	Long Name
BOOLS	BOOL(0..1)				Booleans:\On Modbus:\00001 to...
INTEGER	**INT(0..99)**				**Integers:\On Modbus:\40001...**
PORTADDR	STRING[15]			USER:$00	Serial\Port\Name\
PORTCFG	STRING[15]			9600,E,8,1,N$00	Serial\Port\Configure\String
RCMD	STRUCT				Received\Commands\Informatio\...

To extend the array size, highlight the BOOLS Type and right-click and then choose Make Array.

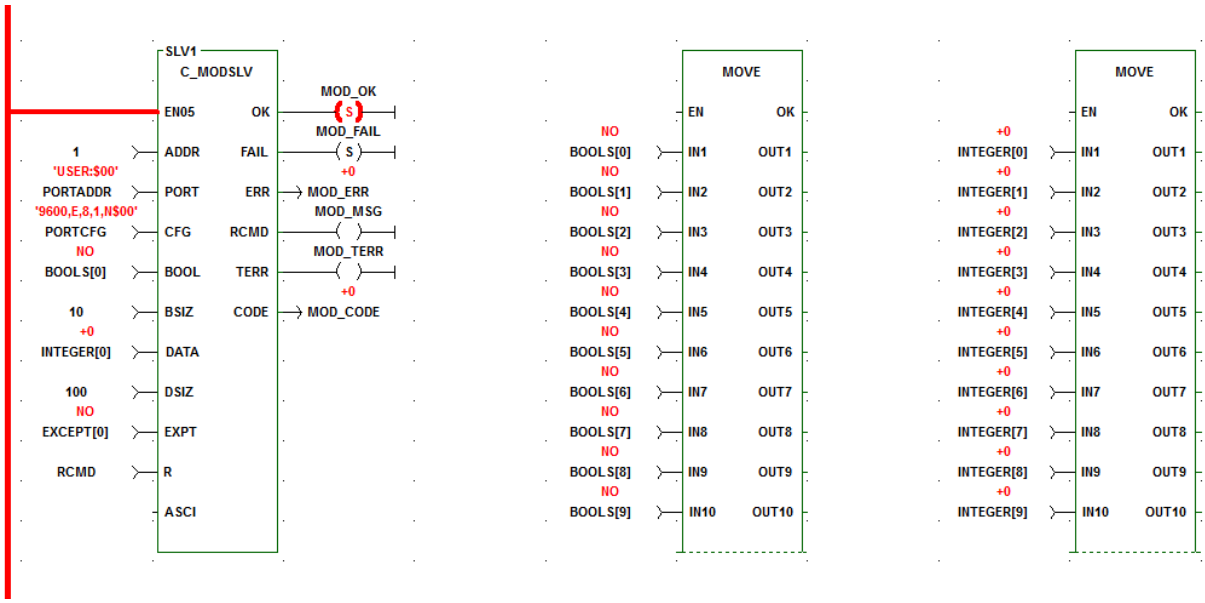


In this example I extended the BOOLS from 0..1 to 0..9 or from 2 to 10 elements. So input 10 into the Enter Array Size box and then the Enter button on your PC's keyboard.



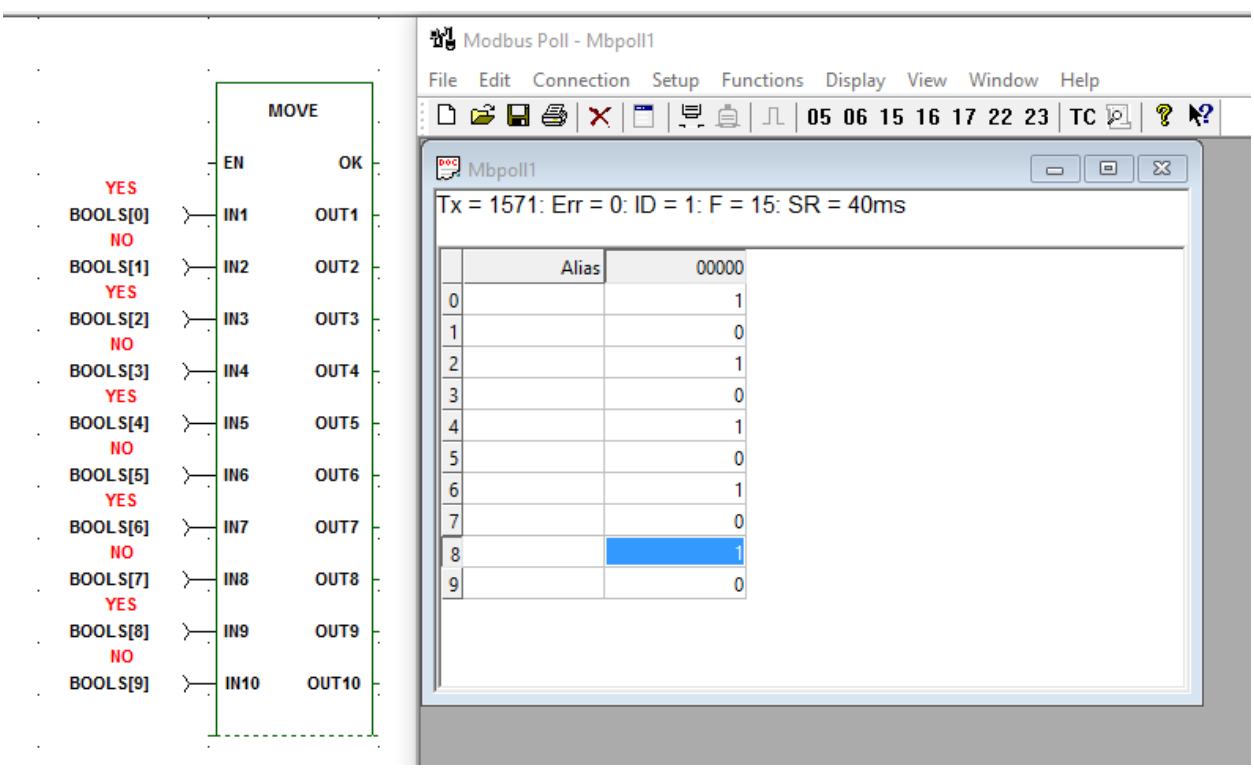


Save and Close the Software Declarations. Another addition will be a departure from monitoring using the View List and add the capability to viewing the values in the ladder itself. Using the MOVE function block as shown:

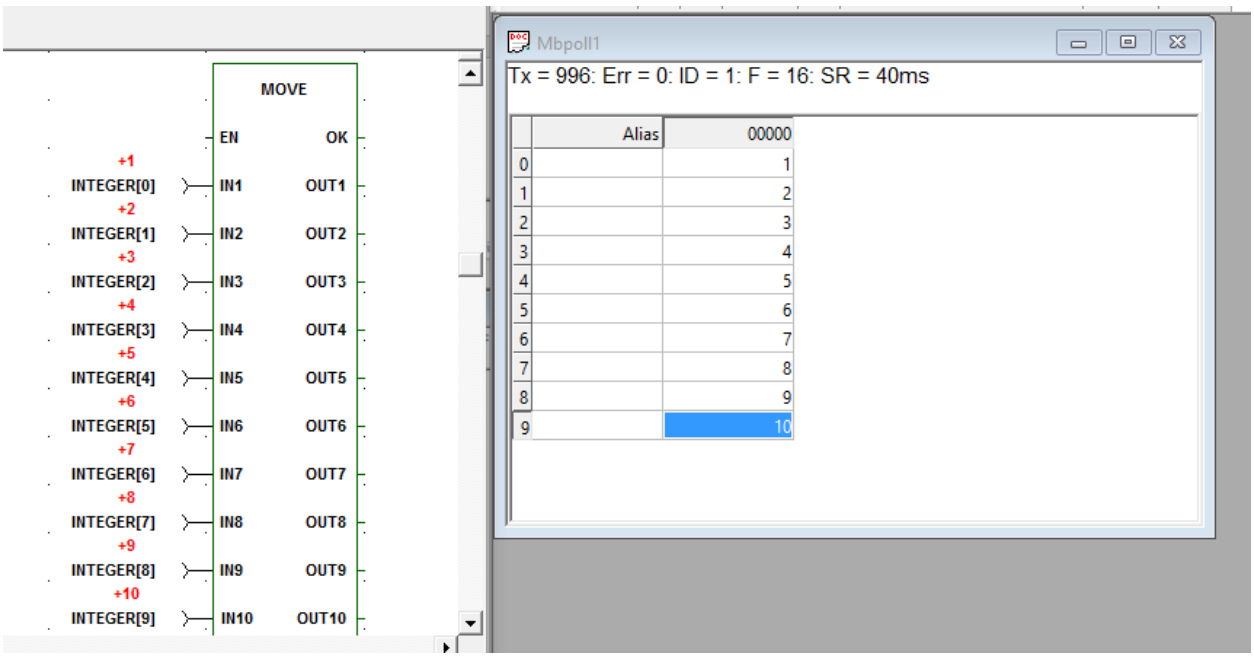


I added the two MOVE blocks, saved the ladder, and the performed a Compile&Download.

Testing BOOLS from Modbus Poll:



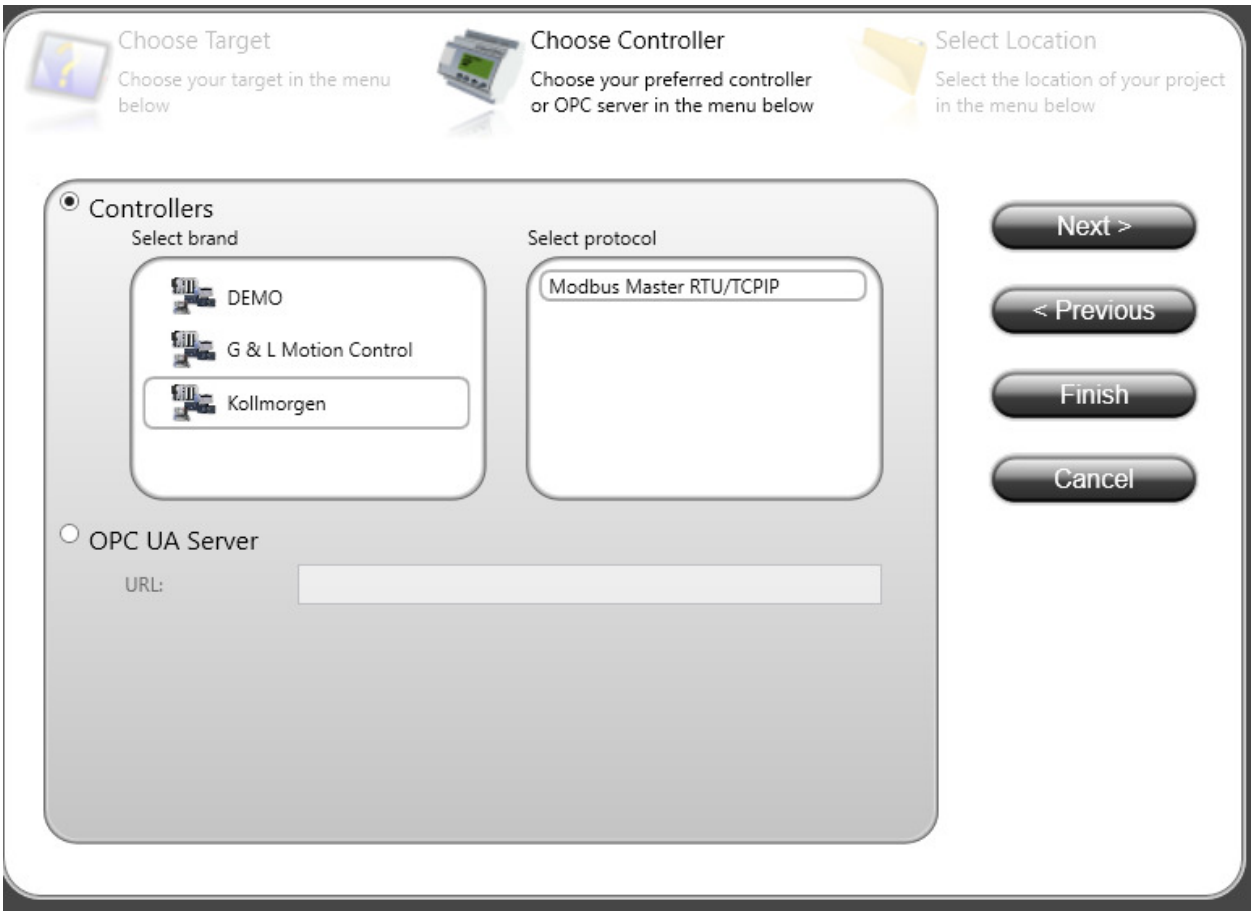
Testing INTEGER using Modbus Poll:



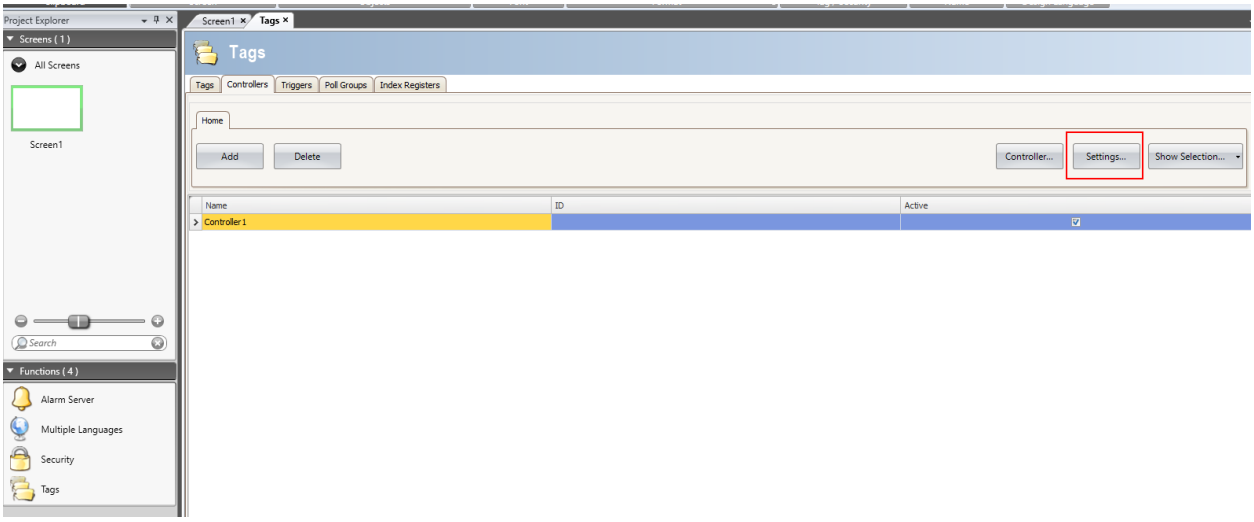
Using KVB with Modbus Serial

Next I will demonstrate using KVB software to do the same thing as Modbus Poll.

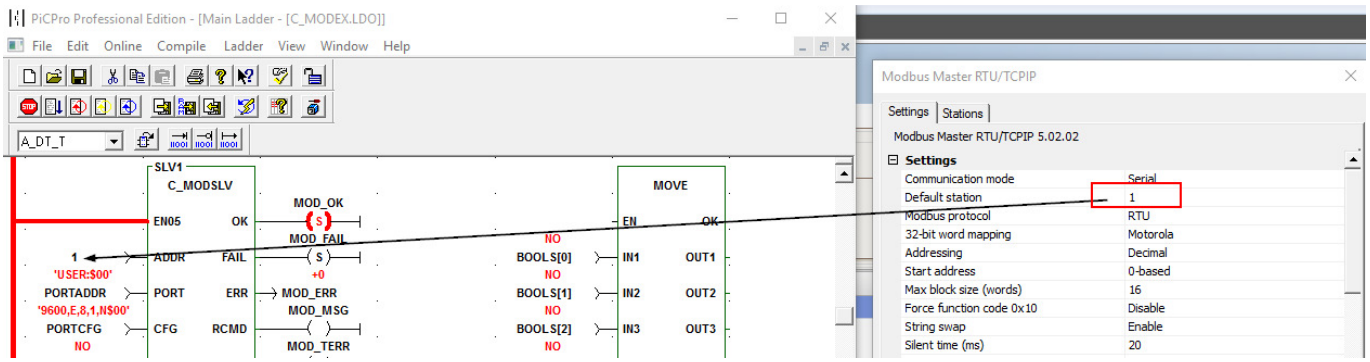
I started by creating a new project and selecting Kollmorgen->Modbus Master RTU/TCPIP as shown below. Note this communication driver supports both Serial (RTU) Modbus and Ethernet based Modbus TCP.



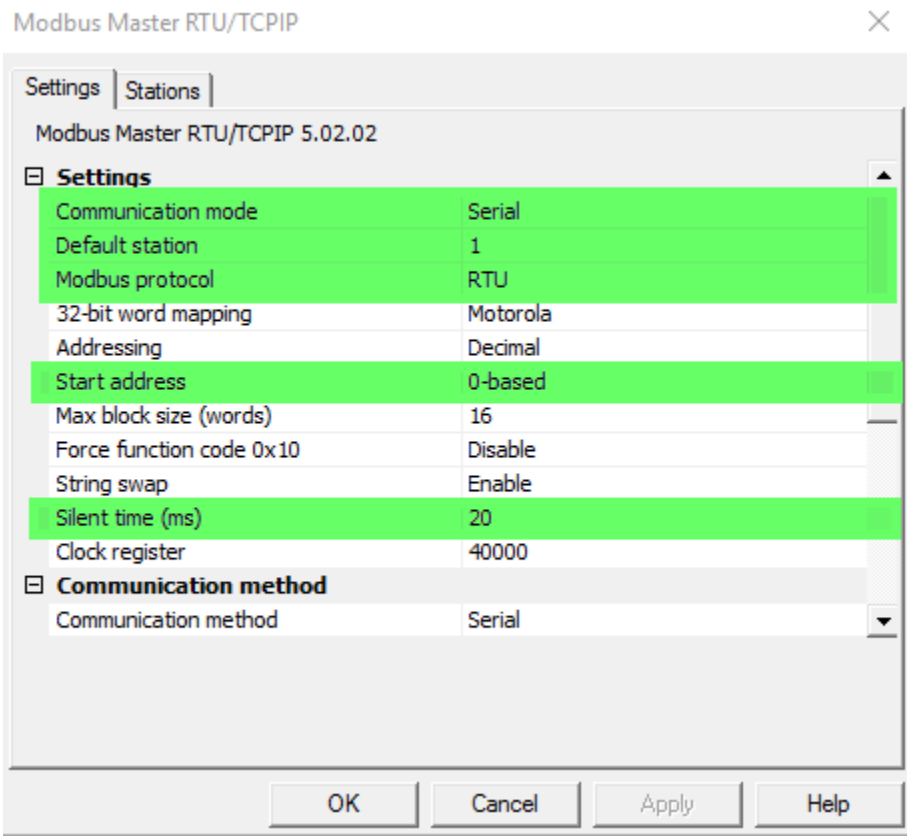
Under Tags->Controllers->Settings:



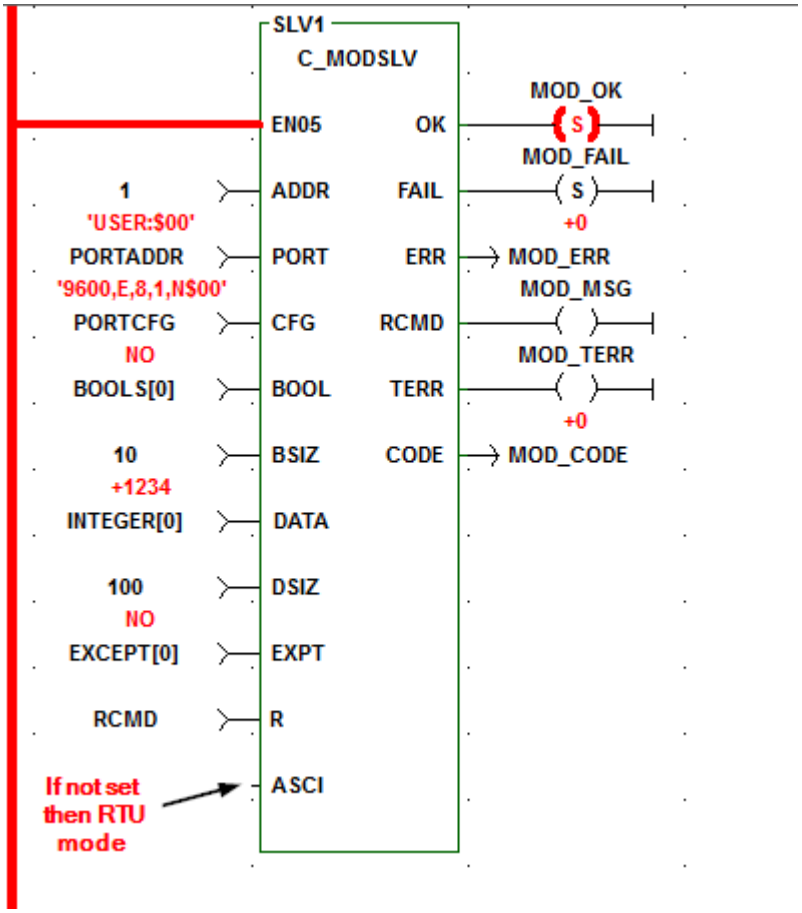
Under the Settings tab I changed the Communications mode to Serial. As with Modbus Poll, the ID or in this case called Default Station needs to be set to the same number as in the C_MODSLV's ADDR input value.



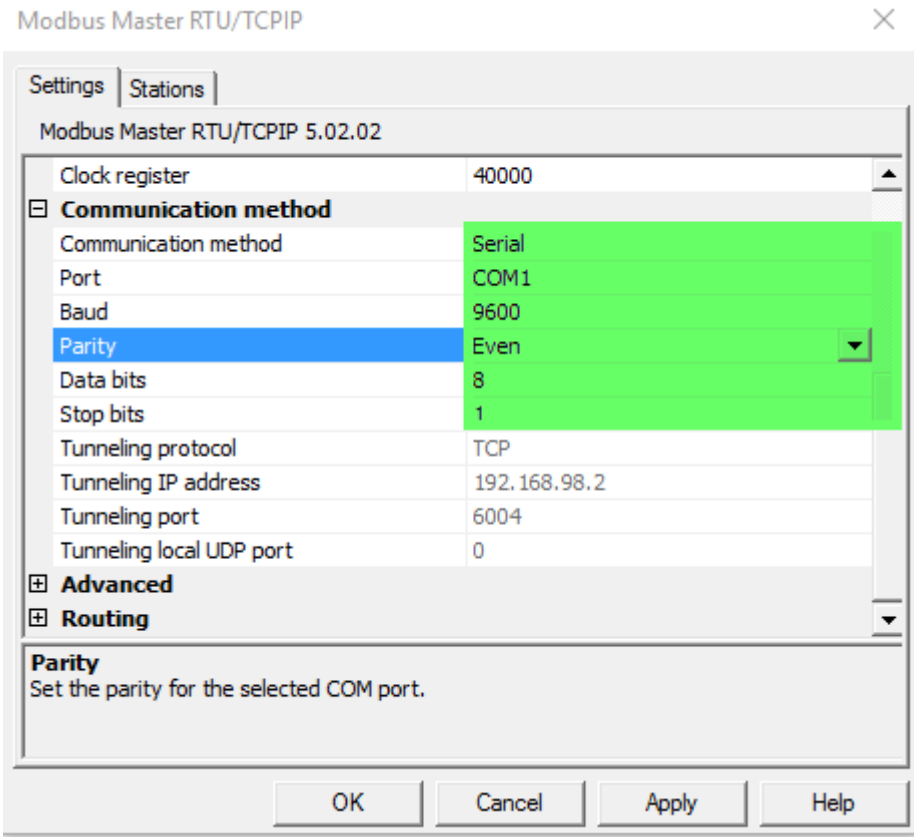
I left the Start address as the default 0-based. If you want the addressing to look like the table in the Modbus Serial ASFB manual you need 1-based but in this example I left it at 0-based because I like the Modbus address count to be the same as the index of the INTEGER[x] array. I also added 20msec of silent time.



As mentioned before, RTU mode is select in the C_MODSLV as follows:

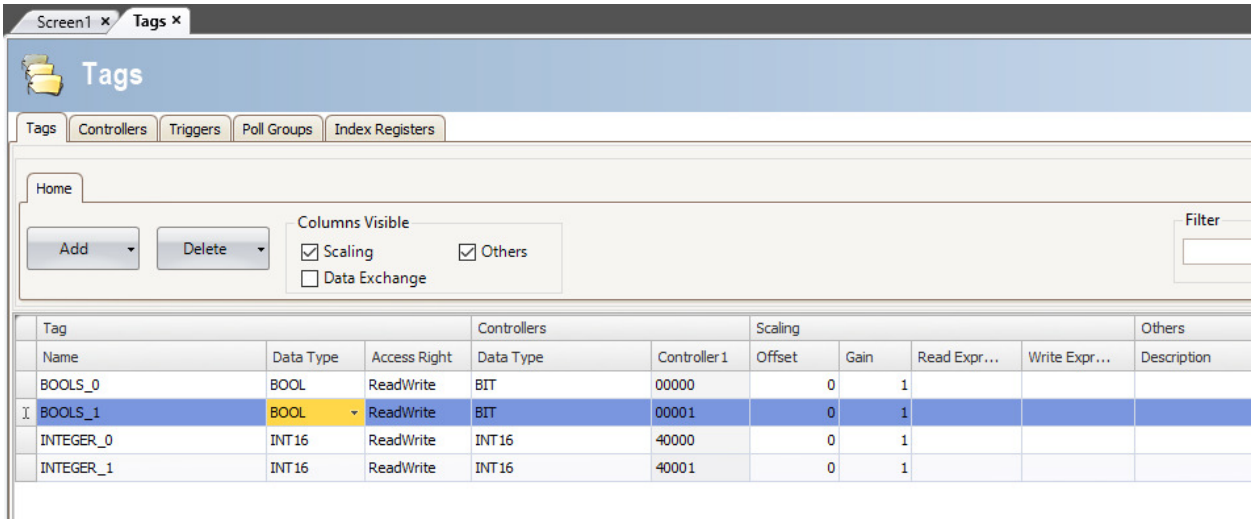


Scrolling down under Communications Method I selected Serial and set the serial port settings to match that of my PC (which I am using as the Modbus Serial Master) This will need to be set according to your hardware that you are using for the Modbus Serial Master.



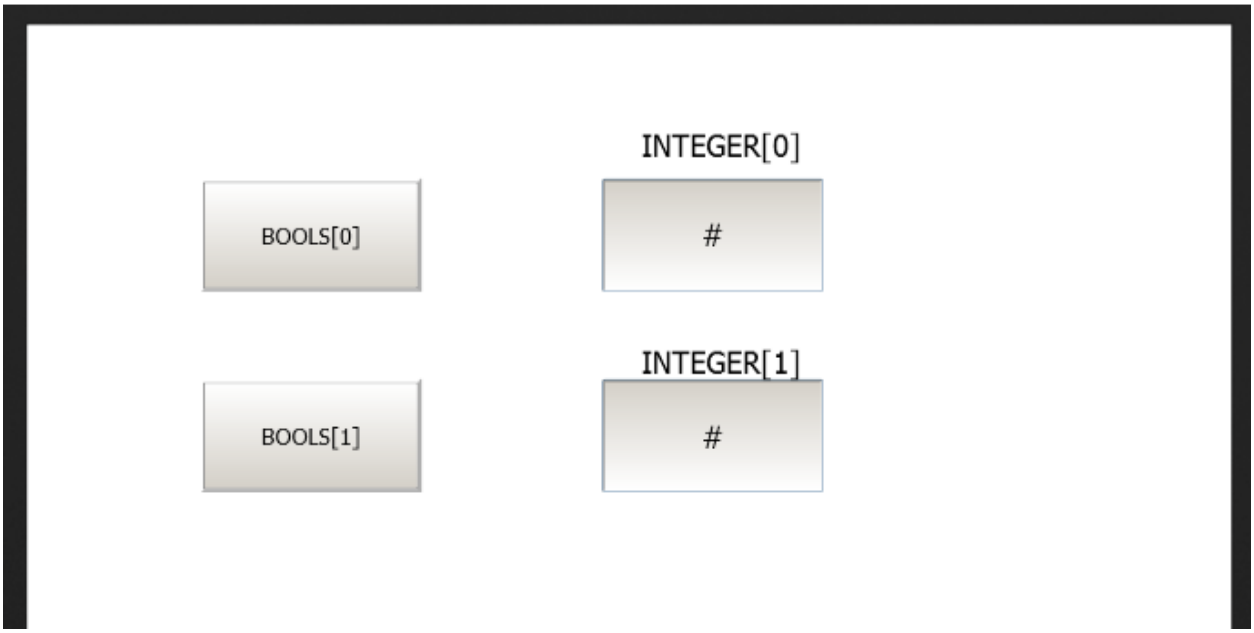
The Stations tab is for Ethernet based communications so I left it at default.

Next I created 4 tags 2 of the BOOL type and 2 of the INT16 type. Note the addressing is 0-based.



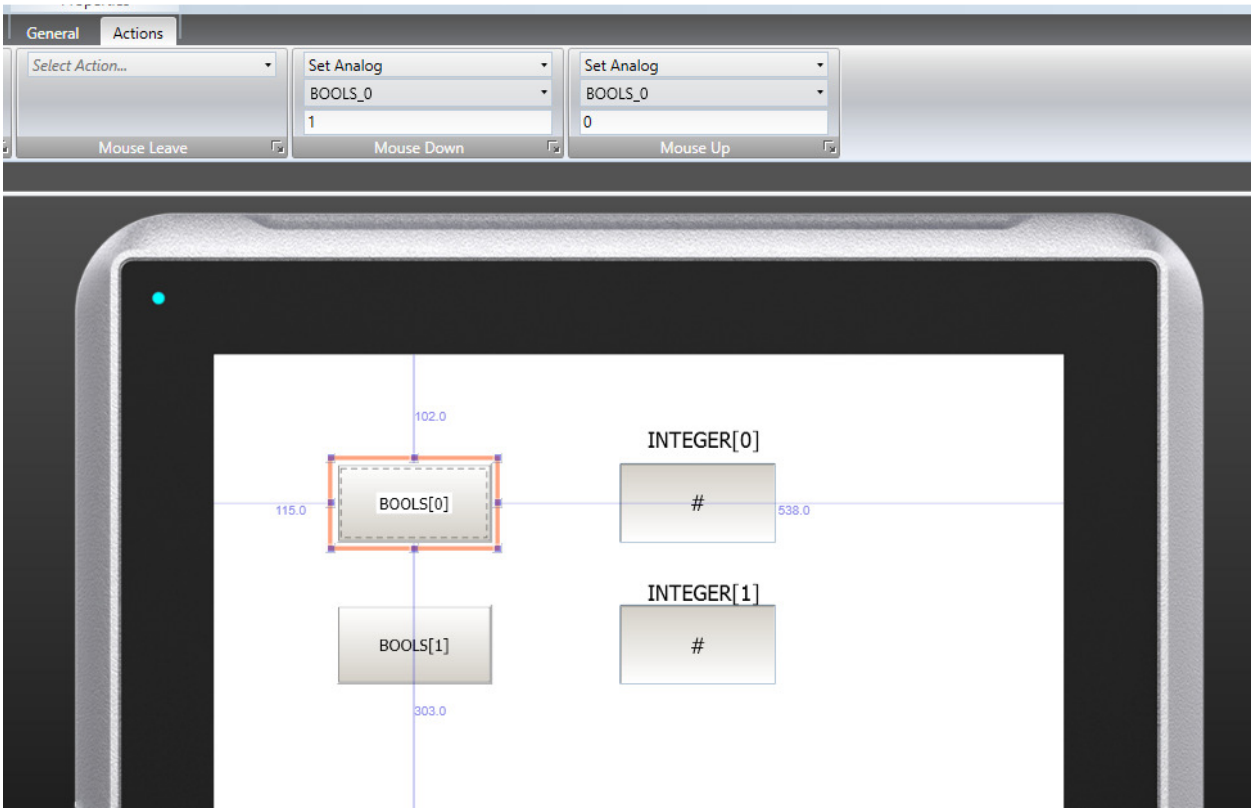
Tag			Controllers		Scaling				Others
Name	Data Type	Access Right	Data Type	Controller1	Offset	Gain	Read Expr...	Write Expr...	Description
BOOLS_0	BOOL	ReadWrite	BIT	00000	0	1			
BOOLS_1	BOOL	ReadWrite	BIT	00001	0	1			
INTEGER_0	INT16	ReadWrite	INT16	40000	0	1			
INTEGER_1	INT16	ReadWrite	INT16	40001	0	1			

Next for Screen 1 I added 2 Buttons and 2 Analog Numerics. I changed the labels on the buttons accordingly and put text labels above the 2 Analog Numerics.

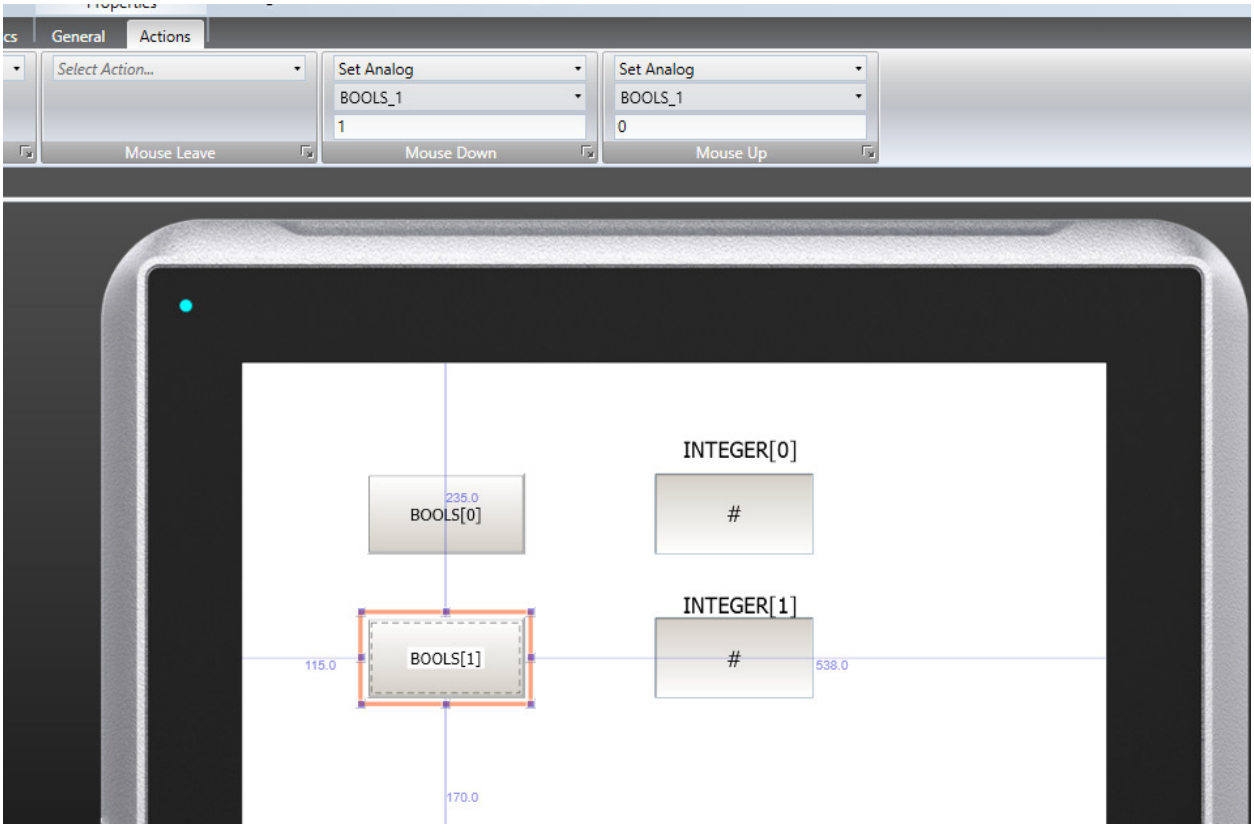


Next it is required to associate a tag and action to each graphic on the screen.

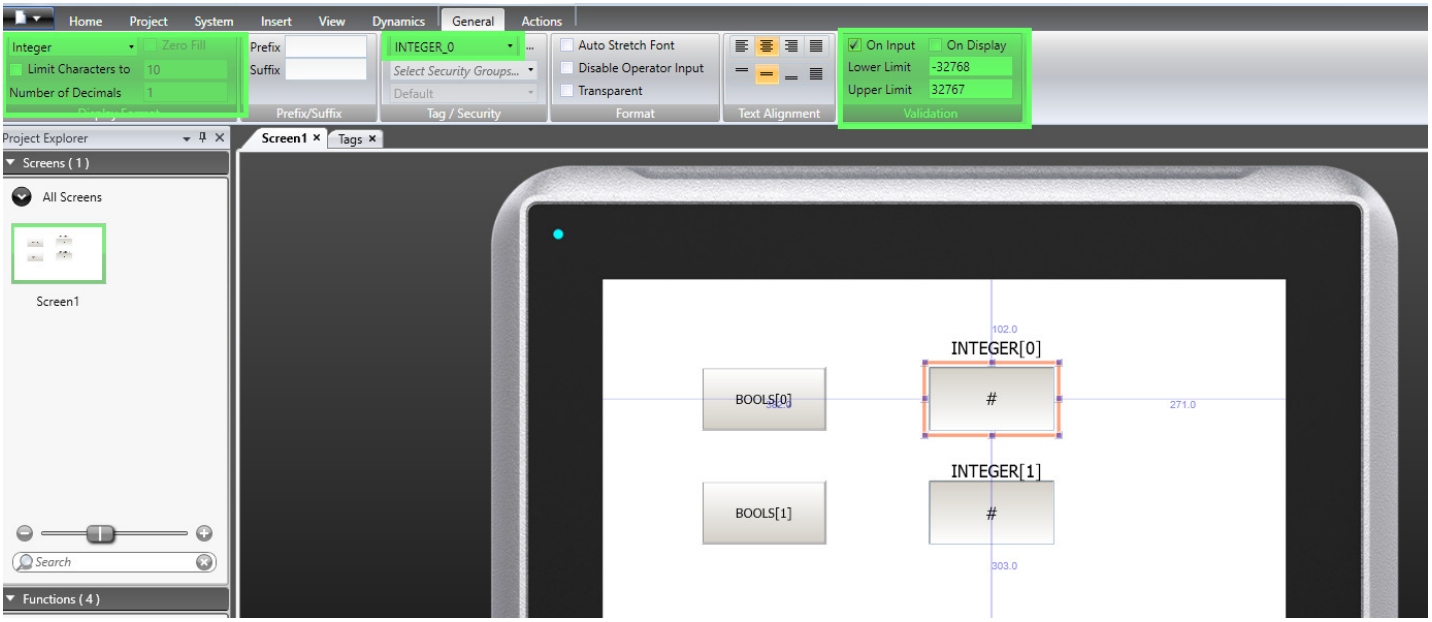
First with the BOOLS[0] button. Set the Mouse Down and Mouse Up as shown so the operation of the button is on press BOOLS[0] will be 1 or ON and on release BOOLS[0] will be 0 or OFF.



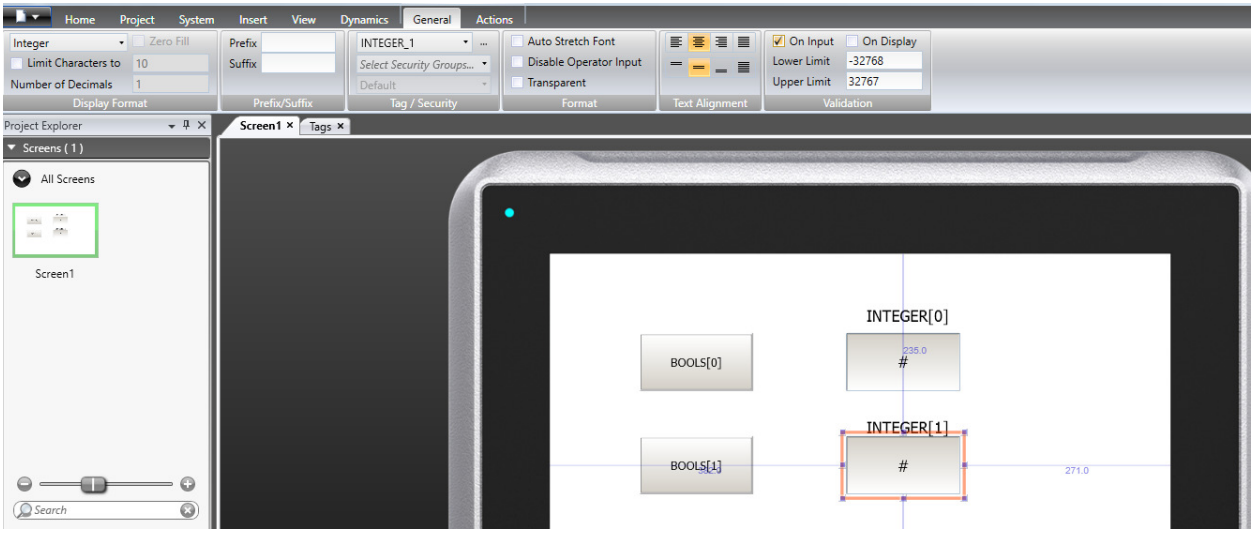
Repeat for the BOOLS_1 button:



Next the Analog Numeric fields will be configured. The tag is selected, Integer type, and On Input is setup to limit the input from the HMI to be within a valid range for a 16 bit signed integer.

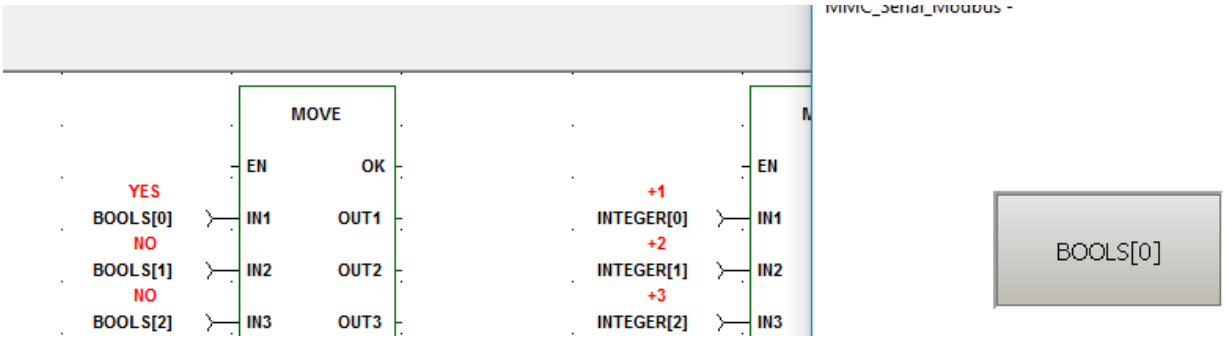


Repeat for INTEGER[1]:

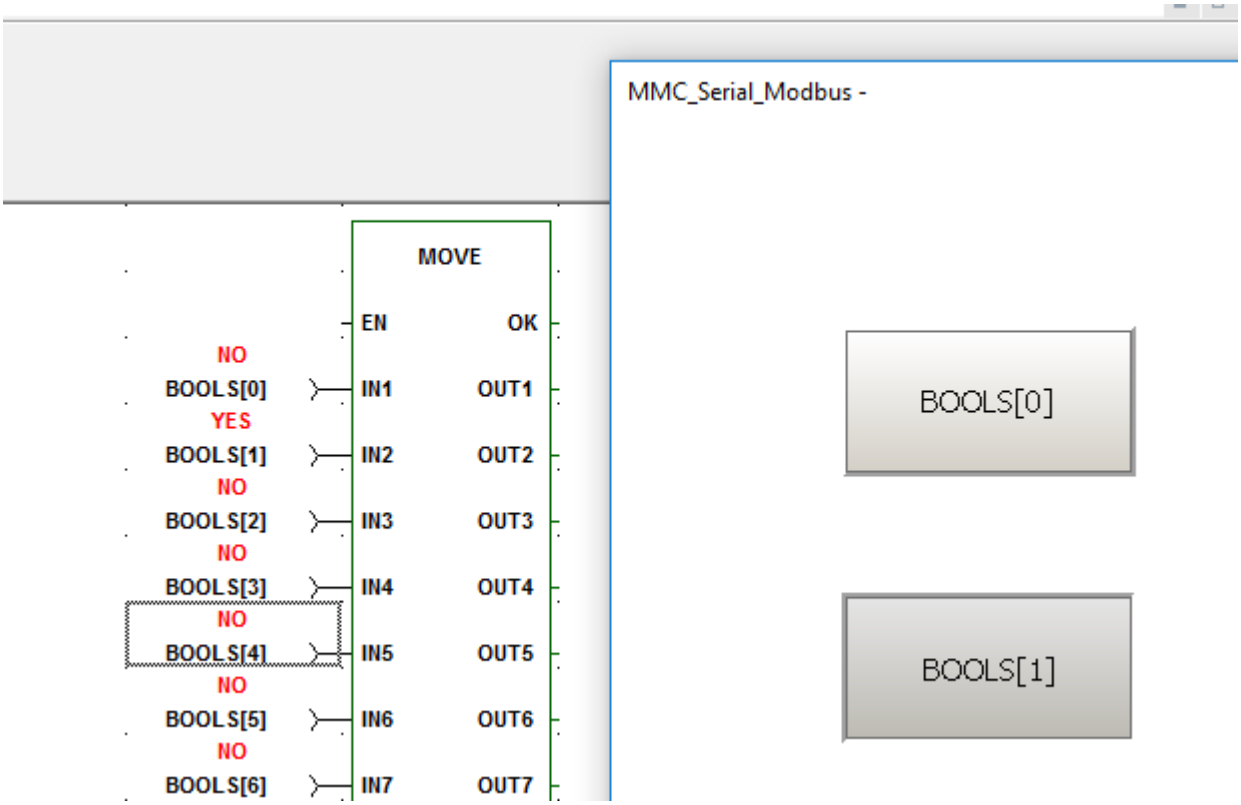


After Save, Build, Run….

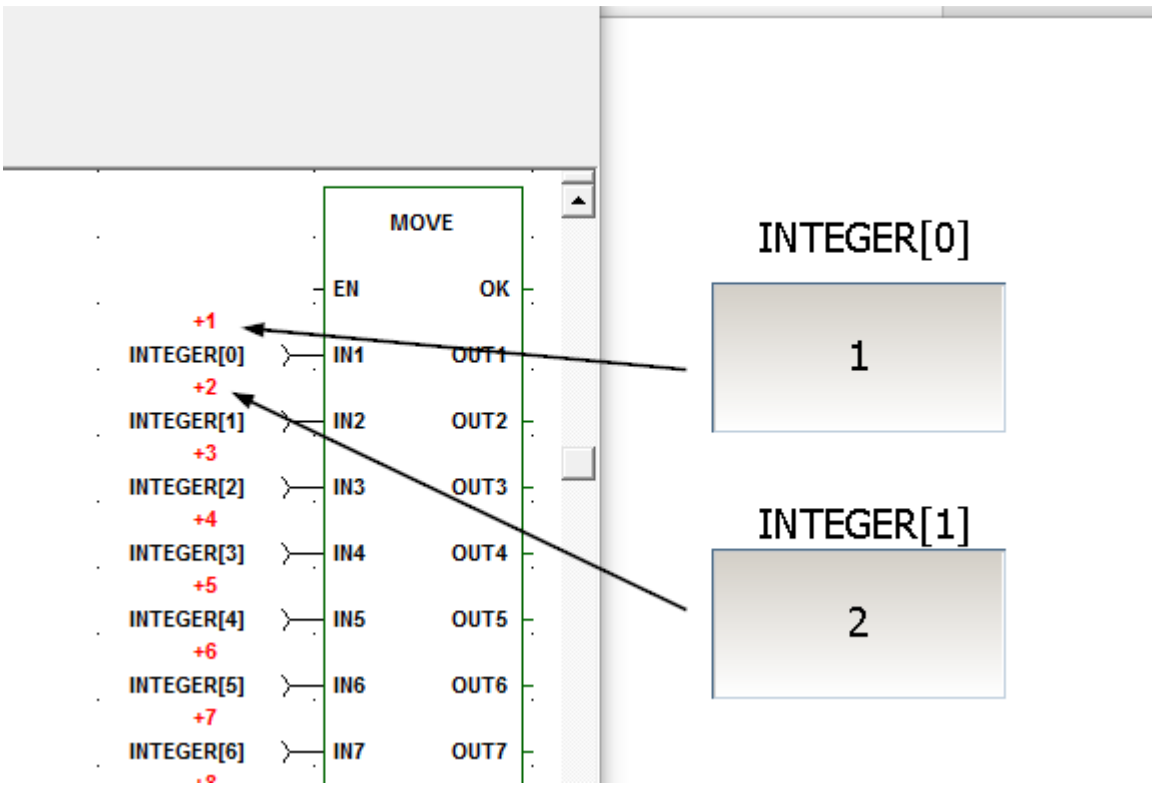
On press of the BOOLS[0] button in run-time:



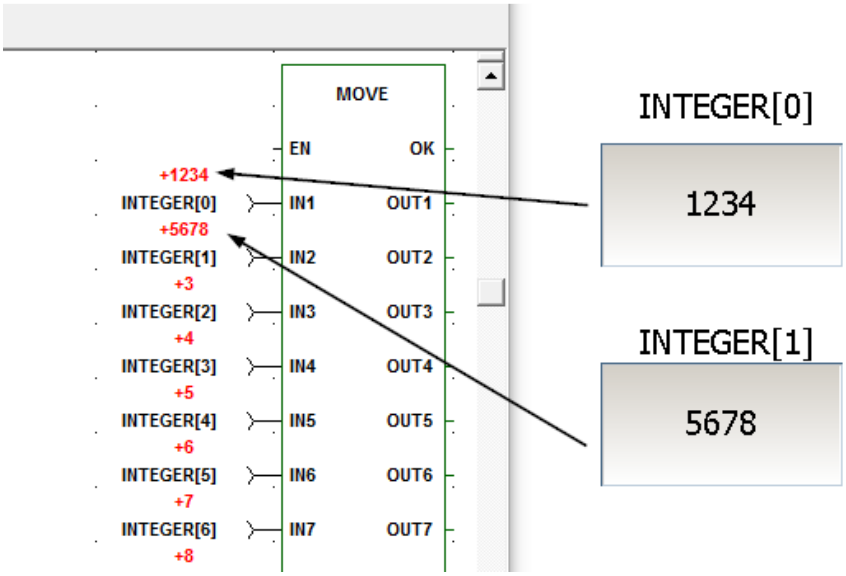
On press of the BOOL[1] button in run-time:



From the Modbus Poll the values were set in the INTEGER array. Initially KVB in run-time shows INTEGER[0] as a value of 1 and INTEGER[1] as a value of 2.



Clicking on the 2 data fields and changing the values:



This concludes the Modbus Serial Example.