

Quantum Automation Series

140 ESI 062 10

ASCII Interface Module

User Guide

840 USE 108 00 Version 2.0



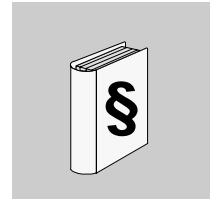
Table of Contents



	Safety Information	5
	About the Book	7
Chapter 1	Configuring the ESI Module	9
	Overview	9
	Topology Overview and Parts List	10
	Running an ESI Application	12
Chapter 2	Introduction to ESI 062 10	19
	At a Glance	19
	Introduction to ESI Module	20
	Application Criteria	22
	Module Description	23
	ESI Module Block Diagramm	25
Chapter 3	Hardware Overview and System Specifications	27
	At a Glance	27
	140 ESI 062 10 ASCII Interface Module	28
	Module Specifications	31
	Quantum Automation Series System Specifications	32
Chapter 4	140 ESI 062 10 Hardware Description	35
	At a Glance	35
	Presentation	36
	Indicators	37
	External Connectors and Switches	39
	Specifications	41
Chapter 5	Configuration Overview	45
	At a Glance	45
	ESI Configuration	46
	Data Flow	49
Chapter 6	ESI Command Line Editors	53
	Overview	53

	Configuration Editor	54
	ASCII Message Editor	58
	ASCII Message Formats	59
Chapter 7	ESI Commands.	65
	At a Glance	65
	Overview on ESI Commands.	66
	ESI Command Structure	67
	Command 0 - NO OPERATION.	68
	Command 1- READ ASCII MESSAGE	69
	Command 2 - WRITE ASCII MESSAGE	71
	Command 3 - GET DATA (Module to Controller)	74
	Command 4 - PUT DATA (Controller to Module)	76
	Command 5 - GET TOD (Time of Day)	78
	Command 6 - SET TOD (Time of Day)	80
	Command 7 - SET MEMORY REGISTERS	83
	Command 8 - FLUSH BUFFER.	85
	Command 9 - ABORT	86
	Command A - GET BUFFER STATUS	87
	Response Structure for Illegal Commands	89
	Module Status Word (Word 11)	90
	Reading beyond valid Register Range	92
Chapter 8	ESI Loadable	95
	Overview	95
	Short Description	96
	Representation.	97
	Parameter Description	98
	Run Time Errors.	101
	READ ASCII Message (Subfunction 1)	102
	WRITE ASCII Message (Subfunction 2)	106
	GET DATA (Subfunction 3)	107
	PUT DATA (Subfunction 4)	109
Appendices	113
	At a Glance	113
Appendix A	Character Set	115
	ASCII Character Set	115
Index	119

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.



DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death, serious injury, or equipment damage.



WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.



CAUTION

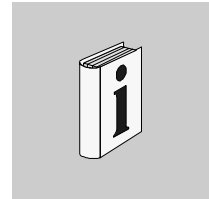
CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE

Electrical equipment should be serviced only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material. This document is not intended as an instruction manual for untrained persons.

© 2002 Schneider Electric All Rights Reserved

About the Book



At a Glance

Document Scope This book provides an overview and specifications of the Quantum ASCII interface module, which provides the capability to communicate and exchange data with third party devices that may not use standard ASCII communications.

Validity Note The data and illustrations found in this book are not binding. We reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be construed as a commitment by Schneider Electric.

Related Documents

Title of Documentation	Reference Number
Quantum Automation Series Hardware Reference Guide	840 USE 100 00
Modicon Modsoft Programmers User Manual	GM-MSFT-001
Modicon Ladder Logic Block Library	840 USE 101 00
Concept User Manual	372 SPU 440 01

Product Related Warnings Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.
No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

User Comments

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

Configuring the ESI Module



Overview

Purpose

This chapter is a quickstart example. It provides a step by step procedure for setting up the Quantum ASCII module.

What's in this Chapter?

This chapter contains the following topics:

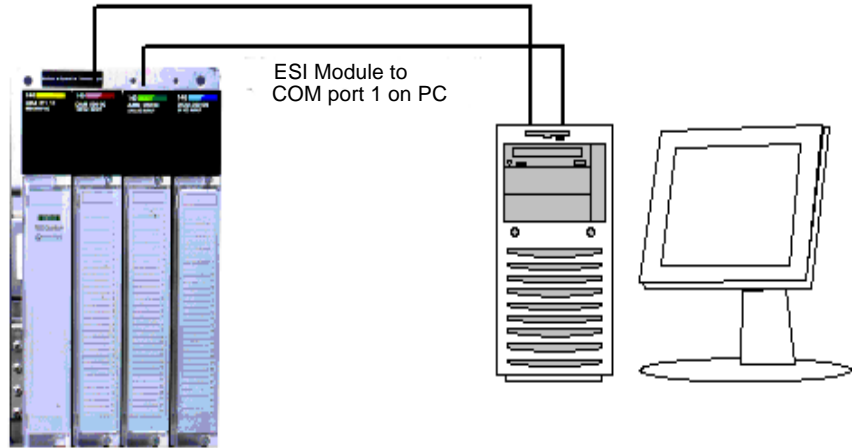
Topic	Page
Topology Overview and Parts List	10
Running an ESI Application	12

Topology Overview and Parts List

Topology Overview Diagram

The following diagram is a basic layout of the application.

Quantum Ethernet module 170NOE7710 to Ethernet port on PC



Parts List

Parts list table

Items	Parts	Description
Quantum PLC	Power Supply Processor (ESI supported by all Quantum processors) 140 ESI 062 10 (firmware ver 1.03) 140 NOE 771 10 (Ethernet module)	PLC hardware
PLC Software	ProWorx NxT Version 2.2 Concept V2.1 and higher	PLC programming software
Modbus Cable		Communication cable between ESI module and PC
Hyper terminal	Terminal emulation that comes with Windows	For capturing the ASCII port input/output
User Manual	840 USE 108 00	140 ESI 062 10 ASCII Interface User Guide

Setting Up the ESI Module

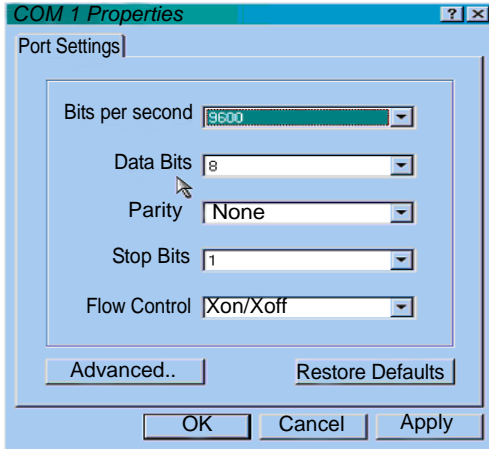
Perform the following steps when setting up the ESI module using Concept.

Step	Action	Information
1	Launch Concept and start a new message	
2	<p>Select the PLC type and configure the IO Map</p> <ul style="list-style-type: none"> When configuring the ESI module, click on the "Parameter" radio button on the top right hand corner and configure the ASCII and Programming port configuration. 	<ul style="list-style-type: none"> In this application the ESI input & output references are mapped to 300001 & 400001 The ASCII port & programming port parameters are set to <ul style="list-style-type: none"> Baud rate = 9600 Data bits = 8 Parity = None Stop bits = 1 Keyboard = off Xon/Xoff = enable
3	<p>Go to Loadables and unpack the loadables from the disk that comes with the user manual. The loadables you need to unpack are:</p> <ul style="list-style-type: none"> nsup.exe esi.exe <p>Then you must load the loadable beginning with:</p> <ul style="list-style-type: none"> NSUP ESI 	Always load the NSUP first, followed by the ESI.
4	<p>Once you have completed the above step, then you can download the program to the processor. Note that in this application we used the 170NOE771 110 ethernet module to communicate with the PC using Concept. We simulated the program from Concept and captured the ASCII port response from Hyper terminal using a Modbus cable to COM port 1 and the PC.</p>	You can use Modbus Plus to communicate with the processor using the SA85 card for PC or PCMIA for laptop users.

Running an ESI Application

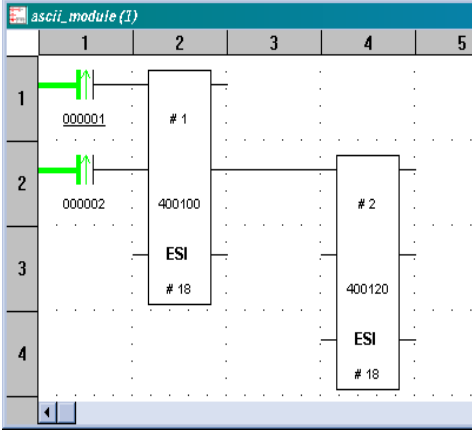
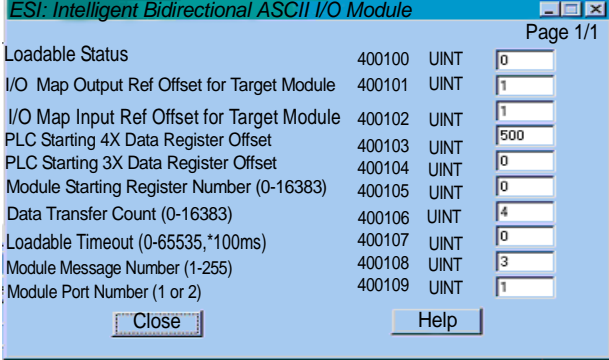
Application Example

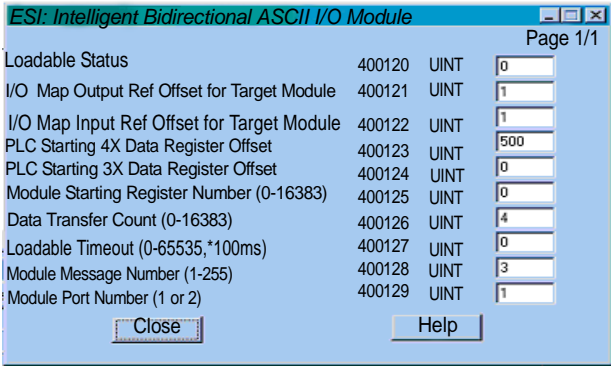
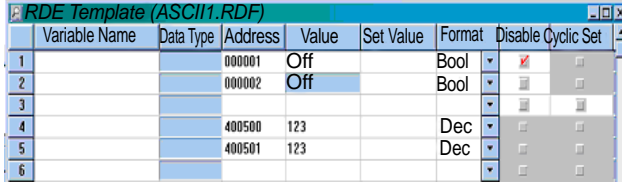
Perform the following steps when using Hyper terminal to capture or display an ESI port response.

Step	Action	Remarks
1	Launch Hyper terminal on the PC.	
2	Configure the COM 1 port of the PC to exactly the same as what's configured for the programming port. 	The flow control must always be set to Xon/Xoff to configure the ESI module. NOTE: The programming port of the ESI module is always Port 1.
3	Once connected with Hyper terminal, press and hold onto the yellow 'RESET' button on the ESI module (located between Port 1 and Port 2) until the display on the ESI module shows an orange light indicating "Status".	

Step	Action	Remarks
4	<p>The Hyper terminal should display as follows:</p> <pre data-bbox="441 235 839 438"> Welcome MODICON QUANTUM ASCII Module Entering Program Mode... Current date is... Current time is... CLI> </pre>	<p>Note: If you do not get this message displaying on the Hyper terminal:</p> <ul style="list-style-type: none"> ● Check the ESI module programming port parameters. ● Check the Hyper terminal COM1 port configurations. ● Check the cables (Should be using Modbus cables).
5	<p>Type 'ascii' at the command prompt. It should now change to the ASCII command prompt ("ASCII").</p>	<p>Refer to the ESI user manual for the different ASCII editor commands</p>
6	<p>You can enter ASCII text messages that you want to display here. In this application we use the following messages.</p> <pre data-bbox="441 1149 898 1352"> Message 1 '123456789' Message 2 <0>,' Please enter your values here', 3x,1L4,/ Message 3 <0>,'These are the values entered',3x,1L4,/ </pre>	<p>You can create 256 ASCII messages per port for this module</p>

Step	Action	Remarks
7	<p data-bbox="467 199 810 224">Example of entering the messages:</p> <div data-bbox="467 233 1037 440" style="border: 1px solid black; padding: 5px;"> <pre data-bbox="485 240 1019 431"> ASCII> new Enter message: ><0>, 'Please enter your values here' , 3x, 1l4, 3X, 1L4, / ASCII > save Enter message number (1..255) >2 Message saved.... ACSII></pre> </div> <ul data-bbox="467 448 1085 646" style="list-style-type: none"> ● <0> is to flush the buffer (clear the buffer in the ASCII port each time the message is executed) ● Followed by a text message ● 3X,1l4,3X,1L4 is the message format <ul style="list-style-type: none"> ● X is the space ● l is the integer format with leading spaces ● L is the integer format with leading zeros 	<p data-bbox="1101 199 1245 399">Refer to the ESI user manual for the different types of ASCII message formats.</p>
8	<p data-bbox="467 662 1078 711">Once you finished with keying the messages just type 'run' at the command prompt.</p>	

Step	Action	Remarks
9	<p data-bbox="444 201 976 224">Example of the application program on Concept 984 LL:</p>  <p data-bbox="444 678 1059 760">In this application program we are trying to read from the Hyper terminal and then output/display the values read from it. The read ESI function block is configured as follows.</p> 	<p data-bbox="1081 201 1218 863">Refer to the ESI user manual for a clear description of the ESI loadable instruction. Create a reference data editor to force the 000001 contact to simulate the ASCII messages on port 1. To do a DX zoom, highlight the function block and press CTRL-D.</p>

Step	Action	Remarks																																																																																	
10	<p>The write ESI function block is configured as follows.</p>  <p>ESI: Intelligent Bidirectional ASCII I/O Module Page 1/1</p> <table border="1"> <tr><td>Loadable Status</td><td>400120</td><td>UINT</td><td>0</td></tr> <tr><td>I/O Map Output Ref Offset for Target Module</td><td>400121</td><td>UINT</td><td>1</td></tr> <tr><td>I/O Map Input Ref Offset for Target Module</td><td>400122</td><td>UINT</td><td>1</td></tr> <tr><td>PLC Starting 4X Data Register Offset</td><td>400123</td><td>UINT</td><td>500</td></tr> <tr><td>PLC Starting 3X Data Register Offset</td><td>400124</td><td>UINT</td><td>0</td></tr> <tr><td>Module Starting Register Number (0-16383)</td><td>400125</td><td>UINT</td><td>0</td></tr> <tr><td>Data Transfer Count (0-16383)</td><td>400126</td><td>UINT</td><td>4</td></tr> <tr><td>Loadable Timeout (0-65535,*100ms)</td><td>400127</td><td>UINT</td><td>0</td></tr> <tr><td>Module Message Number (1-255)</td><td>400128</td><td>UINT</td><td>3</td></tr> <tr><td>Module Port Number (1 or 2)</td><td>400129</td><td>UINT</td><td>1</td></tr> </table> <p>Close Help</p>	Loadable Status	400120	UINT	0	I/O Map Output Ref Offset for Target Module	400121	UINT	1	I/O Map Input Ref Offset for Target Module	400122	UINT	1	PLC Starting 4X Data Register Offset	400123	UINT	500	PLC Starting 3X Data Register Offset	400124	UINT	0	Module Starting Register Number (0-16383)	400125	UINT	0	Data Transfer Count (0-16383)	400126	UINT	4	Loadable Timeout (0-65535,*100ms)	400127	UINT	0	Module Message Number (1-255)	400128	UINT	3	Module Port Number (1 or 2)	400129	UINT	1																																										
Loadable Status	400120	UINT	0																																																																																
I/O Map Output Ref Offset for Target Module	400121	UINT	1																																																																																
I/O Map Input Ref Offset for Target Module	400122	UINT	1																																																																																
PLC Starting 4X Data Register Offset	400123	UINT	500																																																																																
PLC Starting 3X Data Register Offset	400124	UINT	0																																																																																
Module Starting Register Number (0-16383)	400125	UINT	0																																																																																
Data Transfer Count (0-16383)	400126	UINT	4																																																																																
Loadable Timeout (0-65535,*100ms)	400127	UINT	0																																																																																
Module Message Number (1-255)	400128	UINT	3																																																																																
Module Port Number (1 or 2)	400129	UINT	1																																																																																
11	<p>This is the simulation of the program and the results.</p>  <table border="1"> <thead> <tr> <th>Variable Name</th> <th>Data Type</th> <th>Address</th> <th>Value</th> <th>Set Value</th> <th>Format</th> <th>Disable</th> <th>Cyclic</th> <th>Set</th> </tr> </thead> <tbody> <tr><td>1</td><td></td><td>000001</td><td>Off</td><td></td><td>Bool</td><td><input checked="" type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>2</td><td></td><td>000002</td><td>Off</td><td></td><td>Bool</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>4</td><td></td><td>400500</td><td>123</td><td></td><td>Dec</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>5</td><td></td><td>400501</td><td>123</td><td></td><td>Dec</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> </tbody> </table> <p>From Concept when you force the "000001" contact ON over the hyper terminal. You will be prompted to enter values as specified in the ASCII message. In this example we have tried the program three times by forcing the '000001' once after every completion of the program. The contact 000002 is used to abort the function block. The hyper terminal results are as follows:</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <table border="0"> <tr><td>Please enter your values here</td><td></td><td></td></tr> <tr><td>These values are entered</td><td>12</td><td>0012</td></tr> <tr><td>Please enter your values here</td><td></td><td></td></tr> <tr><td>These values are entered</td><td>1200</td><td>1200</td></tr> <tr><td>Please enter your values here</td><td></td><td></td></tr> <tr><td>These values are entered</td><td>123</td><td>0123</td></tr> </table> </div>	Variable Name	Data Type	Address	Value	Set Value	Format	Disable	Cyclic	Set	1		000001	Off		Bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2		000002	Off		Bool	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4		400500	123		Dec	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5		400501	123		Dec	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Please enter your values here			These values are entered	12	0012	Please enter your values here			These values are entered	1200	1200	Please enter your values here			These values are entered	123	0123	<p>For the 1st run of the program, we have entered the values '0012' and '0012'.</p> <p>Fir the 2nd run of the program, we have entered the values '1200' and '1200'.</p> <p>For the 3rd run of the program, we have entered the values '0123' and '0123'.</p> <p>Take note of the results based on the format of the messages.</p>
Variable Name	Data Type	Address	Value	Set Value	Format	Disable	Cyclic	Set																																																																											
1		000001	Off		Bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																											
2		000002	Off		Bool	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																											
3						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																											
4		400500	123		Dec	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																											
5		400501	123		Dec	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																											
6						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																											
Please enter your values here																																																																																			
These values are entered	12	0012																																																																																	
Please enter your values here																																																																																			
These values are entered	1200	1200																																																																																	
Please enter your values here																																																																																			
These values are entered	123	0123																																																																																	

Step	Action	Remarks
12	<p>Note:-</p> <ul style="list-style-type: none">● Only one ESI function block can be executed at one time for each ESI module, regardless of the port. Therefore, if you need to run more than one ASCII command at any one time you must use different ESI modules.● Load the NSUP first, then the ESI.● Do not overlap the module's internal registers.● If you want to make use of the ESI function block, it is only available for 984LL type of programming. If you wish to use the IEC type of programming, you can make use of the registers configured in the I/O map and write your own program without help of the ESI function block.● Always ensure the ASCII ports and programming port parameters are the same as the device connected to it.	

Introduction to ESI 062 10



2

At a Glance

Introduction

This chapter provides an overview of the 140 ESI 062 10 ASCII communication modul functionality and offers help to distinguish whether the module is appropriate for a given application.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Introduction to ESI Module	20
Application Criteria	22
Module Description	23
ESI Module Block Diagramm	25

Introduction to ESI Module

Overview

The Quantum ASCII interface module is a general purpose ASCII interface module providing the ability to communicate and exchange data with third party devices. These devices, typically, are found in industrial environments that do not utilize a standard communication method familiar to industrial automation. Such standard communication methods are using the industry standard Modbus communications, which defines the data query and response strings necessary, along with the physical interface required to communicate between programmable devices. There are many communications standards and field busses available in industrial automation today. Few of these standards are based on RS 232C physical media for serial data streams. Much of the serial data information is not based on one of the available standards; therefore, the need for ASCII interfacing is required. ASCII communications are based on a custom serial protocol using RS232 or RS422/485 physical media.

Physical Media

Features of different physical media:

Standard	Maximum Distance	Physical Attributes	Data Rate Range
RS232	50 feet	Point to Point Multi drop using modems	180 bps to 19200 bps
RS422	400 feet	Point to Point Multi drop using modems	180 bps to 19200 bps
RS485	Wide Range	Multi drop (internal modems) 2 Wire or 4 Wire standards	180 bps to 19200 bps

Serial Device Applications

The majority of these ASCII applications talk directly to printers, bar code readers and scanners, serial devices such as weigh scales, meters and other measurement devices, as well as to other control systems used within the industrial automation application.

These third party devices require communications in a language they can understand in order to enable data transmission to occur between the third party device and the ASCII module.

For example, a scale measuring the total weight of a package, may respond to receiving a 'control A' ASCII character <^A> by returning the package weight. This data is placed into the memory of the ASCII module, which in turn is read by the Quantum controller. The controller may need to make a logical decision of where the package should go if the weight is above a certain pre defined amount. The ASCII module therefore allows integration of data typically found within automation applications by simply knowing the protocol or language the foreign device needs in order to communicate.

Application Criteria

Introduction

The Quantum PLC family offers various solutions for communication with external devices. Depending on the needs of the application the user may select software solutions (XMIT function block using a CPU Modbus port) or hardware solutions (ESI module or ASCII Basic module). The following information helps to find the appropriate solution for a given application.

Application Criteria

The chart below identifies typical applications and the recommended product for that solution. As always when looking at solving application problems, this information is provided as a guide only and not the only answer to application problem.

Application	Description	Recommended Solution
Printer Interface	Generate local reports with imbedded data from the controller or the ASCII module.	ESI Module, J892, or ASCII Basic Module
Communicate to simple Device	Send control characters and receive data from measurement devices.	ESI Module, J892, or XMIT
Bar Code Interface	Send and receive data from bar code reader/scanner.	ESI Module or ASCII Basic module
Communicate to Device	Send control characters and receive data from measurement devices, leading zero's or leading spaces may be sent by the device.	ESI Module or J892
Controller to Controller Interfacing	Emulate manufacturers protocol that support several sub functions.	ASCII Basic Module
External Data Storage	Store data outside of the controller.	ESI Module or ASCII Basic module
Modbus Master and/or Modem Support	Generate full spectrum of Modbus master commands and/or support dial up modems with control characters.	XMIT Function block and controllers local Modbus port
Multiple RS-232 ports	Multiple ports to communicate with external devices are required	ESI Module or ASCII Basic module
RS-232 ports in Distributed I/O	External devices have to be connected to Distributed I/O.	ESI Module or ASCII Basic module

Module Description

Overview

The ESI module consists of five major functional elements:

- Serial ports for device communication
 - Interface to the Quantum controller through the backplane
 - Port buffer
 - Register memory
 - ASCII message storage memory
 - Firmware
-

Serial Ports

The ESI module has implemented 3 logical communication ports. Port 1 and Port 2 are used to communicate to external serial devices while Port 0 is used for programming the module. Port 0 and Port 1 share one physical port. All 3 ports can be set up independently. For a detailed description of the port setup see *Port Command*, p. 55.

Interface to Quantum Controller

The ESI module exchanges data with the Quantum controller through the use of 12 output words for commands and data from the Quantum controller and 12 input words for data to the Quantum controller and command echo and status information. For detailed information about the structure of the command and response structures see *ESI Command Structure*, p. 67.

Port Buffer

The 2 physical ports of the ESI module have an input and an output buffer of 255 characters each. The device side of those buffers is maintained automatically by the optional XON/XOFF handshake. For data transfer to and from the Quantum controller and for buffer control several commands available for status testing are described in detail in this section. *Data Flow*, p. 49.

Register Memory

The ESI module has a 32 kbyte memory, which is organized as 16k 16-bit registers. These registers hold all data coming from and going to the serial ports. They can be accessed by the PUT and the GET command.

ASCII Message Storage

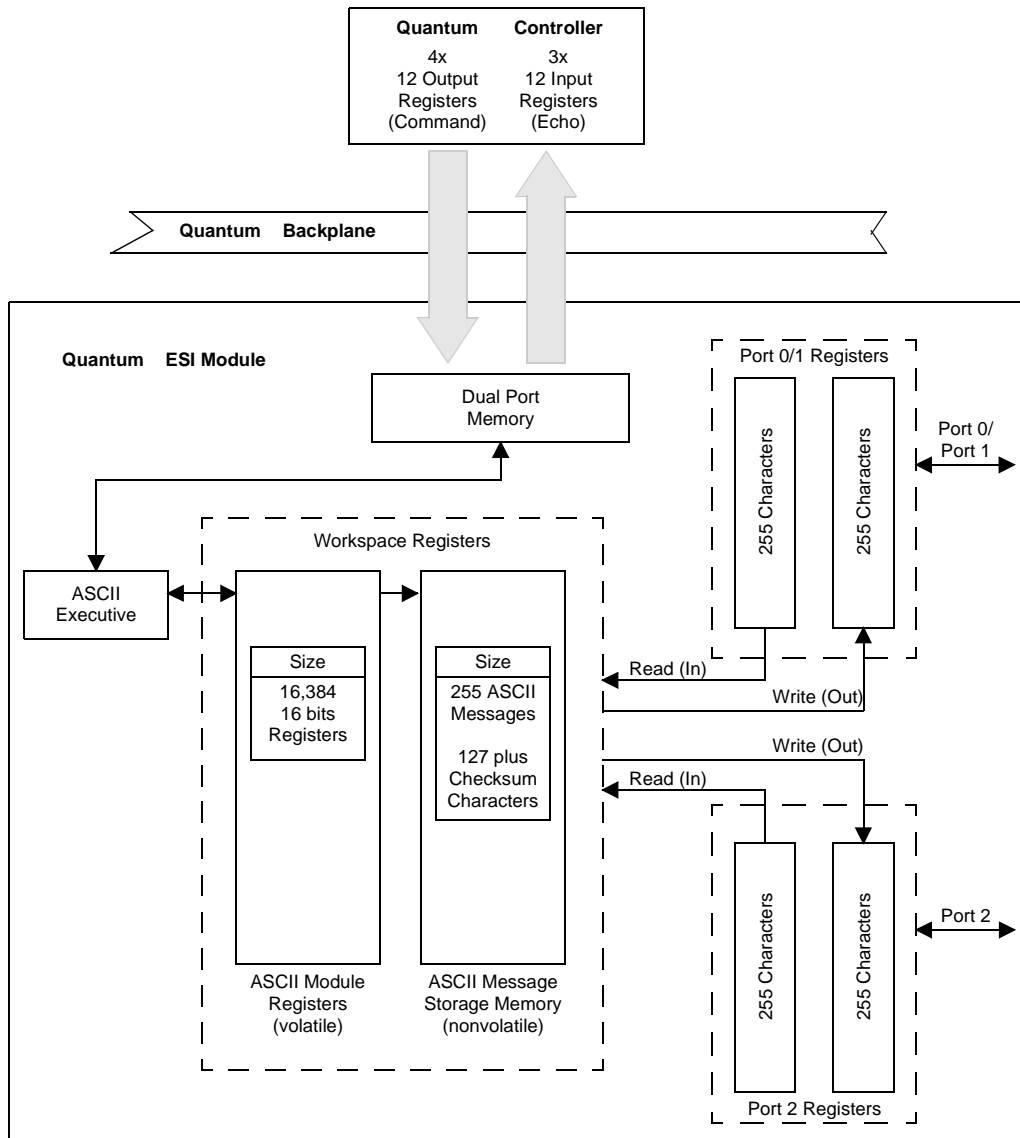
The ESI module can hold up to 255 ASCII messages with 127 characters plus checksum character each. These ASCII messages can be either static texts to be sent to an external device or a definition of how data contained in the register area is to be translated into or from a stream of serial ASCII characters, or a combination of both.

Firmware

The ESI module firmware can be loaded over the local I/O backplane. Upgrades and changes in functionality are supported by updating the flash executive firmware within the ESI module. Users should be aware that the update procedure can only occur over the local I/O backplane, even though the module can be placed in local, remote, or distributed locations. If you are using the ESI module in remote or distributed backplanes, plan to have an empty slot available in the local backplane, or a spare controller system to accommodate future executive upgrades.

ESI Module Block Diagramm

The Elements of the ESI Module The following picture shows the elements of the ESI module:



Hardware Overview and System Specifications

3

At a Glance

Introduction

The following information describes the 140 ESI 062 10 ASCII Interface Module.

What's in this Chapter?

This chapter contains the following topics:

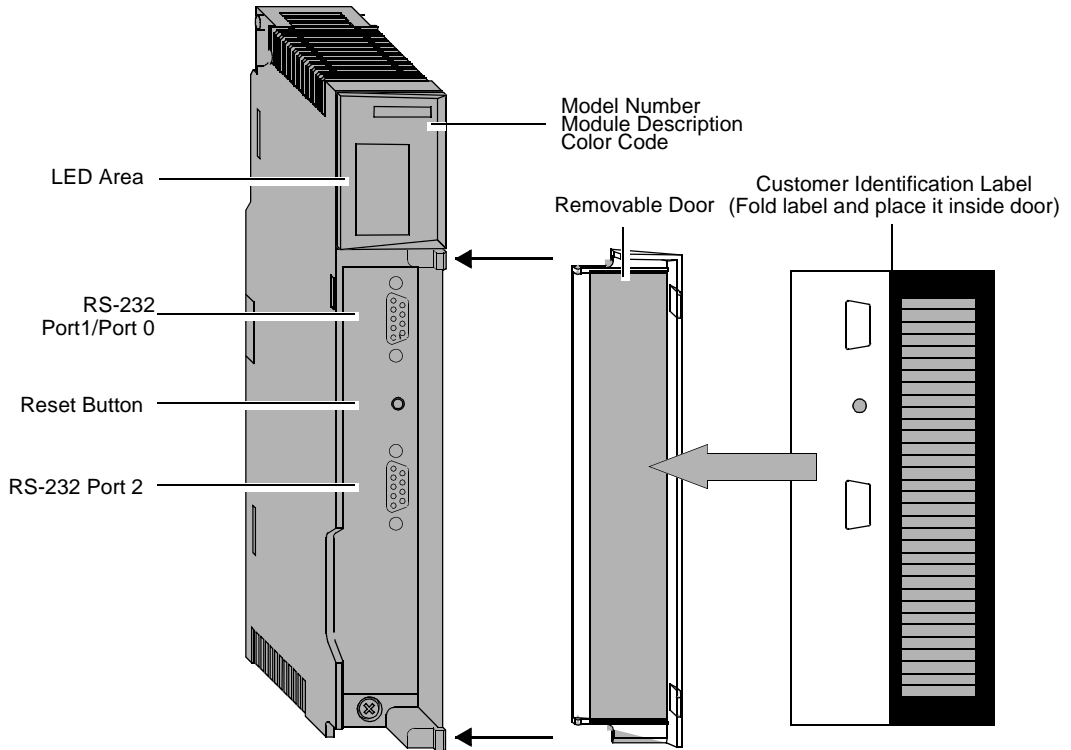
Topic	Page
140 ESI 062 10 ASCII Interface Module	28
Module Specifications	31
Quantum Automation Series System Specifications	32

140 ESI 062 10 ASCII Interface Module

Overview

The 140 ESI 062 10 module is a Quantum communications interface module used to input messages and/or data from an ASCII device to the CPU, output messages and/or data from the CPU to an ASCII device, or bi-directionally exchange messages and/or data between an ASCII device and the CPU.

The following figure shows the 140 ESI 10 ASCII Interface Module.



The ESI features two serial ports that can sustain communication rates of up to 9600 baud. Programming the ESI is done using port configurations and ASCII messages through one of the serial ports using a dumb terminal or with a personal computer using terminal emulation software. The ASCII messages are initiated by using logic running in the Quantum controller.

The ESI can be installed in local, remote I/O, and distributed I/O configurations.

LED Indicators and Descriptions

The information in the following table describes the ESI LEDs.

R Active F Rx1 Error 1 Tx1 Rx2 Error 2 Tx2 Status	LEDs	Color	Indication When On
	R	Green	The module has passed power up diagnostics.
	Active	Green	Bus communication is present.
	F	Red	The module has detected a fault.
	Rx1	Green	Received data on RS-232C Port 1.
	Tx1	Green	Transmitted data on RS-232C Port 1.
	Rx2	Green	Received data on RS-232C Port 2.
	Tx2	Green	Transmitted data on RS-232C Port 2.
	Status	Yellow	Status
	Error 1	Red	There is an error condition on Port 1.
	Error 2	Red	There is an error condition on Port 2.

LED Blinking Sequence

The following table shows the blinking sequence of the F, Status, Error 1, and Error 2 LEDs.

LEDs and Blinking Sequence				Description
F	Status	Error 1	Error 2	
Blinking	Blinking	Blinking	Blinking	The ASCII module is initializing (first power up).
OFF	ON	OFF	OFF	Programming mode
OFF	OFF	ON	N/A	Serial Port 1 incurred a buffer overrun.
OFF	OFF	N/A	ON	Serial Port 2 incurred a buffer overrun.
N/A	Blinking (see the LED Crash Codes table)	OFF	OFF	The ASCII module is in kernel mode and may have an error.

**Status LED
Crash Codes**

The following table shows the Status LED crash codes.

Number of Blinks (one per second)	Code (in hex)	Error
Steady	0000	Requested kernel mode
4	6631	Bad micro controller interrupt
5	6503	RAM address test error
6	6402	RAM data test error
7	6300	PROM checksum error (EXEC not loaded)
	6301	PROM checksum error
	630A	Flash-message checksum error
	630B	Executive watchdog timeout error
8	8000	Kernel other error
	8001	Kernel PROM checksum error
	8002	Flash program error
	8003	Unexpected executive return

**Front Panel
Reset Push
Button**

A recessed push button on the front of the module is used to reset the module.

Module Specifications

Specifications

The following table lists the module specifications for the ESI 062 10 ASCII 2CH.

Data Interface	
RS-232C	2 serial ports (9-pin D-shell), non-isolated
Cabling (Maximum cable length 20 m shielded)	990 NAA 263 20, Modbus Programming Cable, RS-232, 12 ft. (2.7 m)
	990 NAA 263 50, Modbus Programming Cable, RS-232, 50 ft. (15.5 m)
Firmware Specifications	
Port Performance	Burst Speed: 19.2 k baud each port
	Continuous Speed: Application dependent
Depth of Nested Messages	8
Buffer Size	255 Input 255 Output
	255 Output
Number of Messages	255
Maximum Message Length	127 characters plus 1 checksum
Memory	
RAM	256 kb for data and program + 2 kb dual port ram
Flash-ROM	128 kb for program and firmware
Power Dissipation	2 W max
Bus Current Required	300 mA
Fusing	
Internal	None
External	User discretion
Required Addressing	12 Words In, 12 Words Out
Compatibility	
Programming Software	Modsoft V2.4 or Concept 2.0 at a minimum, ProWorx NxT, ProWorx 32
Data Formats Supported	Text, Decimal, Fixed Point, Nested Write Message, Set Register Pointer, Print Time/Date, Repeat, Space, Newline, Control Code, Flush Buffer
Quantum Controllers	All, Executive V2.0 at a minimum
Battery Backup Module	140 XCP 900 00

Quantum Automation Series System Specifications

Overview

All Quantum Automation Series modules are designed to the system specifications that appear in the following tables. Note that the last table lists agency approvals.

Mechanical

The following table lists the mechanical system specifications.

Weight	2 lbs (1 kg) max
Dimensions (H x D x W)	9.84 in x 4.09 in x 1.59 in
	(250 mm x 103.85 mm x 40.34 mm)
Wire Size	1-14 AWG or 2-16 AWG max
	20 AWG min
Material (Enclosures and Bezels)	Lexan
Space Requirements	1 backplane slot

Electrical

The following table lists the electrical system specifications.

RFI Immunity (IEC 1000-4-3)	27... 500 MHz, 10 V/m
Electrostatic Discharge (IEC 1000-4-2)	8 kV air / 4 kV contact
Function I/O Modules with Operating Voltages Less Than 24 Vac or Vdc	
Fast Transients (IEC 1000-4-4)	0.5 kV common mode
Damped Oscillatory Transients	1 kV common mode
	0.5 kV differential mode
Surge Withstand Capability (Transients) (IEC 1000-4-5)	1 kV common mode
	0.5 kV differential mode

Operating Conditions

The following table provides system specifications for operating conditions.

Temperature	0... 60°C (32... 140°F)
Humidity	0... 95% RH noncondensing @ 60°C
Chemical Interactions	Enclosures and bezels are made of Lexan, a polycarbonate that can be damaged by strong alkaline solutions
Altitude	2,000 meters
Vibration)	10... 57 Hz @ 0.075 mm d.a.
	57... 150 Hz @ 1 g
Shock	+/-15 g peak, 11 ms, half-sine wave

Storage Conditions

The following table provides system specifications for storage conditions.

Temperature	40... 85°C (-40... 185°F)
Humidity	0... 95% RH noncondensing @ 60°C
Free Fall	3 ft. (1 m)

Agency Approvals

The following table lists necessary agency approvals.

UL 508
CSA 22.2-142
Factory Mutual Class I, Div 2
European Directive on EMC 89/336/EEC

140 ESI 062 10 Hardware Description



At a Glance

Introduction

The information in this chapter describes the hardware features of the 140 ESI 062 10 module. Product specifications are included at the end of the chapter.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Presentation	36
Indicators	37
External Connectors and Switches	39
Specifications	41

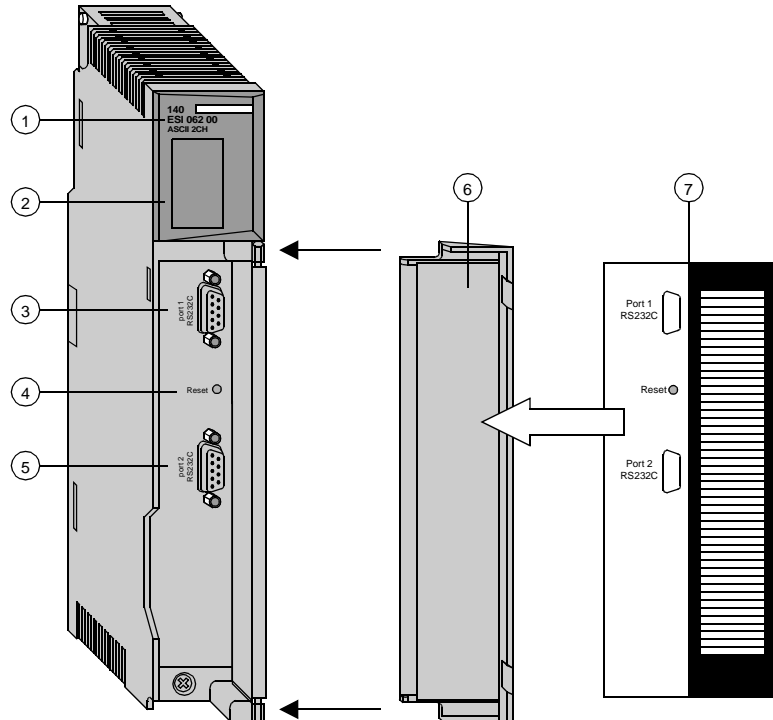
Presentation

Function

The 140 ESI 062 10 module is a Quantum communications interface module used to input messages and/or data from an ASCII device to the CPU, output messages and/or data from the CPU to an ASCII device, or bi directionally exchange messages and/or data between an ASCII device and the CPU.

Illustration

The following figure shows the 140 ESI 062 10 module and its components.



- 1 Model Number, Module Description, Color Code
 - 2 LED Display
 - 3 Port 1 Connector
 - 4 Reset Button
 - 5 Port 2 Connector
 - 6 Removable door
 - 7 Customer Identification Label (Fold label and place it inside door)
-

Indicators

Illustration

The following table shows the LED indicators for the 140 ESI 062 10 module.

R	Active	F
Tx 1		Error 1
Rx 1		
Tx 2		Error 2
Rx 2		
Status		

Descriptions

The following table describes the LED descriptions for the 140 ESI 062 10 module.

LEDs	Color	Indication when ON
R	Green	The module has passed powerup diagnostics.
Active	Green	Bus communication is present.
F	Red	The module has detected a fault.
RX1	Green	Received data on RS-232 Port 1
TX1	Green	Transmitted data on RS-232 Port 1
RX2	Green	Received data on RS-232 Port 2
TX2	Green	Transmitted data on RS-232 Port 2
Status	Yellow	Status
Error 1	Red	There is an error condition on Port 1
Error 2	Red	There is an error condition on Port 2

LED Blinking Sequence

The following table shows the blinking sequence of the F, Status, Error 1, and Error 2 LEDs.

LED				Description
F	Status	Error 1	Error 2	
Blinking	Blinking	Blinking	Blinking	The ASCII module is initializing (first powerup)
OFF	ON	OFF	OFF	Programming mode
OFF	OFF	ON	N/A	Serial Port 1 incurred a buffer overrun
OFF	OFF	N/A	ON	Serial Port 2 incurred a buffer overrun
N/A	Blinking (See Crash Codes)	OFF	OFF	The ASCII module is in kernel mode and may have an error

Status LED Crash Codes

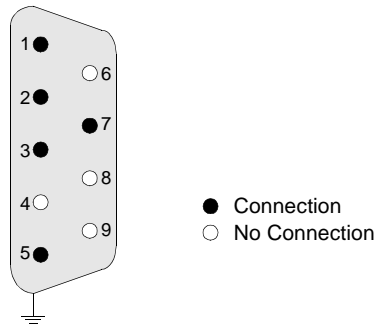
The following table shows a table of Status LED crash codes.

Number of Blinks (one per second)	Code (in hex)	Error
Steady	0000	Requested kernel mode
4	6631	Bad microcontroller interrupt
5	6503	RAM address test error
8	6402	RAM data test error
7	6300	PROM checksum error (EXEC not loaded)
	6301	PROM checksum error
	630A	Flash message checksum error
	630B	Executive watchdog timeout error
8	8000	Kernel other error
	8001	Kernel PROM checksum error
	8002	Flash program error
	8003	Unexpected executive return

External Connectors and Switches

RS-232 Serial Ports

The ESI has two serial ports which it uses to communicate with serial devices. The following diagram shows the pinout connections for the ASCII module serial ports.



Pinout for the RS-232 ports:

Pin	Signal Name	Description
1	DCD	Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	N/A	Not Connected
5	GND	Signal Ground
6	N/A	Not Connected
7	RTS	Request to Send
8	N/A	Not Connected
9	N/A	Not Connected
Shield	N/A	Chassis Ground

Programming Port

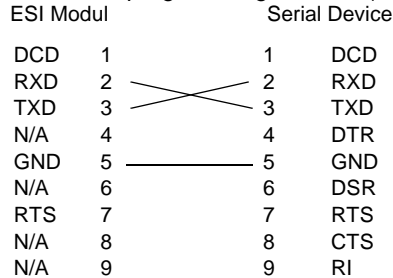
Port 1 is also used as the programming port (Port 0). This mode is entered by pressing the Reset button for more then 4 seconds. When programming mode is entered the serial port is set to a standard terminal communications configuration. Programming mode uses the following port settings:

Parameter	Setting
Baud rate	9600
Data bits	8
Stop bits	1
Parity bit	None (disabled)
Keyboard Mode	ON (Character echo)
XON/XOFF	ON

The serial port configuration has been set this way so that the configuration of the port is a known configuration and may or may not be the same configuration that is used when the module is running.

Minimum Cable Layout

The minimum required cable layout to connect the ESI module either to an external device or a programming terminal (PC) is shown in the following illustration:



Reset Push Button

A recessed push button is located on the front of the module.

The reset push button has two functions:

- Reset the module by a short press
 - Entering programming mode by holding the button pressed for more than 4 seconds
-

Specifications

Data Interface

Data Interface

RS-232	2 serial ports (9-pin D-shell), non-isolated
Cabling (Maximum cable length 20 m shielded)	990 NAA 263 20, Modbus Programming Cable, RS 232, 12 ft (2.7 m)
	990 NAA 263 50, Modbus Programming Cable, RS 232, 50 ft (15.5 m)

Firmware

Firmware Specifications

Port Performance	Burst Speed:	19.2 k baud each port
	Continuous Speed:	Application dependent
Depth of Nested Messages	8	
Buffer Size	255 Input 255 Output	
Number of Messages	255	
Maximum Message Length	127 characters plus 1 checksum	

Memory

Memory Specifications

RAM	256 kb for data and program + 2 kb dual port ram
Flash-ROM	128 kb for program and firmware

Power

Power Specifications

Power Dissipation	2 W max
Bus Current Required	300 mA

Fuses

Required Fuses

Internal	None
External	User discretion

I/O Mapping

Required Addresses

In	12 Words
Out	12 Words

Compatibility

Compatibility

Programming Software	Concept 2.0, ProWorx NxT, ProWorx 32, Modsoft
Data Formats Supported	Text, Decimal, Fixed Point, Nested Write Message, Set Register Pointer, Print Time/Date, Repeat, Space, Newline, Control Code, Flush Buffer
Quantum Controllers	All, Executive V2.0 at a minimum
Battery Backup Module	140 XCP 900 00

Mechanical

Mechanical

IWeight	1 kg max
Dimensions (H x D x W)	250 mm x 103.85 mm x 40.34 mm
Material	(Enclosures and Bezels) Lexan
Space Requirements	1 backplane slot

Electrical

Electrical

RFI Immunity (IEC 1000-4-3)	27 ... 500 MHz, 10 V/m
Electrostatic Discharge (IEC 1000-4-2)	8 kV air / 4 kV contact
Fast Transients (IEC 1000-4-4)	0.5 kV common mode
Damped Oscillatory Transients	1 kV common mode 0.5 kV differential mode
Surge Withstand Capability (Transients) (IEC 1000-4-5)	1 kV common mode 0.5 kV differential mode

Environmental Conditions

Environmental Conditions for Operation

Temperature	0 ... 60°C (32 ... 140°F)
Humidity	0 ... 95% RH noncondensing @ 60°C
Chemical Interactions	Enclosures and bezels are made of Lexan, a polycarbonate that can be damaged by strong alkaline solutions.
Altitude	2,000 meters
Vibration	10 ... 57 Hz @ 0.075 mm d.a. 57 ... 150 Hz @ 1 g
Shock	+/-15 g peak, 11 ms, half-sine wave

Storage Conditions

Storage Conditions

Temperature	~40 ... 85°C (-40 ... 185°F)
Humidity	0 ... 95% RH noncondensing @ 60°C
Free Fall	1 m

Agency Approvals

Agency Approvals

UL 508 CSA 22.2-142 Factory Mutual Class I, Div 2 European Directive on EMC 89/336/EEC

Configuration Overview



5

At a Glance

Introduction

The information in this chapter describes the basics of the configuration mode of the ESI modul. A description of the data flow between external devices and the PLC is included at the end of the chapter.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
ESI Configuration	46
Data Flow	49

ESI Configuration

Overview

The ESI module has a built in command line editor, which is used to configure the port communication settings, the internal clock and the ASCII messages.

Programming Port

The ESI module supports two RS 232 hardware ports, which have their individual parameter settings at runtime. The first port also is used as a programming port. In this mode it has its own set of parameters.

How to Enter the Configuration Mode

To enter the configuration mode of the ESI you have to perform the following steps:

Step	Action
1	Connect a dumb terminal or a PC Terminal emulation like Hyperterminal to port 1. For information about the appropriate cable see <i>RS -232 Serial Ports</i> , p. 39
2	Set the communication parameters of the terminal to 9600 baud, 8 data bit, no parity, 1 stop bit and XON/XOFF flow control.
3	Press the reset button on the front of the ESI module for more then 4 seconds.

The Command Line Editor

After you have entered the configuration mode following the above steps, the yellow Status LED on the ESI front panel will turn on, and you get the following message on your terminal screen:

```
Welcome
MODICON QUANTUM ASCII Module
Entering Program Mode ...
Current date is: Wed 01-01-2002
Current time is: 09:15:10a
CLI> _
```

**Available
Commands**

Command structure of the CLI:

Command		Description	Example
CLI		Sets programming mode to the Command Line Interpreter.	N/A
HELP		Displays available commands and a brief description on the command, or displays help on the command requested (e.g., CLI> HELP ASCII displays help on the ASCII command.)	N/A
RUN		Resets Module and goes into normal running mode.	N/A
CONFIG		Sets programming mode to Configuration Interpreter.	N/A
	DATE	Displays or sets the current date in the module.	See chapter Configuration Editor for examples
	TIME	Displays or sets the current time in the module.	
	PORT	Displays or sets the port parameter settings.	
ASCII		Sets programming mode to ASCII Message Interpreter.	N/A
	NEW	Enters the message editor and holds the new message in the work buffer.	ASCII>new
	EDIT	Displays a specified message, enters the message editor, and saves the specified message when done.	ASCII>edit (message #)
	VIEW	Displays an existing message for viewing.	ASCII>view (message #)
	SAVE	Saves changes made to a specified message in its work buffer.	ASCII>save (message #)
	CLR	Clears a specified message.	ASCII>clr (message #)
	COPY	Copies a specified message to another message.	ASCII>copy (message #) (message #)
	SIM	Simulates a specified message. Shows how many registers are used (for aid in mapping when creating user logic) and the maximum depth of nested messages (for additional debugging tool). Notification is sent if the maximum depth is greater than 8 and also shows the nested message path.	ASCII>sim (message #)

Command	Description	Example
DIR	Display a directory of all available messages. Use of CNTL S and CNTL Q can be used to stop and continue the data being displayed to the terminal.	N/A.
DLOAD	Download messages from a PC to the module. See ASCII Message Transfer for more details.	N/A.
ULOAD	Uploads all programmed messages (1 ... 255).	ASCII>uload
	Uploads a specified programmed message(s) from the module to a PC. See ASCII Message Transfer for more details.	ASCII>uload (message # - message #)

Data Flow

Overview

Exchanging data between the Quantum processor and the serial ports of the ESI module involves the following steps:

Transmit direction:

- Transfer of the data from the PLC registers to the ESI register area through the 12 output registers assigned to the ESI module in the I/O configuration.
- Interpreting the data in the ESI registers based on the ASCII messages and transfer to the port transmit buffer.

Receive direction:

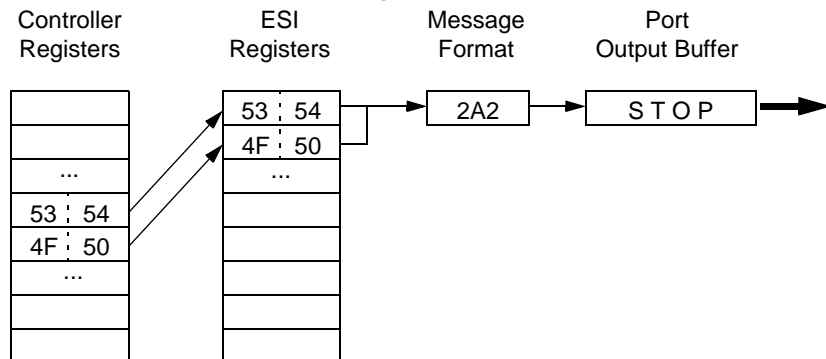
- Interpreting the data in the port receive buffer based on the ASCII messages and transfer to the ESI register area.
- Transfer of the data from the ESI register area to the PLC registers through the 12 input registers assigned to the ESI module in the I/O configuration.

ASCII Messages

The ASCII messages represent the central mechanism of how the data in the ESI registers are formatted for the transmission through the RS-232 ports in either direction. A single 16-bit register for example could represent 2 ASCII characters and thus be transmitted as two characters it could also represent a single number which may be transmitted as an integer with leading spaces resulting in a string of five characters. For a detailed description of the available formats see *ASCII Message Formats*, p. 59.

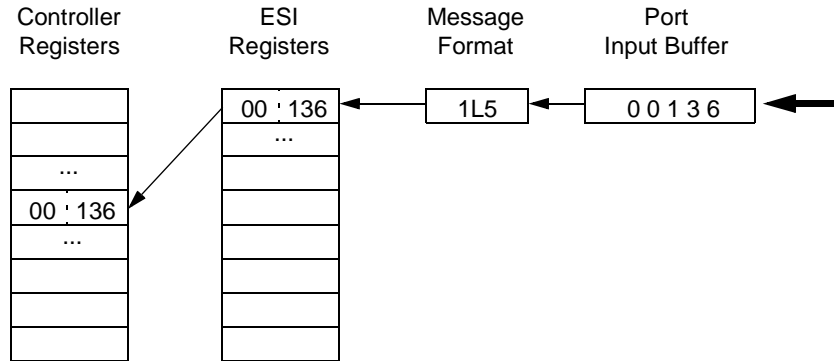
Transmitting Example

The following diagram is an example of transmitting 4 characters from the Quantum controller using the "2A2" message format (2 registers with 2 characters each). Port Buffer content is in ASCII format, register content in hex:



Receiving Example

The following diagram is an example of receiving 1 numerical value from the RS-232 port using the "1L5" message format (1 register, 5 digits with leading zeros). Port Buffer content is in ASCII format, register content in hex:



Note: Enusre the number of incoming characters match the number defined in the ASCII message. If in the above example the device sends "0013", the ESI module would not be able to finish the receive command and would wait until reception of a 5th character.

Possible Synchronisation Problems

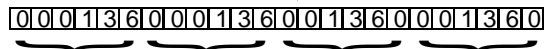
As the ESI module only supports fixed length message formats without start or termination characters, any lost character (or additional unexpected character) can lead to a wrong interpretation of received data. The following examples show the result of 3 different error types. The assumed message format is "1L5 maximum 65,535":

Effect of lost character:

Reason for error:

loss of one character

Buffer content:

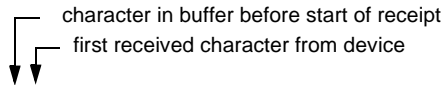


Data interpretation:

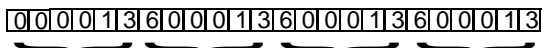
136 136 1360 1360

Effect of buffer not empty at start of reception:

Reason for error:



Buffer content:



Data interpretation:

13 Error Error Error

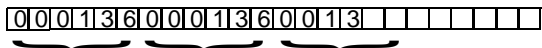
(not a valid Integer)

Effect of terminated reception:

Reason for error:

Device stops transmission

Buffer content:



Data interpretation:

136 136 wait for next 2 characters

FLUSH, ABORT, GET STATUS

To prevent mis-interpretation of data or locking the module the buffer related commands FLUSH BUFFER, ABORT, GET BUFFER STATUS should be used to control the data exchange.

For details of those commands see *List of ESI Commands*, p. 66.

ESI Command Line Editors

6

Overview

At a Glance

The ESI firmware contains an editing environment which can be accessed by dumb terminal connected through port 1. This chapter describes how to use this editor to configure the module and to edit the ASCII message formats.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Configuration Editor	54
ASCII Message Editor	58
ASCII Message Formats	59

Configuration Editor

Overview

The Configuration Editor Interface is part of the programming mode. It is used to configure the serial ports and the time of day clock of the module.

Note: Configuration of the serial ports can also be accomplished through the I/O map. The I/O map overrides any serial port configuration entered in the configuration editor.

Note: Configuration of the time of day clock can also be accomplished with the SET TOD command.

To enter the configuration editor type `CONFIG` at the `CLI>` prompt. The configuration editor displays the prompt `CONFIG>` .

Port Command

The Port Command displays or sets the port parameter settings. Acceptable command format variations include:

```
PORT [n[: [b] [,p] [,d] [,s] [,k] [,x]]]
PORT [n[: [BAUD=b] [,PARITY=p] [,DATA=d] [,STOP=s]
[,KEYBOARD=k] [,XON/XOFF=x]]]
```

Description and range of the elements used in the PORT command:

Index	Description	Range
n	Port number	0, 1, 2
b	Baud rate	50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200
p	Parity setting	N, O, E
d	Number of data bits	5, 6, 7, 8
s	Number of stop bits	1, 2
k	Keyboard mode (Character echo mode)	on, off
x	XON/XOFF mode (Software flow control)	on, off

Examples:

```
PORT 0:1200,n,8,1,on,on
PORT 0:baud=1200, parity=n, data=8, stop=1, keyboard=on, XON/
XOFF=on
PORT 0
Current port parameters are: PORT 0: BAUD=1200, PARITY=NONE
...
Enter new parameters: 4800,n,8,1,off,on
After the Port settings in the module have been changed, the following message will
appear:
Note: The port settings are temporary during this programming
session.
```

Note: Ports 0 and 1 do not support all baud rate and data bit options. Refer to the Module Configuration screen for available options.

Date Command Displays or sets the current date in the module. Acceptable command format variations include:

```
DATE      [mm dd [ yy]]
DATE      [mm/dd [/ yy]]
DATE      [mm.dd [.yy]]
DATE      [mm dd [ YYYY]]
DATE      [mm/dd [/YYYY]]
DATE      [mm.dd [.YYYY]]
```

Description and range of the elements used in the DATE command:

Index	Description	Range
mm	Month	1 ... 12
dd	Day	1 ... 31
yy	Year	00 ... 99
yyyy	Year	1990 ... 2089

Examples:

```
DATE 3 30 95
DATE 3/3 0/1995
DATE
Current date is Wed 3 29 1995
Enter new date: 3.30
```

Note: If the year does not need to be changed, then only the month and day need to be entered. The day of week is automatically figured out by the firmware. The yy years are mapped 00..89 = 2000..2089 and 90..99 = 1990..1999.

Time Command Displays or sets the current time in the module. Acceptable command format variations include:

```
TIME [hh:mm[:ss] [x]]  
TIME [hh.mm[.ss] [x]]
```

Description and range of the elements used in the TIME command:

Index	Description	Range
hh	Hour	1 ... 23
mm	Minute	1 ... 59
ss	Second	1 ... 59
x	Meridian	a, p

Examples:

```
TIME 3:26p  
TIME 3.26.30p  
TIME 15.26  
TIME  
Current time is 3:15:26p  
Enter new time: 3.26.30p
```

Note: The time can be entered in either 12 or 24 hour time format. Not entering the meridian assumes AM unless the hour is 0 or 13 to 23.

ASCII Message Editor

Overview

The ASCII Message Editor Interface is used to program the ASCII message formats in the module. This interface consists of a simple command line interpreter (also similar to the CLI that is in the Modicon B885 002 module), which consists of commands that allow you to display, create, edit, transfer, save, clear, and test ASCII messages. Also included in the command set is a help command, which gives an online list of the available commands and the meaning of each command. To enter the ASCII message editor type `ASCII` at the `CLI>` prompt. The ASCII message editor uses the prompt `ASCII>`

ASCII Message Formats

Overview

The ASCII message formats define how data contained in the CPU is converted to a stream of serial characters and vice versa.

The following table lists the available message formats:

Format	Direction	Description
Text	Output	Static text
ASCII	Output/Input	ASCII characters
Hexadecimal	Output/Input	Hexadecimal numbers
Octal	Output/Input	Octal numbers
Binary	Output/Input	Binary numbers
Integer	Output/Input	Integer numbers
Fixed Point Decimal	Output/Input	Fixed Point Decimal numbers
Time/Date	Output	Time/Date information
Control Characters	Output	Space and Newline characters
Control Sequences	Output	3 digit octal control characters
Nesting	Output/Input	Nesting of messages

Text

An arbitrary ASCII string, enclosed in single quotes (e.g. 'message string') is an output only format. Any message that contains this format will output the text whether or not the message is started from a read or write message command.

'... (text) ...'

ASCII Characters

A variable field of the ASCII format with Number of registers and Field Length:

nAm

n is the number of registers 1..99 (format repeat)

m is the field length 1..2 (number of characters)

Example: 2A2 as an input stands for 2 registers, each containing 2 ASCII characters.

Hexadecimal

A variable field of the Hexadecimal format with Number of registers and Field Length:

nHm

n is the number of registers 1..99 (format repeat)

m is the field length 1..4 (number of numbers)

Example: 2H3 as an input stands for 2 registers, each containing 3 Hexadecimal numbers.

Octal

A variable field of the Octal format with Number of registers and Field Length:

nOm

n is the number of registers 1..99 (format repeat)

m is the field length 1..6 (number of numbers)

Example: 3O4 as an input stands for 3 registers, each containing 4 Octal numbers.

Binary

A variable field of the Binary format with Number of registers and Field Length:

nBm

n is the number of registers 1..99 (format repeat)

m is the field length 1..16 (number of numbers)

Example: 1B8 is an input stands for 1 register containing 8 Binary numbers.

Integer Leading Spaces

A variable field of the Integer/Decimal format using leading spaces for output with Number of registers and Field Length. On input, this format accepts leading zeros and spaces as a 0 (zero).

nIm

n is the number of registers 1..99 (format repeat)

m is the field length 1..5 (number of numbers)

Example: 2I5 as an input stands for 2 registers, each containing 5 Integer/Decimal numbers. Max value is 65,535.

**Integer Leading
Zeroes**

A variable field of the Integer/Decimal format using leading zeroes for output with Number of registers and Field Length. On input this format accepts leading zeroes and spaces as a 0 (zero).

nLm

n is the number of registers 1..99 (format repeat)

m is the field length 1..5 (number of numbers)

Example: 3L5 as an input stands for 3 registers, each containing 5 Integer/Decimal numbers. Max value is 65,535.

**Fixed Point
Decimal**

A variable field of the Fixed Point Decimal format using leading spaces for output with Number of registers and Field Length. On input, this format accepts leading zeros and spaces as a 0 (zero).

nPm.q

n is the number of registers 1..99 (format repeat)

m is the number of numbers + '.' 3..8

q is the number of fraction numbers 1..5

Example: 1P7.2 as an input stands for 1 register containing 4 Decimal numbers followed by a '.' which is followed by 2 more Decimal numbers (the fraction part).

<p>Note: This format should not be mixed up with a floating point format. The placement of the decimal point is for input/output formatting and has no influence on the value in the PLC register (e.g. all 3 values 23.456, 234.56 and 23456 will refer to a register value of 23456).</p>
--

Nested Message

The nesting message format allows a message to call another message. This format can be used within the repeat format; repeat formats can be used in the nested message allowing for indirect nested repeats. The maximum allowable nested message levels is 8. Recursive nesting is not allowed.

Mn

n is the message number 1..255

Example: M6 will run message number 6.

Time There are two different time formats for displaying the time. One is for 12 hour time and the other is for 24 hour time. This is an output only format.

T12 > hh:mm:ss AM/PM (12 hour time)

T24 > hh:mm:ss (24 hour time)

Date There are five different date formats for displaying the date, each having 2 types of formats for displaying the year. This is an output only format.

Dnm

n is the day and month type 1..5

m is the year type 2 or 4

D12 > dd/mm/yy

D14 > dd/mm/yyyy

D22 > mm/dd/yy

D24 > mm/dd/yyyy

D32 > dd mmm yy

D34 > dd mmm yyyy

D42 > mmm dd, yy

D44 > mmm dd, yyyy

D52 > dd.mm.yy

D54 > dd.mm/yyyy

dd = day (1..31)

mm = month (1..12)

mmm = month (JAN, FEB, .. , DEC)

yy = year (0..99) (90 - 99 in 1900's, 0 - 89 in 2000's)

yyyy = year (1990..2089)

Repeat Repetition of several formats; nesting of repeat brackets is not valid.

n(...)

n is the number of times to repeat what is in ()1. .99

Example: 6('Item',1I2,4X,1I5,/) will produce 6 lines, each containing the fields 'Item',1I2,4X,1I5, and a <CR, LF>.

Space	<p>The ASCII message symbol for space is X. This is an output only format.</p> <p>nX n is the number of spaces 1..99</p>								
Newline	<p>The ASCII message symbol for a 'Carriage Return' is I. This is an output only format.</p>								
Control Code	<p>A 3 digit Octal control character (in the range 000 377) enclosed in double quote delimiters. This is an output only format.</p> <p>"###" ### is the octal form of a character</p> <p>Example: "033".</p>								
Flush	<p>Flush the input buffer of the currently running serial port in one of four ways: the entire buffer, a number of characters, up to a character pair, or up to a character pair repeatedly</p> <table><tr><td><0></td><td>flush entire buffer</td></tr><tr><td><1;bbb></td><td>flush until number of characters removed</td></tr><tr><td><2;hhhh></td><td>flush until character pair match</td></tr><tr><td><3;rrr;hhhh></td><td>flush until character pair match repeatedly</td></tr></table> <p>bbb = number of characters (1..255) hhhh = character pair, in hexadecimal (0000..FFFF) rrr = number of repeats (1..255)</p>	<0>	flush entire buffer	<1;bbb>	flush until number of characters removed	<2;hhhh>	flush until character pair match	<3;rrr;hhhh>	flush until character pair match repeatedly
<0>	flush entire buffer								
<1;bbb>	flush until number of characters removed								
<2;hhhh>	flush until character pair match								
<3;rrr;hhhh>	flush until character pair match repeatedly								

Note: The port buffer size is 255 characters.

**ASCII Message
Syntax Rules**

Messages created with the Module's ASCII Message Editor or downloaded using the ASCII Message Transfer are checked after being entered for general and format syntax violations. If any violations are found, the message either is not saved (ASCII Message Transfer) or the user is notified and the violation is pointed out (ASCII Message Editor).

- A format delimiter (,) must separate each format.
 - All text formats must be closed.
 - Formats A,H,O,B,I,L,P,X, and (can have a repeat/number of registers value from 1 to 99.
 - Formats A,H,O,B,I, and L can have a total field size from 1 to 8.
 - Format P can have a total field size from 3 to 8 and a fractional field size from 1 to 5 but the total field size must be at least 2 greater than the fractional field size.
 - Format M (Nested Message) can have any message number 1 to 255 (decimal) as long as it is not recursive.
 - Format T can have 1 of 2 formats: T12 or T24.
 - Format D can have 1 of 10 formats: D12, D14, D22, D24, D32, D34, D42, D44, D52, and D54.
 - Control Code format "####" accepts only 3 digit octal values from 000 to 377.
 - Flush format can have 1 of 4 formats: <0>, <1;bbb>, <2;hhhh>, or <3;rrr;hhhh> where bbb = 1 to 255, hhhh = 0000 to FFFF, and rrr = 1 to 255.
-

**Standard ASCII
Message
Preprocessing
Rules**

Messages created with the Module's ASCII Message Editor or downloaded using the ASCII Message Transfer are preprocessed after being entered to save space and to standardize the messages for interpretation during simulation or running mode.

- Text is not massaged at all.
Example: >'This is text...' >>'This is text...'
 - Spaces preceding the first format are removed.
Example: > 1A4,2X >>1A4,2X
 - Spaces trailing the last format are removed.
Example: >1A4,2X (end) >>1A4,2X(end)
 - Spaces around formats and delimiters are removed.
Example: >1A4 , 2X >>1A4,2X
 - Commas trailing the last format are removed.
Example: >1A4,2X,, >>1A4,2X
 - Commas trailing the last format in a repeat format are removed.
Example: >1A4,2X,3(1I2,1X,,)/ >>1A4,2X,3(1I2,1X)/
 - Non text characters are capitalized.
Example: >'text ',1a4,2x,/ >>'text ',1A4,2X,/
 - All preceding 0's are removed from a number except 0's in flush format's repeat/number value and character pair value.
Example: >01A004,0002X >>1A4,2X
-

ESI Commands



At a Glance

Introduction

The information in this chapter describes the commands which are sent by the CPU to control the communication functions of the ESI module and the response from the ESI module containing data and status information.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Overview on ESI Commands	66
ESI Command Structure	67
Command 0 - NO OPERATION	68
Command 1 - READ ASCII MESSAGE	69
Command 2 - WRITE ASCII MESSAGE	71
Command 3 - GET DATA (Module to Controller)	74
Command 4 - PUT DATA (Controller to Module)	76
Command 5 - GET TOD (Time of Day)	78
Command 6 - SET TOD (Time of Day)	80
Command 7 - SET MEMORY REGISTERS	83
Command 8 - FLUSH BUFFER	85
Command 9 - ABORT	86
Command A - GET BUFFER STATUS	87
Response Structure for Illegal Commands	89
Module Status Word (Word 11)	90
Reading beyond valid Register Range	92

Overview on ESI Commands

List of ESI Commands

There are 11 ASCII module commands which instruct the ESI module serial communications and other housekeeping utilities. These commands are sent to the ESI module by the Quantum controller. Data exchange between the ASCII device and the Quantum controller is integrated into the READ/WRITE command structure described in this section. The output data (the first 4x registers) contains the command; the first input register (3x) contains the response and also the echo of the command.

The following table is a summary of the ESI module commands:

Command	Name	Description
0	No operation	do nothing
1	READ ASCII message	start a read ASCII message
2	WRITE ASCII message	start a write ASCII message
3	GET DATA	transfer data from module to PLC
4	PUT DATA	transfer data from PLC to module
5	GET TOD	get time of day from module
6	SET TOD	set time of day in module
7	SET MEMORY REGISTERS	set registers to value
8	FLUSH BUFFER	flush serial port buffers
9	ABORT	abort ASCII message currently running
A	GET BUFFER STATUS	get port input buffer

ESI Command Structure

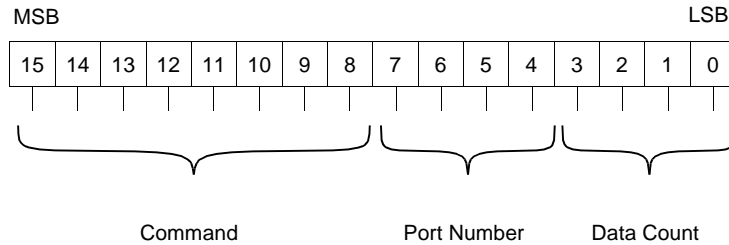
Command Word Format

The command word is the first output register mapped to the module.

The command word format for the ESI module is as follows:

- Bits 0 ... 3 - contain the data count (in words), range is 0 ... 9
- Bits 4 ... 7 - contain the port number, range is 1 ... 2
- Bits 8 ... 15 - contain the command, range is 0 ... A

Structure of the command word:



Note: The bit order is based on the IEC standard, where bit 15 is the most significant bit.

Command 0 - NO OPERATION

Overview

The NO OPERATION command does nothing in or to the ESI module. It is present to allow multiple scan command builds (setting up of Command Words 1 to 11, then setting Command Word 0 to start the command execution) and toggling for repeating command that do not run continuously.

This command is executed continuously until Command Word 0 changes to a command other than NO OPERATION.

Command Structure

Command Structure for Command 0

Word 0 0000 (hex)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Word 1 through Word 11 for Command 0 are not used.

Response Structure

Response Structure for Command 0

Word 0 0000 (hex) Echo Command Word 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Bit 15 is the Status Word Valid bit.

•
•
•

Word 11 XXXX hex Module Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Word 1 through Word 10 for Command 0 returns a 0.

Command 1- READ ASCII MESSAGE

Overview

The READ ASCII MESSAGE command is used to start running a read message on the module, that is, taking ASCII characters from the input/receive buffer of a serial port to fulfil the variable formats of the message. All output only formats still send ASCII characters to the serial port.

To start a message, the module needs to know the following:

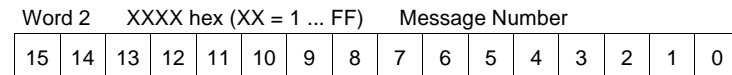
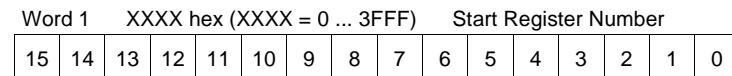
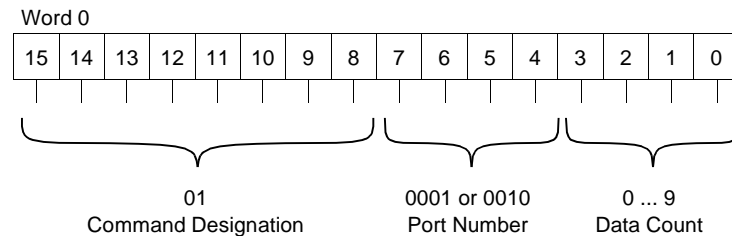
- The port number to be used
- The starting module register number for the data that is processed
- The message number to run

In addition to starting a message, this command is capable of transferring up to nine registers of data from the module to the controller after the message has completed (this is the data count). The data returned is gotten from the starting register number provided in Command Word 1.

This command is executed only the first time it is received. To execute the command again, Command Words 0, 1, or 2 need to be changed. This is done so that the same message does not get continuously run until Command Word 0 changes to a command other than READ ASCII MESSAGE.

Command Structure

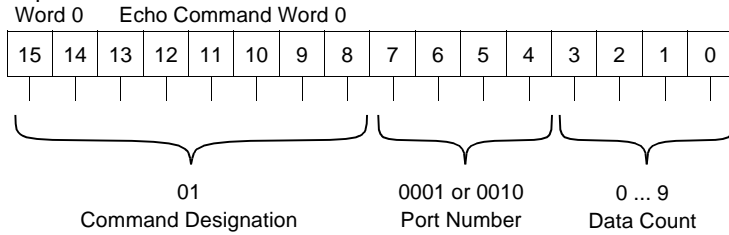
Command Structure for Command 1



Note: Word 3 through Word 11 for Command 1 are not used.

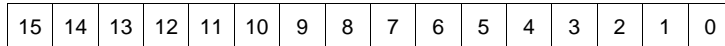
Response Structure

Response Structure for Command 1

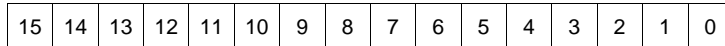


Note: Bit 15 is the Status Word Valid bit.

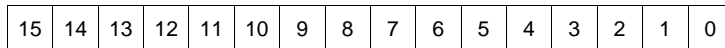
Word 1 XXXX hex (XXXX = 0 ... 3FFF) Echo Start Register Number



Word 2 XXXX hex (XX = 1 ... FF) Echo Message Number

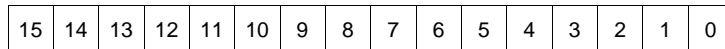


Word 3 XXXX hex Data Word 1



•
•
•

Word 11 XXXX hex Module Status or Data Word 9



Command 2 - WRITE ASCII MESSAGE

Overview

The WRITE ASCII MESSAGE command is used to start running a write message on the module, that is, putting ASCII characters to the output/transmit buffer of a serial port.

To start a message, the module needs to know the following:

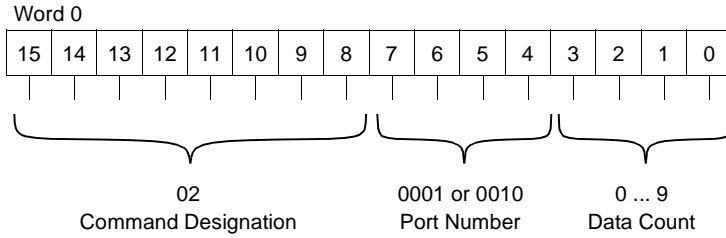
- The port number to be used
- The starting module register number for the data that is processed
- The message number to run

In addition to starting a message, this command is capable of transferring up to nine registers of data from the controller to the module before the message has started (this is the data count). The data sent is stored starting at the start register number provided in Command Word 1.

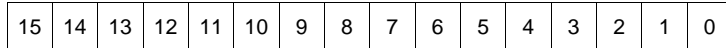
This command is executed only the first time it is received. To execute the command again, Command Words 0, 1, or 2 (plus any data word that is sent - keyed off the data count) need to be changed. This is done so that the same message does not get continuously run until Command Word 0 changes to a command other than WRITE ASCII MESSAGE.

Command Structure

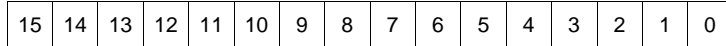
Command Structure for Command 2



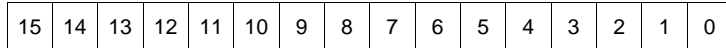
Word 1 XXXX hex (XXXX = 0 ... 3FFF) Start Register Number



Word 2 XXXX hex (XX = 1 ... FF) Message Number

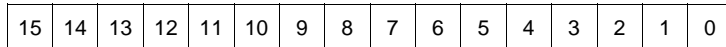


Word 3 XXXX hex Data Word 1



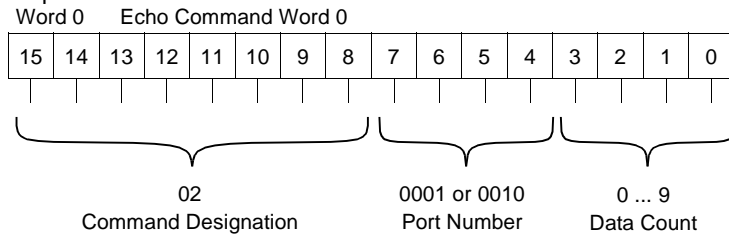
•
•
•

Word 11 XXXX hex Data Word 9



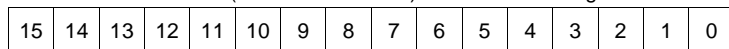
Response Structure

Response Structure for Command 2

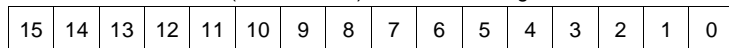


Note: Bit 15 is the Status Word Valid bit.

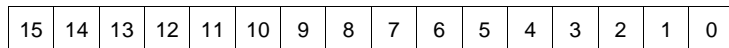
Word 1 XXXX hex (XXXX = 0 ... 3FFF) Echo Start Register Number



Word 2 XXXX hex (XX = 1 ... FF) Echo Message Number

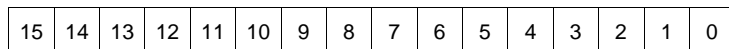


Word 3 XXXX hex



●
●
●

Word 11 XXXX hex Module Status



Note: Word 3 through Word 10 for Command 2 returns a 0.

Command 3 - GET DATA (Module to Controller)

Overview

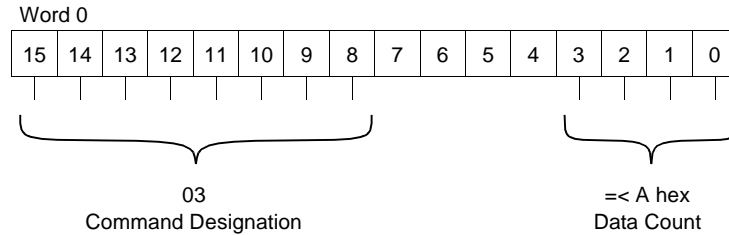
The GET DATA command reads up to 10 words/registers of data from the module starting at the start register number provided in Command Word 1. The data count provided in Command Word 0 determines the number of words to read. The data is returned in Response Words 2 through 11.

Note: If there is an error status to be reported (and is not a command syntax error) and the command requests 10 registers of data, the module will return only 9 words of data and use Response Word 11 for the module status. The Status Word Data bit will be set if Response Word 11 is the module status.

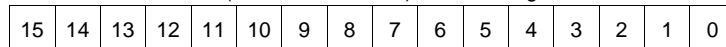
This command is executed continuously until Command Word 0 changes to a command other than GET DATA.

Command Structure

Command Structure for Command 3



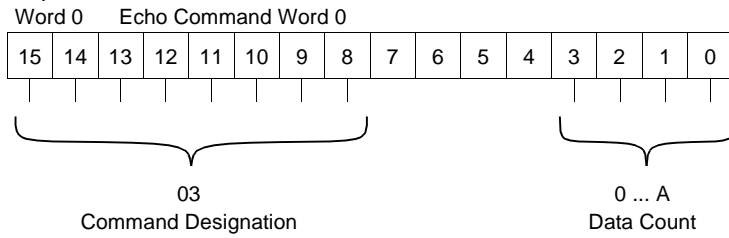
Word 1 XXXX hex (XXXX = 0 ... 3FFF) Start Register Number



Note: Word 2 through Word 11 for Command 3 are not used.

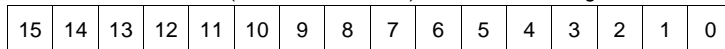
Response Structure

Response Structure for Command 3

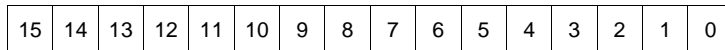


Note: Bit 15 is the Status Word Valid bit.

Word 1 XXXX hex (XXXX = 0 ... 3FFF) Echo Start Register Number

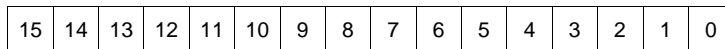


Word 2 XXXX hex Data Word 1



•
•
•

Word 11 XXXX hex Module Status or Data Word 10



Command 4 - PUT DATA (Controller to Module)

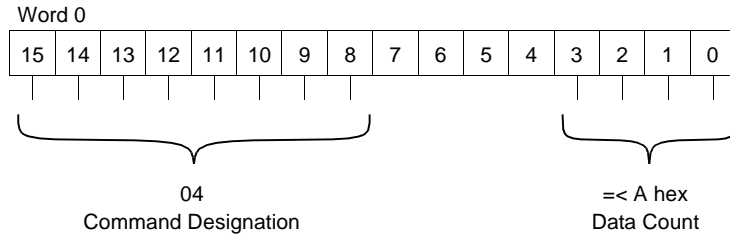
Overview

The PUT DATA command writes up to 10 words/registers of data to the module starting at the start register number provided in Command Word 1. The data is sent in Command Words 2 through 11.

This command is executed continuously until Command Word 0 changes to a command other than GET DATA.

Command Structure

Command Structure for Command 4



Word 1 XXXX hex (XXXX = 0 ... 3FFF) Start Register Number

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

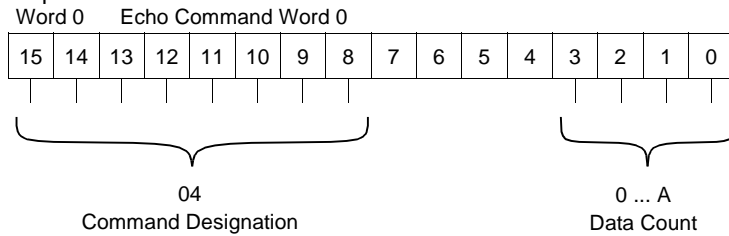
Word 2 XXXX hex Data Word 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

•
•
•

Word 11 XXXX hex Data Word 10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Response Structure
Response Structure for Command 4


Note: Bit 15 is the Status Word Valid bit.

Word 1 XXXX hex (XXXX = 0 ... 3FFF) Echo Start Register Number

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 2 XXXX hex

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

•
•
•

Word 11 XXXX hex Module Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Word 2 through Word 10 for Command 4 returns a 0.

Command 5 - GET TOD (Time of Day)

Overview

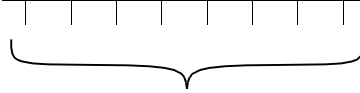
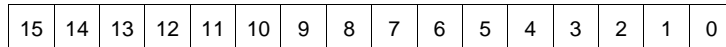
The GET TOD command reads the modules TOD clock and returns the time of day and the date in the Response Words 1 to 7. The format for the time of day and date is identical to that used by the PLC time/date registers.

This command is executed continuously without the need for changing any of the command words.

Command Structure

Command Structure for Command 5

Word 0



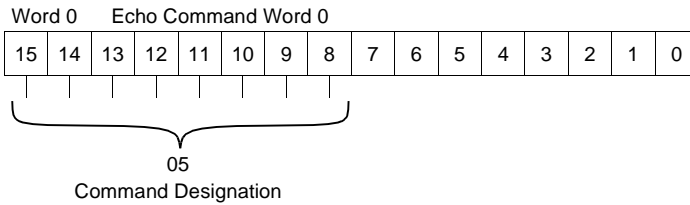
05

Command Designation

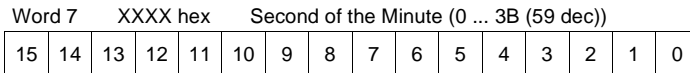
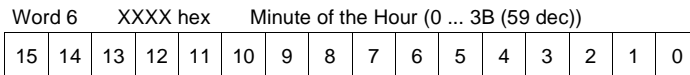
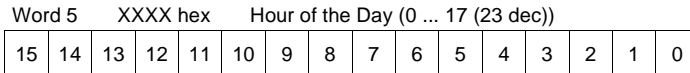
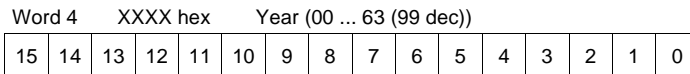
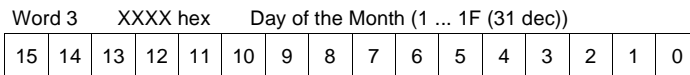
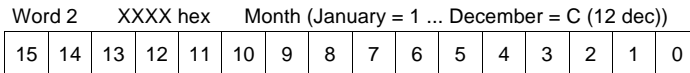
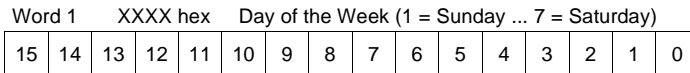
Note: Word 1 through Word 11 for Command 5 are not used.

Response Structure

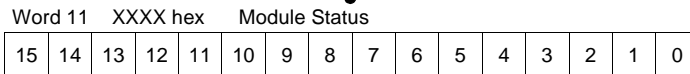
Response Structure for Command 5



Note: Bit 15 is the Status Word Valid bit.



•
•
•



Note: Word 8 through Word 10 for Command 5 returns a 0.

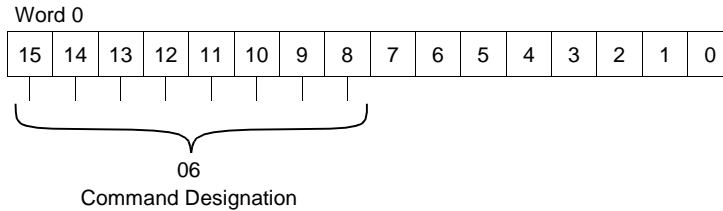
Command 6 - SET TOD (Time of Day)

Overview

The SET TOD command loads the modules TOD clock with the time of day and the date as provided in the Command Words 1 to 7. The format for the time of day and date is identical to that used by the PLC time/date registers.

Note: To synchronize the module's and PLC's TOD clocks, do a block move of the PLC's seven time/date registers to Command Words 1 to 7 and set Command Word 0 to 0600 hex.

This command is executed only the first time it is received. To execute the command again, one of the Command Words, 0 to 7, needs to be changed. This is done so that the same time does not get continuously loaded until Command Word 0 changes to a command other than SET TOD.

**Command
Structure****Command Structure for Command 6**

Note: Bit 15 is the Status Word Valid bit.

Word 1 XXXX hex Day of the Week (1 = Sunday ... 7 = Saturday)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 2 XXXX hex Month (January = 1 ... December = C (12 dec))

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 3 XXXX hex Day of the Month (1 ... 1F (31 dec))

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 4 XXXX hex Year (00 ... 63 (99 dec))

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 5 XXXX hex Hour of the Day (0 ... 17 (23 dec))

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 6 XXXX hex Minute of the Hour (0 ... 3B (59 dec))

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 7 XXXX hex Second of the Minute (0 ... 3B (59 dec))

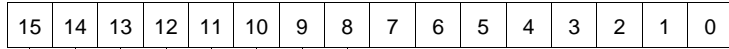
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Word 8 through Word 11 for Command 6 are not used.

Response Structure

Response Structure for Command 6

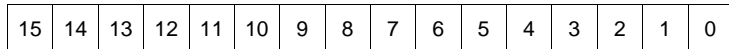
Word 0 Echo Command Word 0



06
Command Designation

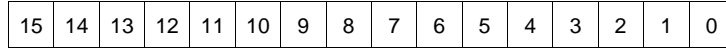
Note: Bit 15 is the Status Word Valid bit.

Word 1 XXXX hex



●
●
●

Word 11 XXXX hex Module Status



Note: Word 1 through Word 10 for Command 6 returns a 0.

Command 7 - SET MEMORY REGISTERS

Overview

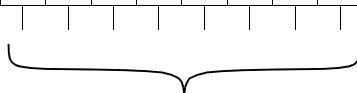
The SET MEMORY REGISTERS command sets module registers to the value provided in Command Word 3. The registers set are designated by: the start register number and the end register number. All registers from the start register up to and including the end register number are set to the value provided.

Command Structure

Command Structure for Command 7

Word 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---



07

Command Designation

Word 1 XXXX hex (XXXX = 0 ... 3FFF) Start Register Number

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 2 XXXX hex (XXXX = 0 ... 3FFF) End Register Number

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

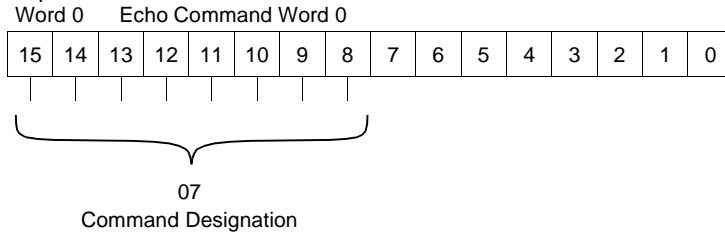
Word 3 XXXX hex Value to set in Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

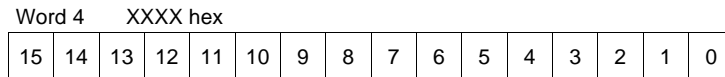
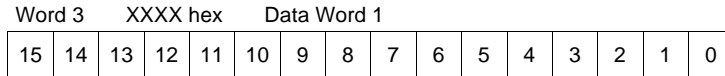
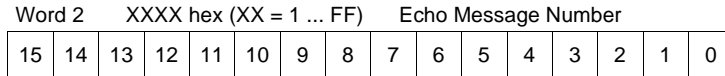
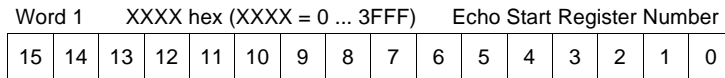
Note: Word 4 through Word 11 for Command 7 are not used.

Response Structure

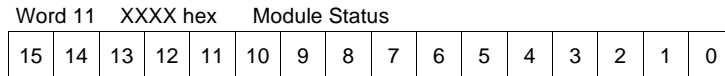
Response Structure for Command 7



Note: Bit 15 is the Status Word Valid bit.



•
•
•



Note: Word 4 through Word 10 for Command 7 returns a 0.

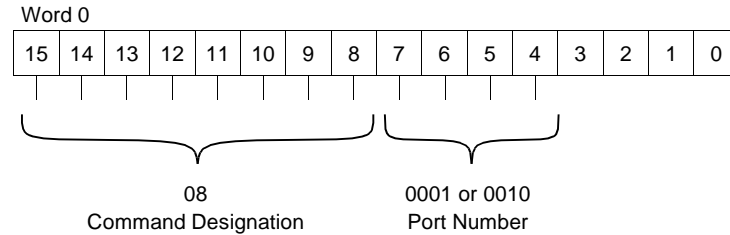
Command 8 - FLUSH BUFFER

Overview

The FLUSH BUFFER command flushes the input buffer for the serial port number provided in the command word. The output buffer is not affected by this command.

Command Structure

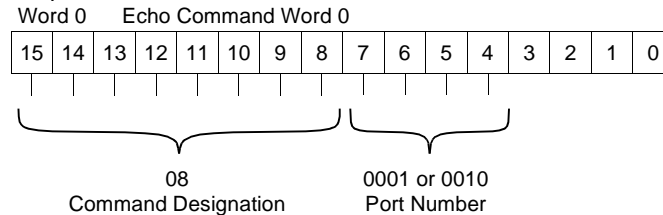
Command Structure for Command 8



Note: Word 1 through Word 11 for Command 8 are not used.

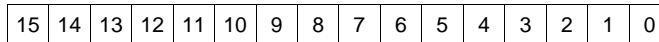
Response Structure

Response Structure for Command 8



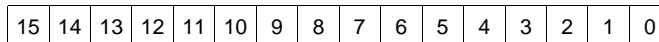
Note: Bit 15 is the Status Word Valid bit.

Word 1 XXXX hex



•
•
•

Word 11 XXXX hex Module Status



Note: Word 3 through Word 10 for Command 8 returns a 0.

Command 9 - ABORT

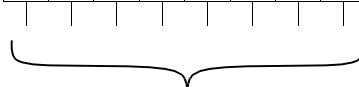
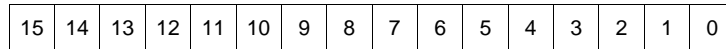
Overview

The ABORT command aborts a running READ or WRITE ASCII MESSAGE and the module is no longer in a busy status. The serial port buffers for the module are not affected by this command, only the message is currently running.

Command Structure

Command Structure for Command 9

Word 0



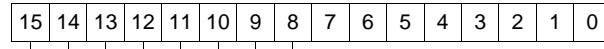
09
Command Designation

Note: Word 1 through Word 11 for Command 9 are not used.

Response Structure

Response Structure for Command 9

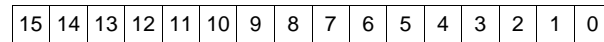
Word 0 Echo Command Word 0



09
Command Designation

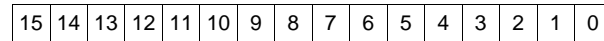
Note: Bit 15 is the Status Word Valid bit.

Word 1 XXXX hex



•
•
•

Word 11 XXXX hex Module Status



Note: Word 3 through Word 10 for Command 9 returns a 0.

Command A - GET BUFFER STATUS

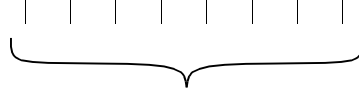
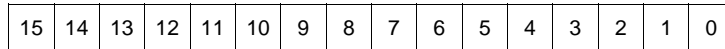
Overview

The GET BUFFER STATUS command reads the number of characters in the input buffer for each port. The range of characters is 1 ... 255.

Command Structure

Command Structure for Command A

Word 0



0A
Command Designation

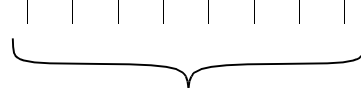
Note: Word 1 through Word 11 for Command A are not used.

Response Structure

Response Structure for Command A

Word 0 Echo Command Word 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---



0A
Command Designation

Note: Bit 15 is the Status Word Valid bit.

Word 1 Port 1 Buffer Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 2 Port 2 Buffer Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Word 3 XXXX hex

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

•
•
•

Word 11 XXXX hex Module Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Word 3 through Word 10 for Command A returns a 0.

Response Structure for Illegal Commands

Response Structure

Response Structure for Illegal Commands

Word 0 Echo Command Word 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Bit 15 is the Status Word Valid bit.



Word 11 XXXX hex Module Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Note: Word 1 through Word 10 returns a 0.
--

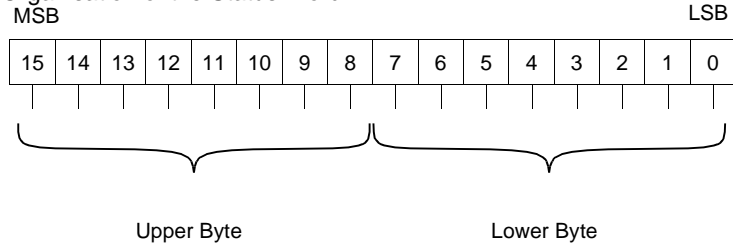
Module Status Word (Word 11)

Overview

The Module Status Word (Word 11 in the response structure) contains valid module status information when bit 15 of Word 0 (in the response structure) is set. The state of this bit can be used to distinguish whether Word 11 in the response structure is being used for data or status.

Organisation of the Status Word

Organisation of the Status Word:



Note: During normal operation, module status information is especially important when Word 11 is used for Module Status or Data Returned in the READ ASCII MESSAGE or GET DATA commands.

Content of the Status Word

Low Byte

Bit from Low Byte								Low Byte (Hex)	Description
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	1	0001	Busy; command running on module
0	0	0	0	0	0	1	0	0002	Invalid message data during command run
0	0	0	1	0	0	0	0	0100	Register end during command run
0	0	1	0	0	0	0	0	0200	Serial buffer overrun error
0	1	0	0	0	0	0	0	0400	Checksum error on message in storage area see upper byte for message number
1	0	0	0	0	0	0	0	8000	Error; see upper byte for message number

High Byte

Bit from High Byte								High Byte (Hex)	Description
15	14	13	12	11	10	9	8		
0	0	0	0	0	0	0	1	0001	Invalid user logic parameter
0	0	0	0	0	0	1	0	0002	Invalid user logic command
0	0	0	1	0	0	0	0	0100	Count out of range
0	0	0	1	0	0	0	1	0101	Starting register out of range
0	0	0	1	0	0	1	0	0102	Ending register out of range
0	0	0	1	0	0	1	1	0103	Invalid register number order (end before start)
0	0	0	1	0	1	0	0	0104	Invalid serial port number requested
0	0	0	1	0	1	0	1	0105	Invalid message number requested
0	0	0	1	0	1	1	0	0106	Requested message number not programmed
0	0	0	1	0	1	1	1	0107	Requested message number in bad storage area
0	0	0	1	1	0	0	0	0108	Configuration parameter error
0	0	1	0	0	0	0	0	0200	Day of the week is incorrect

Reading Beyond Valid Register Range

Overview

If the start register number and the data count are valid, but some of the registers to access are beyond the valid register range, then only the data from the registers in the valid register range are read/written. The data count returned is the number of valid register data returned, and the error code 1280 Hex (End Register number in out of range) is returned in the Module Status Word.

Example

The following example tries to read 10 registers, using the GET command, from the ESI module starting at register 3FFA Hex:

User Logic command = 030A Hex

Start register = 3FFA Hex

Therefore, the data count is 10 and the 6 valid registers (3FFA, 3FFB, 3FFC, 3FFD, 3FFE, and 3FFF Hex) data are returned. The data count returned in the Command Word is 6 (8306 Hex).

The following data are assumed to be in the ESI Registers:

ESI Register	Content (Hex)
3FFA	1111
3FFB	2222
3FFC	3333
3FFD	4444
3FFE	5555
3FFF	6666

The following table shows the command sent to the ESI module and the response:

User Logic Command		User Logic Response	
Register	Content	Register	Content
4x+0	030A Hex	3x+0	8306 Hex
4x+1	3FFA Hex	3x+1	3FFA Hex
4x+2	0000 Hex	3x+2	1111 Hex
4x+3	0000 Hex	3x+3	2222 Hex
4x+4	0000 Hex	3x+4	3333 Hex
4x+5	0000 Hex	3x+5	4444 Hex
4x+6	0000 Hex	3x+6	5555 Hex
4x+7	0000 Hex	3x+7	6666 Hex
4x+8	0000 Hex	3x+8	0000 Hex
4x+9	0000 Hex	3x+9	0000 Hex
4x+10	0000 Hex	3x+10	0000 Hex
4x+11	0000 Hex	3x+11	1280 Hex

ESI Loadable



Overview

Purpose

The ladder logic instructions for the ESI module are optional loadable instructions that can be used in a Quantum controller system to support operations using an ESI module. The controller can use the ESI Instruction to invoke the module. The power of the loadable is its ability to cause a sequence of commands over one or more logic events.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	96
Representation	97
Parameter Description	98
Run Time Errors	101
READ ASCII Message (Subfunction 1)	102
WRITE ASCII Message (Subfunction 2)	106
GET DATA (Subfunction 3)	107
PUT DATA (Subfunction 4)	109

Short Description

Function Description

Note: This instruction is only available if you have unpacked and installed the DX Loadables. For further information, see Installation of DX Loadable in Concept Block Library LL984 (840 USE 506 00, Version 2.6).

The instruction for the ESI module 140 ESI 062 10 are optional loadable instructions that can be used in a Quantum controller system to support operations using a ESI module. The controller can use the ESI instruction to invoke the module. The power of the loadable is its ability to cause a sequence of commands over one or more logic scans.

With the ESI instruction, the controller can invoke the ESI module to:

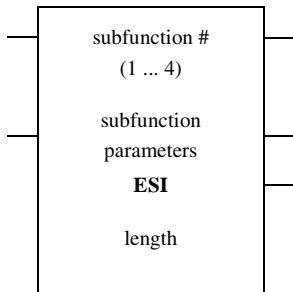
- Read an ASCII message from a serial port on the ESI module, then perform a sequence of GET DATA transfers from the module to the controller.
- Write an ASCII message to a serial port on the ESI module after having performed a sequence of PUT DATA transfers to the variable data registers in the module.
- Perform a sequence of GET DATA transfers (up to 16 384 registers of data from the ESI module to the controller); one Get Data transfer will move up to 10 data registers each time the instruction is solved.
- Perform a sequence of PUT DATA (up to 16 384 registers of data to the ESI module from the controller). One PUT DATA transfer moves up to 10 registers of data each time the instruction is solved.
- Abort the ESI loadable command sequence running.

Note: After placing the ESI instruction in your ladder diagram you must enter the top, middle and bottom parameters. Proceed by double clicking on the instruction. This action produces a form for the entry of the 3 parameters. This parameter must be completed to enable the DX zoom function in the `Edit` menu pull-down.

Representation

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
	0x, 1x	None	ON = enables the subfunction
	0x, 1x	None	Abort current message
(top node)	4x	INT, UINT, WORD	Number of possible subfunction, range 1 ... 4
(middle node)	4x	INT, UINT, WORD	First of eighteen contiguous 4x holding registers which contain the subfunction parameters
bottom node		INT, UINT	Number of subfunction parameter registers, i.e. the length of the table in the middle node
Top output	0x	None	Echoes state of the top input
	0x	None	ON = operation done
	0x	None	ON = error detected

Parameter Description

Top Input

When the input to the top node is powered ON, it enables the ESI instruction and starts executing the command indicated by the subfunction code in the top node.

Middle Input

When the input to the middle node is powered ON, an Abort command is issued. If a message is running when the ABORT command is received, the instruction will complete; if a data transfer is in process when the ABORT command is received, the transfer will stop and the instruction will complete.

**Subfunction #
(Top Node)**

The top node may contain either a 4x register or an integer. The integer or the value in the register must be in the range 1 ... 4.

It represents one of four possible subfunction command sequences to be executed by the instruction:

Subfunction	Command Sequence
1	One command <i>READ ASCII Message, p. 102</i> followed by multiple GET DATA commands
2	Multiple PUT DATA commands followed by one command <i>WRITE ASCII Message, p. 106</i>
3	Zero or more commands <i>GET DATA, p. 107</i>
4	Zero or more commands <i>PUT DATA, p. 109</i>

Note: A fifth command, ABORT ASCII Message, can be initiated by enabling the middle input to the ESI instruction.

**Subfunction
Parameters
(Middle Node)**

The first of eighteen contiguous 4x registers is entered in the middle node. The remaining seventeen registers are implied.

The following subfunction parameters are available:

Register	Parameter	Contents
Displayed	ESI status register	Returned error codes
First implied	Address of the first 4x register in the command structure	Register address minus the leading 4 and any leading zeros, as specified in the I/O Map (e.g., 1 represents register 400001)
Second implied	Address of the first 3x register in the command structure	Register address minus the leading 3 and any leading zeros, as specified in the I/O Map (e.g., 7 represents register 300007)
Third implied	Address of the first 4x register in the controller's data register area	Register address minus the leading 4 and any leading zeros (e.g., 100 representing register 400100)
Fourth implied	Address of the first 3x register in the controller's data register area	Register address minus the leading 3 and any leading zeros (e.g., 1000 representing register 301000)
Fifth implied	Starting register for data register area in module	Number in the range 0 ... 3FFF hex
Sixth implied	Data transfer count	Number in the range 0 ... 4000 hex
Seventh implied	ESI timeout value, in 100 ms increments	Number in the range 0 ... FFFF hex, where 0 means no timeout
Eighth implied	ASCII message number	Number in the range 1 ... 255 dec
Ninth implied	ASCII port number	1 or 2
Note: The registers below are internally used by the ESI loadable. Do not write registers while the ESI loadable is running. For best use, initialize these registers to 0 (zero) when the loadable is inserted into logic.		
10th implied	ESI loadable previous scan power in state	
11th implied	Data left to transfer	
12th implied	Current ASCII module command running	
13th implied	ESI loadable sequence number	
14th implied	ESI loadable flags	
15th implied	ESI loadable timeout value (MSW)	
16th implied	ESI loadable timeout value (LSW)	
17th implied	Parameter Table Checksum generated by ESI loadable	

Note: Once power has been applied to the top input, the ESI loadable starts running. Until the ESI loadable compiles (successfully or in error), the subfunction parameters should not be modified. If the ESI loadable detects a change, the loadable will compile in error (Parameter Table Checksum Error (See *Run Time Errors*, p. 101)).

**Length
(Bottom Node)**

The bottom node contains the length of the table in the middle node, i.e., the number of subfunction parameter registers. For READ/ WRITE operations, the length must be 10 registers. For PUT/GET operations, the required length is eight registers; 10 may be specified and the last two registers will be unused.

Ouputs

Note: NSUP must be loaded before ESI in order for the loadable to work properly. If ESI is loaded before NSUP or ESI is loaded alone, all three outputs will be turned ON.

Middle Output

The middle output goes ON for one scan when the subfunction operation specified in the top node is completed, timed out, or aborted

Bottom Output

The bottom output goes ON for one scan if an error has been detected. Error checking is the first thing that is performed on the instruction when it is enabled. For more details see error checking (See *Run Time Errors*, p. 101).

Run Time Errors

Run Time Errors The command sequence executed by the ESI module (specified by the subfunction value (See *Subfunction # (Top Node)*, p. 98) in the top node of the ESI instruction) needs to go through a series of error checking routines before the actual command execution begins. If an error is detected, a message is posted in the register displayed in the middle node.

The following table lists possible error message codes and their meanings:

Error Code (dec)	Meaning
0001	Unknown subfunction specified in the top node
0010	ESI instruction has timed out (exceeded the time specified in the eighth register of the subfunction parameter table (See <i>Subfunction Parameters (Middle Node)</i> , p. 99)
0101	Error in the READ ASCII Message sequence
0102	Error in the WRITE ASCII Message sequence
0103	Error in the GET DATA sequence
0104	Error in the PUT DATA sequence
1000	<i>Length (Bottom Node)</i> , p. 100 is too small
1001	Nonzero value in both the 4x and 3x data offset parameters
1002	Zero value in both the 4x and 3x data offset parameters
1003	4x or 3x data offset parameter out of range
1004	4x or 3x data offset plus transfer count out of range
1005	3x data offset parameter set for GET DATA
1006	Parameter Table Checksum error
1101	Output registers from the offset parameter out of range
1102	Input registers from the offset parameter out of range
2001	Error reported from the ESI module

Once the parameter error checking has completed without finding an error, the ESI module begins to execute the command sequence.

READ ASCII Message (Subfunction 1)

READ ASCII Message

A READ ASCII command causes the ESI module to read incoming data from one of its serial ports and store the data in internal variable data registers. The serial port number is specified in the tenth (ninth implied) register of the subfunction parameters table. The ASCII message number to be read is specified in the ninth (eighth implied) register of the subfunction parameters table (See *Subfunction Parameters (Middle Node)*, p. 99). The received data is stored in the 16K variable data space in user-programmed formats.

When the top node of the ESI instruction is 1, the controller invokes the module and causes it to execute one READ ASCII command followed by a sequence of GET DATA commands (transferring up to 16,384 registers of data) from the module to the controller.

Command Structure

Command Structure

Word	Content (hex)	Meaning
0	01PD	P = port number (1 or 2); D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2	00xx	Message number, where xx is in the range 1 ... FF (1 ... 255 dec)
3 ... 11	Not used	

Response Structure

Command Structure

Word	Content (hex)	Meaning
0	01PD	Echoes command word 0
1	xxxx	Echoes starting register number from Command Word 1
2	00xx	Echoes message number from Command Word 2
3	xxxx	Data word 1
4	xxxx	Data word 2
...
11	xxxx	Module status or data word 9

A Comparative READ ASCII Message/Put Data Example

Below is an example of how an ESI loadable instruction can simplify your logic programming task in an ASCII read application. Assume that the 12-point bidirectional ESI module has been I/O mapped to 400001 ... 400012 output registers and 300001 ... 300012 input registers. We want to read ASCII message #10 from port 1, then transfer four words of data to registers 400501 ... 400504 in the controller.

Parameterizing of the ESI instruction:

#0001
401000
ESI
#0018

The subfunction parameter table begins at register 401000 . Enter the following parameters in the table:

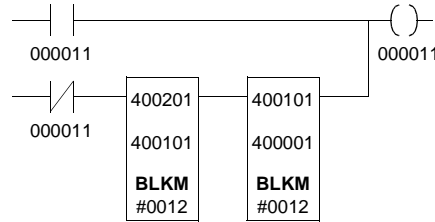
Register	Parameter Value	Description
401000	nnnn	ESI status register
401001	1	I/O mapped output starting register (400001)
401002	1	I/O mapped input starting register (300001)
401003	501	Starting register for the data transfer (400501)
401004	0	No 3x starting register for the data transfer
401005	100	Module start register
401006	4	Number of registers to transfer
401007	600	timeout = 60 s
401008	10	ASCII message number
401009	1	ASCII port number
401010-17	N/A	Internal loadable variables

With these parameters entered to the table, the ESI instruction will handle the read and data transfers automatically in one scan.

Read and Data Transfers without ESI Instruction

The same task could be accomplished in ladder logic **without** the ESI loadable, but it would require the following three networks to set up the command and transfer parameters, then copy the data. Registers 400101 ... 400112 are used as workspace for the output values. Registers 400201 ... 400212 are initial READ ASCII Message command values. Registers 400501 ... 400504 are the data space for the received data from the module.

First Network

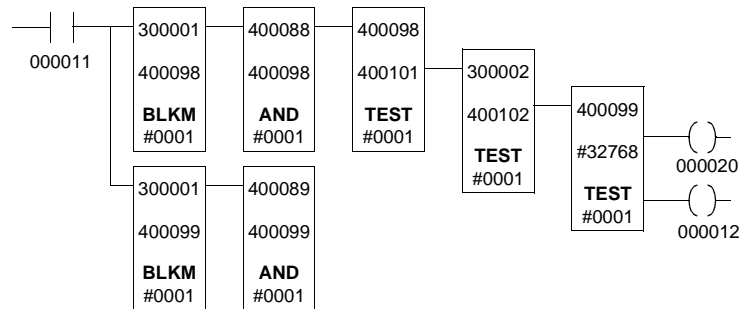


Contents of registers

Register	Value (hex)	Description
400201	0114	READ ASCII Message command, Port 1, Four registers
400202	0064	Module's starting register
400203	nnnn	Not valid: data word 1
...
400212	nnnn	Not valid: data word 10

The first network starts up the READ ASCII Message command by turning ON coil 000011 forever. It moves the READ ASCII Message command into the workspace, then moves the workspace to the output registers for the module.

Second Network



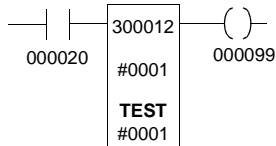
Contents of registers

Register	Value (hex)	Description
400098	nnnn	Workspace for response word
400099	nnnn	Workspace for response word
400088	7FFF	Response word mask
400089	8000	Status word valid bit mask

As long as coil 000011 is ON, READ ASCII Message response Word 0 in the input register is tested to make sure it is the same as command Word 0 in the workspace. This is done by ANDing response Word 0 in the input register with 7FFF hex to get rid of the Status Word Valid bit (bit 15) in Response Word 0.

The module start register in the input register is also tested against the module start register in the workspace to make sure that they are the same.

If both these tests show matches, test the Status Word Valid bit in response Word 0. To do this, AND response Word 0 in the input register with 8000 hex to get rid of the echoed command word 0 information. If the ANDed result equals the Status Word Valid bit, coil 000020 is turned ON indicating an error and/or status in the Module Status Word. If the ANDed result is not the status word valid bit, coil 000012 is turned ON indicating that the message is done and that you can start another command in the module.

Third Network

If coil 000020 is ON, this third network will test the Module Status Word for busy status. If the module is busy, do nothing. If the Module Status Word is greater than 1 (busy), a detected error has been logged in the high byte and coil 000099 will be turned ON. At this point, you need to determine what the error is by using some error-handling logic that you have developed.

WRITE ASCII Message (Subfunction 2)

WRITE ASCII Message

In a WRITE ASCII Message command, the ESI module writes an ASCII message to one of its serial ports. The serial port number is specified in the tenth (ninth implied) register of the subfunction parameters table (See *Subfunction Parameters (Middle Node)*, p. 99). The ASCII message number to be written is specified in the ninth (eighth implied) register of the subfunction parameters table.

When the top node of the ESI instruction is 2, the controller invokes the module and causes it to execute one Write ASCII command. Before starting the WRITE command, subfunction 2 executes a sequence of PUT DATA transfers (transferring up to 16 384 registers of data) from the controller to the module.

Command Structure

Command Structure

Word	Content (hex)	Meaning
0	02PD	P = port number (1 or 2); D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2	00xx	Message number, where xx is in the range 1 ... FF (1 ... 255 dec)
3	xxxx	Data word 1
4	xxxx	Data word 2
...
11	xxxx	Data word 9

Response Structure

Response Structure

Word	Content (hex)	Meaning
0	02PD	Echoes command word 0
1	xxxx	Echoes starting register number from command word 1
2	00xx	Echoes message number from command word 2
3	0000	Returns a zero
...
10	0000	Returns a zero
11	xxxx	Module status

GET DATA (Subfunction 3)

GET DATA

A GET DATA command transfers up to 10 registers of data from the ESI module to the controller each time the ESI instruction is solved in ladder logic. The total number of words to be read is specified in Word 0 of the GET DATA command structure (the data count). The data is returned in increments of 10 in Words 2 ... 11 in the GET DATA response structure.

If a sequence of GET DATA commands is being executed in conjunction with a READ ASCII Message command (via subfunction 1), up to nine registers are transferred when the instruction is solved the first time. Additional data are returned in groups of ten registers on subsequent solves of the instruction until all the data has been transferred.

If there is an error condition to be reported (other than a command syntax error), it is reported in Word 11 in the GET DATA response structure. If the command has requested 10 registers and the error needs to be reported, only nine registers of data will be returned in Words 2 ... 10, and Word 11 will be used for error status.

Note: If the data count and starting register number that you specify are valid but some of the registers to be read are beyond the valid register range, only data from the registers in the valid range will be read. The data count returned in Word 0 of the response structure will reflect the number of valid data registers returned, and an error code (1280 hex) will be returned in the Module Status Word (Word 11 in the response table).

Command Structure

Command Structure

Word	Content (hex)	Meaning
0	030D	D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2 ... 11	Not used	

Response Structure

Response Structure

Word	Content (hex)	Meaning
0	030D	Echoes command word 0
1	xxxx	Echoes starting register number from command word 1
2	xxxx	Data word 1
3	xxxx	Data word 2
...
11	xxxx	Module status or data word 10

PUT DATA (Subfunction 4)

PUT DATA

A PUT DATA command writes up to 10 registers of data to the ESI module from the controller each time the ESI instruction is solved in ladder logic. The total number of words to be written is specified in Word 0 of the PUT DATA command structure (the data count).

The data is returned in increments of 10 in words 2 ... 11 in the PUT DATA command structure. The command is executed sequentially until command word 0 changes to another command other than PUT DATA (040D hex).

Note: If the data count and starting register number that you specify are valid but some of the registers to be written are beyond the valid register range, only data from the registers in the valid range will be written. The data count returned in Word 0 of the response structure will reflect the number of valid data registers returned, and an error code (1280 hex) will be returned in the Module Status Word (Word 11 in the response table).

Command Structure

Command Structure

Word	Content (hex)	Meaning
0	040D	D = data count
1	xxxx	Starting register number, in the range 0 ... 3FFF
2	xxxx	Data word 1
3	xxxx	Data word 2
...
11	xxxx	Data word 10

Response Structure

Response Structure

Word	Content (hex)	Meaning
0	040D	Echoes command word 0
1	xxxx	Echoes starting register number from command word 1
2	0000	Returns a zero
...
10	0000	Returns a zero
11	xxxx	Module status

**A Comparative
PUT DATA
Example**

Below is an example of how an ESI loadable instruction can simplify your logic programming task in a PUT DATA application. Assume that the 12-point bidirectional ESI 062 module has been I/O mapped to 400001 ... 400012 output registers and 300001 ... 300012 input registers. We want to put 30 controller data registers, starting at register 400501, to the ESI module starting at location 100.

Parameterizing of the ESI instruction:

#0004
401000
ESI
#0018

The subfunction parameter table begins at register 401000 . Enter the following parameters in the table:

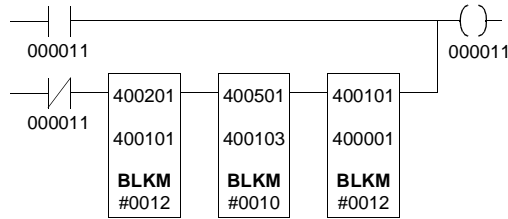
Register	Parameter Value	Description
401000	nnnn	ESI status register
401001	1	I/O mapped output starting register (400001)
401002	1	I/O mapped input starting register (300001)
401003	501	Starting register for the data transfer (400501)
401004	0	No 3x starting register for the data transfer
401005	100	Module start register
401006	30	Number of registers to transfer
401007	0	timeout = never
401008	N/A	ASCII message number
401009	N/A	ASCII port number
401009	N/A	Internal loadable variables

With these parameters entered to the table, the ESI instruction will handle the data transfers automatically over three ESI logic solves.

Handling of Data Transfer without ESI Instruction

The same task could be accomplished in ladder logic **without** the ESI loadable, but it would require the following four networks to set up the command and transfer parameters, then copy data multiple times until the operation is complete. Registers 400101 ... 400112 are used as workspace for the output values. Registers 400201 ... 400212 are initial PUT DATA command values. Registers 400501 ... 400530 are the data registers to be sent to the module.

First Network - Command Register Network

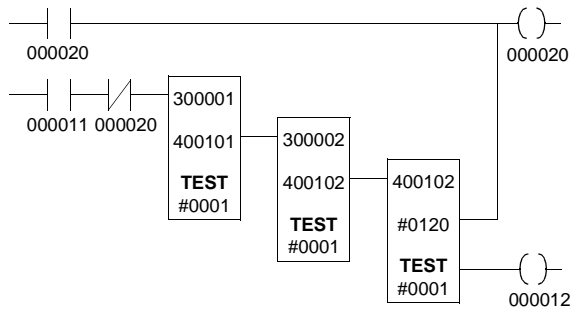


Contents of registers

Register	Value (hex)	Description
400201	040A	PUT DATA command, 10 registers
400202	0064	Module's starting register
400203	nnnn	Not valid: data word 1
...
400212	nnnn	Not valid: data word 10

The first network starts up the transfer of the first 10 registers by turning ON coil 000011 forever. It moves the initial PUT DATA command into the workspace, moves the first 10 registers (400501 ... 400510) into the workspace, and then moves the workspace to the output registers for the module.

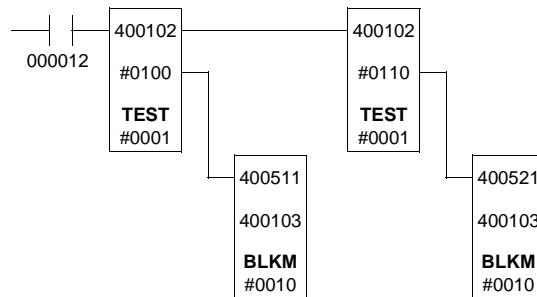
Second Network - Command Register Network



As long as coil 000011 is ON and coil 000020 is OFF, PUT DATA response word 0 in the input register is tested to make sure it is the same as the command word in the workspace. The module start register in the input register is also tested to make sure it is the same as the module start register in the workspace.

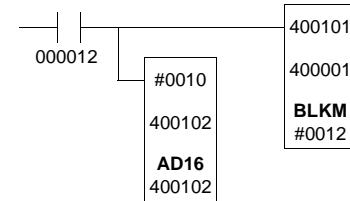
If both these tests show matches, the current module start register is tested against what would be the module start register of the last PUT DATA command for this transfer. If the test shows that the current module start register is greater than or equal to the last PUT DATA command, coil 000020 goes ON indicating that the transfer is done. If the test shows that the current module start register is less than the last PUT DATA command, coil 000012 indicating that the next 10 registers should be transferred.

Third Network - Command Register Network



As long as coil 000012 is ON, there is more data to be transferred. The module start register needs to be tested from the last command solve to determine which set of 10 registers to transfer next. For example, if the last command started with module register 400110, then the module start register for this command is 400120.

Fourth Network - Command Register Network



As long as coil 000012 is ON, add 10 to the module start register value in the workspace and move the workspace to the output registers for the module to start the next transfer of 10 registers.

Appendices



At a Glance

Overview

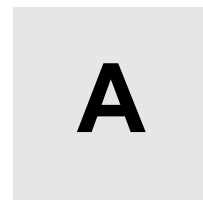
The Appendices provide additional information of general nature.

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Character Set	115

Character Set



ASCII Character Set

Non Printable ASCII Characters

The following table defines the ASCII set in decimal, hexadecimal, character, and control character values.

Decimal	Octal	Hexa-decimal	Character	Character Control
0	00	00	NUL	NULL
1	01	01	SOH	START OF HEADING
2	02	02	STX	START OF TEXT
3	03	03	ETX	END OF TEXT
4	04	04	EOT	END OF TRANSMISSION
5	05	05	ENQ	ENQUIRY
6	06	06	ACK	ACKNOWLEDGE
7	07	07	BEL	BEEP
8	10	08	BS	BACKSPACE
9	11	09	HT	HORIZONTAL TAB
10	12	0A	LF	LINE FEED
11	13	0B	VT	VERTICAL TAB (home)
12	14	0C	FF	FORM FEED
13	15	0D	CR	CARRIAGE RETURN
14	16	0E	SO	SHIFT OUT
15	17	0F	SI	SHIFT IN
16	20	10	DLE	DATALINK ESCAPE
17	21	11	DC1	DEVICE CONTROL ONE
18	22	12	DC2	DEVICE CONTROL TWO
19	23	13	DC3	DEVICE CONTROL THREE
20	24	14	DC4	DEVICE CONTROL FOUR
21	25	15	NAK	NEGATIVE ACKNOWLEDGE

Decimal	Octal	Hexa-decimal	Character	Character Control
22	26	16	SYN	SYNCHRONOUS IDLE
23	27	17	ETB	END OF TRANSMISSION BLOCK
24	30	18	CAN	CANCEL
25	31	19	EM	END OF MEDIUM
26	32	1A	SUB	SUBSTITUTE
27	33	1B	ESC	ESCAPE
28	34	1C	FS	FILE SEPARATOR (cursor right)
29	35	1D	GS	GROUP SEPARATOR (cursor left)
30	36	1E	RS	RECORD SEPARATOR (cursor up)
31	37	1F	US	UNIT SEPARATOR (cursor down)

**Printable ASCII
Characters**

The following table defines the ASCII set in decimal, hexadecimal and character.

Decimal	Octal	Hexa- decimal	Character	Decimal	Octal	Hexa- decimal	Character
32	40	20	SPACE	58	72	3A	:
33	41	21	!	59	73	3B	;
34	42	22	"	60	74	3C	<
35	43	23	#	61	75	3D	=
36	44	24	\$	62	76	3E	>
37	45	25	%	63	77	3F	?
38	46	26	&	64	100	40	@
39	47	27	'	65	101	41	A
40	50	28	(66	102	42	B
41	51	29)	67	103	43	C
42	52	2A	*	68	104	44	D
43	53	2B	+	69	105	45	E
44	54	2C	,	70	106	46	F
45	55	2D	-	71	107	47	G
46	56	2E	.	72	110	48	H
47	57	2F	/	73	111	49	I
48	60	30	0	74	112	4A	J
49	61	31	1	75	113	4B	K
50	62	32	2	76	114	4C	L
51	63	33	3	77	115	4D	M
52	64	34	4	78	116	4E	N
53	65	35	5	79	117	4F	O
54	66	36	6	80	120	50	P
55	67	37	7	81	121	51	Q
56	70	38	8	82	122	52	R
57	71	39	9	83	123	53	S

Printable ASCII Character Set (continued):

Decimal	Octal	Hexa-decimal	Character	Decimal	Octal	Hexa-decimal	Character
84	124	54	T	106	152	6A	j
85	125	55	U	107	153	6B	k
86	126	56	V	108	154	6C	l
87	127	57	W	109	155	6D	m
88	130	58	X	110	156	6E	n
89	131	59	Y	111	157	6F	o
90	132	5A	Z	112	160	70	p
91	133	5B	[113	161	71	q
92	134	5C	\	114	162	72	r
93	135	5D]	115	163	73	s
94	136	5E	^	116	164	74	t
95	137	5F	_	117	165	75	u
96	140	60	'	118	166	76	v
97	141	61	a	119	167	77	w
98	142	62	b	120	170	78	x
99	143	63	c	121	171	79	y
100	144	64	d	122	172	7A	z
101	145	65	e	123	173	7B	{
102	146	66	f	124	174	7C	
103	147	67	g	125	175	7D	}
104	150	68	h	126	176	7E	~
105	151	69	i	127	177	7F	

Index



A

ABORT, 86
Application Criteria, 22
ASCII Character Set, 115
ASCII Message Formats, 59

B

Block Diagramm, 25

C

Character Set, 115
Command
 ABORT, 86
 FLUSH BUFFER, 85
 GET BUFFER STATUS, 87
 GET DATA, 74
 GET TOD, 78
 Illegal Commands, 89
 NO OPERATION, 68
 PUT DATA, 76
 READ ASCII MESSAGE, 69
 SET MEMORY REGISTERS, 83
 SET TOD, 80
 WRITE ASCII MESSAGE, 71
Command Line Editor, 53
Command Structure, 67
Commands, 65
Configuration, 46
Configuration Editor, 54
Connectors, 39

Crash Codes, 38

E

Editor, 53
Error Indicators, 38
ESI hardware description, 28
External Connectors, 39

F

FLUSH BUFFER, 85
front panel connectors and switches
 front panel push button, 30
front panel LEDs
 blinking sequence, 29

G

GET BUFFER STATUS, 87
GET DATA, 74
GET TOD, 78

H

Hardware Description, 35

I

Illegal Commands, 89
Indicators, 37
Introduction, 19

Invalid Register Range, 92

L

LED, 37

M

Message Formats, 59

N

NO OPERATION, 68

P

PUT DATA, 76

R

READ ASCII MESSAGE, 69

S

SET MEMORY REGISTERS, 83

SET TOD, 80

Status Word, 90

Switches, 39

W

WRITE ASCII MESSAGE, 71