ProTalk

PTQ-101M

Quantum Platform

IEC 60870-5-101 Master Communication Module

User Manual

May 14, 2008

ProSoft TECHNOLOGY

Please Read This Notice

Successful application of this module requires a reasonable working knowledge of the Schneider Electric Quantum hardware, the PTQ-101M Module and the application in which the combination is to be used. For this reason, it is important that those responsible for implementation satisfy themselves that the combination will meet the needs of the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

This manual is provided to assist the user. Every attempt has been made to ensure that the information provided is accurate and a true reflection of the product's installation requirements. In order to ensure a complete understanding of the operation of the product, the user should read all applicable Schneider Electric documentation on the operation of the Schneider Electric hardware.

Under no conditions will ProSoft Technology be responsible or liable for indirect or consequential damages resulting from the use or application of the product.

Reproduction of the contents of this manual, in whole or in part, without written permission from ProSoft Technology is prohibited.

Information in this manual is subject to change without notice and does not represent a commitment on the part of ProSoft Technology Improvements and/or changes in this manual or the product may be made at any time. These changes will be made periodically to correct technical inaccuracies or typographical errors.

PTQ Installation and Operating Instructions

The statement "power, input and output (I/O) wiring must be in accordance with Class I, Division 2 wiring methods Article 501-10(b) of the National Electrical Code, NFPA 70 for installations in the U.S., or as specified in section 18-1J2 of the Canadian Electrical Code for installations within Canada and in accordance with the authority having jurisdiction".

The following or equivalent warnings shall be included:

A Warning - Explosion Hazard - Substitution of components may Impair Suitability for Class I, Division 2;

B Warning - Explosion Hazard - When in Hazardous Locations, Turn off Power before replacing Wiring Modules, and

C Warning - Explosion Hazard - Do not Disconnect Equipment unless Power has been switched Off or the Area is known to be Nonhazardous.

D Caution: The Cell used in this Device may Present a Fire or Chemical Burn Hazard if Mistreated. Do not Disassemble, Heat above 100°C (212°F) or Incinerate.

Important Notice:

CAUTION: THE CELL USED IN THIS DEVICE MAY PRESENT A FIRE OR CHEMICAL BURN HAZARD IF MISTREATED. DO NOT DISASSEMBLE, HEAT ABOVE 100°C (212°F) OR INCINERATE.

Maximum battery load = 200 µA.

Maximum battery charge voltage = 3.4 VDC.

Maximum battery charge current = 500 µA.

Maximum battery discharge current = 30 µA.

Your Feedback Please

We always want you to feel that you made the right decision to use our products. If you have suggestions, comments, compliments or complaints about the product, documentation or support, please write or call us.

ProSoft Technology
1675 Chester Avenue, Fourth Floor
Bakersfield, CA 93301
+1 (661) 716-5100
+1 (661) 716-5101 (Fax)
http://www.prosoft-technology.com

PTQ-101M User Manual
May 14, 2008
PSFT.101M.PTQ.UM.08.05.14

Contents

Guide to the PTQ-101M User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	Start Here (page 11)	This Section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Verify Communication, Diagnostic and Troubleshooting	→	Verifying Communication (page 109) Diagnostics and Troubleshooting (page 93)	This section describes how to verify communications with the network. Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview Glossary	→	Reference (page 111) Functional Overview (page 113) Product Specifications (page 111)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	Support, Service and Warranty (page 169)	This section contains Support, Service and Warranty information. Index of chapters.

1 Start Here

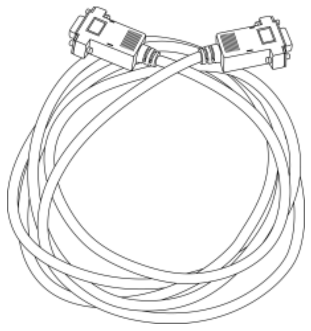
In This Chapter

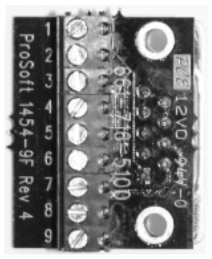
This guide is intended to guide you through the ProTalk module setup process, from removing the module from the box to exchanging data with the processor. In doing this, you will learn how to:

- Set up the processor environment for the PTQ module
- View how the PTQ module exchanges data with the processor
- Edit and download configuration files from your PC to the PTQ module
- Monitor the operation of the PTQ module

1.1 Hardware and Software Requirements

1.1.1 ProTalk Module Carton Contents

	
ProTalk Module	Null Modem Serial Cable

	
1454-9F DB-9 Female to 9 Pos Screw Terminal adapter (Serial protocol modules only)	ProSoft Solutions CD

Note: The DB-9 Female to 5 Pos Screw Terminal adapter is not required on Ethernet modules and is therefore not included in the carton with these types of modules.

1.1.2 Quantum / Unity Hardware

This guide assumes that you are familiar with the installation and setup of the Quantum / Unity hardware. The following should be installed, configured and powered up before proceeding:

- Quantum or Unity Processor
- Quantum rack
- Quantum power supply
- Quantum Modbus Plus Network Option Module (NOM Module) (optional)
- Quantum to PC programming hardware
- NOM Ethernet or Serial connection to PC

1.1.3 PC and PC Software

- Windows-based PC with at least one COM port
- Quantum programming software installed on machine
  or
- Concept™ PLC Programming Software version 2.6
  or
  ProWORX PLC Programming Software
  or
  UnityPro XL PLC Programming Software
- HyperTerminal (used in this guide) This is a communication program that is included with Microsoft Windows. You can normally find it in **Start / Programs / accessories / Communications**.

Note: ProTalk modules are compatible with common Quantum / Unity programming applications, including Concept and UnityPro XL. For all other programming applications, please contact technical support.

2 Configuring the Processor with Concept

In This Chapter

The following steps are designed to ensure that the processor is able to transfer data successfully with the PTQ module. As part of this procedure, you will use Concept configuration software from Schneider Electric to create a project, add the PTQ module to the project, set up data memory for the project, and then download the project to the processor.

Important Note: Concept software does not report whether the PTQ module is present in the rack, and therefore is not able to report the health status of the module when the module is online with the Quantum processor. Please take this into account when monitoring the status of the PTQ module.

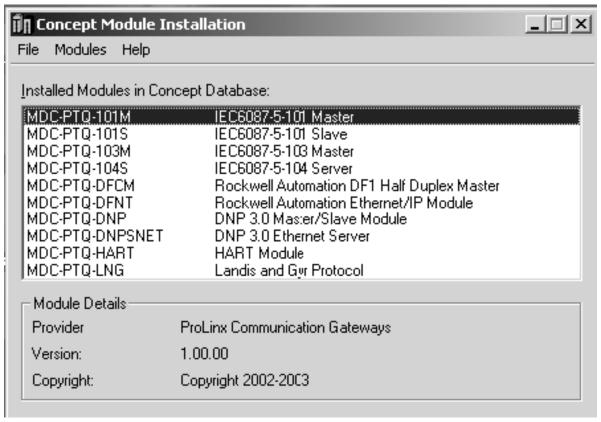
2.1 Information for Concept Version 2.6 Users

This guide uses Concept PLC Programming Software version 2.6 to configure the Quantum PLC. The ProTalk installation CD includes MDC module configuration files that help document the PTQ installation. Although not required, these files should be installed before proceeding to the next section.

2.1.1 Installing MDC Configuration Files

1 From a PC with Concept 2.6 installed, choose **Start / Programs / Concept / ModConnect Tool**.

This action opens the Concept Module Installation dialog box.



Installed Modules in Concept Database:

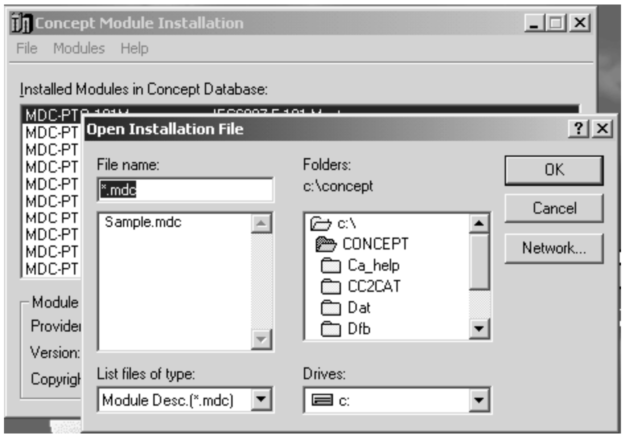
MDC-PTQ-101M	IEC6087-5-101 Master
MDC-PTQ-101S	IEC6087-5-101 Slave
MDC-PTQ-103M	IEC6087-5-103 Master
MDC-PTQ-104S	IEC6087-5-104 Server
MDC-PTQ-DFCM	Rockwell Automation DF1 Half Duplex Master
MDC-PTQ-DFNT	Rockwell Automation Ethernet/IP Module
MDC-PTQ-DNP	DNP 3.0 Master/Slave Module
MDC-PTQ-DNPSNET	DNP 3.0 Ethernet Server
MDC-PTQ-HART	HART Module
MDC-PTQ-LNG	Landis and Gyr Protocol

Module Details

Provider	ProLinx Communication Gateways
Version:	1.00.00
Copyright:	Copyright 2002-2003

2 Choose **File / Open Installation File**.

This action opens the Open Installation File dialog box:



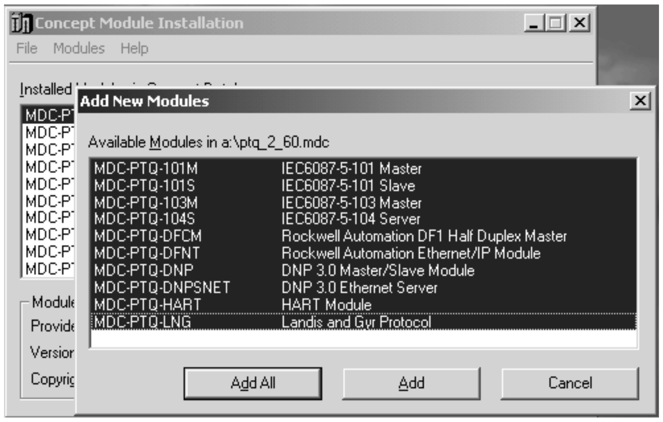
File name: *.mdc
Sample.mdc
Folders: c:\concept
c:\
CONCEPT
Ca_help
CC2CAT
Dat
Dfb
List files of type: Module Desc.(*.mdc)
Drives: c:

3 If you are using a Quantum processor, you will need the MDC files. In the Open Installation File dialog box, navigate to the **MDC Files** directory on the ProTalk CD.

4 Choose the MDC file and help file for your version of Concept:

- Concept 2.6 users: select PTQ_2_60.mdc and PTQMDC.hlp
- Concept 2.5 users: select PTQ_2_50.mdc and PTQMDC.hlp.

Select the files that go with the Concept version you are using, and then click **OK**. This action opens the add New Modules dialog box.



Add New Modules

Available Modules in a:\ptq_2_60.mdc

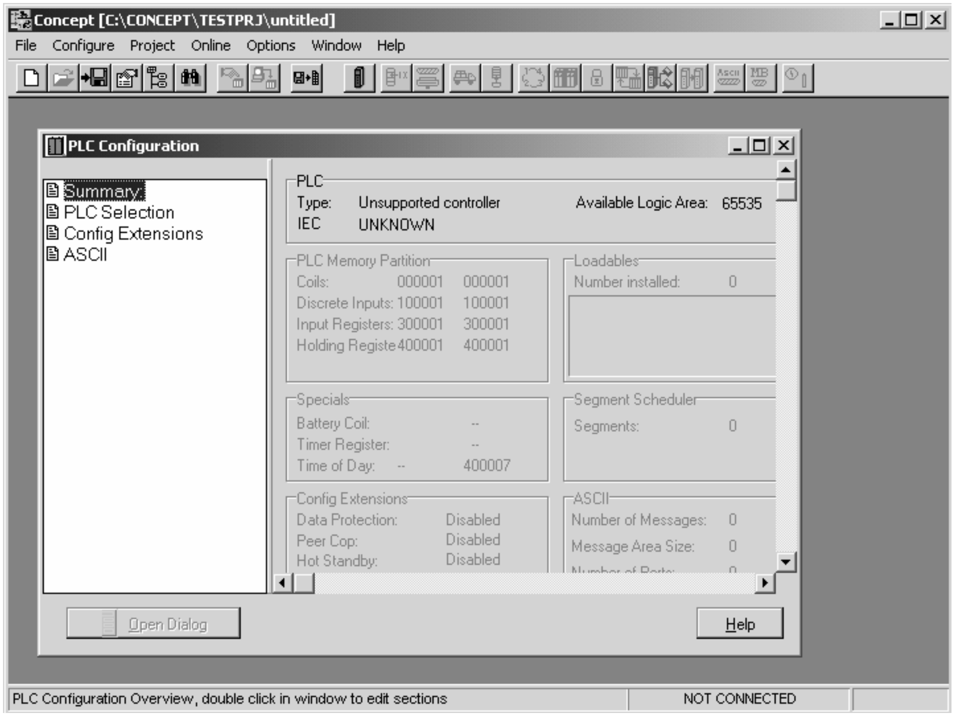
MDC-PTQ-101M	IEC6087-5-101 Master
MDC-PTQ-101S	IEC6087-5-101 Slave
MDC-PTQ-103M	IEC6087-5-103 Master
MDC-PTQ-104S	IEC6087-5-104 Server
MDC-PTQ-DFCM	Rockwell Automation DF1 Half Duplex Master
MDC-PTQ-DFNT	Rockwell Automation Ethernet/IP Module
MDC-PTQ-DNP	DNP 3.0 Master/Slave Module
MDC-PTQ-DNPSNET	DNP 3.0 Ethernet Server
MDC-PTQ-HART	HART Module
MDC-PTQ-LNG	Landis and Gyr Protocol

5 Click the **add all** button. A series of message boxes may appear during this process. Click **Yes** or **OK** for each message that appears.
6 When the process is complete, open the File menu and choose Exit to save your changes.

2.2 Create a New Project

This phase of the setup procedure must be performed on a computer that has the Concept configuration software installed.

1 From your computer, choose **Start / Programs / Concept V2.6 XL.EN / Concept**. This action opens the Concept window.
2 Open the File menu, and then choose **New Project**. This action opens the PLC Configuration dialog box.



Concept [C:\CONCEPT\TESTPRJ\untitled]

PLC Configuration

Summary
PLC Selection
Config Extensions
ASCII

PLC
Type: Unsupported controller   Available Logic Area: 65535
IEC UNKNOWN

PLC Memory Partition
Coils: 000001 000001
Discrete Inputs: 100001 100001
Input Registers: 300001 300001
Holding Registe 400001 400001

Loadables
Number installed: 0

Specials
Battery Coil: --
Timer Register: --
Time of Day: -- 400007

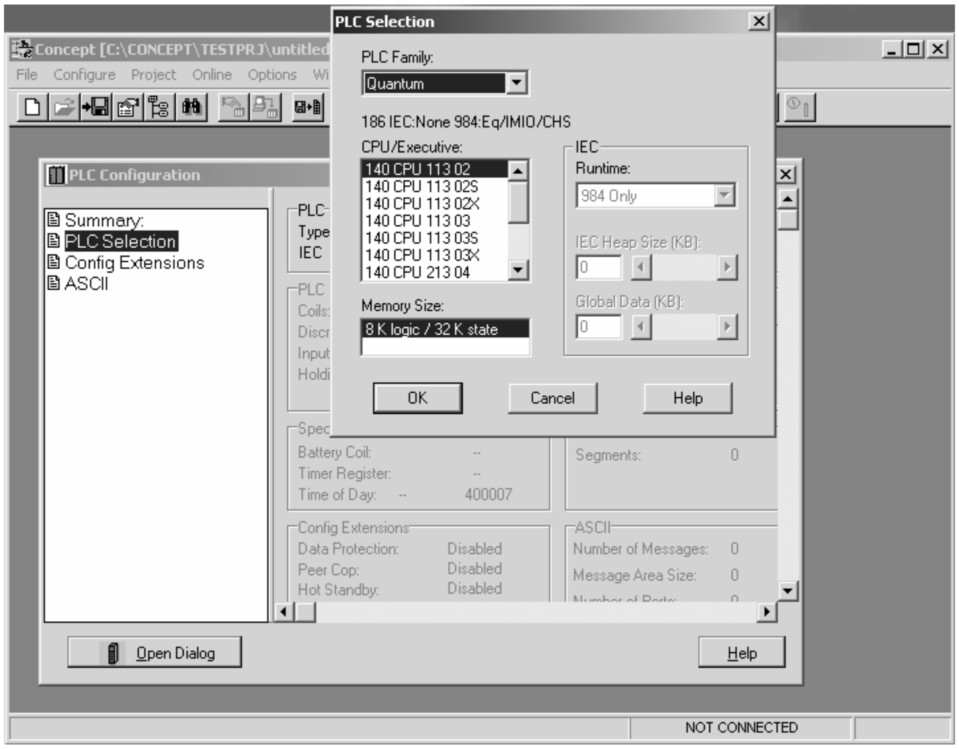
Segment Scheduler
Segments: 0

Config Extensions
Data Protection: Disabled
Peer Cop: Disabled
Hot Standby: Disabled

ASCII
Number of Messages: 0
Message Area Size: 0

PLC Configuration Overview, double click in window to edit sections
NOT CONNECTED

3 In the list of options on the left side of this dialog box, double-click the *PLC Selection* folder. This action opens the PLC Selection dialog box.



PLC Selection

PLC Family: Quantum

186 IEC:None 984:Eq/IMIO/CHS

CPU/Executive:
140 CPU 113 02
140 CPU 113 02S
140 CPU 113 02X
140 CPU 113 03
140 CPU 113 03S
140 CPU 113 03X
140 CPU 213 04

Memory Size: 8 K logic / 32 K state

IEC
Runtime: 984 Only
IEC Heap Size (KB): 0
Global Data (KB): 0

PLC Configuration: Summary, PLC Selection, Config Extensions, ASCII

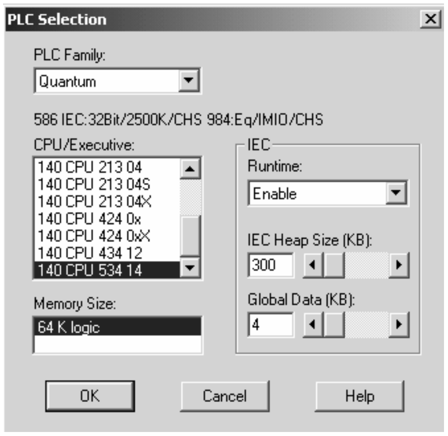
Battery Coil: --
Timer Register: --
Time of Day: -- 400007
Segments: 0

Config Extensions
Data Protection: Disabled
Peer Cop: Disabled
Hot Standby: Disabled

ASCII
Number of Messages: 0
Message Area Size: 0

NOT CONNECTED

4 In the *CPU/Executive* pane, use the scroll bar to locate and select the PLC to configure.



PLC Selection

PLC Family: Quantum

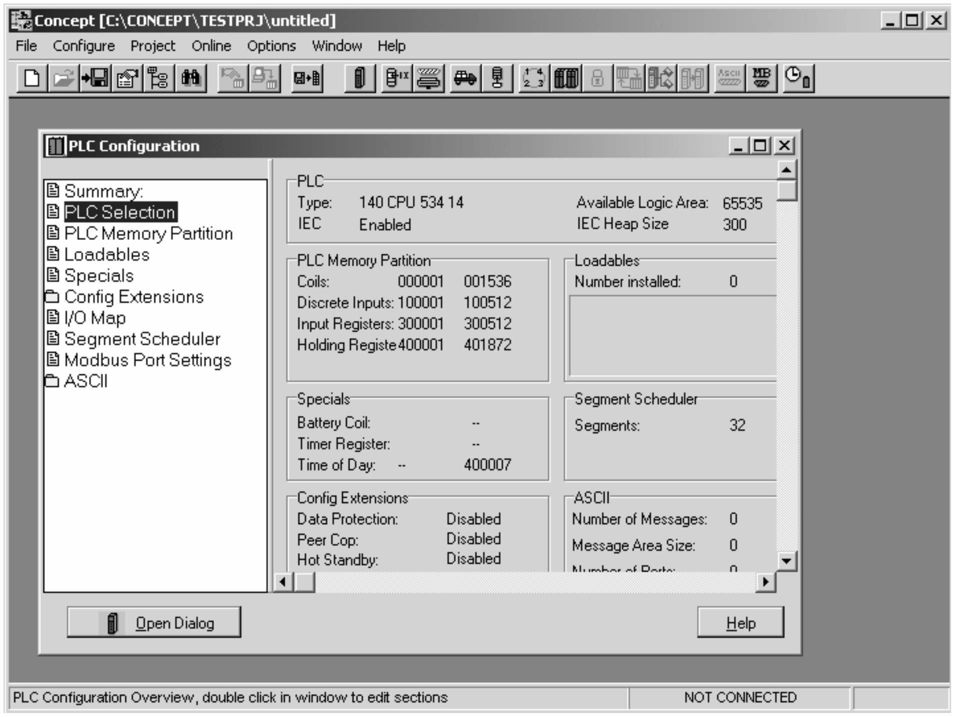
586 IEC:32Bit/2500K/CHS 984:Eq/IMIO/CHS

CPU/Executive:
140 CPU 213 04
140 CPU 213 04S
140 CPU 213 04X
140 CPU 424 0x
140 CPU 424 0xX
140 CPU 434 12
140 CPU 534 14

Memory Size: 64 K logic

IEC
Runtime: Enable
IEC Heap Size (KB): 300
Global Data (KB): 4

5 Click **OK**. This action opens the *PLC Configuration* dialog box, populated with the correct values for the PLC you selected.



PLC
Type: 140 CPU 534 14 Available Logic Area: 65535
IEC Enabled IEC Heap Size 300

PLC Memory Partition
Coils: 000001 001536
Discrete Inputs: 100001 100512
Input Registers: 300001 300512
Holding Registe 400001 401872

Loadables
Number installed: 0

Specials
Battery Coil: --
Timer Register: --
Time of Day: -- 400007

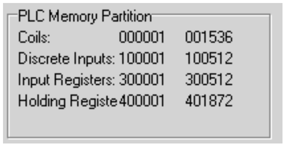
Segment Scheduler
Segments: 32

Config Extensions
Data Protection: Disabled
Peer Cop: Disabled
Hot Standby: Disabled

ASCII
Number of Messages: 0
Message Area Size: 0

PLC Configuration Overview, double click in window to edit sections NOT CONNECTED

6 Make a note of the holding registers for the module. You will need this information when you modify your application as outlined in the ProTalk application Reference Guides. The Holding Registers are displayed in the PLC Memory Partition pane of the PLC Configuration dialog box.

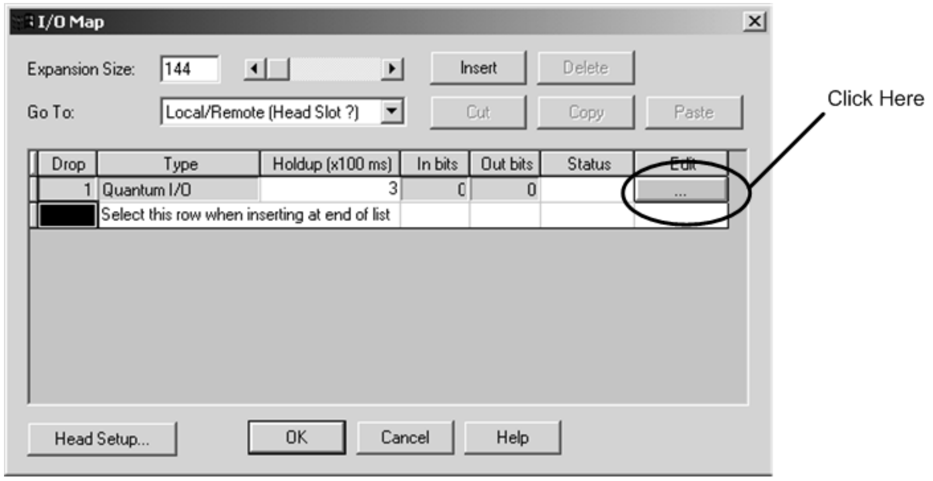


PLC Memory Partition
Coils: 000001 001536
Discrete Inputs: 100001 100512
Input Registers: 300001 300512
Holding Registe 400001 401872

2.3 Add the PTQ Module to the Project

The next step is to add one or more of the PTQ modules to the Project. To add modules:

1 In the list of options on the left side of the *PLC Configuration* dialog box, double-click *I/O Map*. This action opens the I/O Map dialog box.

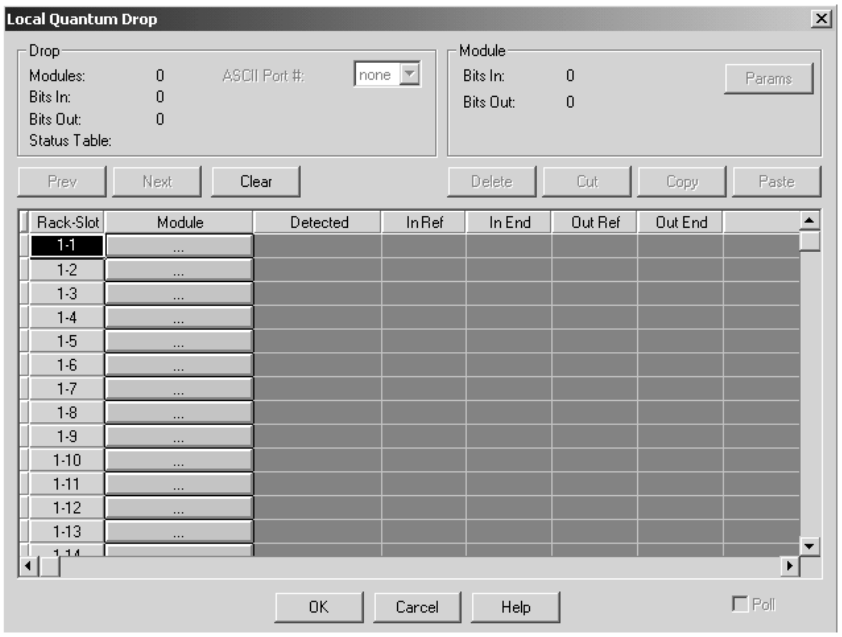


Expansion Size: 144

Go To: Local/Remote (Head Slot ?)

Drop	Type	Holdup (x100 ms)	In bits	Out bits	Status	Edit
1	Quantum I/O	3	0	0		...
	Select this row when inserting at end of list					

2 Click the **Edit** button to open the *Local Quantum Drop* dialog box. This dialog box is where you identify rack and slot locations.



Drop

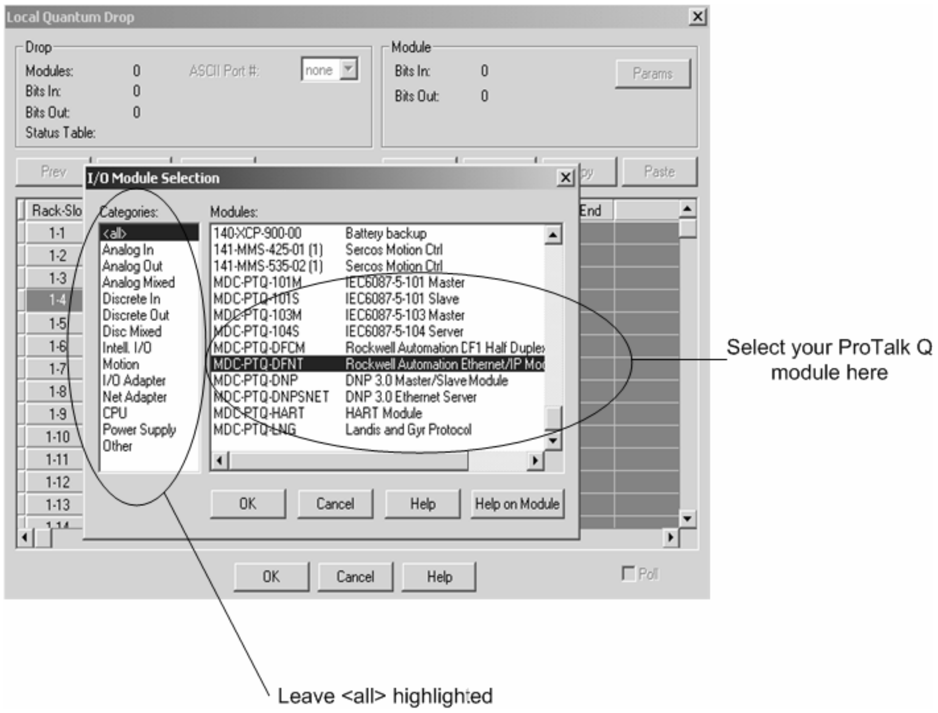
Modules: 0
Bits In: 0
Bits Out: 0
Status Table:
ASCII Port #: none

Module

Bits In: 0
Bits Out: 0

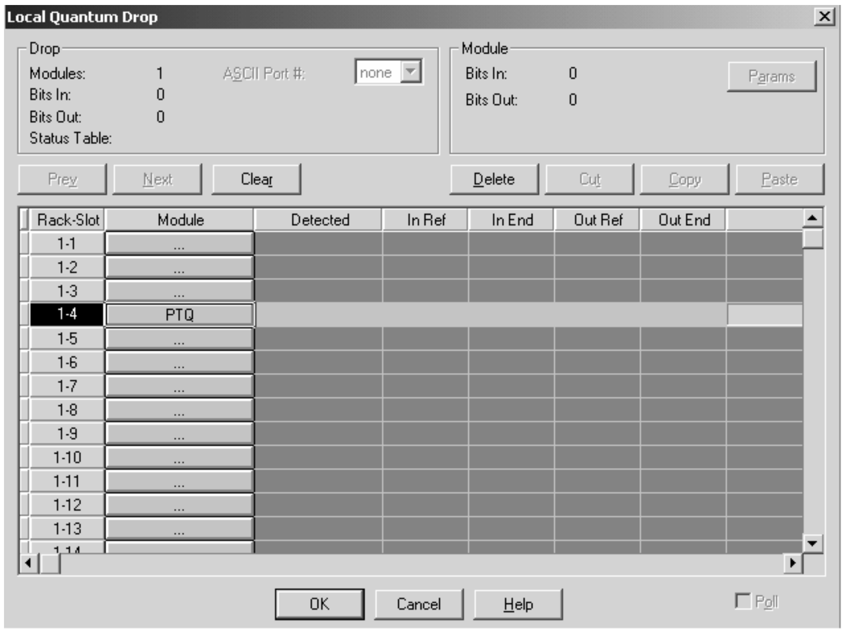
Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End
1-1	...					
1-2	...					
1-3	...					
1-4	...					
1-5	...					
1-6	...					
1-7	...					
1-8	...					
1-9	...					
1-10	...					
1-11	...					
1-12	...					
1-13	...					

3 Click the Module button next to the rack/slot position where the ProTalk module will be installed. This action opens the I/O Module Selection dialog box.



140-XCP-900-00	Battery backup
141-MMS-425-01 (1)	Sercos Motion Ctrl
141-MMS-535-02 (1)	Sercos Motion Ctrl
MDC-PTQ-101M	IEC6087-5-101 Master
MDC-PTQ-101S	IEC6087-5-101 Slave
MDC-PTQ-103M	IEC6087-5-103 Master
MDC-PTQ-104S	IEC6087-5-104 Server
MDC-PTQ-DFCM	Rockwell Automation DF1 Half Duplex
MDC-PTQ-DFNT	Rockwell Automation Ethernet/IP Mo
MDC-PTQ-DNP	DNP 3.0 Master/Slave Module
MDC-PTQ-DNPSNET	DNP 3.0 Ethernet Server
MDC-PTQ-HART	HART Module
MDC-PTQ-LNG	Landis and Gyr Protocol

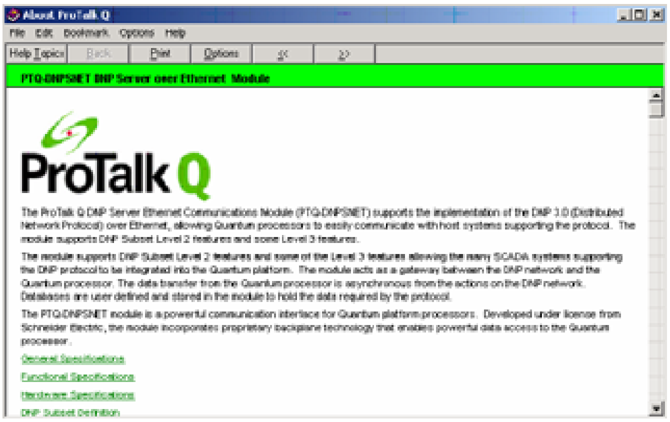
4 In the Modules pane, use the scroll bar to locate and select the ProTalk module, and then click OK. This action copies the description of the ProTalk module next to the assigned rack and slot number of the Local Quantum Drop dialog box.



Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End
1-1	...					
1-2	...					
1-3	...					
1-4	PTQ					
1-5	...					
1-6	...					
1-7	...					
1-8	...					
1-9	...					
1-10	...					
1-11	...					
1-12	...					
1-13	...					

5 Repeat steps 3 through 5 for each ProTalk module you plan to install. When you have finished installing your ProTalk modules, click OK to save your settings. Click Yes to confirm your settings.

Tip: Select a module, and then click the Help on Module button for help pages.


About ProTalk Q
File Edit Bookmark Options Help
Help Topics Back Print Options << >>
ProTalk Q

PTQ-DNPSNET DNP Server over Ethernet Module

The ProTalk Q DNP Server Ethernet Communications Module (PTQ-DNPSNET) supports the implementation of the DNP 3.0 (Distributed Network Protocol) over Ethernet, allowing Quantum processors to easily communicate with host systems supporting the protocol. The module supports DNP Subset Level 2 features and some Level 3 features.

The module supports DNP Subset Level 2 features and some of the Level 3 features allowing the many SCADA systems supporting the DNP protocol to be integrated into the Quantum platform. The module acts as a gateway between the DNP network and the Quantum processor. The data transfer from the Quantum processor is asynchronous from the actions on the DNP network. Databases are user defined and stored in the module to hold the data required by the protocol.

The PTQ-DNPSNET module is a powerful communication interface for Quantum platform processors. Developed under license from Schneider Electric, the module incorporates proprietary backplane technology that enables powerful data access to the Quantum processor.

General Specifications

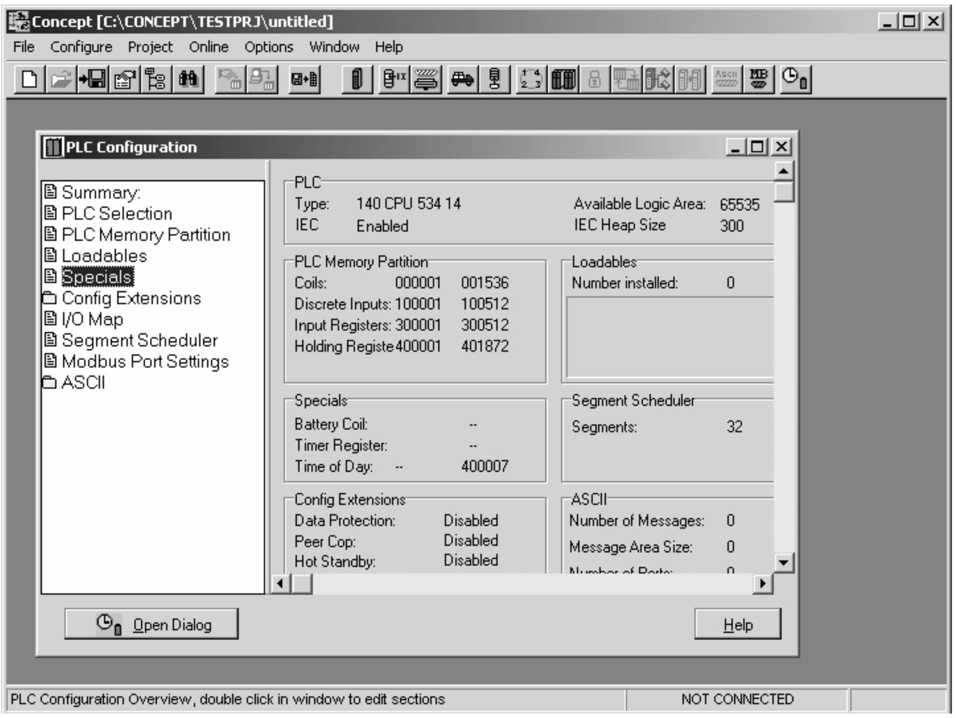
Functional Specifications

Hardware Specifications

DNP Subset Definition

2.4 Set up Data Memory in Project

1 In the list of options on the left side of the PLC Configuration dialog box, double-click Specials.



PLC
Type: 140 CPU 534 14 Available Logic Area: 65535
IEC Enabled IEC Heap Size 300

PLC Memory Partition
Coils: 000001 001536
Discrete Inputs: 100001 100512
Input Registers: 300001 300512
Holding Registe 400001 401872

Loadables
Number installed: 0

Specials
Battery Coil: --
Timer Register: --
Time of Day: -- 400007

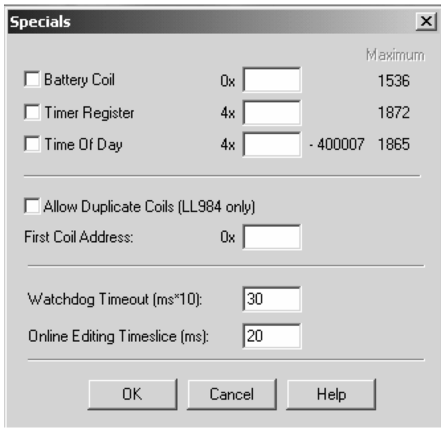
Segment Scheduler
Segments: 32

Config Extensions
Data Protection: Disabled
Peer Cop: Disabled
Hot Standby: Disabled

ASCII
Number of Messages: 0
Message Area Size: 0

PLC Configuration Overview, double click in window to edit sections NOT CONNECTED

2 This action opens the Specials dialog box.



Specials

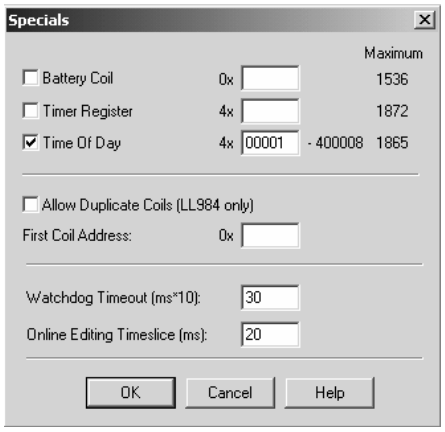
				Maximum
☐ Battery Coil	0x			1536
☐ Timer Register	4x			1872
☐ Time Of Day	4x		- 400007	1865

☐ Allow Duplicate Coils (LL984 only)
First Coil Address: 0x

Watchdog Timeout (ms*10): 30
Online Editing Timeslice (ms): 20

Selecting the Time of Day

1 Select (check) the Time of Day box, and then enter the value 00001 as shown in the following example. This value sets the first time of day register to 400001.



Specials

				Maximum
☐ Battery Coil	0x			1536
☐ Timer Register	4x			1872
☑ Time Of Day	4x	00001	- 400008	1865

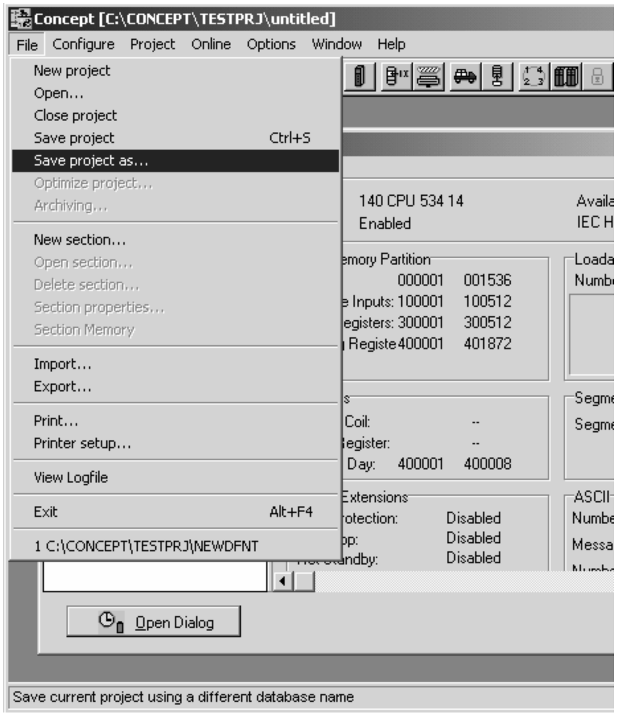
☐ Allow Duplicate Coils (LL984 only)
First Coil Address: 0x

Watchdog Timeout (ms*10): 30
Online Editing Timeslice (ms): 20

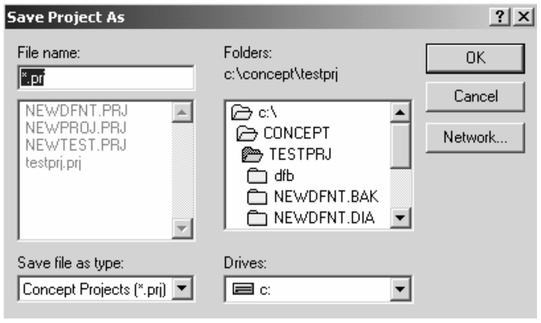
2 Click OK to save your settings and close the Specials dialog box.

Saving your project

1 In the PLC Configuration dialog box, choose File / Save project as.



2 This action opens the Save Project as dialog box.



3 Name the project, and then click OK to save the project to a file.

2.5 How to Set up and Use the Sample Function Block for Concept

2.5.1 EVENTFB Function Block Overview

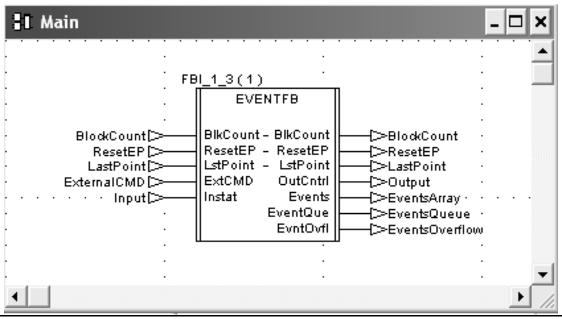
The purpose of the EVENTFB sample function block is to transfer the events into a buffer that consists of an array of elements that stores all data in a convenient format for the user. The block 9903 passes data into a compacted format thus occupying the minimum amount of registers. For example, the block 9903 originally reserves the same register for Hour and Minute (one byte for each value), so the user application would need to extract each value. The EVENTFB sample function block already extracts each event value into a separate register.

The following illustration shows the structure of each element of the buffer (extracted from the data type definition file).

```
TYPE EVENT101:
 STRUCT
  Session : WORD;       (* Session configured for this Master *)
  Sector : WORD;        (* Sector configured for this session *)
  COT  : WORD;          (* Cause of transmission of the event message *)
  Reserved : WORD;      (* Reserved*)
  PointIndex : ARRAY[0..1] OF WORD; (* This is the point index in remote device
that generated the event*)
  ASDU : WORD;          (* ASDU Type *)
  Milliseconds: UINT;   (* Timestamp - milliseconds *)
  Seconds: UINT;        (* Timestamp - Seconds *
  Minutes: BYTE;        (* Timestamp - minutes and hours *)
  Hours: BYTE           (* Timestamp - minutes and hours *)
  Month : BYTE;         (* This contains the month of the event occurred*)
  Day : BYTE;           (* This contains the day of the Event occurred*)
  Year: WORD ;          (* This contains the year the event occurred *)
  Qualifier: WORD;      (* Point qualifier, quality/sequence value see protocol
specification*)
  Value: ARRAY[0..1] OF WORD;    (* Data value - data size depends on ASDU type
*)
 END_STRUCT;
END_TYPE
```

The data structure that stores the incoming events consists on a circular buffer that can store up to 199 events. So the buffer consists on an array of 199 "EVENT101" elements presented previously. The element index can vary from 0 to 199. If the last event updated was located at index 199 then the next event will be copied to index 0.

The following illustration shows an instance example of the EVENTFB function block.



The EVENTFB function block contains the following PINs.

PIN	PIN Type	Data Type	Description
Instat	input	WORD64	Stores the memory area updated by block 9903. The start address must point to the same start address defined for block 9903 backplane data exchange (Point Address parameter).
ResetEP	input/output	INT	Move a value of one to reset the event pointer. This will cause the next event to be written to index 0 at the circular buffer. The register will be automatically reset to zero after the request was processed. This register should be only used for very specific applications (because the circular buffer automatically changes the element pointer from 199 to 0 after the maximum index was reached)
Events	Output	EVENTSTRUCT	Circular buffer that stores all received events in a convenient format for the user application. It can store up to 200 events (index varies from 0 to 199). After event 199 is updated the next event to be received will be automatically updated at index 0.
BlkCount	Input/Output	INT	Incremented after a block is received (and after the events in that block have been read into the circular buffer). The maximum value for this counter is 1000 (then it is automatically reset to 0)
LstPoint	Input/Output	INT	Pointer to the last event index read from the module. For example, if last event was updated at index 5 then this value will have the same value.
ExtCmd	Input	WORD	This external command is used so user can issue different commands while the module receiving events.
OutCntrl	Output	WORD64	Stores data to be sent from the processor to the module the start address it should match what you configured your backplane exchange to start.
EventQue	Output	WORD	Indicates how many events are in the queue to be read.
EvntOvfl	Output	WORD	This will be set to yes (1) if the overflow flag is set due to 199 events in the queue waiting to be read.

Before You Begin

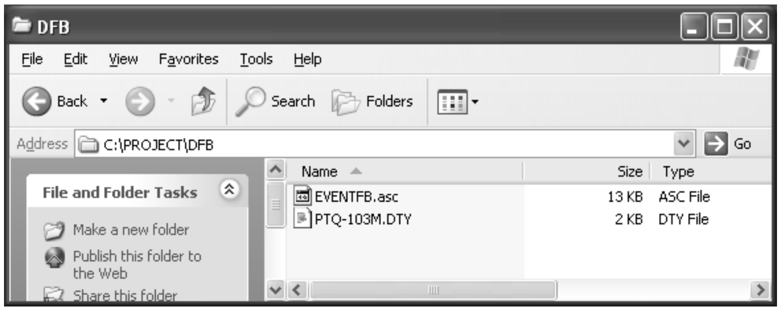
1 Make sure that your computer has the Concept Programming Unit installed.
2 The PTQ-101M firmware revision must support the event pass-thru functionality. This feature is available for version 1.12 or later. Refer to the "V" menu for the SOFTWARE REVISION LEVEL (page 97) value at the debug menu of the PTQ-101M module.
3 Using Windows Explorer create a folder for your Concept project with a "DFB" subfolder. This procedure will consider as an example the folder C:\PROJECT\DFB, where:
   - o C:\PROJECT- will store the main Concept project (.PRJ)

- C:\PROJECT\DFB - will store the data type definition file (PTQ-101M.DTY) and the function block that will be presented later at this document.

4 Refer to the CD-ROM or to the web site for the PTQ101MConcept_Block9903.zip file and extract the following files:

- EVENTFB.asc (function block)
- PTQ-101M.DTY (data type definition)

Use Windows Explorer to move these files to C:\PROJECT\DFB as shown in the following illustration.

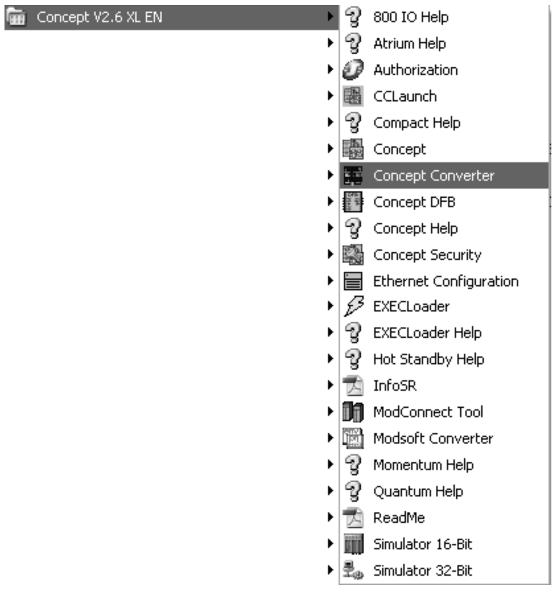


Name	Size	Type
EVENTFB.asc	13 KB	ASC File
PTQ-103M.DTY	2 KB	DTY File

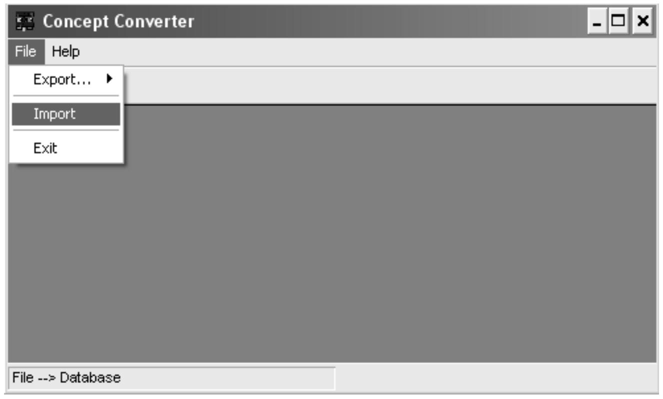
DFB
File Edit View Favorites Tools Help
Back Search Folders
Address C:\PROJECT\DFB Go
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder

Convert the EVENTFB Function Block

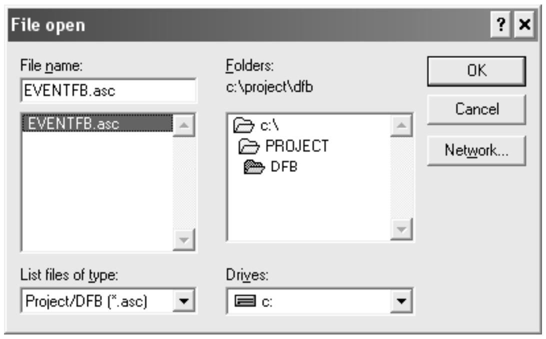
1 Start the Concept v2.6 XL EN - Concept Converter as shown in the following illustration.



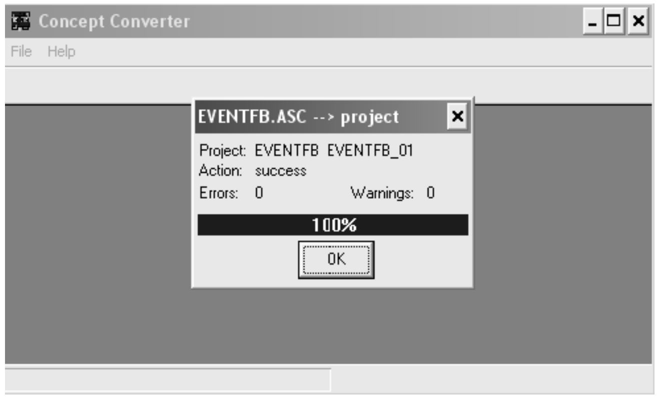
2 When the Concept Converter windows is displayed, open the File menu, and then choose Import



3 Select the EVENTFB.asc file located at C:\PROJECT\DFB as shown in the following illustration.



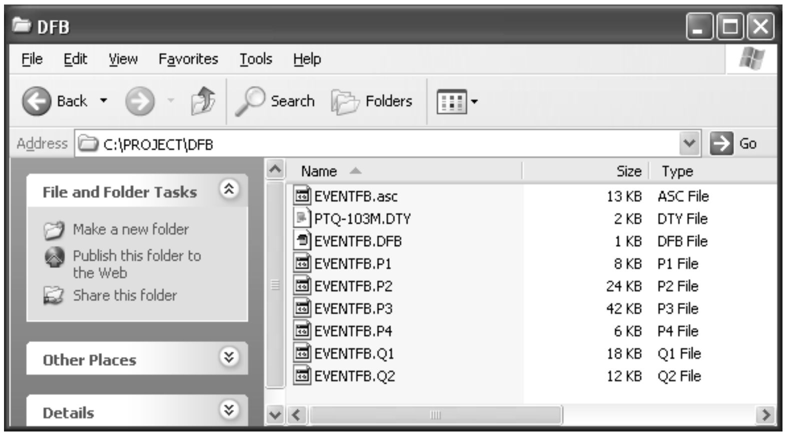
4 When the importing procedure is completed you will observe the following confirmation screen:


EVENTFB.ASC --> project

Project: EVENTFB EVENTFB_01
Action: success
Errors: 0 Warnings: 0

100%

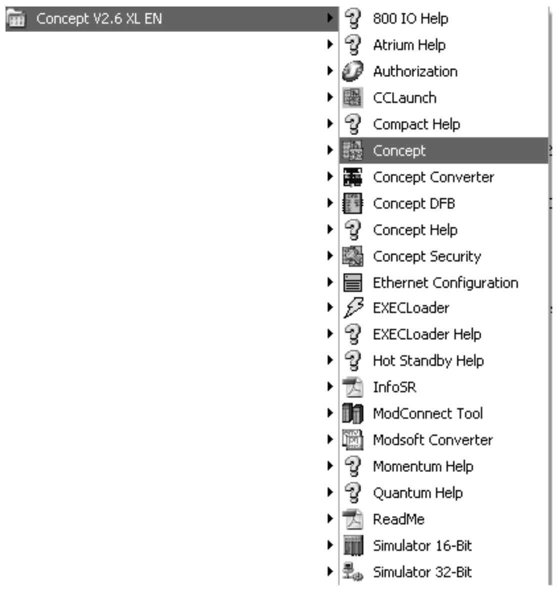
5 Close the Concept Converter tool. Now you can refer to C:\PROJECT\DFB to check that the function block (.DFB) was exported and is ready to be used.



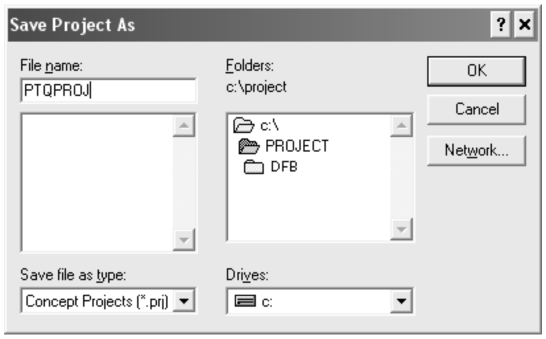
Name	Size	Type
EVENTFB.asc	13 KB	ASC File
PTQ-103M.DTY	2 KB	DTY File
EVENTFB.DFB	1 KB	DFB File
EVENTFB.P1	8 KB	P1 File
EVENTFB.P2	24 KB	P2 File
EVENTFB.P3	42 KB	P3 File
EVENTFB.P4	6 KB	P4 File
EVENTFB.Q1	18 KB	Q1 File
EVENTFB.Q2	12 KB	Q2 File

Setup the Concept Project

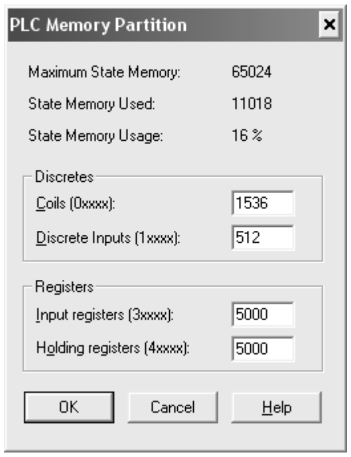
1 Start the Concept software as shown in the following illustration...



2 Create a new project and save it at the C:\PROJECT folder. For this example we will consider the project name as PTQPROJ.



3 At PLC Memory Partition make sure that the processor memory range is configured large enough for the PTQ-101M backplane usage.


PLC Memory Partition

Maximum State Memory:	65024
State Memory Used:	11018
State Memory Usage:	16 %

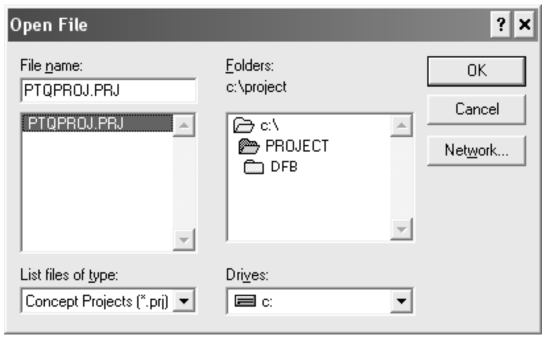
Discretes

Coils (0xxxx):	1536
Discrete Inputs (1xxxx):	512

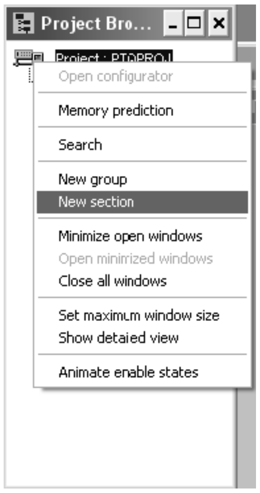
Registers

Input registers (3xxxx):	5000
Holding registers (4xxxx):	5000

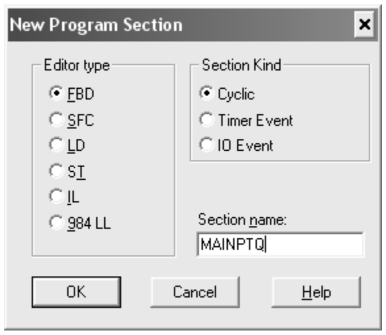
4 On the File menu, choose Close Project. Open the File menu again and then choose Open to open the PTQPROJ file again. This step allows the Concept application to recognize the new data types defined at the PTQ-101M.DTY file.



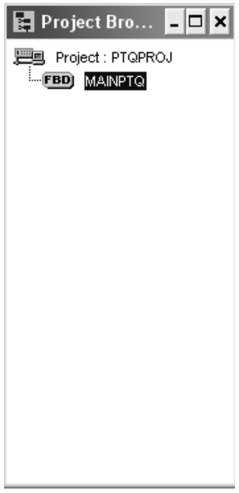
5 Select Project Browser. Select Project: PTQPROJ and click the right mouse button to open a shortcut menu. On the shortcut menu, choose New Section



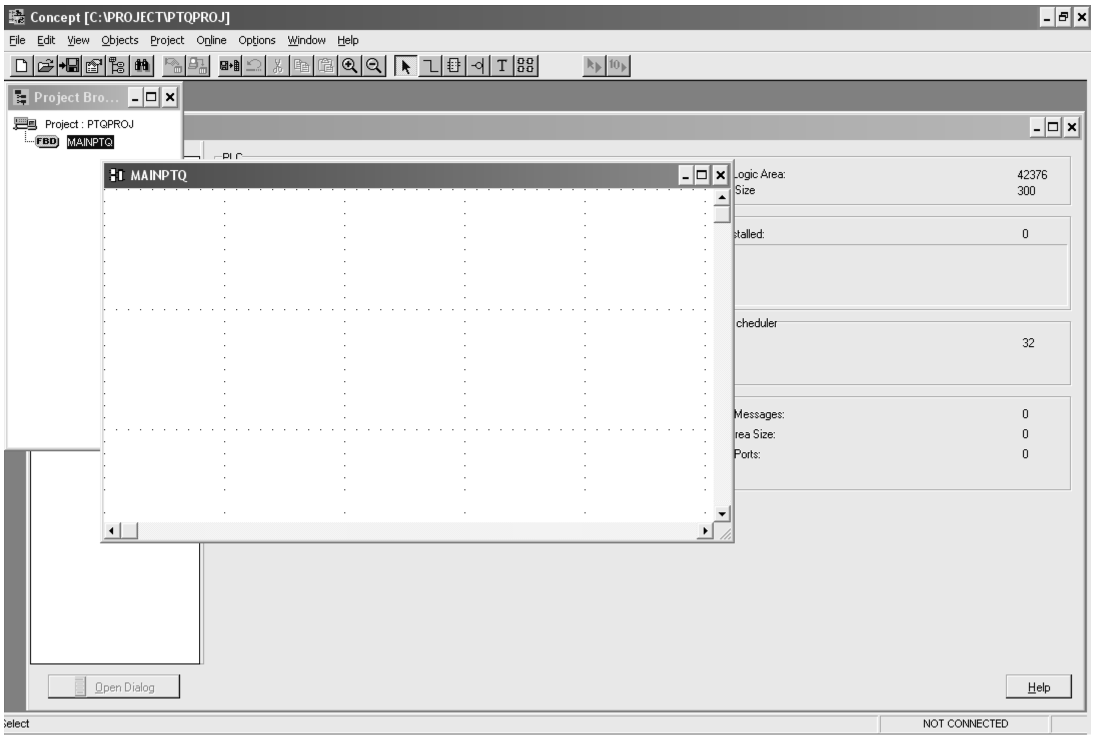
6 Select FBD. The procedure will refer to this section as MAINPTQ. Click OK



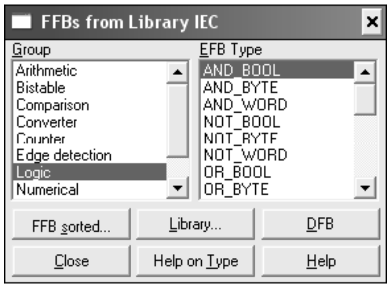
Editor type: FBD
Section Kind: Cyclic
Section name: MAINPTQ



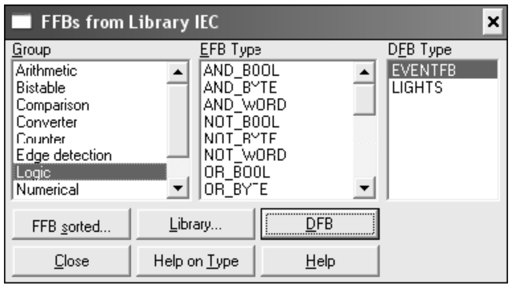
7 Double-Click the section to display the FBD section:



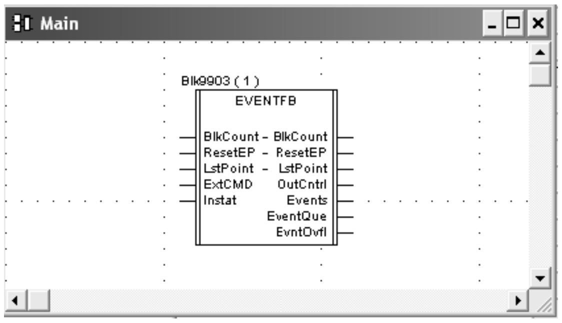
8 Select Objects-FFB Selection…



9 Click the DFB button and select the EVENTFB function block shown in the following illustration... Then close the window.



Now you should see the EVENTFB function block at the FBD section:



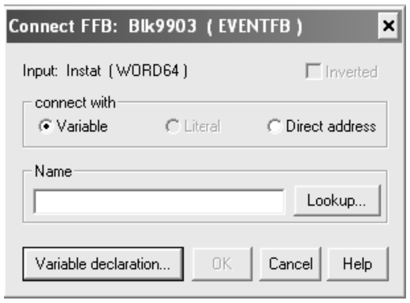
This step will create variables to be associated to the function block PINs. We will start with the Instat PIN. The variable for this PIN must point to the same start address where block 9903 will be copied to. For this example we are considering the following configuration for block 9903:

```
3x Register Start    :  1       #3x start register where data moved from module
to processor (1 to n)
```

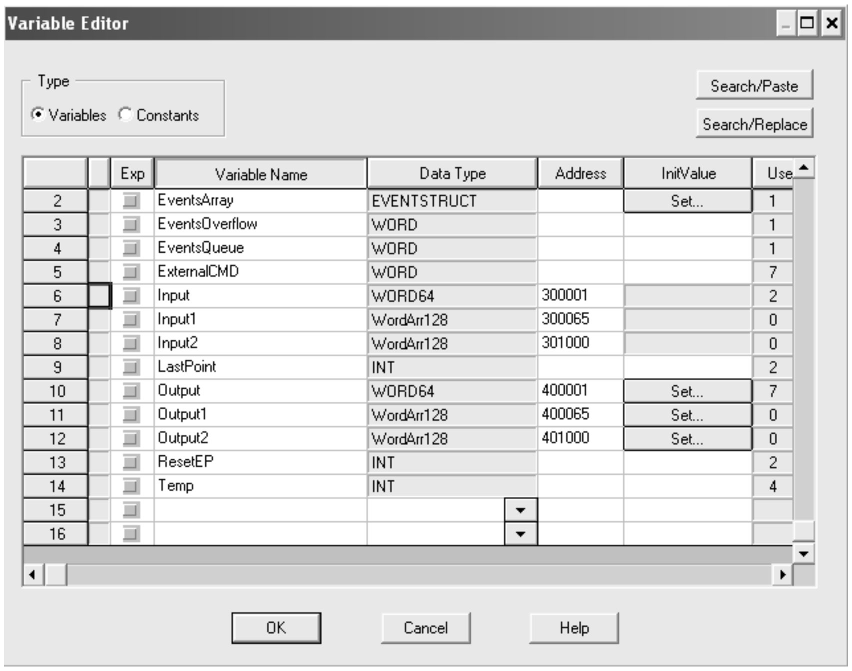
This implies that the variable associated to PIN Instat must also start at the same register address (300001 for this example).
As the Instat PIN will start same as 3x
The variable that associated with PIN OutCntrl must start at the same register address 400001

```
4x Register Start    :  1         #4x start register where data moved from
processor
      to module (1 to n)
```

This example will use the same name as the PIN.



1 Click the Variable declaration button to open the Variable Editor dialog box.


Variable Editor

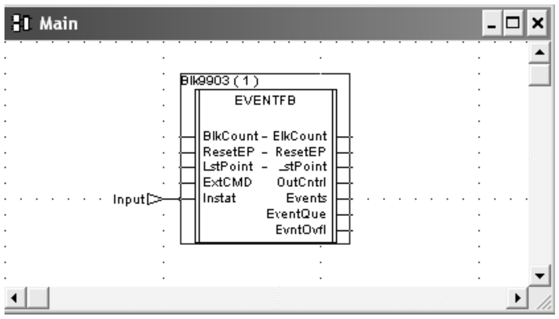
Type
(● Variables ○ Constants)

Search/Paste
Search/Replace

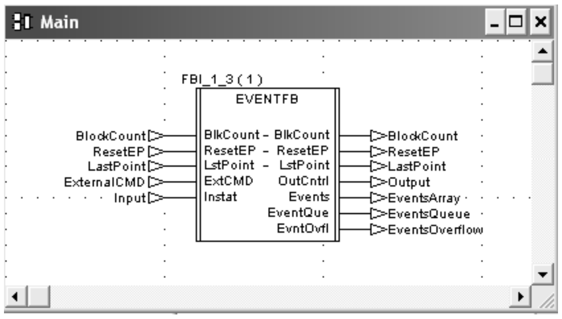
		Exp	Variable Name	Data Type	Address	InitValue	Use
2			EventsArray	EVENTSTRUCT		Set...	1
3			EventsOverflow	WORD			1
4			EventsQueue	WORD			1
5			ExternalCMD	WORD			7
6			Input	WORD64	300001		2
7			Input1	WordArr128	300065		0
8			Input2	WordArr128	301000		0
9			LastPoint	INT			2
10			Output	WORD64	400001	Set...	7
11			Output1	WordArr128	400065	Set...	0
12			Output2	WordArr128	401000	Set...	0
13			ResetEP	INT			2
14			Temp	INT			4
15							
16							

OK Cancel Help

2 Click OK and you should see the new variable associated to the Instat PIN:
3 Repeat for OutCntrl Pin to use for Output.

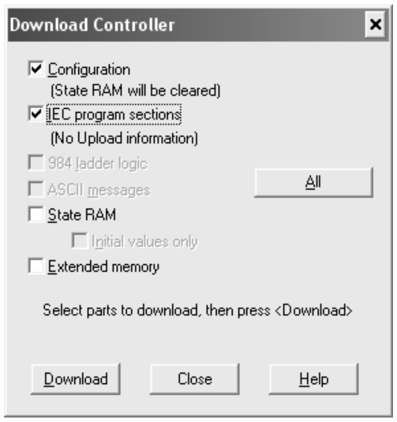


4 Repeat the last item for the other PINS (it is not necessary to associate any memory address to the other variables).



Download the Concept Project

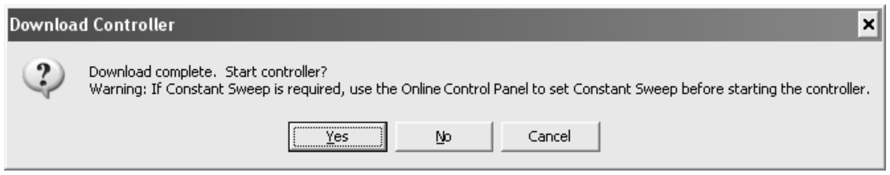
1 Select Online-Download to download the Concept Project. Make sure that the IEC program sections checkbox is selected:


Download Controller

- [✓] Configuration
  (State RAM will be cleared)
- [✓] IEC program sections
  (No Upload information)
- [ ] 984 ladder logic
- [ ] ASCII messages
- [ ] State RAM
  - [ ] Initial values only
- [ ] Extended memory

Select parts to download, then press <Download>

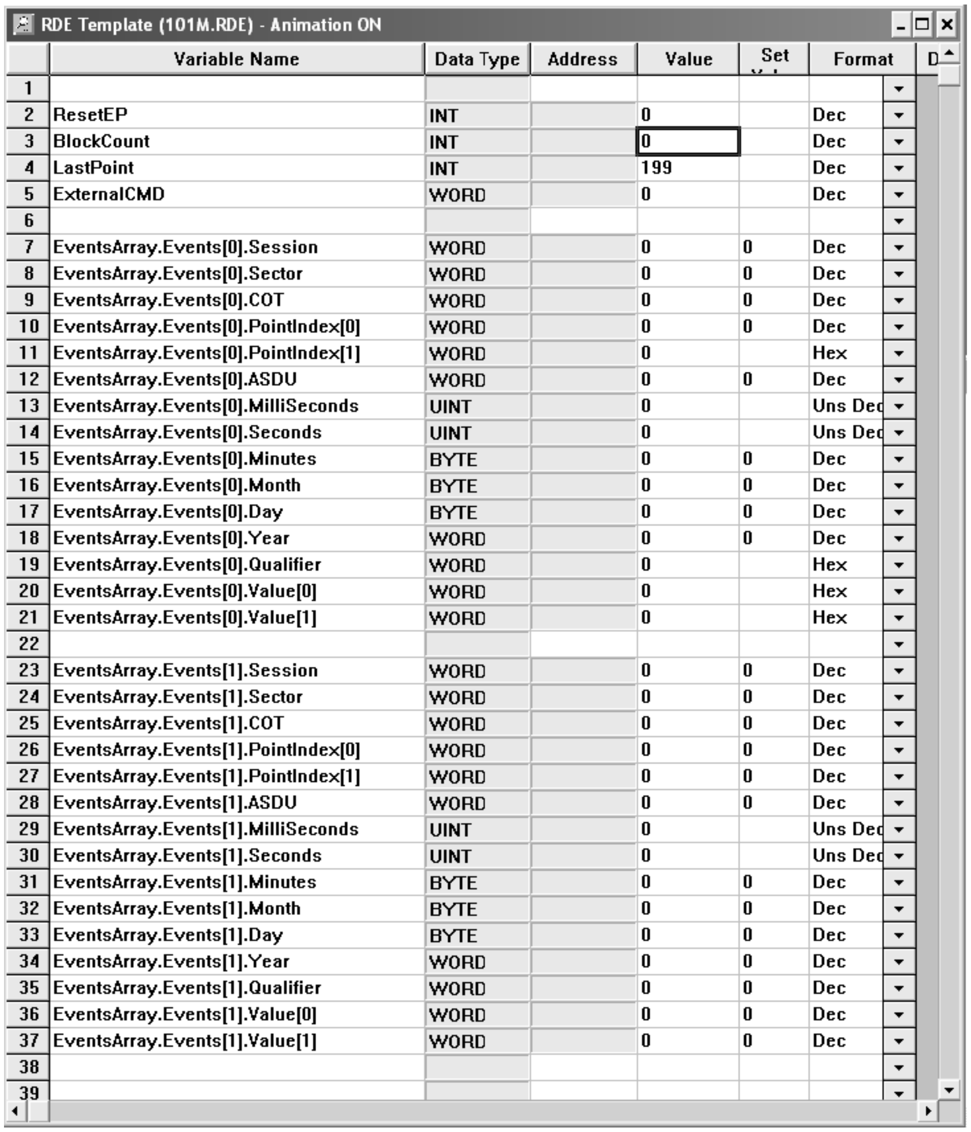
2 When the download is completed you should see the following window. Click Yes.


Download Controller

Download complete. Start controller?
Warning: If Constant Sweep is required, use the Online Control Panel to set Constant Sweep before starting the controller.

Using the EVENTFB Function Block

In order to show how the function block can be used we will create the following Template. This template shows the BlkCount, LstPoint, ResetEP, ExternalCMD variables and also the first two event elements (Events.Event[0] and Events.Event[1]):



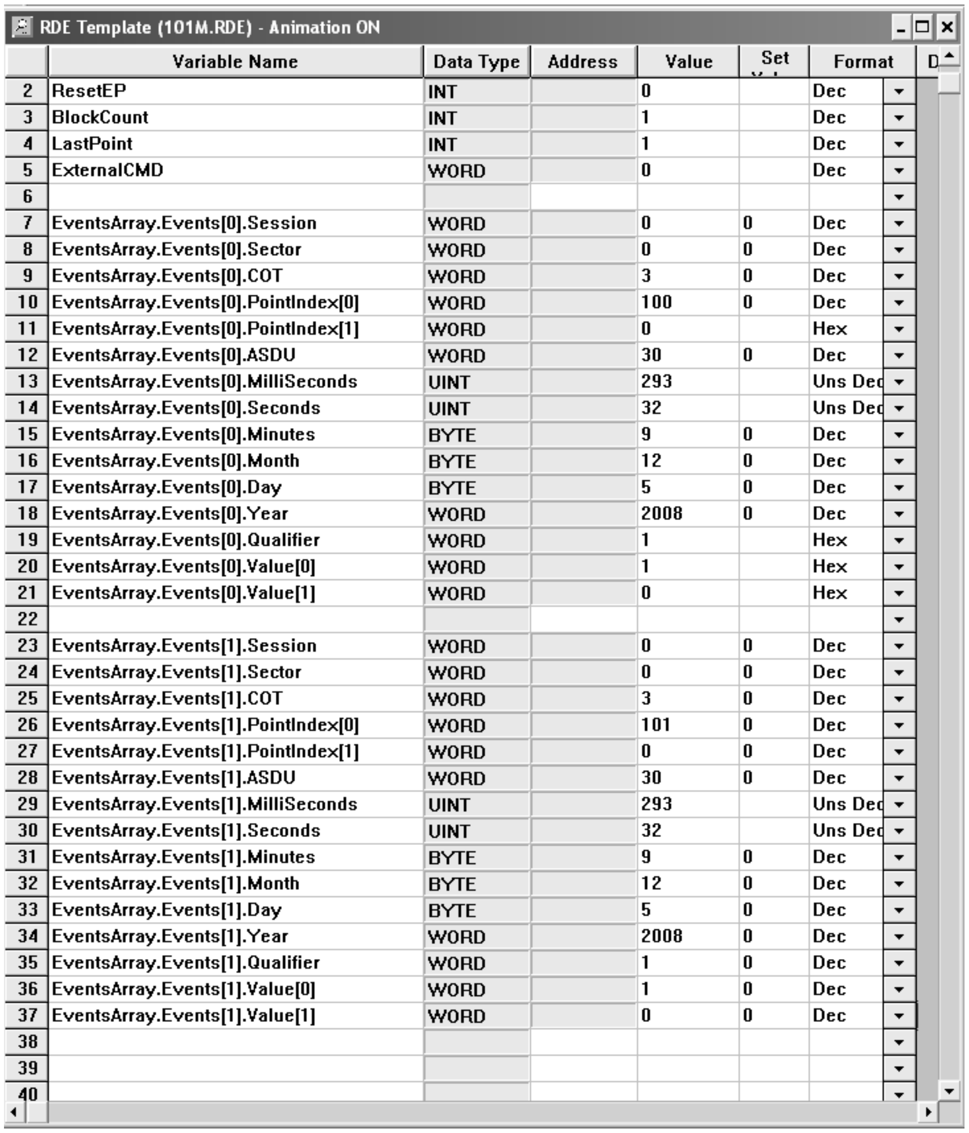
RDE Template (101M.RDE) - Animation ON

	Variable Name	Data Type	Address	Value	Set	Format	D
1							
2	ResetEP	INT		0		Dec	
3	BlockCount	INT		0		Dec	
4	LastPoint	INT		199		Dec	
5	ExternalCMD	WORD		0		Dec	
6							
7	EventsArray.Events[0].Session	WORD		0	0	Dec	
8	EventsArray.Events[0].Sector	WORD		0	0	Dec	
9	EventsArray.Events[0].COT	WORD		0	0	Dec	
10	EventsArray.Events[0].PointIndex[0]	WORD		0	0	Dec	
11	EventsArray.Events[0].PointIndex[1]	WORD		0		Hex	
12	EventsArray.Events[0].ASDU	WORD		0	0	Dec	
13	EventsArray.Events[0].MilliSeconds	UINT		0		Uns Dec	
14	EventsArray.Events[0].Seconds	UINT		0		Uns Dec	
15	EventsArray.Events[0].Minutes	BYTE		0	0	Dec	
16	EventsArray.Events[0].Month	BYTE		0	0	Dec	
17	EventsArray.Events[0].Day	BYTE		0	0	Dec	
18	EventsArray.Events[0].Year	WORD		0	0	Dec	
19	EventsArray.Events[0].Qualifier	WORD		0		Hex	
20	EventsArray.Events[0].Value[0]	WORD		0		Hex	
21	EventsArray.Events[0].Value[1]	WORD		0		Hex	
22							
23	EventsArray.Events[1].Session	WORD		0	0	Dec	
24	EventsArray.Events[1].Sector	WORD		0	0	Dec	
25	EventsArray.Events[1].COT	WORD		0	0	Dec	
26	EventsArray.Events[1].PointIndex[0]	WORD		0	0	Dec	
27	EventsArray.Events[1].PointIndex[1]	WORD		0	0	Dec	
28	EventsArray.Events[1].ASDU	WORD		0	0	Dec	
29	EventsArray.Events[1].MilliSeconds	UINT		0		Uns Dec	
30	EventsArray.Events[1].Seconds	UINT		0		Uns Dec	
31	EventsArray.Events[1].Minutes	BYTE		0	0	Dec	
32	EventsArray.Events[1].Month	BYTE		0	0	Dec	
33	EventsArray.Events[1].Day	BYTE		0	0	Dec	
34	EventsArray.Events[1].Year	WORD		0	0	Dec	
35	EventsArray.Events[1].Qualifier	WORD		0	0	Dec	
36	EventsArray.Events[1].Value[0]	WORD		0	0	Dec	
37	EventsArray.Events[1].Value[1]	WORD		0	0	Dec	
38							
39							

In this example, the remote device has sent two events with timestamp to the module (in same block 9903). The following shows an example of how the variables associated to the EVENTFB function block would be updated.

- BlkCount: shows a value of 1 because the processor has received two Events in one block.
- LstPoint: shows a value of 1 because the last element that was updated has an index of 1 (Events.Event[1]).
- Events.Event[0]: shows the first event received from the module

- Events.Event[1]: shows the second event received from the module



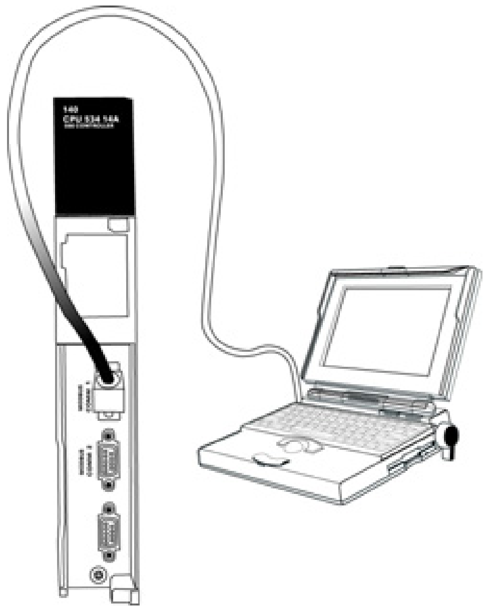
RDE Template (101M.RDE) - Animation ON

	Variable Name	Data Type	Address	Value	Set	Format
2	ResetEP	INT		0		Dec
3	BlockCount	INT		1		Dec
4	LastPoint	INT		1		Dec
5	ExternalCMD	WORD		0		Dec
6						
7	EventsArray.Events[0].Session	WORD		0	0	Dec
8	EventsArray.Events[0].Sector	WORD		0	0	Dec
9	EventsArray.Events[0].COT	WORD		3	0	Dec
10	EventsArray.Events[0].PointIndex[0]	WORD		100	0	Dec
11	EventsArray.Events[0].PointIndex[1]	WORD		0		Hex
12	EventsArray.Events[0].ASDU	WORD		30	0	Dec
13	EventsArray.Events[0].MilliSeconds	UINT		293		Uns Dec
14	EventsArray.Events[0].Seconds	UINT		32		Uns Dec
15	EventsArray.Events[0].Minutes	BYTE		9	0	Dec
16	EventsArray.Events[0].Month	BYTE		12	0	Dec
17	EventsArray.Events[0].Day	BYTE		5	0	Dec
18	EventsArray.Events[0].Year	WORD		2008	0	Dec
19	EventsArray.Events[0].Qualifier	WORD		1		Hex
20	EventsArray.Events[0].Value[0]	WORD		1		Hex
21	EventsArray.Events[0].Value[1]	WORD		0		Hex
22						
23	EventsArray.Events[1].Session	WORD		0	0	Dec
24	EventsArray.Events[1].Sector	WORD		0	0	Dec
25	EventsArray.Events[1].COT	WORD		3	0	Dec
26	EventsArray.Events[1].PointIndex[0]	WORD		101	0	Dec
27	EventsArray.Events[1].PointIndex[1]	WORD		0	0	Dec
28	EventsArray.Events[1].ASDU	WORD		30	0	Dec
29	EventsArray.Events[1].MilliSeconds	UINT		293		Uns Dec
30	EventsArray.Events[1].Seconds	UINT		32		Uns Dec
31	EventsArray.Events[1].Minutes	BYTE		9	0	Dec
32	EventsArray.Events[1].Month	BYTE		12	0	Dec
33	EventsArray.Events[1].Day	BYTE		5	0	Dec
34	EventsArray.Events[1].Year	WORD		2008	0	Dec
35	EventsArray.Events[1].Qualifier	WORD		1	0	Dec
36	EventsArray.Events[1].Value[0]	WORD		1	0	Dec
37	EventsArray.Events[1].Value[1]	WORD		0	0	Dec
38						
39						
40						

2.6 Download the Project to the Processor

The next step is to download (copy) the project file to the Quantum Processor.

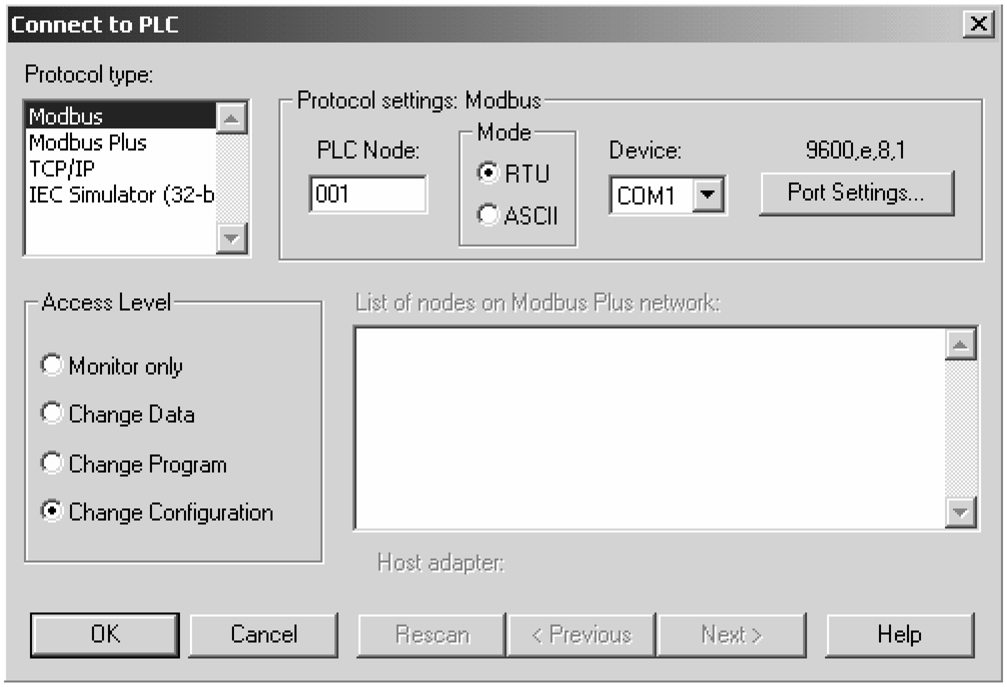
1 Use the null modem cable to connect your PC's serial port to the Quantum processor, as shown in the following illustration.



Note: You can use a Modbus Plus Network Option Module (NOM Module) module in place of the serial port if necessary.

2 Open the PLC menu, and then choose Connect.

3 In the PLC Configuration dialog box, open the Online menu, and then choose Connect. This action opens the Connect to PLC dialog box.


Connect to PLC

Protocol type: Modbus, Modbus Plus, TCP/IP, IEC Simulator (32-b

Protocol settings: Modbus
PLC Node: 001
Mode: [✓ RTU] ASCII
Device: COM1
9600,e,8,1

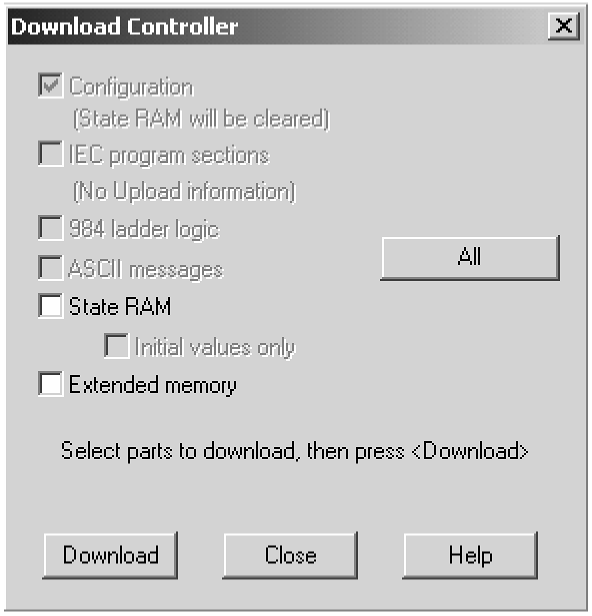
Access Level: Monitor only, Change Data, Change Program, [✓ Change Configuration]

List of nodes on Modbus Plus network:
Host adapter:

4 Leave the default settings as shown and click OK.

Note: Click OK to dismiss any message boxes that appear during the connection process.

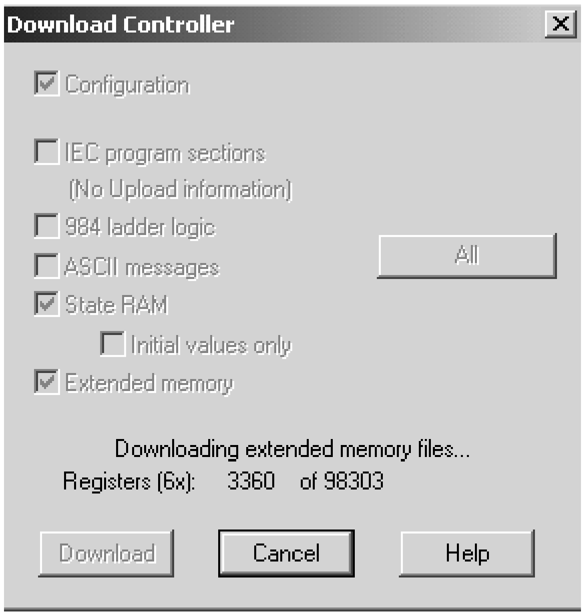
5 In the PLC Configuration window, open the Online menu, and then choose Download. This action opens the Download Controller dialog box.


Download Controller

[✓ Configuration] (State RAM will be cleared)
IEC program sections (No Upload information)
984 ladder logic
ASCII messages
State RAM
Initial values only
Extended memory

Select parts to download, then press <Download>

6 Click all, and then click Download. If a message box appears indicating that the controller is running, click Yes to shut down the controller. The Download Controller dialog box displays the status of the download as shown in the following illustration.



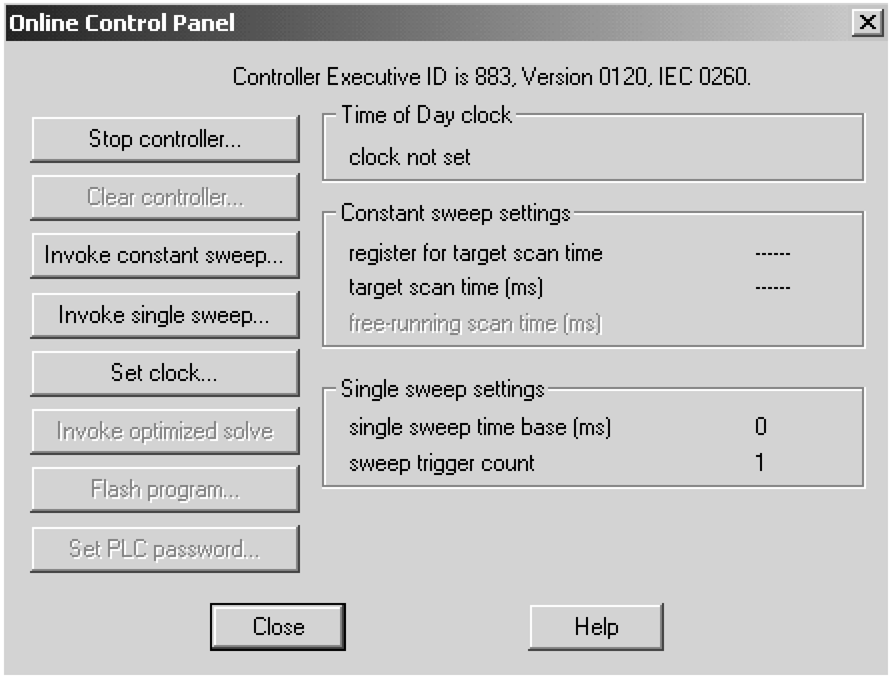
Downloading extended memory files...
Registers (6x): 3360 of 98303

7 When the download is complete, you will be prompted to restart the controller. Click Yes to restart the controller.

2.7 Verify Successful Download

The final step is to verify that the configuration changes you made were received successfully by the module, and to make some adjustments to your settings.

1 In the PLC Configuration window, open the Online menu, and then choose Online Control Panel. This action opens the Online Control Panel dialog box.



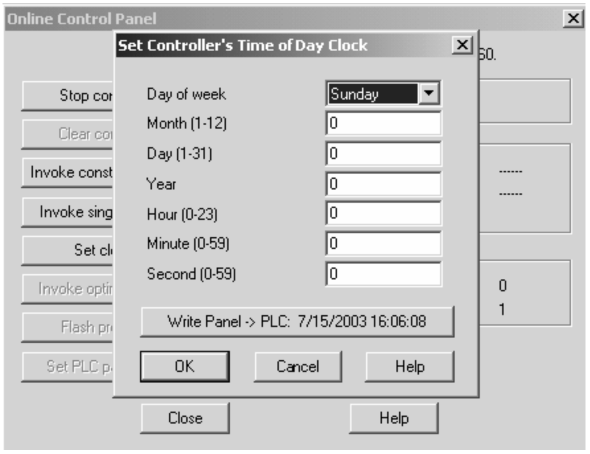
Controller Executive ID is 883, Version 0120, IEC 0260.

Time of Day clock
clock not set

Constant sweep settings
register for target scan time ------
target scan time (ms) ------
free-running scan time (ms)

Single sweep settings
single sweep time base (ms) 0
sweep trigger count 1

2 Click the Set Clock button to open the Set Controller's Time of Day Clock dialog box.

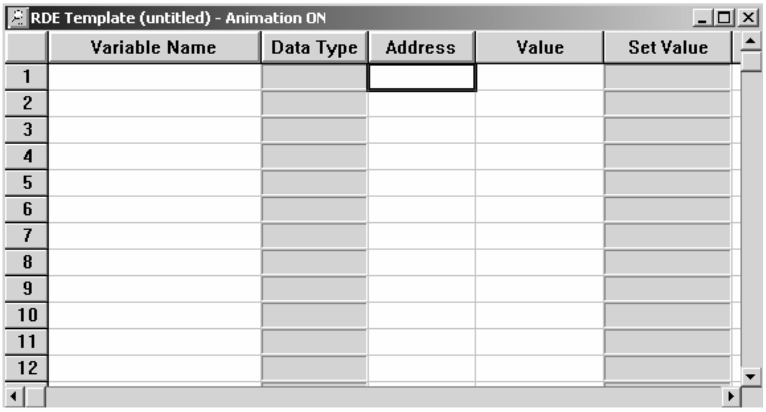

Online Control Panel
Set Controller's Time of Day Clock
Write Panel -> PLC: 7/15/2003 16:06:08
OK
Cancel
Help
Close
Help

Day of week: Sunday
Month (1-12): 0
Day (1-31): 0
Year: 0
Hour (0-23): 0
Minute (0-59): 0
Second (0-59): 0

3 Click the Write Panel button. This action updates the date and time fields in this dialog box. Click OK to close this dialog box and return to the previous window.

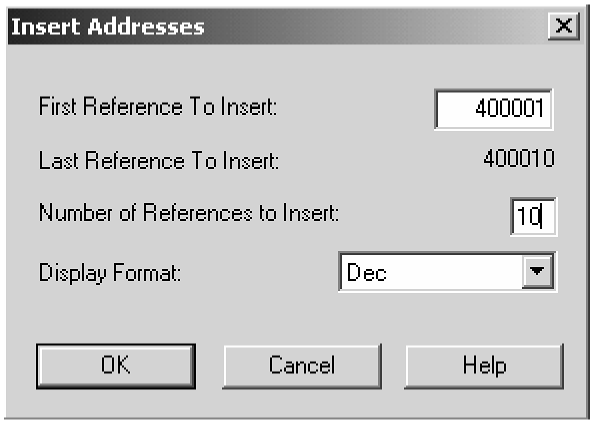
4 Click Close to close the Online Control Panel dialog box.

5 In the PLC Configuration window, open the Online menu, and then choose Reference Data Editor. This action opens the Reference Data Editor dialog box. On this dialog box, you will add preset values to data registers that will later be monitored in the ProTalk module.
6 Place the cursor over the first address field, as shown in the following illustration.



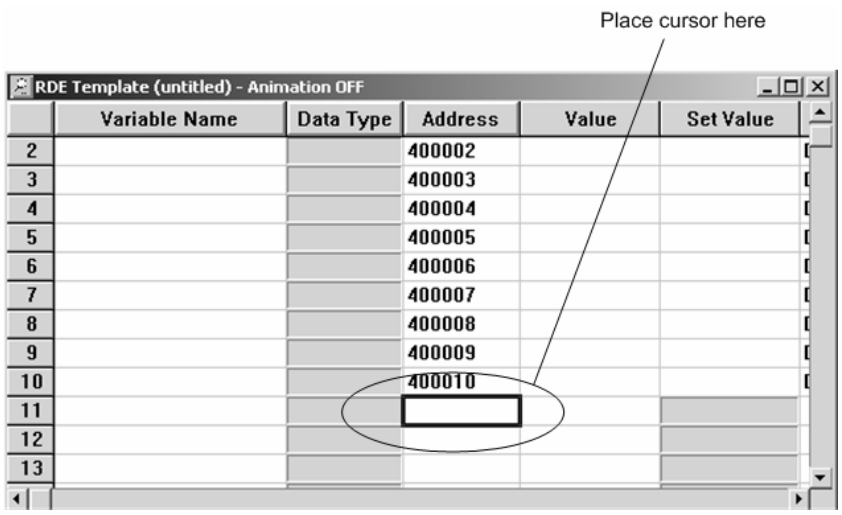
	Variable Name	Data Type	Address	Value	Set Value
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

7 In the PLC Configuration window, open the Templates menu, and then choose Insert addresses. This action opens the Insert addresses dialog box.
8 On the Insert addresses dialog box, enter the values shown in the following illustration, and then click OK.



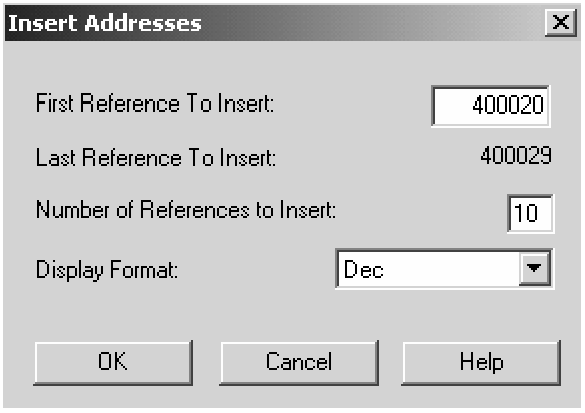
First Reference To Insert: 400001
Last Reference To Insert: 400010
Number of References to Insert: 10
Display Format: Dec

9 Notice that the template populates the address range, as shown in the following illustration. Place your cursor as shown in the first blank address field below the addresses you just entered.



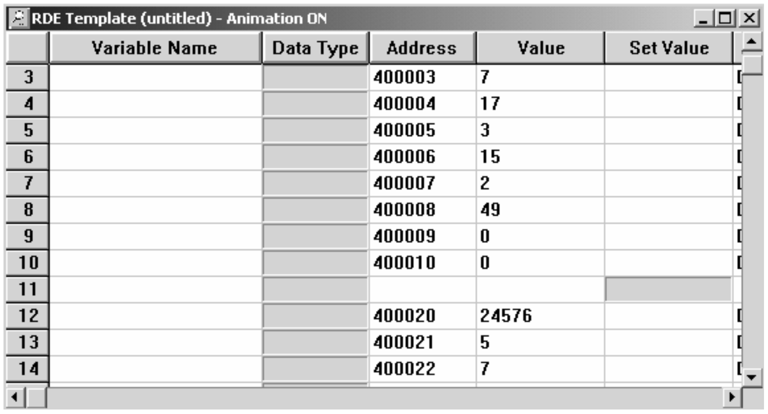
	Variable Name	Data Type	Address	Value	Set Value
2			400002		
3			400003		
4			400004		
5			400005		
6			400006		
7			400007		
8			400008		
9			400009		
10			400010		
11					
12					
13					

10 Repeat steps 6 through 9, using the values in the following illustration:



First Reference To Insert: 400020
Last Reference To Insert: 400029
Number of References to Insert: 10
Display Format: Dec

11 In the PLC Configuration window, open the Online menu, and then choose animate. This action opens the RDE Template dialog box, with animated values in the Value field.



	Variable Name	Data Type	Address	Value	Set Value
3			400003	7	
4			400004	17	
5			400005	3	
6			400006	15	
7			400007	2	
8			400008	49	
9			400009	0	
10			400010	0	
11					
12			400020	24576	
13			400021	5	
14			400022	7	

12 Verify that values shown are cycling, starting from address 400065 on up.
13 In the PLC Configuration window, open the Templates menu, and then choose Save Template as. Name the template ptqclock, and then click OK to save the template.
14 In the PLC Configuration window, open the Online menu, and then choose Disconnect. At the disconnect message, click Yes to confirm your choice.

At this point, you have successfully

- Created and downloaded a Quantum project to the PLC
- Preset values in data registers that will later be monitored in the ProTalk module.

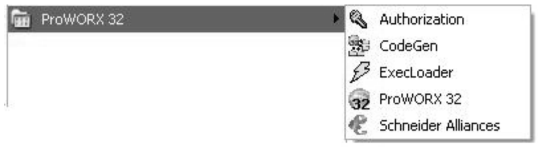
You are now ready to complete the installation and setup of the ProTalk module.

3 Configuring the Processor with ProWORX

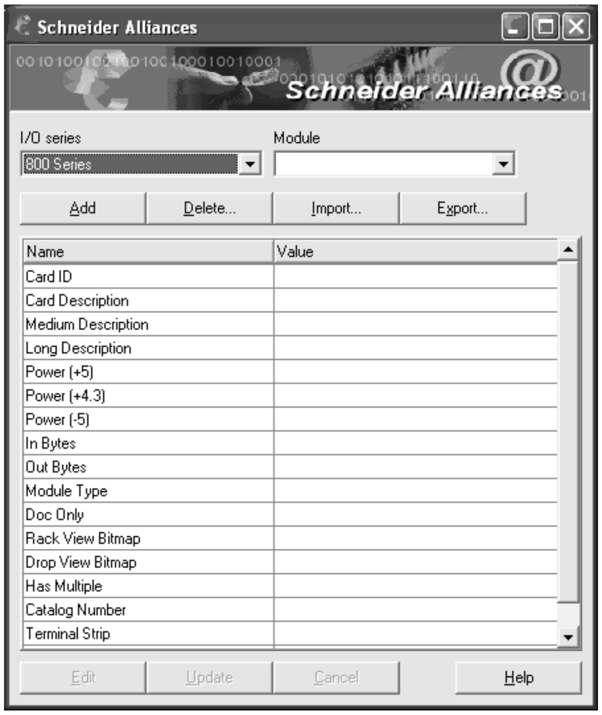
When you use ProWORX 32 software to configure the processor, use the example SaF file provided on the ProTalk Solutions CD-ROM.

Important Note: Proworx software does not report whether the PTQ module is present in the rack, and therefore is not able to report the health status of the module when the module is online with the Quantum processor. Please take this into account when monitoring the status of the PTQ module.

1 Run the Schneider_alliances.exe application that is installed with the Proworx 32 software:

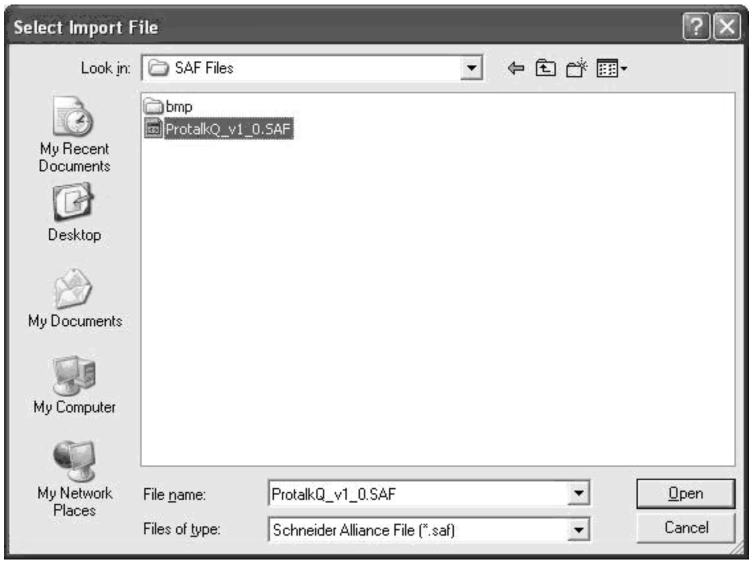


2 Click on Import…



Name	Value
Card ID	
Card Description	
Medium Description	
Long Description	
Power (+5)	
Power (+4.3)	
Power (-5)	
In Bytes	
Out Bytes	
Module Type	
Doc Only	
Rack View Bitmap	
Drop View Bitmap	
Has Multiple	
Catalog Number	
Terminal Strip	

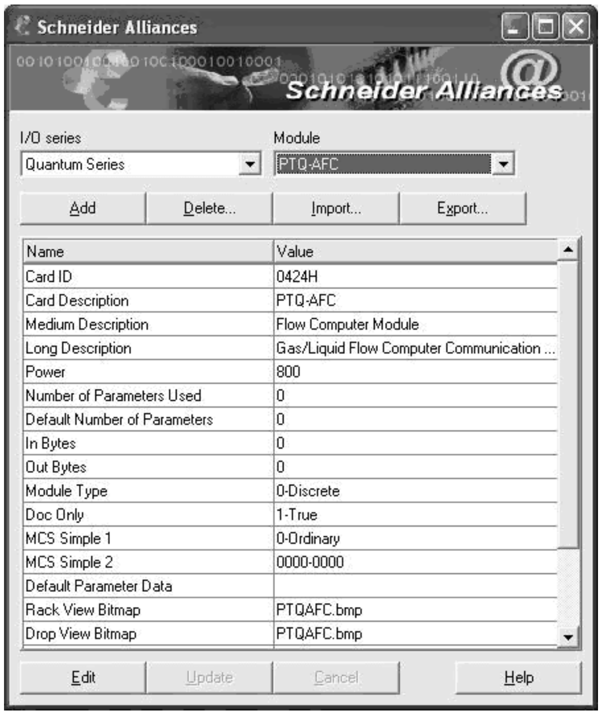
3 Select the .SaF File that is located at the CD-ROM shipped with the PTQ module.



File name: ProtalkQ_v1_0.SAF

Files of type: Schneider Alliance File (*.saf)

4 After you click on Open you should see the PTQ modules imported (select I/O series as Quantum):

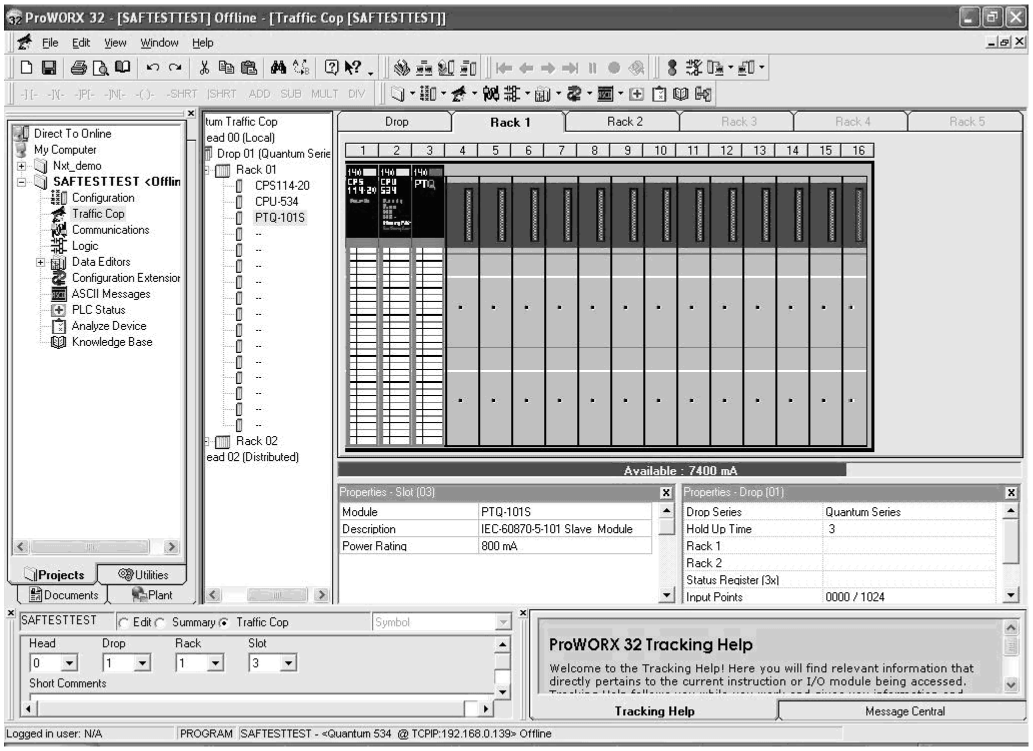


I/O series: Quantum Series

Module: PTQ-AFC

Name	Value
Card ID	0424H
Card Description	PTQ-AFC
Medium Description	Flow Computer Module
Long Description	Gas/Liquid Flow Computer Communication ...
Power	800
Number of Parameters Used	0
Default Number of Parameters	0
In Bytes	0
Out Bytes	0
Module Type	0-Discrete
Doc Only	1-True
MCS Simple 1	0-Ordinary
MCS Simple 2	0000-0000
Default Parameter Data	
Rack View Bitmap	PTQAFC.bmp
Drop View Bitmap	PTQAFC.bmp

Now you can close the Schneider alliances application and run the Proworx 32 software. At the Traffic Cop section, select the PTQ module to be inserted at the slot:



Properties - Slot (03)

Module	PTQ-101S
Description	IEC-60870-5-101 Slave Module
Power Rating	800 mA

Properties - Drop (01)

Drop Series	Quantum Series
Hold Up Time	3
Rack 1	
Rack 2	
Status Register (3x)	
Input Points	0000 / 1024

SAFTESTTEST (Edit (Summary [✓ Traffic Cop] Symbol

Head 0 Drop 1 Rack 1 Slot 3

Short Comments

Welcome to the Tracking Help! Here you will find relevant information that directly pertains to the current instruction or I/O module being accessed.

Logged in user: N/A PROGRAM SAFTESTTEST - <Quantum 534 @ TCPIP:192.168.0.139> Offline

4 Configuring the Processor with UnityPro XL

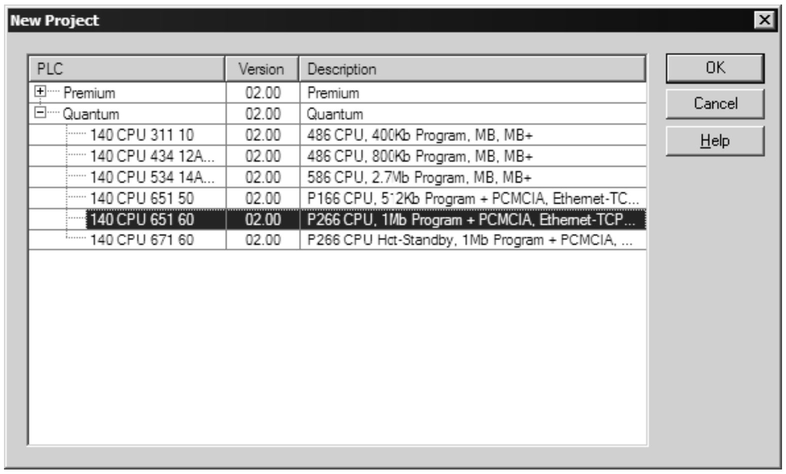
In This Chapter

The following steps are designed to ensure that the processor (Quantum or Unity) is able to transfer data successfully with the PTQ module. As part of this procedure, you will use UnityPro XL to create a project, add the PTQ module to the project, set up data memory for the project, and then download the project to the processor.

4.1 Create a New Project

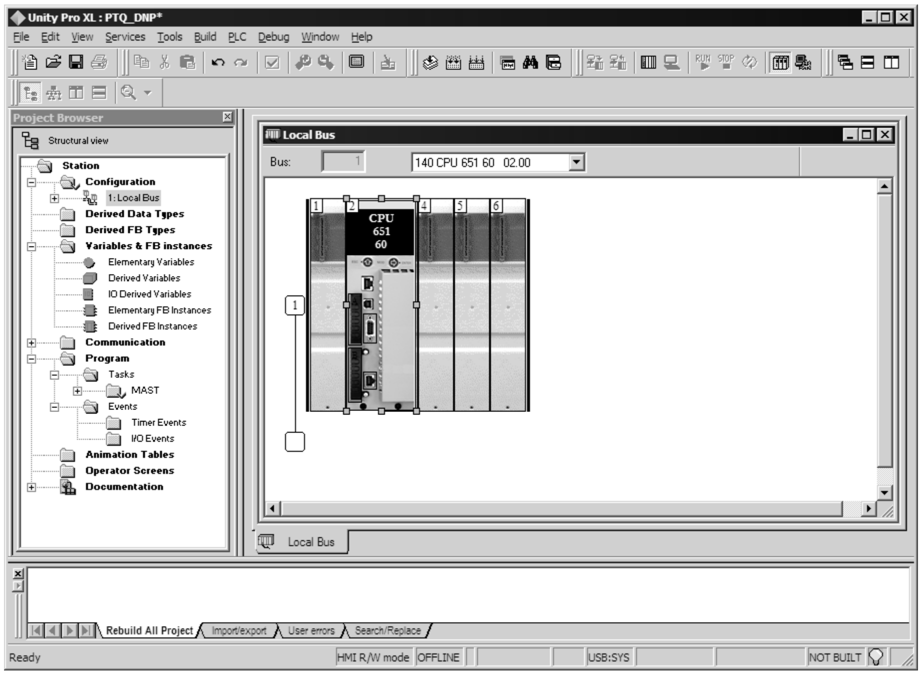
The first step is to open UnityPro XL and create a new project.

1 In the New Project dialog box, choose the CPU type. In the following illustration, the CPU is 140 CPU 651 60. Choose the processor type that matches your own hardware configuration, if it differs from the example. Click OK to continue.

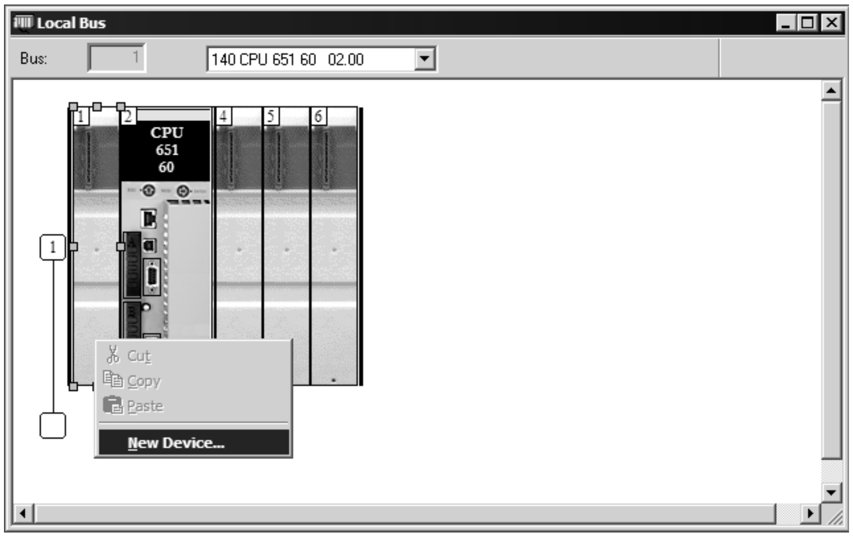


PLC	Version	Description
Premium	02.00	Premium
Quantum	02.00	Quantum
140 CPU 311 10	02.00	486 CPU, 400Kb Program, MB, MB+
140 CPU 434 12A...	02.00	486 CPU, 800Kb Program, MB, MB+
140 CPU 534 14A...	02.00	586 CPU, 2.7Mb Program, MB, MB+
140 CPU 651 50	02.00	P166 CPU, 512Kb Program + PCMCIA, Ethernet-TC...
140 CPU 651 60	02.00	P266 CPU, 1Mb Program + PCMCIA, Ethernet-TCP...
140 CPU 671 60	02.00	P266 CPU Hot-Standby, 1Mb Program + PCMCIA, ...

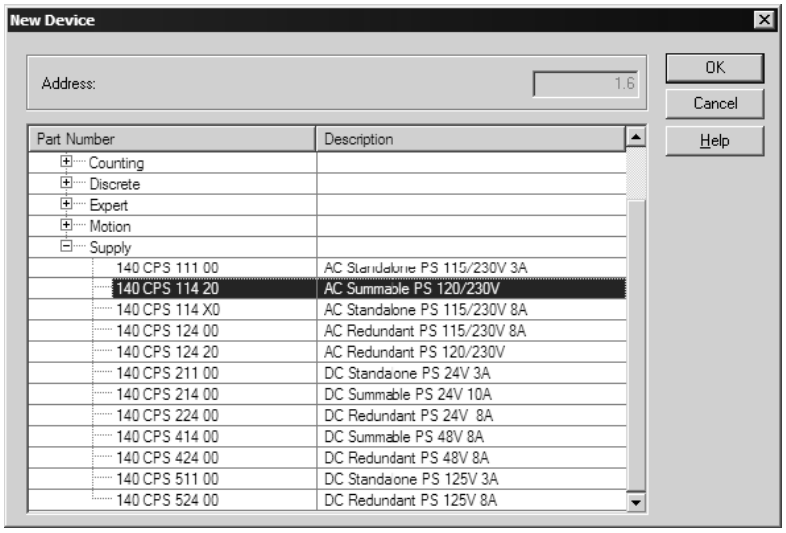
2 The next step is to add a power supply to the project. In the Project Browser, expand the Configuration folder, and then double-click the 1:LocalBus icon. This action opens a graphical window showing the arrangement of devices in your Quantum rack.



3 Select the rack position for the power supply, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose New Device..



4 Expand the Supply folder, and then select your power supply from the list. Click OK to continue.



Address: 1.6

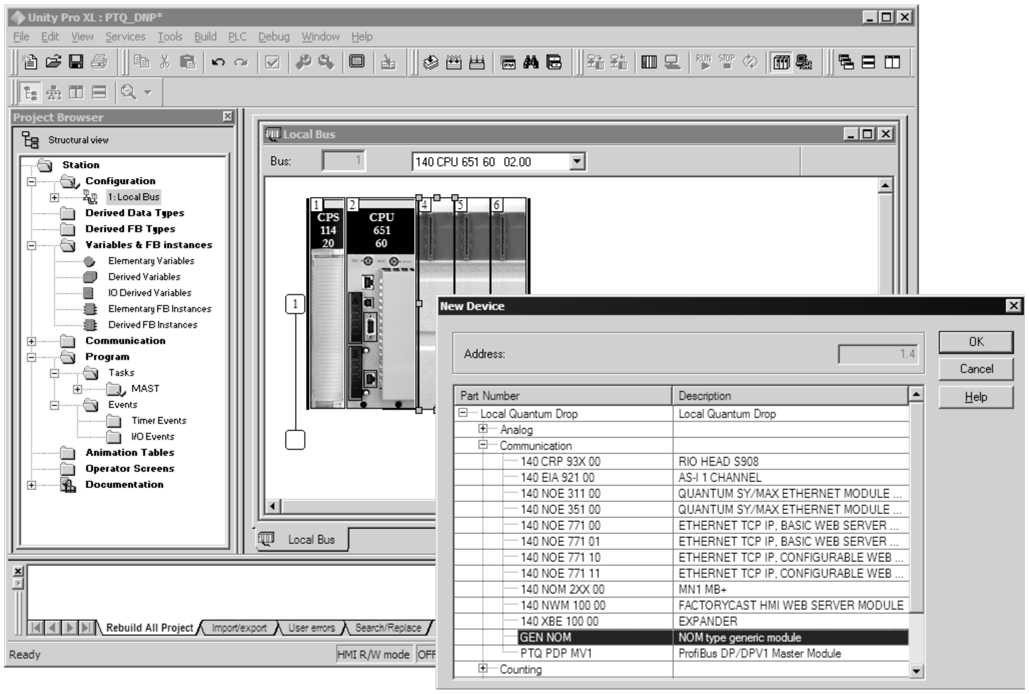
Part Number	Description
⊞ Counting	
⊞ Discrete	
⊞ Expert	
⊞ Motion	
⊟ Supply	
140 CPS 111 00	AC Standalone PS 115/230V 3A
140 CPS 114 20	AC Summable PS 120/230V
140 CPS 114 X0	AC Standalone PS 115/230V 8A
140 CPS 124 00	AC Redundant PS 115/230V 8A
140 CPS 124 20	AC Redundant PS 120/230V
140 CPS 211 00	DC Standaone PS 24V 3A
140 CPS 214 00	DC Summable PS 24V 10A
140 CPS 224 00	DC Redundant PS 24V 8A
140 CPS 414 00	DC Summable PS 48V 8A
140 CPS 424 00	DC Redundant PS 48V 8A
140 CPS 511 00	DC Standaone PS 125V 3A
140 CPS 524 00	DC Redundant PS 125V 8A

5 Repeat these steps to add any additional devices to your Quantum Rack.

4.2 Add the PTQ Module to the Project

The next step is to add the PTQ module.

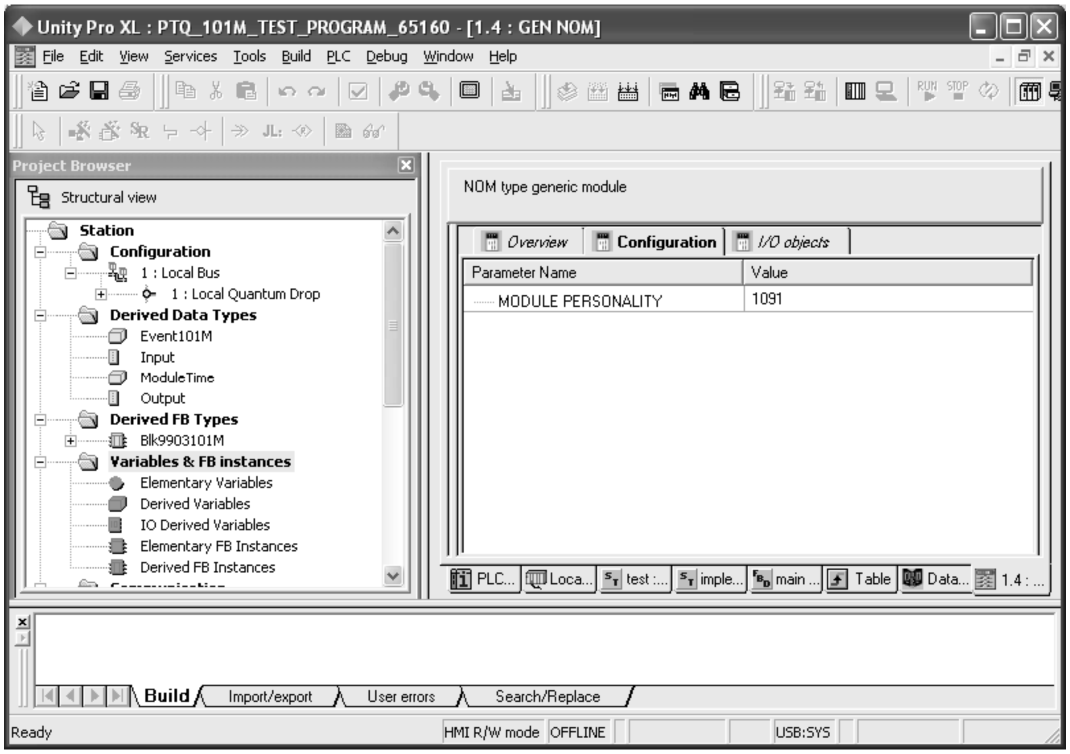
1 Expand the Communication tree, and select GEN NOM. This module type provides extended communication capabilities for the Quantum system, and allows communication between the PLC and the PTQ module without requiring additional programming.



Address: 1.4

Part Number	Description
⊟ Local Quantum Drop	Local Quantum Drop
⊞ Analog	
⊟ Communication	
140 CRP 93X 00	RIO HEAD S908
140 EIA 921 00	AS-I 1 CHANNEL
140 NOE 311 00	QUANTUM SY/MAX ETHERNET MODULE ...
140 NOE 351 00	QUANTUM SY/MAX ETHERNET MODULE ...
140 NOE 771 00	ETHERNET TCP IP, BASIC WEB SERVER ...
140 NOE 771 01	ETHERNET TCP IP, BASIC WEB SERVER ...
140 NOE 771 10	ETHERNET TCP IP, CONFIGURABLE WEB ...
140 NOE 771 11	ETHERNET TCP IP, CONFIGURABLE WEB ...
140 NOM 2XX 00	MN1 MB+
140 NWM 100 00	FACTORYCAST HMI WEB SERVER MODULE
140 XBE 100 00	EXPANDER
GEN NOM	NOM type generic module
PTQ PDP MV1	ProfiBus DP/DPV1 Master Module
⊞ Counting	

2 Next, enter the module personality value. The correct value for this ProTalk module is 1091 decimal (0443 hex).



Parameter Name	Value
MODULE PERSONALITY	1091

3 Before you can save the project in UnityProXL, you must validate the modifications. Open the Edit menu, and then choose Validate. If no errors are reported, you can save the project.

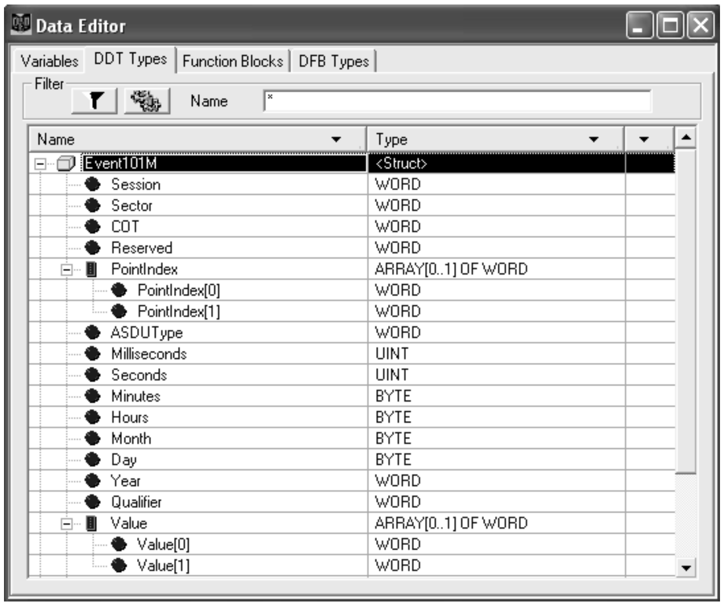
4 Save the project.

4.3 How to Set up and Use the Sample Function Block for Unity

4.3.1 EVENTFB Function Block Overview

The purpose of the EVENTFB sample function block is to transfer the events into a buffer that consists of an array of elements that stores all data in a convenient format for the user. The block 9903 passes data into a compacted format thus occupying the minimum amount of registers. The EVENTFB sample function block already extracts each event value into a separate register.

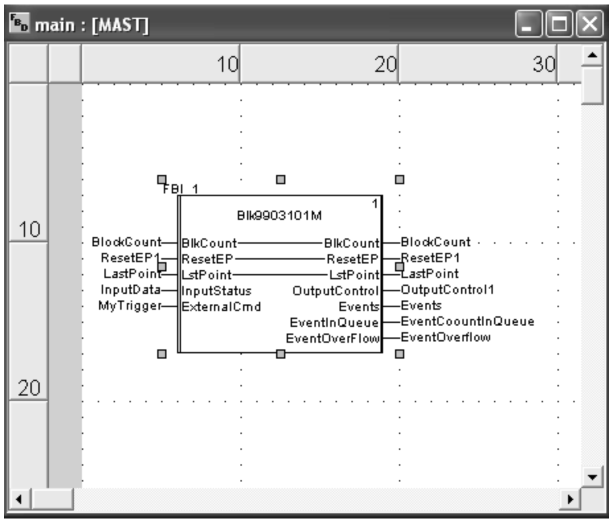
The following illustration shows the structure of each element of the buffer (extracted from the data type definition file):



Name	Type
Event101M	<Struct>
Session	WORD
Sector	WORD
COT	WORD
Reserved	WORD
PointIndex	ARRAY[0..1] OF WORD
PointIndex[0]	WORD
PointIndex[1]	WORD
ASDUType	WORD
Milliseconds	UINT
Seconds	UINT
Minutes	BYTE
Hours	BYTE
Month	BYTE
Day	BYTE
Year	WORD
Qualifier	WORD
Value	ARRAY[0..1] OF WORD
Value[0]	WORD
Value[1]	WORD

The data structure that stores the incoming events consists of a circular buffer that can store up to 199 events. The buffer consists of an array of 199 "EVENT101" elements presented previously. The element index can vary from 0 to 199. If the last event updated was located at index 199 then the next event will be copied to index 0.

The following illustration shows an instance example of the EVENTFB function block:



The EVENTFB function block contains the following PINs:

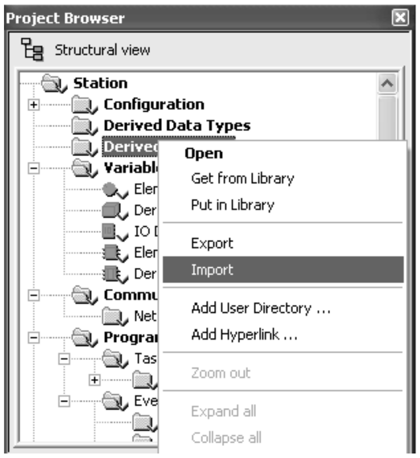
PIN	PIN Type	Data Type	Description
InputStatus	input	Array of WORD	Stores the memory area updated by block 9903. The start address must point to the same start address defined for block 9903 backplane data exchange (Point Address parameter).
ResetEP	input/output	INT	Move a value of one to reset the event pointer. This will cause the next event to be written to index 0 at the circular buffer. The register will be automatically reset to zero after the request was processed. This register should be only used for very specific applications (because the circular buffer automatically changes the element pointer from 199 to 0 after the maximum index was reached)
Events	Output	EVENT101M	Circular buffer that stores all received events in a convenient format for the user application. It can store up to 200 events (index varies from 0 to 199). After event 199 is updated the next event to be received will be automatically updated at index 0.
BlkCount	Input/Output	INT	Incremented after a block is received (and after the events in that block have been read into the circular buffer). The maximum value for this counter is 1000 (then it is automatically reset to 0)
LstPoint	Input/Output	INT	Pointer to the last event index read from the module. For example, if last event was updated at index 5 then this value will have the same value.
ExternalCmd	Input	WORD	This external command is used so user can issue different commands while the module receiving events.

PIN	PIN Type	Data Type	Description
OutputControl	Output	Array of WORD	Stores data to be sent from the processor to the module the start address it should match what you configured your backplane exchange to start.
Eventinqueue	Output	WORD	Indicates how many events are in the queue to be read.
EventOverflow	Output	WORD	This will be set to yes (1) if the overflow flag is set due to 199 events in the queue waiting to be read.

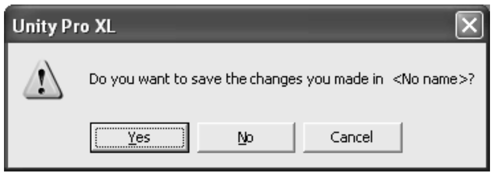
4.3.2 Importing the EVENTFB Function Block

1 Copy the provided function block from the ProSoft Solutions CD-ROM, or download the EVENTFB.XDB from http://www.prosoft-technology.com. For this example, save the Function Block in your My Documents folder.

2 In the Project Browser, select Derived FB Types and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose Import.



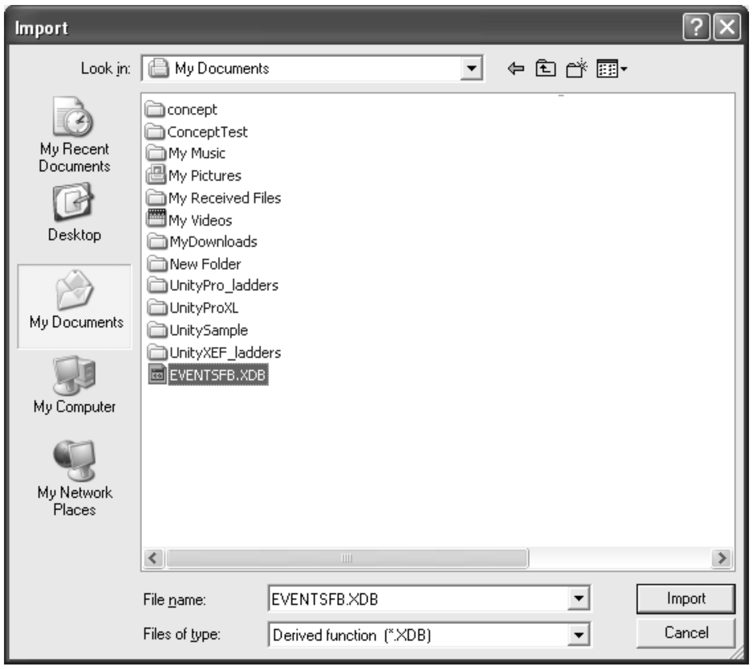
3 This action opens a confirmation dialog box.


Unity Pro XL

Do you want to save the changes you made in <No name>?

4 Click No to discard your changes, unless you are importing this function block to an existing project, in which case click Yes.

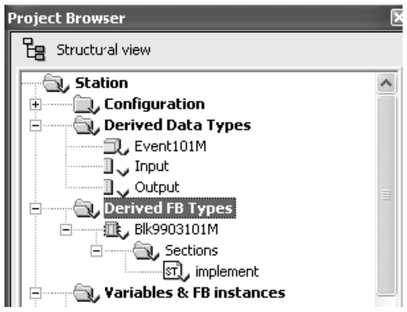
5 In the Import dialog box, click the Import button



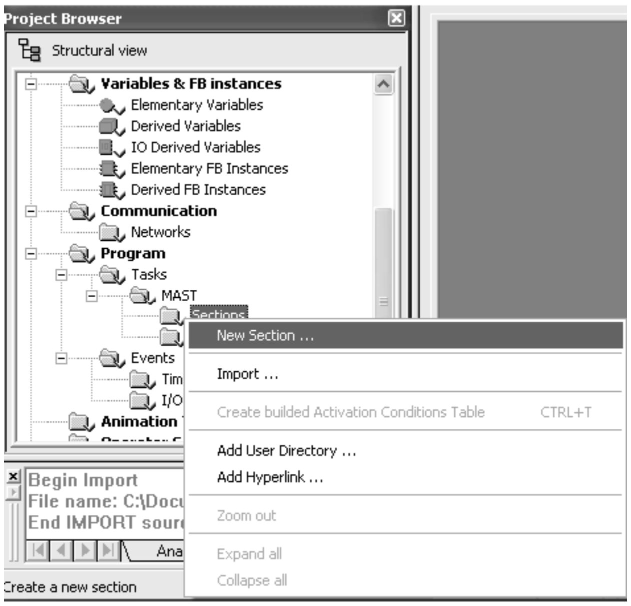
File name: EVENTSFB.XDB

Files of type: Derived function (*.XDB)

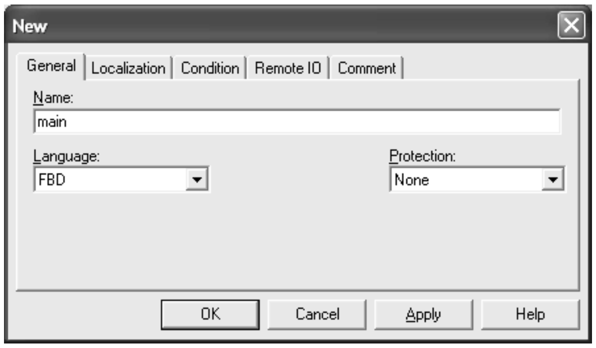
6 In the Project Browser, expand Derived Data types and verify that the import was complete.



7 Next, add the FB section to the programs folder.

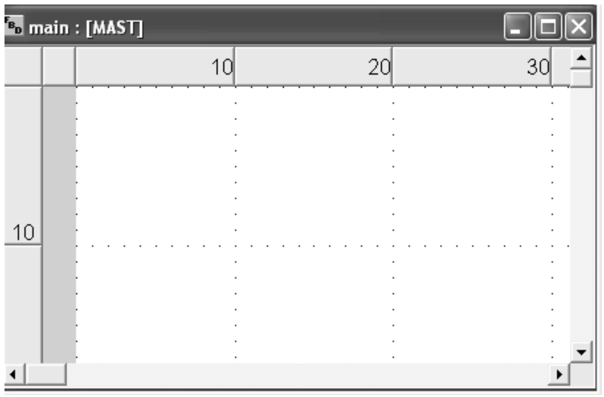


Begin Import
File name: C:\Docu
End IMPORT sour

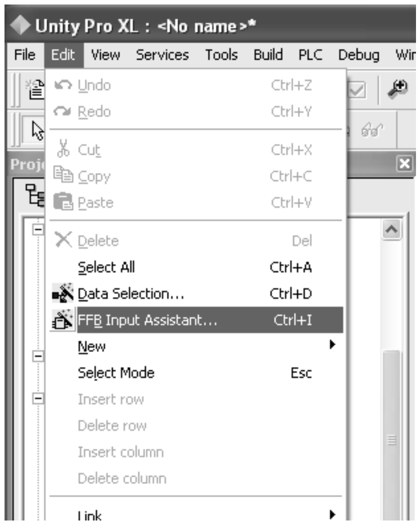


Name: main
Language: FBD
Protection: None

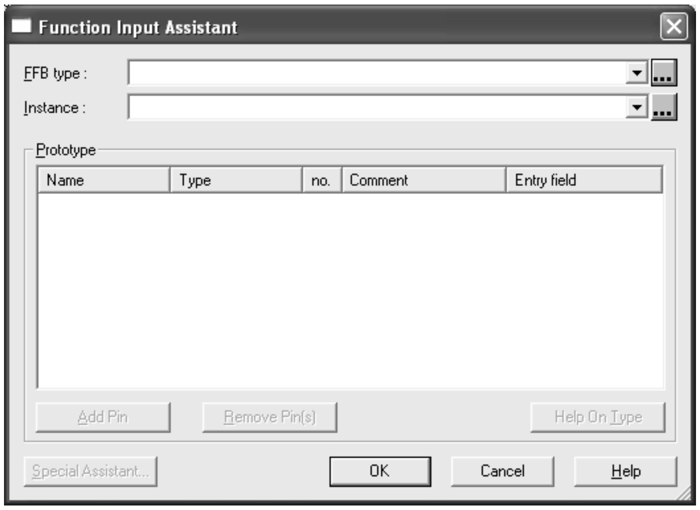
8 Click **OK**



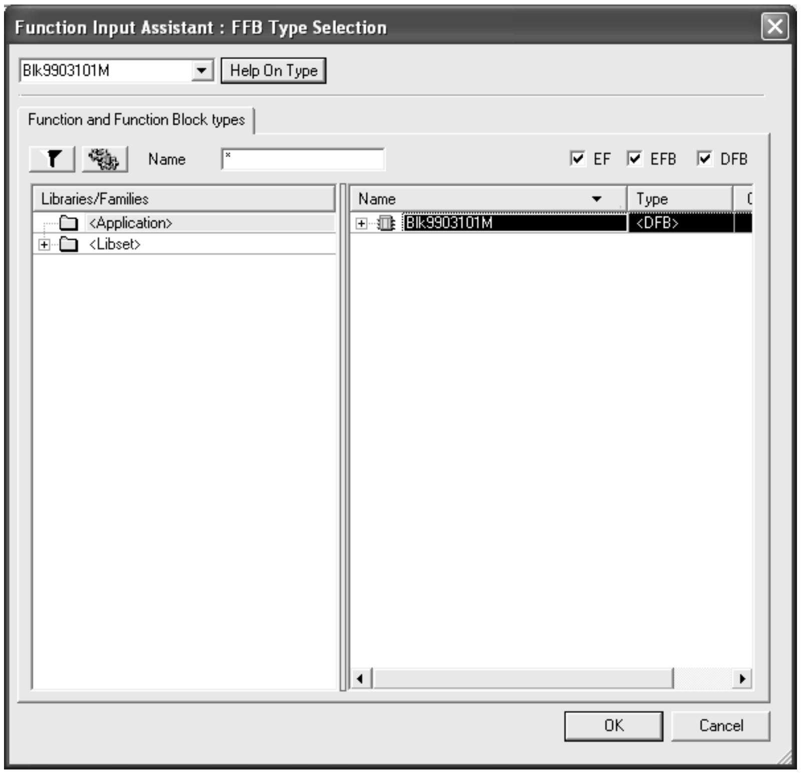
9 The next step is to add the Function block to the **Main** section. Open the **EDIT** Menu and then choose **FFB Input Assistant**



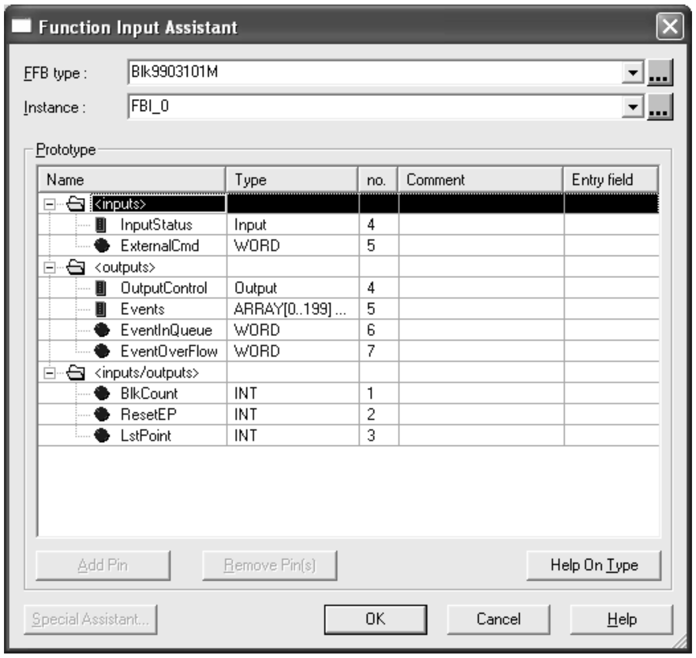
10 This action opens the Function Input Assistant dialog box.



11 Click the button to the right of the FFB type field.



12 Click **OK** to populate the Function Input Assistant dialog box.

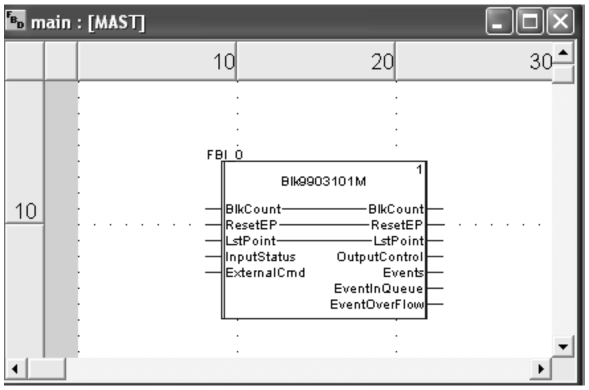


EFB type : Blk9903101M
Instance : FBI_0

Prototype

Name	Type	no.	Comment	Entry field
<inputs>				
InputStatus	Input	4		
ExternalCmd	WORD	5		
<outputs>				
OutputControl	Output	4		
Events	ARRAY[0..199] ...	5		
EventInQueue	WORD	6		
EventOverFlow	WORD	7		
<inputs/outputs>				
BlkCount	INT	1		
ResetEP	INT	2		
LstPoint	INT	3		

13 Click **OK** to dismiss the Function Input Asistant dialog box. Next, click to select the **Main** [Mast] section.



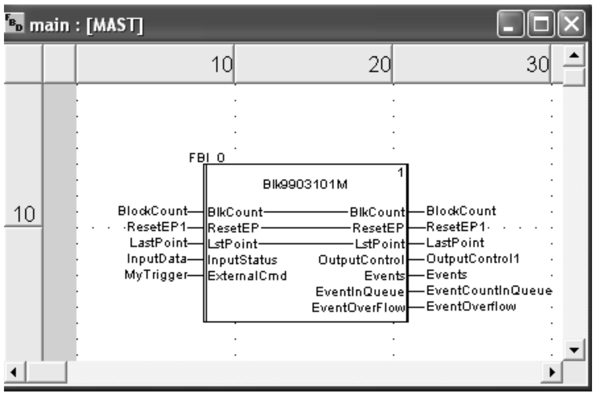
14 The next step is to create variables to associate to the function block PINs. We will start with the Inputstatus PIN. The variable for this PIN must point to the same start address where block 9903 will be copied to, referring the Register Start address entry in the module configuration file.

```
Register Start  :  1       #3x start register where data moved from module to
processor (1 to n)
```

This implies that the variable associated to PIN Inputstatus must also start at the same register address (%Iw1 for this example).
As the Inputstatus PIN will start same as 3x, the variable associated with PIN OutputControl must start at the same register address %MW1

```
4x Register Start    :  1         #4x start register where data moved from
processor to module (1 to n)
```

You must create user variables that match all PINs on the function block. The following illustration shows an example.

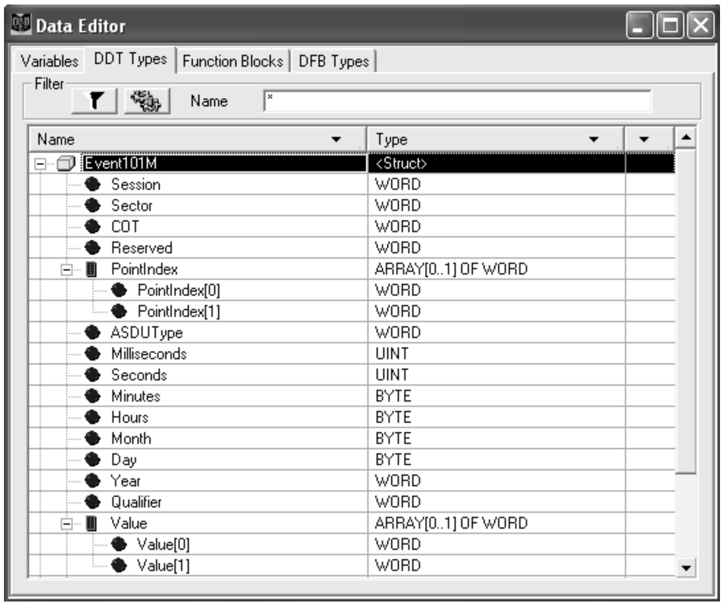


15 Before you can save the project in UnityProXL, you must validate the modifications. Open the Edit menu, and then choose Validate. If no errors are reported, you can save the project.
16 Save the project.
17 Download the project and test the function block

4.3.3 Using the EVENTFB Function Block

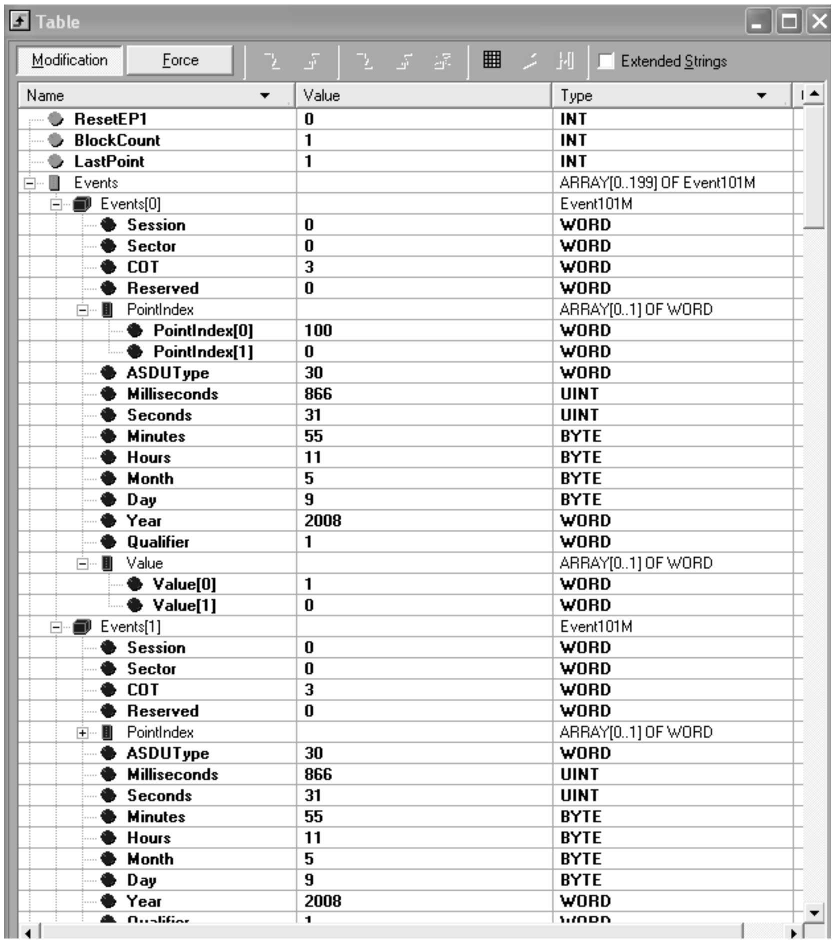
1 Create variables that match the Event format. When you import the function block, derived data types will also be imported.

The variable should match the Event 101M type and should match the following illustration.



Name	Type
Event101M	<Struct>
Session	WORD
Sector	WORD
COT	WORD
Reserved	WORD
PointIndex	ARRAY[0..1] OF WORD
PointIndex[0]	WORD
PointIndex[1]	WORD
ASDUType	WORD
Milliseconds	UINT
Seconds	UINT
Minutes	BYTE
Hours	BYTE
Month	BYTE
Day	BYTE
Year	WORD
Qualifier	WORD
Value	ARRAY[0..1] OF WORD
Value[0]	WORD
Value[1]	WORD

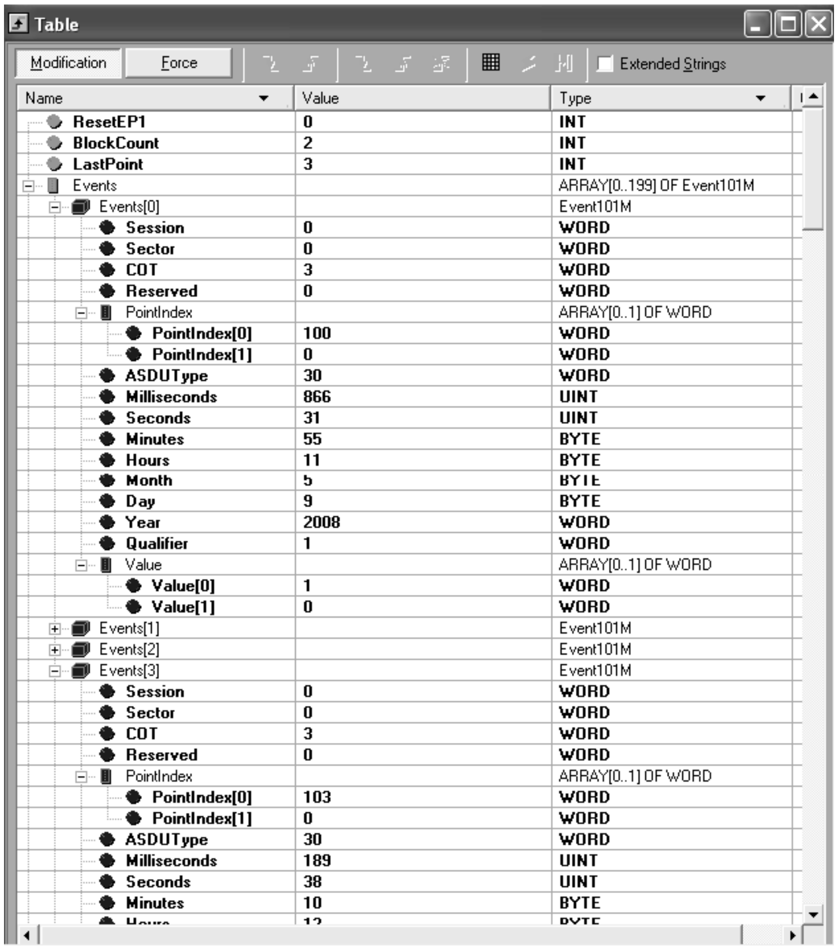
2 In the animation table, create an array of events to copy all 199 events, block count, ResetEP, and the last point, which is the index in the array for the last event to be copied to the array. The following illustration shows that we received two events in one block (block count=1) and last Point =1..



Name	Value	Type
ResetEP1	0	INT
BlockCount	1	INT
LastPoint	1	INT
Events		ARRAY[0..199] OF Event101M
Events[0]		Event101M
Session	0	WORD
Sector	0	WORD
COT	3	WORD
Reserved	0	WORD
PointIndex		ARRAY[0..1] OF WORD
PointIndex[0]	100	WORD
PointIndex[1]	0	WORD
ASDUType	30	WORD
Milliseconds	866	UINT
Seconds	31	UINT
Minutes	55	BYTE
Hours	11	BYTE
Month	5	BYTE
Day	9	BYTE
Year	2008	WORD
Qualifier	1	WORD
Value		ARRAY[0..1] OF WORD
Value[0]	1	WORD
Value[1]	0	WORD
Events[1]		Event101M
Session	0	WORD
Sector	0	WORD
COT	3	WORD
Reserved	0	WORD
PointIndex		ARRAY[0..1] OF WORD
ASDUType	30	WORD
Milliseconds	866	UINT
Seconds	31	UINT
Minutes	55	BYTE
Hours	11	BYTE
Month	5	BYTE
Day	9	BYTE
Year	2008	WORD
Qualifier	1	WORD

In this example, the remote device has sent two events with timestamp to the module (in same block number). The following shows an example of how the variables associated to the EVENTFB function block would be updated.

- BlkCount: shows a value of 2 because the processor has received two blocks 9903
- LstPoint: shows a value of 3 because the last element that was updated has an index of 3.
- Events[0]: shows the first event received from the module
- Events[3]: shows the Last event received from the module



Name	Value	Type
ResetEP1	0	INT
BlockCount	2	INT
LastPoint	3	INT
Events		ARRAY[0..199] OF Event101M
Events[0]		Event101M
Session	0	WORD
Sector	0	WORD
COT	3	WORD
Reserved	0	WORD
PointIndex		ARRAY[0..1] OF WORD
PointIndex[0]	100	WORD
PointIndex[1]	0	WORD
ASDUType	30	WORD
Milliseconds	866	UINT
Seconds	31	UINT
Minutes	55	BYTE
Hours	11	BYTE
Month	5	BYTE
Day	9	BYTE
Year	2008	WORD
Qualifier	1	WORD
Value		ARRAY[0..1] OF WORD
Value[0]	1	WORD
Value[1]	0	WORD
Events[1]		Event101M
Events[2]		Event101M
Events[3]		Event101M
Session	0	WORD
Sector	0	WORD
COT	3	WORD
Reserved	0	WORD
PointIndex		ARRAY[0..1] OF WORD
PointIndex[0]	103	WORD
PointIndex[1]	0	WORD
ASDUType	30	WORD
Milliseconds	189	UINT
Seconds	38	UINT
Minutes	10	BYTE

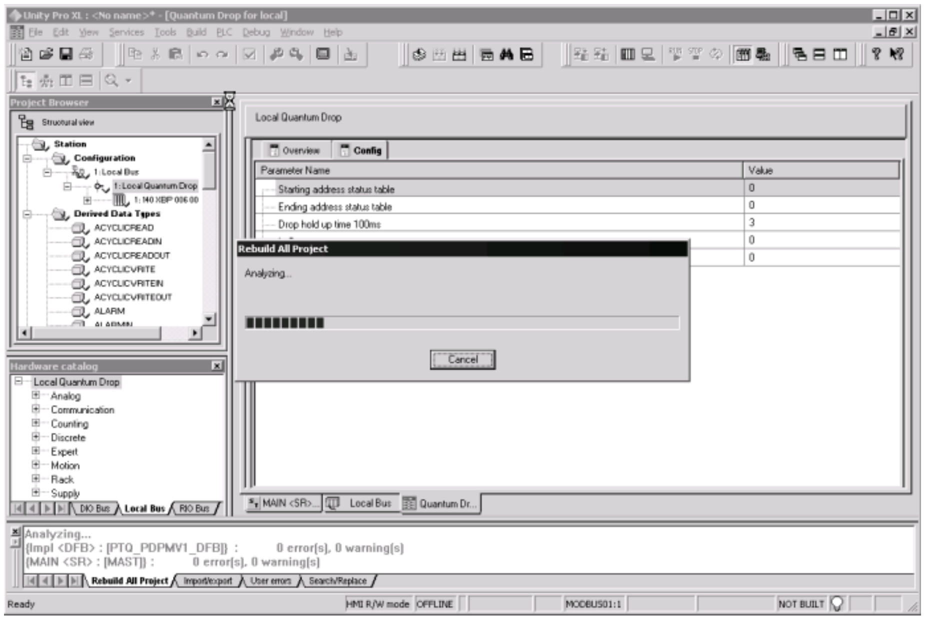
4.4 Build the Project

Whenever you update the configuration of your PTQ module or the processor, you must import the changed configuration from the module, and then build (compile) the project before downloading it to the processor.

Note: The following steps show you how to build the project in Unity Pro XL. This is not intended to provide detailed information on using Unity Pro XL, or debugging your programs. Refer to the documentation for your processor and for Unity Pro XL for specialized information.

To build (compile) the project:

1 Review the elements of the project in the Project Browser.
2 When you are satisfied that you are ready to download the project, open the Build menu, and then choose Rebuild all Project. This action builds (compiles) the project into a form that the processor can use to execute the instructions in the project file. This task may take several minutes, depending on the complexity of the project and the resources available on your PC.
3 As the project is built, Unity Pro XL reports its process in a Progress dialog box, with details appearing in a pane at the bottom of the window. The following illustration shows the build process under way.



Parameter Name	Value
Starting address status table	0
Ending address status table	0
Drop hold up time 100ms	3
	0
	0

Rebuild All Project

Analyzing...

```
Analyzing...
{Impl <DFB> : [PTQ_PDPMV1_DFB]} :        0 error(s), 0 warning(s)
{MAIN <SR> : [MAST]} :        0 error(s), 0 warning(s)
```

Ready | HMI R/W mode | OFFLINE | MODBUS01:1 | NOT BUILT

After the build process is completed successfully, the next step is to download the compiled project to the processor.

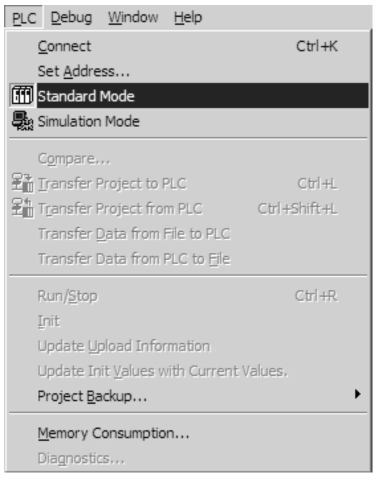
4.5 Connect Your PC to the Processor

The next step is to connect to the processor so that you can download the project file. The processor uses this project file to communicate over the backplane to modules identified in the project file.

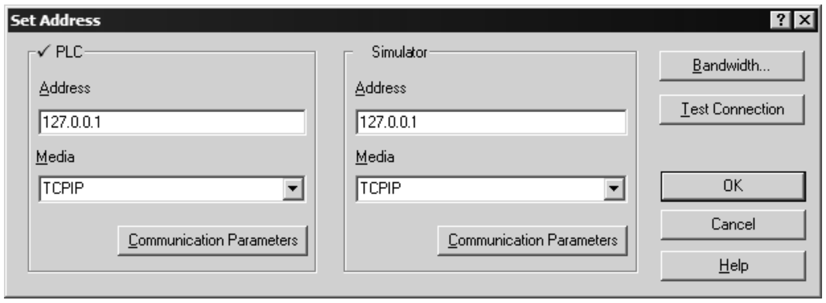
Note: If you have never connected from the PC to your processor before, you must verify that the necessary port drivers are installed and available to UnityPro XL.

To verify address and driver settings in UnityPro XL:

1 Open the PLC menu, and choose Standard Mode. This action turns off the PLC Simulator, and allows you to communicate directly with the Quantum or Unity hardware.


PLC
Debug
Window
Help
Connect Ctrl+K
Set Address...
Standard Mode
Simulation Mode
Compare...
Transfer Project to PLC Ctrl+L
Transfer Project from PLC Ctrl+Shift+L
Transfer Data from File to PLC
Transfer Data from PLC to File
Run/Stop Ctrl+R
Init
Update Upload Information
Update Init Values with Current Values.
Project Backup...
Memory Consumption...
Diagnostics...

2 Open the PLC menu, and choose Set address... This action opens the Set address dialog box. Open the Media dropdown list and choose the connection type to use (TCPIP or USB).

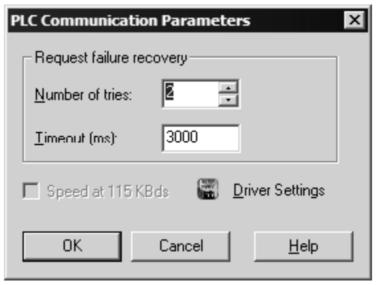


Set Address

✓ PLC
Address: 127.0.0.1
Media: TCPIP

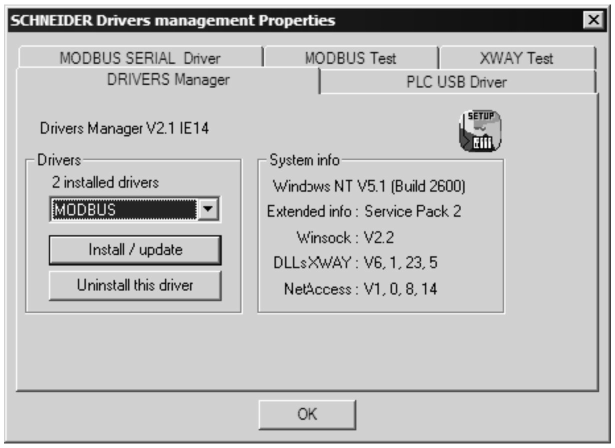
Simulator
Address: 127.0.0.1
Media: TCPIP

3 If the Media dropdown list does not contain the connection method you wish to use, click the Communication Parameters button in the PLC area of the dialog box. This action opens the PLC Communication Parameters dialog box.


PLC Communication Parameters

Request failure recovery
Number of tries: 2
Timeout (ms): 3000

4 Click the Driver Settings button to open the SCHNEIDER Drivers management Properties dialog box.

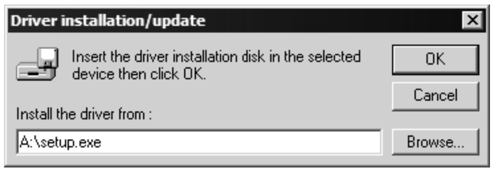

SCHNEIDER Drivers management Properties

Drivers Manager V2.1 IE14

Drivers
2 installed drivers
MODBUS

System info
Windows NT V5.1 (Build 2600)
Extended info : Service Pack 2
Winsock : V2.2
DLLsXWAY : V6, 1, 23, 5
NetAccess : V1, 0, 8, 14

5 Click the Install/update button to specify the location of the Setup.exe file containing the drivers to use. You will need your UnityPro XL installation disks for this step.


Driver installation/update

Insert the driver installation disk in the selected device then click OK.

Install the driver from :
A:\setup.exe

6 Click the Browse button to locate the Setup.exe file to execute, and then execute the setup program. After the installation, restart your PC if you are prompted to do so. Refer to your Schneider Electric documentation for more information on installing drivers for UnityPro XL.

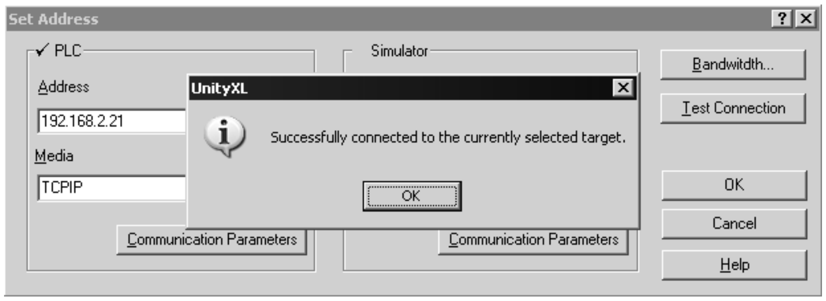
4.5.1 Connecting to the Processor with TCPIP

The next step is to download (copy) the project file to the processor. The following steps demonstrate how to use an Ethernet cable connected from the Processor to your PC through an Ethernet hub or switch. Other connection methods may also be available, depending on the hardware configuration of your processor, and the communication drivers installed in UnityPro XL.

1 If you have not already done so, connect your PC and the processor to an Ethernet hub.
2 Open the PLC menu, and then choose Set address.

- **Important:** Notice that the Set address dialog box is divided into two areas. Enter the address and media type in the PLC area of the dialog box, not the Simulator area.

3 Enter the IP address in the address field. In the Media dropdown list, choose TCPIP.
4 Click the Test Connection button to verify that your settings are correct.

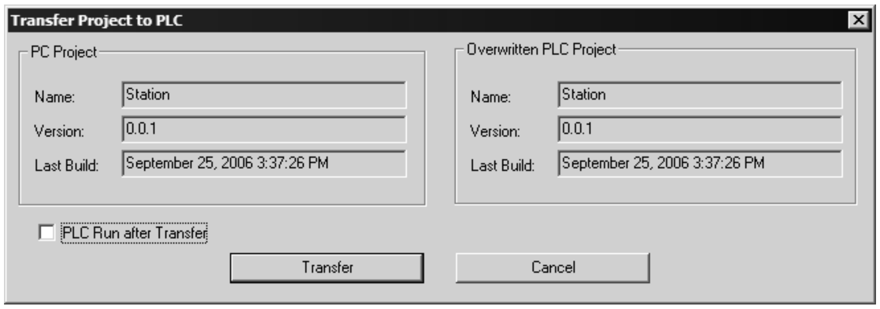


Successfully connected to the currently selected target.

The next step is to download the Project to the Processor.

4.6 Download the Project to the Processor

1 Open the PLC menu and then choose Connect. This action opens a connection between the Unity Pro XL software and the processor, using the address and media type settings you configured in the previous step.
2 On the PLC menu, choose Transfer Project to PLC. This action opens the Transfer Project to PLC dialog box. If you would like the PLC to go to "Run" mode immediately after the transfer is complete, select (check) the PLC Run after Transfer check box.



PC Project		Overwritten PLC Project	
Name:	Station	Name:	Station
Version:	0.0.1	Version:	0.0.1
Last Build:	September 25, 2006 3:37:26 PM	Last Build:	September 25, 2006 3:37:26 PM

3 Click the Transfer button to download the project to the processor. As the project is transferred, Unity Pro XL reports its process in a Progress dialog box, with details appearing in a pane at the bottom of the window.

When the transfer is complete, place the processor in Run mode.

5 Module Configuration

In This Chapter

5.1 Installing and Configuring the Module

This chapter describes how to install and configure the module to work with your application. The configuration process consists of the following steps.

1 Use to identify the module to the processor and add the module to a project.

Note: The software must be in "offline" mode to add the module to a project.

2 Modify the example ladder logic to meet the needs of your application, and copy the ladder logic to the processor. Example ladder logic files are provided on the CD-ROM.

Note: If you are installing this module in an existing application, you can copy the necessary elements from the example ladder logic into your application.

The rest of this chapter describes these steps in more detail.

5.2 Configuration File

In order for the module to operate, a configuration file (IEC101M.CFG) is required. This configuration file contains all the information required to configure the module's master drivers, set up the databases for the controlled devices and established a command list. Each parameter in the file must be set carefully in order for the application to be implemented successfully. The Reference chapter contains an example listing of a IEC101M.CFG file.

The configuration file is separated into sections, with topic header names enclosed in the **[ ]** characters. The configuration file consists of the following sections:

[Section]	**Description**
[Backplane Configuration]	Backplane transfer parameter section
[IEC-870-5-101 Master]	General Configuration for driver
[IEC-870-5-101 Master Port 0]	Configuration for first application port
[IEC-870-5-101 Master Port 1]	Configuration for second application port
[IEC-101 Master Session x]	Definition for each control unit

[Section]	Description
[IEC-101 Master Session x Sector y]	Definition for each sector in each controlled unit
[IEC-101 Master Commands]	Command list to control slave units

After each section header, the file contains a set of parameters. Unique labels are used under each section to specify a parameter. Each label in the file must be entered exactly as shown in the file for the parameter to be identified by the program. If the module is not considering a parameter, look at the label for the data item. Each parameter's value is separated from the label with the '**:**' character. This character is used by the program to delimit the position in the data record where to start reading data. All data for a parameter must be placed after the '**:**' character. For numeric parameter values any text located after the value will not be used. There must be at least one space character between the end of the parameter value and the following text. An example of a parameter entry is given below:

```
Baud Rate: 38400 #Baud rate for master port
```

The parameter label is "Baud Rate" and the parameter value is 38400. The characters after the parameter value are ignored and are used for internal documentation of the configuration file.

Any record that begins with the '#' character is considered to be a comment record. These records can be placed anywhere in the file as long as the '#' character is found in the first column of the line. These lines are ignored in the file and can be used to provide documentation within the configuration file. Liberal use of comments within the file can ease the use and interpretation of the data in the file.

Use any text editor to alter the supplied IEC101M.CFG file for the specific application. You must enter each parameter correctly for successful application of the module. The Reference chapter contains a complete listing of all parameters utilized by the module with a definition of each parameter.

5.2.1 [Backplane Configuration]

This section provides the module with:

- a unique name,
- designates database addresses for input and output on the module and on the processor,
- identifies the method of failure for the communications for the module if the PLC is not in run mode
- describes how to initialize the module upon startup.

The following example shows a sample [Backplane Configuration] section:

```
[Backplane Configuration]
Module Name: PTQ-101M SAMPLE TEST MODULE
#These values are required to define the data area to transfer between the
#module and the processor.
Read Register Start :   0    #Database start register to move to processor
Read Register Count :   50   #Number of words moved from module to
                             #processor
Write Register Start:   1000 #Database start register where data placed
                             #from processor
Write Register Count:   50   #Number of words moved from processor to
                             #module
#Used to define the area in the Processor for the module to interface with
3x Register Start:        1  #3x start register where data moved from
                             #module to processor (1-n)
4x Register Start:        1  #4x start register where data moved from
                             #processor to module (1-n)
Pass-Through Events  :    N  #Pass event messages to processor
```

Modify each of the parameters based on the needs of your application.

Module Name

0 to 80 characters

This parameter assigns a name to the module that can be viewed using the configuration/debug port. Use this parameter to identify the module and the configuration file.

Read Register Start

Range 0 to 3999

This parameter specifies the starting register in the module where the data transferred from the processor will be placed. Valid range for this parameter is 0 to 3999.

Read Register Count

Range 0 to 3999

This parameter specifies the number of registers to be transferred from the module to the processor. Valid entry for this parameter is 0 to 3999.

Write Register Start

0 to 3999

The Write Register Start parameter assigns the starting address for data to retrieve from the processor.

Write Register Count

Range 0 to 4000

This parameter specifies the number of registers to be transferred from the module to the processor. Valid entry for this parameter is 0 to 4000.

3x Register Start

1 to n

The 3x Register Start parameter defines the starting address in the processor's 3x (Quantum) or %iw (Unity) memory area to use for data being moved from the module. Take care to use a starting address that will accommodate the entire block from the module, but that will not overwrite data that is used for other purposes.

4x Register Start

1 to n

The 4x Register Start parameter defines the starting address in the processor's 4x (Quantum) or %iw (Unity) memory area to use for data being moved from the processor to the module. Take care to use a starting address that does not contain data in the processor's registers that is used for other purposes.

Pass-Through Events

Y or N (N = Default)

This parameter specifies if event messages received on the master ports will be passed to the processor. If the parameter is set to N, event messages will not be passed to the processor. If the parameter is set to Y, the module will pass all events received to the processor using block identifier 9903.

5.2.2 [IEC-870-5-101 Master]

This is the configuration for the IEC-870-5-101 master port emulated on the module.

Session Count

1 to 32

This parameter specifies the maximum number of sessions to establish on the module. This corresponds to the number of slaves to be interfaced with the module. This value represents the total number of slaves on all ports.

5.2.3 [IEC-870-5-101 Master Port x]

Baud Rate

This parameter specifies the baud rate to be used on the communication channel (port). Values from 110 to 38.4K are permitted.

Parity

None, Odd, Even

This parameter specifies the parity for this port using the following code definitions: N=none, O=odd, E=even.

RTS On

0 to 65535

The parameter sets the RTS pre-send delay. The value entered represents the number of milliseconds the module will wait after setting the RTS modem line before sending the data.

RTS Off

1 to 65535

This parameter sets the RTS off delay. The value entered represents the number of milliseconds the module will wait after the data packet is sent before dropping the RTS modem line.

Minimum Delay

1 to 65535

This parameter specifies the minimum number of milliseconds to delay before sending the message (setting RTS high). This can be used when the serial network requires time for units to turn off their transmitters.

Receive Timeout

1 to 65535

This value represents the number of milliseconds to wait on a port from the time the first character is received until the last character in the longest message received on the port. This parameter should be set dependent on the baud rate. A value of 2000 should work with most applications.

Single Char ACK F0, 1 or 3

Yes or No

If set to Y, a single character ACK (0xE5) will be sent instead of a fixed length ACK (secondary function code 0) in response to a primary link function code 0, 1 or 3 if there is no access demand for class 1 data (ACD=1). If set to N, the fixed length ACK will be sent.

Use Balanced Mode

Yes or No

This parameter specifies if the port will use balanced mode. If balanced mode is used, only one controlled station will be permitted on the port. If unbalanced mode is used, multiple controlled stations can be used on a port. Select Yes to use balanced mode and No to use unbalanced mode.

5.2.4 [IEC-101 Master Session x]

This section is used to define session x which runs on Port x. The session sections of the configuration file are determined by the number of sessions set in the configuration file. The sessions are referenced by a zero based index value. For example, if the module is configured for four sessions, the configuration file should contain sections for sessions 0 to 3 (that is, [IEC-101 Master Session 0] to [IEC-101 Master Session 3]. Each of these sections will define the characteristics of the specific controlled device to be interfaced.

Communication Port

0 or 1

This parameter sets the port to which the controlled device is connected. On this module, values of 0 and 1 are permitted.

Sector Count

1 to 5

This parameter sets the number of sectors contained in this controlled device. The range of values is from 1 to 5. A sector section is required for each sector in a session to define its database and settings.

Data Link Address

0 to 254 or 0 to 65534

This parameter uniquely defines the data link address for this unit on the communication channel The ranges of values depends on the value set in the DL Address Length parameter.

Common address of ASDU Len

1 or 2

This parameter specifies the number of octets used for the common address of ASDU. This parameter must be set the same for all devices on the network.

Inform. Object address Len

1, 2 or 3

This parameter sets the number of octets used to specify the address for an information object in each sector for this session.

COT octet count

1 or 2

This parameter sets the number of octets used for the COT field in each message. If a value of 2 is selected, the value entered for the Originator Address For COT will accompany each message from the controlling unit.

Originator address for COT

0 to 255

This parameter sets the address to be passed with each message when the COT Octet Count parameter is set to 2.

Failure Delay

0 to 2000

This parameter sets the minimum number of seconds to delay before polling this session when it is not online. This parameter is only used in unbalanced mode.

Confirm Timeout

0 to 2^32-1

This parameter sets the number of milliseconds to wait for a confirm response from the controlled device.

Retry Count

0 to 255

This parameter sets the number of retries to be performed on the controlled device when a communication occurs.

C1/C2 Poll Count Pend

0 to 65535

This parameter sets the maximum number of class 1 and class 2 polls performed on this session before trying the next session. This parameter prevents a session from monopolizing the communication port.

Class 1 Polls

0 to 65535

This parameter sets the maximum number of class 1 polls performed on this session before switching to another session. This parameter prevents a session from monopolizing the communication port.

Class 1 Pend Delay

0 to 2^32-1

This parameter sets the minimum number of milliseconds to delay between class 1 polls for pending data.

Class 2 Pend Delay

0 to 2^32-1

This parameter sets the minimum number of milliseconds to delay between class 2 polls for pending data.

Class 1 Poll Delay

0 to 2^32-1

This parameter sets the minimum number of milliseconds to delay between each class 1 poll.

Class 2 Poll Delay

0 to 2^32-1

This parameter sets the minimum number of milliseconds to delay between each class 2 poll.

Auto Clock Req Mode

0=Sync Only, 1=Load delay/sync, 2=Acquire delay/load delay/sync

This parameter specifies the method used to perform automatic clock synchronization. 0 performs a synchronization without delay, 1 performs synchronization using the fixed Propagation Delay and 2 computes the delay and use this value when synchronization takes place.

Propagation Delay

0 to 65535

This parameter sets the fixed propagation delay to be utilized if the Auto Clock Req Mode parameter is set to a value of 1.

Response Timeout

0 to 2^32-1

This parameter sets the maximum number of milliseconds to wait for a confirmation from the controlled station to a request from this module.

ACTTERM with setpoint

Yes or No

This parameter determines if an ACTTERM will be sent. If the parameter is set to Yes, then setpoint commands will issue an ACTTERM when the command is complete. If the parameter is set to No, ACTCON is the last response to a setpoint command.

5.2.5 [IEC-101 Master Session x Sector y]

This section sets the parameters for a specific sector of a session. Within each session definition, is a parameter that specifies the number of sectors for the session. For each sector defined for a session, there must exist a [IEC-101 Master Session x Sector y] section. Where the x value represents the session index and the y value represents sector index. For example if session 0 contains 1 sector, there must be a section with the following name in the configuration file: [IEC-101 Master Session 0 Sector 0]. The specific sector parameter set and database is defined in this section.

Common ASDU Address

0 to 255 (1 oct) or 0 to 65535 (2 oct)

This parameter sets the common ASDU address to association with this sector of the specified session. The range of address for this parameter are dependent on the length value set in the session section.

Use Time tag commands

Yes or No

This parameter specifies if a time tag field is to be included with commands. This is as specified in the IEC-870-5-104 specification and should only be utilized if the controlled device supports these new data types. If the parameter is set to Yes, a time tag will be added to all commands. If the parameter is set to No, the normal IEC 60870-5-101 data type messages will be utilized.

Online Time Sync.

Yes or No

This parameter specifies if the sector in the controlled device will be sent a time synchronization command when the unit is first recognized as being online. This should only be used for devices that do not send an EOI message after initializing.

Online General Int

Yes or No

This parameter specifies if the sector in the controlled device will be sent a general interrogation command when the unit is first recognized as being online. This should only be used for devices that do not send an EOI message after initializing.

EOI Time Sync.

Yes or No

This parameter specifies if the sector in the controlled device will be sent a time synchronization command after this module received an EOI message from the controlled unit.

EOI General Int

Yes or No

This parameter specifies if the sector in the controlled device will be sent a general interrogation command after this module received an EOI message from the controlled unit.

Database Definition

Database definition for this session/sector.

Data Types are as follows:

Monitored Data

- 1 = Single point
- 3 = Double point
- 5 = Step point
- 7 = Bitstring of 32-bits
- 9 = Measured normalized points
- 11 = Measured scaled points
- 13 = Measured short float points
- 15 = Integrated totals
- 110 = Measured normalized parameter (word-addressing/1 point = 1 data word)
- 111 = Measured scaled parameter (word-addressing/1 point = 1 data word)
- 112 = Measured short float parameters (double-word-addressing/1 point = 2 data words)
- 240 = Integrated totals BCD format (3 word-addressing/1 point = 3 data words)

5.2.6 [IEC-101 Master Commands]

This section contains the commands for the module. This section can contain up to 1000 user defined commands to be executed by the module and sent to the controlled devices. There is no need to place Class 1 or Class 2 polls in the this list for the controlled devices as the master driver for each port will execute these automatically when the port is idle. In order for the port to be idle, make sure that there is idle time available and that the commands do not constantly utilize the ports. The command list section starts with a reserved label **START** and ends with the label **END**. Each row in the file corresponds to an individual command with the first character position in each row left blank (white space).

Enable Code

0 = Disabled
1 = Enabled with Poll Interval (seconds) utilized
2 = Conditional (executed when point in database changes)

This field defines whether or not the command is to be executed and under what conditions. If the parameter is set to 0, the command is disabled and will not be executed in the normal polling sequence. The command can be executed under the control of the PLC processor through the use of a Command Control block. Setting the parameter to a value of 1 for the command causes the command to be executed each scan of the command list if the Poll Interval Time is set to zero. If the Poll Interval time is set, the command will be executed, when the interval timer expires. If the parameter is set to 2, the command will execute only if the internal data associated with the command changes. This value is valid only for write commands.

Database Index

Database Index is the location in the module's database to use as the source for the data in the command. The data type (page 77) field determines the meaning of the index.

Poll Interval

This parameter specifies the minimum frequency at which the module should execute the command. The value is entered in units of seconds. For example, to execute a command every 10 seconds, enter a value of 10 in the field. A value of 0 for the parameter implies that the command should be executed every scan of the list.

Session Index

Session Index represents the session index in the module to associate with the command. This index is set when the session is read in from this file. The range of values for this field is 0 to 31.

Sector Index

Sector Index represents the sector index for the specific session. The range of values for this field is 0 to 4.

Data Type

Data type file represents the ASDU type as follows:

Type	**Description**	**DB Index type**
45	Single point command	Bit address
46	Double point command	Bit address
47	Regulating Step point command	Byte address
48	Setpoint, normalized point command	Word address
49	Setpoint, scaled point command	Word address
50	Setpoint, short float point command	Double-word address
51	Bitstring (32-bits) point command	Double-word address

Type	Description	DB Index type
100	Group interrogation command	NA
101	Counter interrogation command	NA
102	Read command	NA
103	Clock Synchronization	NA
104	Test command (101 standard)	NA
105	Reset process command	NA
107	Test command (104 standard)	NA
110	Parameter, normalized measured value	Word address
111	Parameter, scaled measured value	Word address
112	Parameter, short float value	Double-word address
113	Parameter activation command	NA
242	BCD integrated setpoint command	3 word address
255	Send a class 2 poll	NA

Point Index

Point Index field specifies the address in the remote slave device of the point to interact with.

Qualifier Parameter

The Qualifier Parameter field defined for a command is dependent on the data type used in the command. In order to compute the qualifier for a command, add all the values for the features to use with a command together to form a single number. This number should be entered in the command record. Each data set is discussed below:

Single Point, Double Point, and Regulating Setup

The format of the field for **Single Point (45)**, **Double Point (46)**, **and Regulating Step (47)** commands is as follows:

Single Point, Double Point and Regulating Step Point Commands

Bit	Single	Double	Step
0	Value	Control Value	Control Value
1	0		
2	Qualifier Code		
3			
4			
5			
6			
7	Select/Execute Code		
8	Deselect Code		
9	Use Override Value		
10 to 15	Not Utilized		

The value field for the different data types can be derived from the module's database or that set in the command. The User Override bit is utilized to select the source of the data value. The values for each data type are defined below:

Single Point Value:

- 0=Off
- 1=On

Double Point Value:

- 0=Not permitted
- 1=Off
- 2=On
- 3=Not Permitted

Regulating Point Value (Set by module using database value -1=next lower, 1=next higher unless override enabled):

- 0=Not permitted
- 1=Next step lower if database point is set to -1
- 2=Next step high if database point set to +1
- 3=Not Permitted

The Qualifier Code area defines the operation to perform as defined below:

Qualifier Code (Select one of the following):

- 0=No additional definition (slave dependent)
- 4=Short pulse duration
- 8=Long pulse duration
- 12=Persistent output

The Select/Execute area defines if the command should perform a direct execute or select before execute command sequence. The values for this field are as follows (Select one of the values for the following list):

- 0=Direct execution without select
- 128=Select executed followed by execute
- 256=Deselect command

The value field for the qualifier can be derived from the module's database or be that defined in the qualifier. If the override flag is used, the module will issue the command using the values contained in the qualifier defined for the command. If the override flag is not set, the module will use the value in the database to send to the controlled device. The values to use for the override flag are as follows:

- 0=Use value in database (value field should be set to zero for qualifier parameter)
- 512=Use override value for state (preferred when using block 9902 with value field set for command to execute)

Normalized, Scaled, and Short Float

The format of the field for **Normalized (48)**, **Scaled (49)**, **and Short Float (50)** setpoint command is as follows:

Normalized, Scaled and Short Float Setpoint Commands

Bit	Description
0	Select/Execute Code
1	Deselect Code
2 to 15	Not Utilized

The value read from database for point specified is used with this qualifier to build a command.

The Qualifier Parameter uses one of the following codes:

- 0=Direct execution without select
- 1=Select executed followed by execute
- 2=Deselect command

Bitstring for 32 Bits

The format of the field for **32-Bitstring (51)** setpoint command is as follows:

Bitstring of 32 Bit Command

Bit	Description
0 to 15	Not Utilized

The value read from database for point specified is used with this qualifier to build a command.

The Qualifier Parameter is not currently used to construct commands.

The format of the field for **Interrogation Command (100)** is as follows:

Interrogation Command

Bit	Description
0 to 7	Interrogation Group
8 to 15	Not Utilized

No database value is associated with the construction of this command.

The Qualifier Parameter used with this command defines the interrogation group to request. Only a single group can be requested in a single command. The codes to use for this field are as follows:

- 0=Not used
- 1 to 19 = Reserved by standard
- 20=Station interrogation (global)
- 21=Interrogation group 1
- 22=Interrogation group 2
- 23=Interrogation group 3
- 24=Interrogation group 4
- 25=Interrogation group 5
- 26=Interrogation group 6
- 27=Interrogation group 7
- 28=Interrogation group 8
- 29=Interrogation group 9
- 30=Interrogation group 10
- 31=Interrogation group 11
- 32=Interrogation group 12
- 33=Interrogation group 13
- 34=Interrogation group 14
- 35=Interrogation group 15
- 36=Interrogation group 16
- 37 to 63 = Reserved by standard
- 64 to 255 = Reserved for special use (private range)

Counter Interrogation

The format of the field for **Counter Interrogation Command (101)** is as follows:

Counter Interrogation Command

Bit	Description
0 to 5	Counter Interrogation Group
6 to 7	Freeze/Reset Qualifier
8 to 15	Not Utilized

No database value is associated with the construction of this command.

The Qualifier Parameter used with this command defines the counter interrogation group to request. Only a single group can be requested in a single command. The qualifier also contains the freeze/reset operation to be utilized with the command The codes to use for this field are as follows:

Counter Interrogation Group:

- 0=No counter requested
- 1=Request counter group 1
- 2=Request counter group 2
- 3=Request counter group 3
- 4=Request counter group 4
- 5=Request general counter group
- 6 to 31 = Reserved by standard
- 32 to 63 = Reserved for special use (private range)

Freeze/Reset Qualifier:

- 0=No freeze or reset
- 64=Counter freeze without reset
- 128=Counter freeze with reset
- 192=No freeze with counter reset

Read Command

The format of this field for the **Read (102)** command is as follows:

Read Command

Bit	Description
0 to 15	Not Utilized

No database value is associated with the construction of this command and no qualifier value is used in this release of the software.

Clock Synchronization

The format of this field for the **Clock Synchronization (103)** command is as follows:

Clock Synchronization Command

Bit	Description
0	Synchronization Mode
1	Qualifier
0 to 15	Not Utilized

No database value is associated with the construction of this command.

The Qualifier Parameter for this command has one of the following values:

- 0=Clock synchronization with out delay utilized
- 1=Synchronize clock with delay set
- 2=Measure delay, load delay then synchronize clock

Test Command

The format of this field for the **test command (104 and 107)** is as follows:

Test Command (both 101 and 104 versions)

Bit	Description
0 to 15	Not Utilized

No database value is associated with the construction of this command and no qualifier value is used in this release of the software.

Reset Process

The format of this field for the **Reset Process (105)** command is as follows:

Reset Process Command

Bit	Description
0 to 7	Reset Qualifier
8 to 15	Not Utilized

No database value is associated with the construction of this command.

The Qualifier Parameter has one of the following values as define in the protocol specification:

- 0=Not used
- 1=General reset of process
- 2=Reset pending information with time tag of the event buffer
- 3 to 127 = Reserved by standard
- 128 to 255 = Reserved for special use (private range)

Parameter Setting

The format of this field for the **Parameter Setting (110=Normalized, 111=Scaled, 112=Short float)** is as follows:

Parameter Setting (Normalized, Scaled, Short Float) Command

Bit	Description
0 to 5	Kind of Parameter
6	Local change
7	Operation
8 to 15	Not Utilized

The value from module's database utilized to build the command.

The Qualifier Parameter used with this command is determined by summing the options from lists that follow:

Kind of parameter:

- 0=Not used
- 1=Threshold value
- 2=Smoothing factor (filter time constant)
- 3=Low limit for transmission of measured values
- 3=High limit for transmission of measured values
- 5 to 31 = Reserved by standard
- 32 to 63 = Reserved for special use (private range)

Local parameter change:

- 0=No change
- 64=Change

Parameter in operation:

- 0=Operation
- 128=Not in operation

Parameter Activation

The format of this field for the **Parameter Activation (113)** is as follows:

Parameter Activation Command

Bit	Description
0 to 7	Parameter Qualifier
8	Activation Qualifier
9 to 15	Not Utilized

No database value used with the construction of this command.

The Qualifier Parameter used with the command is determined by summing the options from the lists that follow:

Parameter Qualifier:

- 0=Not used
- 1=Act/Deact of previously loaded parameters (point index = 0)
- 2=Act/Deact of the parameter of the point index specified
- 3=Act/Deact of persistent cyclic or periodic transmission of the addressed object
- 4 to 127 = Reserved by standard
- 128 to 255 = Reserved for special use (private range)

Activation Qualifier:

- 0=Deactivate
- 256=Activate

BCD Integrated Setpoint

The format of this field for **BCD Integrated Setpoint (242)** command is as follows:

BCD Integrated total Setpoint Command

Bit	Description
0	Select/Execute Code
1	Deselect Code
2 to 15	Not Utilized

The value in database is utilized for this command . The data resides in a 6-byte data area in the module.

The Qualifier Parameter used with this command is selected from the following list:

- 0=Direct execution without select
- 1=Select executed followed by execute
- 2=Deselect command

The format of this field for the Class 2 poll (255) command is as follows:

Class 2 Poll

Bit	Description
0 to 15	Not Utilized

No database or qualifier is used with this command.

5.3 Uploading and Downloading the Configuration File

ProSoft modules are shipped with a pre-loaded configuration file. In order to edit this file, you must transfer the file from the module to your PC. After editing, you must transfer the file back to the module.

This section describes these procedures.

Important: The illustrations of configuration/debug menus in this section are intended as a general guide, and may not exactly match the configuration/debug menus in your own module. For specific information about the configuration/debug menus in your module, refer to The Configuration/Debug Menu (page 93).

5.3.1 Required Hardware

You can connect directly from your computer's serial port to the serial port on the module to view configuration information, perform maintenance, and send (upload) or receive (download) configuration files.

ProSoft Technology recommends the following minimum hardware to connect your computer to the module:

- 80486 based processor (Pentium preferred)
- 1 megabyte of memory
- At least one UART hardware-based serial communications port available. USB-based virtual UART systems (USB to serial port adapters) often do not function reliably, especially during binary file transfers, such as when uploading/downloading configuration files or module firmware upgrades.
- A null modem serial cable.

5.3.2 Required Software

In order to send and receive data over the serial port (COM port) on your computer to the module, you must use a communication program (terminal emulator).

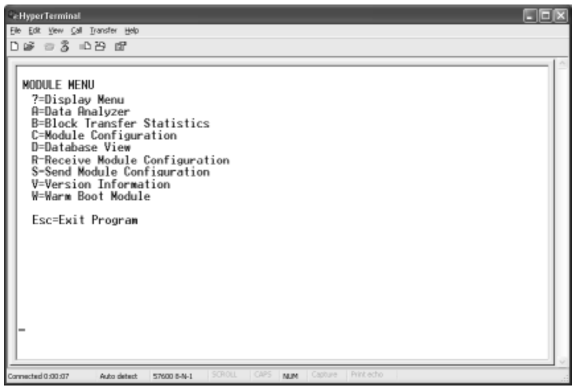
A simple communication program called HyperTerminal is pre-installed with recent versions of Microsoft Windows operating systems. If you are connecting from a machine running DOS, you must obtain and install a compatible communication program. The following table lists communication programs that have been tested by ProSoft Technology.

DOS	ProComm, as well as several other terminal emulation programs
Windows 3.1	Terminal
Windows 95/98	HyperTerminal
Windows NT/2000/XP	HyperTerminal

The module uses the Zmodem file transfer protocol to send (upload) and receive (download) configuration files from your module. If you use a communication program that is not on the list above, please be sure that it supports Zmodem file transfers.

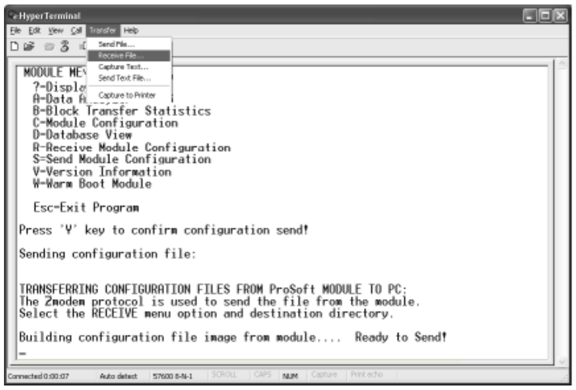
5.3.3 Transferring the Configuration File to Your PC

1 Connect your PC to the Configuration/Debug port of the module using a terminal program such as HyperTerminal. Press **[?]** to display the main menu.


HyperTerminal
File Edit View Call Transfer Help
Connected 0:00:07 Auto detect 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

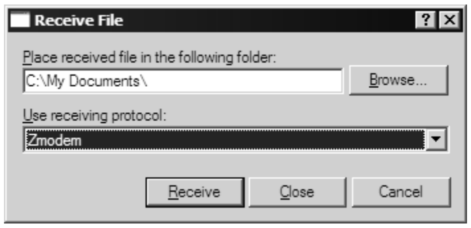
```
MODULE MENU
 ?=Display Menu
 A=Data Analyzer
 B=Block Transfer Statistics
 C=Module Configuration
 D=Database View
 R=Receive Module Configuration
 S=Send Module Configuration
 V=Version Information
 W=Warm Boot Module

 Esc=Exit Program
```

2 From the **Transfer** menu in HyperTerminal, select **Receive File**.


HyperTerminal
File Edit View Call Transfer Help
Send File...
Receive File...
Capture Text...
Send Text File...
Capture to Printer
Connected 0:00:07 Auto detect 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

```
MODULE ME
 ?=Displa
 A=Data A
 B=Block Transfer Statistics
 C=Module Configuration
 D=Database View
 R=Receive Module Configuration
 S=Send Module Configuration
 V=Version Information
 W=Warm Boot Module

 Esc=Exit Program
Press 'Y' key to confirm configuration send!

Sending configuration file:


TRANSFERRING CONFIGURATION FILES FROM ProSoft MODULE TO PC:
The Zmodem protocol is used to send the file from the module.
Select the RECEIVE menu option and destination directory.

Building configuration file image from module....  Ready to Send!
```

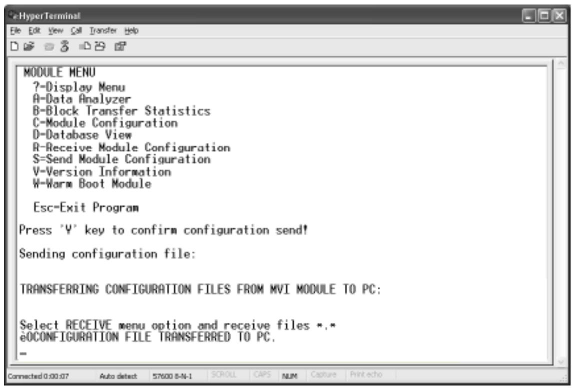
3 In the Receive File dialog box, browse to the location on your PC where the configuration file should be stored, and select Zmodem (or Zmodem with Crash Recovery) as the receiving protocol.



Place received file in the following folder: C:\My Documents\
Use receiving protocol: Zmodem

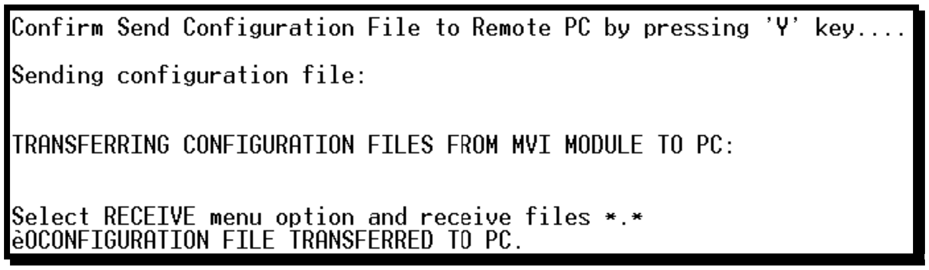
When you have completed your selections, click Close.

4 Press **[S]** (Send Module Configuration), and then press **[Y]** to confirm the transfer.



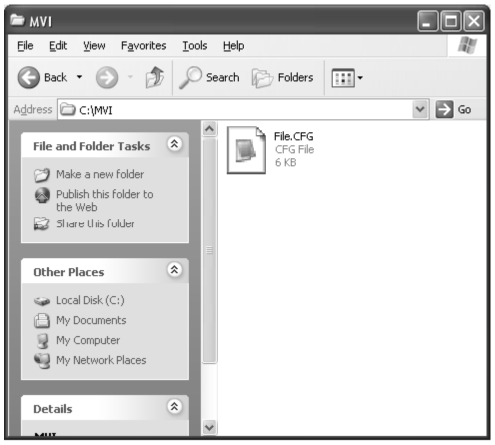
```
MODULE MENU
  ?-Display Menu
  A-Data Analyzer
  B-Block Transfer Statistics
  C-Module Configuration
  D-Database View
  R-Receive Module Configuration
  S=Send Module Configuration
  V-Version Information
  W-Warm Boot Module

  Esc=Exit Program
Press 'Y' key to confirm configuration send!

Sending configuration file:


TRANSFERRING CONFIGURATION FILES FROM MVI MODULE TO PC:


Select RECEIVE menu option and receive files *.*
èOCONFIGURATION FILE TRANSFERRED TO PC.
_
```

The file transfer will then begin automatically, using the protocol and location you specified in Step 3.
When the configuration file has been transferred to your PC, the dialog box will indicate that the transfer is complete.



```
Confirm Send Configuration File to Remote PC by pressing 'Y' key....

Sending configuration file:


TRANSFERRING CONFIGURATION FILES FROM MVI MODULE TO PC:


Select RECEIVE menu option and receive files *.*
èOCONFIGURATION FILE TRANSFERRED TO PC.
```

The configuration file is now on your PC at the location you specified.

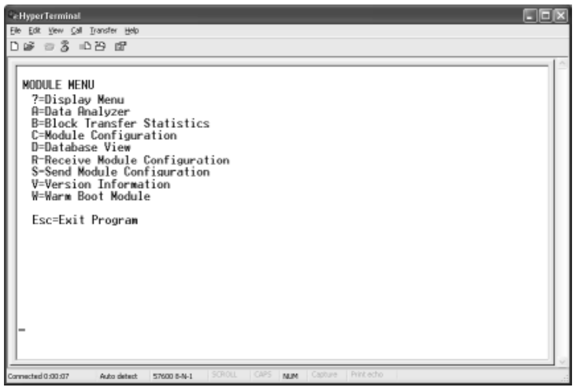


5 You can now open and edit the file in a text editor such as Notepad. When you have finished editing the file, save it and close Notepad.

5.3.4 Transferring the Configuration File to the Module

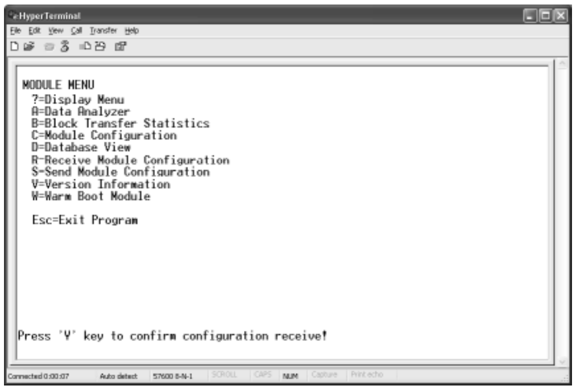
Perform the following steps to transfer a configuration file from your PC to the module.

1 Connect your PC to the Configuration/Debug port of the module using a terminal program such as HyperTerminal. Press **[?]** to display the main menu.



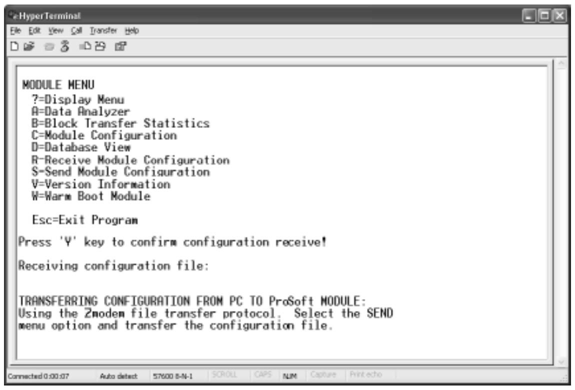
```
MODULE MENU
 ?=Display Menu
 A=Data Analyzer
 B=Block Transfer Statistics
 C=Module Configuration
 D=Database View
 R=Receive Module Configuration
 S=Send Module Configuration
 V=Version Information
 W=Warm Boot Module

 Esc=Exit Program
```

2 Press **[R]** (Receive Module Configuration). The message "Press Y key to confirm configuration receive!" is displayed at the bottom of the screen.



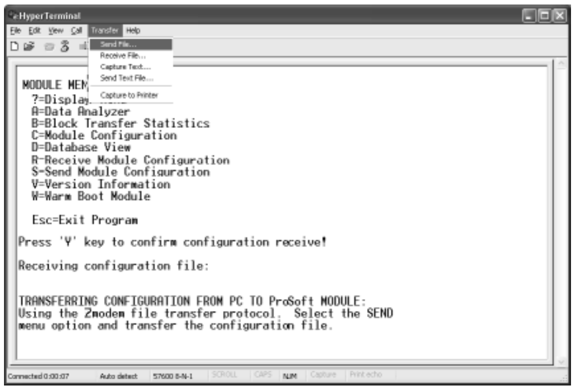
```
MODULE MENU
 ?=Display Menu
 A=Data Analyzer
 B=Block Transfer Statistics
 C=Module Configuration
 D=Database View
 R=Receive Module Configuration
 S=Send Module Configuration
 V=Version Information
 W=Warm Boot Module

 Esc=Exit Program

Press 'Y' key to confirm configuration receive!
```

3 Press **[Y]**. The screen now indicates that the PC is ready to send.



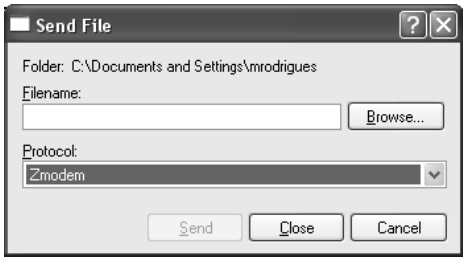
```
MODULE MENU
 ?=Display Menu
 A=Data Analyzer
 B=Block Transfer Statistics
 C=Module Configuration
 D=Database View
 R=Receive Module Configuration
 S=Send Module Configuration
 V=Version Information
 W=Warm Boot Module

 Esc=Exit Program
Press 'Y' key to confirm configuration receive!

Receiving configuration file:


TRANSFERRING CONFIGURATION FROM PC TO ProSoft MODULE:
Using the Zmodem file transfer protocol.  Select the SEND
menu option and transfer the configuration file.
```

4 From the **Transfer** menu in HyperTerminal, select **Send File**.


HyperTerminal
File Edit View Call Transfer Help
Send File...
Receive File...
Capture Text...
Send Text File...
Capture to Printer
Connected 0:00:07 Auto detect 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

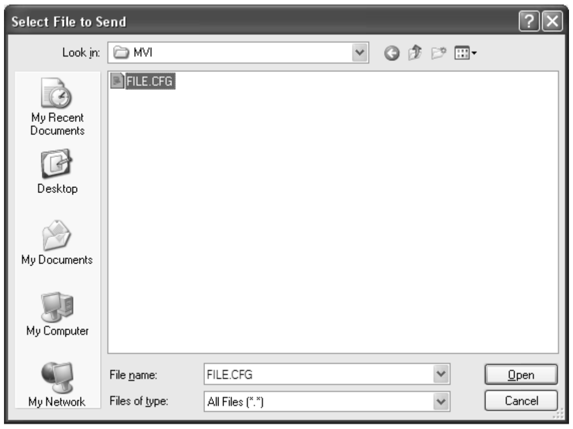
```
MODULE MEN
  ?=Displa
  A=Data Analyzer
  B=Block Transfer Statistics
  C=Module Configuration
  D=Database View
  R-Receive Module Configuration
  S-Send Module Configuration
  V=Version Information
  W=Warm Boot Module

  Esc=Exit Program

Press 'Y' key to confirm configuration receive!

Receiving configuration file:


TRANSFERRING CONFIGURATION FROM PC TO ProSoft MODULE:
Using the Zmodem file transfer protocol.  Select the SEND
menu option and transfer the configuration file.
```

The Send File dialog appears.



Folder: C:\Documents and Settings\mrodrigues
Filename:
Protocol: Zmodem

5 Use the Browse button to locate the configuration file your computer.

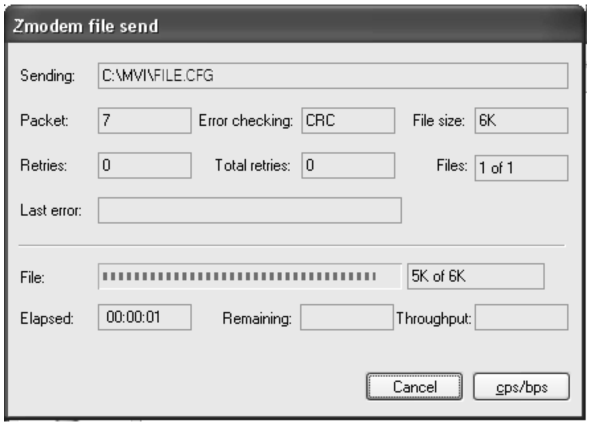


Look in: MVI
FILE.CFG
File name: FILE.CFG
Files of type: All Files (*.*)

Note: This procedure assumes that you are uploading a newly edited configuration file from your PC to the module. However, configuration files are also available on the ProSoft CD as well as the ProSoft Technology web site.

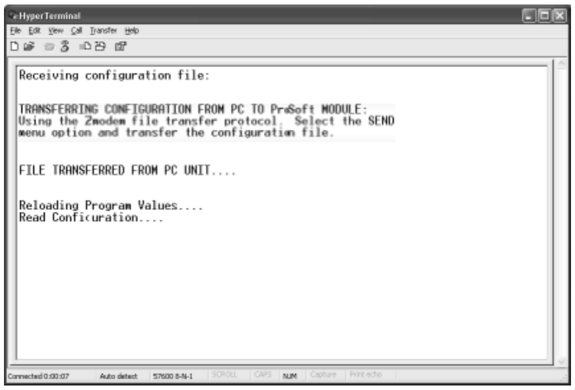
6 Select Zmodem as the protocol.

7 Click the Send button. This action opens the Zmodem File Send dialog box.



Sending: C:\MVI\FILE.CFG
Packet: 7 Error checking: CRC File size: 6K
Retries: 0 Total retries: 0 Files: 1 of 1
Last error:
File: 5K of 6K
Elapsed: 00:00:01 Remaining: Throughput:

When the upload is complete, the screen indicates that the module has reloaded program values and displays information about the module.


HyperTerminal
File Edit View Call Transfer Help
Connected 0:00:07 Auto detect 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

```
Receiving configuration file:

TRANSFERRING CONFIGURATION FROM PC TO ProSoft MODULE:
Using the Zmodem file transfer protocol. Select the SEND
menu option and transfer the configuration file.


FILE TRANSFERRED FROM PC UNIT....


Reloading Program Values....
Read Conficuration....
```

8 Your module now contains the new configuration.

6 Diagnostics and Troubleshooting

In This Chapter

The module provides information on diagnostics and troubleshooting in the following forms:

- Status data values are transferred from the module to the processor.
- Data contained in the module can be viewed through the Configuration/Debug port attached to a terminal emulator.
- LED status indicators on the front of the module provide information on the module's status.

6.1 The Configuration/Debug Menu

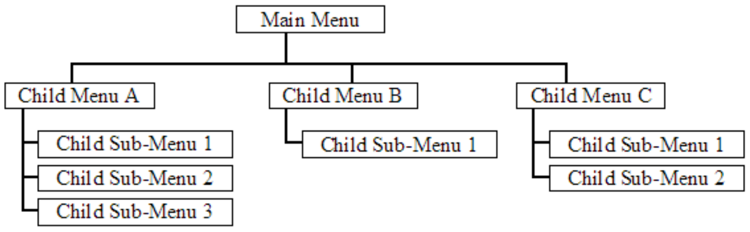
The Configuration and Debug menu for this module is arranged as a tree structure, with the Main Menu at the top of the tree, and one or more sub-menus for each menu command. The first menu you see when you connect to the module is the Main menu.

Because this is a text-based menu system, you enter commands by typing the command letter from your computer keyboard in the terminal application (for example, HyperTerminal). The module does not respond to mouse movements or clicks. The command executes as soon as you press the command letter — you do not need to press **[Enter]**. When you type a command letter, a new screen will be displayed in your terminal application.

6.1.1 Navigation

All of the sub-menus for this module contain commands to redisplay the menu or return to the previous menu. You can always return from a sub-menu to the next higher menu by pressing **[M]** on your keyboard.

The organization of the menu structure is represented in simplified form in the following illustration:



The remainder of this section shows you the menus available for this module, and briefly discusses the commands available to you.

6.1.2 Keystrokes

The keyboard commands on these menus are almost always non-case sensitive. You can enter most commands in lower case or capital letters.

The menus use a few special characters (**[?]**, **[-]**, **[+]**, **[@]**) that must be entered exactly as shown. Some of these characters will require you to use the **[Shift]**, **[Ctrl]** or **[Alt]** keys to enter them correctly. For example, on US English keyboards, enter the **[?]** command as **[Shift][/]**.

Also, take care to distinguish capital letter **[I]** from lower case letter **[l]** (L) and number **[1]**; likewise for capital letter **[O]** and number **[0]**. Although these characters look nearly the same on the screen, they perform different actions on the module.

6.2 Required Hardware

You can connect directly from your computer's serial port to the serial port on the module to view configuration information, perform maintenance, and send (upload) or receive (download) configuration files.

ProSoft Technology recommends the following minimum hardware to connect your computer to the module:

- 80486 based processor (Pentium preferred)
- 1 megabyte of memory
- At least one UART hardware-based serial communications port available. USB-based virtual UART systems (USB to serial port adapters) often do not function reliably, especially during binary file transfers, such as when uploading/downloading configuration files or module firmware upgrades.
- A null modem serial cable.

6.3 Required Software

In order to send and receive data over the serial port (COM port) on your computer to the module, you must use a communication program (terminal emulator).

A simple communication program called HyperTerminal is pre-installed with recent versions of Microsoft Windows operating systems. If you are connecting from a machine running DOS, you must obtain and install a compatible communication program. The following table lists communication programs that have been tested by ProSoft Technology.

DOS	ProComm, as well as several other terminal emulation programs
Windows 3.1	Terminal
Windows 95/98	HyperTerminal
Windows NT/2000/XP	HyperTerminal

The module uses the Zmodem file transfer protocol to send (upload) and receive (download) configuration files from your module. If you use a communication program that is not on the list above, please be sure that it supports Zmodem file transfers.

6.4 Using the Configuration/Debug Port

To connect to the module's Configuration/Debug port:

1 Connect your computer to the module's port using a null modem cable.
2 Start the communication program on your computer and configure the communication parameters with the following settings:

Baud Rate	57,600
Parity	None
Data Bits	8
Stop Bits	1
Software Handshaking	None

3 Open the connection. When you are connected, press the **[?]** key on your keyboard. If the system is set up properly, you will see a menu with the module name followed by a list of letters and the commands associated with them.

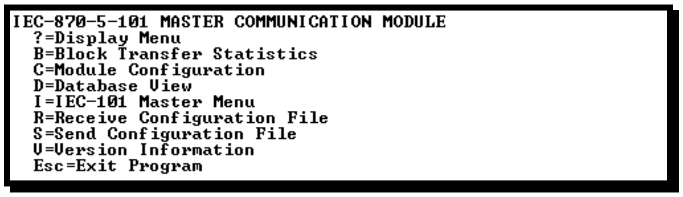
If there is no response from the module, follow these steps:

1 Verify that the null modem cable is connected properly between your computer's serial port and the module. A regular serial cable will not work.
2 Verify that another program is not controlling the COM port.
3 Verify that your communication software is using the correct settings for baud rate, parity and handshaking.
4 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

If you are still not able to establish a connection, you can contact ProSoft Technology Technical Support for further assistance.

6.4.1 Main Menu

When you first connect to the module from your computer, your terminal screen will be blank. To activate the main menu, press the [?] key on your computer's keyboard. If the module is connected properly, the following menu will appear on your terminal screen:



```
IEC-870-5-101 MASTER COMMUNICATION MODULE
  ?=Display Menu
  B=Block Transfer Statistics
  C=Module Configuration
  D=Database View
  I=IEC-101 Master Menu
  R=Receive Configuration File
  S=Send Configuration File
  V=Version Information
  Esc=Exit Program
```

Caution: Some of the commands available to you from this menu are designed for advanced debugging and system testing only, and can cause the module to stop communicating with the processor or with other devices, resulting in potential data loss or other failures. Only use these commands if you are specifically directed to do so by ProSoft Technology Technical Support staff. Some of these command keys are not listed on the menu, but are active nevertheless. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing Block Transfer Statistics

Press **[B]** from the Main Menu to view the Block Transfer Statistics screen.

Use this command to display the configuration and statistics of the backplane data transfer operations between the module and the processor. The information on this screen can help determine if there are communication problems between the processor and the module.

Tip: To determine the number of blocks transferred each second, mark the numbers displayed at a specific time. Then some seconds later activate the command again. Subtract the previous numbers from the current numbers and divide by the quantity of seconds passed between the two readings.

Viewing Module Configuration

Press **[C]** to view the Module Configuration screen.

Use this command to display the current configuration and statistics for the module.

Opening the Database Menu

Press **[D]** to open the Database View menu. Use this menu command to view the current contents of the module's database.

Opening the IEC-101 Master Menu

Press **[I]** from the Main Menu to open the IEC-870-5-101 Master Driver Menu. Use this menu command to view detailed configuration information for the module.

Transferring the Configuration File from PC to PTQ module

Press **[R]** to send (upload) the configuration file from your PC to the module and store the file on the module's Compact Flash Disk.

Press **[Y]** to confirm the file transfer, and then follow the instructions on the terminal screen to complete the file transfer process.

After the file has been successfully downloaded, the module will restart the program and load the new configuration information. Review the new configuration using menu commands **[6]** and **[0]** to verify that the module is configured correctly.

Transferring the Configuration File from PTQ module to PC

Press **[S]** to receive (download) the configuration file from the module to your PC.

Press **[Y]** to confirm the file transfer, and then follow the instructions on the terminal screen to complete the file transfer process.

After the file has been successfully downloaded, you can open and edit the file to change the module's configuration.

Viewing Version Information

Press **[V]** to view Version information for the module.

Use this command to view the current version of the software for the module, as well as other important values. You may be asked to provide this information when calling for technical support on the product.

Values at the bottom of the display are important in determining module operation. The Program Scan Counter value is incremented each time a module's program cycle is complete.

Tip: Repeat this command at one-second intervals to determine the frequency of program execution.

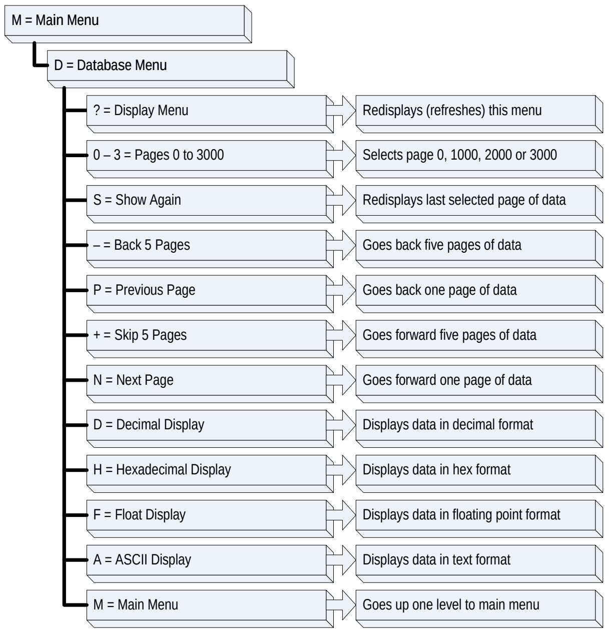
Exiting the Program

Caution: Some of the commands available to you from this menu are designed for advanced debugging and system testing only, and can cause the module to stop communicating with the processor or with other devices, resulting in potential data loss or other failures. Only use these commands if you are specifically directed to do so by ProSoft Technology Technical Support staff. Some of these command keys are not listed on the menu, but are active nevertheless. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

Press **[Esc]** to restart the module and force all drivers to be loaded. The module will use the configuration stored in the module's Flash ROM to configure the module.

6.4.2 Database View Menu

Press **[D]** from the Main Menu to open the Database View menu. Use this menu command to view the current contents of the module's database. Press **[?]** to view a list of commands available on this menu.


M = Main Menu
D = Database Menu
? = Display Menu
Redisplays (refreshes) this menu
0 – 3 = Pages 0 to 3000
Selects page 0, 1000, 2000 or 3000
S = Show Again
Redisplays last selected page of data
– = Back 5 Pages
Goes back five pages of data
P = Previous Page
Goes back one page of data
+ = Skip 5 Pages
Goes forward five pages of data
N = Next Page
Goes forward one page of data
D = Decimal Display
Displays data in decimal format
H = Hexadecimal Display
Displays data in hex format
F = Float Display
Displays data in floating point format
A = ASCII Display
Displays data in text format
M = Main Menu
Goes up one level to main menu

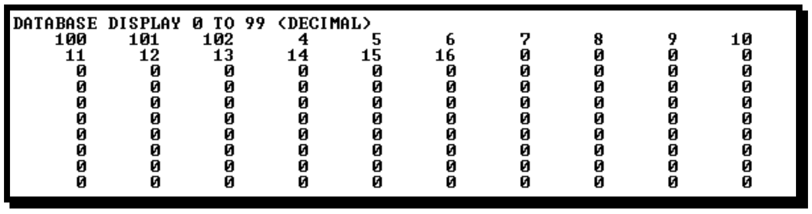
Viewing Register Pages

To view sets of register pages, use the keys described below:

Command	**Description**
[0]	Display registers 0 to 99
[1]	Display registers 1000 to 1099
[2]	Display registers 2000 to 2099

And so on. The total number of register pages available to view depends on your module's configuration.

Displaying the Current Page of Registers Again


```
DATABASE DISPLAY 0 TO 99 (DECIMAL)
    100     101     102       4       5       6       7       8       9      10
     11      12      13      14      15      16       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
```

This screen displays the current page of 100 registers in the database.

Moving Back Through 5 Pages of Registers

Press **[-]** from the Database View menu to skip back to the previous 500 registers of data.

Viewing the Previous 100 Registers of Data

Press **[P]** from the Database View menu to display the previous 100 registers of data.

Skipping 500 Registers of Data

Hold down **[Shift]** and press **[=]** to skip forward to the next 500 registers of data.

Viewing the Next 100 Registers of Data

Press **[N]** from the Database View menu to select and display the next 100 registers of data.

Viewing Data in Decimal Format

Press **[D]** to display the data on the current page in decimal format.

Viewing Data in Hexadecimal Format

Press **[H]** to display the data on the current page in hexadecimal format.

Viewing Data in Floating Point Format

Press **[F]** from the Database View menu. Use this command to display the data on the current page in floating point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they are not displayed properly.

Viewing Data in ASCII (Text) Format

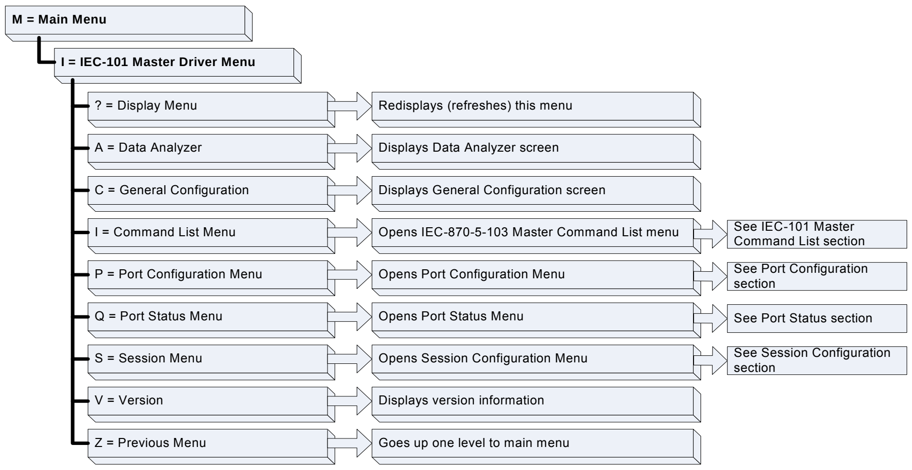
Press **[A]** to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

6.4.3 IEC-101M Master Menu

Press **[I]** from the Main Menu to open the ICE-870-5-101 Master Driver Menu. Use this menu command to view detailed configuration information for the module.


M = Main Menu
I = IEC-101 Master Driver Menu
? = Display Menu
Redisplays (refreshes) this menu
A = Data Analyzer
Displays Data Analyzer screen
C = General Configuration
Displays General Configuration screen
I = Command List Menu
Opens IEC-870-5-103 Master Command List menu
See IEC-101 Master Command List section
P = Port Configuration Menu
Opens Port Configuration Menu
See Port Configuration section
Q = Port Status Menu
Opens Port Status Menu
See Port Status section
S = Session Menu
Opens Session Configuration Menu
See Session Configuration section
V = Version
Displays version information
Z = Previous Menu
Goes up one level to main menu

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Opening the Data Analyzer Menu

Press **[A]** to open the Data Analyzer Menu. Use this command to view all bytes of data transferred on each port. Both the transmitted and received data bytes are displayed. Refer to Data Analyzer for more information about this menu.

Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press [S] to stop the data analyzer, and then press [M] to return to the main menu. This action will allow the module to resume its normal high speed operating mode.

Viewing Protocol Configuration

Press **[C]** to view configuration information for the 101M protocol.

Opening the Command List Menu

Press **[I]** to open the Command List menu. Use this command to view the configured command list for the module.

Viewing Port Configuration

Press **[P]** to view configuration information for the application port.

Use this command to display detailed configuration information for the port.

Viewing Port Communication Status

Press **[Q]** to view the port communication status for the application port.

Use this command to view communication status and statistics for the selected port. This information can be informative when trouble-shooting communication problems.

Opening the Session Configuration Menu

Press **[S]** to open the Session Configuration menu. Use this command to view the session configuration data.

Refer to *Session Configuration Menu* for more information about the commands on this menu.

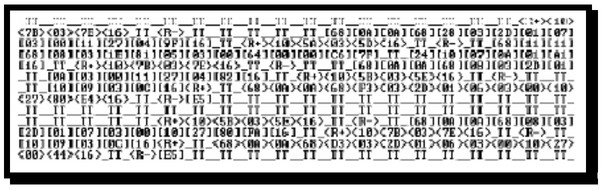
6.4.4 Data Analyzer

The data analyzer mode allows you to view all bytes of data transferred on each port. Both the transmitted and received data bytes are displayed. Use of this feature is limited without a thorough understanding of the protocol.

Note: The Port selection commands on the Data Analyzer menu differs very slightly in different modules, but the functionality is basically the same. Use the illustration above as a general guide only. Refer to the actual data analyzer menu on your module for the specific port commands to use.
Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press [S] to stop the data analyzer, and then press [M] to return to the main menu. This action will allow the module to resume its normal high speed operating mode.

Analyzing Data for the first application port

Press **[1]** to display I/O data for the first application port in the Data Analyzer. The following illustration shows an example of the Data Analyzer output.



```
_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_<R+><10>
<7B><03><7E><16>_TT_<R->_TT__TT__TT__TT__TT_[68][0A][0A][68][28][03][2D][01][07]
[03][00][11][27][04][9F][16]_TT_<R+><10><5A><03><5D><16>_TT_<R->_TT_[68][11][11]
[68][08][03][1E][81][05][03][00][64][00][00][C6][7F]_TT_[24][10][07][0A][01][A1]
[16]_TT_<R+><10><7B><03><7E><16>_TT_<R->_TT__TT_[68][0A][0A][68][08][03][2D][01]
_TT_[0A][03][00][11][27][04][82][16]_TT_<R+><10><5B><03><5E><16>_TT_<R->_TT__TT_
_TT_[10][09][03][0C][16]<R+>_TT_<68><0A><0A><68><F3><03><2D><01><06><03><00><10>
<27><00><E4><16>_TT_<R->[E5]_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_
_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_
_TT__TT__TT__TT__TT_<R+><10><5B><03><5E><16>_TT_<R->_TT_[68][0A][0A][68][08][03]
[2D][01][07][03][00][10][27][80][FA][16]_TT_<R+><10><7B><03><7E><16>_TT_<R->_TT_
[10][09][03][0C][16]<R+>_TT_<68><0A><0A><68><D3><03><2D><01><06><03><00><10><27>
<00><44><16>_TT_<R->[E5]_TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT__TT_
```

Analyzing Data for the second application port

Press **[2]** to display I/O data for the second application port in the Data Analyzer.

Displaying Timing Marks in the Data Analyzer

You can display timing marks for a variety of intervals in the data analyzer screen. These timing marks can help you determine communication-timing characteristics.

Key	Interval
[5]	1 milliseconds ticks
[6]	5 milliseconds ticks
[7]	10 milliseconds ticks
[8]	50 milliseconds ticks
[9]	100 milliseconds ticks
[0]	Turn off timing marks

Removing Timing Marks in the Data Analyzer

Press **[0]** to turn off timing marks in the Data Analyzer screen.

Viewing Data in Hexadecimal Format

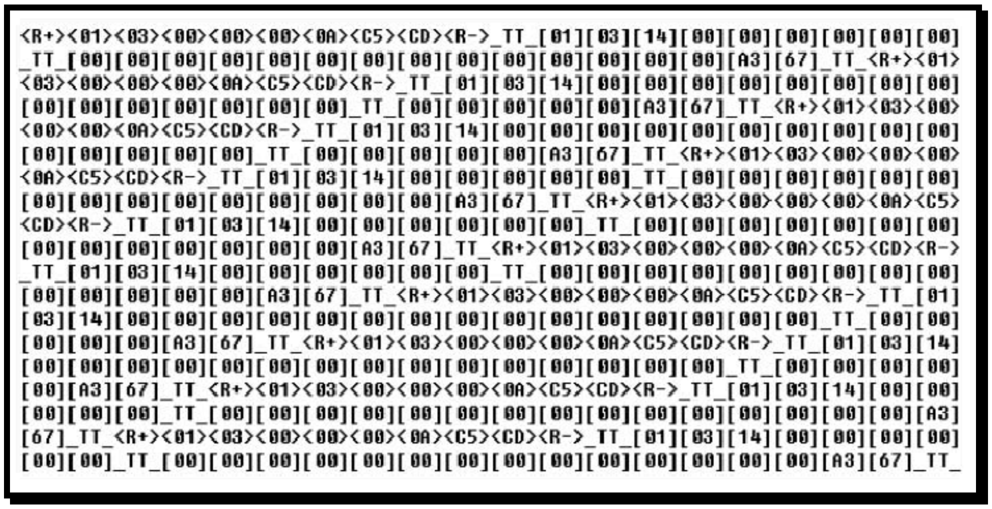
Press **[H]** to display the data on the current page in hexadecimal format.

Viewing Data in ASCII (Text) Format

Press **[A]** to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Starting the Data Analyzer

Press **[B]** to start the data analyzer. After the key is pressed, all data transmitted and received on the currently selected port will be displayed. An example display is shown below:



```
<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00][00]
_TT_[00][00][00][00][00][00][00][00][00][00][00][00][00][00][A3][67]_TT_<R+><01>
<03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00][00][00][00]
[00][00][00][00][00][00][00]_TT_[00][00][00][00][00][A3][67]_TT_<R+><01><03><00>
<00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00][00][00][00][00][00]
[00][00][00][00][00]_TT_[00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00>
<0A><C5><CD><R->_TT_[01][03][14][00][00][00][00][00]_TT_[00][00][00][00][00][00]
[00][00][00][00][00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5>
<CD><R->_TT_[01][03][14][00][00][00][00][00][00]_TT_[00][00][00][00][00][00][00]
[00][00][00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->
_TT_[01][03][14][00][00][00][00][00][00]_TT_[00][00][00][00][00][00][00][00][00]
[00][00][00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01]
[03][14][00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]_TT_[00][00]
[00][00][00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14]
[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00]_TT_[00][00][00][00]
[00][A3][67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00]
[00][00][00]_TT_[00][00][00][00][00][00][00][00][00][00][00][00][00][00][00][A3]
[67]_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT_[01][03][14][00][00][00][00]
[00][00]_TT_[00][00][00][00][00][00][00][00][00][00][00][00][00][00][A3][67]_TT_
```

The Data Analyzer displays the following special characters:

Character	Definition
[ ]	Data enclosed in these characters represent data received on the port.
< >	Data enclosed in these characters represent data transmitted on the port.
<R+>	These characters are inserted when the RTS line is driven high on the port.
<R->	These characters are inserted when the RTS line is dropped low on the port.

Character	Definition
<CS>	These characters are displayed when the CTS line is recognized high.
_TT_	These characters are displayed when the timing mark interval has been reached. This parameter is user defined.

Stopping the Data Analyzer

Press **[S]** to stop the data analyzer. Use this option to freeze the display so the data can be analyzed. To restart the analyzer, press **[B]**.

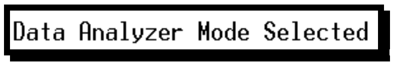
Important: When in analyzer mode, program execution will slow down. Only use this tool during a troubleshooting session. Before disconnecting from the Config/Debug port, please press [S] to stop the data analyzer, and then press [M] to return to the main menu. This action will allow the module to resume its normal high speed operating mode.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

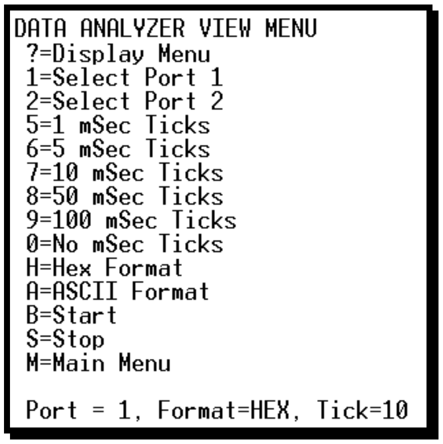
6.4.5 Data Analyzer Tips

From the main menu, press **[A]** for the "Data Analyzer". You should see the following text appear on the screen:



```
Data Analyzer Mode Selected
```

After the "Data Analyzer" mode has been selected, press **[?]** to view the Data Analyzer menu. You will see the following menu:



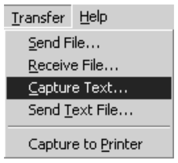
```
DATA ANALYZER VIEW MENU
 ?=Display Menu
 1=Select Port 1
 2=Select Port 2
 5=1 mSec Ticks
 6=5 mSec Ticks
 7=10 mSec Ticks
 8=50 mSec Ticks
 9=100 mSec Ticks
 0=No mSec Ticks
 H=Hex Format
 A=ASCII Format
 B=Start
 S=Stop
 M=Main Menu

 Port = 1, Format=HEX, Tick=10
```

From this menu, you can select the "Port", the "format", and the "ticks" that you can display the data in.

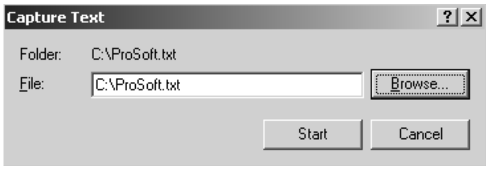
For most applications, HEX is the best format to view the data, and this does include ASCII based messages (because some characters will not display on HyperTerminal and by capturing the data in HEX, we can figure out what the corresponding ASCII characters are supposed to be).

The Tick value is a timing mark. The module will print a _TT for every xx milliseconds of no data on the line. Usually 10milliseconds is the best value to start with.

After you have selected the Port, Format, and Tick, we are now ready to start a capture of this data. The easiest way to do so is to go up to the top of you HyperTerminal window, and do a **Transfer / Capture Text** as shown below:



After selecting the above option, the following window will appear:

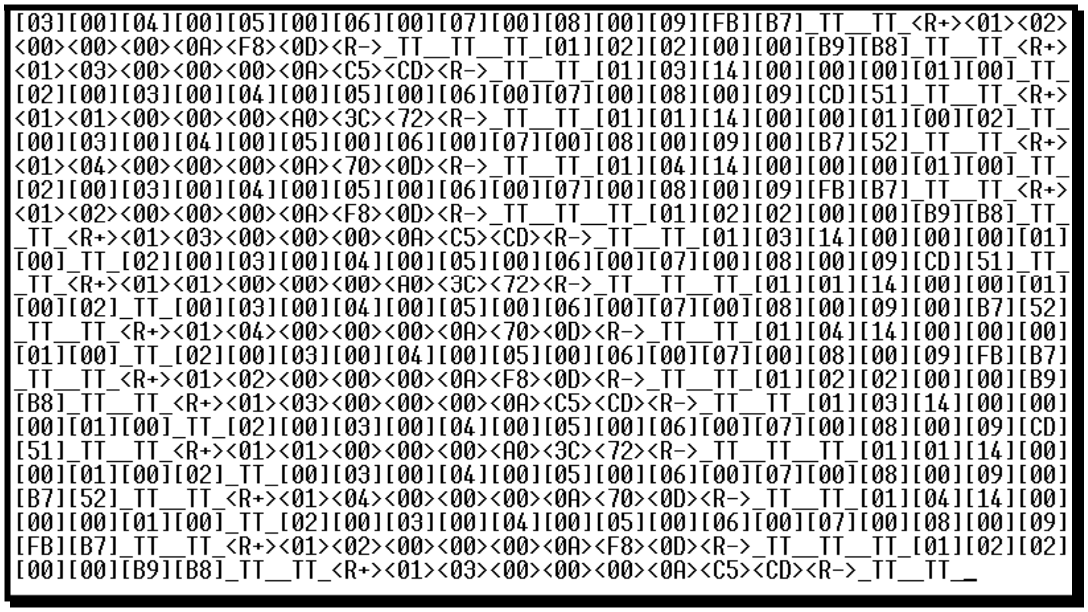


Folder: C:\ProSoft.txt
File: C:\ProSoft.txt

Next name the file, and select a directory to store the file in. In this example, we are creating a file ProSoft.txt and storing this file on our root C: drive. After you have done this, press the Start button.

Now you have everything that shows up on the HyperTerminal screen being logged to a file called ProSoft.txt. This is the file that you will then be able to email to ProSoft Technical Support to assist with issues on the communications network.

To begin the display of the communications data, you will then want to press 'B' to tell the module to start printing the communications traffic out on the debug port of the module. After you have pressed 'B', you should see something like the following:



```
[03][00][04][00][05][00][06][00][07][00][08][00][09][FB][B7]_TT__TT_<R+><01><02>
<00><00><00><0A><F8><0D><R->_TT__TT__TT_[01][02][02][00][00][B9][B8]_TT__TT_<R+>
<01><03><00><00><00><0A><C5><CD><R->_TT__TT_[01][03][14][00][00][00][01][00]_TT_
[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][CD][51]_TT__TT_<R+>
<01><01><00><00><00><A0><3C><72><R->_TT__TT_[01][01][14][00][00][01][00][02]_TT_
[00][03][00][04][00][05][00][06][00][07][00][08][00][09][00][B7][52]_TT__TT_<R+>
<01><04><00><00><00><0A><70><0D><R->_TT__TT_[01][04][14][00][00][00][01][00]_TT_
[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][FB][B7]_TT__TT_<R+>
<01><02><00><00><00><0A><F8><0D><R->_TT__TT__TT_[01][02][02][00][00][B9][B8]_TT_
_TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT__TT_[01][03][14][00][00][00][01]
[00]_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][CD][51]_TT_
_TT_<R+><01><01><00><00><00><A0><3C><72><R->_TT__TT__TT_[01][01][14][00][00][01]
[00][02]_TT_[00][03][00][04][00][05][00][06][00][07][00][08][00][09][00][B7][52]
_TT__TT_<R+><01><04><00><00><00><0A><70><0D><R->_TT__TT_[01][04][14][00][00][00]
[01][00]_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][FB][B7]
_TT__TT_<R+><01><02><00><00><00><0A><F8><0D><R->_TT__TT_[01][02][02][00][00][B9]
[B8]_TT__TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT__TT_[01][03][14][00][00]
[00][01][00]_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09][CD]
[51]_TT__TT_<R+><01><01><00><00><00><A0><3C><72><R->_TT__TT__TT_[01][01][14][00]
[00][01][00][02]_TT_[00][03][00][04][00][05][00][06][00][07][00][08][00][09][00]
[B7][52]_TT__TT_<R+><01><04><00><00><00><0A><70><0D><R->_TT__TT_[01][04][14][00]
[00][00][01][00]_TT_[02][00][03][00][04][00][05][00][06][00][07][00][08][00][09]
[FB][B7]_TT__TT_<R+><01><02><00><00><00><0A><F8><0D><R->_TT__TT__TT_[01][02][02]
[00][00][B9][B8]_TT__TT_<R+><01><03><00><00><00><0A><C5><CD><R->_TT__TT__
```

The <R+> means that the module is transitioning the communications line to a transmit state.

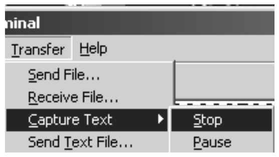
All characters shown in <> brackets are characters being sent out by the module.

The <R-> shows when the module is done transmitting data, and is now ready to receive information back.

And finally, all characters shown in the [ ] brackets is information being received from another device by the module.

After taking a minute or two of traffic capture, you will now want to stop the "Data Analyzer". To do so, press the 'S' key, and you will then see the scrolling of the data stop.

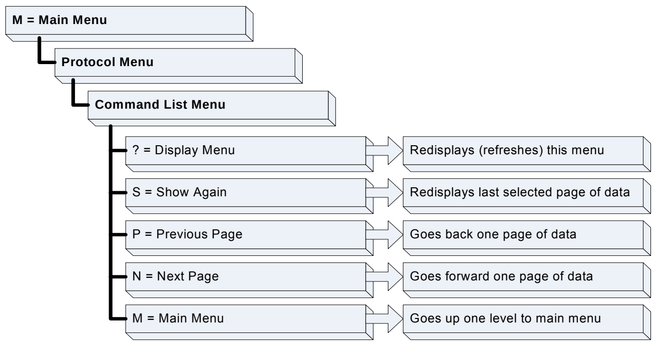
When you have captured the data you want to save, open the Transfer menu and choose Capture Text. On the secondary menu, choose Stop.



You have now captured, and saved the file to your PC. This file can now be used in analyzing the communications traffic on the line, and assist in determining communication errors.

6.4.6 Master Command List Menu

Use this menu to view the command list for the module. Press **[?]** to view a list of commands available on this menu.


M = Main Menu
Protocol Menu
Command List Menu
? = Display Menu
Redisplays (refreshes) this menu
S = Show Again
Redisplays last selected page of data
P = Previous Page
Goes back one page of data
N = Next Page
Goes forward one page of data
M = Main Menu
Goes up one level to main menu

Redisplaying the Current Page

Press **[S]** to display the current page of data.

Viewing the Previous 50 Commands

Press **[-]** to view the previous 50 commands.

Viewing the Previous Page of Commands

Press **[P]** to display the previous page of commands.

Viewing the Next 50 Commands

Press **[+]** to view the next 50 commands from the master command list.

Viewing the Next Page of Commands

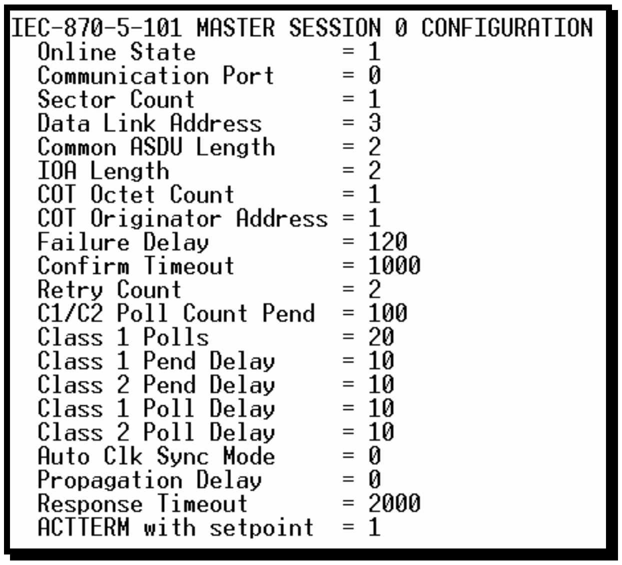
Press **[N]** to display the next page of commands.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

6.4.7 Session Configuration Menu

Press **[S]** from the IEC-101 Master Driver Menu to open the Session Configuration menu. Use this command to view the session configuration for each controlled device.



```
IEC-870-5-101 MASTER SESSION 0 CONFIGURATION
  Online State           = 1
  Communication Port     = 0
  Sector Count           = 1
  Data Link Address      = 3
  Common ASDU Length     = 2
  IOA Length             = 2
  COT Octet Count        = 1
  COT Originator Address = 1
  Failure Delay          = 120
  Confirm Timeout        = 1000
  Retry Count            = 2
  C1/C2 Poll Count Pend  = 100
  Class 1 Polls          = 20
  Class 1 Pend Delay     = 10
  Class 2 Pend Delay     = 10
  Class 1 Poll Delay     = 10
  Class 2 Poll Delay     = 10
  Auto Clk Sync Mode     = 0
  Propagation Delay      = 0
  Response Timeout       = 2000
  ACTTERM with setpoint  = 1
```

Online State

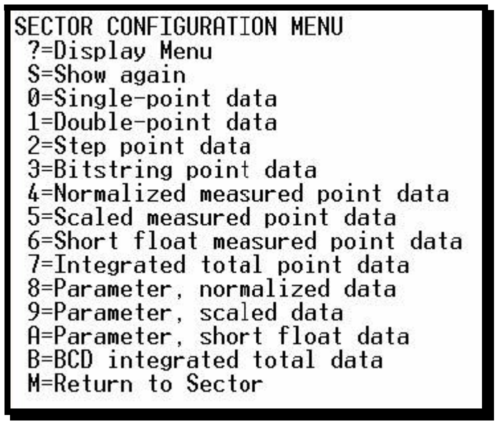
The Online State indicator displays 0 if the module is not online, 1 if the module is online.

Session State

The Session State indicator displays 1 if there is a configuration error, or 2 if the module is ready for communication. If the session is not in use, the Session State indicator displays 0.

6.4.8 Sector Configuration Menu

Press **[1]** from the IEC-101 Master Driver Menu to open the Sector Configuration menu. Use this command to view the contents of the Sector Configuration Databases for each session (controlled device). The module supports up to three sectors (databases) per session.



```
SECTOR CONFIGURATION MENU
 ?=Display Menu
 S=Show again
 0=Single-point data
 1=Double-point data
 2=Step point data
 3=Bitstring point data
 4=Normalized measured point data
 5=Scaled measured point data
 6=Short float measured point data
 7=Integrated total point data
 8=Parameter, normalized data
 9=Parameter, scaled data
 A=Parameter, short float data
 B=BCD integrated total data
 M=Return to Sector
```

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Opening the Sector Database Menu

Press **[D]** from the Sector Configuration menu to open the Sector Database menu. Use this command to look at the configuration and current value for each point.

The *IEC-870-Master Command List Menu* section has more information about the commands on this menu.

Redisplaying the Current Page

Press **[S]** to display the current page of data.

Displaying the Next Page

Press **[N]** to display the next 100 registers. Use this command to step forward through the data a page at a time.

Displaying the Previous Page

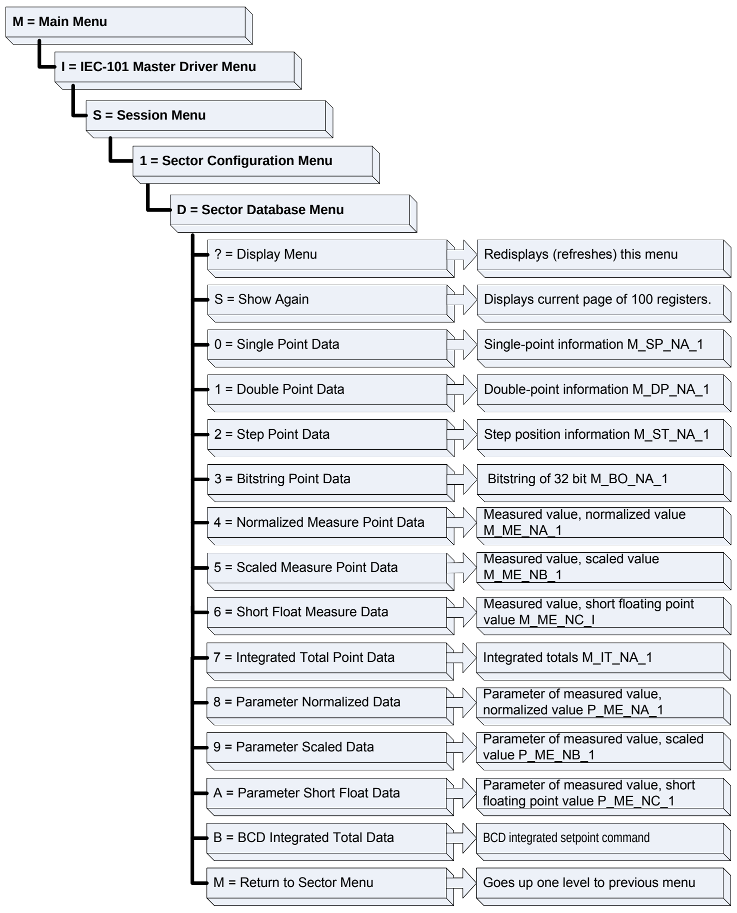
Press **[P]** to display the previous 100 registers. Use this command to step backward through the data a page at a time.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

6.4.9 Sector Database Menu

Press **[D]** from the Sector Configuration menu to open the Sector Database menu. Use this command to display the sector database values. Each session (controlled device) contains one or more data sets (sectors) that are defined by the vendor of the device.


M = Main Menu
I = IEC-101 Master Driver Menu
S = Session Menu
1 = Sector Configuration Menu
D = Sector Database Menu
? = Display Menu
Redisplays (refreshes) this menu
S = Show Again
Displays current page of 100 registers.
0 = Single Point Data
Single-point information M_SP_NA_1
1 = Double Point Data
Double-point information M_DP_NA_1
2 = Step Point Data
Step position information M_ST_NA_1
3 = Bitstring Point Data
Bitstring of 32 bit M_BO_NA_1
4 = Normalized Measure Point Data
Measured value, normalized value M_ME_NA_1
5 = Scaled Measure Point Data
Measured value, scaled value M_ME_NB_1
6 = Short Float Measure Data
Measured value, short floating point value M_ME_NC_I
7 = Integrated Total Point Data
Integrated totals M_IT_NA_1
8 = Parameter Normalized Data
Parameter of measured value, normalized value P_ME_NA_1
9 = Parameter Scaled Data
Parameter of measured value, scaled value P_ME_NB_1
A = Parameter Short Float Data
Parameter of measured value, short floating point value P_ME_NC_1
B = BCD Integrated Total Data
BCD integrated setpoint command
M = Return to Sector Menu
Goes up one level to previous menu

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Redisplaying the Current Page

Press **[S]** to display the current page of data.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

6.5 LED Status Indicators

The LEDs indicate the module's operating status as follows:

ProSoft Module	Color	Status	Indication
DEBUG	Green	On	Data is being transferred between the module and a remote terminal using the Configuration/Debug port.
		Off	No data is being transferred on the Configuration/Debug port.
PRT1	Green	On	Data is being transferred between Port 1 and the slave
		Off	No data
PRT2	Green	On	Data is being transferred between Port 2 and the slave
		Off	No data
CFG/ERR	N/A	Off	Not Used
ERR1	Red	Off	The PTQ-101M is working normally.
		On	The PTQ-101M module program has recognized an application error.
ERR2	N/A	Off	Not used in application
ERR3	N/A	Off	Not used in application
Active	Green	On	The LED is on when the module recognizes a processor and is able to communicate if the [Backplane Data Movement] section specifies data transfer commands.
		Off	The LED is off when the module is unable to speak with the processor. The processor either absent or not running.
BAT	Red	Off	The battery voltage is OK and functioning.
		On	The battery voltage is low or the battery is not present. The battery LED will illuminate briefly upon the first installation of the module or if the unit has been un-powered for an extended period of time. This behavior is normal, however should the LED come on in a working installation please contact ProSoft Technology.

If your module is not operating, and the status LEDs are not illustrated in the table above, please call ProSoft Technology for technical assistance.

7 Reference

In This Chapter

7.1 Product Specifications

The IEC 60870-5-101 Master Communication Module allows Quantum backplane I/O compatible processors to interface easily with IEC 60870-5-101 slave (controlled unit) devices.

7.1.1 Features and Benefits

The PTQ-101M module interfaces up to 32 serial communication devices with a Quantum or Unity processor. Two communication ports on the module act as controlling devices (masters) to interface with controlled devices on their own networks. Each port is individually configurable and can be set for balanced or unbalanced mode. Data is exchanged between the serial network and the processor using the internal database contained in the module and direct control by the controller's ladder logic.

7.1.2 General Specifications

- Single Slot - Quantum backplane compatible
- The module is recognized as an Options module and has access to PLC memory for data transfer
- Configuration data is stored in non-volatile memory in the ProTalk module
- Up to six modules can be placed in a rack
- Local rack - The module must be placed in the same rack as processor.
- Compatible with common Quantum / Unity programming tools.
- Quantum data types supported: 0x, 1x, 3x, 4x
- High speed data transfer across backplane provides quick data update times.
- Sample function blocks available.

7.1.3 Hardware Specifications

Specification	Value
Backplane Current Load	800 mA @ 5 V
Operating Temperature	0 to 60°C (32 to 140°F)
Storage Temperature	-40 to 85°C (-40 to 185°F)
Relative Humidity	5% to 95% (non-condensing)
Vibration	Sine vibration 4-100 Hz in each of the 3 orthogonal axes
Shock	30G, 11 mSec. in each of the 3 orthogonal axes
LED Indicators	Module Status Backplane Transfer Status Serial Port Activity LED Serial Activity and Error LED Status
Configuration Serial Port (PRT1)	DB-9M PC Compatible RS-232 only No hardware handshaking
Application Serial Ports	(PRT2, PRT3) DB-9M PC Compatible RS-232/422/485 jumper selectable RS-422/485 screw termination included RS-232 handshaking configurable 500V Optical isolation from backplane

7.1.4 Functional Specifications

- Built in accordance to the approved international specification
- Two independent master ports completely user configurable
- Support for balanced and unbalanced mode
- Up to 32 sessions
- Up to five sectors (separate databases) for each session
- Individual database definition for each sector
- 1000 commands to control stations
- Processor can issue control commands directly to the module or a controlled device (10 at each scan)
- Pass-through of event messages from controlled device to processor for logging of time-tagged events
- Supports clock synchronization from/to the processor
- Receives events from the slave and sends them to the processor
- Supports monitored data
  - Single-point
  - Double-point
  - Step-point
  - Measured-point
  - Bitstring 32-bit
  - Integrated total point
- Class 1 and Class 2 delay parameter in the configuration file
- Complete set up and monitoring of module through Unity Pro XL or Concept software and user constructed configuration file (IEC101M.CFG)
- All data related to the module is contained in user data files to simplify monitoring and interfacing with the module

7.2 Functional Overview

This section provides an overview of how the PTQ-101M module transfers data using the 101M protocol. You should understand the important concepts in this chapter before you begin installing and configuring the module.

The standards used to build the module are listed in the following table:

Publication	Title
IEC 60870-5-101	Companion Standard for Basic Telecontrol Tasks
IEC 60870-5-101 Amendment 1	Companion Standard for Basic Telecontrol Tasks
IEC 60870-5-1	Transmission Frame Formats
IEC 60870-5-2	Link Transmission Procedures
IEC 60870-5-3	General Structure of Application Data
IEC 60870-5-4	Definition and Coding of Application Information Elements
IEC 60870-5-5	Basic Application Functions
IEC 60870-5-104	Network access for IEC 60870-5-101 using standard transport profiles

These documents should be obtained, reviewed, and understood in order to fully appreciate the protocol implementation. Most of the complexity of the protocol is hidden from the user and simplified in the application of the module. Detailed questions of about the protocol can be answered by reading these documents. In addition to calling our technical support group, there is also help available for the protocol using the following mail list Web Site: www.TriangleMicroWorks.com/iec870-5 (http://www.trianglemicroworks.com/iec870-5). Go to this site to join the mail list and to review questions and answers from mail list users.

7.2.1 General Concepts

The following discussion explains several concepts that are important for understanding the operation of the PTQ-101M module.

Module Power Up

On power up the module begins performing the following logical functions:

1 Initialize hardware components
   - o Initialize Quantum backplane driver
   - o Test and clear all RAM
   - o Initialize the serial communication ports

2 Read configuration for module from IEC101M.CFG file on Compact Flash Disk

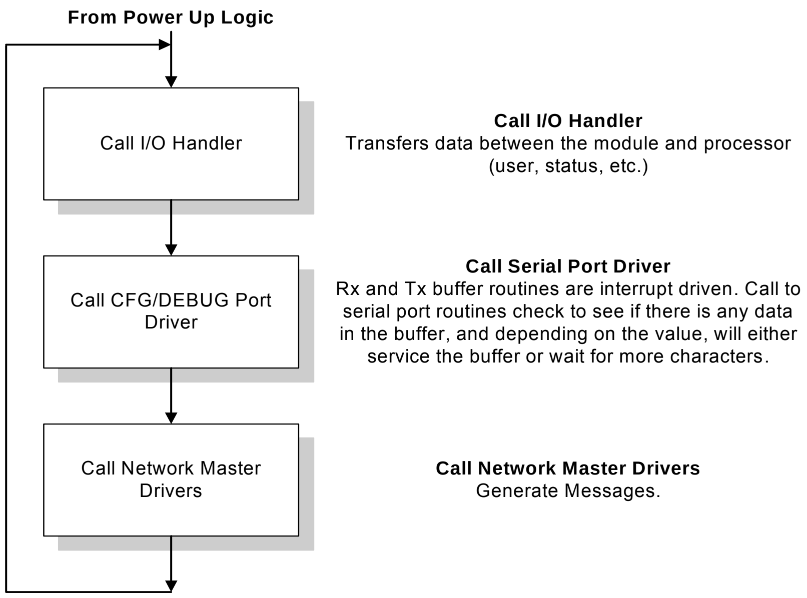
3 Initialize the databases and ports

4 Set up the serial communication interface for the debug/configuration port

After the module has received the configuration, the module will begin receiving and transmitting messages with devices on the serial networks.

Main Logic Loop

Upon completing the power up configuration process, the module enters an infinite loop that performs the following functions:


From Power Up Logic
Call I/O Handler
Call CFG/DEBUG Port Driver
Call Network Master Drivers
Call I/O Handler
Transfers data between the module and processor (user, status, etc.)
Call Serial Port Driver
Rx and Tx buffer routines are interrupt driven. Call to serial port routines check to see if there is any data in the buffer, and depending on the value, will either service the buffer or wait for more characters.
Call Network Master Drivers
Generate Messages.

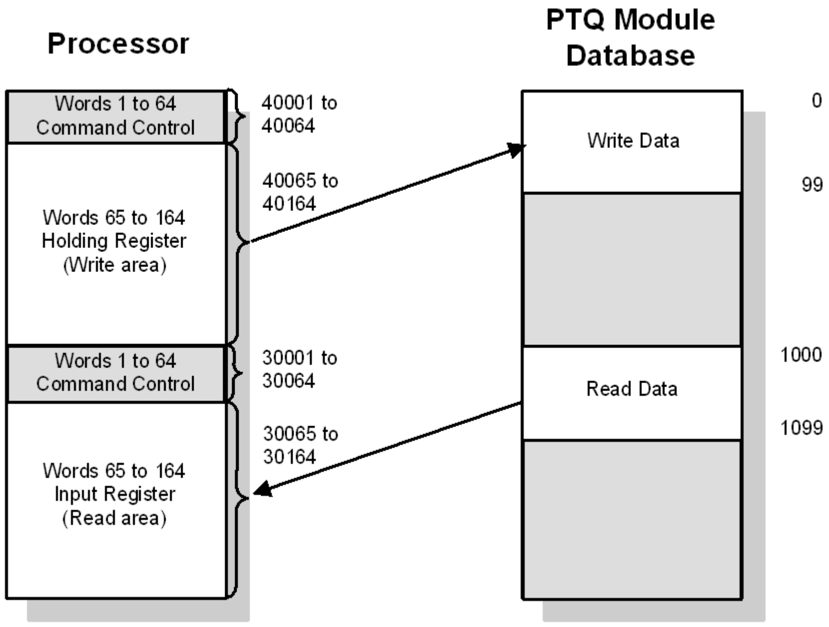
7.2.2 Backplane Data Transfer

The current version of the PTQ-101M backplane driver (version 2.10 or newer), uses a Large I/O model, which differs from previous versions of the backplane driver in that it transfers all of the data in the Read and Write databases between the module and the processor on every scan.

The [Backplane Configuration] section of the configuration file defines the starting registers for read and write operations, as well as the number of registers to use for each data area.

```
#These values are required to define the data area to transfer between the
#module and the processor.
Read Register Start :   0    #Database start register to move to processor
Read Register Count :   50   #Number of words moved from module to
                             #processor
Write Register Start:   1000 #Database start register where data placed
                             #from processor
Write Register Count:   50   #Number of words moved from processor to
                             #module
Pass-Through Events  :  N    #Pass event messages to processor (N =No events
                             #will be passed to the processor, Y=Yes will be
                             #passed to the processor consuming block 9903
#Used to define the area in the Processor for the module to interface with
3x Register Start:        1  #3x start register where data moved from
                             #module to processor (1 to n)
4x Register Start:        1  #4x start register where data moved from
                             #processor to module (1 to n)
```

The values in the example configuration file section above are illustrated in the following diagram.


Processor
PTQ Module Database
Words 1 to 64 Command Control
40001 to 40064
Words 65 to 164 Holding Register (Write area)
40065 to 40164
Words 1 to 64 Command Control
30001 to 30064
Words 65 to 164 Input Register (Read area)
30065 to 30164
Write Data
0
99
Read Data
1000
1099

The module transfers the entire read and write areas at the end of every processor scan. The module will hold the processor scan for a certain period of time, which allows the module to transfer the entire read and write areas. This means that the larger the read and write areas, the longer the processor scan time will be.

Note: The diagram above shows the memory addresses for a Quantum processor. If you are deploying the PTQ-101M with a Unity processor, substitute %MW for read only data, and %IW for read/write data.

Data Exchange

The module transfers all the configured read or write data at the end of each processor scan. You can configure up to 4000 words in each direction. The more data you configure, the longer the processor scan will be.

Words 0 through 63 in each read/write block are reserved for command control. Refer to Command Control (page 131) for more information on command control blocks. The following table shows the relationship between the processor memory and the module database areas.

Note: Refer to Backplane Data Transfer (page 114) for the example configuration values that are used in the following tables.

Module Database	Register	Unity Register	Description
Read Data	3x	%IW	Input Register
Write Data	4x	%MW	Holding Register

The data mapping in the following example shows the relationship between processor and PTQ-101M memory addresses, assuming a 4x register start value of 40001 and a PTQ-101M database start value of 0.

Processor Memory Address	**Module Database Address**
40065	0
40066	1
40067	2
40068	3
40069	4
…	…
40164	99

The data mapping in the following example shows the relationship between processor and PTQ-101M memory addresses, assuming a 3x register start value of 30001 and a PTQ-101M database start value of 2000.

Processor Memory Address	**Module Database Address**
30065	2000
30066	2001
30067	2002
30068	2003
30069	2004
…	…
30164	2099

Command Control Block

The first 64 words of each block are reserved for command control. Each command control block has a Block ID number (shown in parentheses below) that identifies the command control instruction. The PTQ-101M module supports the following command control blocks:

- Status Block (9250)
- User Constructed Command block (9901)
- Command Control Block (9902)
- Events messages from Master port (9903)\
- Command List Error data (9950)
- Read Module's Time to Processor (9970)
- Set Module's Time Using Processor Time (9971)
- Warm Boot (9998) or Cold Boot (9999)

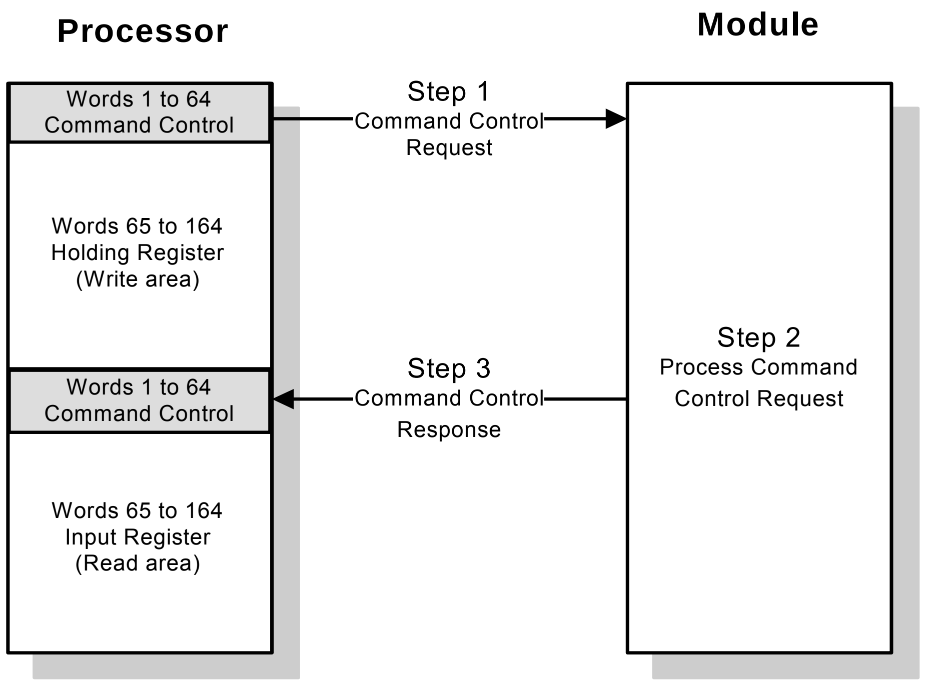
The value in word 0 of this 64 word block is the block sequence number. This number identifies whether the contents of the block have changed. This is the actual trigger to send the control request to the module.

Processor logic must be built to handle the command control functionality. The logic would typically follow these steps:

1 Move the block request to output command control area.
2 Move a new value to the output block sequence number.

3 If the input block sequence number equals the output block sequence number + 1, copy the block response to appropriate variables in the module's memory.

Note: Command Control blocks are not copied to the module database. You must define variables in the module's main memory, and use processor logic to process the command control request.


Processor
Module
Words 1 to 64 Command Control
Words 65 to 164 Holding Register (Write area)
Words 1 to 64 Command Control
Words 65 to 164 Input Register (Read area)
Step 1 Command Control Request
Step 2 Process Command Control Request
Step 3 Command Control Response

The following table shows the contents of the command control area when a command control block such as 9970 (Read Module's Time to Processor) is issued.

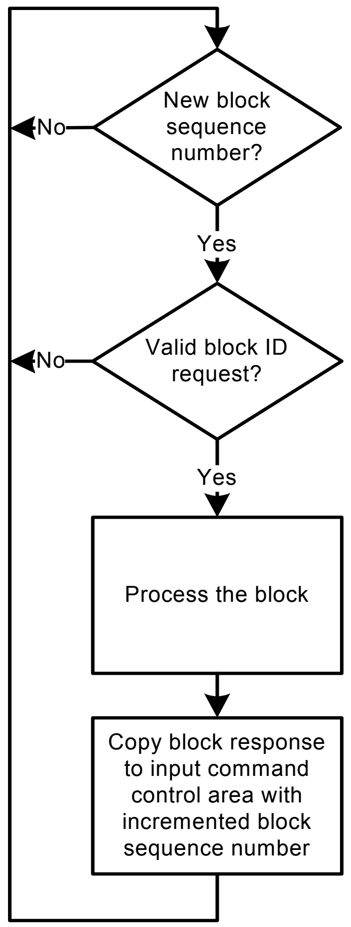
Note: The diagram above shows the memory addresses for a Quantum processor. If you are deploying the PTQ-101M with a Unity processor, substitute %MW for read only data, and %IW for read/write data.
Note: The processor memory locations in the example tables below use the 3x register start and 4x register start values defined in Backplane Data Transfer. You can configure any valid 3x and 4x start address that is not used by other processes.

Command Control Word	Description
40001	Output sequence number
40002	Block ID
40003	Block request word 1
40004	Block request word 2
40005	Block request word 3
…	…
40064	Block request word 62

The following table shows the results of the PTQ-101M response to the command control block.

Command Control Word	Description
30001	Input sequence number
30002	Block ID
30003	Block response word 1
30004	Block response word 2
30005	Block response word 3
…	…
30064	Block response word 62

The module recognizes that there is a new block request when it identifies that the block sequence number has changed. If the block ID is valid, the module will process the block and copy the response to the input command control area (3x for or %IW for Unity). The module will increment the block sequence number by one, as shown in the following illustration.


New block sequence number?
No
Yes
Valid block ID request?
No
Yes
Process the block
Copy block response to input command control area with incremented block sequence number

Status Block (9250)

If a value of 9250 is placed in the control register, Status data will be sent from the processor to the module.

The following table shows the block format for write.

Block Format for Write

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the block identification code of 9250 for the block.
2 to 63	Spare	Not used.

Block Format for Read

Offset	Parameter	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9250 for the block.
2	Scan Count	This status value contains a counter incremented on each scan of the module's main loop.
3 to 4	Product Name	This two-word data area contains the text values representing the product name. These words contain the text '87S6' for the MVI69 platform.
5 to 6	Revision	This two-word data area contains the text values for the revision number.
7 to 8	Op Sys #	This two-word data area contains the text values for the operating system number.
9 to 10	Run Number	This two-word data area contains the text values for the run number.
11	Read Blk Cnt	This word contains the total number of block read operations successfully executed.
12	Write Blk Cnt	This word contains the total number of block write operations successfully executed.
13	Parse Blk Cnt	This word contains the total number of write blocks successfully parsed.
14	Error Blk Cnt	This word contains the total number of block transfer errors.
15	Event Msg Cnt	This word contains the number of event messages waiting to send to the processor.
16	Event Msg Overflow	This word contains a value of 0 if the event message buffer has not overflowed. If the event buffer overflows, this word will be set to a value of 1.
17	Session Count	This word contains the number of session configured in the module.
18	Current Cmd	This word contains the index of the current command being executed in the command list.
19	Cmd Busy Flag	This word is set to zero if no command is currently being executed and waiting on a response. If the word is set to 1, a command is currently executing.

Offset	Parameter	Description
20	Cmd Count	This word contains the count of the number of commands configured for the module.
21	Cmd Delay	This word contains the command delay counter preset. There is a fixed delay between each command to permit the module to perform class polls on controlled stations.
22	Cmd Queue	This word is set to zero if the command executing is from the command list. If the executing command is from the command queue, the word will be set to 1.
23	Cmd Queue Count	This word contains the number of active commands in the command queue for the module. Up to 100 commands can be buffered in this queue. These commands are transferred from the processor to the module using special command blocks.
24 to 25	Online Status	This double word value contains a bit for each of the 32 potential sessions in the module. If the bit is set for a session in the double word, the station is online. If the bit is clear, the station is offline. Use this value to determine if commands sent from the processor will have a chance of succeeding.
26	CH 0 State	This word contains the state machine value for channel 0.
27	Cmd Req	This word contains the number of commands transferred out channel 0.
28	Cmd Resp	This word contains the number of command response messages received on channel 0.
29	Cmd Err	This word contains the number of command errors recognized on channel 0.
30	Requests	This word contains the total number of messages transmitted on channel 0.
31	Responses	This word contains the total number of messages received on channel 0.
32	Err Sent	This word contains the number of error messages sent on channel 0.
33	Err Received	This word contains the number of error messages received on channel 0.
34	Cfg Err	This bit mapped word is used to recognize any configuration errors for channel 0. Refer to the configuration error word table (page 152) for a definition of each bit.
35	Current Error	This word contains the error code for the current command executing on channel 0.
36	Last Error	This word contains the error code for the last error recognized on channel 0.
37	CH 1 State	This word contains the state machine value for channel 1.
38	Cmd Req	This word contains the number of commands transferred out channel 1.
39	Cmd Resp	This word contains the number of command response messages received on channel 1.
40	Cmd Err	This word contains the number of command errors recognized on channel 1.

Offset	Parameter	Description
41	Requests	This word contains the total number of messages transmitted on channel 1.
42	Responses	This word contains the total number of messages received on channel 1.
43	Err Sent	This word contains the number of error messages sent on channel 1.
44	Err Received	This word contains the number of error messages received on channel 1.
45	Cfg Err	This bit mapped word is used to recognize any configuration errors for channel 1. Refer to the configuration error word table (page 152) for a definition of each bit.
46	Current Error	This word contains the error code for the current command executing on channel 1.
47	Last Error	This word contains the error code for the last error recognized on channel 1.
48 to 63	Spare	Not used

Read Module's Time to Processor (9970)

If a value of 9970 is placed in the control register, the processor will read the module's current time.

Block Format for Write

The following table shows the block format for write:

Word Offset in Block	Data Field(s)	Description
0	Sequence Number	This number triggers the request for the module. When this number changes, the module will process the command control request.
1	Block ID	This field contains the value of 9970 identifying the block type to the module.

Block Format for Read

The module responds to a valid 9970 request with a block containing the requested date and time. The block format is shown in the following table:

Word Offset in Block	Data Field(s)	Description
0	Sequence Number	This is the sequence number received by the module, incremented by one, after the request is processed.
1	Block ID	This word will contain the value of 9970.
2	Year	This field contains the four-digit year to be used with the new time value.
3	Month	This field contains the month value for the new time. Valid Values: 1 to 12.
4	Day	This field contains the day value for the new time. Valid Values: 1 to 31.
5	Hour	This field contains the hour value for the new time. Valid Values: 0 to 23
6	Minute	This field contains the minute value for the new time. Valid Values: 0 to 59.
7	Seconds	This field contains the second value for the new time. Valid Values: 0 to 59.
8	Milliseconds	This field contains the millisecond value for the new time. Valid Values: 0 to 999.

Set Module's Time Using Processor Time (9971)

If a value of 9971 is placed in the control register, Module time is set using the processor's time. The following table shows the block format for write.

Block Format for Write

Word Offset in Block	Data Field(s)	Description
0	Write Block ID	This word will contain the value of 9971.
1	Year	This field contains the four-digit year to be used with the new time value.
2	Month	This field contains the month value for the new time. Valid Values: 1 to 12.
3	Day	This field contains the day value for the new time. Valid Values: 1 to 31.
4	Hour	This field contains the hour value for the new time. Valid Values: 0 to 23
5	Minute	This field contains the minute value for the new time. Valid Values: 0 to 59.
6	Seconds	This field contains the second value for the new time. Valid Values: 0 to 59.
7	Milliseconds	This field contains the millisecond value for the new time. Valid Values: 0 to 999.

Block Format for Read

Word Offset in Block	Data Field(s)	Description
0	Sequence Number	This is the sequence number received by the module, incremented by one, after the request is processed.
1	Block ID	This word will contain the value of 9971

Warm Boot (9998) or Cold Boot (9999)

If the processor places a value of 9998 in this register, the module will perform a warm-boot operation. If the processor places a value of 9999 in this register, the module will perform a cold-boot operation. In this application module, both of these operations perform the same function. They exit the program and then restart the program. Many of the program parameters set in the user configuration must be set at program initialization and cannot be set while the program is running. Therefore, both functions operate the same way.

Block Format for Write

Word Offset in Block	Data Field(s)	Description
0	Sequence Number	This number triggers the request for the module. When this number changes, the module will process the command control request.
1	Block ID	This word will contain the value of 9998 (Warm Boot) or 9999 (Cold Boot)

The logic must set the values of the sequence number and block ID for one processor scan only.

Refer to Implementing Ladder to Support Special Functions (page 123) for sample code that handles these command control blocks.

Implementing Ladder to Support Special Functions

In order to use Special Functions (Command Control), you must implement some form of control logic. The following section uses structured text language to illustrate how a typical function might be implemented.

Example: Rebooting the Module.

MyTrigger is a variable that triggers this logic

OutputControl variable array starts at register 4000001

The first instruction guarantees that the processor requests this block for only one scan.

The second instructions sets the Block Number (9999 = ColdBoot) and then sets the sequence number to 1.

```
IF MyTrigger>0 AND OutputControl1[1]> 0 THEN
 OutputControl1[0]:= InputData[0];
 OutputControl1[1]:=0;
  MyTrigger :=0;
END_IF;

IF (MyTrigger=9999)OR (MyTrigger=9998) OR (MyTrigger=9250) THEN
 OutputControl1[1] :=MyTrigger;
 Temp:=WORD_TO_INT(OutputControl1[0]);
 Temp:=Temp+1;
 OutputControl1[0]:=INT_TO_WORD(Temp);

 END_IF;
```

Example: Retrieving the time of day from the module.

This logic shows an example on how to request a block 9970 from the module (Read Module's Time) and read the response to the processor.

Assumptions:

- MyTrigger is a variable that triggers this logic
- OutputControl variable array starts at register 4000001
- InputControl variable array starts at register 3000001
- MyTime variables store the date and time values to be read from the module

Sets the Block Number (9970=Read Module's Time) and then increments the output sequence number (OutputControl[1]) by one. Once the module reads a new output sequence number from the processor it will process this request. So remember that the actual trigger is moving a new output block sequence number value to the module. Moving the block number (9970) is not the trigger to request this task from the module.

MyTrigger is set to -1 as an indication that the logic is waiting for the response from the module.

```
IF (MyTrigger=9970) THEN
 OutputControl1[1] :=MyTrigger;
 Temp:=WORD_TO_INT(OutputControl1[0]);
 Temp:=Temp+1;
 OutputControl1[0]:=INT_TO_WORD(Temp);

END_IF;
```

When the request is processed, the module will send the block response and increment the received output sequence number by 1. So the output sequence number is one less than the input sequence number the module has sent a new block. Once the block is received the processor logic copies the received data to the appropriate variables. The logic also clears the trigger for the next request.*)

```
IF (InputData[1]=9970) THEN
GetTime.MyYear :=InputData[2];
GetTime.MyMonth :=InputData[3];
GetTime.MyDay :=InputData[4];
GetTime.MyHour :=InputData[5];
GetTime.MyMinute :=InputData[6];
GetTime.MySecond :=InputData[7];
GetTime.MyMillisecond :=InputData[8];
END_IF
```

Example: Setting the time of day to the module.

This logic shows an example on how to request a block 9971 from the module (Read Module's Time).

Assumptions:

- MyTrigger is a variable that triggers this logic
- OutputControl variable array starts at register 4000001
- InputControl variable array starts at register 3000001
- MyTime variables store the date and time values to be written to the module

Sets the Block Number (9971=Write Module's Time) and then increments the output sequence number (OutputControl[1]) by one. Once the module reads a new output sequence number from the processor it will process this request. So remember that the actual trigger is moving a new output block sequence number value to the module. Moving the block number (9970) is not the trigger to request this task from the module. MyTrigger is set to -1 as an indication that the logic is waiting for the response from the module.

```
IF (MyTrigger=9971) THEN
OutputControl1[1] :=MyTrigger;
OutputControl1[2] :=SetTime.MyYear;
OutputControl1[3] :=SetTime.MyMonth;
OutputControl1[4] :=SetTime.MyDay;
OutputControl1[5] :=SetTime.MyHour;
OutputControl1[6] :=SetTime.MyMinute;
OutputControl1[7] :=SetTime.MySecond;
OutputControl1[8] :=SetTime.MyMillisecond;
Temp:=WORD_TO_INT(OutputControl1[0]);
Temp:=Temp+1;
OutputControl1[0]:=INT_TO_WORD(Temp);

END_IF;
```

Once the request was processed the module will send the block response and increment the received output sequence number by 1. So the output sequence number is one less than the input sequence number the module has sent a new block. *)

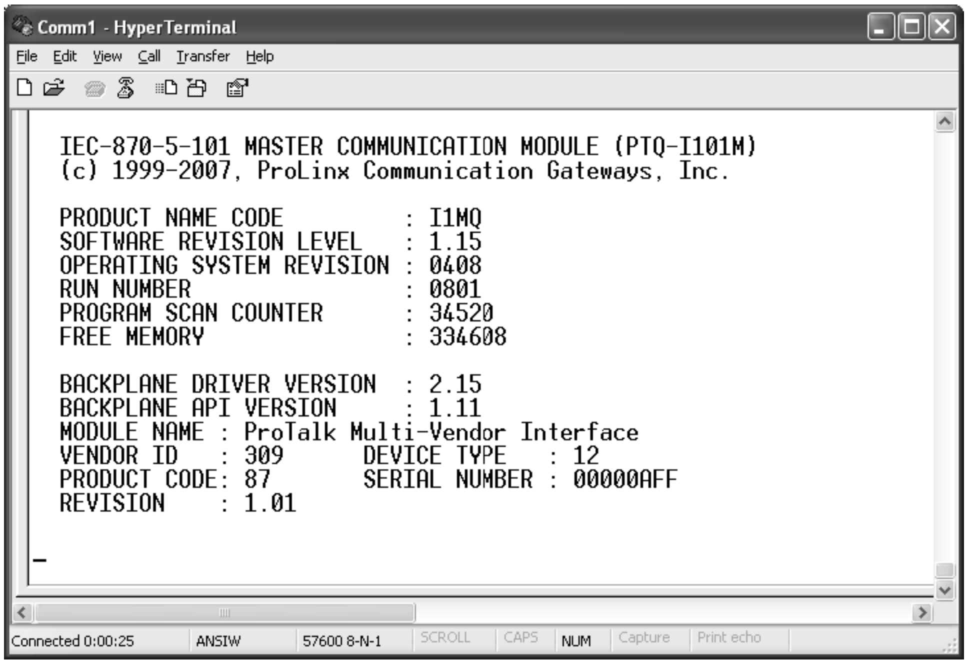
Event Pass-Through Block (9903)

Event Pass-Through Functionality

The event pass-through functionality allows the module to pass events to the processor after these are received from the IEC-8707-5-101 slave devices. We are considering events as the messages associated to supported ASDU types that contain timestamp (Hour:Minute:Seconds:Milliseconds).

Note: The event pass-through functionality is only available for version 1.12 or later.

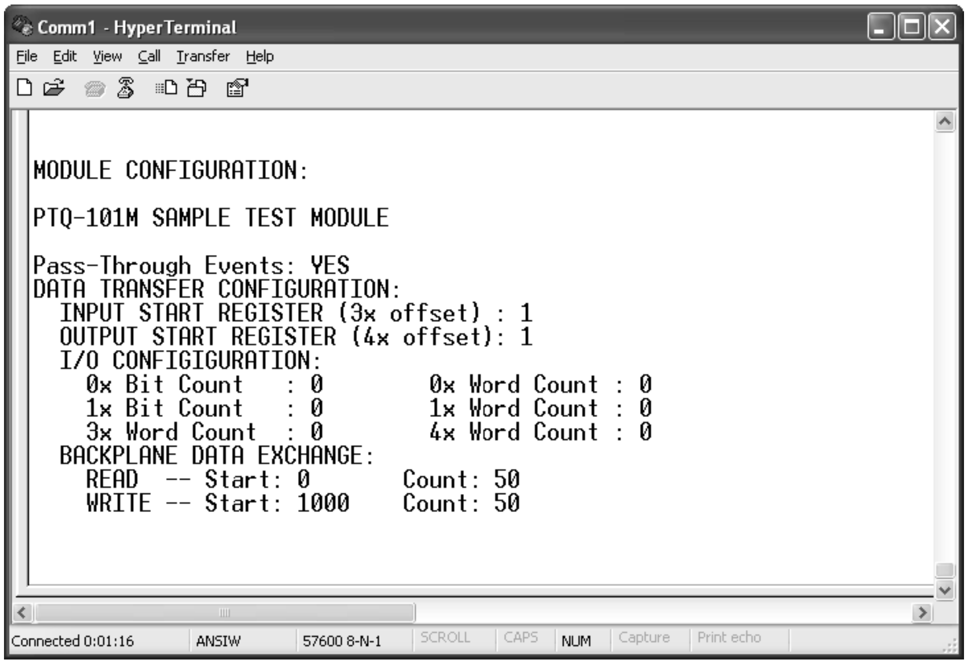
To verify the firmware revision of your module, press the [V] key from the main menu and look for the SOFTWARE REVISION LEVEL value. If your module does not have version 1.12 installed, please contact the ProSoft Technology tech support team for information on how to upgrade your module.



```
IEC-870-5-101 MASTER COMMUNICATION MODULE (PTQ-I101M)
(c) 1999-2007, ProLinx Communication Gateways, Inc.

PRODUCT NAME CODE         : I1MQ
SOFTWARE REVISION LEVEL   : 1.15
OPERATING SYSTEM REVISION : 0408
RUN NUMBER                : 0801
PROGRAM SCAN COUNTER      : 34520
FREE MEMORY               : 334608

BACKPLANE DRIVER VERSION  : 2.15
BACKPLANE API VERSION     : 1.11
MODULE NAME : ProTalk Multi-Vendor Interface
VENDOR ID   : 309       DEVICE TYPE   : 12
PRODUCT CODE: 87        SERIAL NUMBER : 00000AFF
REVISION    : 1.01

_
```

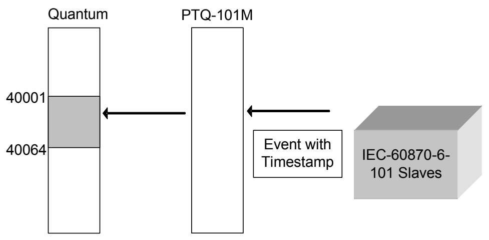
The event pass-through functionality must be initially enabled by the user through the following configuration parameter:

```
Pass-Through Events:    Y    #Pass event messages to processor
```



```
MODULE CONFIGURATION:

PTQ-101M SAMPLE TEST MODULE

Pass-Through Events: YES
DATA TRANSFER CONFIGURATION:
  INPUT START REGISTER (3x offset) : 1
  OUTPUT START REGISTER (4x offset): 1
  I/O CONFIGIGURATION:
    0x Bit Count   : 0       0x Word Count : 0
    1x Bit Count   : 0       1x Word Count : 0
    3x Word Count  : 0       4x Word Count : 0
  BACKPLANE DATA EXCHANGE:
    READ  -- Start: 0       Count: 50
    WRITE -- Start: 1000    Count: 50
```

The following illustration shows the basic idea of the event pass-through functionality. When the module receives the event from the remote device, it will build block 9903, which will be copied to the processor at the configured memory address:



Event Pass-Through Block Format

The block that is copied from the module to the processor has the following format. Each block can contain up to 4 events. The number of events per block will typically depend on the rate between how fast the module receives the events and how fast these can be passed to the processor (typically depends on the processor scan rate).

Block Format for Read

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9903 for the block.
2	Event Count	This field contains the number of events present in the block. Values of 1 to 4 are valid.
3 to 16	Event 1	Event message
17 to 30	Event 2	Event message
31 to 44	Event 3	Event message
45 to 58	Event 4	Event message
59 to 61	Spare	Not used
62	Event count in queue	Number of events in queue still waiting to send
63	Event Overflow	Event buffer overflow

The format of each 14 word data region in the block is shown in the following table.

Word Offset	Definitions	Description
0	Session Index	This field contains the session index used to define the controlled unit in the module from which the event was generated.
1	Sector Index	This field contains the sector index used to define the database within the controlled unit from which the event was generated.
2	COT	This field contains the COT for the event message received from the IED. If the size of the COT is a single byte, the originator address will always be zero. The COT is in the LSB and the originator address is in the MSB.
3	Reserved	This field is reserved for future use and is added here to keep the structure double-word aligned for all platforms.
4 to 5	Point Index	This field contains the point index in the remote device that generated the event.
6	ASDU Type	This field contains the ASDU type code for the data contained in the message.
7	Milliseconds and Seconds	This word contains the seconds and milliseconds when the event occurred.
8	Minutes and Hours	This field contains the minutes and hours the event occurred.
9	Month and Day	This field contains the month and day of the month the event occurred.
10	Year	This field contains the year the event occurred.
11	Qualifier	This field contains the point qualifier, quality or sequence value as described in the protocol specification.
12 to 13	Value	This field contains the double word value for the point associated with the event message.

The processor logic should recognize the event count value greater than zero and read all events in the block. After that this value should be reset to zero to prepare the logic for the next incoming block. Refer to the following topic that shows a sample function block for the event pass-thru functionality.

7.2.3 Data Type Mapping and Addressing

When interfacing data in the processor to that of the IEC 60870-5-101 protocol, it is important that the user understand the mapping of the data types to their corresponding representation in the modules database. The table that follows lists the data types supported by the module and their associated storage representation.

IEC-870-5-101 Data Types

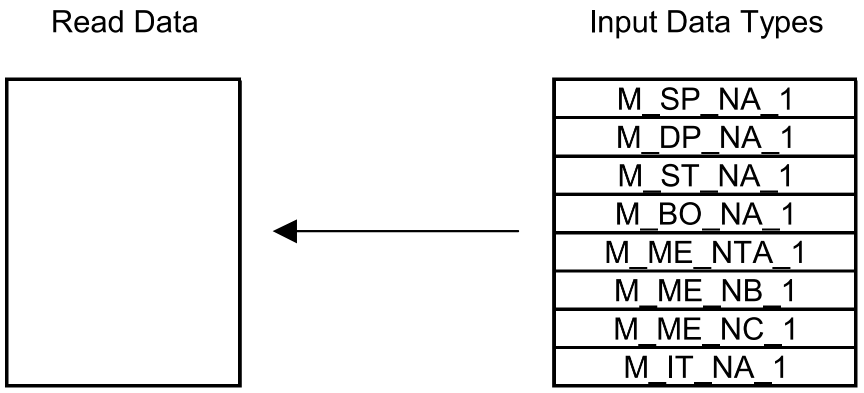
Type ID	Type	Description	Data representation
1	M_SP_NA_1 (7.3.1.1)	Monitored Single-point Information: This data type stores a single binary input point. Associated time-tagged event information for this type are M_SP_TA_1 (2) and M_SP_TB_1 (30).	Single bit value (7.2.6.1) with 0=Off and 1=On.
3	M_DP_NA_1 (7.3.1.3)	Monitored Dual-point Information: This data type stores a dual-point binary input value (that is, valve status). Associated time-tagged event information for this type are M_DP_TA_1 (4) and M_DP_TB_1 (31).	Dual-bit status (7.2.6.2) with 00b (0 decimal) = indeterminate or intermediate, 01b (1 decimal) = Off, 10b (2 decimal) = On and 11b (3 decimal) = indeterminate.
5	M_ST_NA_1 (7.3.1.5)	Monitored Step-point Information: This data type is used for step position of transformers or other step position information. The value for the position ranges from -64 to 63. Associated time-tagged event information for this type are M_ST_TA_1 (6) and M_ST_TB_1 (32).	Step data (7.2.6.5) is stored in a single character value with bits 0 to 6 (-64 to +63) representing the step position and bit 7 representing the following states: 0 = Equipment is not in transient state 1 = Equipment in transient state
7	M_BO_NA_1 (7.3.1.7)	Monitored Bitstring of 32-bit data --This data type stores 32-bit data in binary form. Each bit in the string has a value of 0 or 1. Associated time-tagged event information for this type are M_BO_TA_1 (8) and M_BO_TB_1 (33).	Each of the 32 bits in the bitstring has a value of 0 or 1 (7.2.6.13).
9	M_ME_NA_1 (7.3.1.9)	Monitored Normalized Measured Value: This data type is used for analog input data. Associated time-tagged event information for this type are M_ME_TA_1 (10) and M_ME_TD_1 (34).	Normalized values (7.2.6.6) are stored in a word (16-bit) data area with a range of -1..+1-2^{-15}
11	M_ME_NB_1 (7.3.1.11)	Monitored Scaled Measured Value --This data type is used for analog input data. Associated time-tagged event information for this type are M_ME_TB_1 (12) and M_ME_TE_1 (35).	Scaled values (7.2.6.7) are stored in a word (16-bit) data area with a range of -2^{15}.. +2^{15}-1

Type ID	Type	Description	Data representation
13	M_ME_NC_1 (7.3.1.13)	Monitored Measured Value, Short Floating-Point Number: This data type is used for analog input data stored in floating point format according to the IEEE STD 754, QDS format. Associated time-tagged event information for this type are M_ME_TC_1 (14) and M_ME_TE_1 (36).	Short floating-point number stored in IEEE STD 754 format (Fraction, Exponent, Sign) (7.2.6.8)
15	M_IT_NA_1 (7.3.1.15)	Monitored Integrated Total-point Information --This data type stores meter or other count data. Associated time-tagged event information for this type are M_IT_TA_1 (15)and M_IT_TB_1 (37).	Binary counter data (7.2.6.9) is stored in a double-word (32-bit) value with a range of -2^{31}..+2^{31}-1.
45	C_SC_NA_1 (7.3.2.1)	Single-point Command: This command controls a single binary point such as a relay.	Single bit value (7.2.6.15) with 0 = Off and 1 = On
46	C_DC_NA_1 (7.3.2.2)	Double-point Command: This command controls a dual-point binary control device such as a trip/close relay.	Double Command (7.2.6.16) with 0 = Not permitted 1 = Off 2 = On 3 = Not permitted
47	C_RC_NA_1 (7.3.2.3)	Regulating Step Command: This command controls a stepping device such as a transformer.	Regulating Step Command (7.2.6.17) with 0 = Not permitted 1 = Next step lower 2 = Next step higher 3 = Not permitted
48	C_SE_NA_1 (7.3.2.4)	Setpoint Command, Normalized Value: This command controls an analog device.	Normalized values (7.2.6.6) are stored in a word (16-bit)data area with a range of -1..+1-2^{-15}
49	C_SE_NB_1 (7.3.2.5)	Setpoint Command, Scaled Value: This command controls an analog device.	Scaled values (7.2.6.7) are stored in a word (16-bit) data area with a range of -2^{15}.. +2^{15}-1
50	C_SE_NC_1 (7.3.2.6)	Setpoint Command, Short Floating-Point Format: This command controls an analog device accepting an IEEE STD 754 floating-point format value.	Short floating-point number stored in IEEE STD 754 format (Fraction, Exponent, Sign) (7.2.6.8)
51	C_BO_NA_1 (7.3.2.7)	Setpoint Command, 32-bit Bitstring: This command controls a bitstring in a device.	Each of the 32 bits in the bitstring has a value of 0 or 1 (7.2.6.13).

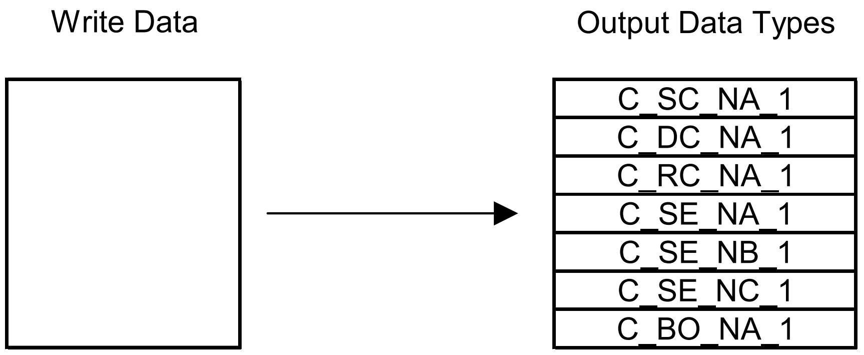
The data addressing is resumed in the following table:

Data	Size	Example
Single Point	1 bit	Address 1600 refers to word 100, bit 1 in database
Dual Point	2 bits	Address 1600 refers to word 100, bits 1 and 2 in database
Step Point	1 byte	Address 200 refers to word 100, lower byte in database
Bitstring 32 bit	2 words	Address 50 refers to word 100 and 101 in database
Normalized Measured Value	1 word	Address 100 refers to word 100 in database
Scaled Measured Value	1 word	Address 100 refers to word 100 in database
Short Float Point Measured Value	2 words	Address 50 refers to words 100 and 101 in database
Integrated Total Point	2 words	Address 50 refers to words 100 and 101 in database

Another important concept to understand is the direction of data transfer for the different data types with reference to the controller. The following illustration shows the data types (monitored data) that are transferred from the module to the processor.



The next diagram shows the movement of control data from the processor to the module. This data is then sent to the controlled devices on the serial networks.



As blocks are transferred between the module and the processor, each block contains block identification codes that define the content or function of the block of data transferred. The block identification codes used by the module are displayed in the following table:

Block Range	Descriptions
0	Null block. Used for handshaking
1	Read or write data. Used for handshaking
9250	Status Data
9901	User Constructed Command
9902	Command Control Block (Add command to Command List Queue)
9903	Event Messages from Master port
9950	Command List Error data
9970	Set PLC time using module's time
9971	Set module's time using PLC time
9998	Warm Boot Request from PLC (Block contains no data)
9999	Cold Boot Request from PLC (Block contains no data)

Block identification codes 9901 to 9999 are used for special control blocks to control the module. Each of these blocks is discussed in the following topics.

Normal Data Transfer

Normal data transfer includes the transferring of data received by or to be transmitted to the master drivers and the status data. These data are transferred through read (input image) and write (output image) blocks. Refer to Module Configuration for a description of the data objects used with the blocks and the ladder logic required. The following topics discuss the structure and function of each block.

Input Data (3x Register Data)

These blocks of data transfer information from the module to the Quantum processor. The structure of the input image used to transfer this data is shown in the following table:

Offset	Description	Length
0	Sequence Counter	1
1	Block ID	1
2 to 63	Command Response Data	62
64 to n	Read Data	0 to 3999

Output Data (4x Register Data)

These blocks of data transfer information from the Quantum processor to the module. The structure of the output image used to transfer this data is shown in the following table.

Offset	Description	Length
0	Sequence Counter	1
1	Block ID	1
2 to 63	Command Data	62
64 to n	Write Data	0 to 3999

Command Control Blocks

Block identification codes greater than 9900 are utilized to perform special functions in the module. Each control block recognized and used by the module is defined in the following topics.

Command Codes	Descriptions
9250	Status Block
9901	User Constructed Command
9902	Command Control Block (Add command to Command List Queue)
9903	Event Messages from Master port
9950	Command List Error data
9970	Set PLC time using module's time
9971	Set module's time using PLC time
9999	Cold Boot Request from PLC (Block contains no data)

Note: The command code in the I/O area is also referred to as the block ID.

Status Data Block (9250)

This block is used to request status data from the module by the processor.

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the block identification code of 9250 for the block.
2 to 63	Spare	Not used.

Block Format from Module (3x Register Data)

Offset	Parameter	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9250 for the block.
2	Scan Count	This status value contains a counter incremented on each scan of the module's main loop.
3 to 4	Product Name	This two-word data area contains the text values representing the product name.
5 to 6	Revision	This two-word data area contains the text values for the revision number.
7 to 8	Op Sys #	This two-word data area contains the text values for the operating system number.
9 to 10	Run Number	This two-word data area contains the text values for the run number.
11	Read Blk Cnt	This word contains the total number of block read operations successfully executed.
12	Write Blk Cnt	This word contains the total number of block write operations successfully executed.
13	Parse Blk Cnt	This word contains the total number of write blocks successfully parsed.
14	Error Blk Cnt	This word contains the total number of block transfer errors.
15	Event Msg Cnt	This word contains the number of event messages waiting to send to the processor.
16	Event Msg Overflow	This word contains a value of 0 if the event message buffer has not overflowed. If the event buffer overflows, this word will be set to a value of 1.
17	Session Count	This word contains the number of session configured in the module.
18	Current Cmd	This word contains the index of the current command being executed in the command list.
19	Cmd Busy Flag	This word is set to zero if no command is currently being executed and waiting on a response. If the word is set to 1, a command is currently executing.
20	Cmd Count	This word contains the count of the number of commands configured for the module.

Offset	Parameter	Description
21	Cmd Delay	This word contains the command delay counter preset. There is a fixed delay between each command to permit the module to perform class polls on controlled stations.
22	Cmd Queue	This word is set to zero if the command executing is from the command list. If the executing command is from the command queue, the word will be set to 1.
23	Cmd Queue Count	This word contains the number of active commands in the command queue for the module. Up to 100 commands can be buffered in this queue. These commands are transferred from the processor to the module using special command blocks.
24 to 25	Online Status	This double word value contains a bit for each of the 32 potential sessions in the module. If the bit is set for a session in the double word, the station is online. If the bit is clear, the station is offline. Use this value to determine if commands sent from the processor will have a chance of succeeding.
26	CH 0 State	This word contains the state machine value for channel 0.
27	Cmd Req	This word contains the number of commands transferred out channel 0.
28	Cmd Resp	This word contains the number of command response messages received on channel 0.
29	Cmd Err	This word contains the number of command errors recognized on channel 0.
30	Requests	This word contains the total number of messages transmitted on channel 0.
31	Responses	This word contains the total number of messages received on channel 0.
32	Err Sent	This word contains the number of error messages sent on channel 0.
33	Err Received	This word contains the number of error messages received on channel 0.
34	Cfg Err	This bit mapped word is used to recognized any configuration errors for channel 0. Refer to the configuration error word table for a definition of each bit.
35	Current Error	This word contains the error code for the current command executing on channel 0.
36	Last Error	This word contains the error code for the last error recognized on channel 0.
37	CH 1 State	This word contains the state machine value for channel 1.
38	Cmd Req	This word contains the number of commands transferred out channel 1.
39	Cmd Resp	This word contains the number of command response messages received on channel 1.
40	Cmd Err	This word contains the number of command errors recognized on channel 1.

Offset	Parameter	Description
41	Requests	This word contains the total number of messages transmitted on channel 1.
42	Responses	This word contains the total number of messages received on channel 1.
43	Err Sent	This word contains the number of error messages sent on channel 1.
44	Err Received	This word contains the number of error messages received on channel 1.
45	Cfg Err	This bit mapped word is used to recognized any configuration errors for channel 1. Refer to the configuration error word table for a definition of each bit.
46	Current Error	This word contains the error code for the current command executing on channel 1.
47	Last Error	This word contains the error code for the last error recognized on channel 1.
48 to 63	Spare	Not used

User Constructed Command Block (9901)

Block identification code 9901 is used to issue one or more user constructed commands. When the module receives a block 9901 identification code, it will place the included commands into the command queue.

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the block identification code of 9901 for the block.
2	Command Count	This field defines the number of user commands contained in the block. The valid range for the field is 1 to 10.
3 to 8	Command #1	Data required to build the user defined command in the command queue.
9 to 14	Command #2	Data required to build the user defined command in the command queue.
15 to 20	Command #3	Data required to build the user defined command in the command queue.
21 to 26	Command #4	Data required to build the user defined command in the command queue.
27 to 32	Command #5	Data required to build the user defined command in the command queue.
33 to 38	Command #6	Data required to build the user defined command in the command queue.
39 to 44	Command #7	Data required to build the user defined command in the command queue.
45 to 50	Command #8	Data required to build the user defined command in the command queue.

Word Offset in Block	Data Field(s)	Description
51 to 56	Command #9	Data required to build the user defined command in the command queue.
57 to 62	Command #10	Data required to build the user defined command in the command queue.
63	Spare	Not used.

The following fields are used for each 6-word record in the command list:

Word Offset	Definitions	Description
0	Database Index	Address in module to associate with the command
1	Session Index	Session index defined in the module to associate with the command.
2	Sector Index	Sector index for session as defined in the module.
3	Data Type	ASDU data type associated with the command.
4	Point Index	Information object address for the point on which command operates.
5	Qualifier	Qualifier as defined for the command list. This parameter is data type dependent.

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9901 for the block.
2 to 63	Spare	Not used.

Command Control Block (9902)

The block 9902 identification code is used by the PLC to send a list of commands to be placed in the command queue. Commands placed in the queue with this method need not have their enable bit set.

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the value of 9902 identifying the enable command to the module.
2	Command count	This field contains the number of commands to enable in the command list. Valid values for this field are 1 to 60.
3 to 62	Command Numbers to enable	These 60 words of data contain the command numbers in the command list to enable. The commands in the list will be placed in the command queue for immediate processing by the module. The first command in the list has an index of 0.
63	Spare	Not Used

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9902 for the block.

Event Message Block (9903)

Block identification code 9903 is used to request event messages received on the master port for the processor.

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the value of 9903 to request event data
2 to 63	Spare	Not used

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9903 for the block.
2	Event Count	This field contains the number of events present in the block. Values of 1 to 4 are valid.
3-16	Event 1	Event message
17-30	Event 2	Event message
31-44	Event 3	Event message
45-58	Event 4	Event message
59-61	Spare	Not used
62	Event count in queue	Number of events in queue still waiting to send
63	Event Overflow	Event buffer overflow

The format of each 14 word data region in the block is as follows:

Word Offset	Definitions	Description
0	Session Index	This field contains the session index used to define the controlled unit in the module from which the event was generated.
1	Sector Index	This field contains the sector index used to define the database within the controlled unit from which the event was generated.

Word Offset	Definitions	Description
2	COT	This field contains the COT for the event message received from the IED. If the size of the COT is a single byte, the originator address will always be zero. The COT is in the LSB and the originator address is in the MSB.
3	Reserved	This field is reserved for future use and is added here to keep the structure double-word aligned for all platforms.
4 to 5	Point Index	This field contains the point index in the remote device that generated the event.
6	ASDU Type	This field contains the ASDU type code for the data contained in the message.
7	Milliseconds and Seconds	This word contains the seconds and milliseconds when the event occurred.
8	Minutes and Hours	This field contains the minutes and hours the event occurred.
9	Month and Day	This field contains the month and day of the month the event occurred.
10	Year	This field contains the year the event occurred.
11	Qualifier	This field contains the point qualifier, quality or sequence value as described in the protocol specification.
12 to 13	Value	This field contains the a double word value for the point associated with the event message.

Command List Error Data Block (9950)

Block 9950 identification code is used to request the Command List Error Table from the module. The format for the block is shown below:

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the value of 9950 identifying the block type to the module.
2	Number of Commands to report	This field contains the number of commands to report in the response message. The value has a range of 1 to 60.
3	Start Index of First Command	This parameter sets the index in the command list where to start. The first command in the list has a value of 0. The last index in the list has a value of MaxCommands - 1.
4 to 63	Spare	Not Used

Response to a block 9950 request: The module will respond to a valid request with a block containing the requested error information. The format for the block is shown below:

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the value of 9950 identifying the block type to the PLC.
2	Number of Commands reported	This field contains the number of commands contained in the block that need to be processed by the PLC. This field will have a value of 1 to 60.
3	Start Index of First Command	This field contains the index in the command list for the first value in the file. This field will have a value of 0 to MaxCommands-1.
4 to 63	Command List Errors	Each word of this area contains the last error value recorded for the command. The command index of the first value (offset 4) is specified in word 3 of the block. The number of valid command errors in the block is set in word 2 of the block. Refer to the command error list to interpret the error codes reported.

Get Module Date and Time Block (9970)

Block 9970 identification code is used to request the module's date and time. This data can be used to set the PLC clock.

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the value of 9970 identifying the block type to the module.
2 to 63	Not Used	Not Used

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9970 for the block.
2	Year	This field contains the four-digit year to be used with the new time value.
3	Month	This field contains the month value for the new time. Valid entry for this field is in the range of 1 to 12.
4	Day	This field contains the day value for the new time. Valid entry for this field is in the range of 1 to 31.
5	Hour	This field contains the hour value for the new time. Valid entry for this field is in the range of 0 to 23.
6	Minute	This field contains the minute value for the new time. Valid entry for this field is in the range of 0 to 59.
7	Seconds	This field contains the second value for the new time. Valid entry for this field is in the range of 0 to 59.
8	Milliseconds	This field contains the millisecond value for the new time. Valid entry for this field is in the range of 0 to 999.
9 to 63	Not Used	Not Used

Set Module Time Block (9971)

Block identification code 9971 is used to pass the clock time in the PLC to the module. The date and time provided will be used to set the module's clock.

Block Format from Processor (4x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the user wishes to request a new command block.
1	Block ID	This field contains the block identification code of 9971 for the block.
2	Year	This field contains the four-digit year to be used with the new time value.
3	Month	This field contains the month value for the new time. Valid entry for this field is in the range of 1 to 12.
4	Day	This field contains the day value for the new time. Valid entry for this field is in the range of 1 to 31.
5	Hour	This field contains the hour value for the new time. Valid entry for this field is in the range of 0 to 23.
6	Minute	This field contains the minute value for the new time. Valid entry for this field is in the range of 0 to 59.
7	Seconds	This field contains the second value for the new time. Valid entry for this field is in the range of 0 to 59.
8	Milliseconds	This field contains the millisecond value for the new time. Valid entry for this field is in the range of 0 to 999.
9 to 63	Spare	Not Used

Block Format from Module (3x Register Data)

Word Offset in Block	Data Field(s)	Description
0	Sequence Counter	This field contains a new value each time the block is handled.
1	Block ID	This field contains the block identification code of 9971 for the block.
2 to 63	Spare	Not Used

Cold Boot Block (9999)

Block 9999 performs a cold-boot operation on the module. The format of the block constructed by the processor is as follows:

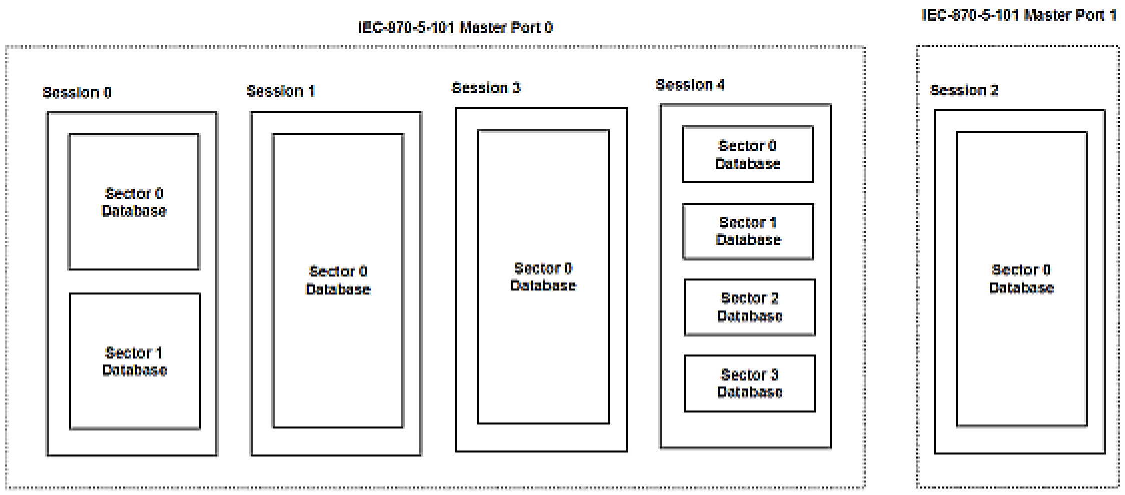
Offset	Description	Length
0	9990	1
1 to 63	Spare	63

In this version of the module, the warm and cold boot processes perform the same operation, because many of the variables that must be initialized are fixed when the module first boots and cannot be changed after the application starts.

7.2.4 Master Driver

The master driver supported on each application port of the module emulates an IEC 60870-5-101 master device. Configuration of each port is independent and should be connected to different serial networks.

Each port on the module communicates with one or more controlled stations on what are referred to as sessions. A session represents a controlled device with a unique data link layer address. Each session (controlled device) contains one or more data sets (sectors) that are defined by the vendor of the device. The following illustration shows these relationships.

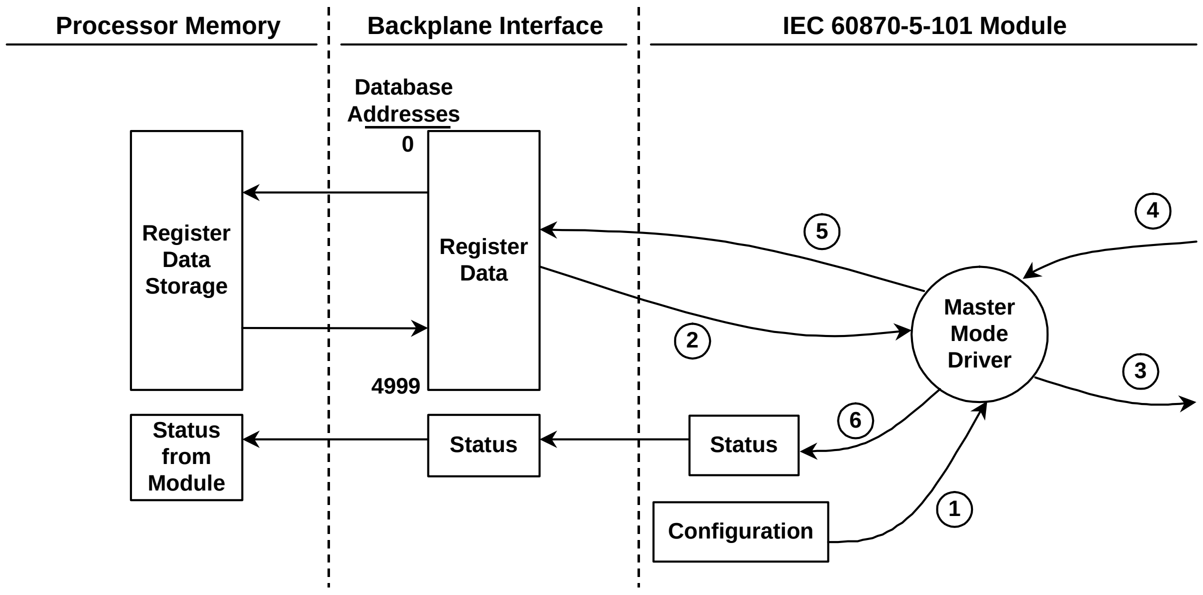


Port 0 on the module communicates with 4 sessions (0, 1, 3 and 4) each of which has their own data set(s). Session 1 only has one sector (all data for device contained in a single database). This sector is addressed by the master using the Common address of ASDU value set for the sector in the configuration file. Session 0 contains two sectors each with their own unique Common address of ASDU value to identify the sector. Port 0 must operate in unbalanced mode as more than one device exists on the network.

Port 1 may operate in balanced mode as it only contains one device on the network. This device is defined in the Session 2 section of the configuration file. In this example, all data of the device is stored in a single sector.

The module supports two application ports. Thirty-two session can be defined on the module with each session being assigned to an application port. Within each session, up to five sectors can be defined. This system permits a very flexible assignment of resources in the module. The definition of the data associated with each sector in the system is defined by the user in the configuration file.

The following diagram shows the functionality of the master driver:



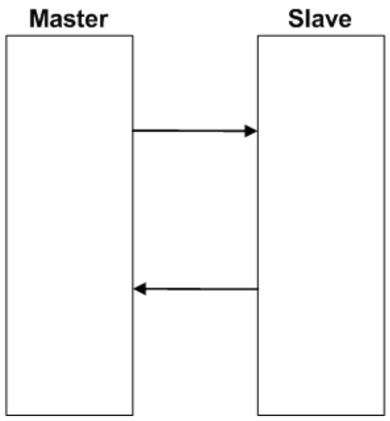
1 The master driver is configured as specified by the IEC101M.CFG file
2 The master will construct control commands using the data in the database
3 The master will send these commands and class polls out on the serial network
4 Response messages or spontaneous messages generated by controlled devices on the serial network are received by the master driver
5 Monitor data (static and event) received by the master is passed to the module's database and passed to the processor
6 Additionally, status data for the module is passed to the processor

7.3 IEC-60870-5-101 Protocol Implementation

This section presents an overview of how the PTQ-101M works, while skipping the complex details of the protocol specification. If you require more information about the implementation of the protocol, refer to the protocol specification (IEC 60870-5-101 2003). For more information on the configuring and modifying the backplane implementation of the protocol with the PTQ-101M, refer to Customizing the Sample Configuration File.

The IEC-60870-5-101 protocol applies to telecontrol equipment and systems with coded bit serial data transmission for monitoring and controlling geographically widespread processes.

Any application using the IEC-60870-5-101 protocol will have a master (controlling station) and one or more slaves (controlled stations). The master will constantly monitor and control the data from each slave in the network.



This module works as an IEC-60870-5-101 Master. It can receive monitor data, and control (sends commands to slaves) and receive events from slaves, as explained in the following topics.

7.3.1 General Parameter Configuration

Communication Parameters

The following parameters should be configured for serial communication:

```
Baud Rate         : 9600 #Baud rate for port 110 to 38400
Parity            :    2 #0=None, 1=Odd, 2=Even
RTS On            :    0 #0 to 65536 mSec before message
RTS Off           :    0 #0 to 65536 mSec after message
Minimum Delay     :   20 #Minimum # of mSec before response sent
Receive Timeout   : 2000 #Maximum mSec from first char to last to wait
Hardware Handshaking:  0 #Hardware handshaking 0=None, 1=RTS/CTS, 2=DTR/DSR,
                         #3=modem
```

Each one of these parameters should be adjusted for each specific application.

Data Link Configuration

The protocol specification document IEC 60870-5-2 specifies an unambiguous address (number) for each link. Each address may be unique within a specific system, or may be unique within a group of links sharing a common channel. The protocol specification defines that the Data Link Address may have 0, 1 or 2 octets. The first option should be used only during Balanced Mode.

Configure the following parameter to set the number of octets to be used for the Data Link Address value. It is essential that the slave unit also uses the same number of octets configured for the PTQ-101M.

```
Data link address length  :     2 #0, 1, or 2 octets used for DL address
```

You must also configure the actual Data Link Address value using the following parameter:

```
Data link address         :     1 #Range depends on the configured DL
                                     #Length
```

This value identifies the module's address in the network.

ASDU Configuration

The protocol specification document IEC 60870-5-3 describes the Basic Application Data Units that are used in the protocol. It also defines the Application Service Data Unit (ASDU) used by the protocol for data communication.

You can configure the number of bytes to be used for the following ASDU components:

Component	Abbreviation	Possible Number of Octets
Cause of Transmission	COT	1 or 2
Common Address of ASDU	CASDU	1 or 2
Information Object Address	IOA	1, 2 or 3

These parameters must be configured to match the master's configuration, as shown in the following example:

```
Common Address of ASDU len:     2 #Range is 1 or 2 octets
Inform. Object Address len:     3 #Range is 1, 2, or 3 octets
Cause of Trans Octets     :     2 #Number of COT octets (1 or 2)
```

The Common Address of ASDU is the station address. The module only allows the addressing of the whole station (some devices allows different Common Addresses to identify particular station sectors). You should configure the Common ASDU Address with the following parameter:

```
Common Address of ASDU    :     1 # Range depends on number of octets for
                                   # CASDU
```

You should also configure the maximum number of bytes that the slave should support for each ASDU response to the Module. The range lies between 25 and 252 bytes. Configure the following parameter in the slave unit to set the maximum number of bytes to be transferred at every ASDU response.

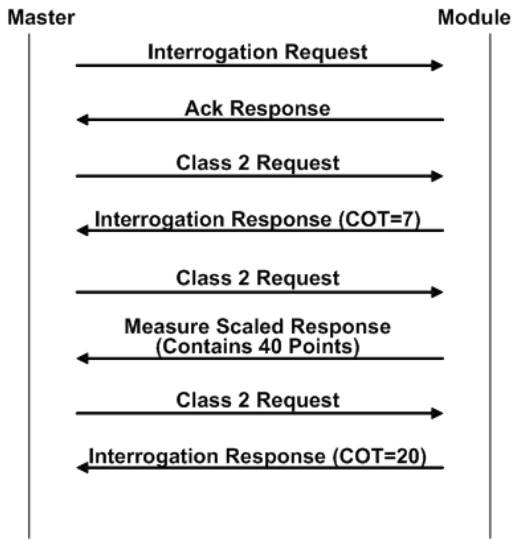
```
Maximum ASDU Resp Len     :   252 #Max ASDU response message
```

A value less than 252 causes the slave response to break down the message into multiple parts, requiring more response messages.

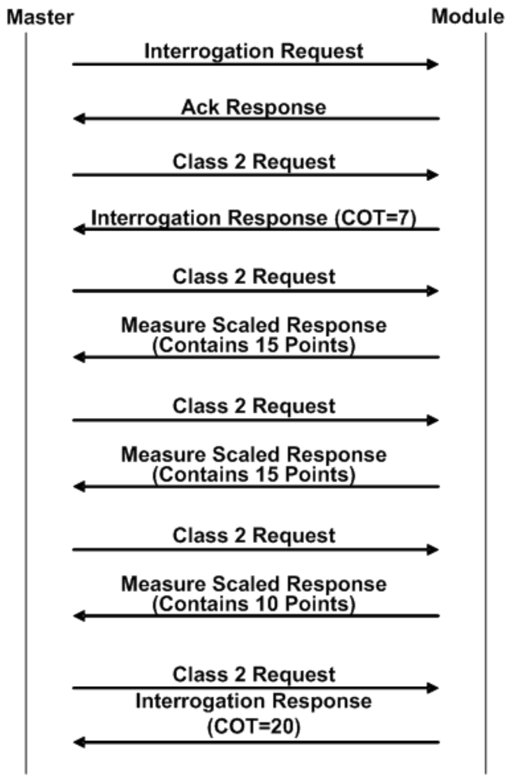
Example - Changing the ASDU Length:

Considering that the Module sends a General Interrogation request to poll forty M_ME_NB points (measured scaled points in Unbalanced Mode):

If Maximum ASDU Resp Len = 252:


Master
Module
Interrogation Request
Ack Response
Class 2 Request
Interrogation Response (COT=7)
Class 2 Request
Measure Scaled Response (Contains 40 Points)
Class 2 Request
Interrogation Response (COT=20)

This example shows that the slave sends all 40 measured scaled points in one single message, if the message is not greater than 252 bytes. However, some master devices may not support messages containing this number of bytes. If the master for a given application supports only 100 bytes, the following communication procedure would occur:


Master
Module
Interrogation Request
Ack Response
Class 2 Request
Interrogation Response (COT=7)
Class 2 Request
Measure Scaled Response (Contains 15 Points)
Class 2 Request
Measure Scaled Response (Contains 15 Points)
Class 2 Request
Measure Scaled Response (Contains 10 Points)
Class 2 Request
Interrogation Response (COT=20)

As shown in the previous diagram, the module sends out 15 messages (15 points, 15 points and 10 points) instead of sending the whole 40 points in one single message.

Note: This example shows the case where IOA Length = 3 bytes, COT Length = 2 bytes and CASDU Length = 2 bytes.
Important: If the database parser gets a point index that is not valid, the whole database is set as invalid and no points are reported. Because the index 0 is not valid (the protocol does not support this index value), the driver considers it invalid. For example, if you set the size of the ASDU to 1 and you set a value of 1000 for a point index, this is also invalid as the indexes can only go from 1 to 255.

Balanced and Unbalanced Modes

The module supports balanced and unbalanced modes. In balanced mode, each station may initiate message transfers. If the links from the master unit to several slaves share a common physical channel, then these links must be operated in an unbalanced mode to avoid the possibility of more than one device attempting to transmit on the channel at the same time.

Select the communication mode with the following parameter:

```
Use Balanced Mode  :      0 #0=No, 1=Use balanced mode
```

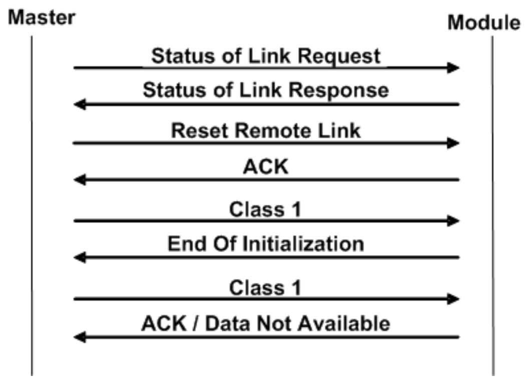
7.3.2 Module Initialization

After the module powers up, a specific initialization procedure occurs, depending on the communication mode you selected (Balanced or Unbalanced).

Unbalanced mode

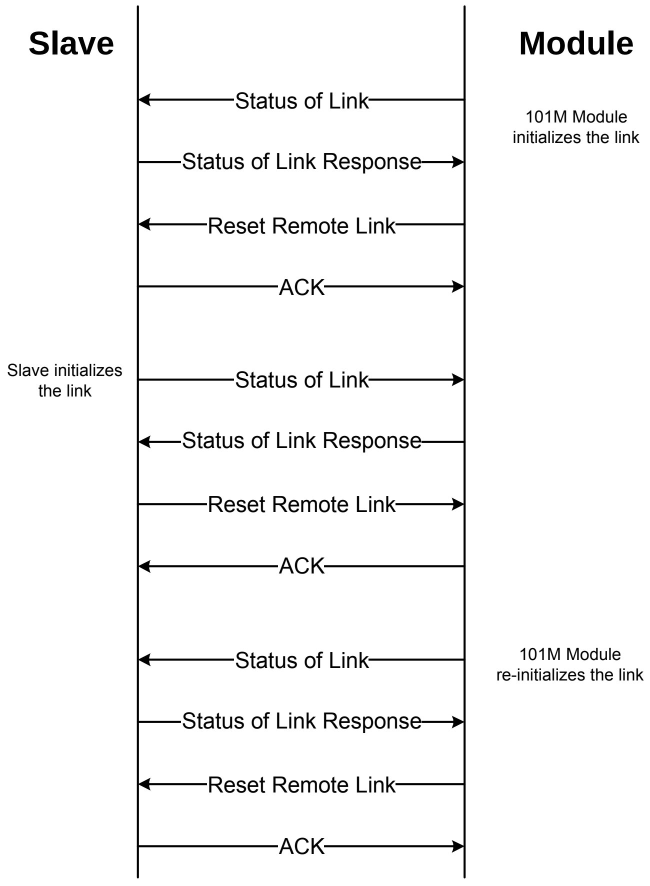
In order to start communications between the master and the slave units, the master tries to establish the link connection by transmitting repeated "Request Status of Link" at specific time out intervals. When the module's link is available it will respond with a "Status of Link" response. Then, the master transmits a "Reset of Remote Link" message and the slave responds with an ACK response. Then the master sends two consecutive Class 1 requests. The slave responds to the first Class 1 request with an "End of Initialization" response and the second Class 1 request with an Ack message.

The following illustration shows a typical initialization procedure for the unbalanced mode:



Balanced Mode

During balanced mode, the link must be initialized in both directions. The PTQ-101M module also always reinitializes the link after it receives an initialization request from the master. Therefore, the following initialization occurs during balanced mode, after PTQ-101M boots.

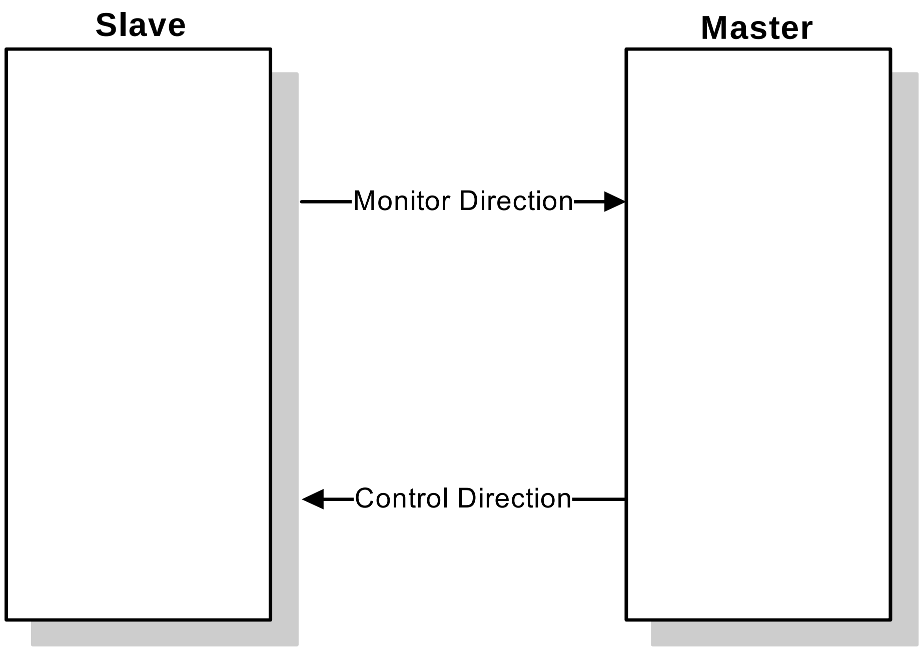


After the initialization procedure is completed, the master and the PTQ-101M start communicating. If during communication the master fails to respond to a message from the module, the PTQ-101M will retry for a configured number of times. If the master still fails to respond, the module will initialize the line again.

7.3.3 Monitor Direction and Control Direction

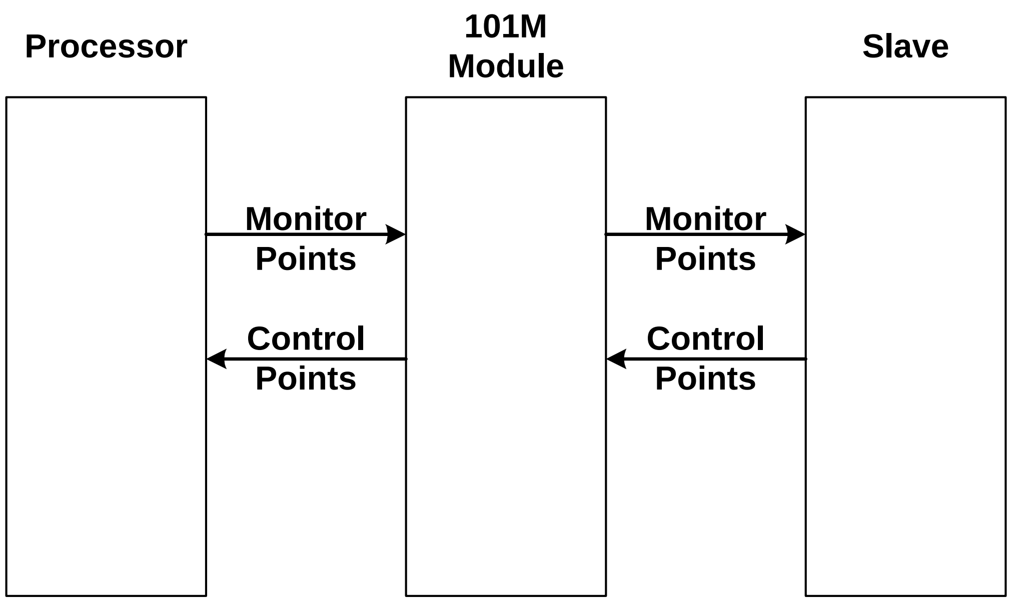
The protocol specification defines two directions of data: monitor direction and control direction. These directions are defined by the protocol specification as follows:

- **Monitor Direction**: The direction of transmission from a slave to the master
- **Control Direction**: The direction of transmission from the master to a slave



The points that are typically transferred from the slave to the master are also known as Monitor Points (or Monitor Information Objects). The points that are typically transferred from the master to the slave are also known as Control Points (or Command Information Objects).

For the PTQ-101M, the control and monitor points would be transferred as shown in the following illustration.



You must make sure that all points are configured in the correct location in the PTQ-101M module database in order to be properly updated from/to the processor.

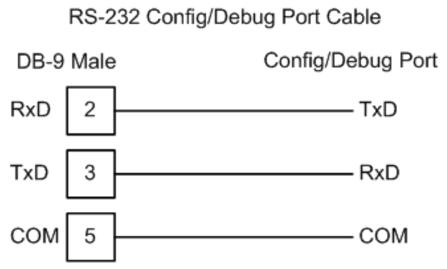
7.4 Cable Connections

The application ports on the PTQ-101M module support RS-232, RS-422, and RS-485 interfaces. Please inspect the module to ensure that the jumpers are set correctly to correspond with the type of interface you are using.

Note: When using RS-232 with radio modem applications, some radios or modems require hardware handshaking (control and monitoring of modem signal lines). Enable this in the configuration of the module by setting the UseCTS parameter to 1.

7.4.1 RS-232 Configuration/Debug Port

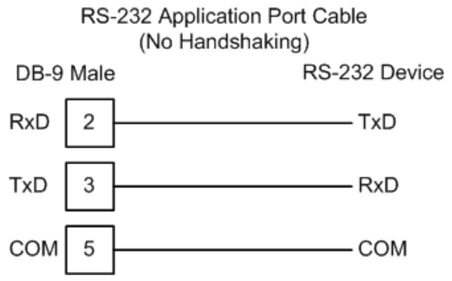
This port is physically a DB-9 connection. This port permits a PC based terminal emulation program to view configuration and status data in the module and to control the module. The cable for communications on this port is shown in the following diagram:



The Ethernet port on this module (if present) is inactive.

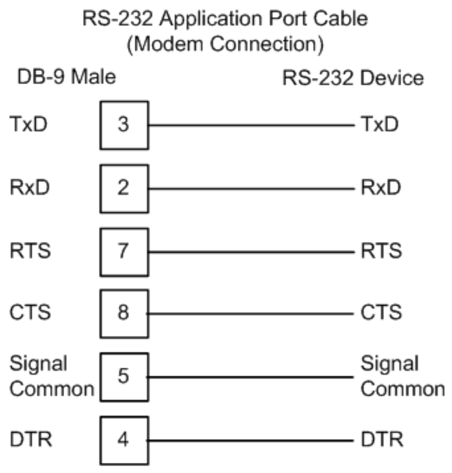
7.4.2 RS-232

When the RS-232 interface is selected, the use of hardware handshaking (control and monitoring of modem signal lines) is user definable. If no hardware handshaking will be used, the cable to connect to the port is as shown below:



RS-232: Modem Connection

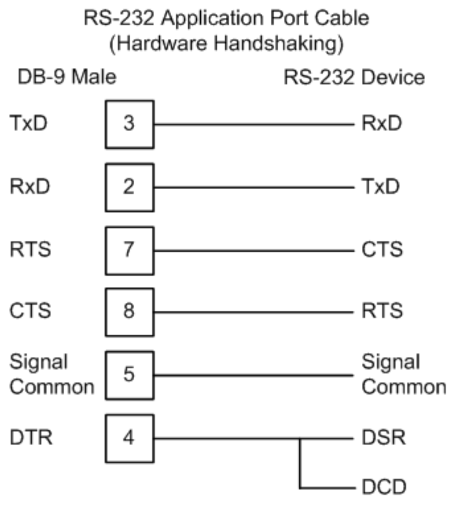
This type of connection is required between the module and a modem or other communication device.



The "Use CTS Line" parameter for the port configuration should be set to 'Y' for most modem applications.

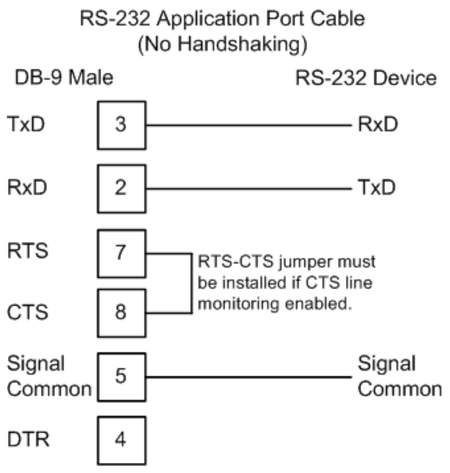
RS-232: Null Modem Connection (Hardware Handshaking)

This type of connection is used when the device connected to the module requires hardware handshaking (control and monitoring of modem signal lines).



RS-232: Null Modem Connection (No Hardware Handshaking)

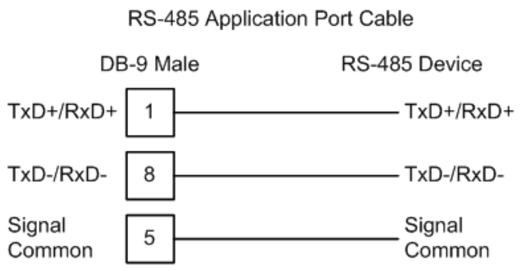
This type of connection can be used to connect the module to a computer or field device communication port.


RS-232 Application Port Cable
(No Handshaking)
DB-9 Male
RS-232 Device
TxD
3
RxD
RxD
2
TxD
RTS
7
CTS
8
RTS-CTS jumper must be installed if CTS line monitoring enabled.
Signal Common
5
Signal Common
DTR
4

Note: If the port is configured with the "Use CTS Line" set to 'Y', then a jumper is required between the RTS and the CTS line on the module connection.

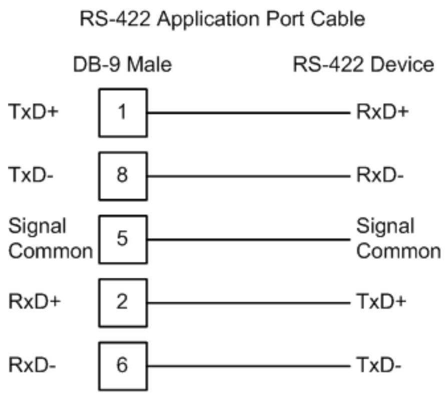
7.4.3 RS-485

The RS-485 interface requires a single two or three wire cable. The Common connection is optional and dependent on the RS-485 network. The cable required for this interface is shown below:



Note: Terminating resistors are generally not required on the RS-485 network, unless you are experiencing communication problems that can be attributed to signal echoes or reflections. In this case, install a 120 ohm terminating resistor on the RS-485 line.

7.4.4 RS-422



RS-485 and RS-422 Tip

If communication in the RS-422/RS-485 mode does not work at first, despite all attempts, try switching termination polarities. Some manufacturers interpret +/- and A/B polarities differently.

7.5 PTQ-101M Status Data Area

This section contains a listing of the data contained in the PTQ-101M status data object, configuration error word and module error codes.

7.5.1 Error/Status Data Format

Offset	Parameter	Description
0	Scan Count	This status value contains a counter incremented on each scan of the module's main loop.
1 to 2	Product Name	This two-word data area contains the text values representing the product name. These words contain the text '87S5' for the PTQ platform.
3 to 4	Revision	This two-word data area contains the text values for the revision number.
5 to 6	Op Sys #	This two-word data area contains the text values for the operating system number.
7 to 8	Run Number	This two-word data area contains the text values for the run number.
9	Read Blk Cnt	This word contains the total number of block read operations successfully executed.
10	Write Blk Cnt	This word contains the total number of block write operations successfully executed.
11	Parse Blk Cnt	This word contains the total number of write blocks successfully parsed.
12	Error Blk Cnt	This word contains the total number of block transfer errors.
13	Event Msg Cnt	This word contains the number of event messages waiting to send to the processor.

Offset	Parameter	Description
14	Event Msg Overflow	This word contains a value of 0 if the event message buffer has not overflowed. If the event buffer overflows, this word will be set to a value of 1.
15	Session Count	This word contains the number of session configured in the module.
16	Current Cmd	This word contains the index of the current command being executed in the command list.
17	Cmd Busy Flag	This word is set to zero if no command is currently being executed and waiting on a response. If the word is set to 1, a command is currently executing.
18	Cmd Count	This word contains the count of the number of commands configured for the module.
19	Cmd Delay	This word contains the command delay counter preset. There is a fixed delay between each command to permit the module to perform class polls on controlled stations.
20	Cmd Queue	This word is set to zero if the command executing is from the command list. If the executing command is from the command queue, the word will be set to 1.
21	Cmd Queue Count	This word contains the number of active commands in the command queue for the module. Up to 100 commands can be buffered in this queue. These commands are transferred from the processor to the module using special command blocks.
22 to 23	Online Status	This double word value contains a bit for each of the 32 potential sessions in the module. If the bit is set for a session in the double word, the station is online. If the bit is clear, the station is offline. Use this value to determine if commands sent from the processor will have a chance of succeeding.
24	CH 0 State	This word contains the state machine value for channel 0.
25	Cmd Req	This word contains the number of commands transferred out channel 0.
26	Cmd Resp	This word contains the number of command response messages received on channel 0.
27	Cmd Err	This word contains the number of command errors recognized on channel 0.
28	Requests	This word contains the total number of messages transmitted on channel 0.
29	Responses	This word contains the total number of messages received on channel 0.
30	Err Sent	This word contains the number of error messages sent on channel 0.
31	Err Received	This word contains the number of error messages received on channel 0.
32	Cfg Err	This bit mapped word recognizes any configuration errors for channel 0. Refer to the configuration error word table for a definition of each bit.
33	Current Error	This word contains the error code for the current command executing on channel 0.
34	Last Error	This word contains the error code for the last error recognized on channel 0.
35	CH 1 State	This word contains the state machine value for channel 1.
36	Cmd Req	This word contains the number of commands transferred out channel 1.

Offset	Parameter	Description
37	Cmd Resp	This word contains the number of command response messages received on channel 1.
38	Cmd Err	This word contains the number of command errors recognized on channel 1.
39	Requests	This word contains the total number of messages transmitted on channel 1.
40	Responses	This word contains the total number of messages received on channel 1.
41	Err Sent	This word contains the number of error messages sent on channel 1.
42	Err Received	This word contains the number of error messages received on channel 1.
43	Cfg Err	This bit mapped word recognizes any configuration errors for channel 1. Refer to the configuration error word table for a definition of each bit.
44	Current Error	This word contains the error code for the current command executing on channel 1.
45	Last Error	This word contains the error code for the last error recognized on channel 1.

The following table defines the contents of the configuration error word. Each bit in the word corresponds to an error condition recognized when the module is configured. There is a separate word for each application port. This data is reported in the status data area previously defined.

Bit	Code	Description
0	0x0001	Invalid baud rate selected
1	0x0002	Invalid parity selected
2	0x0004	Received timeout set to 0
3	0x0008	Invalid Port selected for a session
4	0x0010	Invalid sector count for session
5	0x0020	Could not allocate memory for sector of a session.
6	0x0040	Invalid length data for session: ▪ Data link length ▪ Command address of ASDU length ▪ Information object address length ▪ COT octet count
7	0x0080	Invalid failure delay or confirm timeout for session.
8	0x0100	
9	0x0200	
10	0x0400	
11	0x0800	
12	0x1000	
13	0x2000	
14	0x4000	
15	0x8000	

7.5.2 Error Codes

The following table lists all potential errors that can be generated by the IEC 60870-5-101 master driver:

Error	Description
51	Physical layer error - Error transmitting message
52	Physical layer error - Intercharacter timeout occurred before message fully received.
53	Physical layer error - Frame not entirely received before timeout condition.
54	Physical layer error - Invalid frame length.
101	Link layer error - Invalid checksum received
102	Link layer error - Address unknown to module
103	Link layer error - Link established
104	Link layer error - Link failed
105	Link layer error - Received primary
106	Link layer error - FCB error discard
107	Link layer error - FCB error repeat
108	Link layer error - Invalid start character received
109	Link layer error - Invalid second character received
110	Link layer error - Invalid ending character received
111	Link layer error - Length mismatch error
112	Link layer error - Illegal function
113	Link layer error - No confirmation received
114	Link layer error - No ACK received
115	Link layer error - Sequence unknown
116	Link layer error - Out of sequence
117	Link layer error - Remote close
118	Link layer error - Unexpected ACK
119	Link layer error - Request cancelled
201	Application layer error - Length mismatch
202	Application layer error - Address unknown
203	Application layer error - Response late
251	RBE error - Clock event buffer overflow
252	RBE error - Event buffer overflow
271	Data error - Address unknown
281	Control error - Illegal operation
282	Control error - Illegal value
283	Control error - Not selected
301	Initialization error - Database
302	Initialization error - Out of memory
401	Channel open error
501	Session error - Database
502	Session error - Configuration
601	No memory to receive message

Error	Description
602	Session not reserved
603	Illegal session
604	Session is reserved
605	Session is not available
701	No memory to transmit message
702	ASDU not supported
703	Duplicate request
704	Illegal sector
705	Control mode is illegal
801	Partial stop request
802	Stop request failed
901	Response timeout
902	Negative COT in response
903	Session is offline
904	Session is disabled
905	Select confirmation received, waiting to execute
906	Execute confirmation has not be received

7.6 Configuration Data Definition

This section contains a listing of the parameters and their definitions for the PTQ-101M module configuration file definition (IEC101M.CFG).

[Section]/Item	Value	Range	Description
[Backplane Configuration]			Backplane transfer parameters
Module Name:		0 to 80 characters	This parameter assigns a name to the module that can be viewed using the configuration/debug port. Use this parameter to identify the module and the configuration file.
4x Register Start:		1 to n	This parameter sets the first register in the processor where the data transferred from the processor to the module is present.
Write Register Start:		0 to 3999	This parameter specifies the starting register in the module where the data transferred from the processor will be placed. Valid range for this parameter is 0 to 3999.
Write Register Count:		0 to 4000	This parameter specifies the number of registers to transfer from the processor to the module. Valid entry for this parameter is 0 to 4000.
3x Register Start:		1 to n	This parameter sets the first register in the processor where the data transferred from the module to the processor will be placed.
Read Register Start:		0 to 3999	This parameter specifies the starting register in the module where data will be transferred from the module to the processor. Valid range for this parameter is 0 to 3999.

[Section]/Item	Value	Range	Description
Read Register Count:		0 to 4000	This parameter specifies the number of registers to be transferred from the module to the processor. Valid entry for this parameter is 0 to 4000.
Pass-Through Events:		Yes or No	This parameter specifies if event messages received on the master ports will be passed to the processor. If the parameter is set to No, event messages will not be passed to the processor. If the parameter is set to Yes, the module will pass all events received to the processor using block identifier 9903.

[Section]/Item	Value	Range	Description
[IEC 60870-5-101 MASTER]			Module level parameters
Session Count:		1 to 32	This parameter specifies the maximum number of session to establish on the module. This corresponds to the number of slaves to be interfaced with the module. This value represents the total number of slaves on both ports combined.

[Section]/Item	Value	Range	Description
[IEC 60870-5-101 MASTER PORT x]			Settings for each communication port on module
Baud Rate:		Value for baud rate	This parameter specifies the baud rate to be used on the communication channel (port). Values from 110 to 38.4K are permitted.
Parity:		None, Odd, Even	This parameter sets the parity to be used on the port. The values correspond to the following settings: None, Odd, Even.
RTS On:		0 to 65535	The parameter sets the RTS pre-send delay. The value entered represents the number of milliseconds the module will wait after setting the RTS modem line before sending the data.
RTS Off:		1 to 65535	This parameter sets the RTS off delay. The value entered represents the number of milliseconds the module will wait after the data packet is sent before dropping the RTS modem line.
Minimum Delay:		1 to 65535	This parameter specifies the minimum number of milliseconds to delay before sending the message (setting RTS high). This can be used when the serial network requires time for units to turn off their transmitters.
Receive Timeout:		1 to 65535	This value represents the number of milliseconds to wait on a port from the time the first character is received until the last character in the longest message received on the port. This parameter should be set dependent on the baud rate. A value of 2000 should work with most applications.
Single char ACK F0,1 or 3:		Yes or No	This parameter specifies if the single E5 character will be used for ACK messages.

[Section]/Item	Value	Range	Description
Use Balanced Mode:		Yes or No	This parameter specifies if the port will use balanced mode. If balanced mode is used, only one controlled station will be permitted on the port. If unbalanced mode is used, multiple controlled stations can be used on a port. Select Yes to use balanced mode and No to use unbalanced mode.

[Section]/Item	Value	Range	Description
[IEC-101 MASTER SESSION x]			Settings for each session utilized
Communication Port:		0 or 1	This parameter sets the port to which the controlled device is connected. On this module, values of 0 and 1 are permitted.
Sector Count:		1 to 5	This parameter sets the number of sectors contained in this controlled device. The range of values is from 1 to 5. A sector section is required for each sector in a session to define its database and settings.
DL Address Length:		0, 1 or 2	This parameter sets the number of octets used to define the data link address for the session. A value of 0 is only permitted when balanced mode is used.
Data Link Address:		0 to 254 or 0 to 65534	This parameter uniquely defines the data link address for this unit on the communication channel The ranges of values depends on the value set in the DL Address Length parameter.
Common Address of ASDU Len:		1 or 2	This parameter sets the number of octets used for the common address of ASDU for each sector for this session.
Inform. Object Address Len:		1, 2 or 3	This parameter sets the number of octets used to specify the address for an information object in each sector for this session.
COT Octet Count:		1 or 2	This parameter sets the number of octets used for the COT field in each message. If a value of 2 is selected, the value entered for the Originator Address For COT will accompany each message from the controlling unit.
Originator Address For COT:		0 to 255	This parameter sets the address to be passed with each message when the COT Octet Count parameter is set to 2.
Failure Delay:		0 to 2000	This parameter sets the minimum number of seconds to delay before polling this session when it is not online. This parameter is only used in unbalanced mode.
Confirm Timeout:		0 to 2^32-1	This parameter sets the number of milliseconds to wait for a confirm response from the controlled device.
Retry Count:		0 to 255	This parameter sets the number of retries to be performed on the controlled device when a communication error occurs.

[Section]/Item	Value	Range	Description
C1/C2 Poll Count Pend:		0 to 65535	This parameter sets the maximum number of class 1 and class 2 polls performed on this session before trying the next session. This parameter prevents a session from monopolizing the communication port.
Class 1 Polls:		0 to 65535	This parameter sets the maximum number of class one polls performed on this session before switching to another session. This parameter prevents a session from monopolizing the communication port.
Class 1 Pend Delay:		0 to 2^32-1	This parameter sets the minimum number of milliseconds to delay between class 1 polls for pending data.
Class 2 Pend Delay:		0 to 2^32-1	This parameter sets the minimum number of milliseconds to delay between class 2 polls for pending data.
Class 1 Poll Delay:		0 to 2^32-1	This parameter sets the minimum number of milliseconds to delay between each class 1 poll.
Class 2 Poll Delay:		0 to 2^32-1	This parameter sets the minimum number of milliseconds to delay between each class 2 poll.
Auto Clock Req Mode:		0, 1 or 2	This parameter specifies the method used to perform automatic clock synchronization. 0 performs a synchronization without delay, 1 performs synchronization using the fixed Propagation Delay and 2 computes the delay and use this value when synchronization takes place.
Propagation Delay:		0 to 65535	This parameter sets the fixed propagation delay to be utilized if the Auto Clock Req Mode parameter is set to a value of 1.
Response Timeout:		0 to 2^32-1	This parameter sets the maximum number of milliseconds to wait for a confirmation from the controlled station to a request from this module.
ACTTERM with setpoint:		Yes or No	This parameter specifies what the last message will be in the response to a setpoint command. If the parameter is set to Yes, an ACTTERM will be the last response, and if set to No, ACTCON will be the last response.

[Section]/Item	Value	Range	Description
[IEC-101 MASTER SESSION x SECTOR y]			This section sets the parameters for a specific sector of a session.
Common ASDU Address:		0 to 255 (1 oct) or 0 to 65535 (2 oct)	This parameter sets the common ASDU address to association with this sector of the specified session. The range of address for this parameter are dependent on the length value set in the session section.

[Section]/Item	Value	Range	Description
Use Time Tag Commands:		Yes or No	This parameter specifies if a time tag field is to be included with commands. This is as specified in the IEC-870-5-104 specification and should only be utilized if the controlled device supports these new data types. If the parameter is set to Yes, a time tag will be added to all commands. If the parameter is set to No, the normal IEC 60870-5-101 data type messages will be utilized.
Online Time Sync:		Yes or No	This parameter specifies if the sector in the controlled device will be sent a time synchronization command when the unit is first recognized as being online. This should only be utilized for devices that do not send an EOI message after initializing.
Online General Int:		Yes or No	This parameter specifies if the sector in the controlled device will be sent a general interrogation command when the unit is first recognized as being online. This should only be utilized for devices that do not send an EOI message after initializing.
EOI Time Sync:		Yes or No	This parameter specifies if the sector in the controlled device will be sent a time synchronization command after this module received an EOI message from the controlled unit.
EOI General Int:		Yes or No	This parameter specifies if the sector in the controlled device will be sent a general interrogation command after this module received an EOI message from the controlled unit.
# Data Type Point # DB Address			
# --------- ---------- ----------			
START			
END			

[Section]/Item	[redacted]	Description
[IEC-101 MASTER COMMANDS]	[redacted]	This section contains the commands for the module
# Enable Database Poll Session Sector Data Point Qualifier		
# Code Index Interval Index Index Type Index Parameter		
# ------ -------- -------- ------- ------ ---- ----- ---------		
START		
END		

7.7 Database Form

7.7.1 Form to Define Sector Database

Session Index #:

Sector Index #:

Data Type	**Point Index**	**Database Address**

7.8 Command List Form

7.8.1 Form to Define Command List

Enable Code	Database Index	Poll Interval	Session Index	Sector Index	Data Type	Point Index	Qualifier Parameter

7.9 Interoperability

This companion standard presents sets of parameters and alternatives from which subsets have to be selected to implement particular telecontrol systems. Certain parameter values, such as the number of octets in the COMMON ADDRESS of ASDUs represent mutually exclusive alternatives. This means that only one value of the defined parameters is admitted per system. Other parameters, such as the listed set of different process information in command and in monitor direction allow the specification of the complete set or subsets, as appropriate for given applications. This clause summarizes the parameters of the previous clauses to facilitate a suitable selection for a specific application. If a system is composed of equipment stemming from different manufacturers it is necessary that all partners agree on the selected parameters.

The selected parameters should be crossed in the white boxes (replace "☐" with "☒").

NOTE In addition, the full specification of a system may require individual selection of certain parameters for certain parts of the system, such as the individual selection of scaling factors for individually addressable measured values.

Network configuration

(network-specific parameter)

☒ Point-to-point ☒ Multipoint-party line

☒ Multiple point-to-point ☒ Multipoint-star

Physical Layer

(network-specific parameter)

Transmission speed (control direction)

Unbalanced interchange Unbalanced interchange Balanced interchange

circuit V.24/V.28 circuit V.24/V.28 circuit X.24/X.27

Standard Recommended if >1 200 bit/s

☐ 100 bit/s ☒ 2400 bit/s ☒ 2400 bit/s ☐ 56000 bit/s

☐ 200 bit/s ☒ 4800 bit/s ☒ 4800 bit/s ☐ 64000 bit/s

☒ 300 bit/s ☒ 9600 bit/s ☒ 9600 bit/s

☒ 600 bit/s ☒ 19200 bit/s

☒ 1200 bit/s ☒ 38400 bit/s

Transmission speed (monitor direction)

Unbalanced interchange Unbalanced interchange Balanced interchange

circuit V.24/V.28 circuit V.24/V.28 circuit X.24/X.27

Standard Recommended if >1 200 bit/s

☐ 100 bit/s ☒ 2400 bit/s ☒ 2400 bit/s ☐
56000 bit/s

☐ 200 bit/s ☒ 4800 bit/s ☒ 4800 bit/s ☐
64000 bit/s

☒ 300 bit/s ☒ 9600 bit/s ☒ 9600 bit/s

☒ 600 bit/s ☒ 19200 bit/s

☒ 1200 bit/s ☒ 38400 bit/s

Link Layer

(network-specific parameter)

Frame format FT 1.2, single character 1 and the fixed time out interval are used exclusively in this companion standard.

Link transmission procedure **Address field of link**

☒ Balanced transmission ☒ Not present (balanced transmission only)

☒ Unbalanced transmission ☒ One octet

☒ Two octets

☒ Structured

Frame length ☒ Unstructured

255 Maximum length L (number of octets)

Application Layer

Transmission mode for application data

Mode 1 (Least significant octet first), as defined in clause 4.10 of IEC 60870-5-4, is used exclusively in this companion standard.

Common address of ASDU

(system-specific parameter)

☒ One octet ☒ Two octets

Information object address

(system-specific parameter)

☒ One octet ☒ structured

☒ Two octets ☒ unstructured

☒ Three octets

Cause of transmission

(system-specific parameter)

☒ One octet ☒ Two octets (with originator address)

Selection of standard ASDUs

Process information in monitor direction

(station-specific parameter)

☒ <1> := Single-point information M_SP_NA_1

☒ <2> := Single-point information with time tag M_SP_TA_1

☒ <3> := Double-point information M_DP_NA_1

☒ <4> := Double-point information with time tag M_DP_TA_1

☒ <5> := Step position information M_ST_NA_1

☒ <6> := Step position information with time tag M_ST_TA_1

☒ <7> := Bitstring of 32 bit M_BO_NA_1

☒ <8> := Bitstring of 32 bit with time tag M_BO_TA_1

☒ <9> := Measured value, normalized value M_ME_NA_1

☒ <10> := Measured value, normalized value with time tag M_ME_TA_1

☒ <11> := Measured value, scaled value M_ME_NB_1

☒ <12> := Measured value, scaled value with time tag M_ME_TB_1

☒ <13> := Measured value, short floating point value M_ME_NC_1

☒ <14> := Measured value, short floating point value with time tag M_ME_TC_1

☒ <15> := Integrated totals M_IT_NA_1

☒ <16> := Integrated totals with time tag M_IT_TA_1

☐ <17> := Event of protection equipment with time tag M_EP_TA_1

☐ <18> := Packed start events of protection equipment with time tag M_EP_TB_1

☐ <19> := Packed output circuit information of protection equipment with time tag M_EP_TC_1

☐ <20> := Packed single-point information with status change detection M_PS_NA_1

☐ <21> := Measured value, normalized value without quality descriptor M_ME_ND_1

☒ <30> := Single-point information with time tag CP56Time2a M_SP_TB_1

☒ <31> := Double-point information with time tag CP56Time2A M_DP_TB_1

☒ <32> := Step position information with time tag CP56Time2A M_ST_TB_1

☒ <33> := Bitstring of 32 bit with time tag CP56Time2A M_BO_TB_1

☒ <34> := Measured value, normalized value with time tag CP56Time2A M_ME_TD_1

☒ <35> := Measured value, scaled value with time tag CP56Time2A M_ME_TE_1

☒ <36> := Measured value, short floating point value with time tag CP56Time2A M_ME_TF_1

☒ <37> := Integrated totals with time tag CP56Time2A M_IT_TB_1

☐ <38> := Event of protection equipment with time tag CP56Time2A M_EP_TD_1

☐ <39> := Packed start events of protection equipment with time tag CP56time2A M_EP_TE_1

☐ <40> := Packed output circuit information of protection equipment with time tag CPT56Time2a M_EP_TF_1

Process information in control direction

(station-specific parameter)

☒ <45> := Single command C_SC_NA_1

☒ <46> := Double command C_DC_NA_1

☒ <47> := Regulating step command C_RC_NA_1

☒ <48> := Set point command, normalized value C_SE_NA_1

☒ <49> := Set point command, scaled value C_SE_NB_1

☒ <50> := Set point command, short floating point value C_SE_NC_1

☒ <51> := Bitstring of 32 bit C_BO_NA_1

System information in monitor direction

(station-specific parameter)

☒ <70> := End of initialization M_EI_NA_1

System information in control direction

(station-specific parameter)

☒ <100> := Interrogation command C_IC_NA_1

☒ <101> := Counter interrogation command C_CI_NA_1

☒ <102> := Read command C_RD_NA_1

☒ <103> := Clock synchronization command C_CS_NA_1

☒ <104> := Test command C_TS_NB_1

☒ <105> := Reset process command C_RP_NC_1

☒ <106> := Delay acquisition command C_CD_NA_1

Parameter in control direction

(station-specific parameter)

☒ <110> := Parameter of measured value, normalized value P_ME_NA_1

☒ <111> := Parameter of measured value, scaled value P_ME_NB_1

☒⑨<112> := Parameter of measured value, short floating point value P_ME_NC_1

☒ <113> := Parameter activation P_AC_NA_1

File transfer

(station-specific parameter)

☐ <120> := File ready F_FR_NA_1

☐ <121> := Section ready F_SR_NA_1

☐ <122> := Call directory, select file, call file, call section F_SC_NA_1

☐ <123> := Last section, last segment F_LS_NA_1

☐ <124> := Ack file, ack section F_AF_NA_1

☐ <125> := Segment F_SG_NA_1

☐ <126> := Directory F_DR_TA_1

Basic Application Functions

Station initialization

(station-specific parameter)

☒ Remote initialization

General Interrogation

(system- or station-specific parameter)

☒ global

☒	group 1	☒	group 7	☒	group 13
☒	group 2	☒	group 8	☒	group 14
☒	group 3	☒	group 9	☒	group 15
☒	group 4	☒	group 10	☒	group 16
☒	group 5	☒	group 11		
☒	group 6	☒	group 12		

Addresses per group have to be defined

Clock synchronization

(station-specific parameter)

☒ Clock synchronization

Command transmission

(object-specific parameter)

☒ Direct command transmission ☒ Select and execute command

☒ Direct set point command transmission ☒ Select/execute set point cmd

☒ C_SE_ACTTERM used

☒ No additional definition

☒ Short pulse duration (duration determined by a system parameter in the outstation)

☒ Long pulse duration (duration determined by a system parameter in the outstation)

☒ Persistent output

Transmission of Integrated totals

(station- or object-specific parameter)

☒ Counter request ☒ General request counter

☒ Counter freeze without reset ☒ Request counter group 1

☒ Counter freeze with reset ☒ Request counter group 2

☒ Counter reset ☒ Request counter group 3

☒ Request counter group 4

Addresses per group have to be defined

Parameter loading

(object-specific parameter)

☒ Threshold value

☒ Smoothing factor

☒ Low limit for transmission of measured value

☒ High limit for transmission of measured value

Parameter activation

(object-specific parameter)

☒ Act/deact of persistent cyclic or periodic transmission of the addressed object

File transfer

(station-specific parameter)

☐ File transfer in monitor direction

☐ File transfer in control direction

8 Support, Service & Warranty

In This Chapter

Be sure and read the full Warranty that can be found on our web site at www.prosoft-technology.com for details and other terms and conditions. The content in this summary is subject to change without notice. The content is current at date of publication.

ProSoft Technology, Inc. strives to provide meaningful support to its customers. Should any questions or problems arise, please feel free to contact us at:

Internet	Web Site: http://www.prosoft-technology.com/support
	E-mail address: support@prosoft-technology.com

Those of us at ProSoft Technology, Inc. want to provide the best and quickest support possible, so before calling please have the following information available. You may wish to fax this information to us prior to calling.

1. Product Version Number
2. System architecture
3. Network details

In the case of hardware, we will also need the following information:

1. Module configuration and contents of file
2. Module Operation
3. Configuration/Debug status information
4. LED patterns
5. Information about the processor and user data files as viewed through the development software and LED patterns on the processor
6. Details about the networked devices interfaced, if any

For technical support calls within the United States, an after-hours answering system allows pager access to one of our qualified technical and/or application support engineers at any time to answer your questions.

8.1 How to Contact Us: Sales and Support

All ProSoft Technology Products are backed with full technical support. Contact our worldwide Technical Support team and Customer Service representatives directly by phone or email:

USA / Latin America (excluding Brasil) (Office in California)

+1(661) 716-5100
+1(661) 716-5101 (Fax)
1675 Chester Avenue, 4th Floor
Bakersfield, California 93301
U.S.A.
+1.661.716.5100, support@prosoft-technology.com
Languages spoken include: English, Spanish

Asia Pacific Sales (office in Malaysia)

+603.7724.2080
+603.7724.2090 (Fax)
C210, Damansara Intan,
1 Jalan SS20/27, 47400 Petaling Jaya
Selangor, Malaysia
+603.7724.2080, asiapc@prosoft-technology.com
Languages spoken include: Chinese, Japanese, English

Asia Pacific Support (office in China)

+86.21.64518356 x 8011
+86.21.64756957 (Fax)
4/F, No. 16 Hongcao Road
Shanghai, China 200233
China
+86.21.64518356 x 8011, zhang@prosoft-technology.com
Languages spoken include: Chinese, English

Europe / Middle East / Africa (office in Toulouse, France)

+33 (0) 5.34.36.87.20
+33 (0) 5.61.78.40.52 (Fax)
Zone d'activité de Font Grasse
17, rue des Briquetiers
F-31700 Blagnac
France
+33 (0) 5.34.36.87.20. support.emea@prosoft-technology.com
Languages spoken include: French, English

Brasil (office in Sao Paulo)

+55-11-5084-5178
+55-11-5083-3776 (Fax)
Rua Vergueiro, 2949 - sala 182 - Edifício Vergueiro Work Center
Vila Mariana - São Paulo
Cep: 04101-300 - Brasil
+55-11-5084-5178, eduardo@prosoft-technology.com
Languages spoken include: Portuguese, English

8.2 Return Material Authorization (RMA) Policies and Conditions

The following RMA Policies and Conditions apply to any returned product. These RMA Policies are subject to change by ProSoft without notice. For warranty information, see Section C below entitled "Limited Warranty". In the event of any inconsistency between the RMA Policies and the Warranty, the Warranty shall govern.

8.2.1 All Product Returns

1 In order to return a Product for repair, exchange or otherwise, the Customer must obtain a Returned Material Authorization (RMA) number from ProSoft and comply with ProSoft shipping instructions.
2 In the event that the Customer experiences a problem with the Product for any reason, Customer should contact ProSoft Technical Support at one of the telephone numbers listed above in Section A. A Technical Support Engineer will request several tests in an attempt to isolate the problem. If after these tests are completed, the Product is found to be the source of the problem, ProSoft will issue an RMA.
3 All returned Products must be shipped freight prepaid, in the original shipping container or equivalent, to the location specified by ProSoft, and be accompanied by proof of purchase. The RMA number is to be prominently marked on the outside of the shipping box. Customer agrees to insure the Product or assume the risk of loss or damage in transit. Products shipped to ProSoft without an RMA number will be returned to the Customer, freight collect. Contact ProSoft Technical Support for further information.
4 Out of warranty returns are not allowed on RadioLinx accessories such as antennas, cables, and brackets.

The following policy applies for Non-Warranty Credit Returns:

A 10% Restocking Fee if Factory Seal is *not* broken
B 20% Restocking Fee if Factory Seal is broken

ProSoft retains the right, in its absolute and sole discretion, to reject any non-warranty returns for credit if the return is not requested within three (3) months after shipment of the Product to Customer, if the Customer fails to comply with ProSoft's shipping instructions, or if the Customer fails to return the Product to ProSoft within six (6) months after Product was originally shipped.

8.3 Procedures for Return of Units Under Warranty

1 A Technical Support Engineer must pre-approve all product returns.
2 Module is repaired or replaced after a Return Material Authorization Number is entered and a replacement order is generated.
3 Credit for the warranted item is issued within 10 business days after receipt of product and evaluation of the defect has been performed by ProSoft. The credit will only be issued provided the product is returned with a valid Return Material Authorization Number and in accordance with ProSoft's shipping instructions.

a) If no defect is found, a credit is issued.

b) If a defect is found and is determined to be customer generated or if the defect is otherwise not covered by ProSoft's Warranty, or if the module is not repairable, a credit is not issued and payment of the replacement module is due.

8.4 Procedures for Return of Units Out of Warranty

1 Customer sends unit in for evaluation.

2 If no defect is found, Customer will be charged the equivalent of US $100 plus shipping, duties and taxes that may apply. A new Purchase Order will be required for this evaluation fee.

If the unit is repaired the charge to the Customer will be 30%* of the list price plus any shipping, duties and taxes that may apply. A new Purchase Order will be required for a product repair.

3 For an immediate exchange, a new module may be purchased and sent to Customer while repair work is being performed. Credit for purchase of the new module will be issued when the new module is returned in accordance with ProSoft's shipping instructions and subject to ProSoft's policy on non-warranty returns. This is in addition to charges for repair of the old module and any associated charges to Customer.

4 If, upon contacting ProSoft Customer Service, the Customer is informed that unit is believed to be unrepairable, the Customer may choose to send unit in for evaluation to determine if the repair can be made. Customer will pay shipping, duties and taxes that may apply. If unit cannot be repaired, the Customer may purchase a new unit.

8.4.1 Un-repairable Units

- 3150-All
- 3750
- 3600-All
- 3700
- 3170-All
- 3250
- 1560 can be repaired, if defect is the power supply
- 1550 can be repaired, if defect is the power supply
- 3350
- 3300
- 1500-All

*** 30% of list price is an estimated repair cost only. The actual cost of repairs will be determined when the module is received by ProSoft and evaluated for needed repairs.**

8.4.2 Purchasing Warranty Extension

As detailed below in ProSoft's Warranty, the standard Warranty Period is one year (or in the case of RadioLinx modules, three years) from the date of delivery. The Warranty Period may be extended for an additional charge, as follows:

- Additional 1 year = 10% of list price
- Additional 2 years = 20% of list price
- Additional 3 years = 30% of list price

8.5 LIMITED WARRANTY

This Limited Warranty ("Warranty") governs all sales of hardware, software and other products (collectively, "Product") manufactured and/or offered for sale by ProSoft, and all related services provided by ProSoft, including maintenance, repair, warranty exchange, and service programs (collectively, "Services"). By purchasing or using the Product or Services, the individual or entity purchasing or using the Product or Services ("Customer") agrees to all of the terms and provisions (collectively, the "Terms") of this Limited Warranty. All sales of software or other intellectual property are, in addition, subject to any license agreement accompanying such software or other intellectual property.

8.5.1 What Is Covered By This Warranty

A *Warranty On New Products*: ProSoft warrants, to the original purchaser only, that the Product that is the subject of the sale will (1) conform to and perform in accordance with published specifications prepared, approved, and issued by ProSoft, and (2) will be free from defects in material or workmanship; provided these warranties only cover Product that is sold as new. This Warranty expires one year (or in the case of RadioLinx modules, three years) from the date of shipment (the "Warranty Period"). If the Customer discovers within the Warranty Period a failure of the Product to conform to specifications, or a defect in material or workmanship of the Product, the Customer must promptly notify ProSoft by fax, email or telephone. In no event may that notification be received by ProSoft later than 15 months (or in the case of RadioLinx modules, 39 months) from the date of delivery. Within a reasonable time after notification, ProSoft will correct any failure of the Product to conform to specifications or any defect in material or workmanship of the Product, with either new or used replacement parts. Such repair, including both parts and labor, will be performed at ProSoft's expense. All warranty service will be performed at service centers designated by ProSoft. If ProSoft is unable to repair the Product to conform to this Warranty after a reasonable number of attempts, ProSoft will provide, at its option, one of the following: a replacement product, a full refund of the purchase price or a credit in the amount of the purchase price. All replaced product and parts become the property of ProSoft. These remedies are the Customer's only remedies for breach of warranty.

B *Warranty On Services*: Material and labor used by ProSoft to repair a verified malfunction or defect are warranted on the terms specified above for new Product, provided said warranty will be for the period remaining on the original new equipment warranty or, if the original warranty is no longer in effect, for a period of 90 days from the date of repair.

C The Warranty Period for RadioLinx accessories (such as antennas, cables, brackets, etc.) are the same as for RadioLinx modules, that is, three years from the date of shipment.

8.5.2 What Is Not Covered By This Warranty

A ProSoft makes no representation or warranty, expressed or implied, that the operation of software purchased from ProSoft will be uninterrupted or error free or that the functions contained in the software will meet or satisfy the purchaser's intended use or requirements; the Customer assumes complete responsibility for decisions made or actions taken based on information obtained using ProSoft software.

B With the exception of RadioLinx accessories referenced in paragraph 1(c) this Warranty does not cover any product, components, or parts not manufactured by ProSoft.

C This Warranty also does not cover the failure of the Product to perform specified functions, or any other non-conformance, defects, losses or damages caused by or attributable to any of the following: (i) shipping; (ii) improper installation or other failure of Customer to adhere to ProSoft's specifications or instructions; (iii) unauthorized repair or maintenance; (iv) attachments, equipment, options, parts, software, or user-created programming (including, but not limited to, programs developed with any IEC 61131-3 programming languages, or "C") not furnished by ProSoft; (v) use of the Product for purposes other than those for which it was designed; (vi) any other abuse, misapplication, neglect or misuse by the Customer; (vii) accident, improper testing or causes external to the Product such as, but not limited to, exposure to extremes of temperature or humidity, power failure or power surges outside of the limits indicated on the product specifications; or (viii) disasters such as fire, flood, earthquake, wind or lightning.

D The information in this Agreement is subject to change without notice. ProSoft shall not be liable for technical or editorial errors or omissions made herein; nor for incidental or consequential damages resulting from the furnishing, performance or use of this material. The user guides included with your original product purchased by you from ProSoft, contains information protected by copyright. No part of the guide may be duplicated or reproduced in any form without prior written consent from ProSoft.

8.5.3 DISCLAIMER REGARDING HIGH RISK ACTIVITIES

PRODUCT MANUFACTURED OR SUPPLIED BY PROSOFT IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE (INCLUDING, WITHOUT LIMITATION, THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION OF COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL, DIRECT LIFE SUPPORT MACHINES OR WEAPONS SYSTEMS), IN WHICH THE FAILURE OF THE PRODUCT COULD LEAD DIRECTLY OR INDIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE (COLLECTIVELY, "HIGH RISK ACTIVITIES"). PROSOFT SPECIFICALLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR HIGH RISK ACTIVITIES.

8.5.4 DISCLAIMER OF ALL OTHER WARRANTIES

THE WARRANTIES SET FORTH IN PARAGRAPH 1 ABOVE ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

*8.5.5 LIMITATION OF REMEDIES***

IN NO EVENT WILL PROSOFT (OR ITS DEALER) BE LIABLE FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES BASED ON BREACH OF WARRANTY, BREACH OF CONTRACT, NEGLIGENCE, STRICT TORT, OR ANY OTHER LEGAL THEORY. DAMAGES THAT PROSOFT AND ITS DEALER WILL NOT BE RESPONSIBLE FOR INCLUDE, BUT ARE NOT LIMITED TO: LOSS OF PROFITS; LOSS OF SAVINGS OR REVENUE; LOSS OF USE OF THE PRODUCT OR ANY ASSOCIATED EQUIPMENT; LOSS OF DATA; COST OF CAPITAL; COST OF ANY SUBSTITUTE EQUIPMENT, FACILITIES, OR SERVICES; DOWNTIME; THE CLAIMS OF THIRD PARTIES, INCLUDING CUSTOMERS OF THE PURCHASER; AND INJURY TO PROPERTY.

** Some areas do not allow time limitations on an implied warranty, or allow the exclusion or limitation of incidental or consequential damages. In such areas the above limitations may not apply. This Warranty gives you specific legal rights, and you may also have other rights which vary from place to place.

8.5.6 Time Limit for Bringing Suit

Any action for breach of warranty must be commenced within 15 months (or in the case of RadioLinx modules, 39 months) following shipment of the Product.

8.5.7 No Other Warranties

Unless modified in writing and signed by both parties, this Warranty is understood to be the complete and exclusive agreement between the parties, suspending all oral or written prior agreements and all other communications between the parties relating to the subject matter of this Warranty, including statements made by salesperson. No employee of ProSoft or any other party is authorized to make any warranty in addition to those made in this Warranty. The Customer is warned, therefore, to check this Warranty carefully to see that it correctly reflects those terms that are important to the Customer.

8.5.8 Intellectual Property

A Any documentation included with Product purchased from ProSoft is protected by copyright and may not be photocopied or reproduced in any form without prior written consent from ProSoft.
B ProSoft's technical specifications and documentation that are included with the Product are subject to editing and modification without notice.
C Transfer of title shall not operate to convey to Customer any right to make, or have made, any Product supplied by ProSoft.
D Customer is granted no right or license to use any software or other intellectual property in any manner or for any purpose not expressly permitted by any license agreement accompanying such software or other intellectual property.
E Customer agrees that it shall not, and shall not authorize others to, copy software provided by ProSoft (except as expressly permitted in any license agreement accompanying such software); transfer software to a third party separately from the Product; modify, alter, translate, decode, decompile, disassemble, reverse-engineer or otherwise attempt to derive the source code of the software or create derivative works based on the software; export the software or underlying technology in contravention of applicable US and international export laws and regulations; or use the software other than as authorized in connection with use of Product.

8.5.9 Additional Restrictions Relating To Software And Other Intellectual Property

In addition to complying with the Terms of this Warranty, Customers purchasing software or other intellectual property shall comply with any license agreement accompanying such software or other intellectual property. Failure to do so may void this Warranty with respect to such software and/or other intellectual property.

8.5.10 Allocation of risks

This Warranty allocates the risk of product failure between ProSoft and the Customer. This allocation is recognized by both parties and is reflected in the price of the goods. The Customer acknowledges that it has read this Warranty, understands it, and is bound by its Terms.

8.5.11 Controlling Law and Severability

This Warranty shall be governed by and construed in accordance with the laws of the United States and the domestic laws of the State of California, without reference to its conflicts of law provisions. If for any reason a court of competent jurisdiction finds any provisions of this Warranty, or a portion thereof, to be unenforceable, that provision shall be enforced to the maximum extent permissible and the remainder of this Warranty shall remain in full force and effect. Any cause of action with respect to the Product or Services must be instituted in a court of competent jurisdiction in the State of California.

Index