# TECHNICAL NOTE

*MOTION PRODUCT AND ENGINEERING GROUP*

**Subject:** MP940 Network Setup via MotionWorks +

**Product:** MP940/MW+

**Summary:** This technical note outlines the details in assigning and the setup of Master/Slaves in a MP940 Network System. Two types of high speed networks exist in the MP940, DeviceNet$^{TM}$ and Mechatrolink. Mechatrolink is used as the primary example, however, the setup from the software is virtually identical when applying DeviceNet$^{TM}$.

## Detailed Summary:

One feature of the MP940 is its ability to network with other components in the system. These components can be other MP940 controllers, digital inputs and outputs, or analog inputs and outputs. When networking, it is necessary to first define a master, which will control the rest of the system. All components in the system that are not the master are considered slaves. The master and slaves use the network to communicate data back-and-forth to each other. This way, it is possible for the outputs (or physical inputs) on one node to affect the inputs (or physical outputs) of another node. Two different network protocols are available for use with the MP940: Mechatrolink, Yaskawa's proprietary high-speed network, and DeviceNet$^{TM}$. All nodes that are connected in the network must use the same protocol with only one Master (MP940 can be a Mechatrolink or DeviceNet$^{TM}$ Network Master). In both cases, the MP940 only supports network I/O in a Polled I/O fashion. Setting up a network between system components can be very powerful if done properly, and can be a necessity for many systems.

MP940 w/ Mechatrolink: Can pass up to eight 16-bit input words and eight 16-bit output words.

MP940 w/ Devicenet: Can pass up to 256 bytes

Note: Throughout this document, DeviceNet$^{TM}$ and MotionWorks+$^{TM}$ will be used, all uses of this word are in reference to the trademark.

## Table Of Contents:

## Objective:

The objective of this Technical Note is to demonstrate how to set up a network with 2 or more nodes over either Mechatrolink or DeviceNet. Although this paper highlights the details for Mechatrolink using an MP940 as a master, similar methods can be applied when setting up a DeviceNet network with an MP940 as a master. MotionWorks+ 2.XX™ will be used to properly configure the master and slaves for networking, and to have the nodes communicating with each other.

## Background:

It is assumed that the user has some prior understanding of how to program in MotionWorks+ and can create the code needed for the desired motion. Also, this document will not discuss the wiring and configuration of the servo, amplifier, and controller. This is the responsibility of the user.

For examples of network setup using MotionWorks+ (minimum ver 2.05), refer to your C:/Program Files/Yaskawa/Motion Works+/Examples directory which will be available after program installation. Also refer to Motion Works+ users manual YEA-SIA-C887-1.5C

## Setting-Up Network Hardware:

Before starting, one MP940 should be designated as the master controller. All other nodes (MP940 or an I/O module) will be the slaves of the master module. Several possible configurations are shown in Figure 1 (a-c), however, only one master can be on a single network at one time.

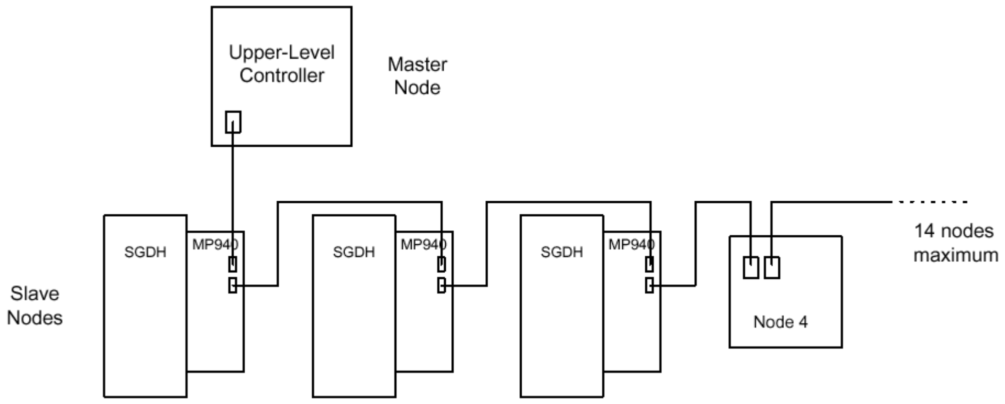
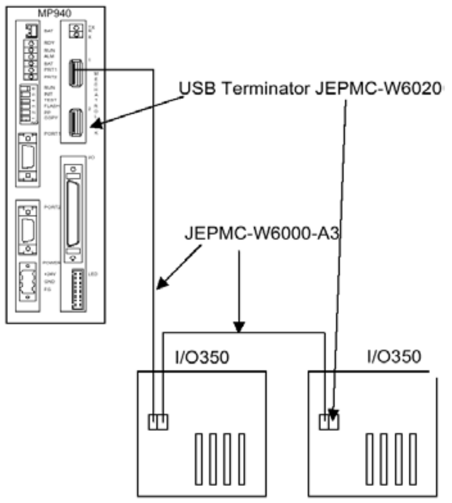


**Figure 1-a: Sample Configuration 1**



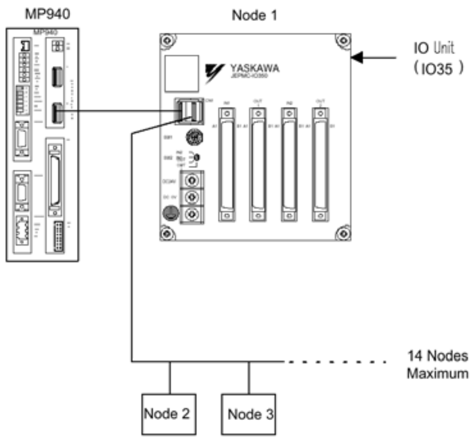


Figure 1-b: Sample Configuration 2 Figure 1-c: Sample Configuration 3

It is important to configure and program the master node first. This is because all the slave nodes, inputs, outputs, and variables need to be defined in the master before they can be used in the slave nodes.

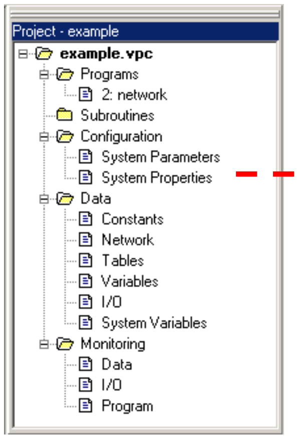
## MotionWorks+ Master Program

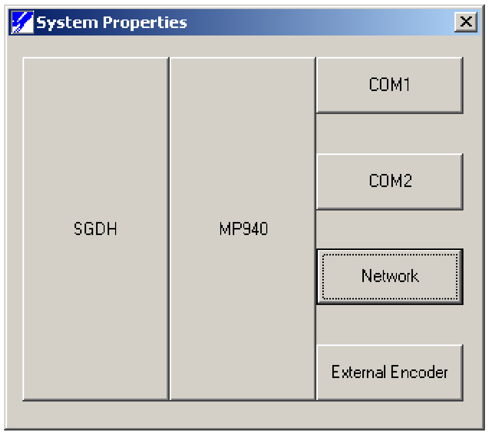
### *Configuring the Master Node:*

To begin, open a new or existing project.

To properly configure MotionWorks+ for networking, it is necessary to first begin by changing the *System Properties*.

1. Double-click on *System Properties* in the Project Window under the *Configuration* heading.
2. Single-click on the *Network* button in the *System Properties* window.
3. Make changes to the *Network Properties*





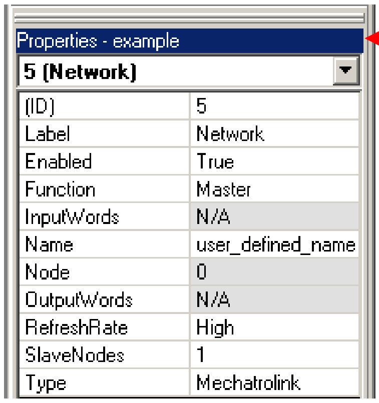
a. Change *Enabled* to "True"

b. Change *Function* to "Master"

c. Use a short description of the axis function in the *Name* field

d. Enter how many slave nodes will be used in the *SlaveNodes* field

e. Change *Type* to either "Mechatrolink" or "DeviceNet"



| | |
|---|---|
| (ID) | 5 |
| Label | Network |
| Enabled | True |
| Function | Master |
| InputWords | N/A |
| Name | user_defined_name |
| Node | 0 |
| OutputWords | N/A |
| RefreshRate | High |
| SlaveNodes | 1 |
| Type | Mechatrolink |

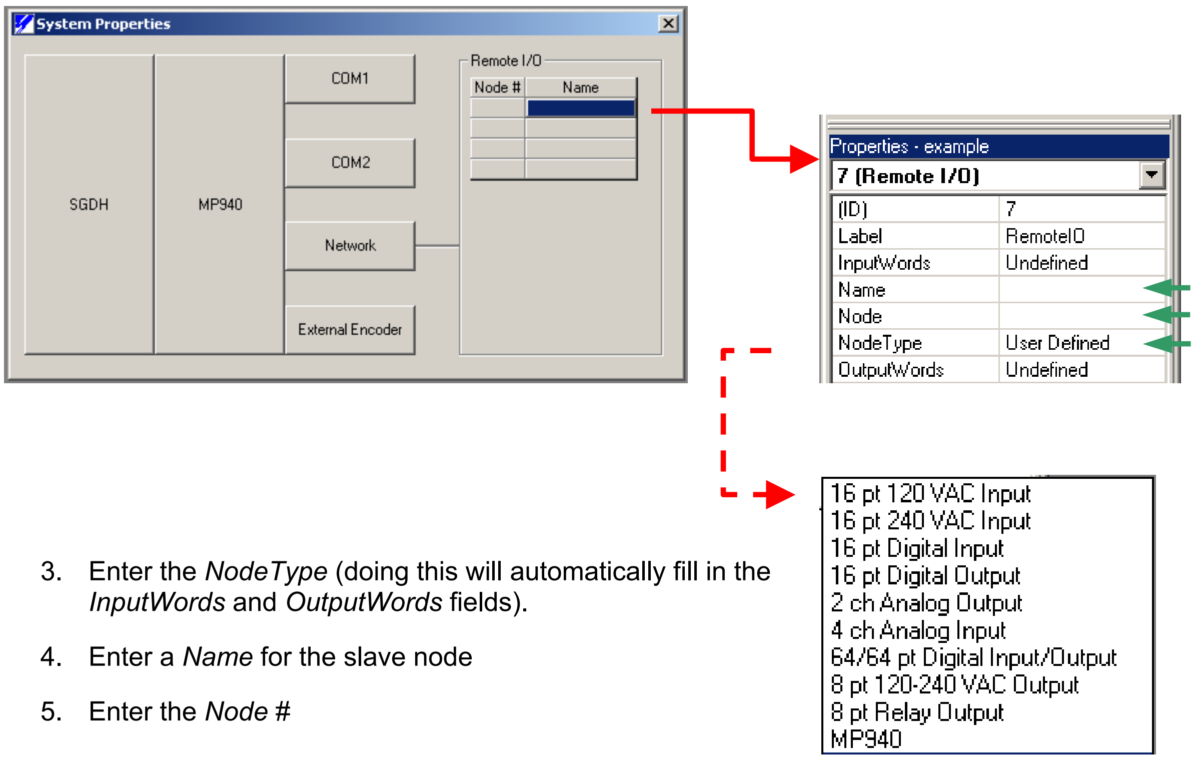
## Properties of the Network

| | |
|---|---|
| (ID): | The ID of the current module (set automatically). |
| Label: | The label for this network. |
| Enabled: | This block activates or deactivates the function upon execution. |
| Function: | Select either *master* or *slave*. |
| Input Words: | This property is only enabled if the *Function* property is set to slave. It defines the number of Input words for the MP940 as a slave. If the *Function* property is set to Master, then this value is automatically determined by the total input words from the *Slave Node* list. For Mechatrolink, the maximum per slave unit = 8-16 bit words. For DeviceNet, the maximum is 256 bytes. In both cases, the minimum=8 |
| Name: | Name for the slave node. This will be used for the purpose of providing default I/O names in the allocation wizard. |
| Node: | Set the address of the module. The maximum value is dependent on the *Type* property (Mechatrolink: 63, DeviceNet: 64). |
| Output Words: | This property is only enabled if the *Function* property is set to slave. It defines the number of Output words for the MP940 as a slave. If the *Function* is set to Master, then this value is determined by the total output words from the *Slave Node* list. For Mechatrolink, the maximum per slave unit = 8-16 bit words. For DeviceNet, the maximum is 256 bytes. In both cases, the minimum=8 |
| Refresh Rate: | Select either *high* or *low* scan. This will be the interval that all the network data is updated within the user program. Note that the network itself may operate at a faster rate and is set automatically by MW+: For Mechatrolink, the network transfer rate is 1ms for 1-6 nodes, 2ms for 7-14 nodes, 4ms for 15-29 nodes. This should be considered when setting the MP940 scan time property setting and Refresh Rate network setting. |
| Slave Nodes: | This property is only enabled if the *Function* is set to master. It defines the number of slaves on the network (Mechatrolink: 0-29, DeviceNet: 0-63). |
| Type: | Select either Mechatrolink or DeviceNet. Note: Changing this parameter after setting up *Remote I/O Properties* will delete all nodes. |

As soon as a number is entered in the *SlaveNodes* field, the *System Properties* window expands to show a *Remote I/O* branch coming out of the *Network* button (see below). Listed under the *Remote I/O* heading are 2 columns: *Node #* and *Name*. The number of rows should correspond to what was entered in the *SlaveNodes* field.

## *Configuring the Slave Node:*

1. Begin by clicking on a cell in the *Name* column.
2. You will then see the *Properties Window* changed to *Remote I/O*



Properties - example
7 (Remote I/O)

| | |
|---|---|
| (ID) | 7 |
| Label | RemoteIO |
| InputWords | Undefined |
| Name | |
| Node | |
| NodeType | User Defined |
| OutputWords | Undefined |

16 pt 120 VAC Input
16 pt 240 VAC Input
16 pt Digital Input
16 pt Digital Output
2 ch Analog Output
4 ch Analog Input
64/64 pt Digital Input/Output
8 pt 120-240 VAC Output
8 pt Relay Output
MP940

3. Enter the *NodeType* (doing this will automatically fill in the *InputWords* and *OutputWords* fields).
4. Enter a *Name* for the slave node
5. Enter the *Node #*

Continue changing the properties for each slave node until all are completed. The updates to the *Name* and *Node* fields are shown in the *System Properties* window in the *Remote I/O* area.

Brief explanations for the *Remote I/O Properties* are shown below:

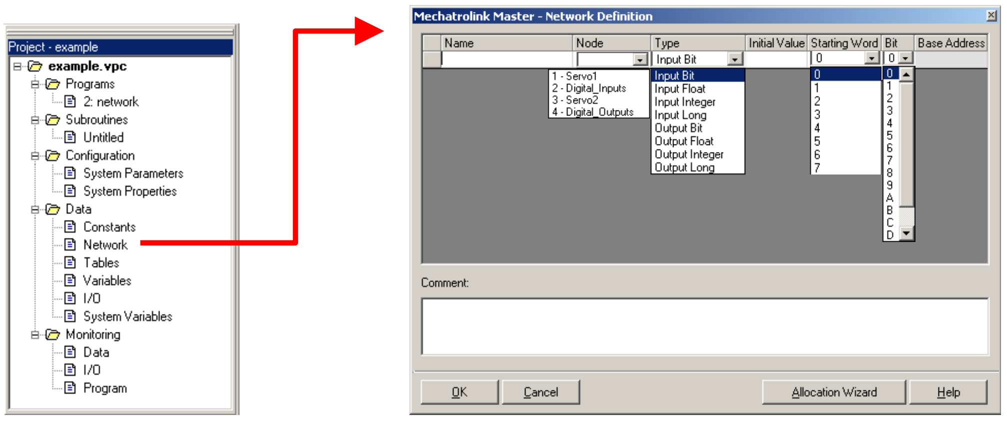
## Properties of the Remote I/O

| | |
|---|---|
| Input Words: | This property is only enabled if the *Node Type* property is set to "User Defined". This is the number of input words allocated to this node. Note: If the network *Type* is "DeviceNet", this property defines the number of Input Bytes. |
| Name: | User-defined description of the network device that will be useful in identifying this node. The name will also be used if the network variable *Allocation Wizard* is used. |
| Node: | Set the address of the module. This applies to "DeviceNet" and "Mechatrolink". The maximum value is dependent on the *Node Type* property: Mechatrolink is 29 and DeviceNet is 64. |
| Node Type: | This is a list of all available Mechatrolink nodes. When a particular module is selected, the input and output words are automatically set. |

## *Defining the Network Data*

Now that the master and slave nodes have been configured, the network data needs to be defined. The network data is all the information that will be passed between controllers and I/O. This information can come in the form of bits, floating point, integer, or long numbers. It can be used to trigger events, control certain parameters of motion control (time, velocity, torque, position, etc.), or set outputs (alarm LED's, pushbutton lights, etc). But before any motion can be created, the variables need to be defined.

1. Double-click on the *Network* under the *Data* branch of the *Properties Window*.



2. Enter all variables that will be used on the network

   a. Name: The user-given name of the network variables.

   b. Node: Choose which node the variable will be communicated to.

   c. Type: Determines the potential magnitude of the variable (what type of number it is: bit, integer, long, floating point)

   d. Initial Value: Presets the value of the variable when the power is turned on. If no initial value is entered, the variable is not initialized when the power is turned on (which is useful for preserving variable values when the power is cycled).

   e. Starting Word: The cells in this column contain a drop-down box. The contents are a list of valid words starting at zero. The maximum value is derived from either of two places, depending on the configuration of the controller, as follows:

      i. If the controller is configured as a network master, the Starting Word range is defined by the *System Properties* > *Remote I/O Block* > *Input/Output Word* properties.

      ii. If the controller is configured as a network slave, the Starting Word range is defined by the *System Properties* > *Network Block* > *Input/Output Word* properties.

   f. Bit: Bit selection is only available if the *Type* is "input bit" or "output bit". Select "0" through "F" from the drop-down list. If a bit type variable is not defined, this cell appears gray, and cannot be set.

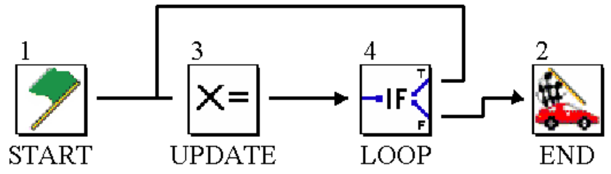
   g. Base Address: A read only field indicating the register location of the variable in the controller memory. This is provided for debugging with MotionWorks™ or accessing data via Memobus serial communication. Base Address is determined by the information specified in the Node, Starting Word, and Bit columns, and prevents overlapping.

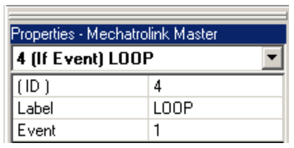
   h. HMI: If this checkbox is selected, the variable will be included in the HMI variable export file (*.dat). This file allows the programmer to import tag names and register allocations directly into an EXOR HMI programming environment (Designer Software), to reduce programming time.

All network variables that are used for all nodes need to be defined in this area. It may be useful to print a copy of these variables and their necessary information (Name, Node, Type, Initial Value, Starting Word, Bit, and Base Address) to use when defining the network variables in the projects that will be loaded onto the slave components. (Note: Only MP940 nodes need projects downloaded to them. I/O modules need only be connected through the Mechatrolink Physical Layer; nothing is downloaded onto them).

## Updating the Network Data

Set-up of a simple program to constantly update the network variables is shown below:



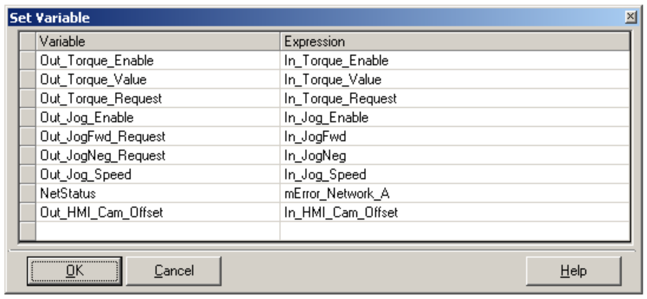


Properties - Mechatrolink Master

4 (If Event) LOOP

| | |
|---|---|
| ( ID ) | 4 |
| Label | LOOP |
| Event | 1 |

This program will continuously loop through the UPDATE block setting all the *Variables* in the *Set Variable* table equal to the value in the corresponding *Expression* column, shown below. All of these variables were created and defined as shown in the *Network Definition* section above. In this case, the slave servo will directly mimic the master servo.



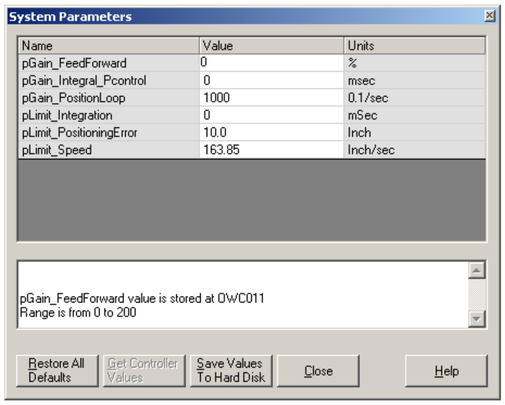
Set Variable

| Variable | Expression |
|---|---|
| Out_Torque_Enable | In_Torque_Enable |
| Out_Torque_Value | In_Torque_Value |
| Out_Torque_Request | In_Torque_Request |
| Out_Jog_Enable | In_Jog_Enable |
| Out_JogFwd_Request | In_JogFwd |
| Out_JogNeg_Request | In_JogNeg |
| Out_Jog_Speed | In_Jog_Speed |
| NetStatus | mError_Network_A |
| Out_HMI_Cam_Offset | In_HMI_Cam_Offset |

This program will need to be used in conjunction with the other programs that the user must create for controlling the motion of the system.

Before the slave controllers are modified, it is important to first check several properties. The user should look at the *System Parameters* and make the appropriate changes. These setting will vary greatly depending on the system. Regardless of the system:

1. *pGain_PositionLoop* > 0. (If *pGain_PositionLoop* = 0, then the servo will not turn.)
2. *pLimit_PositioningError* > 0. (If *pLimit_PositioningError* = 0, an A.9F error will most likely occur.)

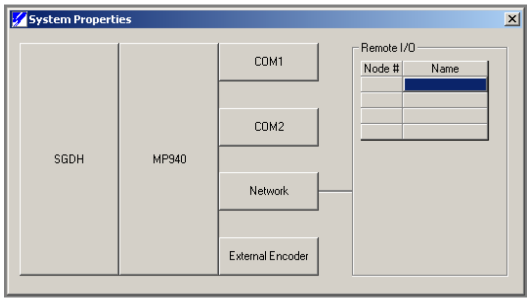


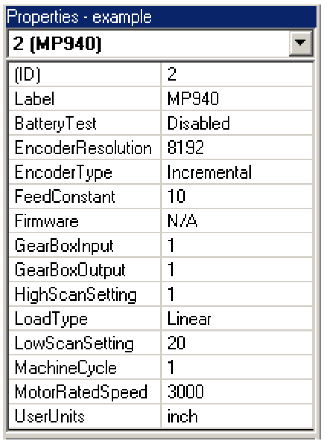
System Parameters

| Name | Value | Units |
|---|---|---|
| pGain_FeedForward | 0 | % |
| pGain_Integral_Pcontrol | 0 | msec |
| pGain_PositionLoop | 1000 | 0.1/sec |
| pLimit_Integration | 0 | mSec |
| pLimit_PositioningError | 10.0 | Inch |
| pLimit_Speed | 163.85 | Inch/sec |

pGain_FeedForward value is stored at OWC011
Range is from 0 to 200

It is also important that the *Properties* of the MP940 make sense given the mechanical setup for the application. Again, these will vary greatly depending on the system, but having values in these fields that are reasonably close to what they need to be is essential. Pay special attention to *FeedConstant*, *MachineCycle*, and *UserUnits*.




Properties - example
2 (MP940)

| | |
|---|---|
| (ID) | 2 |
| Label | MP940 |
| BatteryTest | Disabled |
| EncoderResolution | 8192 |
| EncoderType | Incremental |
| FeedConstant | 10 |
| Firmware | N/A |
| GearBoxInput | 1 |
| GearBoxOutput | 1 |
| HighScanSetting | 1 |
| LoadType | Linear |
| LowScanSetting | 20 |
| MachineCycle | 1 |
| MotorRatedSpeed | 3000 |
| UserUnits | inch |

<u>Gearbox Input:</u> If a mechanical gear is used, enter the value which corresponds to the number of times the input shaft rotates for the number of times the output shaft rotates (integer value). If a 10:1 gearbox is used, enter 10.

<u>Gearbox Output:</u> If a mechanical gear is used, enter the value which corresponds to the number of times the input shaft rotates for the number of times the output shaft rotates (integer value). If a 10:1 gearbox is used, enter 1.

<u>Feed Constant:</u> The number in user units that the load travels for each revolution of the FINAL output shaft of the mechanical system. Example: Ball screw of 6mm pitch, the feed constant setting would be 6. Example: Belt and pully with last pully of 10" diameter, the feed constant would be 10*pi.

<u>Machine Cycle:</u> This is the maximum position value of the internal encoder. When this value is reached, the position wraps back to "0". Machine cycle is only applicable if Movement Type is set to Rotary. If the application involves camming, the machine cycle may be the product length.

<u>User Units:</u> Select the user units to be displayed throughout the project. User units may also be entered.

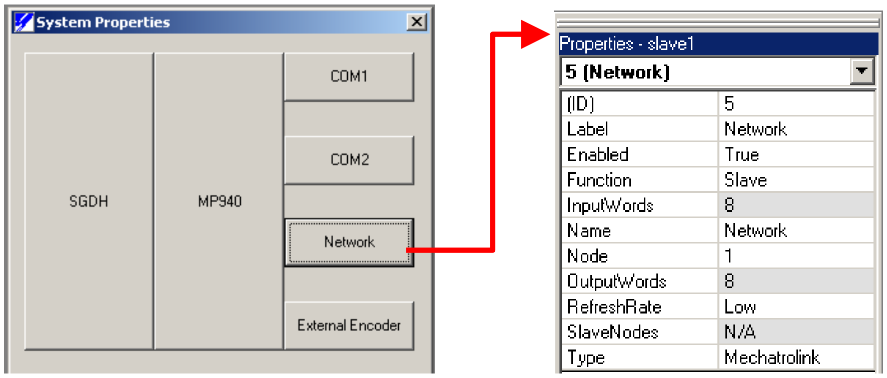
<u>Load Type:</u> rotary implies an axis rotating with a cycle period. The position data is automatically modularized to fit within one cycle period or machine cycle.

<u>Motor Rated Speed:</u> Rated speed of motor in RPM. Example: SGMGH=1500 rpm

## Motion Works+ Slave Program

Create a new project for each of the MP940 nodes in the network. The steps taken to properly configure this slave controller are quite similar to those taken to configure the master controller.

### *Configuring the Slave Node:*



Properties - slave1

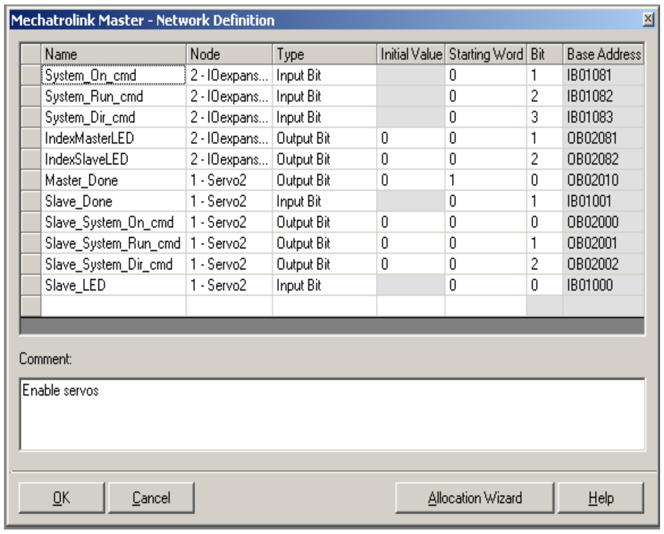
5 (Network)

| | |
|---|---|
| (ID) | 5 |
| Label | Network |
| Enabled | True |
| Function | Slave |
| InputWords | 8 |
| Name | Network |
| Node | 1 |
| OutputWords | 8 |
| RefreshRate | Low |
| SlaveNodes | N/A |
| Type | Mechatrolink |

1) *Function* should be set to *Slave*.
2) *Node* should correspond to the *Node#* as set up in the master. This tells the slave what node it is, so that when the master is referring to it, the slave will know.
3) Choose *Type* (Mechatrolink or DeviceNet)

Next, the network variables need to be defined. Remember above, when it was recommended to print a copy of the variables defined in the master? This is where that printout will come in handy. In the master, ALL network variables had to be defined. However, in the slave, only the variables that are passed between the master and the particular slave need to be defined. Slave2 is not interested in what is being communicated between Slave3 and the master. Remember, each slave can only communicate with the master, not with any of the other slaves. If Slave2 wants to pass along a bit to Slave3, it must be first sent from Slave2 to the master, then passed from the master to Slave3. Only the variables defined in the master table for that particular slave, will be passed to that slave.

If the network variables are compared for 2 sample MP940's (shown below) linked on a network, it can be seen that the inputs on the master correspond to the outputs on the slave, and the outputs on the master correspond to the inputs on the slave.

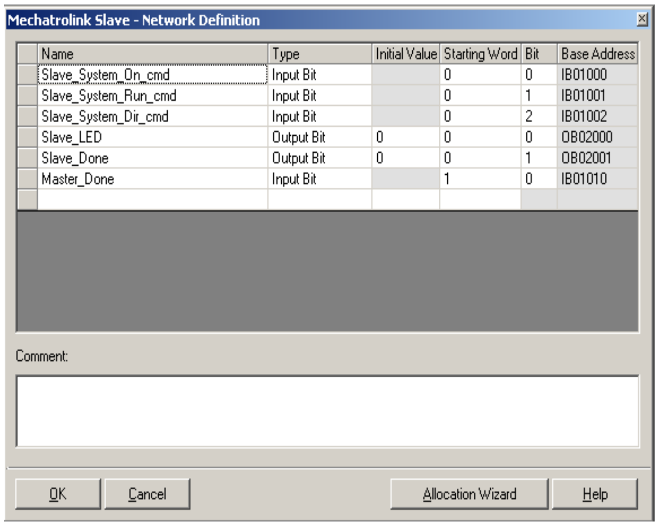

Mechatrolink Master - Network Definition

| Name | Node | Type | Initial Value | Starting Word | Bit | Base Address |
|---|---|---|---|---|---|---|
| System_On_cmd | 2 - IOexpans... | Input Bit | | 0 | 1 | IB01081 |
| System_Run_cmd | 2 - IOexpans... | Input Bit | | 0 | 2 | IB01082 |
| System_Dir_cmd | 2 - IOexpans... | Input Bit | | 0 | 3 | IB01083 |
| IndexMasterLED | 2 - IOexpans... | Output Bit | 0 | 0 | 1 | OB02081 |
| IndexSlaveLED | 2 - IOexpans... | Output Bit | 0 | 0 | 2 | OB02082 |
| Master_Done | 1 - Servo2 | Output Bit | 0 | 1 | 0 | OB02010 |
| Slave_Done | 1 - Servo2 | Input Bit | | 0 | 1 | IB01001 |
| Slave_System_On_cmd | 1 - Servo2 | Output Bit | 0 | 0 | 0 | OB02000 |
| Slave_System_Run_cmd | 1 - Servo2 | Output Bit | 0 | 0 | 1 | OB02001 |
| Slave_System_Dir_cmd | 1 - Servo2 | Output Bit | 0 | 0 | 2 | OB02002 |
| Slave_LED | 1 - Servo2 | Input Bit | | 0 | 0 | IB01000 |

Comment:

Enable servos

Network Variables: Master


Mechatrolink Slave - Network Definition

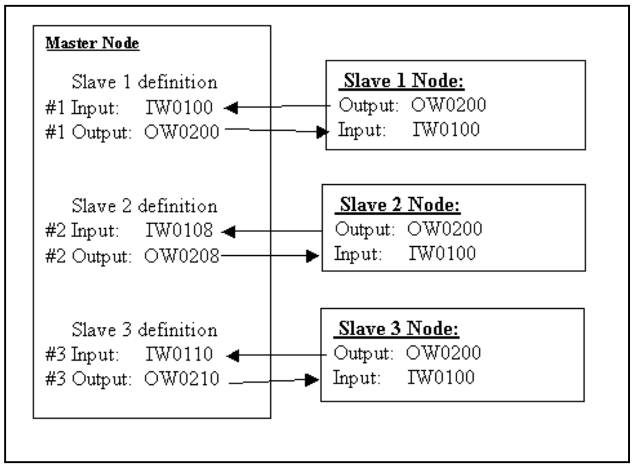
| Name | Type | Initial Value | Starting Word | Bit | Base Address |
|---|---|---|---|---|---|
| Slave_System_On_cmd | Input Bit | | 0 | 0 | IB01000 |
| Slave_System_Run_cmd | Input Bit | | 0 | 1 | IB01001 |
| Slave_System_Dir_cmd | Input Bit | | 0 | 2 | IB01002 |
| Slave_LED | Output Bit | 0 | 0 | 0 | OB02000 |
| Slave_Done | Output Bit | 0 | 0 | 1 | OB02001 |
| Master_Done | Input Bit | | 1 | 0 | IB01010 |

Comment:

Network Variables: Slave 1

In the above sample, if the master wants to turn on the slave 1 system operation, the master needs to set its "Slave_System_On_cmd" output bit (defined automatically at OB02000). The slave will receive the update at the next network scan in its "Slave_System_On_cmd" input bit (defined automatically at IB01000).

To visualize how the master and each individual slave pass data, imagine the master handing out stacks of paper. The master knows that the top page of each stack is the first word for each slave. The slave receiving the stack also knows that the top page of the stack corresponds to first word. The diagram below illustrates this point with one MP940 master and three MP940 slaves. It shows that the first stack consists of input and output words, intended for the first slave. Notice base register addresses in the master's slave definitions. MP940s are capable of passing 8-16 bit input and 8-16 bit output words. The register values reflect that in hex. Since the master coordinates each slaves data, each slave setup can be identical – note the register addressing.



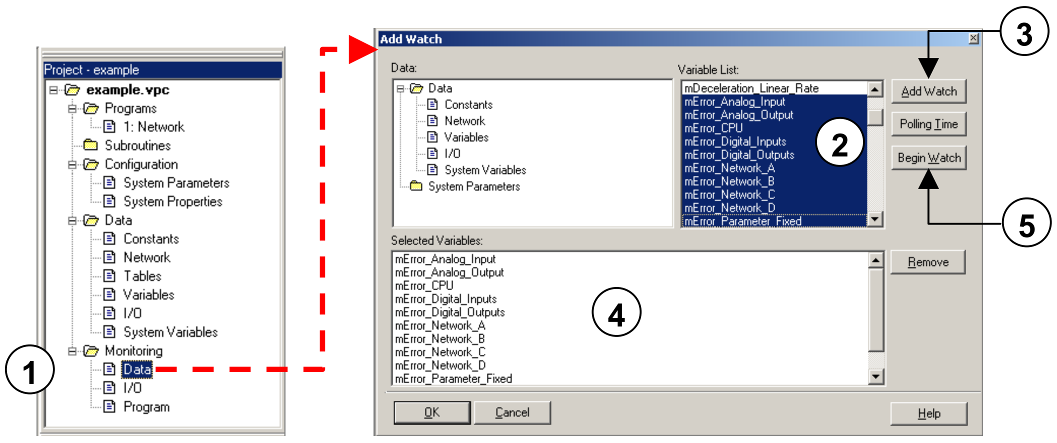
Note that it is not necessary for the *Name* of a variable in the slave to be the same as the *Name* of the same variable (at the same address) in the master. In our example above, we used "Slave_System_On_cmd" in both the master and the slave. Doing this is recommended, as it makes it much easier to determine what is happening in the program, but it is not essential for the program to run properly since it is only a name tag.

To illustrate this point, the *Name* of the *Base Address* IB01000 could have been changed in the master from "Slave_LED" to "LED_Slave" and the slave would still completely understand the information being sent. However, if the *Name* was kept as "Slave_LED" and the address changed to from IB01000 to IB01006, then the slave information would be sent to an empty address and the system would not function properly.

## Troubleshooting

Continue with the above steps for all MP940 nodes. Until each node knows its identity (slave and node #), the master will continue to show an A.9F error. Only when all of the nodes are properly programmed and configured will the A.9F error disappear.

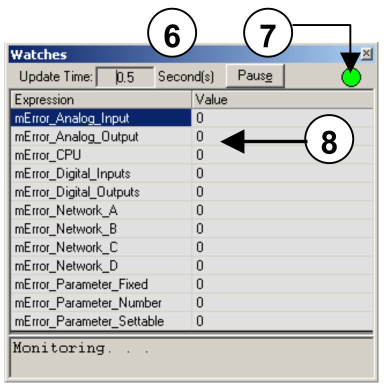
1. Double-click on *Monitoring* > *Data* in the *Project Window*



Variable List:
mDeceleration_Linear_Rate
mError_Analog_Input
mError_Analog_Output
mError_CPU
mError_Digital_Inputs
mError_Digital_Outputs
mError_Network_A
mError_Network_B
mError_Network_C
mError_Network_D
mError_Parameter_Fixed

Selected Variables:
mError_Analog_Input
mError_Analog_Output
mError_CPU
mError_Digital_Inputs
mError_Digital_Outputs
mError_Network_A
mError_Network_B
mError_Network_C
mError_Network_D
mError_Parameter_Fixed

2. Highlight the variables to monitor in the *Variable List*
3. Click on the *Add Watch* button
4. The highlighted variables will appear in the *Selected Variables* window
5. To start monitoring, click the *Begin Watch* button
6. The monitoring window labeled *Watches*, should appear
7. The green shape in the corner should change between a circle and a square. This means the window is reading and updating.
8. The main information determining what type of error has occurred will appear in the *mError_CPU* field. If *mError_CPU*=:
   a. 256 – Math error
   b. 512 – LIO (local input/output) error
   c. 4096 – SVA error
   d. 8192 – CNTR error
   e. 32768 – Network error



| Expression | Value |
|---|---|
| mError_Analog_Input | 0 |
| mError_Analog_Output | 0 |
| mError_CPU | 0 |
| mError_Digital_Inputs | 0 |
| mError_Digital_Outputs | 0 |
| mError_Network_A | 0 |
| mError_Network_B | 0 |
| mError_Network_C | 0 |
| mError_Network_D | 0 |
| mError_Parameter_Fixed | 0 |
| mError_Parameter_Number | 0 |
| mError_Parameter_Settable | 0 |

```
Monitoring. . .
```

The other variables in the *Watch List* will offer more insight as to the nature of the problem, but that is beyond the scope of this paper. Please consult Yaskawa at 1-800-YASKAWA.