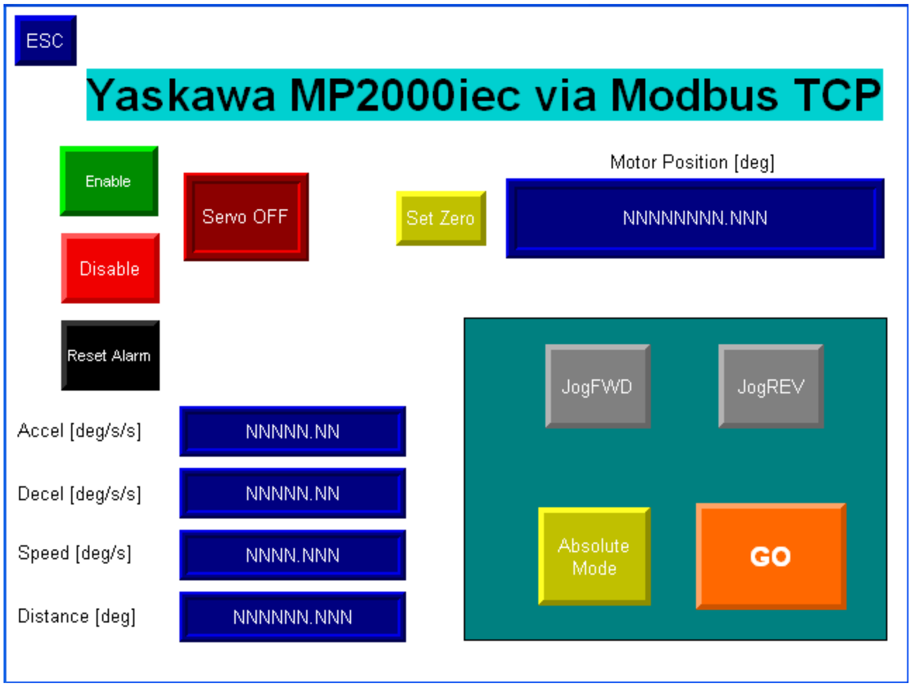
Application Note

MPiec Communication to PanelView Plus HMI via Modbus/TCP

Applicable Product: MPiec, MotionWorks IEC



Yaskawa Electric America
2121 Norman Drive South
Waukegan, IL 60085
1-800-927-5292

Subject: Application Note	Product: MPiec, MotionWorks IEC	Doc#: AN.MP2000iec.02
Title: MPiec Communication to PanelView Plus HMI via Modbus/TCP		

Application Overview

An essential element of machine control system design is the interaction between the machine controller and the operator interface device. One of the more popular lines of Human-Machine Interfaces (HMIs), is the PanelView Plus Series from Allen Bradley. The PanelView Plus and the associated RSView Studio programming Software is a familiar HMI environment for many machine builders, and enjoys a wide range of acceptance. Seamlessly and easily interfacing the Yaskawa MPiec Series machine controllers to the PanelView Plus HMIs over an Ethernet connection allows customers to leverage the enhanced motion capabilities of the controller while preserving engineering costs already sunk into HMI screen development.

This Application Note summarizes the key steps to create the communication interface between these two products using the KEPServer Enterprise OPC Server included with RSView Studio. The note takes the reader through an example application that illustrates how to set up the MPiec series machine controller as a Modbus Slave, create control program variables in the Modbus address space, create a device connection and OPC tag database in KEPserver Enterprise, map Modbus registers to controller variables, connect graphical HMI elements to the OPC tags, and download and run the HMI program. There is a corresponding eLearningVideo in the Training section of the Yaskawa Website that contains example program files available for download.

Products Used:

Component	Product and Model Number
Controller	MP2300Siec (standard 2-axis demo kit). (This application note is applicable to all MPiec series machine controllers)
Servopack	SGDV-R90F11A
Motor	SGMAV-01A3A61
HMI	Allen Bradley PanelView Plus 700 Cat. 2711P-T7C4D1
Software	Yaskawa MotionWorks IEC Express v1.1.3.9 (PRO version program files also available) Rockwell RSView Studio - Machine Edition, v4.00.00 (CPR 7), Patch/Pack version 00, Build 60 KEPServer Enterprise v4.170.328-U
Example Program Files (Found under the eLV link)	MP2300SiecToPVPlus700.zwe (.zwt for PRO version) PVPlusToMP2300Siec_SimpleApp.apa MP2300SiecSimpleApp.pfe
eLearningVideo	eLV.MP2000iec.01.IECtoPVPlusHMI

Setting up the Network

The test setup used consists of a simple network hub at the center of three Ethernet devices. Each device has a unique IP address on the same subnet. In the example provided, the addresses are set as shown in Figure 1.

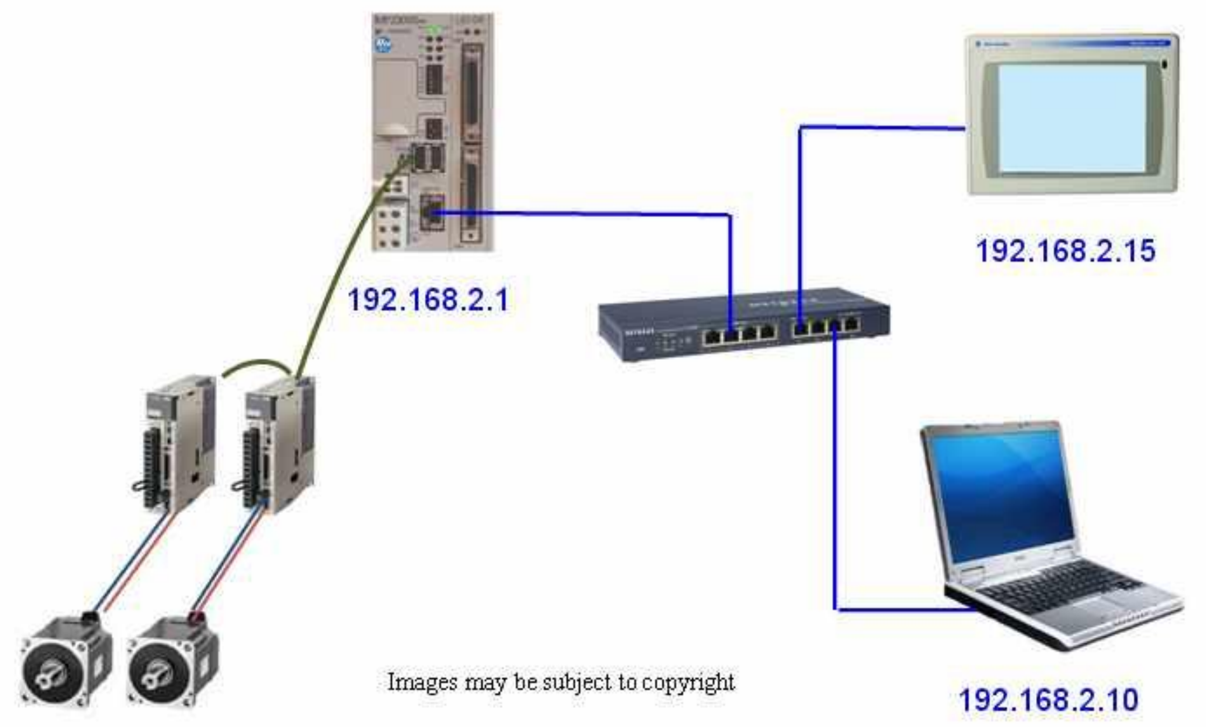


Figure 1: Network Setup

Retrieving the Example Programs

All example program files can be found at eLV.MP2000iec.01.IECtoPVPlusHMI, in a zipped file.

MotionWorks IEC program.

Both an Express and a Professional version of the program have been provided in a zipped format. Launch MotionWorks IEC and open the appropriate program. This application note describes the Express version.

Choose File_Open and navigate to the folder where the zipped file 'MP2300SiecToPVPlus700.zwe' is stored.

Select and open the file. Once the file is open, Choose Make to compile the project.

KEPServer Enterprise

Launch KEPServer Enterprise KEPServer Enterprise V4.0 and open the file 'MP2300SiecSimpleApp.pfe'.

RSView Studio – Machine Edition

Opening this file is a multi-step process.

Restore the archive file by launching the Application Manager from the Tools Menu.

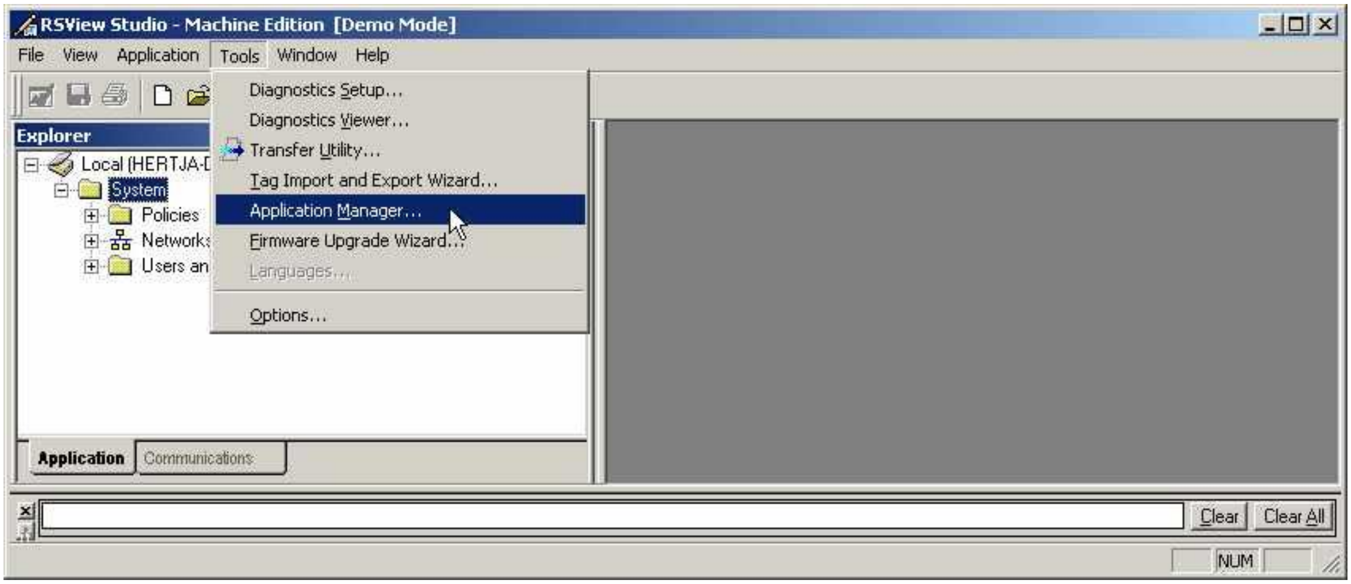
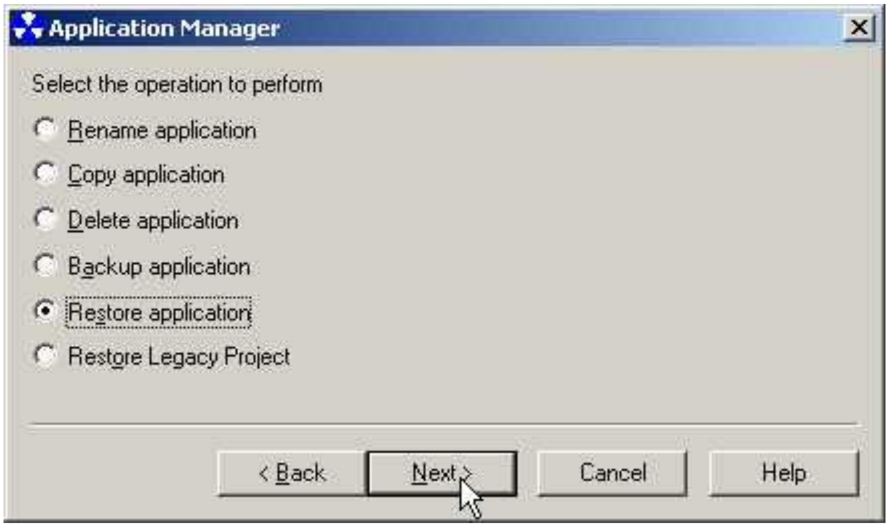


Figure 2: Launching Application Manager

Choose Machine Edition, then the action to restore a project as shown in Figure 3.

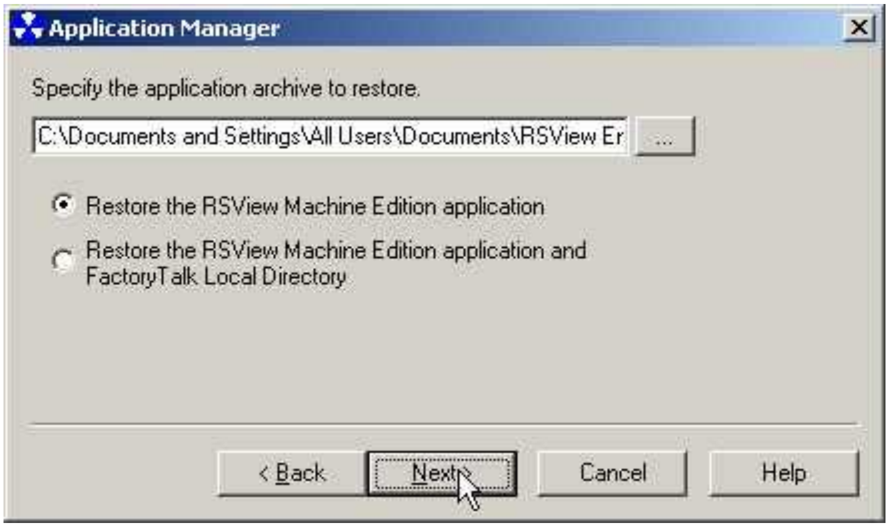


Select the operation to perform

○ Rename application
○ Copy application
○ Delete application
○ Backup application
● Restore application
○ Restore Legacy Project

Figure 3: Restore RSView Application

Browse to find the archive file ‘PVPlusToMP2300Siec_SimpleApp.apa’ and click Next.



Application Manager

Specify the application archive to restore.

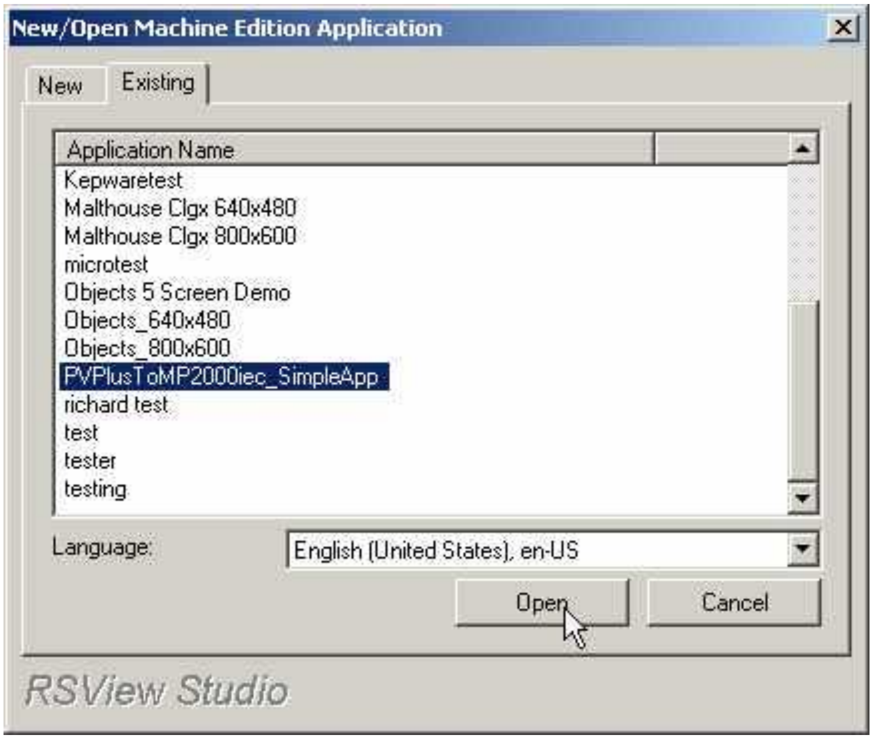
C:\Documents and Settings\All Users\Documents\RSView Er

● Restore the RSView Machine Edition application

○ Restore the RSView Machine Edition application and FactoryTalk Local Directory

Figure 4: Browse to find the archive file

After the project has been restored, it will appear in the project listing. Open it by choosing File_Open Application… and selecting the desired project from the list of Existing projects as shown in Figure 5.



New/Open Machine Edition Application

Application Name
Kepwaretest
Malthouse Clgx 640x480
Malthouse Clgx 800x600
microtest
Objects 5 Screen Demo
Objects_640x480
Objects_800x600
PVPlusToMP2000iec_SimpleApp
richard test
test
tester
testing

Language: English (United States), en-US

Figure 5: Opening an Existing Project

Description of Example Programs

HMI Screen Layout and Functionality

Figure 6 shows the final HMI screen that allows the operator to enable and disable Servo Axis 1, establish an absolute zero position, set basic move parameters, jog the motor, and make single incremental or absolute moves.

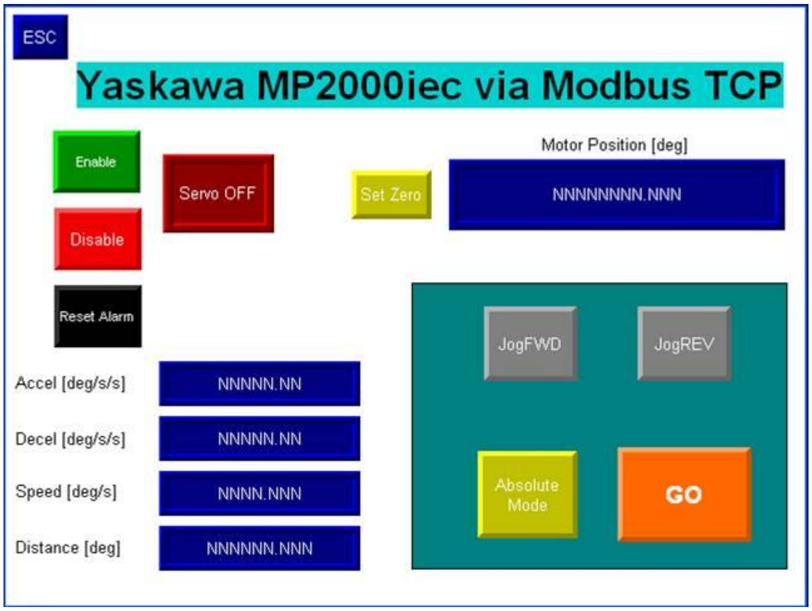


Figure 6: HMI Screen

- Momentary pushbuttons include: Enable, Disable, Reset Alarm, Set Zero, Jog Fwd, Jog Rev, and Go.
- The element labeled Servo OFF is an indicator that shows servopack status via a change in appearance.
- A numeric display shows current motor absolute position in degrees.
- The Mode button is a toggled element that switches between absolute and incremental move mode
- Data entry elements are provided to change Acceleration, Deceleration, Speed and Distance (Position) of the moves

MP2300Siec Function Block Diagram in Main POU

The example program in the MP2300Siec has been built using a simple template. Although the sample code was built on an MP2300Siec controller template, this application note and example code can be used for any MPiec series machine controller. As the MotionWorks IEC Express project tree in Figure 7 shows, the program includes the Firmware Library PLCopenPlus-v_2_2 and the User Libraries PLCopen Toolbox v018 and Yaskawa Toolbox v007. The project tree contains 3 simple POUs that initialize and run the control program. The main control code is found, naturally, in the “Main” POU.

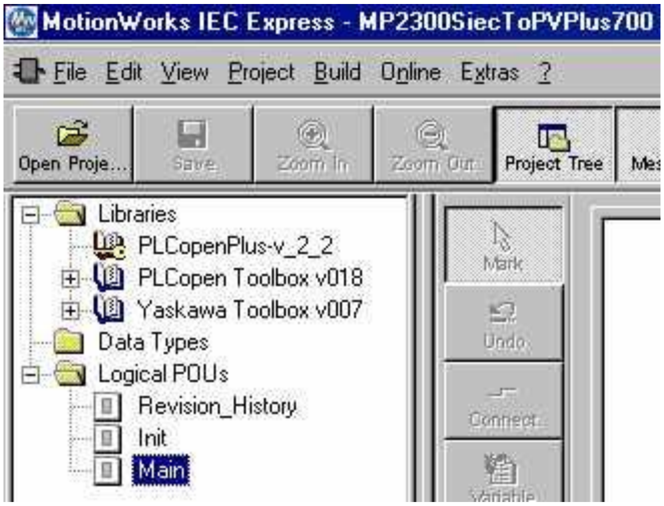


Figure 7: Project Tree with POUs and Included Libraries

Figure 8 shows the section of code for enabling the servo and setting an absolute zero position. A rising edge of the HMI Enable pushbutton will set an internal Enable bit fed into the AxisControl function block from the PLCopen Toolbox. AxisControl returns the servopack status which is fed to the HMI status output.

MC_SetPosition is a standard PLCopen function block. In this example, the desired position is hard-coded to 0.0. since the motor features an absolute encoder, this zero position will be retained after loss of power, but can be re-established at any time.

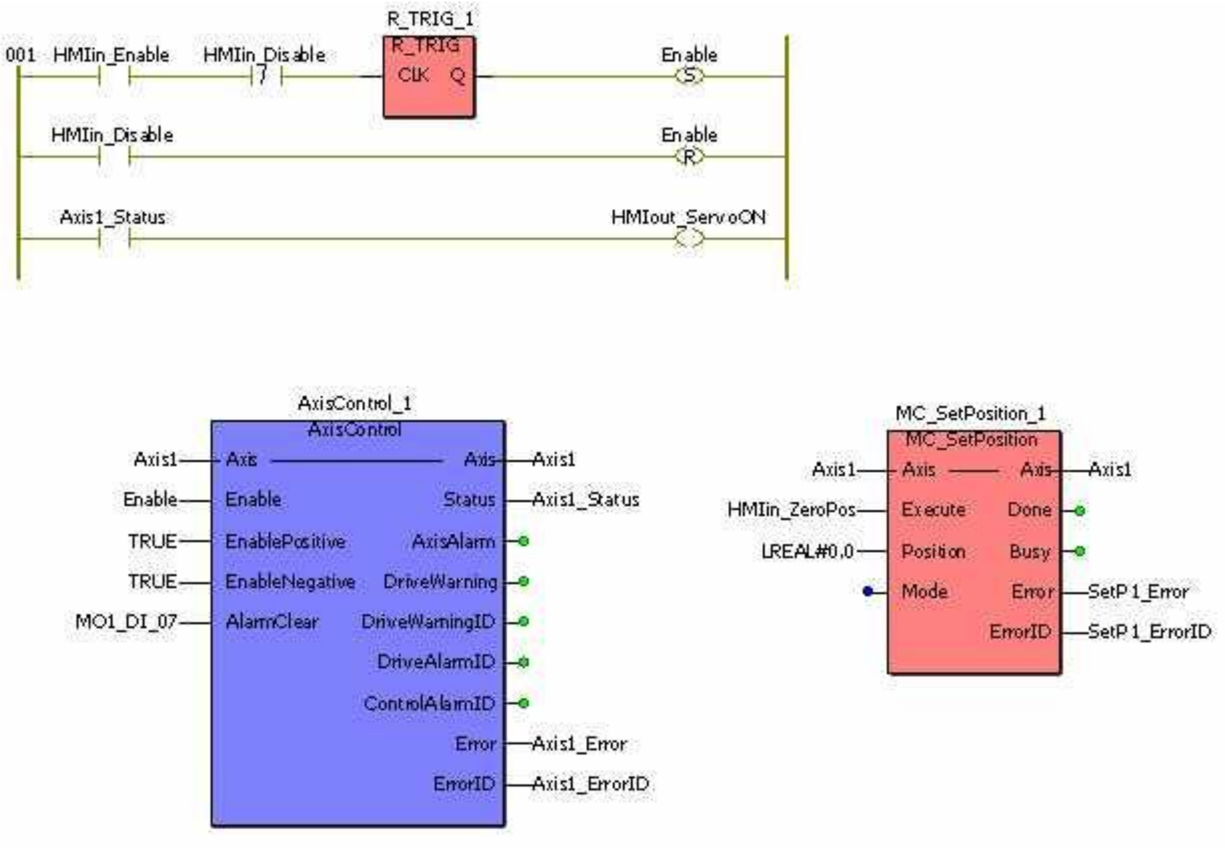


Figure 8: Enable Servo and Set Absolute Zero Position

The next section of code, shown in Figure 9, uses a Toolbox function block to retrieve axis information to be stored into a structure. The full structure layout is defined in the file “PLCopen Toolbox v018 → Data Types → MotionBlock Types”. A single element of the structure is passed into the HMI motor position output variable.

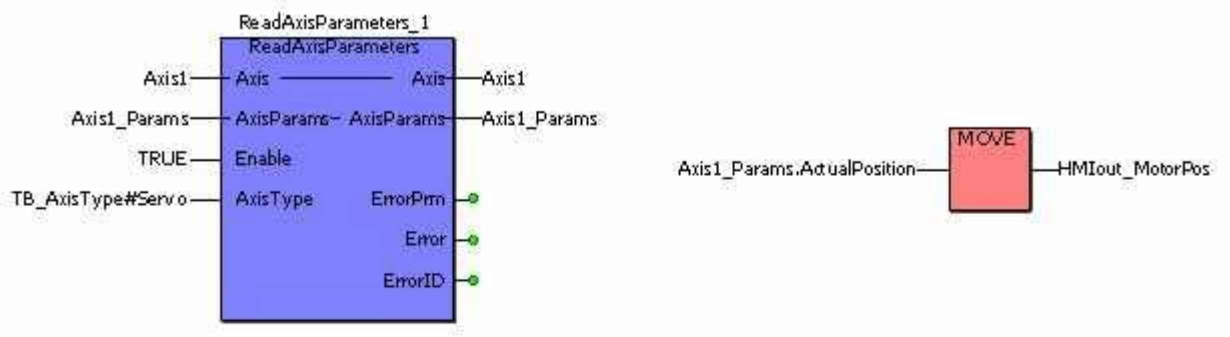


Figure 9: Read Axis Information and Transfer Motor Position to HMI

Jogging the motor is easily accomplished with the Jog function block from the Toolbox as shown in Figure 10.

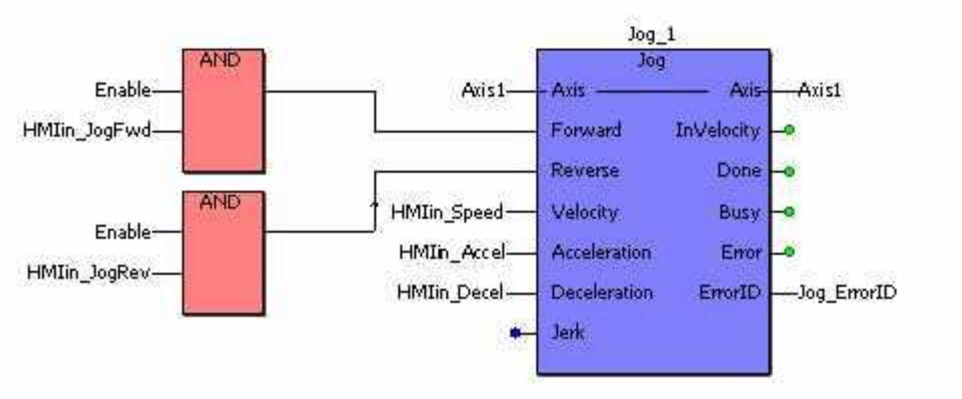


Figure 10: Jogging the motor

Incremental and Absolute moves are accomplished with the appropriate PLCopen motion blocks as shown in Figure 11. A handy feature of the AND block is that the number of inputs to the block may be increased and they may be inverted by a simple checkbox in the AND dialog box. For the MC_MoveRelative block, an interlock has been added to the edge-sensitive Execute input to make sure the previous move finishes before another move can begin.

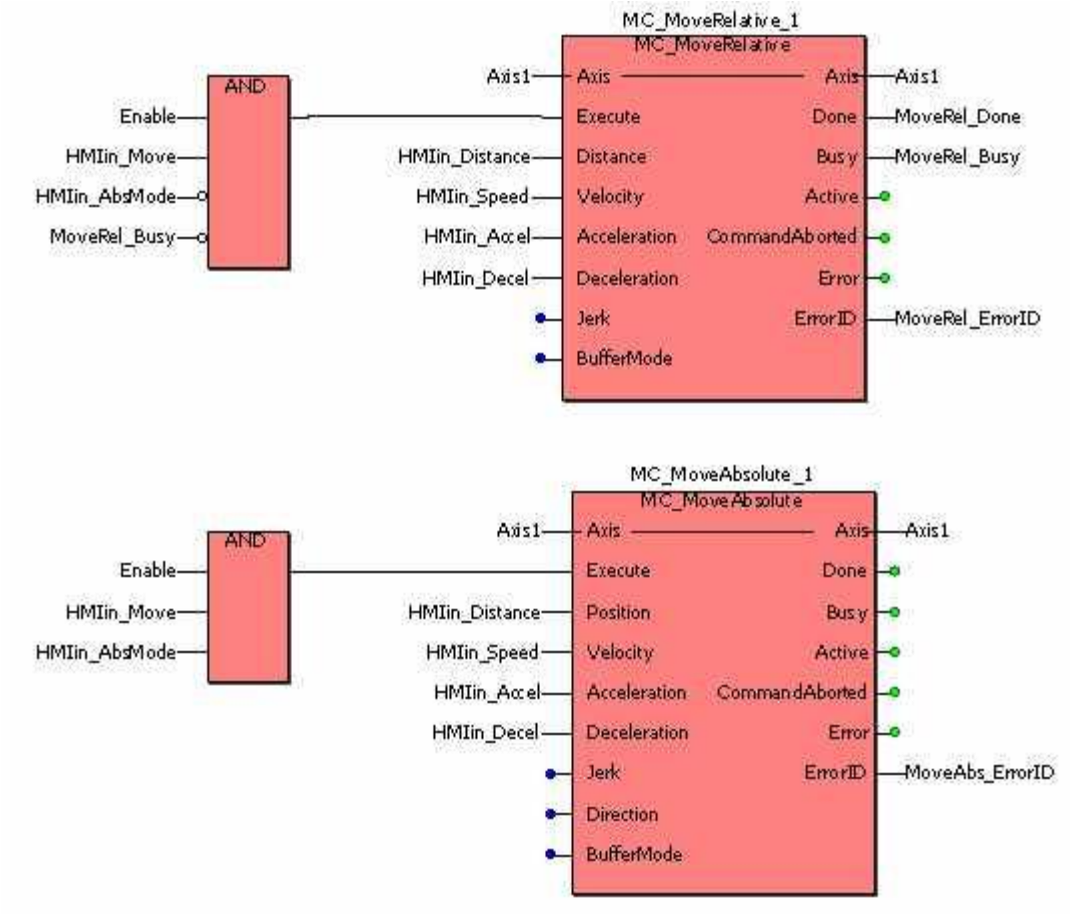


Figure 11: Incremental and Absolute Moves

Lastly, Figure 12 shows how controller alarms can be cleared using either the HMI button, OR the switch labeled DI7 on the demo kit.

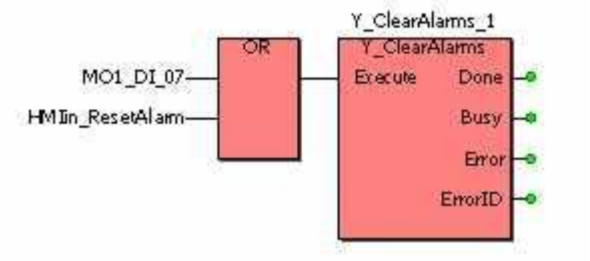


Figure 12: Clearing Alarms

4 Steps to Establish Information Transfer via Modbus/TCP

Step 1) Configure the MPiec series machine controller as a Modbus/TCP Slave (Server)

With the program file open in MotionWorks IEC, Click the button in the menu bar to launch the Hardware Configurator.

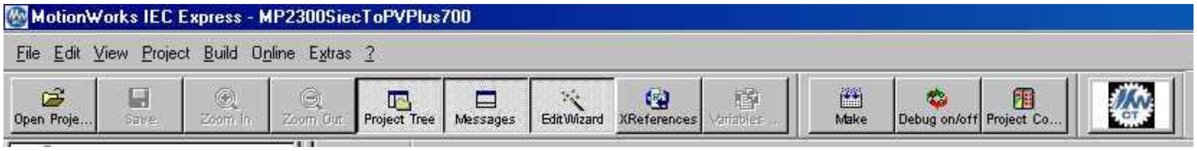


Figure 13: Menu Bar in MotionWorks IEC Express

The following procedure will work for both a controller with an existing configuration and for a completely blank controller where the entire project archive has been deleted. An alternative method may be to go Online immediately, then choose the *Offline* Configuration when prompted.

When the Configurator opens, go under the Online Menu and Choose Controller Configuration Utilities…

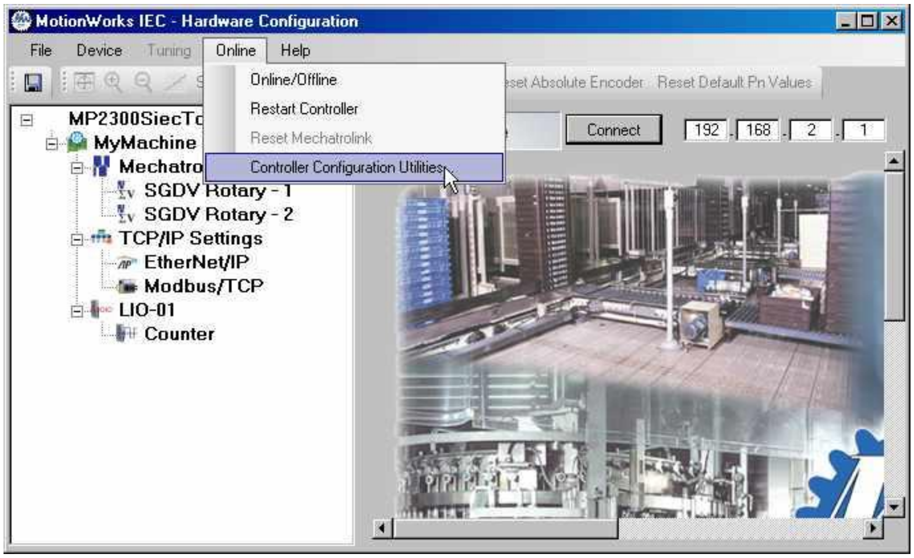
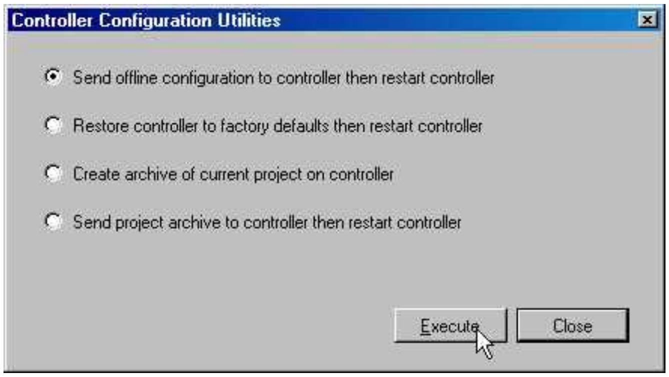


Figure 14: Hardware Configurator – Config Utilities

Select and Execute “Send offline configuration to controller then restart controller”. This will also send all servopack parameters to the servopacks. A power cycle is best since the servopacks may also need to be reset.


Controller Configuration Utilities

● Send offline configuration to controller then restart controller

○ Restore controller to factory defaults then restart controller

○ Create archive of current project on controller

○ Send project archive to controller then restart controller

Figure 15: Send Offline Configuration and Parameters

Click ‘Connect’ to go online with the controller, then click on the Modbus/TCP listing in the project tree to enter the configuration dialog box for Modbus/TCP communications.

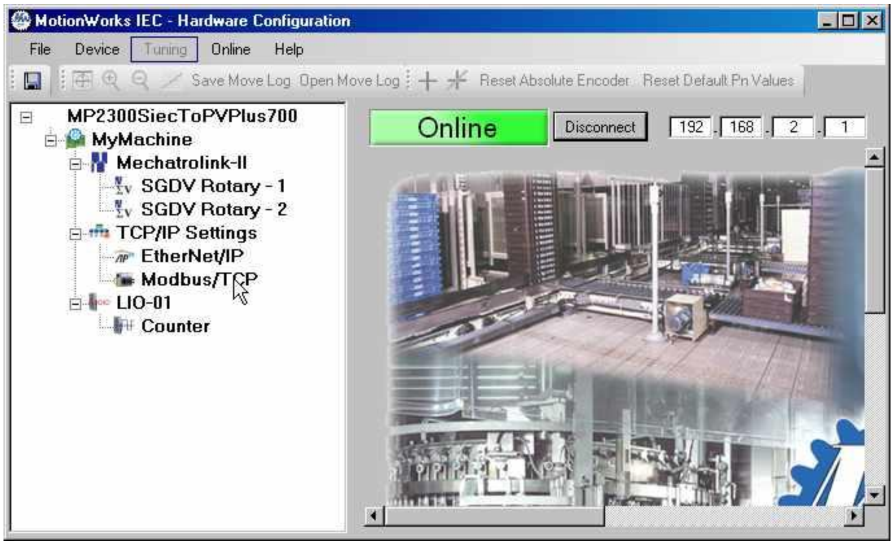
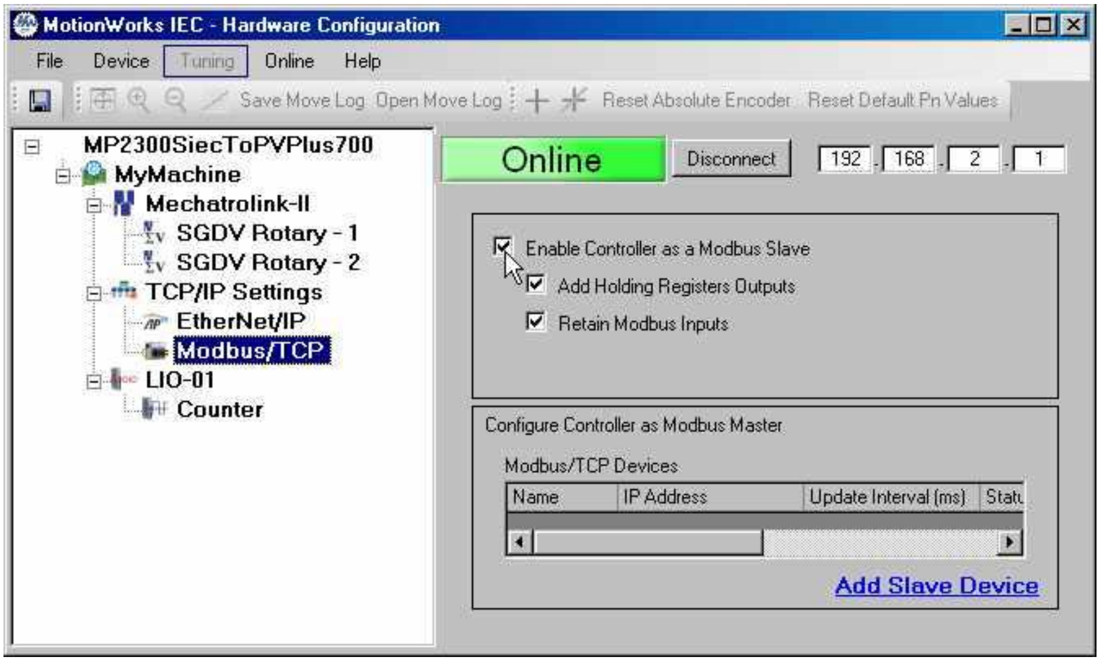


Figure 16: Enter Modbus/TCP configuration

Check the box to enable the controller as a Modbus Slave as shown in Figure 17. Optionally, check the boxes to add Holding Register Outputs to the configuration and to retain the previous values of Modbus communication registers if communication is lost or restarted. If checked, the loss of recipe data may be prevented.



192 . 168 . 2 . 1

☑ Enable Controller as a Modbus Slave

☑ Add Holding Registers Outputs

☑ Retain Modbus Inputs

Figure 17: Modbus/TCP configuration dialog box

Save the configuration while online, then cycle power to the controller. That's IT!

Step 2) Create Global Variables in the Modbus Address Space

In the Global Variables section, the Hardware Configurator will automatically create groups for Modbus Variables, labeled by Modbus function. It will be up to the programmer to create variables and assign physical addresses within this space. To review, the essential Modbus function are listed in Table 1, and the format for IEC61131-3 addressing is listed in Table 2.

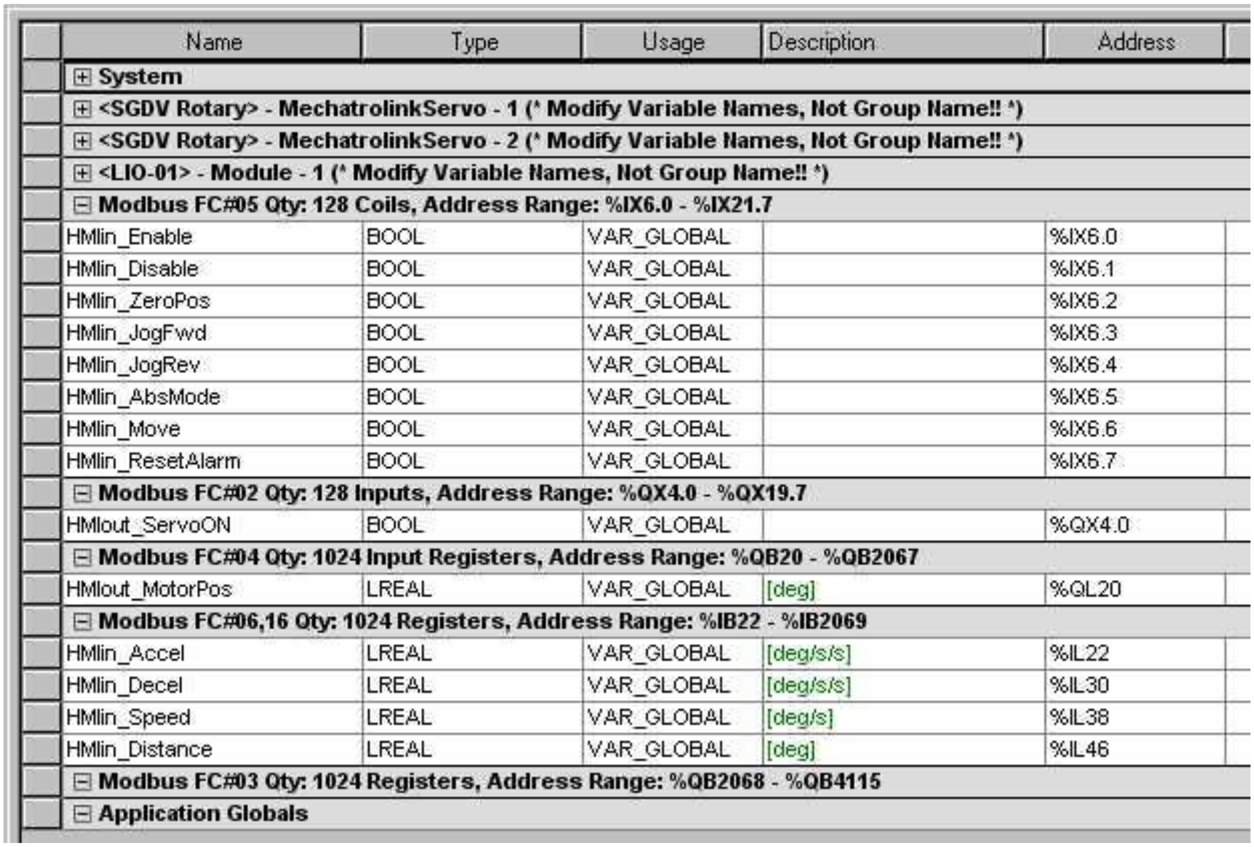
Modbus Function Code	Description
02	Read Single Coil
04	Read Single Holding Register
05	Write Single Coil
06	Write Single Holding Register
16	Write Multiple Holding Registers

Table 1: Modbus Functions

Physical Address			
Location Prefix	Size Prefix	Type	Length
%I - Physical Input %Q - Physical Output %M - Memory Location	X	Bit	1
	B	Byte	8
	W	Word	16
	D	Double Word	32
	L	Long	64

Table 2: IEC61131-3 Addressing format

In the IEC61131-3 system, memory is referenced by the starting Byte number. Therefore, one word contains two bytes, one double word contains four bytes, and one long contains eight bytes. Figure 18 shows the resulting global variable listing as created in the example program. Note how the physical addresses are referenced by the starting byte number and the 64-bit variables in the last section are separated by eight bytes.

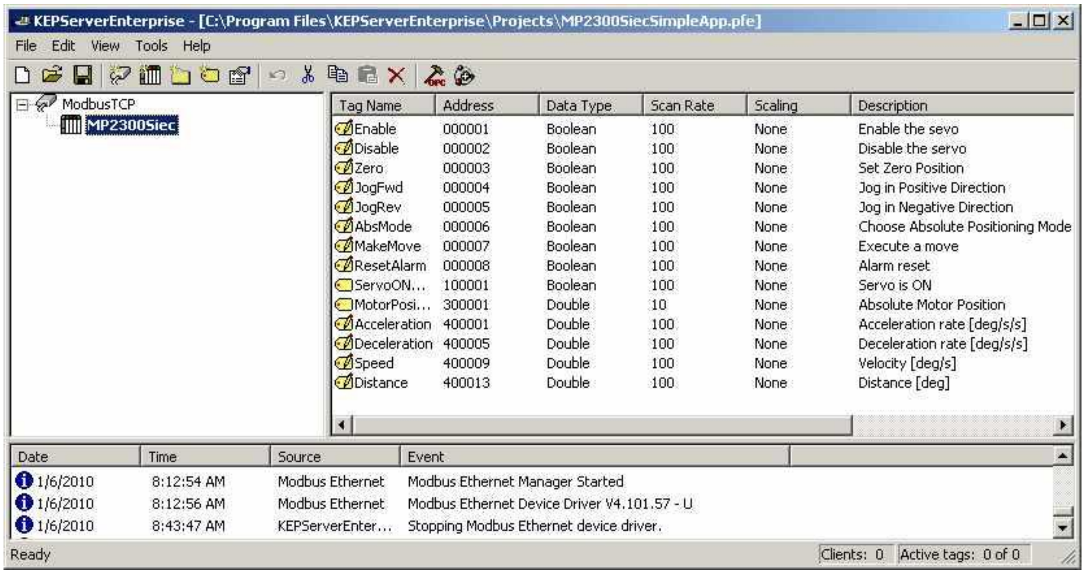


Name	Type	Usage	Description	Address
⊞ System				
⊞ <SGDV Rotary> - MechatrolinkServo - 1 (* Modify Variable Names, Not Group Name!! *)				
⊞ <SGDV Rotary> - MechatrolinkServo - 2 (* Modify Variable Names, Not Group Name!! *)				
⊞ <LIO-01> - Module - 1 (* Modify Variable Names, Not Group Name!! *)				
⊟ Modbus FC#05 Qty: 128 Coils, Address Range: %IX6.0 - %IX21.7				
HMIin_Enable	BOOL	VAR_GLOBAL		%IX6.0
HMIin_Disable	BOOL	VAR_GLOBAL		%IX6.1
HMIin_ZeroPos	BOOL	VAR_GLOBAL		%IX6.2
HMIin_JogFwd	BOOL	VAR_GLOBAL		%IX6.3
HMIin_JogRev	BOOL	VAR_GLOBAL		%IX6.4
HMIin_AbsMode	BOOL	VAR_GLOBAL		%IX6.5
HMIin_Move	BOOL	VAR_GLOBAL		%IX6.6
HMIin_ResetAlarm	BOOL	VAR_GLOBAL		%IX6.7
⊟ Modbus FC#02 Qty: 128 Inputs, Address Range: %QX4.0 - %QX19.7				
HMIout_ServoON	BOOL	VAR_GLOBAL		%QX4.0
⊟ Modbus FC#04 Qty: 1024 Input Registers, Address Range: %QB20 - %QB2067				
HMIout_MotorPos	LREAL	VAR_GLOBAL	[deg]	%QL20
⊟ Modbus FC#06,16 Qty: 1024 Registers, Address Range: %IB22 - %IB2069				
HMIin_Accel	LREAL	VAR_GLOBAL	[deg/s/s]	%IL22
HMIin_Decel	LREAL	VAR_GLOBAL	[deg/s/s]	%IL30
HMIin_Speed	LREAL	VAR_GLOBAL	[deg/s]	%IL38
HMIin_Distance	LREAL	VAR_GLOBAL	[deg]	%IL46
⊟ Modbus FC#03 Qty: 1024 Registers, Address Range: %QB2068 - %QB4115				
⊟ Application Globals				

Figure 18: Global Modbus variables in example program

Step 3) Create a Tag Database within KEPServer Enterprise, an OPC Server acting as the Modbus/TCP Master (Client)

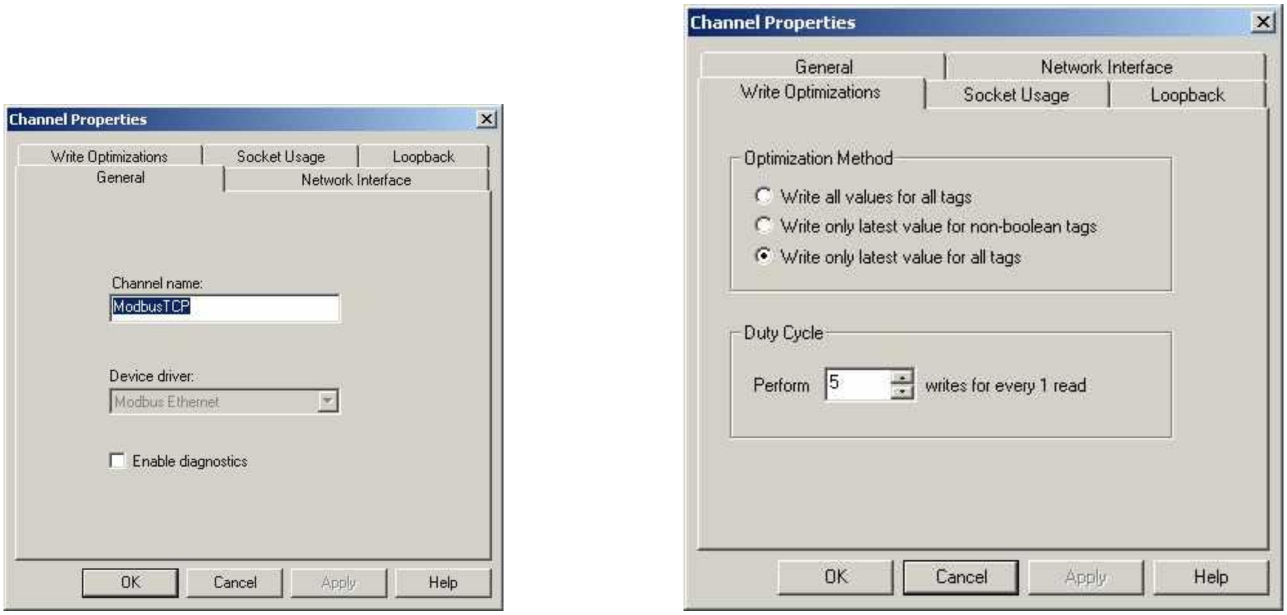
First, a “connection channel” must be established. The example code has created a Channel called “ModbusTCP” and assigned the properties shown Figures 20 and 21.


KEPServerEnterprise - [C:\Program Files\KEPServerEnterprise\Projects\MP2300SiecSimpleApp.pfe]
File
Edit
View
Tools
Help
ModbusTCP
MP2300Siec
Ready
Clients: 0
Active tags: 0 of 0

Tag Name	Address	Data Type	Scan Rate	Scaling	Description
Enable	000001	Boolean	100	None	Enable the sevo
Disable	000002	Boolean	100	None	Disable the servo
Zero	000003	Boolean	100	None	Set Zero Position
JogFwd	000004	Boolean	100	None	Jog in Positive Direction
JogRev	000005	Boolean	100	None	Jog in Negative Direction
AbsMode	000006	Boolean	100	None	Choose Absolute Positioning Mode
MakeMove	000007	Boolean	100	None	Execute a move
ResetAlarm	000008	Boolean	100	None	Alarm reset
ServoON...	100001	Boolean	100	None	Servo is ON
MotorPosi...	300001	Double	10	None	Absolute Motor Position
Acceleration	400001	Double	100	None	Acceleration rate [deg/s/s]
Deceleration	400005	Double	100	None	Deceleration rate [deg/s/s]
Speed	400009	Double	100	None	Velocity [deg/s]
Distance	400013	Double	100	None	Distance [deg]

Date	Time	Source	Event
1/6/2010	8:12:54 AM	Modbus Ethernet	Modbus Ethernet Manager Started
1/6/2010	8:12:56 AM	Modbus Ethernet	Modbus Ethernet Device Driver V4.101.57 - U
1/6/2010	8:43:47 AM	KEPServerEnter...	Stopping Modbus Ethernet device driver.

Figure 19: KEPServer Enterprise main screen showing channel/device tree and tag listing



Channel name: ModbusTCP

Device driver: Modbus Ethernet

□ Enable diagnostics

Optimization Method

○ Write all values for all tags

○ Write only latest value for non-boolean tags

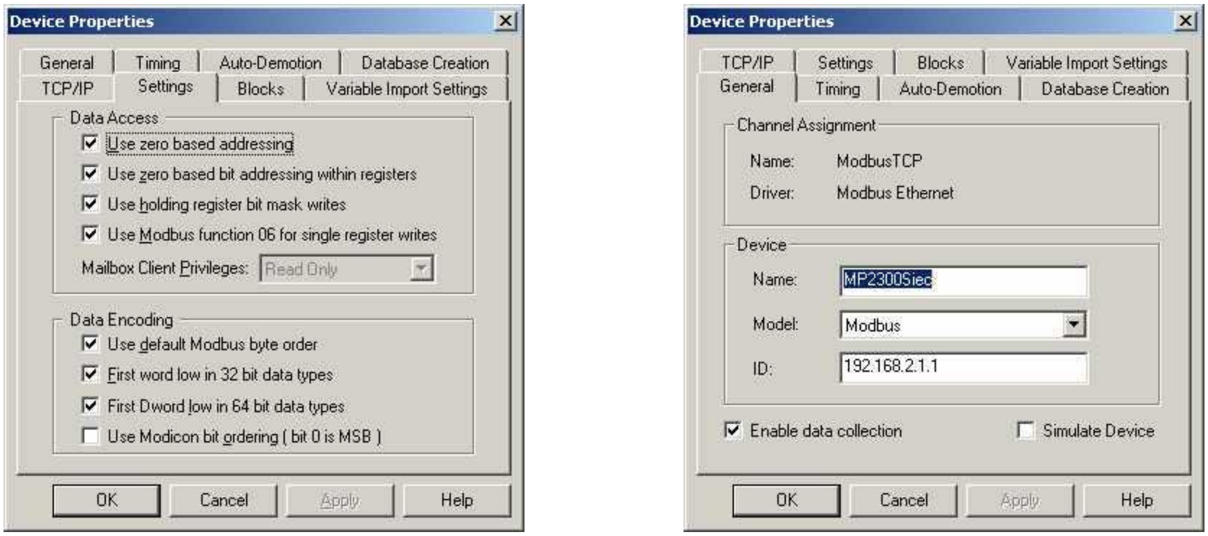
● Write only latest value for all tags

Duty Cycle

Perform 5 writes for every 1 read

Figures 20 and 21: Properties for the Communication Channel

Next, a Device must be added under the Channel. Figures 22 through 26 show the Device Properties for the example application.



Device Properties

General | Timing | Auto-Demotion | Database Creation | TCP/IP | Settings | Blocks | Variable Import Settings

Data Access

☑ Use zero based addressing

☑ Use zero based bit addressing within registers

☑ Use holding register bit mask writes

☑ Use Modbus function 06 for single register writes

Mailbox Client Privileges: Read Only

Data Encoding

☑ Use default Modbus byte order

☑ First word low in 32 bit data types

☑ First Dword low in 64 bit data types

□ Use Modicon bit ordering (bit 0 is MSB)

Device Properties

TCP/IP | Settings | Blocks | Variable Import Settings | General | Timing | Auto-Demotion | Database Creation

Channel Assignment

Name: ModbusTCP

Driver: Modbus Ethernet

Device

Name: MP2300Siec

Model: Modbus

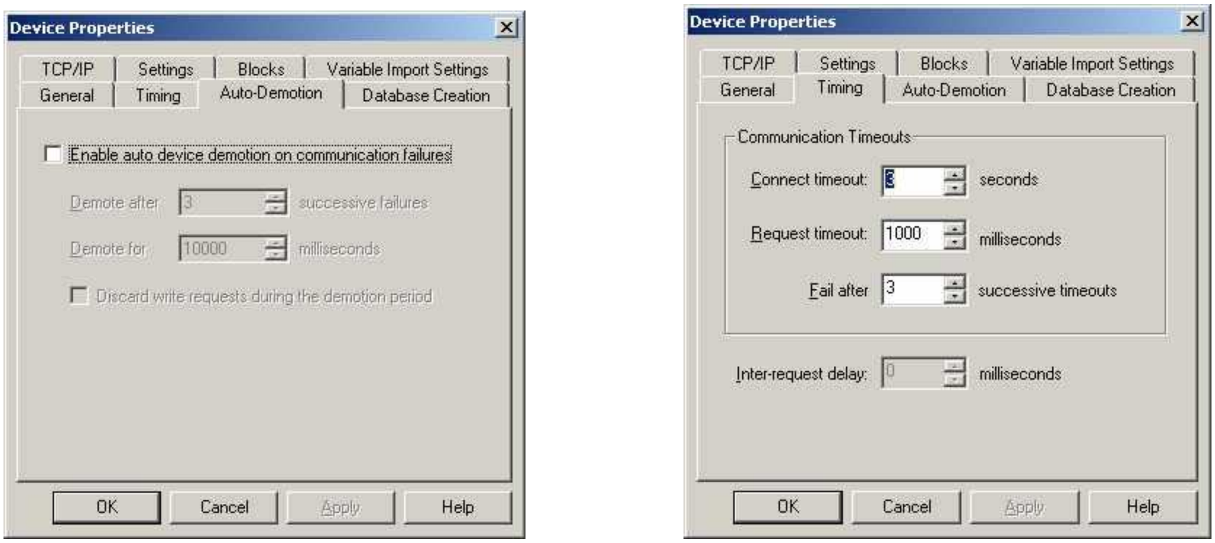
ID: 192.168.2.1.1

☑ Enable data collection

□ Simulate Device

Figures 22 and 23: Data settings and General Settings for the remote Device

Under the 'General' tab in Figure 23, the device is given a name and the TCP/IP address is entered. Note the extra ".1" added to the end of the normal address. This is an essential device identifier for KEPServer Enterprise and is necessary for proper communications. Properties under the remaining tabs are essentially left at default values. For Modbus applications, the TCP/IP port should be set to 502.



☐ Enable auto device demotion on communication failures

Demote after: 3 successive failures

Demote for: 10000 milliseconds

☐ Discard write requests during the demotion period

Communication Timeouts

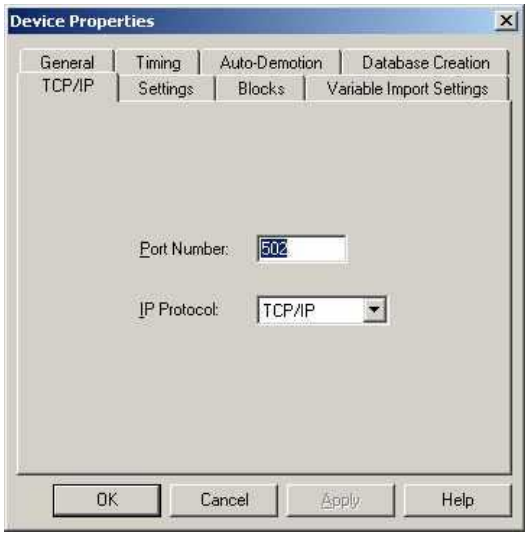
Connect timeout: 3 seconds

Request timeout: 1000 milliseconds

Fail after 3 successive timeouts

Inter-request delay: 0 milliseconds

Figure 24 and 25: Auto-Demotion and Timing settings for the remote Device



Port Number: 502

IP Protocol: TCP/IP

Figure 26: TCP/IP Port setting.

Once the Channel and Device are configured, the programmer can create Tags under the Device. During Tag entry, a Modbus *address* and *datatype* will need to be specified. These settings are critical and must match the corresponding variable in the remote device. However, the syntax for both of these settings is different in KEPServer than in IEC61131-3. Datatype conversions are shown in Table 3.

Datatype Conversion			
IEC61131-3	**Size**	**Signed**	**KEPServer**
BOOL	1	no	Boolean
SINT	8	yes	Char
INT	16	yes	Short
DINT	32	yes	Long
USINT	8	no	Byte
UINT	16	no	Word
UDINT	32	no	Dword
REAL	32	yes	Float
LREAL	64	yes	Double
TIME	32	no	-
BYTE	8	no	Byte
WORD	16	no	Word
DWORD	32	no	DWord

Table 3: Datatype conversions between IEC61131-3 and KEPServer

The Modbus addressing scheme relies on a pre-defined image with fixed address sections accessible by particular Modbus Functions. The Modbus Memory area image shown in Figure 27 lists the fixed address space. For example, the Modbus command to write a Holding Register points into the memory area starting at 40000. Likewise, the Modbus command to read a single coil register points to the area starting at 10000.

Important Note → Modbus uses a 1-based addressing scheme. Therefore, the first data element in each section will actually be found at the starting address + 1 (i.e. 10001, 30001, 40001)

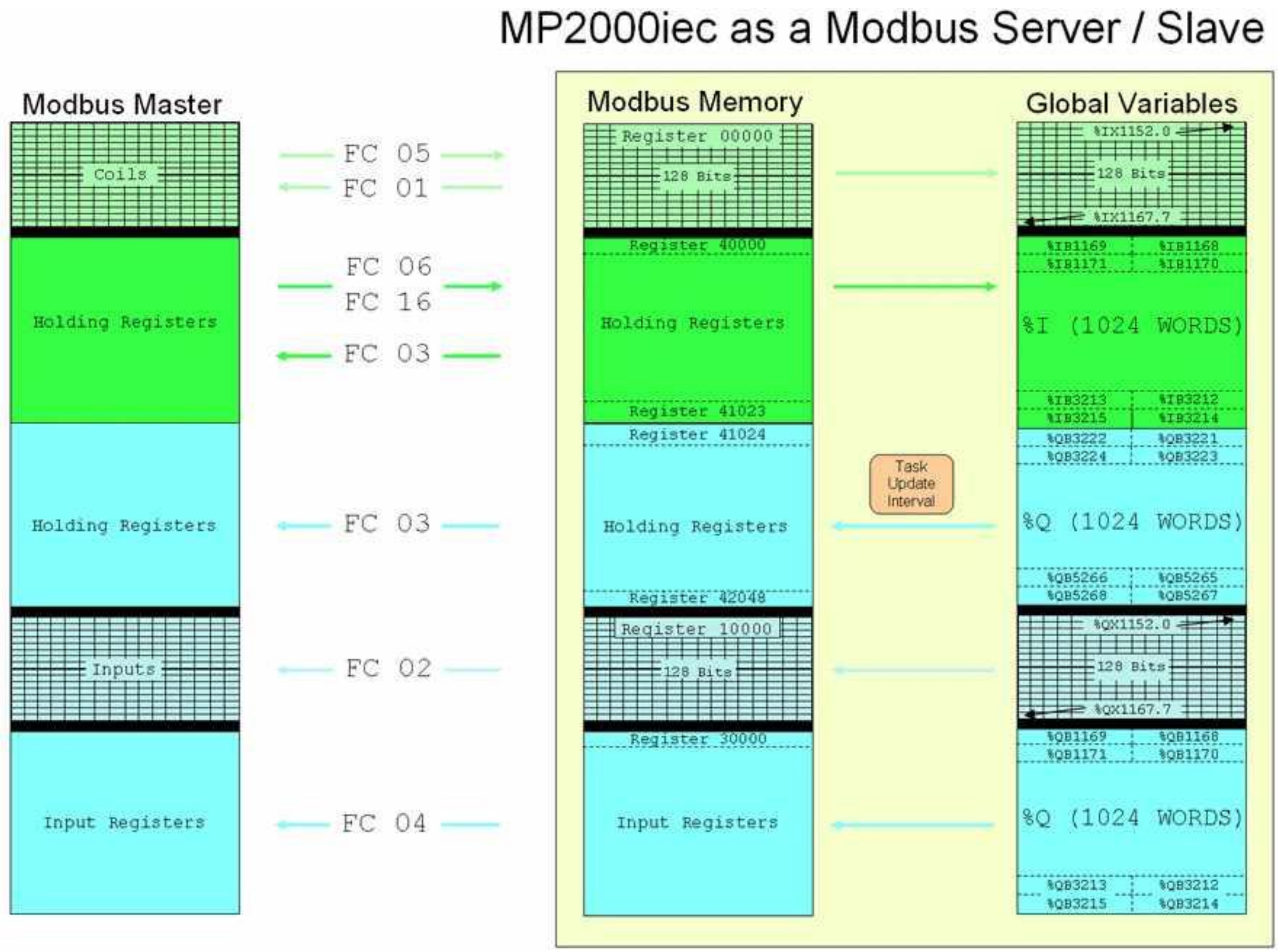
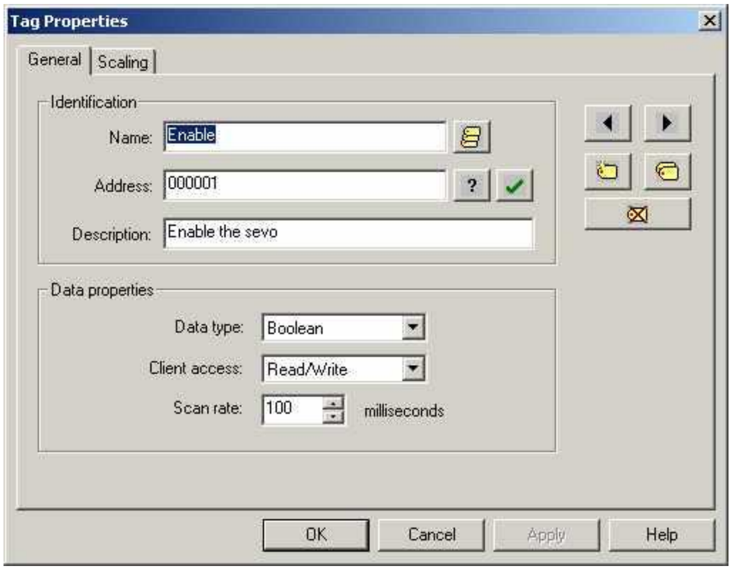


Figure 27: Modbus Address Map

Looking at a few of the Tag properties from the list under the KEPServer device, we see that the tag named “Enable” has been set to a Boolean datatype mapped to address 000001. It was placed in this space so the master could write the data to the slave with FC 05. Similarly, the tag named “MotorPosition” has been defined as a Double datatype and maped into the area of holding registers to be read by the master at address 300001.


Tag Properties

General | Scaling

Identification

Name: Enable

Address: 000001

Description: Enable the sevo

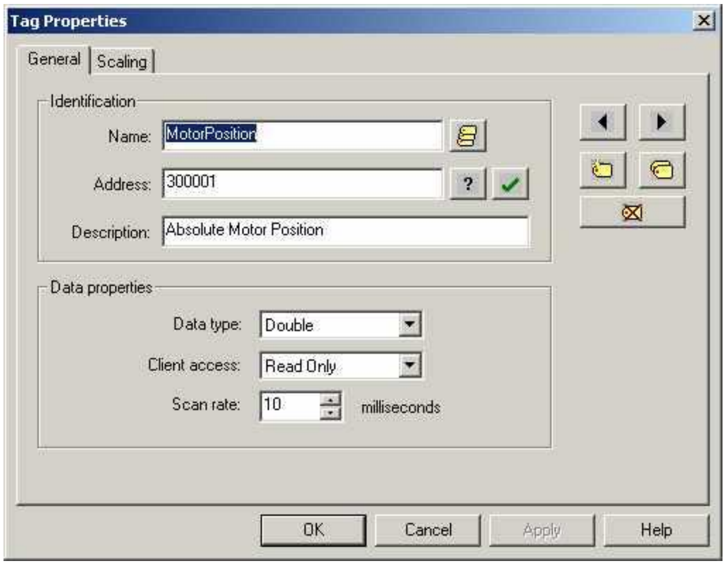
Data properties

Data type: Boolean

Client access: Read/Write

Scan rate: 100 milliseconds

Figure 28: Tag properties for “Enable”


Tag Properties

General | Scaling

Identification

Name: MotorPosition

Address: 300001

Description: Absolute Motor Position

Data properties

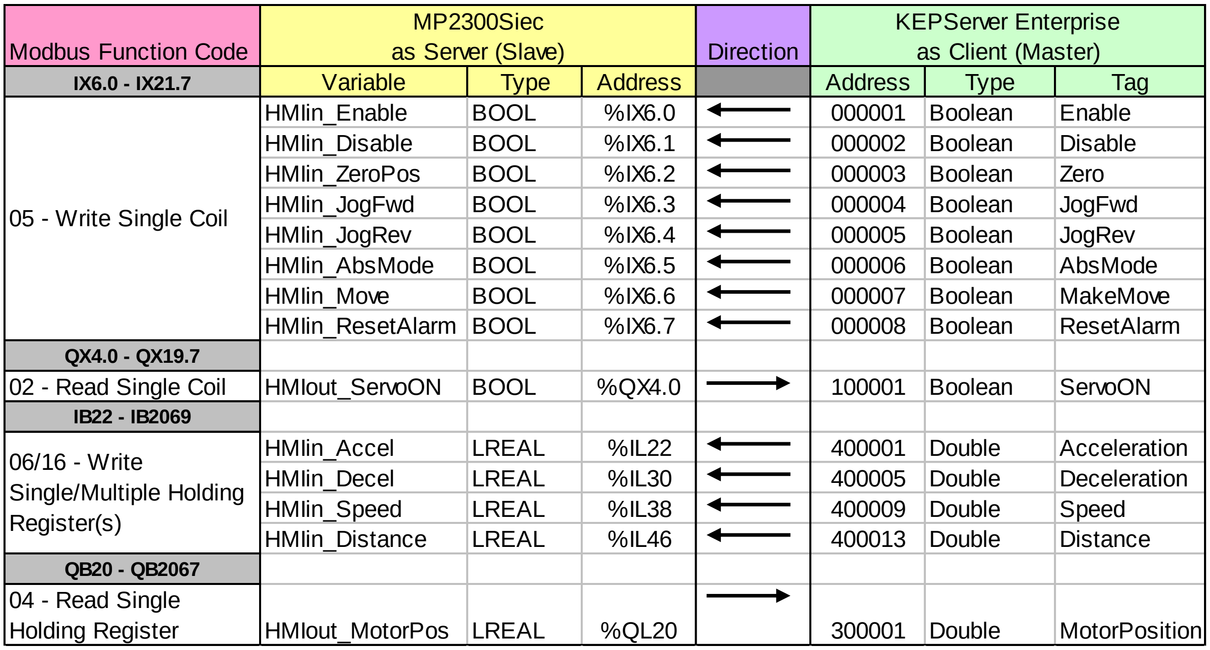
Data type: Double

Client access: Read Only

Scan rate: 10 milliseconds

Figure 29: Tag properties for “MotorPosition”

The rest of the tags are defined in the example program to match the Global Variables in the MPiec series machine controller. The final mapping between the MotionWorks IEC program and the KEPServer OPC server is shown in Figure 30.

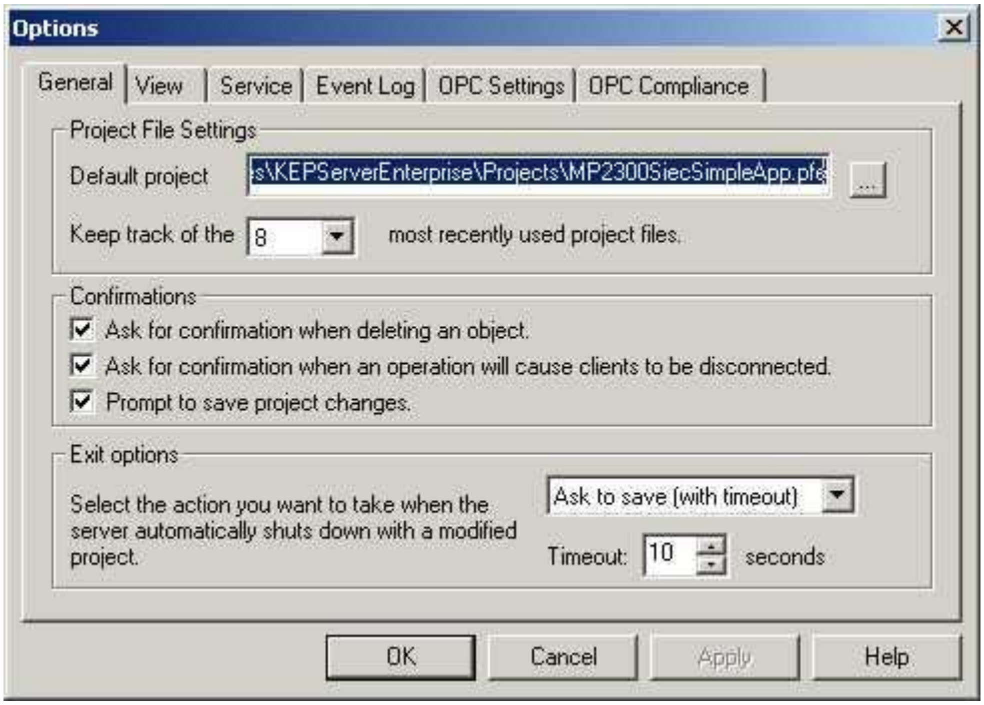


Modbus Function Code	MP2300Siec as Server (Slave)			Direction	KEPServer Enterprise as Client (Master)		
IX6.0 - IX21.7	Variable	Type	Address		Address	Type	Tag
05 - Write Single Coil	HMIin_Enable	BOOL	%IX6.0	←	000001	Boolean	Enable
	HMIin_Disable	BOOL	%IX6.1	←	000002	Boolean	Disable
	HMIin_ZeroPos	BOOL	%IX6.2	←	000003	Boolean	Zero
	HMIin_JogFwd	BOOL	%IX6.3	←	000004	Boolean	JogFwd
	HMIin_JogRev	BOOL	%IX6.4	←	000005	Boolean	JogRev
	HMIin_AbsMode	BOOL	%IX6.5	←	000006	Boolean	AbsMode
	HMIin_Move	BOOL	%IX6.6	←	000007	Boolean	MakeMove
	HMIin_ResetAlarm	BOOL	%IX6.7	←	000008	Boolean	ResetAlarm
QX4.0 - QX19.7							
02 - Read Single Coil	HMIout_ServoON	BOOL	%QX4.0	→	100001	Boolean	ServoON
IB22 - IB2069							
06/16 - Write Single/Multiple Holding Register(s)	HMIin_Accel	LREAL	%IL22	←	400001	Double	Acceleration
	HMIin_Decel	LREAL	%IL30	←	400005	Double	Deceleration
	HMIin_Speed	LREAL	%IL38	←	400009	Double	Speed
	HMIin_Distance	LREAL	%IL46	←	400013	Double	Distance
QB20 - QB2067							
04 - Read Single Holding Register	HMIout_MotorPos	LREAL	%QL20	→	300001	Double	MotorPosition

Figure 30: Final address mapping between MP200iec controller and KEPServer Enterprise

After all the tags have been defined, it is a good idea to test the connection to verify the information is transferring properly. This can be done by launching "Quick Client" from the toolbar. Click on the Quick Client Icon . A connection will be established with the slave device so that tag status can be monitored. If the tag is passing information properly, the status under the Quality column will indicate "Good". Use the Debug mode on the MPiec series machine controller to change values and observe the corresponding change in Tag values in the Quick Client. Similarly, the programmer can right-click on the tag value and choose "Synchronous Write" to change a Tag's value. A corresponding change in the controller global variable should also be observed.

The last step when configuring the KEPServer Enterprise file is to set it as the default, or startup file. The procedure is as shown in Figure 31. Go under Tools_Options_GeneralTab and browse for the desired project file.



Project File Settings

Default project: s\KEPServerEnterprise\Projects\MP2300SiecSimpleApp.pfe

Keep track of the 8 most recently used project files.

Confirmations

☑ Ask for confirmation when deleting an object.

☑ Ask for confirmation when an operation will cause clients to be disconnected.

☑ Prompt to save project changes.

Exit options

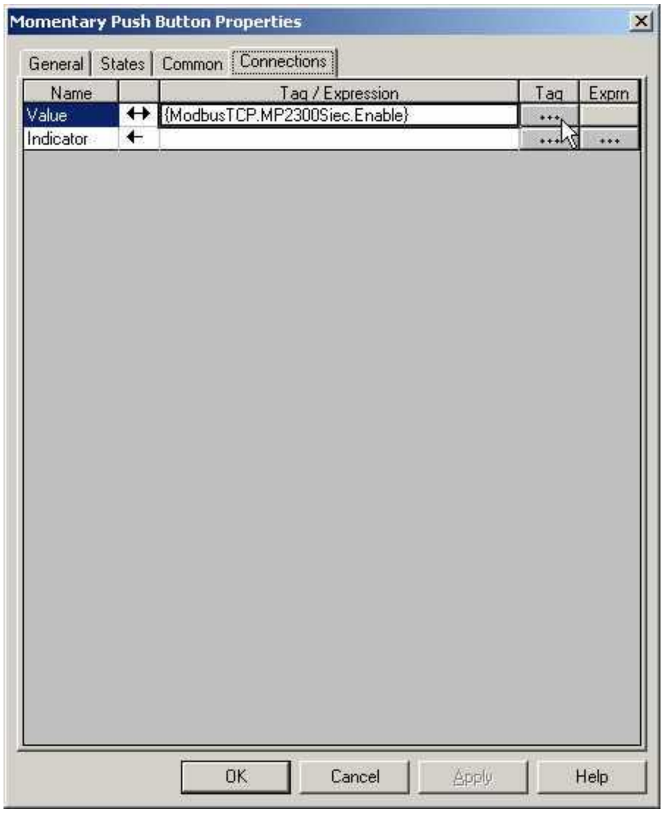
Select the action you want to take when the server automatically shuts down with a modified project. Ask to save (with timeout)

Timeout: 10 seconds

Figure 31: Setting the startup file in KEPServer Enterprise

Step 4) Create HMI Screens with Elements tied to the OPC Server

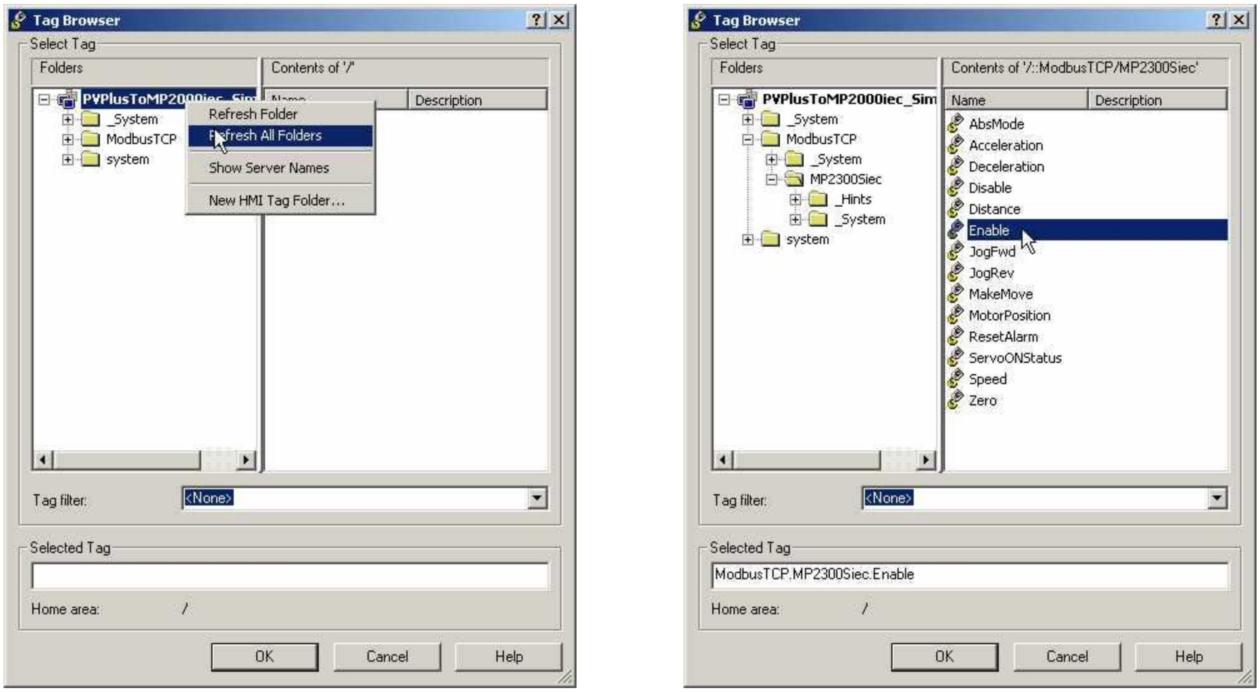
The final step in the process is to add graphic elements such as pushbuttons, gauges, and numeric indicators to a display screen created in RSView Studio and connect them to the OPC tags. To make this connection, open the display found in the RSView project, double-click on an element to bring up its properties dialog box, and go under the ‘Connections’ tab. Figure 32 shows the dialog for the ‘Enable’ momentary pushbutton element.



Name		Tag / Expression	Tag	Expr
Value	↔	{ModbusTCP.MP2300Siec.Enable}	...	
Indicator	←		...	...

Figure 32: ‘Enable’ momentary pushbutton properties

Click on the ellipses under the Tag column to browse for the OPC tag. If changes have been made to the OPC server file, be sure to right-click on the project and choose ‘Refresh All Folders’ to get the latest listings. Navigate under the proper Connection and Device folder to view and select from the available tags as shown in Figure 33.



Tag filter: <None>

Home area: /

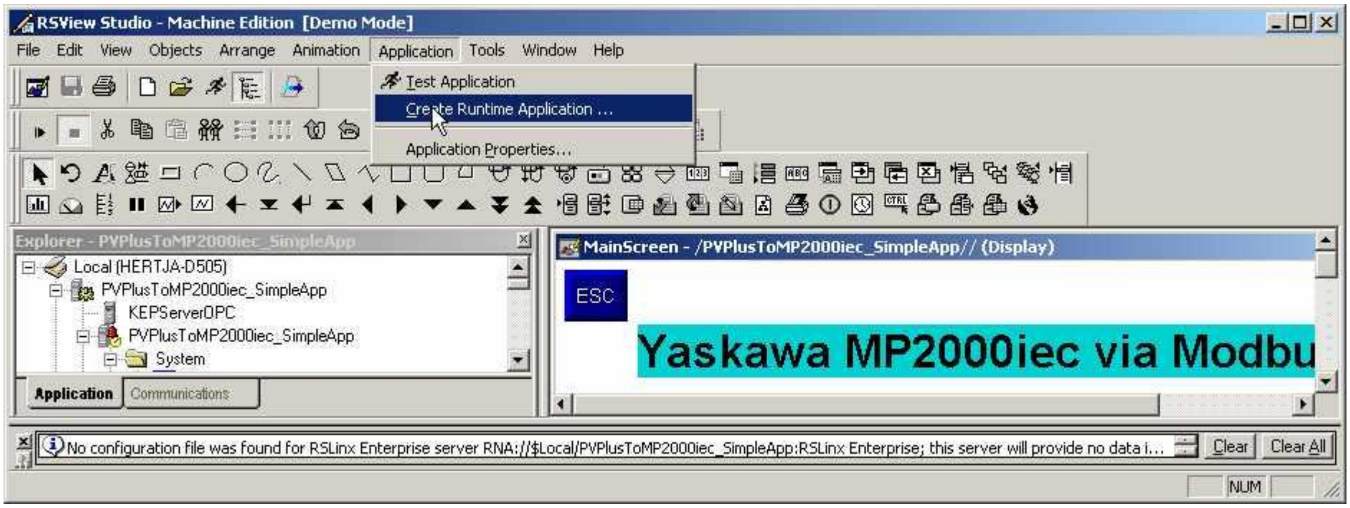
Tag filter: <None>

Selected Tag: ModbusTCP.MP2300Siec.Enable

Home area: /

Figure 33: Browsing the available tags for connection to a graphical element

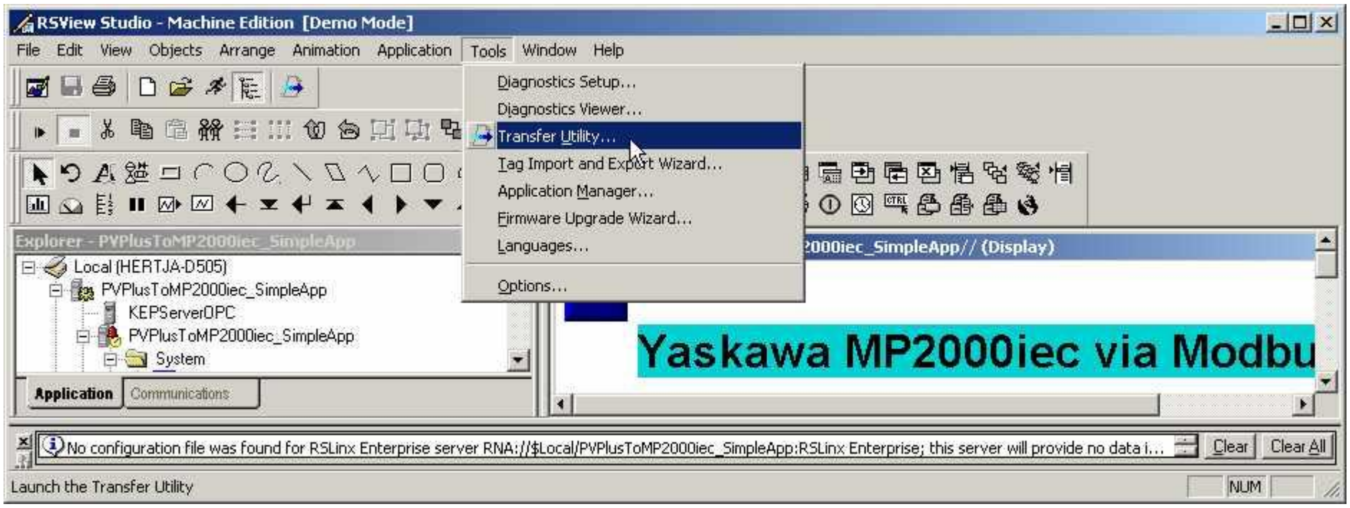
Once all elements in the display are connected to OPC tags, the project is ready to be compiled and transferred to the HMI. To compile, choose Application → Create Runtime Application… from the main menu bar of RSView Studio. Give the runtime file an appropriate name and location.


RSView Studio - Machine Edition [Demo Mode]
File
Edit
View
Objects
Arrange
Animation
Application
Tools
Window
Help
Test Application
Create Runtime Application ...
Application Properties...
Explorer - PVPlusToMP2000iec_SimpleApp
Local (HERTJA-D505)
PVPlusToMP2000iec_SimpleApp
KEPServerOPC
PVPlusToMP2000iec_SimpleApp
System
Application
Communications
MainScreen - /PVPlusToMP2000iec_SimpleApp// (Display)
ESC
Yaskawa MP2000iec via Modbu
Clear
Clear All
NUM

No configuration file was found for RSLinx Enterprise server RNA://$Local/PVPlusToMP2000iec_SimpleApp:RSLinx Enterprise; this server will provide no data i...

Figure 34: Compiling the HMI project

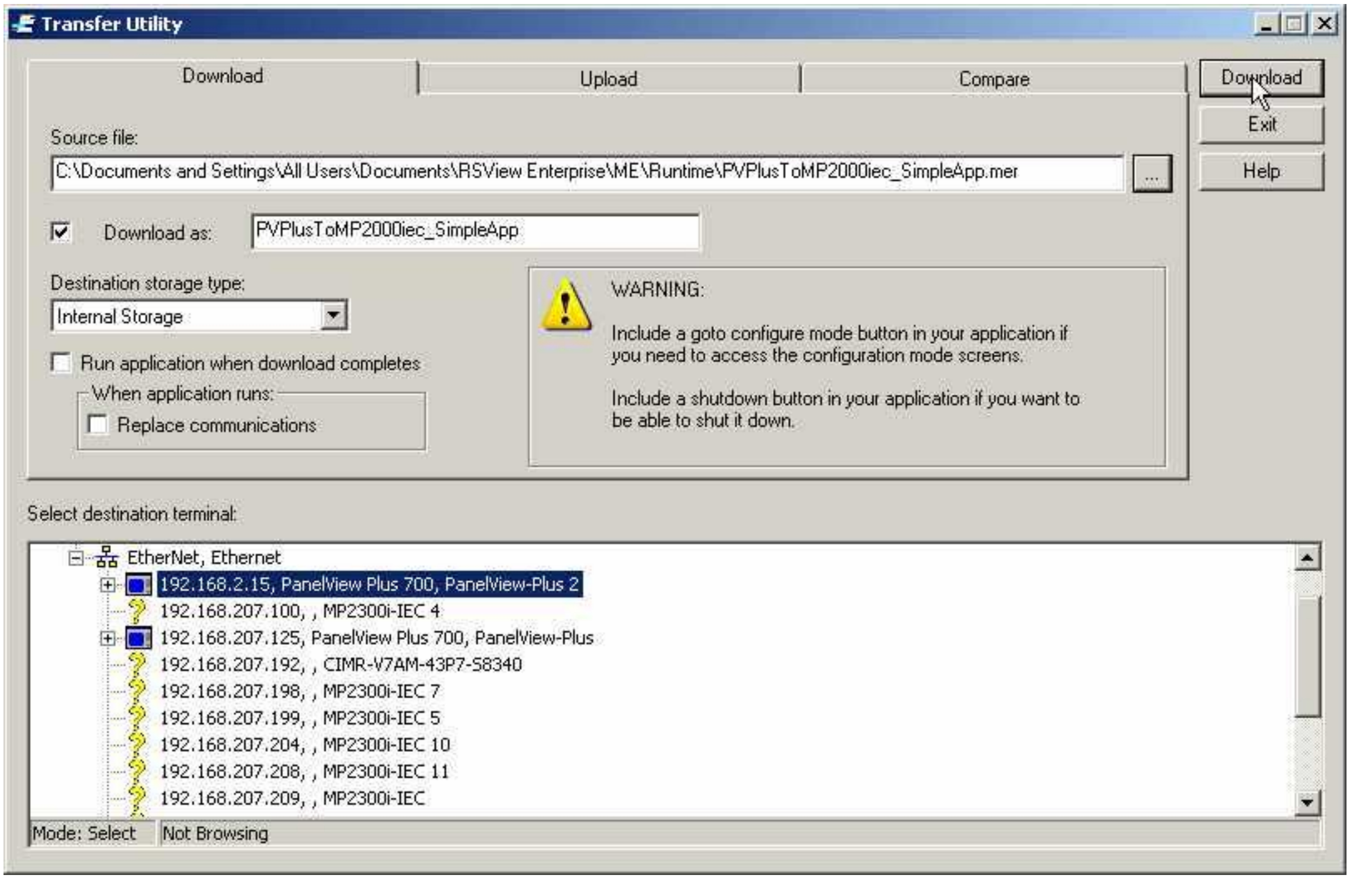
When finished, transfer the runtime project to the HMI using the Transfer Utility found under the Tools menu.


RSView Studio - Machine Edition [Demo Mode]
File
Edit
View
Objects
Arrange
Animation
Application
Tools
Window
Help
Diagnostics Setup...
Diagnostics Viewer...
Transfer Utility...
Tag Import and Export Wizard...
Application Manager...
Firmware Upgrade Wizard...
Languages...
Options...
Explorer - PVPlusToMP2000iec_SimpleApp
Local (HERTJA-D505)
PVPlusToMP2000iec_SimpleApp
KEPServerOPC
PVPlusToMP2000iec_SimpleApp
System
Application
Communications
2000iec_SimpleApp// (Display)
Yaskawa MP2000iec via Modbu
Clear
Clear All
NUM

No configuration file was found for RSLinx Enterprise server RNA://$Local/PVPlusToMP2000iec_SimpleApp:RSLinx Enterprise; this server will provide no data i...

Launch the Transfer Utility

Figure 35: Tools_Transfer Utility…



Source file: C:\Documents and Settings\All Users\Documents\RSView Enterprise\ME\Runtime\PVPlusToMP2000iec_SimpleApp.mer

☑ Download as: PVPlusToMP2000iec_SimpleApp

Destination storage type: Internal Storage

□ Run application when download completes

When application runs:

□ Replace communications

WARNING:

Include a goto configure mode button in your application if you need to access the configuration mode screens.

Include a shutdown button in your application if you want to be able to shut it down.

Select destination terminal:

```
EtherNet, Ethernet
  192.168.2.15, PanelView Plus 700, PanelView-Plus 2
  192.168.207.100, , MP2300i-IEC 4
  192.168.207.125, PanelView Plus 700, PanelView-Plus
  192.168.207.192, , CIMR-V7AM-43P7-S8340
  192.168.207.198, , MP2300i-IEC 7
  192.168.207.199, , MP2300i-IEC 5
  192.168.207.204, , MP2300i-IEC 10
  192.168.207.208, , MP2300i-IEC 11
  192.168.207.209, , MP2300i-IEC
```

Mode: Select | Not Browsing

Figure 36: Transfer Dialog box

Browse to the runtime file, provide a name for the file as it will appear in the HMI, and select the HMI device from the list of devices configured on the network.

Operation

To run the program on the HMI, first Load, then Run the project. An alarm popup should appear indicating that connections are properly established. This box can be cleared so that the rest of the display's buttons will be accessible. Finished!