

Using Ethernet/IP to communicate between DVT and an SLC-5/05 System

Document ID:???

Original Author: Mateo Londono

Last Modified: 03/27/02

Last Editor: Mateo Londono

Abstract

This document describes how to use EtherNet/IP to transfer data between DVT systems (Series 600 and Series 500) and an SLC 5/05. The purpose of this document is to describe the configuration steps needed to use this feature in both an SLC and a DVT system. This document does NOT describe EtherNet/IP. For readers that are not familiar with Ethernet/IP, the references provide some useful sources of information.

Introduction

The SLC5/05 processor communicates using PCCC (Programmable Controller Communication Commands) application-level commands. To allow peer-to-peer communications between different families of Allen-Bradley processors, the ControlLogix, PLC5E and SLC5/05 added support for EtherNet/IP with embedded PCCC commands (“EIP/PCCC.”) While the ControlLogix system fully supports EIP with generic implicit and explicit messaging, the SLC5/05 only supports EIP explicit messages directed to the vendor specific PCCC object (class 0x67.)

Starting with FWK 2.5, DVT has added support for EIP/PCCC to allow direct communication with the SLC5/05 processor over EtherNet/IP.

SLC5/05 Firmware Versions

According to Allen-Bradley’s documentation, The **SLC5/05** was upgraded with EIP support in May 1999, with **series A** firmware revision **OS501, FRN5**. If the SLC5/05 is an earlier revision it can be Flash updated with new firmware. All **series B and C** SLC5/05’s support EIP.

Activation

In the DVT system, EtherNet/IP must be activated for communication with SLC5/05. For a procedure on how to activate EtherNet/IP see reference #1.

Configuration for EIP/PCCC Explicit Messaging -DVT

DVT's Configuration

The user interface for DVT's EIP/PCCC explicit messaging is the same as for the generic EIP support. It involves a simple model for transferring data. Special script function calls are used to read and write inspection data to and from reserved EIP registers. There are 4 blocks of EIP data, as follows:

Name	Description	Number Available	Range of valid Indexes	Size of Block
SINTS	8-bit Signed Integer	256	0-255	256 bytes
INTS	16-bit Signed Integer	128	0-127	256 bytes
DINTS	32-bit Signed Integer	64	0-63	256 bytes
REALS	32-bit Floating Point	64	0-63	256 bytes

Table 1: EIP Data blocks inside DVT systems.

The following DVT script functions allow the user to read and write from these blocks of memory. They can be found under the OEM node in the script editor tree.

```
AB_RegisterWriteSINT (index, value);  
AB_RegisterWriteINT (index, value);  
AB_RegisterWriteREAL (index, value);  
AB_RegisterWriteString(index, value); -Uses SINTS block
```

```
value = AB_RegisterReadSINT (index);  
value = AB_RegisterReadINT (index);  
value = AB_RegisterReadREAL (index);  
strvar = AB_RegisterReadString (index);-Uses SINTS block
```

Note: the ranges for index and value in the above function calls are determined by the data types and size of the data blocks. See Table 1.

It is important to understand that when these functions are executed data is updated **only** in within the DVT system. Communication with the SLC5/05 occurs when initiated by a MSG instruction from the PLC (See next section). This happens independently and asynchronously from the inspections.

The SLC5/05 uses files of different types to store data. Each data element in the file is accessed via a file number and an offset. For example, to access the third element on an integer file numbered 7, one would write "N7:3".

To implement EIP/PCCC feature, DVT has assigned 4 different file numbers to EIP data blocks to allow use of the above syntax to refer to each of their elements. One can think of DVT systems as having 4 SLC-style files with the following characteristics:

Name	Number	EIP Data Block	Num of elemets	Size (bytes)
DVT Integer File	7	INTS	128	256
DVT Float File	8	REALS	64	256
DVT ASCII File	9	SINTS	256	256
DVT String File	10	SINTS	256	256

Table 2: DVT's SLC-Style data files for use in EIP/PCCC messages.

SLC5/05 Configuration

In the SLC5/05, use a MSG instruction to read and write data from/to DVT systems. File numbers and types in the DVT system are fixed and must be specified in the MSG configuration as described above. In the SLC5/05, the user can configure most files and the numbers used are not fixed. For the purpose of this example let us assume that the following files exist in the SLC and will be used for communication with a DVT system.

File	Num of elemets
N7	8
F8	102
A9	23
ST15	5

Table 3: Example Data Files in an SLC5/05

The first step in the MSG configuration is to designate the type of target device to a PLC5. This tells the SLC to use Typed Read/Write commands. These commands are embedded inside EIP explicit messages in EIP/PCCC.

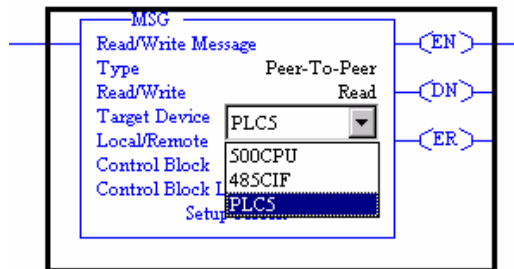


Figure 1: Selection of PLC5 as Target Device

Next, bring up the MSG setup screen and select **Channel 1** for Ethernet communications. Notice this adds a new field called **MultiHop** on the general tab. Also notice the **Message Timeout** field; it specifies the number of seconds to wait for a response after the MSG is sent before timing out with an error. This value depends on the application.

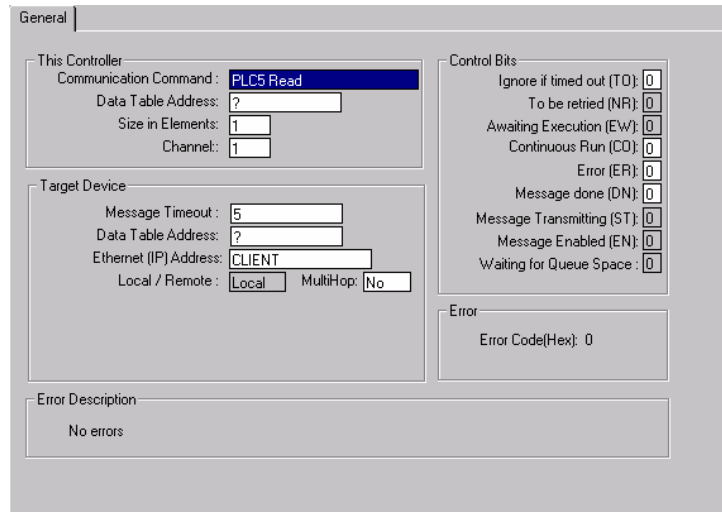


Figure 2: MSG configuration - channel selection

Since the SLC supports other protocols over Ethernet TCP/IP there are certain configuration parameters that must be specified in the MSG instruction to ensure that the SLC uses EIP/PCCC. To force the use of EIP/PCCC, choose **MultiHop=Yes** in the MSG setup. This brings up an additional tab in the MSG configuration dialog labeled MultiHop. The only thing to edit in this tab is the IP address on the first line. This IP address **MUST** correspond to the IP of the target DVT system.

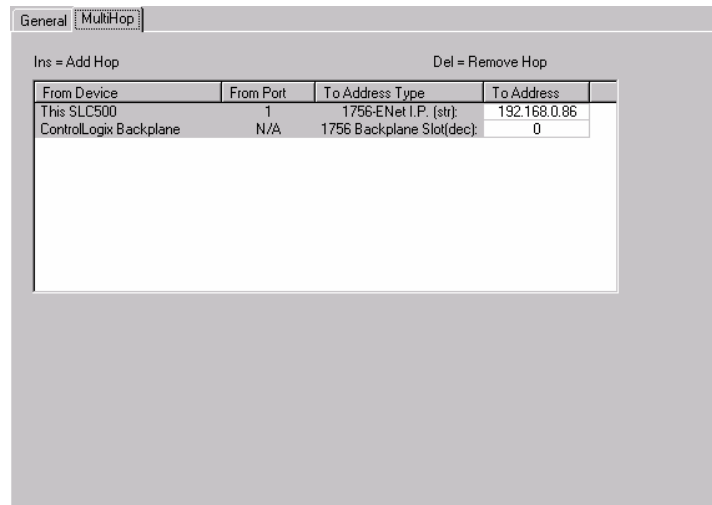


Figure 3: MSG configuration IP address

Finally, the destination and target **Data Table Address** plus the **Size in Elements** fields must be specified to complete the MSG instruction configuration. Here the SLC is expecting addresses of the form “N7:3 .” The following examples illustrate this aspect. Remember that the examples assume the SLC has the files listed in Table 3.

Explicit Messaging Example 1

Suppose you are performing a measurement with at DVT system and you need to send the value to a FLOAT in an SLC5/05.

The following example script writes the value from the inspections into the EIP registers for subsequent transfer to the SLC.

```
// Writes a measurement value at index 2 of the
// DVT FLOAT File #8. (equivalent to REALS Block)

AB_RegisterWriteReal (2, Hole.Radius);
```

In the SLC configure the Following MSG instruction:

The screenshot shows the 'MSG' configuration dialog box. The 'General' tab is selected, and the 'MultiHop' option is checked. The 'This Controller' section has 'Communication Command' set to 'PLC5 Read', 'Data Table Address' set to 'F8:2', 'Size in Elements' set to '2', and 'Channel' set to '1'. The 'Target Device' section has 'Message Timeout' set to '23', 'Data Table Address' set to 'F8:2', 'Local / Remote' set to 'Local', and 'MultiHop' set to 'Yes'. The 'Control Bits' section has several checkboxes, all of which are currently unchecked. The 'Error' section shows 'Error Code(Hex): 0'. The 'Error Description' field at the bottom contains the text 'No errors'.

Figure 4: MSG configuration for example 1

In the **This Controller** area, the value of “F8:2” for the **Data Table Address** field tells the SLC that data from the MSG is to be placed in float 2 of float file #8.

In the **Target Device** area, the value of “F8:2” for the **Data Table Address** field tells the DVT system that DVT Float File #8 is to be accessed starting at index 2. Remember that the DVT Float File has 64 elements (0-63.)

Explicit Messaging Example 2

Suppose you are reading a 2D Data matrix code with at DVT system and you need to send the code into a string in an SLC5/05.

The following example script writes the code from the inspections into the EIP registers for subsequent transfer to the SLC.

```
// Puts DataMatrix Code at index 2 of the  
// DVT String File #10. (equivalent to SINTS Block)  
  
AB_RegisterWriteString (2, Reader.String);
```

In the SLC configure the Following MSG instruction:

The screenshot shows the configuration for a Message (MSG) instruction. The 'General' tab is selected. The 'This Controller' section is configured with 'Communication Command' set to 'PLC5 Read', 'Data Table Address' set to 'ST15:0', 'Size in Elements' set to '1', and 'Channel' set to '1'. The 'Target Device' section is configured with 'Message Timeout' set to '23', 'Data Table Address' set to 'ST10:2', and 'Local / Remote' set to 'Local' and 'MultiHop' set to 'Yes'. The 'Control Bits' section shows various control bits set to '0'. The 'Error' section shows 'Error Code(Hex): 0'. The 'Error Description' section shows 'No errors'.

Figure 5: MSG configuration for example 2

In the **This Controller** area, the value of “ST15:0” for the **Data Table Address** field tells the SLC that string data from the MSG is to be placed in string 0 of string file #15. In the **Target Device** area, the value of “ST10:0” for the **Data Table Address** field tells the DVT system that DVT String File #10 is to be accessed starting at index 2. Notice how the offset into the DVT string file is treated differently than for the other types. In the case of strings this offset specifies the starting location of the string in the SINTS data block rather than the element index as for all other types. Notice that in the case of strings the SLC only allows transferring 1 at a time. Any other value for the **Size in Elements** field will result in an error. There is a size limitation of 82 characters on strings imposed by the SLC.

Explicit Messaging Example 3

Suppose you need to send data to a DVT system from an SLC5/05, specifically a single integer. This situation may arise in application where the operator must change a parameter from time to time w/o the need to use framework. A good example might be the maximum number of defects allowed. Once this value is transferred to the EIP data blocks in the DVT system, it can be accessed by background and foreground scripts and used to make PASS/FAIL decisions.

In the SLC configure the Following MSG instruction:

The screenshot shows the 'MSG Configuration' dialog box with the following settings:

- General / MultiHop** (selected tab)
- This Controller:**
 - Communication Command: PLC5 Write
 - Data Table Address: N7:2
 - Size in Elements: 2
 - Channel: 1
- Target Device:**
 - Message Timeout: 23
 - Data Table Address: N7:2
 - Local / Remote: Local (selected)
 - MultiHop: Yes (selected)
- Control Bits:**
 - Ignore if timed out (TO): 0
 - To be retried (NR): 0
 - Awaiting Execution (EW): 0
 - Continuous Run (CO): 0
 - Error (ER): 0
 - Message done (DN): 0
 - Message Transmitting (ST): 0
 - Message Enabled (EN): 0
 - Waiting for Queue Space: 0
- Error:**
 - Error Code(Hex): 0
- Error Description:**
 - No errors

Figure 6: MSG Configuration for example 3.

In the **This Controller** area, the value of “N7:2” for the **Data Table Address** field tells the SLC that the data to be sent must come from integer 2 of integer file #7.

In the **Target Device** area, the value of “N7:2” for the **Data Table Address** field tells the DVT system that the incoming data is to be placed in DVT Integer File #7 at index 2.

Remember that the DVT Integer File has 128 elements (0-127.)

The following script reads a value from the EIP registers (placed there during the last EIP/PCCC message from the SLC.)

```
// Reads value at index 2 of the DVT integer File #7.  
// (equivalent to INTS Block.)
```

```
int NumDefects;  
NumDefects = AB_RegisterReadINT (2);
```

```
// Use Numdefects to make PASS/FAIL Decisions
```

Sequence of Events For Examples 1 and 2

The correct sequence of events is:

1. With inspections running the script containing the AB_RegisterWrite() functions has to be executed at least once. This places the most recent inspection data in the DVT data blocks (data from the blocks can also be loaded into local script variables.) Note that if the system is triggered w/o inspections running, the AB_RegisterWrite() functions will not execute! This is because they are treated as an output of the system.
2. A MSG instruction from a SLC5/05 is sent to the DVT system. This instruction will then exchange data with the DVT data blocks that contain the latest information updated from the last inspection/

Typical Operation:

1. PLC triggers the DVT system with a Digital Output.
2. PLC monitors outputs from the DVT system to know when the inspection has taken place and if the result is PASS.
3. A MSG instruction is then enabled to exchange data between the two systems.

Sequence of Events For Example 3

The correct sequence of events is:

3. A MSG instruction from a SLC5/05 is sent to the DVT system. This instruction will update the data in the DVT EIP data blocks.
4. An inspection occurs. A script within the inspection reads from the EIP data blocks and uses the data to make decisions.

Typical Operation:

1. PLC sends MSG to DVT system.
2. PLC triggers the DVT system with a Digital Output.
3. PLC monitors outputs from the DVT system to know when the inspection has taken place and if the result is PASS.

References

1. DVT EtherNet/IP Introduction (To be defined)
2. DVT Publication: “ Using Ethernet/IP to communicate between DVT and an ControlLogix System.”
3. Rockwell Automation Publication “Communicating with RA Products Using EtherNet/IP Explicit Messaging.”
4. Secondary information sources are available from: <http://www.ethernert-ip.org>.