MicroComm DXL

DXL Wonderware Host Example

April 2010

Printed in Canada

Table of Contents

1 Introduction

The MicroComm DXI and DXL systems are designed to allow external control and monitoring of intercom functions using a number of “host port” protocols.

Among the protocols the DXI and DXL can work with are protocols in which the DXI or DXL system can emulate a PLC, allowing a standard industrial control package such as Wonderware InTouch to communicate with the intercom system using standard protocols.

Two PLC protocols that the DXI and DXL systems can use are Modbus Ethernet (Modbus TCP/IP) and Omron Ethernet (Omron FINS).

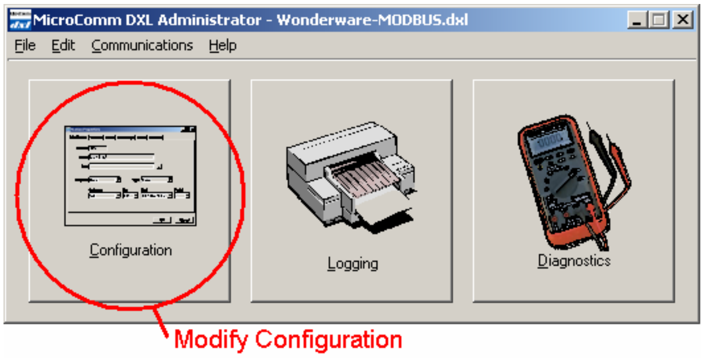
This document will outline how to make up a simple intercom system interface using Wonderware InTouch along with either of the Wonderware supplied Modbus MBTCP DA Server or Omron FINS Server.

Examples in this document assume you have some knowledge of Wonderware InTouch and its scripting language.

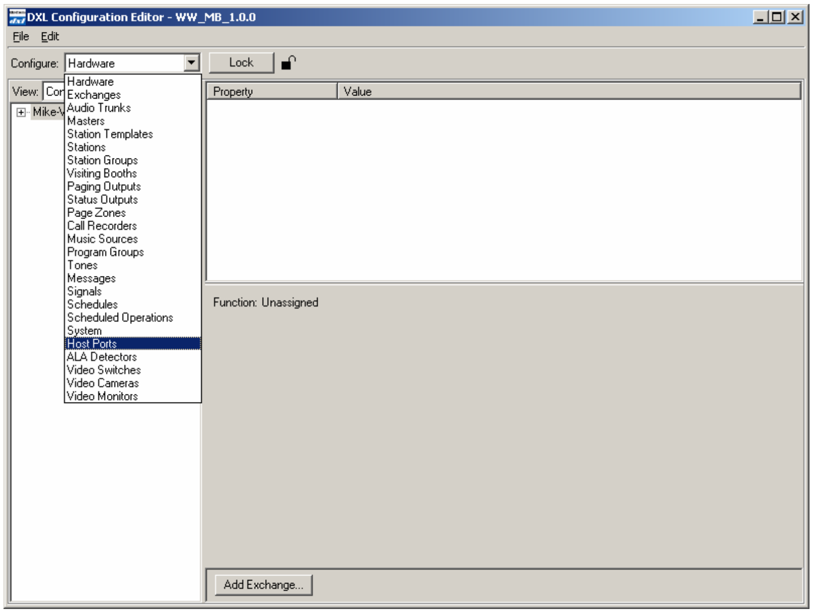
2 DXL Host Protocol Setup

DXL Administrator (provided with the intercom system) is the Windows program used to configure all aspects of the intercom system’s operation. Using this program, we will now configure the DXL to communicate with an external host (typically a touchscreen master). The overall configuration of an intercom system is beyond the scope of this document, so the following example assumes that a configuration has already been created and all that needs to be configured is the host port. For information on general intercom configuration, consult the documents “Microcomm DXL Configuration Example” and “Microcomm DXL Administration Software & Local User Interface”.

Start DXL Administrator and open the intercom systems configuration. The following example has a configuration with one exchange (consisting of one DCC), one TMM master, and six single button intercom stations.



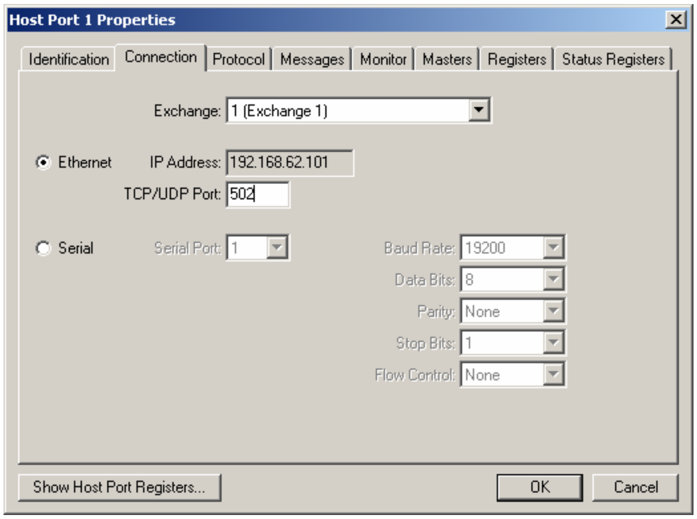
Select “Host Ports” from the drop down menu.



Configure: Hardware

Function: Unassigned

Click the “New” button to add a new host port. On the “Connection” tab, select the exchange that the host will connect to. This example uses an Ethernet connection rather than serial, so select Ethernet. Selecting an exchange will automatically fill in the IP Address field with the IP address assigned to the selected exchange. The IP address of the exchange is set in the exchange properties window and cannot be changed here. The port number to be used for host-DXL communication is entered in to the “Port” field. Modbus typically uses port 502 and Omron FINS typically uses port 9600, although they can both be configured differently. In this example, the typical port numbers are used.



Exchange: 1 (Exchange 1)

● Ethernet IP Address: 192.168.62.101

TCP/UDP Port: 502

○ Serial Serial Port: 1

Baud Rate: 19200

Data Bits: 8

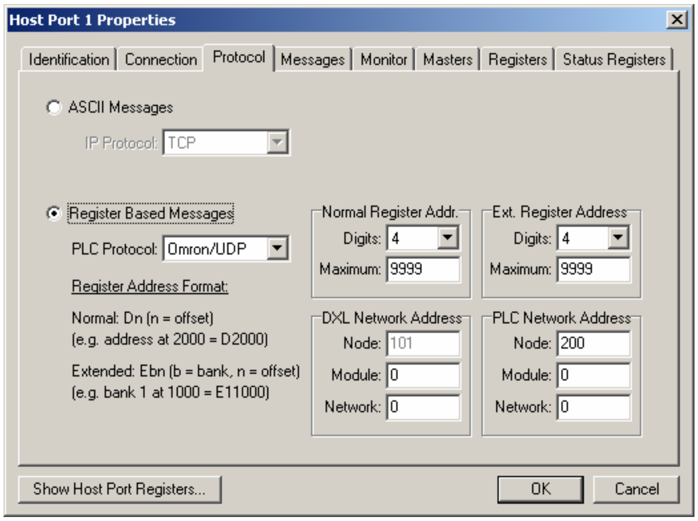
Parity: None

Stop Bits: 1

Flow Control: None

On the “Protocol” tab, select “Register Based Messages” and set the “PLC Protocol” to either “Modbus/TCP” or “Omron/UDP”. Now set how many digits the PLC uses to specify register addresses. Note that DXL specifies digits differently than Wonderware because the DXL does not count the first (unchanging) digit. In this example, “Digits” is set to 4 in DXL administrator. This results in addresses of the form “4xxxx” (for Modbus) or “Dxxxx” (for Omron).

If Modbus is being used, the “PLC Network Address Node” can be left as zero. If Omron FINS is being used, we also need to specify “Node”, “Network”,



○ ASCII Messages

IP Protocol: TCP

● Register Based Messages

PLC Protocol: Omron/UDP

Register Address Format:

Normal: Dn (n = offset)
(e.g. address at 2000 = D2000)

Extended: Ebn (b = bank, n = offset)
(e.g. bank 1 at 1000 = E11000)

Normal Register Addr.
Digits: 4
Maximum: 9999

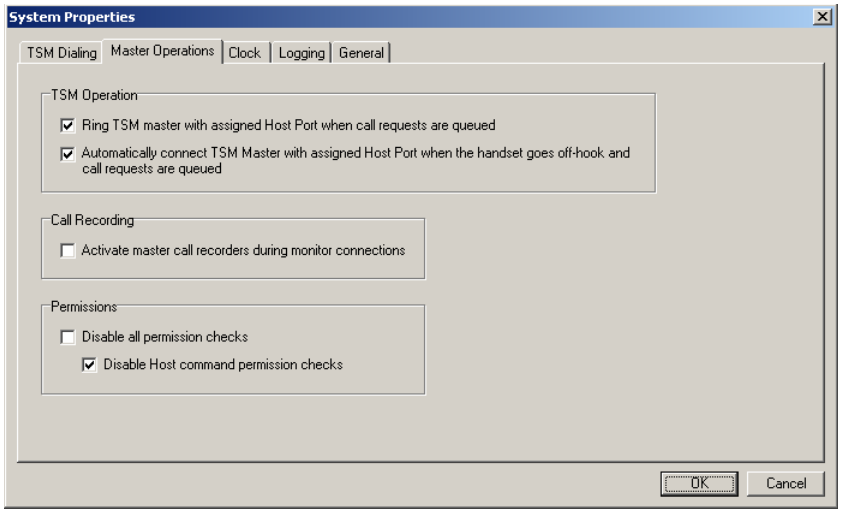
Ext. Register Address
Digits: 4
Maximum: 9999

DXL Network Address
Node: 101
Module: 0
Network: 0

PLC Network Address
Node: 200
Module: 0
Network: 0

and “Module”. The “PLC Network Address Node” should be set to the last byte of the host computer’s IP address (for example, if the PLC’s IP address is 192.168.0.200, then PLC node is 200). “Network” and “Module” are set to zero in this example.

The DXL system has a mechanism to limit the permissions for masters to call stations, masters, page zones, or other functions. Typically this is used for stand-alone masters to limit what they can call, but this also normally applies to host-controlled masters as well. You can set the DXL system to always permit a command from a host to operate (bypassing normal permissions). To do this, select “System” from the main drop down menu, then go to the “Master Operations” tab. Un-check the “Disable All Permission Checks” and check the “Disable Host Permission Checks” check boxes.



System Properties

TSM Operation

☑ Ring TSM master with assigned Host Port when call requests are queued

☑ Automatically connect TSM Master with assigned Host Port when the handset goes off-hook and call requests are queued

Call Recording

☐ Activate master call recorders during monitor connections

Permissions

☐ Disable all permission checks

☑ Disable Host command permission checks

Later, we will need to define registers to pass information between the DXL and it’s host but first we will set up the host computer side of the interface.

3 Wonderware I/O Driver Setup

Before programming the graphic user interface of the touchscreen or HMI itself, first the link between the touchscreen/HMI host and the DXI or DXL system must be created.

This example covers two different communication protocol servers: Omron FINS and Modbus TCP/IP. Both are set up in a similar manner. First the I/O server is configured and second, “topics” are set up. Topics are essentially communication channels between Wonderware and the I/O server. Only one communication protocol should be used at a time, so go through the section pertaining to your protocol and then skip to Section 3: Simple DXL Example Using Status Registers.

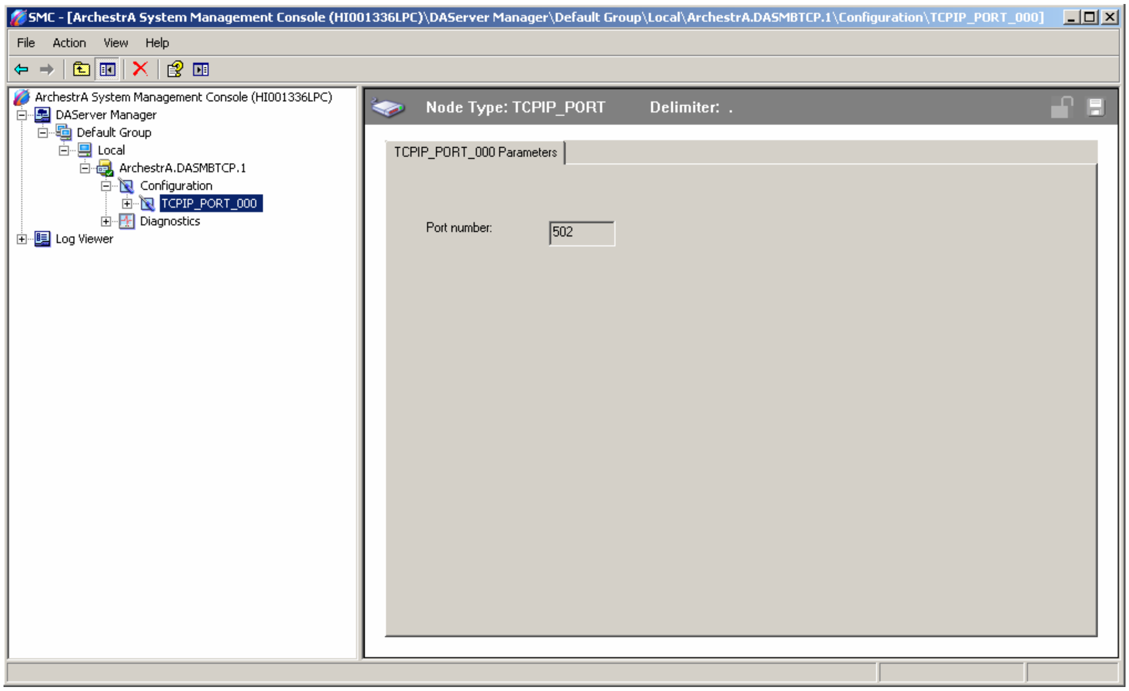
These sections are applicable to both DXI and DXL setup. Anything that mentions DXL below is equally applicable to setup for the DXI system as well.

3.1.1 Modbus Communication Setup: MBTCP DA Server

First, you will need to install the Wonderware MBTCP DA Server using the Wonderware Device Integration setup disc. The Modbus TCP/IP driver should be found in the “Schneider Automation” section.

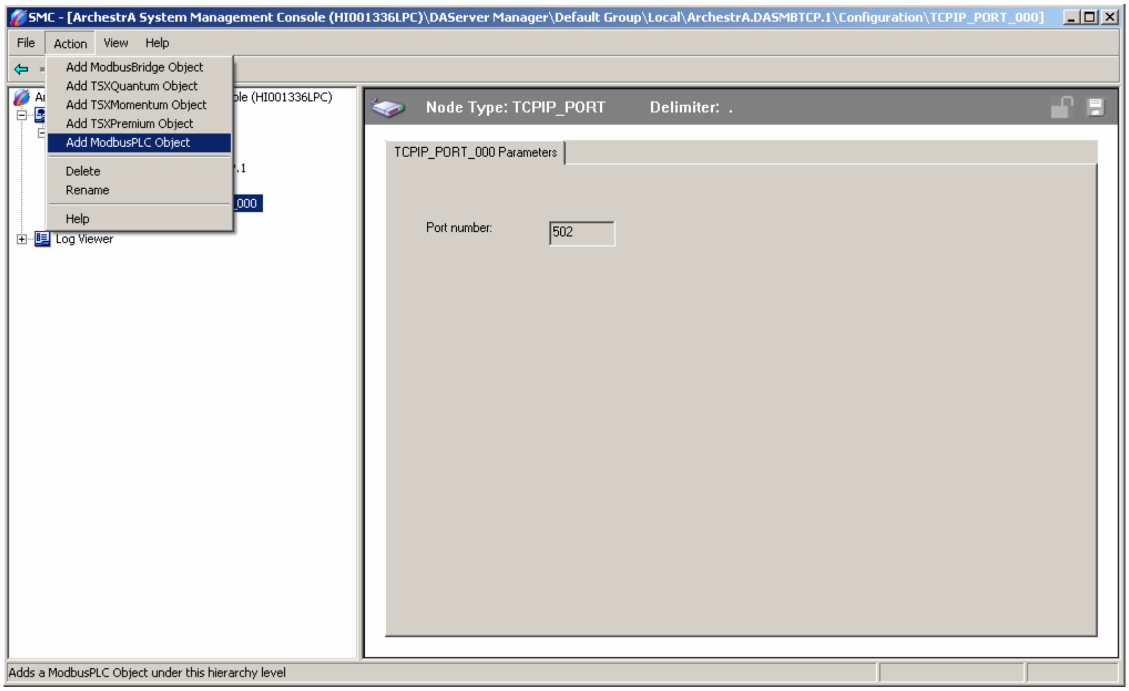
Set up the MBTCP DA Server by opening the Wonderware ArchestrA System Management Console.

Open the tree view up by expanding the
”DAServer Manager/Default Group/Local/ArchestrA.DASMBTCP.1/Configuration” entry.


SMC - [ArchestrA System Management Console (HI001336LPC)\DAServer Manager\Default Group\Local\ArchestrA.DASMBTCP.1\Configuration\TCPIP_PORT_000]
File
Action
View
Help
ArchestrA System Management Console (HI001336LPC)
DAServer Manager
Default Group
Local
ArchestrA.DASMBTCP.1
Configuration
TCPIP_PORT_000
Diagnostics
Log Viewer
Node Type: TCPIP_PORT
Delimiter: .
TCPIP_PORT_000 Parameters

Port number: 502

Select the “TCPIP_Port_000” entry then select the “Action/Add ModbusPLC Object” menu item.



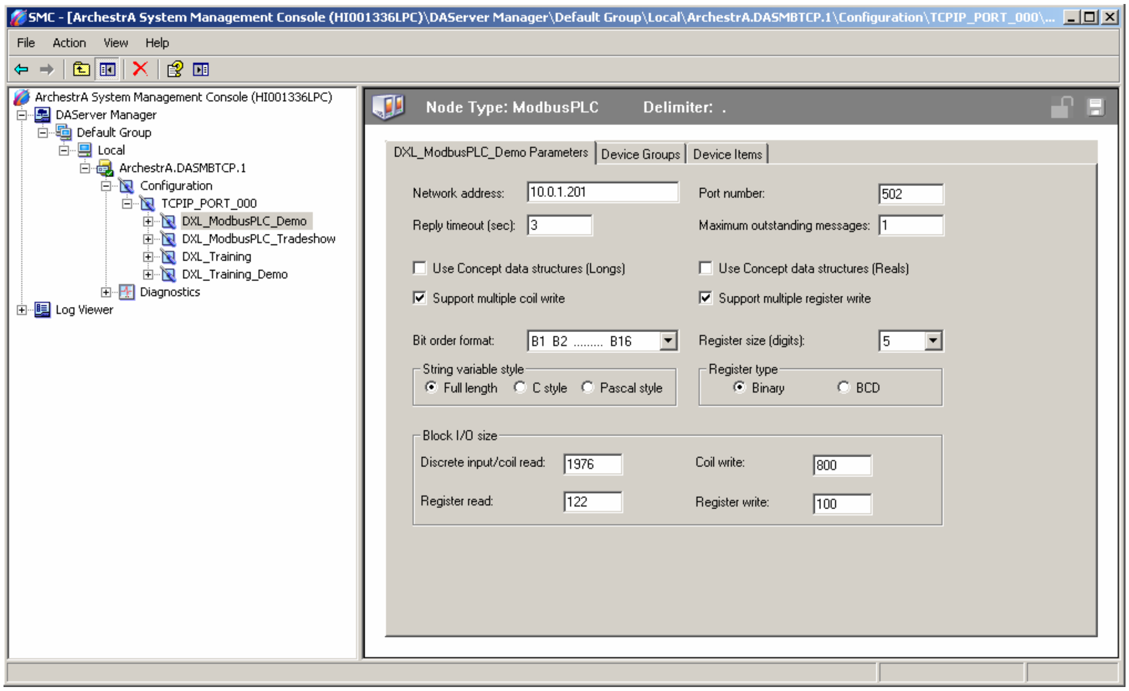
Port number: 502

Adds a ModbusPLC Object under this hierarchy level

Name your new object with a descriptive name. Then change the Maximum outstanding messages, Use Concept data structures (Longs), and Use Concept data structures (Reals) parameters as below.

In the “Network address” field you will need to set the network address to the address that the DXL system is configured as, which is not necessarily the address shown.

Also you may change the Register size (digits) value. This determines the starting number of the "register space". If this is set to 5, the DXI or DXL registers are 40000-49999, while if this is set to 6, the DXL registers are 400000-499999. This example uses registers in the range of 40000-49999 so this is set to 5 here.


SMC - [ArchestrA System Management Console (HI001336LPC)\DAServer Manager\Default Group\Local\ArchestrA.DASMBTCP.1\Configuration\TCPIP_PORT_000\...
File
Action
View
Help
ArchestrA System Management Console (HI001336LPC)
DAServer Manager
Default Group
Local
ArchestrA.DASMBTCP.1
Configuration
TCPIP_PORT_000
DXL_ModbusPLC_Demo
DXL_ModbusPLC_Tradeshow
DXL_Training
DXL_Training_Demo
Diagnostics
Log Viewer
DXL_ModbusPLC_Demo Parameters
Device Groups
Device Items

Node Type: ModbusPLC Delimiter: .

Network address: 10.0.1.201

Port number: 502

Reply timeout (sec): 3

Maximum outstanding messages: 1

☐ Use Concept data structures (Longs)

☐ Use Concept data structures (Reals)

☑ Support multiple coil write

☑ Support multiple register write

Bit order format: B1 B2 .... B16

Register size (digits): 5

String variable style: ● Full length ○ C style ○ Pascal style

Register type: ● Binary ○ BCD

Block I/O size

Discrete input/coil read: 1976

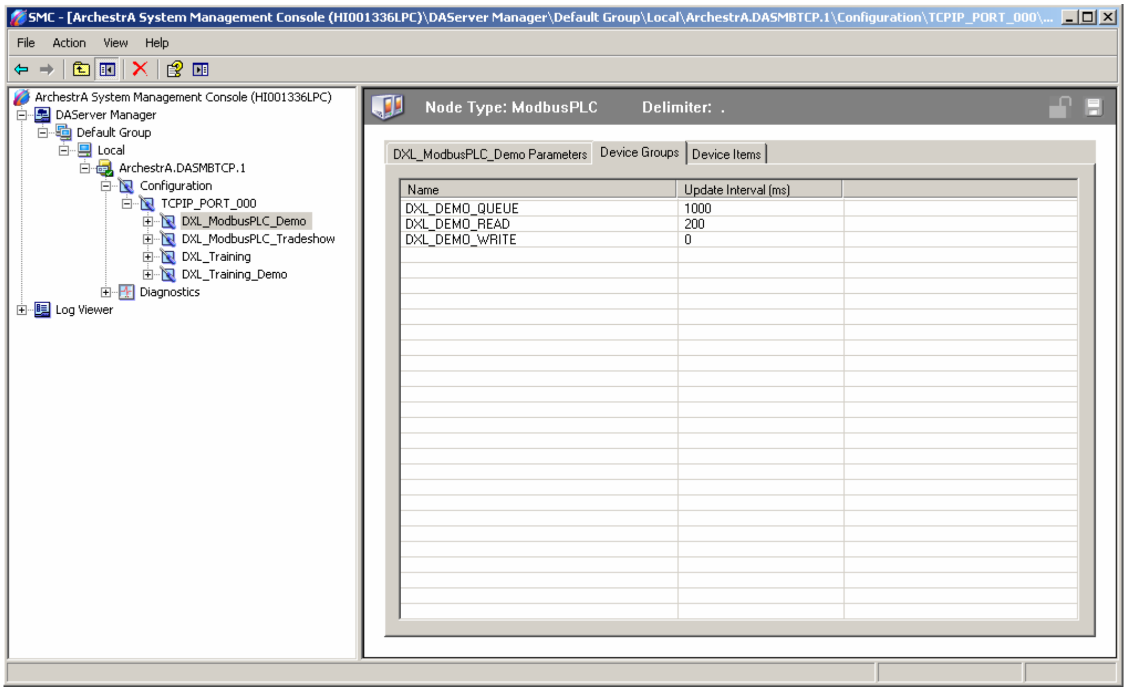
Coil write: 800

Register read: 122

Register write: 100

Now that the basic parameters of the DXL "PLC" are set up, the topics can be defined.

Select the "Device Groups" tab, then right click any of the blank lines and select "Add". Change the name of the newly added topic to a descriptive identifier of the DXL interface, and change the Update Interval to how often the host will poll the DXL system for new status updates. Add the below topics, then click the "save" icon (the floppy disk icon) on the top right corner.


SMC - [ArchestrA System Management Console (HI001336LPC)\DAServer Manager\Default Group\Local\ArchestrA.DASMBTCP.1\Configuration\TCPIP_PORT_000\...
File
Action
View
Help
ArchestrA System Management Console (HI001336LPC)
DAServer Manager
Default Group
Local
ArchestrA.DASMBTCP.1
Configuration
TCPIP_PORT_000
DXL_ModbusPLC_Demo
DXL_ModbusPLC_Tradeshow
DXL_Training
DXL_Training_Demo
Diagnostics
Log Viewer
DXL_ModbusPLC_Demo Parameters
Device Groups
Device Items

Node Type: ModbusPLC Delimiter: .

Name	Update Interval (ms)
DXL_DEMO_QUEUE	1000
DXL_DEMO_READ	200
DXL_DEMO_WRITE	0

Each PLC can be assigned multiple topics that can determine how often various registers within the device are updated. In this example, we will create three “topic names” for interfacing to the DXL. While you do not need to make more than one topic name, creating multiple topic names (particularly for read and write) can reduce the amount of network traffic required in the system.

The “DXL_DEMO_QUEUE” topic will be the topic name for the DXL queue registers. These are the top calls on the master’s queue. The update interval of 1000 ms shown here will update these every 1 second. These registers are optional, and you only need them if you want to have an intercom call queue for display of the top calls on the system.

The “DXL_DEMO_READ” topic will be the topic name for the DXL status registers. These are updates from the DXL such as the status of a station’s call request button. You will want to set the update rate here to an appropriate value. In this example the update interval of 200 ms will update the call request and other status information 5 times per second.

The “DXL_DEMO_WRITE” topic will be the topic name for the DXL command registers. The command registers are used to send commands (such as connection commands) from the host to the DXL system. Since these are one way only (from the host to DXL) these do not need to be periodically read back, so the update interval can be set to 0 for this topic.

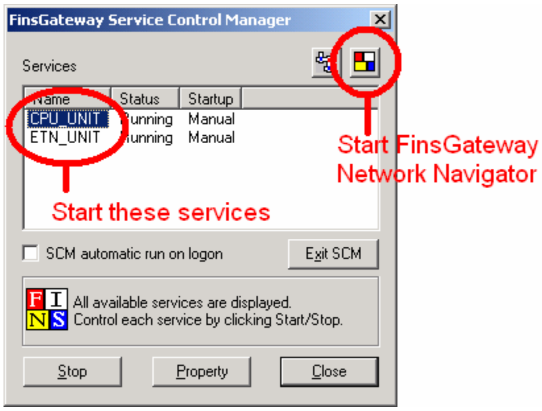
Remember the topic names you have used, as these names will be used within the Touchscreen/HMI software later.

This concludes the setup of the MBTCP DA Server. You can now exit the ArchestrA System Management Console.

3.1.2 OMRON Communication Setup: FINSGTWY Server

First, you will need to install Wonderware Factory Suite 2000, OMRON Fins Gateway Server, and OMRON Fins Gateway Ethernet Server using the Wonderware setup discs. These may be found on the “Device Integration” disc. Wonderware Factory Suite 2000 is found in the Wonderware folder and both OMRON servers are found in the OMRON folder. Make sure you have the latest version of the Ethernet server “ETN_UNIT” since older versions will not start on machines using DHCP to obtain an IP address. Check http://www.plcsoft.ne.jp/soft/ for updates.

After the software is installed, open the Fins Gateway Service Manager. A “FinsGateway Service Manager” icon will appear in your system tray- right click it and select “Setting”. Start the services “CPU_UNIT” and “ETN_UNIT” then click on the “Start FinsGateway Network Navigator” button.

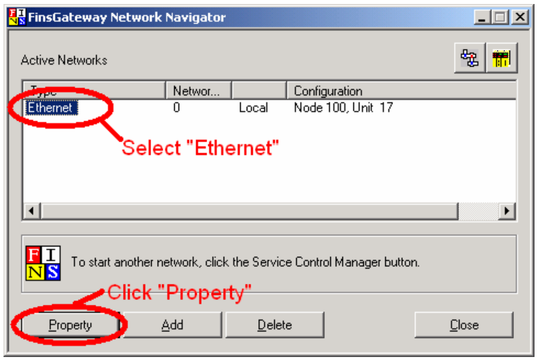

FinsGateway Service Control Manager
Services
Start FinsGateway Network Navigator
Start these services
Exit SCM
Stop
Property
Close

Name	Status	Startup
CPU_UNIT	unning	Manual
ETN_UNIT	unning	Manual

☐ SCM automatic run on logon

All available services are displayed.
Control each service by clicking Start/Stop.

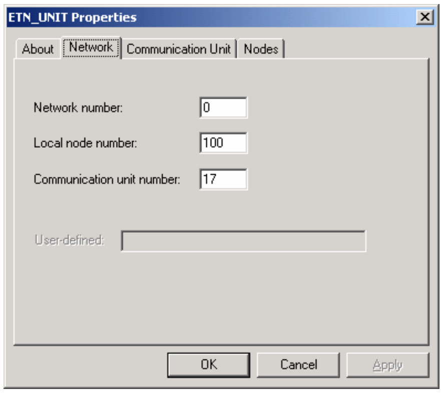
The FinsGateway Network navigator window will open up. This is where we will set up IP addresses for the host computer and the DXL. Select “Ethernet” and click on “Property”.



Type	Networ...		Configuration
Ethernet	0	Local	Node 100, Unit 17

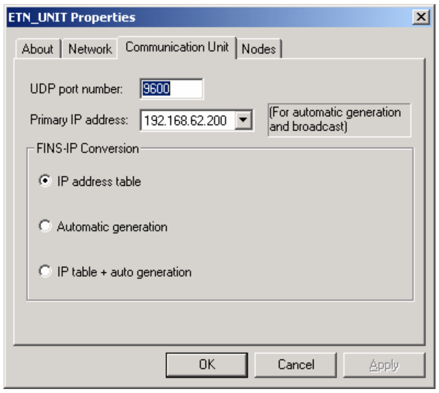
To start another network, click the Service Control Manager button.

Select the “Network” tab of the Ethernet properties window. This is where the OMRON network number, node number, and communication number of the host computer are entered.



Network number: 0
Local node number: 100
Communication unit number: 17
User-defined:

Select the “Communication Unit” tab. This is where the IP address and port number of the host computer are defined. Select the “IP Address Table” option in the “FINS-IP Conversion” section

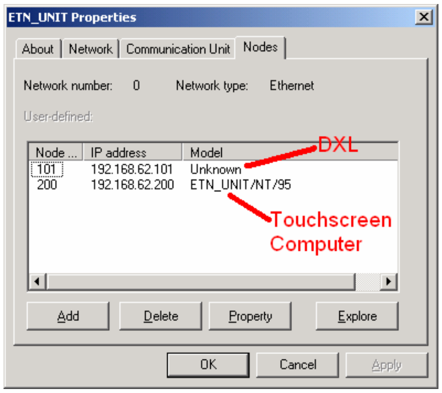


UDP port number: 9600
Primary IP address: 192.168.62.200
(For automatic generation and broadcast)

FINS-IP Conversion

● IP address table
○ Automatic generation
○ IP table + auto generation

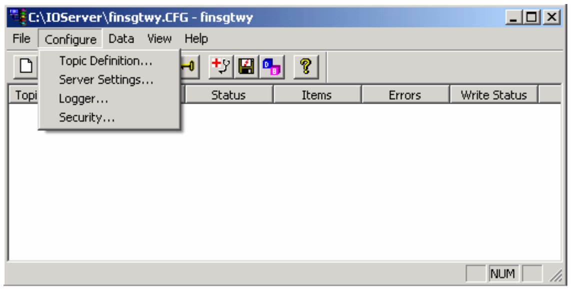
Select the “Nodes” tab. We need to define OMRON network nodes for the DXL and the host computer. Create one node with the IP address listed on the “Communication Unit” tab and the node number listed on the “Network” tab. The network navigator should fill in the “Model” column for this node as “ETN_UNIT/NT/95” automatically. Create another node with the IP address and node number of the DXL (these must match the IP address and node defined in the intercom system configuration created with DXL Administrator). Note that the node number should be the same as the last byte of the IP address. For example, if the IP address is 192.168.0.2 then the node number is 2.



Network number: 0 Network type: Ethernet
User-defined:

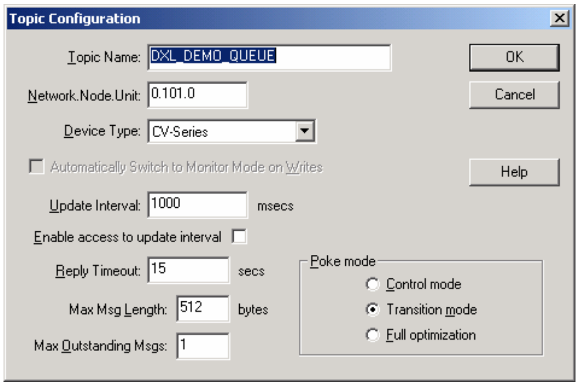
Node ...	IP address	Model
101	192.168.62.101	Unknown
200	192.168.62.200	ETN_UNIT/NT/95

Now we need to define the topics. Open the Omron Fins Gateway IO Server (installed with Wonderware Factory Suite 2000). Select “Topic Definition” from the “Configure” menu.



Each PLC can be assigned multiple topics that can determine how often various registers within the device are updated. In this example, we will create three topics for interfacing to the DXL. While you do not need to make more than one topic name, creating multiple topic names (particularly for read and write) can reduce the amount of network traffic required in the system. Each topic needs to be assigned an address to point to the proper device on the OMRON network (in this case, the DXL) and an update interval to tell it how often to check the topic for new data. All other settings should be left at their defaults (as shown below). The network address is entered into the “Network.Node.Unit” field. Network and unit number are both typically zero. Node number is the node number assigned to the DXL and should be identical to the last byte of the DXL’s IP address.

The “DXL_DEMO_QUEUE” topic (shown here) will be the topic name for the DXL queue registers. These are the top calls on the master’s queue. The queue does not need to be updated extremely quickly so an update interval of 1000ms will be sufficient. These registers are optional, and you only need them if you want to have an intercom call queue for display of the top calls on the system.



Topic Name: DXL_DEMO_QUEUE
Network.Node.Unit: 0.101.0
Device Type: CV-Series
☐ Automatically Switch to Monitor Mode on Writes
Update Interval: 1000 msecs
Enable access to update interval ☐
Reply Timeout: 15 secs
Max Msg Length: 512 bytes
Max Outstanding Msgs: 1
Poke mode
○ Control mode
● Transition mode
○ Full optimization

The “DXL_DEMO_READ” topic will be the topic name for the DXL status registers. These are updates from the DXL such as the status of a station’s call request button. The update rate set here will determine the delay between the call request button of a station being pressed and that call request arriving at the master. In this example we will use 200ms, which will update the call request and other status information 5 times per second.

The “DXL_DEMO_WRITE” topic will be the topic name for the DXL command registers. The command registers are used to send commands (such as connection commands) from the host to the DXL system. Since these are one way only (from the host to DXL) these do not need to be periodically read back, so the update interval can be set to 0 for this topic.

Remember the topic names you have used, as these names will be used within the touchscreen/HMI software later.

4 Creating a Wonderware Intouch Window

You can now create the Touchscreen/HMI interface window.

With the Wonderware InTouch Application Manager, create a new application, and then start WindowMaker. Right click on the “Windows” icon and select “New…”.

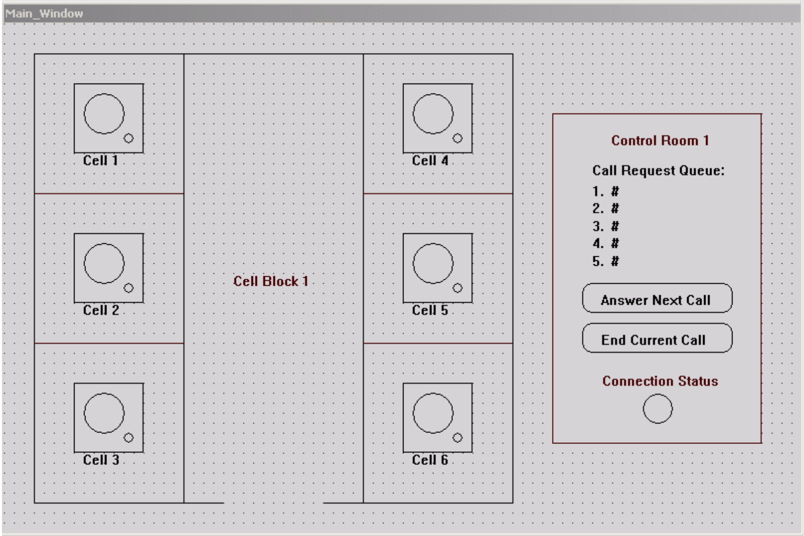
For this demo, we will create an intercom system consisting of one host master and six intercom stations.

You can create an intercom icon by grouping together circles and squares. Be sure to use the “Make Symbol” button rather than the “Make Cell” button to group the objects or you will not be able to add the proper actions to the intercom icons. It is easiest to create one icon, group it into a symbol, and then copy and paste it five more times.

We can also add a list to show the queue of call requests coming in to the host master (Master 1). Make a rectangle and add text to it using the text button T. Wonderware uses a number of symbols as placeholders for text, depending on how you would like it formatted. For our purposes, “#” is the text placeholder so enter “#” where you want the station number to show up. Our queue will display a maximum of 5 call requests.

You will also want to add buttons (in this case, rectangles with text in them) used to answer the next queued call and to end the current call, and an indicator of whether the host to intercom connection is working.

Your window should end up looking something like the one below:



5 Simple DXL Example Using Station Status Registers

This section shows how to configure a simple interface using the DXL “Station Status Registers”.

This portion applies **only** to DXL systems, as at this time only DXL systems have individual status registers for stations and masters.

5.1 Setting up Register Addresses on the DXL

When using a register based host protocol, such as Modbus or Omron, information to be passed between the DXL and host computer is placed in a series of registers which the DXL and computer can read from/write to. The location of these registers and whether the DXL or computer writes to them varies based on the communication mode used. In Peer to Peer mode, the host computer writes commands to command registers on the DXL and the DXL writes responses into status registers on the computer. In Polled mode, all registers are located on the DXL. The host computer writes commands to the DXL command registers and has to regularly read (poll) the DXL status registers to see what is going on. This example uses Polled mode.

There are four basic types of registers needed for this example.

The M1Command registers are used to control Master 1 (the host master). Commands are sent from the host computer to the DXL, with the first register (M1Command1) specifying the command and the other four registers giving parameters that tell the DXL how to execute the command. For example, to make the DXL place a call from Master 1 to Station 6, the host would write 7 (the code for “Ical”, the connect to intercom command) into M1Command1, 1 (the ID number of the master making the call) into M1Command2, 6 (the ID number of the station being called) into M1Command3, and 0 into the remaining M1Command registers (since they are not used for intercom calls).

The DXL writes to the M1Status registers to indicate the connection state of Master 1. M1Status1 contains a number indicating the type of device that Master 1 is connected to (0=nothing, 1=master, 2=station, 3=page zone), and M1Status2 contains the ID number of the device.

The M1Queue registers contain the type and ID number of Master 1’s call request queue. M1Queue1 contains the type of device at the top of Master 1’s queue, M1Queue2 contains the ID number of the device at the top of Master 1’s queue, M1Queue3 and M1Queue4 contain the type and ID of the second device in the queue, and so on. The queue registers are optional, but they are useful for displaying an ordered list of call requests on the Touchscreen/HMI. The size of the queue is configurable via DXL Administrator and the number of registers required would always be twice the number of devices that the queue can store. Note that regardless of the queue length set here, the DXL will still queue up call requests internally; the queue registers simply determine how much of that queue will be available for the host to read.

The DXL writes to the Station Status registers to indicate the connection state of the stations. Each bit of a station register indicates a different condition that the station is either in or not. For any single condition, 1 means yes and 0 means no. The bits in a station status register that concern us in this example are bit 0 (the least significant bit), which represents whether the station is in a call or not, and bit 2, which represents whether the station has a call request pending or not.

For a complete set of commands, parameters, and status bits please see the document “DXL Host Interface Specifications”, available at http://www.harding.ca.

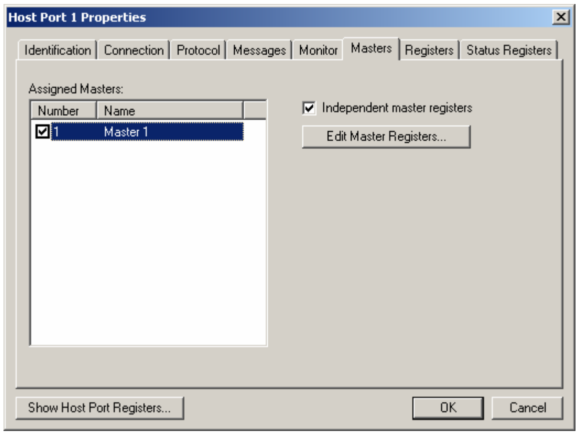
DXL Wonderware Example

For this example, we need to define the following registers:

Register/Tag Name	Modbus Register Address	Omron FINS Register Address
M1Command1	40001	D0001
M1Command2	40002	D0002
M1Command3	40003	D0003
M1Command4	40004	D0004
M1Command5	40005	D0005
M1Status1	40006	D0006
M1Status2	40007	D0007
M1Queue1	40011	D0011
M1Queue1	40012	D0012
M1Queue3	40013	D0013
M1Queue4	40014	D0014
M1Queue5	40015	D0015
M1Queue6	40016	D0016
M1Queue7	40017	D0017
M1Queue8	40018	D0018
M1Queue9	40019	D0019
M1Queue10	40020	D0020
S1Status	40021	D0021
S2Status	40022	D0022
S3Status	40023	D0023
S4Status	40024	D0024
S5Status	40025	D0025
S6Status	40026	D0026

The text and pictures below are for the Modbus protocol, but setting up registers for use with Omron is extremely similar: simply replace the “4” at the start of every Modbus address with a “D”.

Select "Host Ports" from the drop down menu in DXL Administrators configuration editor and double click the host port we created earlier to bring up its properties window. Select the "Masters" tab. To define command and status registers for our master, we need to have selected the master from the list on the left (i.e., there is a checkmark in the box next to Master 1's name) and have "Independent Master Registers" selected.

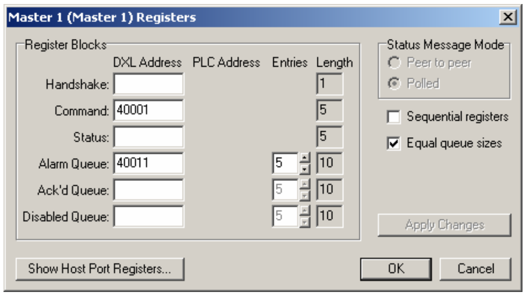

Host Port 1 Properties

Assigned Masters:

Number	Name
☑ 1	Master 1

☑ Independent master registers

Highlight "Master 1" and click on "Edit Master Registers". The fields under "DXL Address" are the starting addresses of the different sets of registers. "Length" indicates how many registers are contained in a set. Change the settings to match the following screenshot and click "OK". This will make Master 1's command registers 40001-40005 and Master 1's Queue registers 40011-40021. Note that register addresses are arbitrary, but duplicating the settings shown here will make it easier to follow the Wonderware tag setup in this document, since it uses the same addresses.


Master 1 (Master 1) Registers

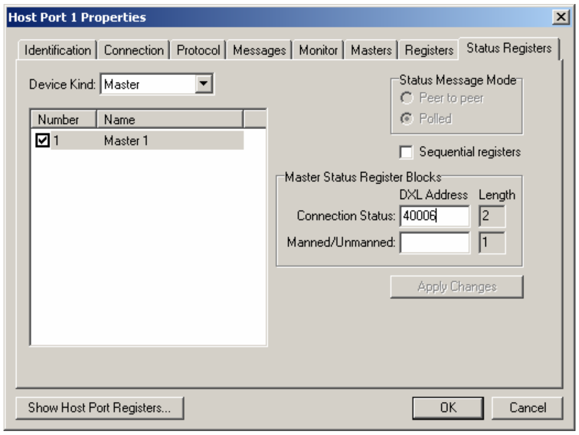
Register Blocks

	DXL Address	PLC Address	Entries	Length
Handshake:				1
Command:	40001			5
Status:				5
Alarm Queue:	40011		5	10
Ack'd Queue:			5	10
Disabled Queue:			5	10

Status Message Mode
○ Peer to peer
● Polled

☐ Sequential registers
☑ Equal queue sizes

Click on the "Status Registers" tab and select "Masters" from the drop down menu. Check the box next to Master 1 and, while Master 1 is highlighted, enter 40006 into the "Connection Status" field. This will set Master 1's connection status register addresses to 40006 and 40007.


Host Port 1 Properties

Device Kind: Master

Number	Name
☑ 1	Master 1

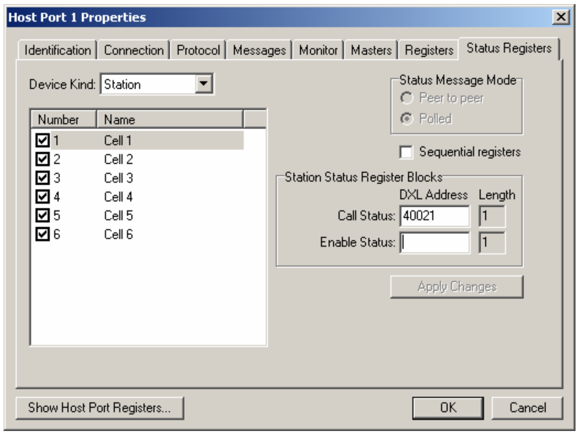
Status Message Mode
○ Peer to peer
● Polled

☐ Sequential registers

Master Status Register Blocks

	DXL Address	Length
Connection Status:	40006	2
Manned/Unmanned:		1

Select “Station” from the drop down menu. For each station, assign a register address for “Call Status”. Station call status blocks are only one register long. This example uses address 40021 for Station 1, 40022 for station 2, and so on until Station 6, which uses address 40026.



Device Kind: Station

Number	Name
☑ 1	Cell 1
☑ 2	Cell 2
☑ 3	Cell 3
☑ 4	Cell 4
☑ 5	Cell 5
☑ 6	Cell 6

Status Message Mode: ○ Peer to peer ● Polled

☐ Sequential registers

Station Status Register Blocks

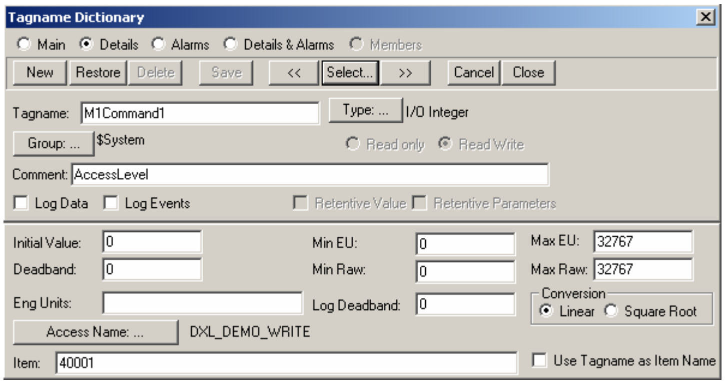
	DXL Address	Length
Call Status:	40021	1
Enable Status:		1

5.2 Setting up Wonderware Tags for registers in the DXL System:

Wonderware uses “tags” to keep track of data, both internal to the program and obtained from external sources such as the DXL. In this section we will define tags that correspond to the registers we defined on the DXL and a tag internal to Wonderware that monitors the status of the host connection.

5.2.1 Creating Tags for communication with the DXL

Double clicking the “Tagname Dictionary” icon will bring up the following screen:



○ Main ● Details ○ Alarms ○ Details & Alarms ○ Members

Tagname: M1Command1

Type: I/O Integer

Group: $System

○ Read only ● Read Write

Comment: AccessLevel

☐ Log Data ☐ Log Events ☐ Retentive Value ☐ Retentive Parameters

Initial Value: 0

Deadband: 0

Eng Units:

Min EU: 0

Min Raw: 0

Log Deadband: 0

Max EU: 32767

Max Raw: 32767

Conversion: ● Linear ○ Square Root

Access Name: DXL_DEMO_WRITE

Item: 40001

☐ Use Tagname as Item Name

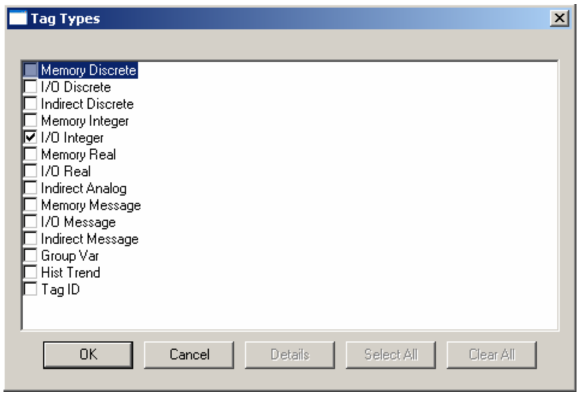
This screen is used to create and edit tag data. We need to define the following tags:

- M1Command1 to M1Command5 (used to send host master commands to the DXL)
- M1Status1 to M1Status2 (used to read the connection status of the host master)
- M1Queue1 to M1Queue10 (used to keep track of the call request queue for the host)
- S1Status to S6Status (used to read the status of stations 1 to 6)

The tags M1Command1 to M1Command5 should be set as “Read/Write”. All the other tags should be set as “Read Only”.

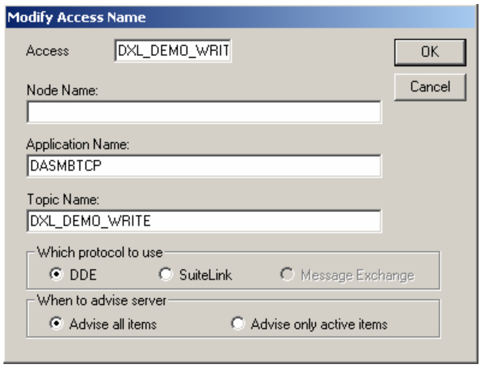
A tag’s “Item” is set to the address of the DXL register that will be associated with the tag. DXL register addresses were defined using DXL Administrator in section 5.1 of this document.

Clicking “Tag Type” brings up a prompt with all the possible tag types. Memory registers are internal to the Wonderware programming and I/O registers are obtained from Modbus TCP/IP or other protocols. All of the tags we are defining should be of the type “I/O Integer”.


Tag Types

☐ Memory Discrete
☐ I/O Discrete
☐ Indirect Discrete
☐ Memory Integer
☑ I/O Integer
☐ Memory Real
☐ I/O Real
☐ Indirect Analog
☐ Memory Message
☐ I/O Message
☐ Indirect Message
☐ Group Var
☐ Hist Trend
☐ Tag ID

Each tag needs to have an Access Name associated with it. These access names are associated with the topics that we defined in Section 2 of this document and will tell Wonderware where to get tag data from and how often to look for new tag data. Click on the “Access Name:…” button and then on the “Add…” button in the window that appears. You will now be looking at the “Modify Access Name” window.


Modify Access Name

Access: DXL_DEMO_WRIT
Node Name:
Application Name: DASMBTCP
Topic Name: DXL_DEMO_WRITE
Which protocol to use: ● DDE ○ SuiteLink ○ Message Exchange
When to advise server: ● Advise all items ○ Advise only active items

“Application Name” is the IO Communication Server. Modbus TCP/IP communication protocol uses the “DASMBTCP” server, and Omron FINS communication protocol uses the “FINSGTWY” server.

The “Topic Name” is the topic name assigned while configuring the IO Server. Create one access name for each of the topic names (DXL_DEMO_READ, DXL_DEMO_WRITE, and DXL_DEMO_QUEUE).

The “Protocol” is the method used to exchange data between the DA Server and Wonderware and can be set to DDE or SuiteLink.

The “When to advise server” item indicates whether to constantly poll these registers (Advise all items) or when to only poll them when they are being used by any running script or when any icon using them is visible (Advise only active items). For this example it does not matter what “When to advise server” is set to, since there will always be visible icons using the tags, but we recommend “Advise all items” to be safe.

Create the rest of the tags as listed in the table in section 5.1, making sure that the register addresses in DXL Administrator match the “Item” addresses in Wonderware, and assigning the following access names to the tags:

- Tags M1Queue1 to M1Queue10 will use the access name associated with the DXL_DEMO_QUEUE topic
- Tags M1Command1 to M1Command5 will use the access name associated with the DXL_DEMO_WRITE topic
- Tags M1Status1 to M1Status2 and S1Status to S6Status will use the access name associated with the DXL_DEMO_READ topic

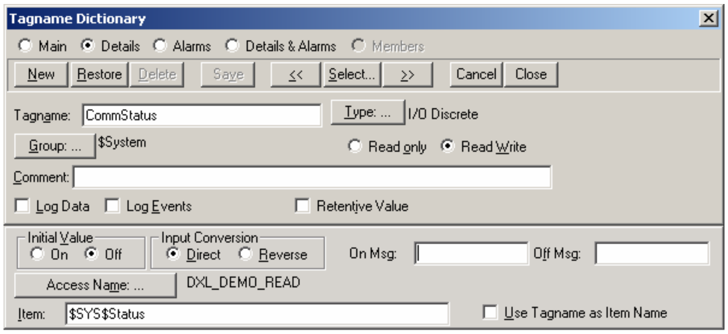
You should now have all the DXL I/O tags defined. Double check by clicking on “Select…”, which brings up a list of all tags defined in the current Wonderware project.

5.2.2 Wonderware Tag for determining communications status of DXL System

This tag is a DA server tag that indicates the connection status of the DXL system

(Discrete is the Wonderware term for a Boolean On/Off value)

This is set to 1 when the DA Server is connected to the PLC (DXL system) and the DXL is responding to polls, or 0 when the DA Server is not connected to the PLC (DXL system).



Tagname Dictionary

○ Main ● Details ○ Alarms ○ Details & Alarms ○ Members

Tagname: CommStatus

Type: I/O Discrete

Group: $System

○ Read only ● Read Write

Comment:

☐ Log Data ☐ Log Events ☐ Retentive Value

Initial Value: ○ On ● Off

Input Conversion: ● Direct ○ Reverse

On Msg: Off Msg:

Access Name: DXL_DEMO_READ

Item: SYSStatus

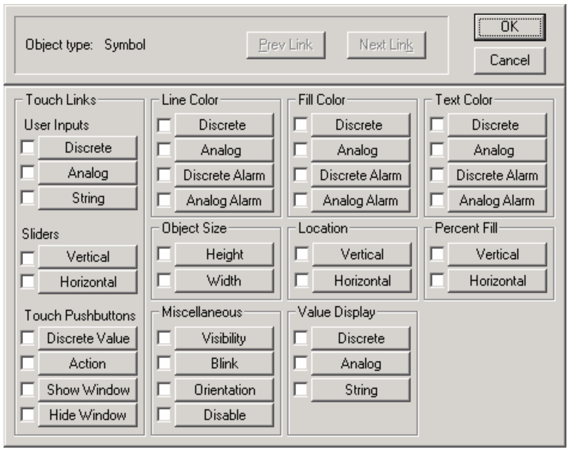
☐ Use Tagname as Item Name

5.3 Configuring Host Behaviour

Now we need to associate user actions with commands we want sent to the DXL, and tag values with what we want displayed on screen. For information on the commands and register structure used for communication between the DXL and a host, see “DXL Host Interface Specifications”, which is available for download at the Harding Instruments website.

5.3.1 Configuring the Communication Status Indicator

Double click on the communication status indicator to bring up the properties window for the communication status indicator:

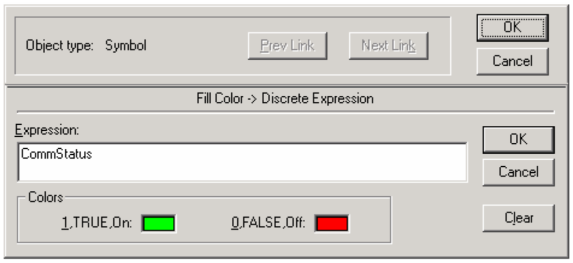


Object type: Symbol

We want the indicator to be green when communication between the host and DXL is working, and red when there is a problem. Click on the “Fill Color: Discrete” button to get to the following window.

Set “Expression” to the name name of our communication status tag (CommStatus).

```
CommStatus
```



Object type: Symbol

Fill Color -> Discrete Expression

Expression: CommStatus

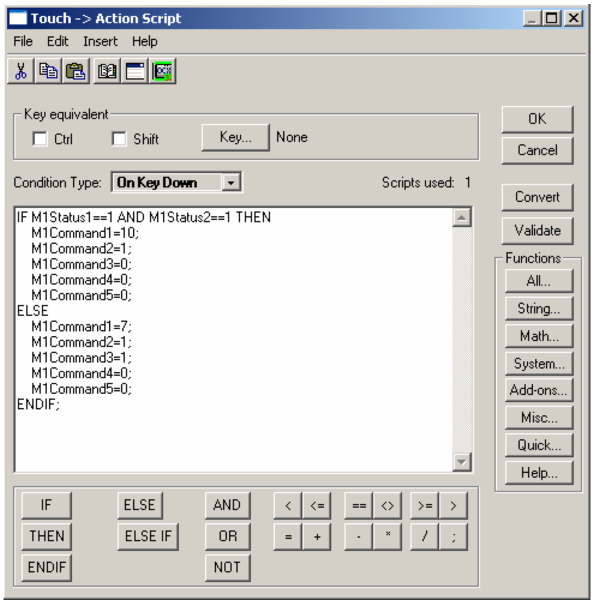
Colors: 1,TRUE,On: 🟩 0,FALSE,Off: 🟥

By clicking on the color boxes at the bottom of the window, you can set the color that will be displayed when CommStatus is equal to 1 (communication is Ok) or equal to 0 (communication problem).

5.3.2 Assigning Actions/Status Indicators to Intercom Icons

Double-click an intercom icon to bring up its properties window then click the “Action” button to set up what will happen when the intercom icon is clicked on. We will set up the icon so that clicking it will either cause a call to be made from the host master to the station (if that master is not already in a call to that station), or to end the call (if there is already a call in progress) between the master and that station. Enter the following in the script window:

```
IF M1Status1==1 AND M1Status2==1 THEN
  M1Command1=10;
  M1Command2=1;
  M1Command3=0;
  M1Command4=0;
  M1Command5=0;
ELSE
  M1Command1=7;
  M1Command2=1;
  M1Command3=1;
  M1Command4=0;
  M1Command5=0;
ENDIF;
```



Condition Type: On Key Down

Scripts used: 1

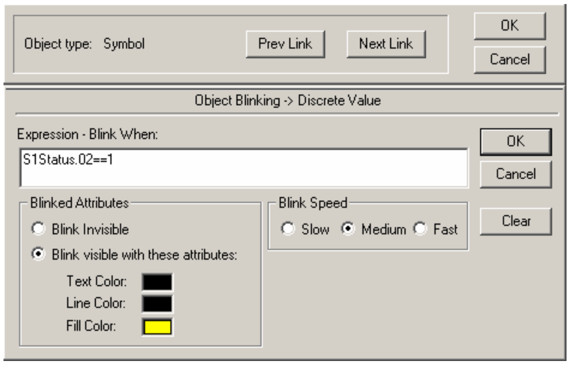
IF M1Status1==1 AND M1Status2==1 THEN
M1Command1=10;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
ELSE
M1Command1=7;
M1Command2=1;
M1Command3=1;
M1Command4=0;
M1Command5=0;
ENDIF;

The script checks M1Status1 and M1Status2 to determine if the host master is connected to a station and if it is, if it connected to Station 1. If the master is connected to station 1, the “EndC” command (code 10) to end the call on Master 1 is sent to the DXL (via the M1Command tags). If the host master is not connected to Station 1, an “Ical” command (code 7) is sent to the DXL to place a call from Master 1 to Station 1.

To set up the icon to blink when the call button has been pressed, select the “Blink” button in the station’s properties window to bring up this window. Select a blink Fill Color as shown and enter the following into the ”Expression – Blink When:” window

```
S1Status.02==1
```

This setting blinks the intercom icon between its normal colour and yellow when a call request comes in (i.e. when bit 2 of S1Status is a 1).



Object type: Symbol

Expression - Blink When: S1Status.02==1

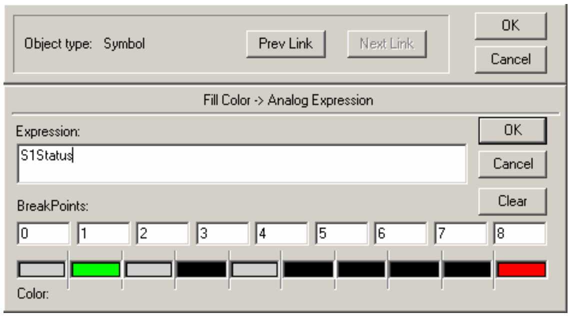
○ Blink Invisible
● Blink visible with these attributes:
Text Color: ■
Line Color: ■
Fill Color: 🟨

○ Slow ● Medium ○ Fast

To set up the icon to change the intercom station color when it is being called from the master, select the “Analog Fill Color” button.

Set the “Expression” to

```
S1Status
```



Object type: Symbol

Fill Color -> Analog Expression

Expression: S1Status

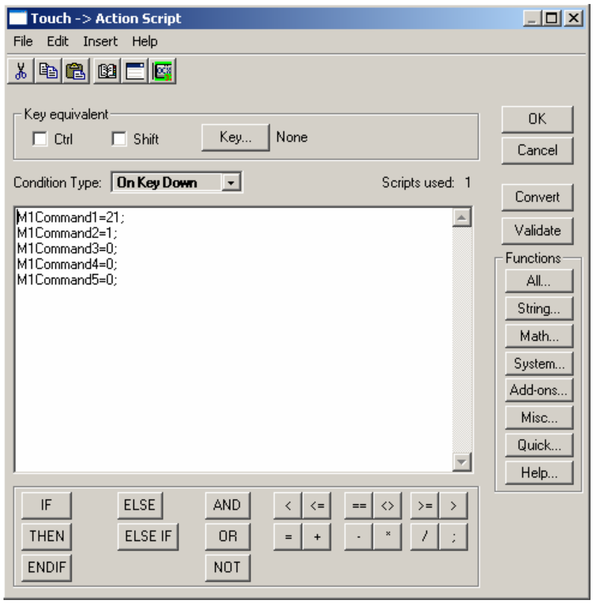
BreakPoints: 0 1 2 3 4 5 6 7 8

Break points are the values of the expression at which the color will change. For station 1, we want to look at values of the tag S1Status so the expression is simply S1Status. S1Status will have a value of 1 when Station 1 is in a call, and the setting above will cause its icon to turn green. If S1Status has a value of 2 then Station 1 is listening to music. If S1Status has a value of 4, Station 1 has a pending call request. If S1Status is greater than or equal to 8, station 1 is faulted and its icon will turn red. In this example we are not setting up music listening so it doesn’t matter what color 2 is. 4 (call request pending) should be set to grey, or else the station will flash between black and yellow instead of grey and yellow when a call request is made.

5.3.3 Assigning Actions to the Answer Next Call and End Call Buttons

This is very similar to defining what will happen when an intercom icon is clicked on. Double click on the “Answer Next Call” button to bring up that button’s properties window, then click on “Action”. The script below sends the “Next” command to the DXL telling it to connect Master 1 (the host master) to the next station in Master 1’s call request queue.

```
M1Command1=21;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```



Touch -> Action Script

Key equivalent: ☐ Ctrl ☐ Shift None

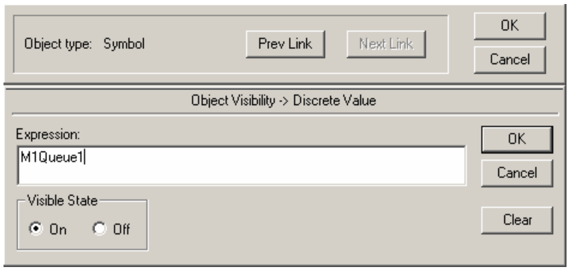
Condition Type: On Key Down

Scripts used: 1

```
M1Command1=21;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```

We may also want to make the “Answer Next Call” button disappear if there are no call requests queued. From the “Answer Next Call” button properties window, click on “Miscellaneous: Visibility” and enter the following expression.

```
M1Queue1
```



Object type: Symbol

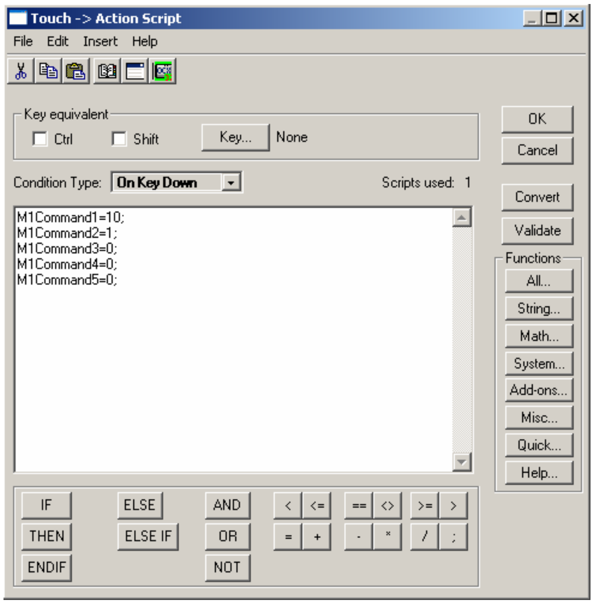
Object Visibility -> Discrete Value

Expression: M1Queue1

Visible State: ● On ○ Off

When M1Queue1 is not equal to zero, there is a call queued for the host master so the button should be visible. Since M1Queue1 is the top of the queue, it is the only queue tag we need to check.

To set up the “End Current Call” button, go to its “Actions” window. The script below sends the DXL the “EndC” command to end the call between Master 1 and whatever Master 1 is connected to.



Key equivalent: ☐ Ctrl ☐ Shift Key... None

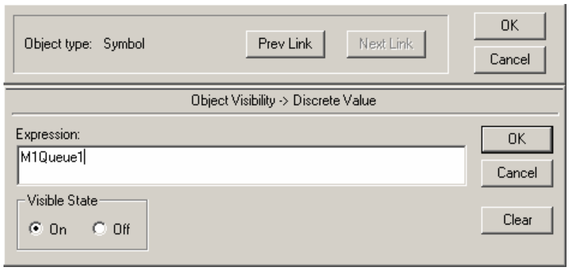
Condition Type: On Key Down

Scripts used: 1

```
M1Command1=10;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```

```
M1Command1=10;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```

We can make the “End Current Call” button disappear if there is no call to end. Select “Miscellaneous: Visibility” from the “End Current Call” buttons properties window and enter the following expression:



Object type: Symbol

Expression: M1Queue1

Visible State: ● On ○ Off

```
M1Queue1
```

M1Status1 will be equal to zero if the host master does not have a call in progress, and the above expression will evaluate to zero, and the button will disappear.

5.3.4 Playing a sound when call requests are active

There are two ways to play a sound when there are calls in the intercom queue.

One would be to periodically sound a reminder tone every few seconds until all calls are answered.

The other method would be to play a sound when a new call comes into the queue. Methods to do each of the above are shown below.

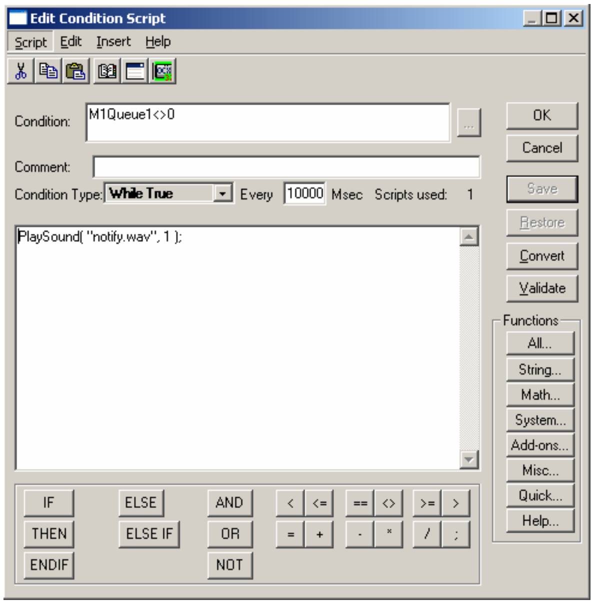
You can have a sound play when a new call comes into the queue and also have a periodic reminder tone by using both methods.

5.3.4.1 *Playing a periodic sound when calls are in the queue*

To play a periodic reminder tone, you can use the queue registers. When there is a call in the queue, the M1Status1 queue register will be non-zero.

You can create a condition script which will act periodically whenever M1Status1 <> 0, and have it play a sound or do other commands. The condition will be "M1Queue1<>0", the Condition Type should be "While TRUE", and the period between tones would be in the Every Msec box (10000 = every 10.000 seconds). To play a notify sound using a sound from the Windows sound files, you can use the following script command

```
PlaySound("notify.wav",1);
```



Condition: M1Queue1<>0

Comment:

Condition Type: While True Every 10000 Msec Scripts used: 1

```
PlaySound( "notify.wav", 1 );
```

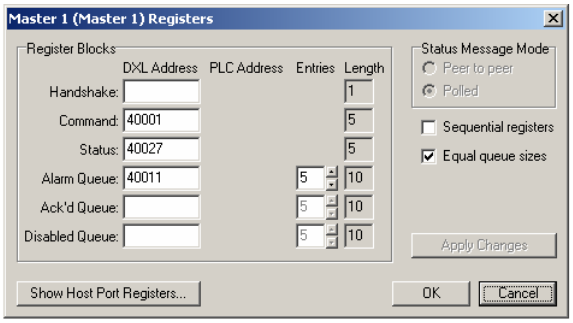
5.3.4.2 *Playing a sound when a new call enters the queue*

To play a sound when a new call enters the queue, you have to know when a new call comes into the queue.

The master message status block (usually used for more advanced host interfacing) can be used for this. Whenever there is activity on the DXL system that a host might need to know, the DXL system can send a message to the host consisting of a 5 register register block containing a command number and up to 4 additional parameters. A new call request consists of a block of 5 registers, with the first register being 1 (call request command), the second register being the master number, and the third register being the station number. You can assign a set of master status registers in the DXL Administrator, then add a master message status register block and write a condition script to play a sound whenever command 1 (Call request) is received.

First, go into the DXL Administrator, host ports, then click on the host you created. Then go to the Masters tab, select your master, then click the "Edit Master Registers…" button. Then add a set of Master message status registers starting at 40027.

Save the configuration, upload it to your DXL system, and Activate the new configuration.



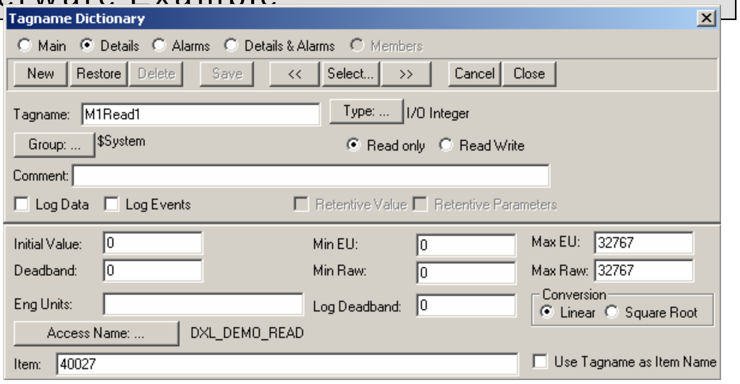
Register Blocks

	DXL Address	PLC Address	Entries	Length
Handshake:				1
Command:	40001			5
Status:	40027			5
Alarm Queue:	40011		5	10
Ack'd Queue:			5	10
Disabled Queue:			5	10

Status Message Mode
○ Peer to peer
● Polled

☐ Sequential registers
☑ Equal queue sizes

Secondly, you need to create tags for these master status register tags. In this example we will call them M1Read1 to M1Read5, with M1Read1 being assigned to register 40027 through M1Read5 being assigned to register 40031. Associate these with the DXL_READ topic. The Wonderware tag entry screen for 40027 is as follows.

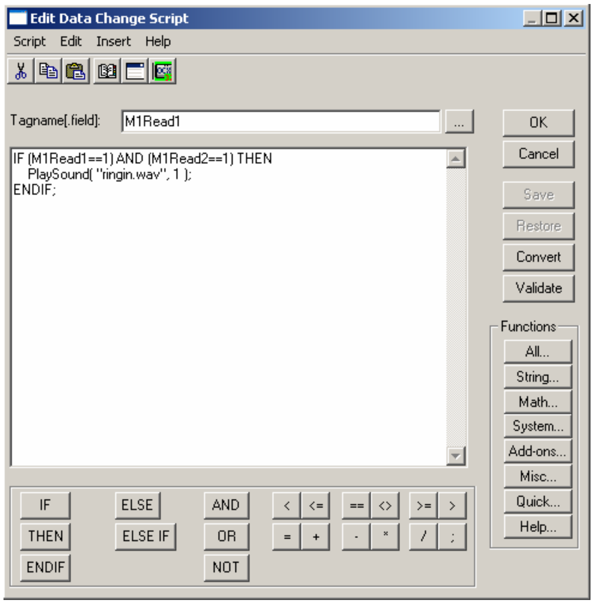


○ Main ● Details ○ Alarms ○ Details & Alarms ○ Members

Tagname: M1Read1
Type: ... I/O Integer
Group: ... $System
● Read only ○ Read Write
Comment:
☐ Log Data ☐ Log Events ☐ Retentive Value ☐ Retentive Parameters
Initial Value: 0
Deadband: 0
Eng Units:
Min EU: 0
Min Raw: 0
Log Deadband: 0
Max EU: 32767
Max Raw: 32767
Conversion: ● Linear ○ Square Root
Access Name: ... DXL_DEMO_READ
Item: 40027
☐ Use Tagname as Item Name

Last, you need to create a data change script that runs when M1Read1 changes, and that plays a sound when the first register equals 1 and the second register equals your master number. Create a new Data Change script, with Tagname = “M1Read1” and the script containing

```
IF (M1Read1==1) AND (M1Read2==1) THEN
       PlaySound(“Ringin.wav”,1);
ENDIF;
```



Tagname[.field]: M1Read1

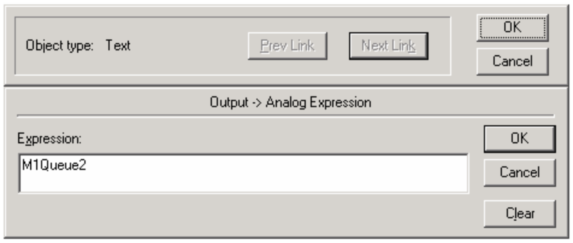
```
IF (M1Read1==1) AND (M1Read2==1) THEN
   PlaySound( "ringin.wav", 1 );
ENDIF;
```

5.3.5 Configuring Queue Text Display

Three different kinds of call requests can occur in a DXL system: a call request from a station, a call request from a master, and an audio level alarm. First, we will learn to display only the ID number of the device calling the master. After that we will look at a more complex example that displays the device type and ID number.

5.3.5.1 Simple Text Display

To display the station number in the call request queue list, double click on the “#” symbol in the list to open its properties window and then select “Value Display: Analog”. M1Queue1 and M1Queue2 contain the device type and device ID, respectively, of the first call request in Master 1’s queue. We therefore want to set the Expression to “M1Queue2”, so its value will take place of the “#” symbol, as shown here.



Object type: Text
Output -> Analog Expression
Expression:
M1Queue2

```
M1Queue2
```

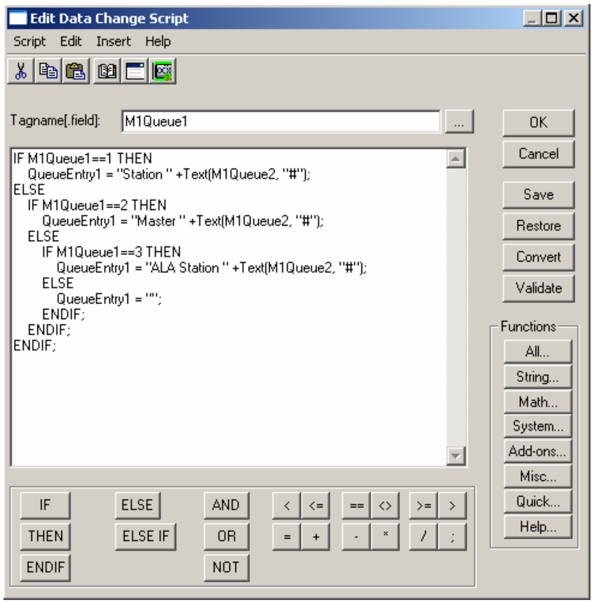
5.3.5.2 More Complex Text Display Using Strings and Scripts

To make a more useful text display we need to define a few more tags to hold the text to be displayed, and then write some scripts (small programs) to manipulate those tags. Define the following tags with a tag type of “Memory Message”:

- QueueEntry1
- QueueEntry2
- QueueEntry3
- QueueEntry4
- QueueEntry5

These tags will hold the display text for each of the entries in the host masters queue. We will now write a set of scripts to automatically enter the proper text into these tags based on the content of the tags M1Queue1 to M1Queue10. Expand the “Scripts” item in the application explorer pane and right click on “Data Change”. When any of the M1Queue tags change value, there has been some change in the call request queue and we need to evaluate what should be displayed.

The script shown here checks the type and ID number of the first call request in the queue, and puts an appropriate string of text into QueueEntry1. This script is executed when M1Queue1 changes value and an identical script should be created that executes when M1Queue2 changes value. The script checks the value of M1Queue1 to determine which prefix (“Station “, “Master “, or “ALA Station “) should be appended to the QueueEntry1 string, then appends the ID number of the device requesting a call (which is contained in M1Queue2). The Text() function used here takes a number from a register and converts it into a string that can be stored in the QueueEntry tag. Its syntax is “Text(<tag to be converted to string>, <number formatting>)”. Please see Wonderware’s scripting language documentation for more information.



Tagname[.field]: M1Queue1

```
IF M1Queue1==1 THEN
  QueueEntry1 = "Station " +Text(M1Queue2, "#");
ELSE
  IF M1Queue1==2 THEN
    QueueEntry1 = "Master " +Text(M1Queue2, "#");
  ELSE
    IF M1Queue1==3 THEN
      QueueEntry1 = "ALA Station " +Text(M1Queue2, "#");
    ELSE
      QueueEntry1 = "";
    ENDIF;
  ENDIF;
ENDIF;
```

```
IF M1Queue1==1 THEN
  QueueEntry1 = "Station " +Text(M1Queue2, "#");
ELSE
  IF M1Queue1==2 THEN
    QueueEntry1 = "Master " +Text(M1Queue2, "#");
  ELSE
    IF M1Queue1==3 THEN
      QueueEntry1 = "ALA Station " +Text(M1Queue2, "#");
    ELSE
      QueueEntry1 = "";
    ENDIF;
  ENDIF;
ENDIF;
```

For each pair of M1Queue tags representing a single QueueEntry tag, a similar script needs to be created. QueueEntry2 is updated when M1Queue3 or M1Queue4 change value, QueueEntry3 is updated when M1Queue5 or M1Queue6 change value, QueueEntry4 is updated when M1Queue7 or M1Queue8 change value, and QueueEntry5 is updated when M1Queue9 or M1Queue10 change value.

6 DXL Example Using Master Status Messages

This section shows how to configure an interface using the more complicated but powerful “Master Status Message” registers. The premise is similar to the above, with commands being sent from the host to the DXL system through a block of 5 registers. However, in this case, the DXL system does not have separate registers for stations indicating their status nor two registers for masters indicating what they are connected to. Instead, the DXL system sends information to the host with a block of 5 registers in much the same way as the commands above, with the first register being the command number and the remaining four registers indicating the parameters. For example, an intercom call request would be sent as the register 1 (“Icrq”, intercom call request) then the master number the call is going to, then the station number that is calling in.

In order to use these in your host you will need to write short scripts that take these status messages and set internal registers, likely in much the form of the above with each station having its own status for calling in and in an call, and each master indicating what it is connected to.

This portion applies both to DXL and DXI systems.

In this example, the scripts will be written such that the station status registers and master connection status registers will be the same as the simplified example above, so that the same programming as the above example can be used with both DXI and DXL systems, and with enhanced features as necessary.

6.1 Setting up Register Addresses on the DXL system

When using a register based host protocol, such as Modbus or Omron, information to be passed between the DXL and host computer is placed in a series of registers which the DXL and computer can read from/write to. The location of these registers and whether the DXL or computer writes to them varies based on the communication mode used. In Peer to Peer mode, the host computer writes commands to command registers on the DXL and the DXL writes responses into status registers on the computer. In Polled mode, all registers are located on the DXL. The host computer writes commands to the DXL command registers and has to regularly read (poll) the DXL status registers to see what is going on. This example uses Polled mode.

There are three basic types of registers needed for this example.

The M1Command registers are used to control Master 1 (the host master). Commands are sent from the host computer to the DXL, with the first register (M1Command1) specifying the command and the other four registers giving parameters that tell the DXL how to execute the command. For example, to make the DXL place a call from Master 1 to Station 6, the host would write 7 (the code for “Intercom Call”) into M1Command1, 1 (the ID number of the master making the call) into M1Command2, 6 (the ID number of the station being called) into M1Command3, and 0 into the remaining M1Command registers (since they are not used for intercom calls).

The M1Read registers indicate the message received from the DXL system. Status update messages are sent from the DXL to the host, with the first register (M1Read1) specifying the status message code and the other four registers giving parameters relevant to the command. For example, when an intercom call button is pressed from Station 3 which calls into Master, the DXL would write 1 (the “Icrq” code indicating an intercom call request) into M1Read1, 1 (the ID number of the master that the station is calling into) into M1Read2, 3 (the ID number of the station calling in) into M1Read3, and 0 into the remaining M1Read registers (since they are not used for intercom call requests).

The M1Queue registers contain the type and ID number of Master 1's call request queue. M1Queue1 contains the type of device at the top of Master 1's queue, M1Queue2 contains the ID number of the device at the top of Master 1's queue, M1Queue3 and M1Queue4 contain the type and ID of the second device in the queue, and so on. The queue registers are optional, but they are useful for displaying an ordered list of call requests on the host. The size of the queue is configurable via the DXL Administrator or DXI SAC software and the number of registers required would always be twice the number of devices that the queue can store. Note that regardless of the queue length set here, the DXL will still queue up call requests internally; the queue registers simply determine how much of that queue will be available for the host to read.

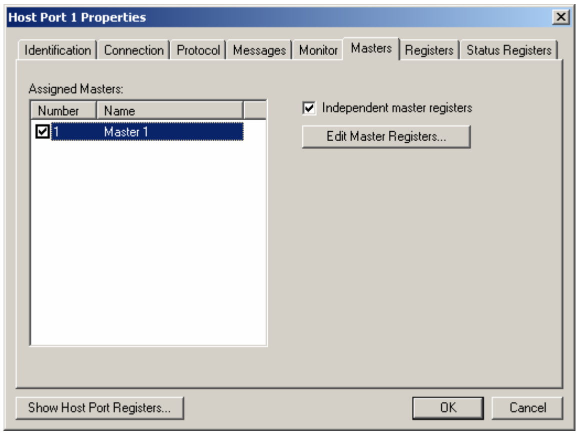
For a complete set of commands and parameters, please see the document "DXL Host Interface Specifications", available at http://www.harding.ca.

For this example, we need to define the following registers:

Register/Tag Name	Modbus Register Address	Omron FINS Register Address
M1Command1	40001	D0001
M1Command2	40002	D0002
M1Command3	40003	D0003
M1Command4	40004	D0004
M1Command5	40005	D0005
M1Queue1	40011	D0011
M1Queue1	40012	D0012
M1Queue3	40013	D0013
M1Queue4	40014	D0014
M1Queue5	40015	D0015
M1Queue6	40016	D0016
M1Queue7	40017	D0017
M1Queue8	40018	D0018
M1Queue9	40019	D0019
M1Queue10	40020	D0020
M1Read1	40027	D0027
M1Read2	40028	D0028
M1Read3	40029	D0029
M1Read4	40030	D0030
M1Read5	40031	D0031

The text and pictures below are for the Modbus protocol, but setting up registers for use with Omron is extremely similar: simply replace the "4" at the start of every Modbus address with a "D".

Select “Host Ports” from the drop down menu in DXL Administrators configuration editor and double click the host port we created earlier to bring up its properties window. Select the “Masters” tab. To define command and status registers for our master, we need to have selected the master from the list on the left (i.e., there is a checkmark in the box next to Master 1’s name) and have “Independent Master Registers” selected.

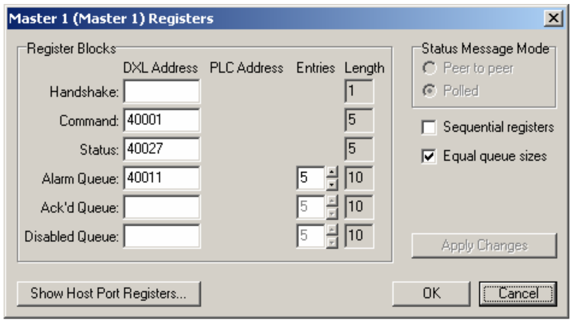


Assigned Masters:

Number	Name
☑ 1	Master 1

☑ Independent master registers

Highlight “Master 1” and click on “Edit Master Registers”. The fields under “DXL Address” are the starting addresses of the different sets of registers. “Length” indicates how many registers are contained in a set. Change the settings to match the following screenshot and click “OK”. This will make Master 1’s command registers 40001-40005, Master 1’s read status registers 40027-40031 and Master 1’s Queue registers 40011-40021. Note that register addresses are arbitrary, but duplicating the settings shown here will make it easier to follow the Wonderware tag setup in this document, since it uses the same addresses.



Register Blocks

	DXL Address	PLC Address	Entries	Length
Handshake:				1
Command:	40001			5
Status:	40027			5
Alarm Queue:	40011		5	10
Ack'd Queue:			5	10
Disabled Queue:			5	10

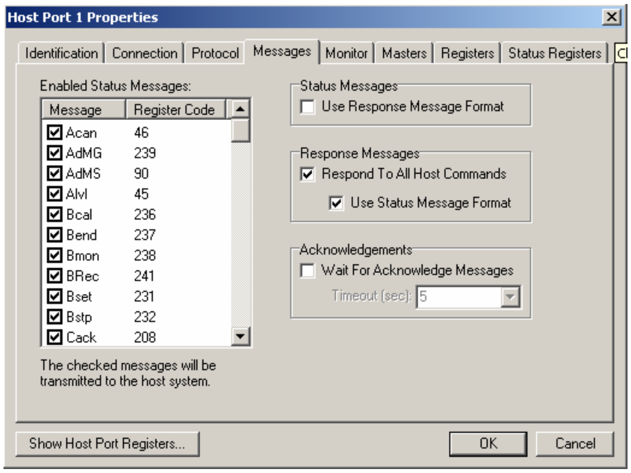
Status Message Mode
○ Peer to peer
● Polled

☐ Sequential registers
☑ Equal queue sizes

6.2 Turning off the “Done” Setting for Master Status Messages

This next step will make the script which decodes the master status messages simpler. Normally, the DXL system can indicate whether an action happened independently (such as a master station with its own display and keypad that calling an intercom using its keypad) or through a host response (such as a touchscreen initiating a call to a station). Independent actions will be simply the command (“Ical 1 2” indicates a call was connected from master 1 to station 2), while responses to host commands will be the command sent, prefixed by “Done” (“Done Ical 1 2” indicates that the host request to connect master 1 to station 2 was successful). The DXL system can be set so that whether the master initiated the connection itself, or through a response to a host command, it will send the message without the “Done” (so all connections will simply be “Ical 1 2”).

To set this behavior, go to the Host Port properties, Messages tab, and check mark the “Respond to All Host Commands” box and the “Use Status Message Format” box as shown here.



Enabled Status Messages:

Message	Register Code
☑ Acan	46
☑ AdMG	239
☑ AdMS	90
☑ Alvl	45
☑ Bcal	236
☑ Bend	237
☑ Bmon	238
☑ BRec	241
☑ Bset	231
☑ Bstp	232
☑ Cack	208

The checked messages will be transmitted to the host system.

Status Messages
☐ Use Response Message Format

Response Messages
☑ Respond To All Host Commands
☑ Use Status Message Format

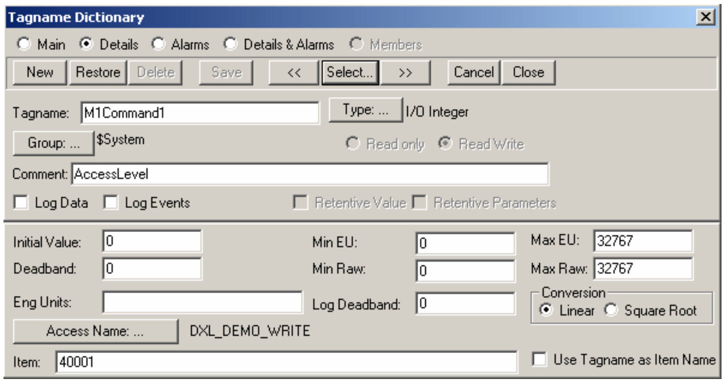
Acknowledgements
☐ Wait For Acknowledge Messages
Timeout (sec): 5

6.3 Setting up Wonderware Tags for registers in the DXL System:

Wonderware uses “tags” to keep track of data, both internal to the program and obtained from external sources such as the DXL. In this section we will define tags that correspond to the registers we defined on the DXL and a tag internal to Wonderware that monitors the status of the host connection

6.3.1 Creating Tags for communication with the DXL

Double clicking the “Tagname Dictionary” icon will bring up the following screen:



○ Main ● Details ○ Alarms ○ Details & Alarms ○ Members

Tagname: M1Command1

Type: I/O Integer

Group: $System

○ Read only ● Read Write

Comment: AccessLevel

☐ Log Data ☐ Log Events ☐ Retentive Value ☐ Retentive Parameters

Initial Value: 0

Deadband: 0

Eng Units:

Min EU: 0

Min Raw: 0

Log Deadband: 0

Max EU: 32767

Max Raw: 32767

Conversion: ● Linear ○ Square Root

Access Name: DXL_DEMO_WRITE

Item: 40001

☐ Use Tagname as Item Name

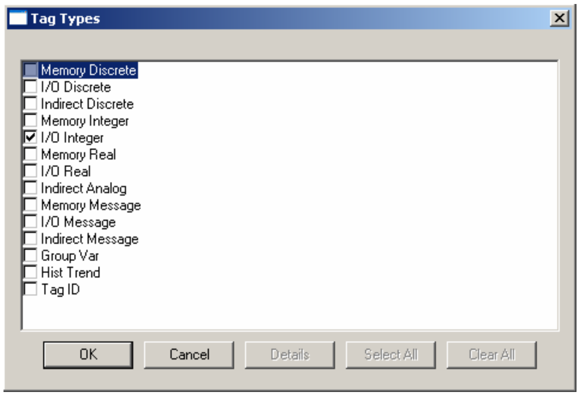
This screen is used to create and edit tag data. We need to define the following tags:

- M1Command1 to M1Command5 (used to send host master commands to the DXL)
- M1Queue1 to M1Queue10 (used to keep track of the call request queue for the host)
- M1Read1 to M1Read5 (used to receive status messages from the DXL)

The tags M1Command1 to M1Command5 should be set as “Read/Write”. All the other tags should be set as “Read Only”.

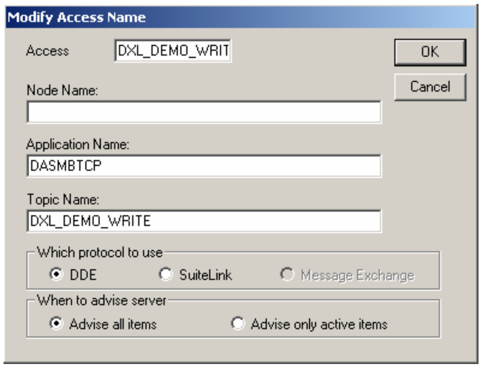
A tag’s “Item” is set to the address of the DXL register that will be associated with the tag. DXL register addresses were defined using DXL Administrator in section 6.1 of this document.

Clicking “Tag Type” brings up a prompt with all the possible tag types. Memory registers are internal to the Wonderware programming and I/O registers are obtained from Modbus TCP/IP or other protocols. All of the tags we are defining should be of the type “I/O Integer”.



☐ Memory Discrete
☐ I/O Discrete
☐ Indirect Discrete
☐ Memory Integer
☑ I/O Integer
☐ Memory Real
☐ I/O Real
☐ Indirect Analog
☐ Memory Message
☐ I/O Message
☐ Indirect Message
☐ Group Var
☐ Hist Trend
☐ Tag ID

Each tag needs to have an Access Name associated with it. These access names are associated with the topics that we defined in Section 2 of this document and will tell Wonderware where to get tag data from and how often to look for new tag data. Click on the “Access Name:…” button and then on the “Add…” button in the window that appears. You will now be looking at the “Modify Access Name” window.



Access: DXL_DEMO_WRIT
Node Name:
Application Name: DASMBTCP
Topic Name: DXL_DEMO_WRITE
Which protocol to use: ● DDE ○ SuiteLink ○ Message Exchange
When to advise server: ● Advise all items ○ Advise only active items

“Application Name” is the IO Communication Server. Modbus TCP/IP communication protocol uses the “DASMBTCP” server, and Omron FINS communication protocol uses the “FINSGTWY” server.

The “Topic Name” is the topic name assigned while configuring the IO Server. Create one access name for each of the topic names (DXL_DEMO_READ, DXL_DEMO_WRITE, and DXL_DEMO_QUEUE).

The “Protocol” is the method used to exchange data between the DA Server and Wonderware and can be set to DDE or SuiteLink.

The “When to advise server” item indicates whether to constantly poll these registers (Advise all items) or when to only poll them when they are being used by any running script or when any icon using them is visible (Advise only active items). For this example it does not matter what “When to advise server” is set to, since there will always be visible icons using the tags, but we recommend “Advise all items” to be safe.

Create the rest of the tags as listed in the table in section 6.1, making sure that the register addresses in DXL Administrator match the “Item” addresses in Wonderware, and assigning the following access names to the tags:

- Tags M1Queue1 to M1Queue10 will use the access name associated with the DXL_DEMO_QUEUE topic
- Tags M1Command1 to M1Command5 will use the access name associated with the DXL_DEMO_WRITE topic
- Tags M1Read1 to M1Read5 will use the access name associated with the DXL_DEMO_READ topic

You should now have all the DXL I/O tags defined. Double check by clicking on “Select…”, which brings up a list of all tags defined in the current Wonderware project.

6.3.2 Creating Tags for station status and master connection status

The most common way for the host to display station status is to have a separate register (or bit) for each station indicating the station’s call request and connection status. This protocol does not provide that information directly; instead the host software must set this internal information by reading the master status messages, interpreting the status message codes it is interested in (such as “Icrq” call request messages) and use the parameters (I.E. master and station number for an “Icrq” message) to set up its own internal registers which keep track of the station’s status.

Likewise you will need to interpret the “master connection” messages such as “Ical” and “EndC” messages sent by the DXL system when it connects or disconnects the audio, and set the master connection status appropriately.

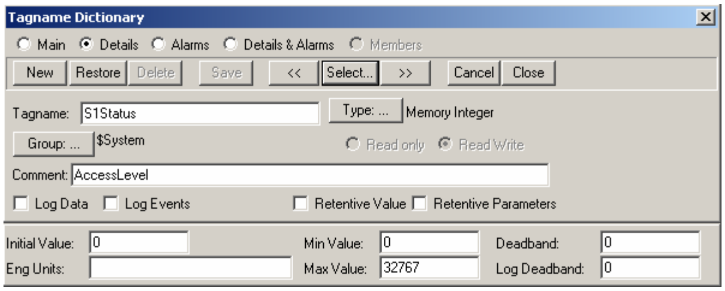
In this example, the same format as the DXL's native "Station Status Registers" and "Master Status Registers" will be used.

The Station Status registers will indicate the connection state of the stations. Each bit of the station register indicates a different condition that the station is either in or not. For any single condition, 1 means yes and 0 means no. The bits in a station status register that concern us in this example are bit 0 (the least significant bit), which represents whether the station is in a call or not, and bit 2, which represents whether the station has a call request pending or not. These registers will be named "S1Status" to "S6Status"

The Master Connection Status registers will indicate the connection state of Master 1. In this example, to be consistent with the M1Status1 will contains a number indicating the type of device that Master 1 is connected to (0=nothing, 1=master, 2=station, 3=page zone), and M1Status2 contains the ID number of the device.

To create the register tags to store the status of the individual stations, create these new tags in the Tagname Dictionary, using the "Memory Integer" type. The settings for S1Status are shown here.

Create S1Status through S6Status, then M1Status1 and M1Status2.



○ Main ● Details ○ Alarms ○ Details & Alarms ○ Members

Tagname: S1Status

Memory Integer

$System

○ Read only ● Read Write

Comment: AccessLevel

☐ Log Data ☐ Log Events ☐ Retentive Value ☐ Retentive Parameters

Initial Value: 0

Eng Units:

Min Value: 0

Max Value: 32767

Deadband: 0

Log Deadband: 0

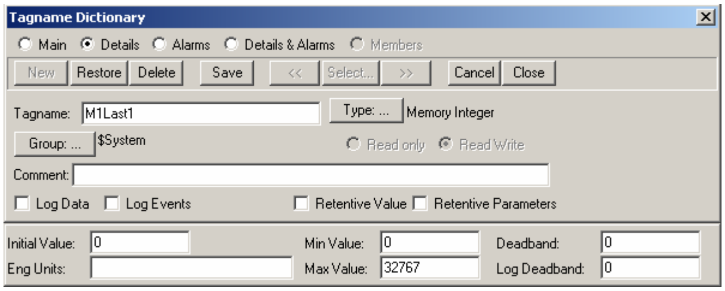
The values in these registers will be set by the script that evaluates the master status messages.

6.3.3 Creating Tags for determining when new messages arrive

When the DXL system has a new message to send (for example, if a call request button is pushed), it will send it to the host (if using peer to peer mode) or have it available on its master status registers for the next time the host reads it (if using polled mode). Ultimately the host will have updated registers (tags) with the new information in it. Whenever a new status message is received the host software should interpret this message with its scripts.

However, these scripts should only be executed when a new message is received, I.E. when any of the status message registers change. In this example, the scripts will keep a temporary copy of the last block of 5 registers received. On a periodic basis, it will compare the current value of the I/O registers and compare them to the last block received. If the contents of any of the I/O registers is different than the equivalent register in the "last received" registers, then a new message is received.

In this example, we will create new "Memory Integer" tags "M1Last1" through "M1Last5" to store the previous values that were received in "M1Read1" through "M1Read5" for this purpose. An example window for M1Last1 is shown here.



○ Main ● Details ○ Alarms ○ Details & Alarms ○ Members

Tagname: M1Last1

Memory Integer

$System

○ Read only ● Read Write

Comment:

☐ Log Data ☐ Log Events ☐ Retentive Value ☐ Retentive Parameters

Initial Value: 0

Eng Units:

Min Value: 0

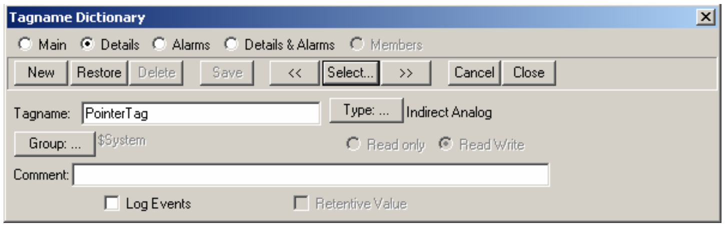
Max Value: 32767

Deadband: 0

Log Deadband: 0

6.3.4 Creating a Pointer Tag to simulate array operation

The scripts that follow will write to the station status tag for a station whenever a relevant message for that station is received. An indirect pointer type is created to simulate an array so that the script can set a bit in that station's tag, given a station number in a parameter received from the DXL system.



Tagname: PointerTag
Type: ... Indirect Analog
Group: ... $System
○ Read only ● Read Write
Comment:
☐ Log Events ☐ Retentive Value

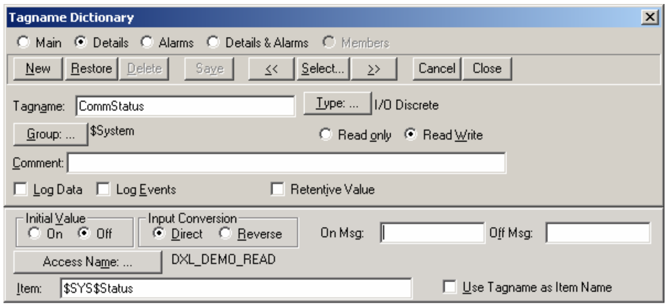
Create a new tag with the tagname "PointerTag", of "Indirect Analog" type as shown here.

6.3.5 Wonderware Tag for determining communications status of DXL System

This tag is a DA server tag that indicates the connection status of the DXL system

(Discrete is the Wonderware term for a Boolean On/Off value)

This is set to 1 when the DA Server is connected to the PLC (DXL system) and the DXL is responding to polls, or 0 when the DA Server is not connected to the PLC (DXL system).



Tagname: CommStatus
Type: ... I/O Discrete
Group: ... $System
○ Read only ● Read Write
Comment:
☐ Log Data ☐ Log Events ☐ Retentive Value
Initial Value: ○ On ● Off
Input Conversion: ● Direct ○ Reverse
On Msg:
Off Msg:
Access Name: ... DXL_DEMO_READ
Item: SYSStatus
☐ Use Tagname as Item Name

6.4 Creating a Script to set Station Status and Master Connection Status

This step is to create a script for interpreting the messages, and setting the internal status registers when new call requests are queued or connections made.

The main script will run whenever there is a new message.

This example will look for "Icrq" call request messages, "Ican" cancel call request messages, "Ical" master-to-intercom call connect messages, and "EndC" and "Iend" master end call messages.

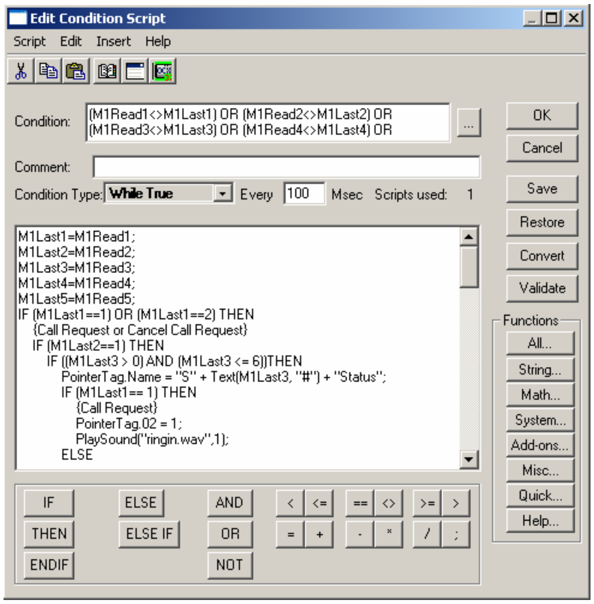
Create a Condition script.

Set Condition to:

```
(M1Read1<>M1Last1) OR (M1Read2<>M1Last2) OR
(M1Read3<>M1Last3) OR (M1Read4<>M1Last4) OR
(M1Read5<>M1Last5)
```

Set Condition Type to "While True".

Set Every Msec to 100 msec.



Condition: (M1Read1<>M1Last1) OR (M1Read2<>M1Last2) OR (M1Read3<>M1Last3) OR (M1Read4<>M1Last4) OR
Comment:
Condition Type: While True Every 100 Msec Scripts used: 1

```
M1Last1=M1Read1;
M1Last2=M1Read2;
M1Last3=M1Read3;
M1Last4=M1Read4;
M1Last5=M1Read5;
IF (M1Last1==1) OR (M1Last1==2) THEN
   {Call Request or Cancel Call Request}
   IF (M1Last2==1) THEN
      IF ((M1Last3 > 0) AND (M1Last3 <= 6))THEN
         PointerTag.Name = "S" + Text(M1Last3, "#") + "Status";
         IF (M1Last1== 1) THEN
            {Call Request}
            PointerTag.02 = 1;
            PlaySound("ringin.wav",1);
         ELSE
```

This must be faster than the poll rate defined in your topic setup. If it is not, you will lose messages, as your Modbus polling will occur faster than you are processing the messages. This will cause occasional call requests or other updates to be missed. It is recommended to set this to at least half the time (I.E. twice the scan rate) as the polling rate. The poll rate is set in Section 3 above. In this case, the poll rate is 200 ms, so this script execution rate should be half that, or at least every 100 ms.

The script in the screenshot is only the first part. The whole script is shown below.

```
M1Last1=M1Read1;
M1Last2=M1Read2;
M1Last3=M1Read3;
M1Last4=M1Read4;
M1Last5=M1Read5;
IF (M1Last1==1) OR (M1Last1==2) THEN
      {Call Request or Cancel Call Request}
      IF (M1Last2==1) THEN
            IF ((M1Last3 > 0) AND (M1Last3 <= 6))THEN
                  PointerTag.Name = "S" + Text(M1Last3, "#") + "Status";
                  IF (M1Last1==1) THEN
                        {Call Request}
                        PointerTag.02 = 1;
                        PlaySound("ringin.wav",1);
                  ELSE
                        {Cancel Call Request}
                        PointerTag.02 = 0;
                  ENDIF;
            ENDIF; {Stations 1-6}
      ENDIF; {Master 1}
ENDIF; {Call Request}
IF (M1Last1==7) THEN
      {Call Connect}
      IF (M1Last2 == 1) THEN
            IF ((M1Last3 > 0) AND (M1Last3 <= 6)) THEN
                  PointerTag.Name = "S" + Text(M1Last3, "#") + "Status" ;
                  PointerTag.00 = 1;
            ENDIF;
            M1Status1 = 1;
            M1Status2 = M1Last3;
      ENDIF;
ENDIF; {Call Connect}
IF ((M1Last1==10) OR (M1Last1==200) OR (M1Last1==201) OR (M1Last1==202)) THEN
      {Call Disconnect}
      IF (M1Last2 == 1) THEN
            IF (M1Status1==1) THEN
                  IF ((M1Status2 > 0) AND (M1Status2 <= 6)) THEN
                        PointerTag.Name = "S" + Text(M1Status2, "#") + "Status" ;
                        PointerTag.00 = 0;
                  ENDIF;
            ENDIF;
            M1Status1 = 0;
            M1Status2 = 0;
      ENDIF;
ENDIF; {Call Disconnect}
```

The following outlines the details for this script.

6.4.1 Setting last register values

Once a change is detected, set the last register values to the current register values

```
M1Last1=M1Read1;
M1Last2=M1Read2;
M1Last3=M1Read3;
M1Last4=M1Read4;
M1Last5=M1Read5;
```

6.4.2 Setting station status for call request and call request cancel

If the message received is an Intercom Call Request (“Icrq”, code 1) or Intercom Call Request Cancel (“Ican”, code 2), then set the call request status for the station.

```
IF (M1Last1==1) OR (M1Last1==2) THEN
      {Call Request or Cancel Call Request}
```

Only process this call request if the second parameter indicates it is for this master (master 1), and the third parameter indicates the station is between station 1 and 6

```
      IF (M1Last2==1) THEN
            IF ((M1Last3 > 0) AND (M1Last3 <= 6))THEN
```

This next part takes the third parameter which is the station number, and sets a pointer tag name to “Station#Status” where # is the station number in the third parameter.

If this message was a call request (M1Last1==1), it will set bit 2 in that tag to 1. Otherwise if the message was a call request cancel (M1Last1==2), it will set bit 2 in that tag to 0.

Essentially, if the third parameter is “6” then it will set bit 2 of the “Station6Status” to 1 (if this is a call request) or 0 (if this is a call request cancel). Also, if this is a call request, play a “ring” tone on the PC speakers.

```
                  PointerTag.Name = "S" + Text(M1Last3, "#") + "Status";
                  IF (M1Last1==1) THEN
                        {Call Request}
                        PointerTag.02 = 1;
                        PlaySound("ringin.wav",1);
                  ELSE
                        {Cancel Call Request}
                        PointerTag.02 = 0;
                  ENDIF;
```

Finally, close up the if statements

```
            ENDIF; {Stations 1-6}
      ENDIF; {Master 1}
ENDIF; {Call Request}
```

6.4.3 Setting station status and master connection status for intercom call connections

If the message received is an Intercom Call Connect (“Ical”, code 7), then set the call in progress status for the station, and set the master connection status to “connected to that station.”

```
IF (M1Last1==7) THEN
      {Call Connect}
```

Only process this call connection if the second parameter indicates it is for this master (master 1), and the third parameter indicates the station is between station 1 and 6

```
      IF (M1Last2==1) THEN
            IF ((M1Last3 > 0) AND (M1Last3 <= 6))THEN
```

Set “Station#Status” bit 0 to 1 (call is in progress). The station number that is being called is in the third parameter.

```
                  PointerTag.Name = "S" + Text(M1Last3, "#") + "Status" ;
                  PointerTag.00 = 1;
            ENDIF;
```

Set M1Status1 to 1 (In a call to a station) and M1Status2 to the station number it is in a call to (set from the third parameter), and close off the IF statements.

```
            M1Status1 = 1;
            M1Status2 = M1Last3;
      ENDIF;
ENDIF; {Call Connect}
```

6.4.4 Setting station status and master connection status for call disconnections

If the message received is an Intercom Call Disconnect (“EndC”, code 10), Intercom End call (“Iend”, code 200), Master End call (“Mend”, code 201), or Page End call (“Pend”, code 202), then set the call ended status for the station, and set the master connection status to “connected to nothing.” While this example does not use page or master calls, if the master makes calls from its keypad (if it has one), this will correctly show the master’s status on the host if it happens to make a page or master call.

```
IF ((M1Last1==10) OR (M1Last1==200) OR (M1Last1==201) OR (M1Last1==202)) THEN
      {Call Disconnect}
```

Only process this message if the message is for this master (M1Last2==1)

```
      IF (M1Last2 == 1) THEN
```

If the master was currently connected to a station (Master connect status register M1Status1==1), and the station is a valid station (between 1 station 1 and 6) then set that station’s status to “not in a connection” by setting “Station#Status” bit 0 to 0. The station number it was connected to was in the M1Status2 register (set when the master was connected to the station above).

```
            IF (M1Status1==1) THEN
                  IF ((M1Status2 > 0) AND (M1Status2 <= 6)) THEN
                        PointerTag.Name = "S" + Text(M1Status2, "#") + "Status" ;
                        PointerTag.00 = 0;
                  ENDIF;
            ENDIF:
```

Set the master connected status to “connected to no device, no ID number” no matter what it was connected to previously.

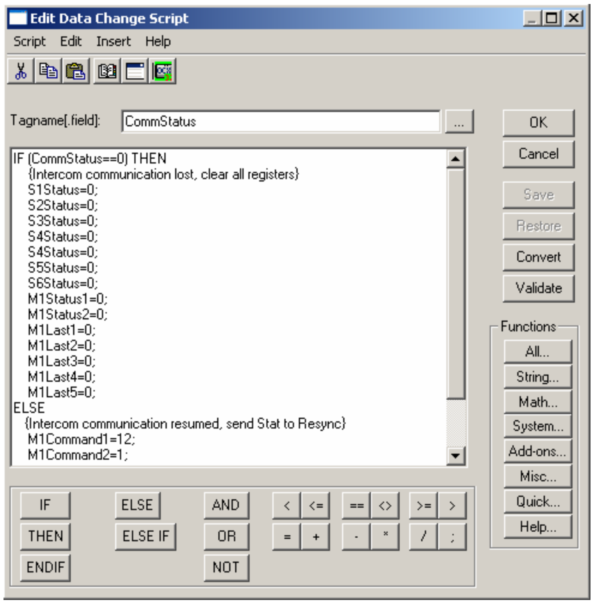
```
            M1Status1 = 0;
            M1Status2 = 0;
      ENDIF;
ENDIF; {Call Disconnect}
```

6.5 Creating a Script to Request Current Status on Startup and Communication Loss

The DXL intercom system operates independently of the host computers, and can have call requests and other activity before the host software is activated, or if the communication is lost between the host and the DXL system. When the host starts up, or after communication is back up, there is a method to update the host with the current DXL status. This can be done with the Stat command, which requests sending all information regarding one master, or the AllS command, which requests sending all information for all masters. If the host issues this when communication is resumed, then the intercom system will send all related activity for this master or masters as standard messages as if that activity happened just now. For example, any active call requests will be sent with the “Icrq” command (code 1) just as if the button was pressed now, and call connections would also be re-sent as if it was connected now.

To configure the host to send this command to the DXL system, you can create a data change script. Set the Tagname to the CommStatus tag. Then enter the following script.

```
IF (CommStatus==0) THEN
    {Intercom communication lost,
    clear all registers}
    S1Status=0;
    S2Status=0;
    S3Status=0;
    S4Status=0;
    S4Status=0;
    S5Status=0;
    S6Status=0;
    M1Status1=0;
    M1Status2=0;
    M1Last1=0;
    M1Last2=0;
    M1Last3=0;
    M1Last4=0;
    M1Last5=0;
ELSE
   {Intercom communication resumed,
    send Stat to Resync}
    M1Command1=12;
    M1Command2=1;
    M1Command3=0;
    M1Command4=0;
    M1Command5=0;
ENDIF;
```



Tagname[.field]: CommStatus

```
IF (CommStatus==0) THEN
   {Intercom communication lost, clear all registers}
   S1Status=0;
   S2Status=0;
   S3Status=0;
   S4Status=0;
   S4Status=0;
   S5Status=0;
   S6Status=0;
   M1Status1=0;
   M1Status2=0;
   M1Last1=0;
   M1Last2=0;
   M1Last3=0;
   M1Last4=0;
   M1Last5=0;
ELSE
  {Intercom communication resumed, send Stat to Resync}
   M1Command1=12;
   M1Command2=1;
```

The CommStatus tag was previously defined, and indicates the connection status (0=not connected, 1=connected).

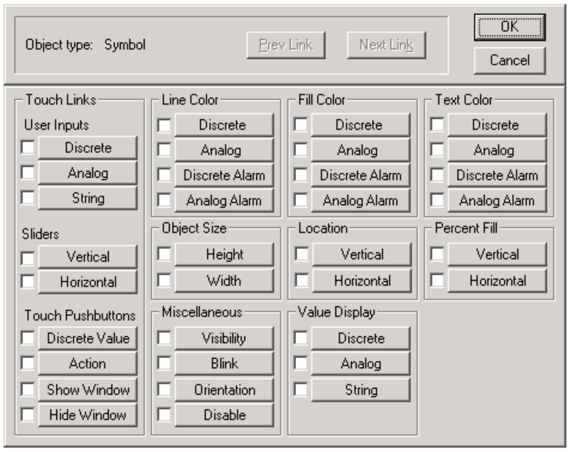
When the communication is lost (CommStatus==0), the script sets all stations to not connected and not calling in, and the master to not connected. When the communication link resumes, the script sends the Stat command (code 12) for master 1 to indicate to the DXL system to re-send the system status messages including active call requests and audio connections.

6.6 Configuring Host Behaviour

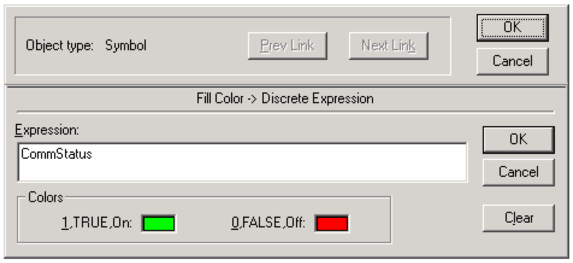
Now we need to associate user actions with commands we want sent to the DXL, and tag values with what we want displayed on screen. For information on the commands and register structure used for communication between the DXL and a host, see “DXL Host Interface Specifications”, which is available for download at the Harding Instruments website.

6.6.1 Configuring the Communication Status Indicator

Double click on the communication status indicator to bring up the properties window for the communication status indicator:



We want the indicator to be green when communication between the host and DXL is working, and red when there is a problem. Click on the “Fill Color: Discrete” button to get to the following window.



Expression: CommStatus

Colors

1,TRUE,On: 🟩

0,FALSE,Off: 🟥

Set “Expression” to the name name of our communication status tag (CommStatus).

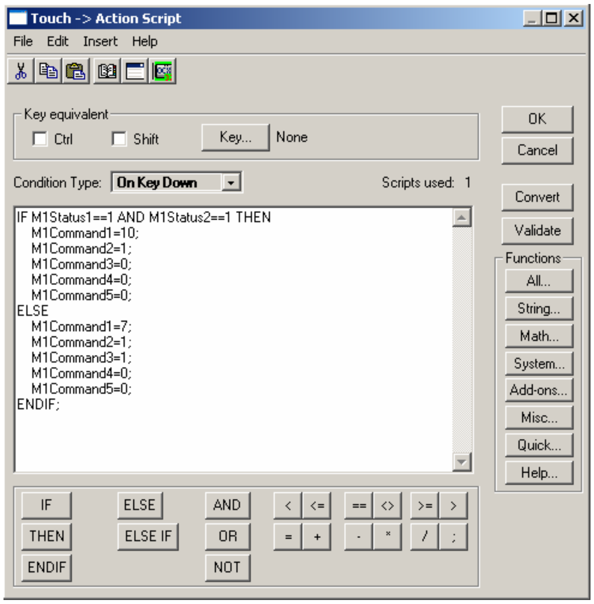
```
CommStatus
```

By clicking on the color boxes at the bottom of the window, you can set the color that will be displayed when CommStatus is equal to 1 (communication is Ok) or equal to 0 (communication problem).

6.6.2 Assigning Actions/Status Indicators to Intercom Icons

Double-click an intercom icon to bring up its properties window then click the “Action” button to set up what will happen when the intercom icon is clicked on. We will set up the icon so that clicking it will either cause a call to be made from the host master to the station (if that master is not already in a call to that station), or to end the call (if there is already a call in progress) between the master and that station. Enter the following in the script window:

```
IF M1Status1==1 AND M1Status2==1 THEN
  M1Command1=10;
  M1Command2=1;
  M1Command3=0;
  M1Command4=0;
  M1Command5=0;
ELSE
  M1Command1=7;
  M1Command2=1;
  M1Command3=1;
  M1Command4=0;
  M1Command5=0;
ENDIF;
```



Condition Type: On Key Down

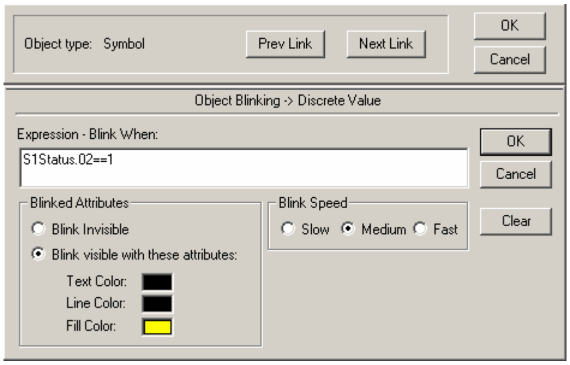
Scripts used: 1

```
IF M1Status1==1 AND M1Status2==1 THEN
   M1Command1=10;
   M1Command2=1;
   M1Command3=0;
   M1Command4=0;
   M1Command5=0;
ELSE
   M1Command1=7;
   M1Command2=1;
   M1Command3=1;
   M1Command4=0;
   M1Command5=0;
ENDIF;
```

The script checks M1Status1 and M1Status2 to determine if the host master is connected to a station and if it is, if it connected to Station 1. If the master is connected to station 1, the “EndC” command (code 10) to end the call on Master 1 is sent to the DXL (via the M1Command tags). If the host master is not connected to Station 1, an “Ical” command (code 7) is sent to the DXL to place a call from Master 1 to Station 1.

To set up the icon to blink when the call button has been pressed, select the “Blink” button in the station’s properties window to bring up this window. Select a blink Fill Color as shown and enter the following into the ”Expression – Blink When:” window

```
S1Status.02==1
```



Object type: Symbol

Expression - Blink When: S1Status.02==1

○ Blink Invisible
● Blink visible with these attributes:

Text Color: ■
Line Color: ■
Fill Color: 🟨

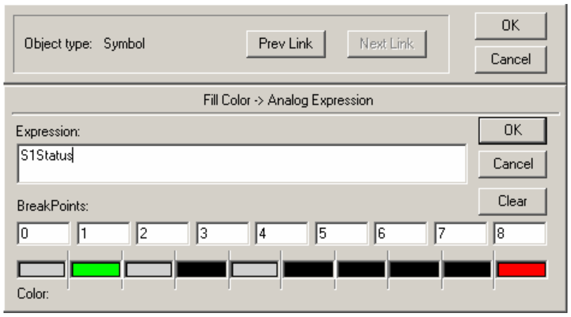
○ Slow ● Medium ○ Fast

This setting blinks the intercom icon between its normal colour and yellow when a call request comes in (i.e. when bit 2 of S1Status is a 1).

To set up the icon to change the intercom station color when it is being called from the master, select the “Analog Fill Color” button.

Set the “Expression” to

```
S1Status
```



Object type: Symbol

Fill Color -> Analog Expression

Expression: S1Status

BreakPoints: 0 1 2 3 4 5 6 7 8

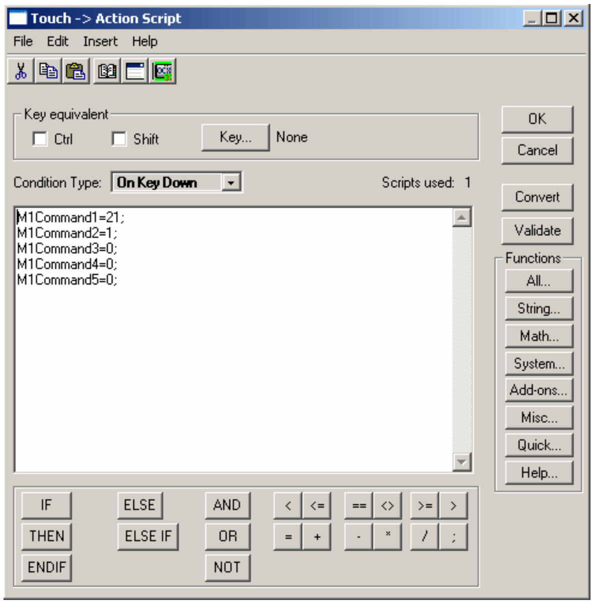
Color: ⬜ 🟩 ⬜ ⬛ ⬜ ⬛ ⬛ ⬛ ⬛ 🟥

Break points are the values of the expression at which the color will change. For station 1, we want to look at values of the tag S1Status so the expression is simply S1Status. S1Status will have a value of 1 when Station 1 is in a call, and the setting above will cause its icon to turn green. If S1Status has a value of 2 then Station 1 is listening to music. If S1Status has a value of 4, Station 1 has a pending call request. If S1Status is greater than or equal to 8, station 1 is faulted and its icon will turn red. In this example we are not setting up music listening so it doesn’t matter what color 2 is. 4 (call request pending) should be set to grey, or else the station will flash between black and yellow instead of grey and yellow when a call request is made.

6.6.3 Assigning Actions to the Answer Next Call and End Call Buttons

This is very similar to defining what will happen when an intercom icon is clicked on. Double click on the “Answer Next Call” button to bring up that button’s properties window, then click on “Action”. The script below sends the “Next” command to the DXL telling it to connect Master 1 (the host master) to the next station in Master 1’s call request queue.

```
M1Command1=21;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```



Touch -> Action Script

Key equivalent: ☐ Ctrl ☐ Shift None

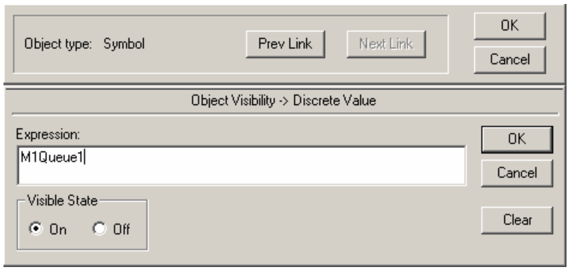
Condition Type: On Key Down

Scripts used: 1

```
M1Command1=21;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```

We may also want to make the “Answer Next Call” button disappear if there are no call requests queued. From the “Answer Next Call” button properties window, click on “Miscellaneous: Visibility” and enter the following expression.

```
M1Queue1
```



Object type: Symbol

Object Visibility -> Discrete Value

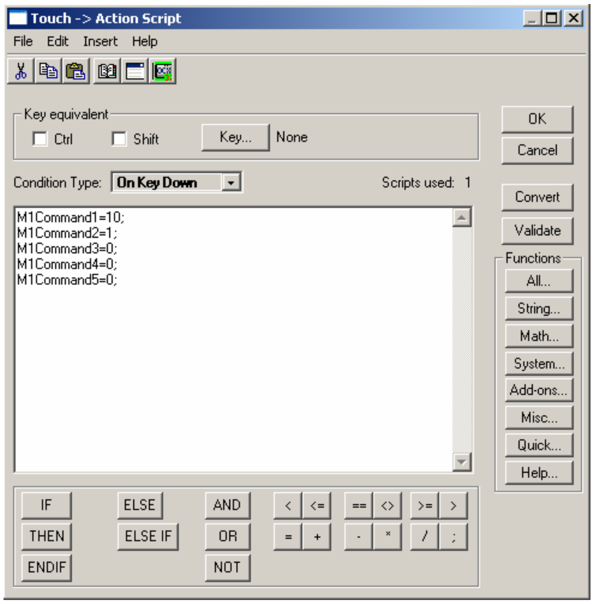
Expression: M1Queue1

Visible State: ● On ○ Off

When M1Queue1 is not equal to zero, there is a call queued for the host master so the button should be visible. Since M1Queue1 is the top of the queue, it is the only queue tag we need to check.

To set up the “End Current Call” button, go to its “Actions” window. The script below sends the DXL the “EndC” command to end the call between Master 1 and whatever Master 1 is connected to.

```
M1Command1=10;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```



Key equivalent: ☐ Ctrl ☐ Shift Key... None

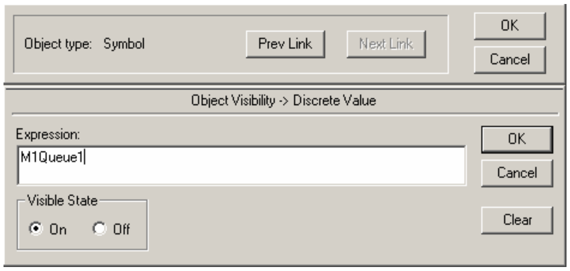
Condition Type: On Key Down

Scripts used: 1

```
M1Command1=10;
M1Command2=1;
M1Command3=0;
M1Command4=0;
M1Command5=0;
```

We can make the “End Current Call” button disappear if there is no call to end. Select “Miscellaneous: Visibility” from the “End Current Call” buttons properties window and enter the following expression:

```
M1Queue1
```



Object type: Symbol

Expression: M1Queue1

Visible State: ● On ○ Off

M1Status1 will be equal to zero if the host master does not have a call in progress, and the above expression will evaluate to zero, and the button will disappear.

6.6.4 Playing a sound when call requests are active

There are two ways to play a sound when there are calls in the intercom queue.

One would be to periodically sound a reminder tone every few seconds until all calls are answered.

The other method would be to play a sound when a new call comes into the queue. Methods to do each of the above are shown below. That method is already included in the scripts above, but you can remove the PlaySound command to not play a sound when a call request comes in.

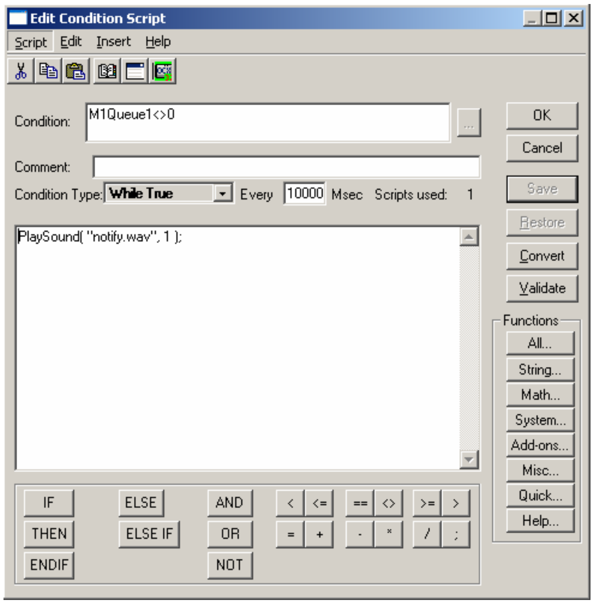
You can have a sound play when a new call comes into the queue and also have a periodic reminder tone by using both methods.

6.6.4.1 *Playing a periodic sound when calls are in the queue*

To play a periodic reminder tone, you can use the queue registers. When there is a call in the queue, the M1Status1 queue register will be non-zero.

You can create a condition script which will act periodically whenever M1Status1 <> 0, and have it play a sound or do other commands. The condition will be “M1Queue1<>0”, the Condition Type should be “While TRUE”, and the period between tones would be in the Every Msec box (10000 = every 10.000 seconds). To play a notify sound using a sound from the Windows sound files, you can use the following script command

```
PlaySound(“notify.wav”,1);
```



Condition: M1Queue1<>0

Comment:

Condition Type: While True Every 10000 Msec Scripts used: 1

```
PlaySound( "notify.wav", 1 );
```

6.6.4.2 *Playing a sound when a new call enters the queue*

This is already done in the script made up earlier where the condition is:

```
(M1Read1<>M1Last1) OR (M1Read2<>M1Last2) OR
(M1Read3<>M1Last3) OR (M1Read4<>M1Last4) OR
(M1Read5<>M1Last5)
```

This portion of the script handles when new call request or call request cancel messages come in.

```
IF (M1Last1==1) OR (M1Last1==2) THEN
      {Call Request or Cancel Call Request}
      IF (M1Last2==1) THEN
            IF ((M1Last3 > 0) AND (M1Last3 <= 6))THEN
                  PointerTag.Name = "S" + Text(M1Last3, "#") + "Status";
                  IF (M1Last1==1) THEN
                        {Call Request}
                        PointerTag.02 = 1;
                        PlaySound("ringin.wav",1);
                  ELSE
                        {Cancel Call Request}
                        PointerTag.02 = 0;
                  ENDIF;
            ENDIF; {Stations 1-6}
      ENDIF; {Master 1}
ENDIF; {Call Request}
```

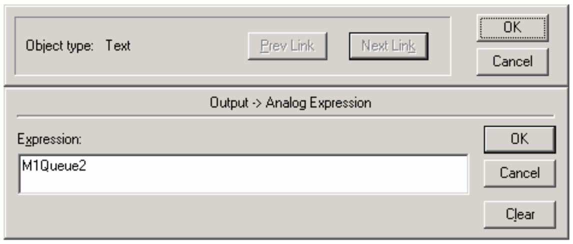
To change or remove the tone, change the “PlaySound(“ringin.wav”,1);” line in this script

6.6.5 Configuring Queue Text Display

Three different kinds of call requests can occur in a DXL system: a call request from a station, a call request from a master, and an audio level alarm. First, we will learn to display only the ID number of the device calling the master. After that we will look at a more complex example that displays the device type and ID number.

6.6.5.1 Simple Text Display

To display the station number in the call request queue list, double click on the "#" symbol in the list to open its properties window and then select "Value Display: Analog". M1Queue1 and M1Queue2 contain the device type and device ID, respectively, of the first call request in Master 1's queue. We therefore want to set the Expression to "M1Queue2", so its value will take place of the "#" symbol, as shown here.



Object type: Text

Expression: M1Queue2

```
M1Queue2
```

6.6.5.2 More Complex Text Display Using Strings and Scripts

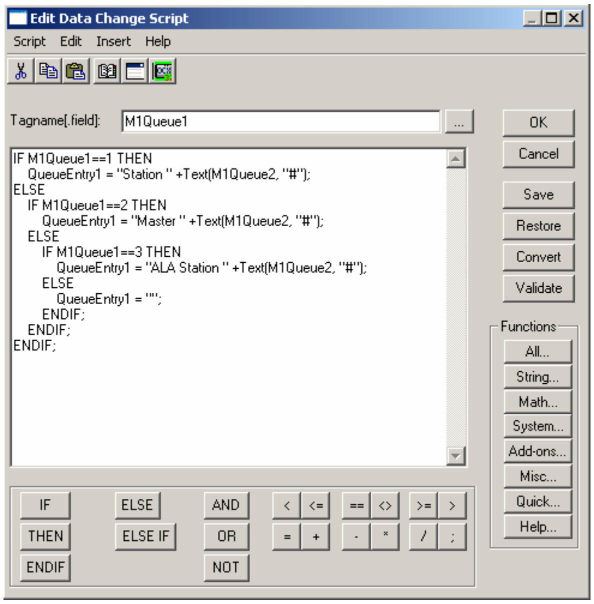
To make a more useful text display we need to define a few more tags to hold the text to be displayed, and then write some scripts (small programs) to manipulate those tags.

Define the following tags with a tag type of "Memory Message":

- QueueEntry1
- QueueEntry2
- QueueEntry3
- QueueEntry4
- QueueEntry5

These tags will hold the display text for each of the entries in the host masters queue. We will now write a set of scripts to automatically enter the proper text into these tags based on the content of the tags M1Queue1 to M1Queue10. Expand the "Scripts" item in the application explorer pane and right click on "Data Change". When any of the M1Queue tags change value, there has been some change in the call request queue and we need to evaluate what should be displayed.

The script shown here checks the type and ID number of the first call request in the queue, and puts an appropriate string of text into QueueEntry1. This script is executed when M1Queue1 changes value and an identical script should be created that executes when M1Queue2 changes value. The script checks the value of M1Queue1 to determine which prefix (“Station “, “Master “, or “ALA Station “) should be appended to the QueueEntry1 string, then appends the ID number of the device requesting a call (which is contained in M1Queue2). The Text() function used here takes a number from a register and converts it into a string that can be stored in the QueueEntry tag. Its syntax is “Text(<tag to be converted to string>, <number formatting>)”. Please see Wonderware’s scripting language documentation for more information.



Tagname[.field]: M1Queue1

```
IF M1Queue1==1 THEN
  QueueEntry1 = "Station " +Text(M1Queue2, "#");
ELSE
  IF M1Queue1==2 THEN
    QueueEntry1 = "Master " +Text(M1Queue2, "#");
  ELSE
    IF M1Queue1==3 THEN
      QueueEntry1 = "ALA Station " +Text(M1Queue2, "#");
    ELSE
      QueueEntry1 = "";
    ENDIF;
  ENDIF;
ENDIF;
```

```
IF M1Queue1==1 THEN
  QueueEntry1 = "Station " +Text(M1Queue2, "#");
ELSE
  IF M1Queue1==2 THEN
    QueueEntry1 = "Master " +Text(M1Queue2, "#");
  ELSE
    IF M1Queue1==3 THEN
      QueueEntry1 = "ALA Station " +Text(M1Queue2, "#");
    ELSE
      QueueEntry1 = "";
    ENDIF;
  ENDIF;
ENDIF;
```

For each pair of M1Queue tags representing a single QueueEntry tag, a similar script needs to be created. QueueEntry2 is updated when M1Queue3 or M1Queue4 change value, QueueEntry3 is updated when M1Queue5 or M1Queue6 change value, QueueEntry4 is updated when M1Queue7 or M1Queue8 change value, and QueueEntry5 is updated when M1Queue9 or M1Queue10 change value.