

**3100/3150-ROC**  
**Fisher ROC**  
**Communications**  
Revision 1.2  
March 8, 2001

**USER MANUAL**

ProSoft Technology, Inc.  
9801 Camino Media, Suite 105  
Bakersfield, CA 93311  
(661) 664-7208  
(661) 664-7233 (fax)  
E-mail address: [prosoft@prosoft-technology.com](mailto:prosoft@prosoft-technology.com)  
Web Site: <http://www.prosoft-technology.com>  
FTP Site: <ftp://ftp.prosoft-technology.com>

---

**Please Read This Notice**

Successful application of the ROC card requires a reasonable working knowledge of the Allen-Bradley PLC or SLC hardware and the application in which the combination is to be used. For this reason, it is important that those responsible for implementing the ROC satisfy themselves that the combination will meet the needs of the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

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

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

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

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

---

## **Quick Start Implementation Guide**

Integration of the ROC module into a PLC or SLC application is easier if a series of steps are followed. In order to assist the first time users of our products in getting operational quickly, we have come up with this step-by-step implementation guide.



### **First Time Users**

Although the following steps are to assist you in implementing the module, we recommend that you attempt to experiment with the example logic available off of our FTP site before laying out your application. This step will allow you to gain insight into how the module works prior to making decisions, which will impact the long-term success of the installation.

Starting with one of the ladder logic programs available for the ROC complete the following steps:

If hand entering the ladder logic by hand for the SLC, remember the following:

- Configure the slot as a 1746-BAS module in 5/02 mode
  - Be sure to enter the Transfer Enable and Done bits as shown in the example logic
- a) Edit the ladder logic provided on disk as needed for the application (See Section 3.0)
    - Verify rack and slot location in program
    - Modify ladder instruction addresses as needed
  - c) Setup the Communication Configuration parameters (See Section 4.2)
    - Determine each port's communication configuration requirements:
      - Master or Slave, Parity, Stop Bits, Baud Rate, RTS delay requirements
    - Identify memory mapping requirements
    - Set the Read Data, Write Data, and the Command Block Count parameters
    - Set the Slave and Master Error Table pointers are needed for the application
  - d) Setup the Command List if configuring a Master (See Section 4.4)
    - Be sure to review register map of slave device to build most effective memory map
  - e) Identify the module jumper requirements (See Appendix C)
  - f) Make up the communication cables (See Section 8). Make sure that no matter what type of connection is being made up that a jumper is in place to satisfy the CTS signal. Normally this signal will be jumpered to RTS.
  - g) Place processor into the run mode
  - h) Monitor the data table for the Master and Slave Error Status values (See Section 5.1.4)

## **Table of Contents**

1	Functional Overview.....	6
1.1	General.....	6
1.2	Hardware Overview.....	6
1.3	General Concepts .....	7
1.3.1	Module Power Up and Reset .....	8
1.3.2	Main Loop Logic.....	8
1.3.3	The Data Space in the module .....	9
1.3.4	The Backplane Data Transfer Process.....	10
1.3.5	Interlocking the Block Transfers .....	13
1.3.6	SLC Processor Configuration.....	14
1.4	Data Flow.....	14
1.4.1	General concepts.....	14
1.4.2	Reading data from the module .....	15
1.4.3	Writing data to the module .....	15
1.4.4	Master Port Driver.....	15
1.5	ROC Support of Fisher ROC Functionality .....	16
2	Getting Going - A Step by Step Approach.....	17
3	Ladder Logic Overview .....	18
3.1	Operational Overview.....	18
3.2	Ladder Logic .....	18
4	Writing to the Module.....	19
4.1	Block Transferring to the Module .....	19
4.2	Communications Configuration [ BTW Block ID 255 ].....	20
4.3	Writing Into Module Data Memory [ BTW Block ID Codes 0-79 ].....	25
4.3.1	Ladder Logic to Write Data to Module .....	25
4.3.2	Block Transfer Data Structure .....	26
4.4	Command List Configuration - Master Mode [ BTW Block ID Codes 80-99 ].....	27
4.4.1	Command List Ladder Logic.....	27
4.4.2	Command List Structure .....	28
4.4.3	Editing the Command List.....	30
4.5	Command Control Mode - Master Mode .....	30
4.5.1	The BTW Block Structure .....	30
4.5.2	Controlling the Commands.....	31
4.5.3	Example Command List.....	31
5	Reading from the Module .....	33
5.1	Transferring data from the module [ BTR Block ID 0 to 79 ].....	33
5.1.1	The Read Data Block Structure.....	33
5.1.2	Moving the data from the module to the processor.....	34
5.1.3	Ladder Logic to Read Module Data .....	34
5.1.4	Slave Error Code Table.....	35
5.1.5	Master Error Code Table.....	36
5.1.6	Error Status Codes .....	37
5.2	Decoding Command Done and Command Error Bits - Master Mode .....	38
5.2.1	The Block Structure .....	39
5.2.2	Ladder Logic .....	39
6	Fisher ROC Command Configuration.....	40
6.1	Fisher ROC Commands.....	40
6.1.1	Opcode 180 and 181 Examples .....	41
6.1.2	Point Type Support .....	42
6.2	Floating Point Support .....	42
6.3	Store And Forward.....	43
7	Diagnostics and Troubleshooting.....	44
7.1	3100 PLC Platform LED Indicators .....	44

## Table of Contents

---

7.2	3150 SLC Platform LED Indicators .....	45
7.3	Troubleshooting - General.....	45
8	Cable Connection Diagrams .....	48
A	Support, Service and Warranty.....	50
B	Product Specifications.....	52
C	Jumper Configurations .....	54
D	Product Revision History .....	56
E	Read, Write and Command Block Count Values usage.....	57
F	Example Ladder Logic .....	58

# 1 Functional Overview

This section is intended to give the reader an overview of the ROC module operating concepts. Details associated with the ladder logic and the data transfer across the backplane is covered in later sections and in the Appendix.

## 1.1 General

The ROC products are single slot rack resident modules, which have been designed to provide a tightly integrated Fisher ROC communication interface for the Allen-Bradley 1771 and 1746 I/O platforms. The product will support the following processors:

3100-ROC for 1771 Platform

- PLC 5 family
- PLC 2 family
- PLC 3 family

3150-ROC for 1746 Platform

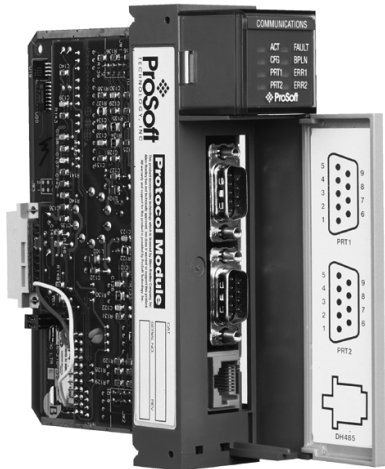
- SLC 5/02, 5/03, 5/04, 5/05

The module will work in the local rack with the processor or can be installed in a remote rack using Remote I/O communications to link the racks, in the case of the PLC, or can be placed in an extended rack in the case of the SLC.

The two forms in which the product is available are shown below:



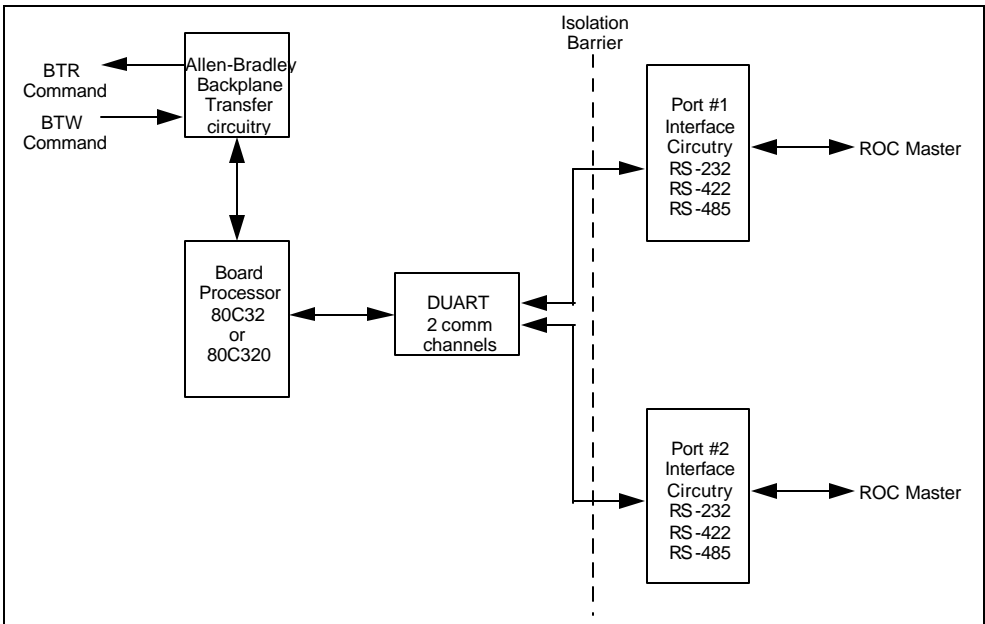
3100 Module  
1771 Platform



3150 Module  
1746 Platform

## 1.2 Hardware Overview

The design of the ROC module for the two hardware platforms is very similar. The following discussion, unless identified otherwise, will apply to both the 3100 and the 3150 platforms. The figure below shows the functional components on the modules.



Hardware layout diagram of 3100/3150 modules

The primary functional components on the boards are:

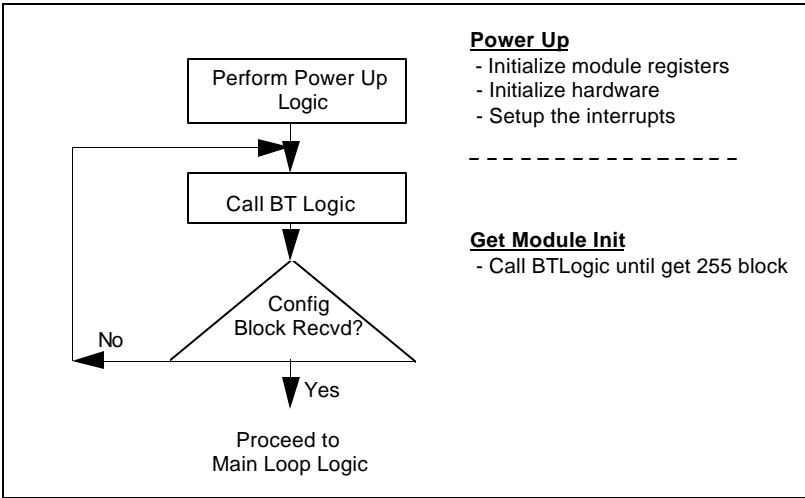
- ?? A microcontroller responsible for the overall operation of the board, including:
  - Backplane communications with Allen-Bradley processor
  - Transferring data from module to PLC
  - Accepting data from PLC into the module
  - Servicing DUART communications
  - LED Status Indications
- ?? An Allen-Bradley backplane chipset responsible for servicing the communications between the module and the A-B processor. The chipset contains proprietary technology licensed from A-B designed to:
  - In the case of the PLC the chipset has been designed to communicate with backplane using the Block Transfer commands, transferring 64 words at a time
  - In the case of the SLC, the chipset has been designed to communicate with the backplane using the M0/M1 files. As there is no real Block Transfer functionality in the SLC, we have implemented a form of block transfer using the I/O table to control the handshaking between the module and the processor. Up to 64 words may be transferred at a time. Shown below, presuming the module is in slot 1, are these bits:
    - I:1/0    Transfer Enable  
This bit is set by the module and used by the ladder logic to enable the movement of data over the backplane
    - O:1/0    Transfer Done  
This bit is set by the ladder logic to communicate to the module that the ladder has completed the data transfer
- ?? The port interface circuitry providing the physical interface to the real world. The ports and the interface circuitry are optically isolated from the rest of the card, and therefore the backplane, providing a high level of protection to the A-B processor. Both ports are capable of supporting:
  - RS-232
  - RS-422, also called a 4 wire connection
  - RS-485, also called a 2 wire connection

1.3 General Concepts

The following discussion covers several concepts, which are key to understanding the operation of the ProSoft module.

### 1.3.1 Module Power Up and Reset

On power up, or after pressing the reset pushbutton (3100 only), the module begins performing its logical functions. These functions shown in the flow chart



included here, include:

1. Initialize hardware
  - Initialize the backplane
  - Initialize the DUART
2. Initialize Module registers
  - Clear the Module Data Block
  - Clear Command List
  - Clear Error Status Tables
  - Preset constants

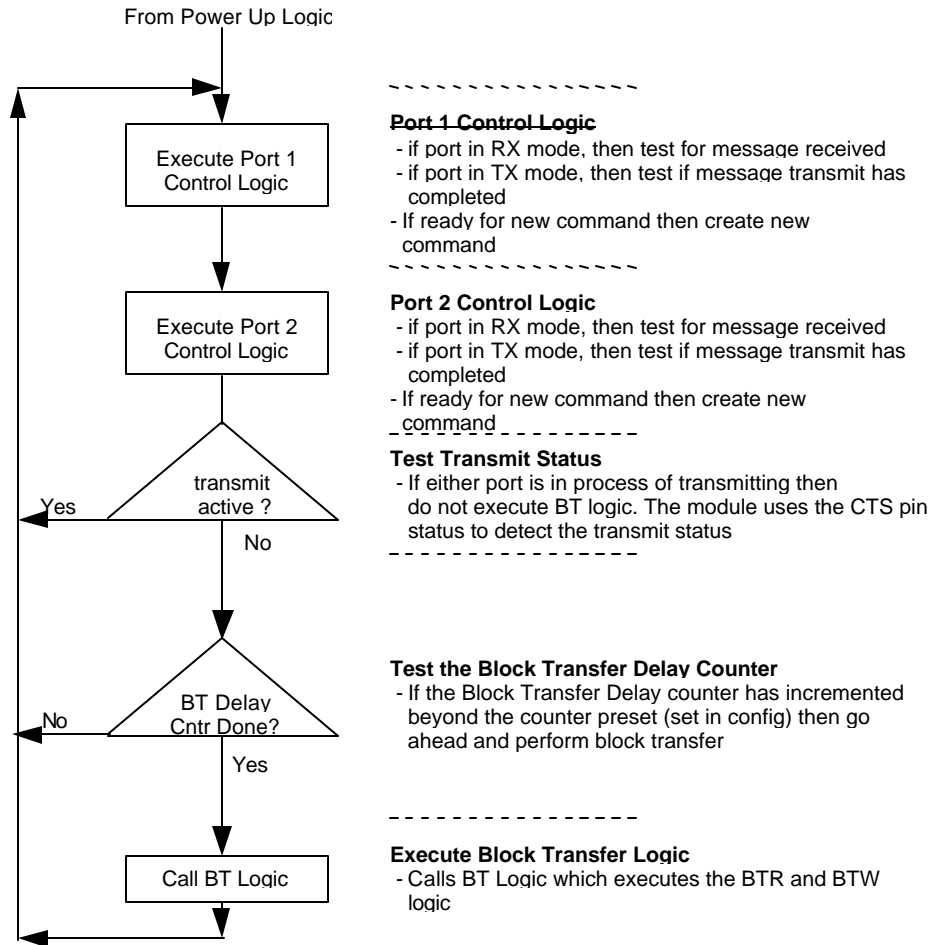
Once the register space has been initialized, the module will begin to block transfer with the ladder logic. The first block transfer sent from the module will initiate the configuration process, causing the ladder logic to move a 255 configuration block to the module. Once the module is configured, it will begin the Main Logic Loop.

### 1.3.2 Main Loop Logic

Upon completing the power up configuration process, the module jumps into an infinite loop, which includes the following functions:

1. Port 1 and Port 2 handlers
  - Detect end of message condition
  - Call message handlers
  - Initiate commands
2. Block Transfer
  - Test CTS pin to assure module is not in transmit mode
  - - Test Block Transfer Delay counter
  - If all OK then block transfer

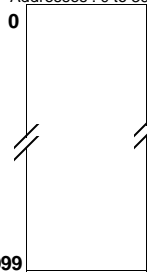




**1.3.3 The Data Space in the module**

One of the concepts, which are important to develop an understanding of, is the relationship between the data space in the module and how this data can be moved between the module and the PLC/SLC processor.

The following discussion explains the data structure in the module and how this data can be moved between the module and the ladder program. Some key points to understand:

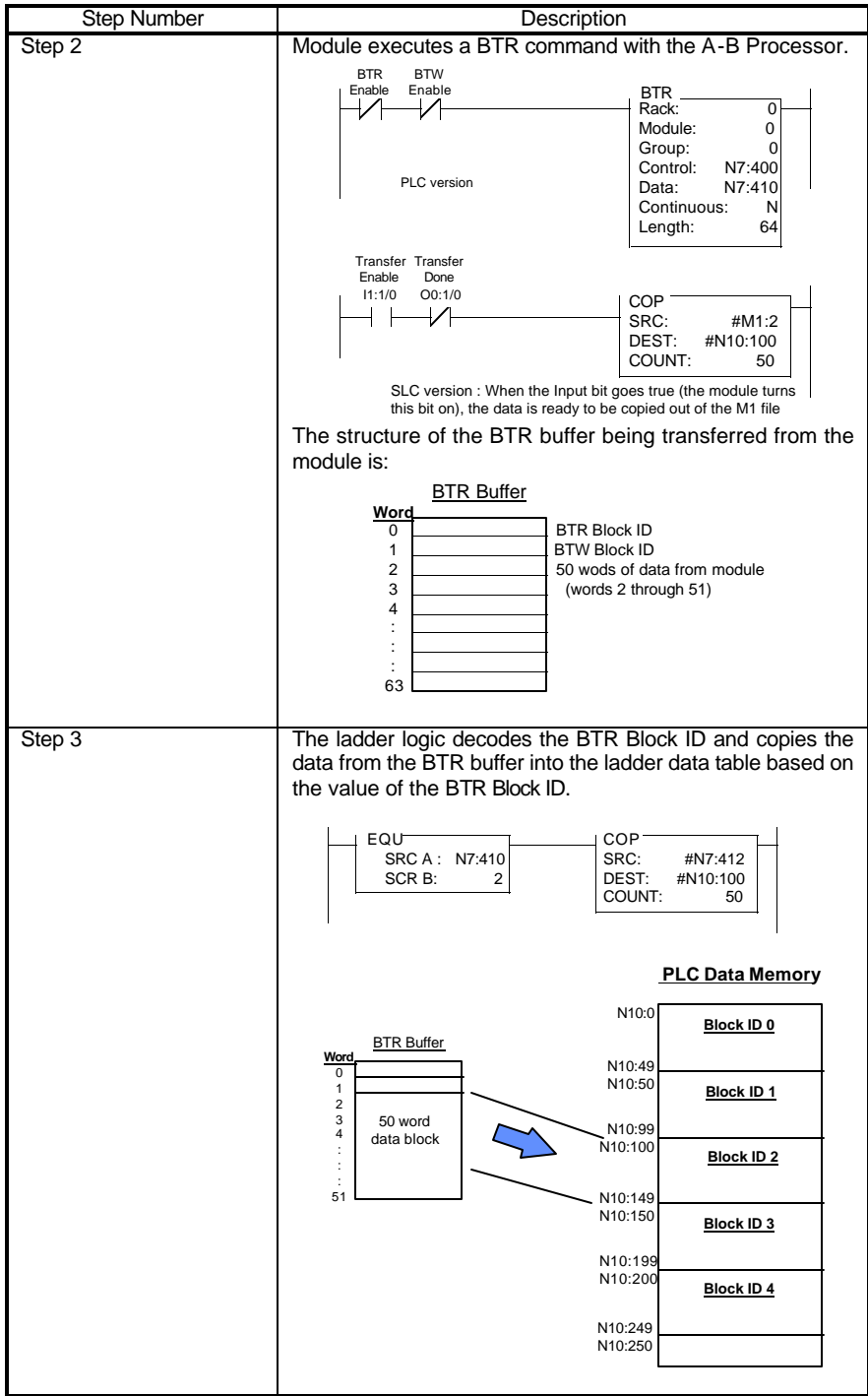
Key Point	Description
Size of data register space in the module	The module maintains a 4000 word data space which can be used as needed by the application for data storage  <b>Module Memory</b> 4000 word block of 16 bit registers Addresses : 0 to 3999  

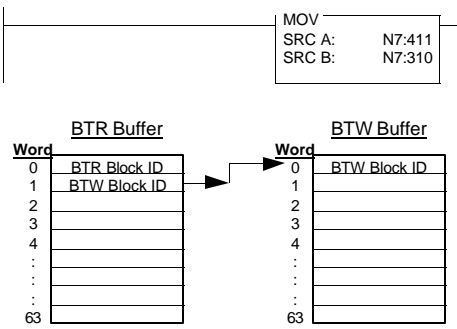
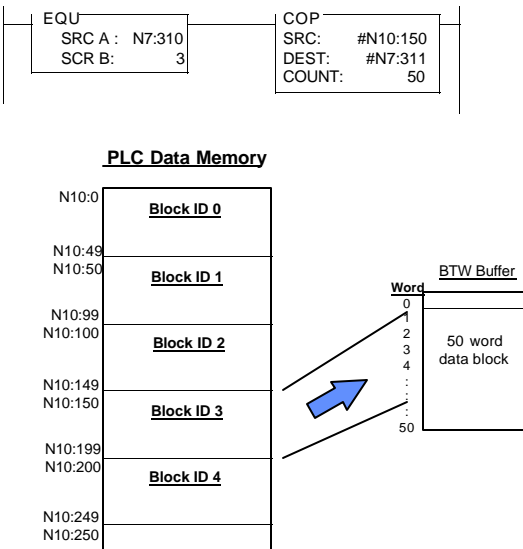
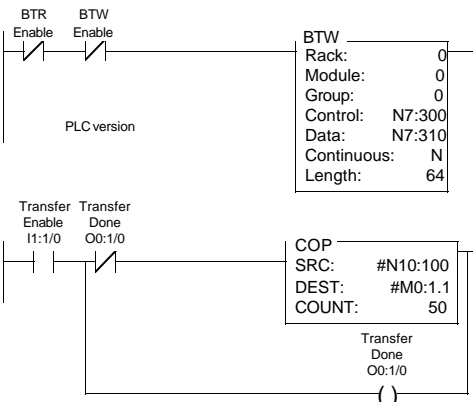
Key Point	Description
How 4000 word module data space is broken down in blocks	<p>This 4000 words block of data is logically broken down into 80 fifty (50) word blocks: 80 blks x 50 wrds/blk=4000 words</p> <p><b>Module Memory</b> 4000 word block of 16 bit registers Block ID 0 to 79 Addresses : 0 to 3999</p>
How data is 'paged' between the module and processor	<p>Via the data transfer sequence outlined in the next section, 50 word blocks of data (or 'pages') are transferred bi-directionally between the module and the PLC/SLC processor.</p>
How data 'page' is placed in the processor's data table	<p>The placement of data in the PLC/SLC processor is controlled by the user and the application ladder logic. Any available data file in the processor can be used as a source of data for the module and as a destination for data from the module.</p>

### 1.3.4 The Backplane Data Transfer Process

The following table provides an overview of the data transfer process between the A-B processor and the module. This process is effectively controlled by the ladder logic in the processor. The following provides some insight into the steps, which occur in the module and in the ladder to effect a successful data transfer. Reference can be made to the example logic in the Appendix to see an actual implementation.

Step Number	Description
Step 1	<p>Module generates BTR and BTW Block ID numbers based on the following logic:</p> <p><b>BTW Block ID</b> if ( BTW Block ID &gt;= Write Block Cnt ) then BTW Block ID = 80 elseif( BTW Block ID &gt;= 80 + Command Block Cnt ) then BTW Block ID = Write Block Start else BTW block ID = BTW block ID + 1</p> <p><b>BTR Block ID</b> if ( BTR Block ID &gt;= Read Block Cnt ) then BTR Block ID = Read Block Start else BTR block ID = BTR block ID + 1</p>



Step Number	Description
Step 4	<p>Transfer the BTR Block ID from the BTR Buffer to the BTW buffer.</p> 
Step 5	<p>Copy ladder data memory, whether Data, Command List or Configuration, to the BTW buffer. The actual data copied depends on the decoding of the BTW Block ID number.</p> 
Step 6	<p>Execute the BTW Command</p>  <p>PLC version : When the ladder logic has transferred the ladder data into the M0 file, the Transfer Done bit is set by the ladder. This bit is used by the module to determine when the transfer process is complete.</p>

Step Number	Description
Step 7	The module receives the BTW data. After decoding the BTW Block ID number, the module will transfer the BTW buffer data into the correct location in the modules memory.

**1.3.5 Interlocking the Block Transfers**

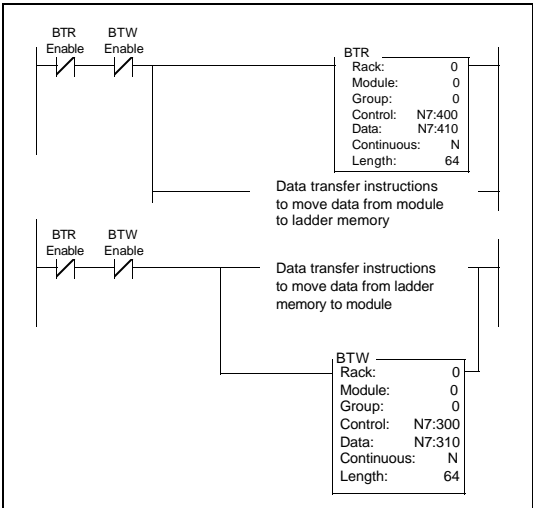
One of the fundamental assumptions that the module makes is that there will be one BTR per one BTW command. In the module, upon completing the BTR instruction, the module jumps immediately to the BTW instruction. To the programmer who follows our example logic this has rather minor implications.

Problems arise however when a ladder logic implementation is attempted which does not meet the module's block transfer expectations. Specifically, the following must be adhered to when programming the ladder logic for the module:

PLC Program using BTR/BTW Instructions

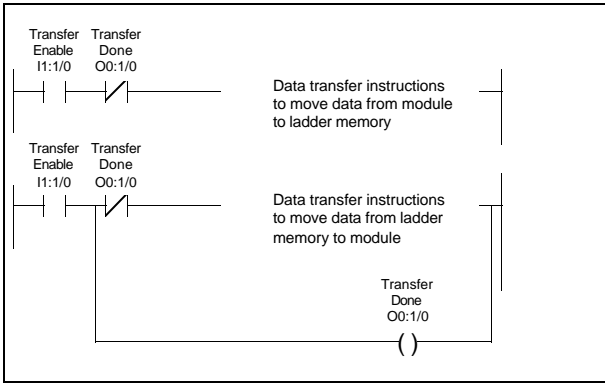
In the 1771 types of processors (PLC 2, PLC 3 and PLC 5), the BTR and BTW Enable bits must be used to enable the Block Transfer Instructions. With this type of programming, the PLC is guaranteed not to execute two block transfers at the same time, and the BTR and BTW instructions are guaranteed to alternate.

Ample examples of this type of block transfer programming are available in A-B documentation as well as in the example ladder logic program in the Appendix.



SLC Program using M0/M1 Instructions

In the SLC processors, there is no true mechanism for guaranteeing the integrity of data block transfers, as there is in the PLC platform. For this reason we have developed a handshaking mechanism, which is designed to assure that all the words in the M0 and M1, files are transferred in unison. Following this mechanism is the only way that we can assure that the data in a block corresponds to the Block ID being transferred. The basic ladder programming which must be implemented in an SLC application is as follows:



**1.3.6 SLC Processor Configuration**

When initially setting up the SLC program file, or when moving the module from one slot to another, the user must configure the slot to accept the ROC module.

It is important that the slot containing the ProSoft module be configured as follows:

- 1746-BAS module with 5/02 or greater configuration
- or enter 13106 for the module ID code
- Configure the M0/M1 files for 64 words
- Configure I/O for 8 words

The following is a step by step on how to configure these files using Allen-Bradley APS software. Other software packages users should follow similar steps.

From the Main Menu:

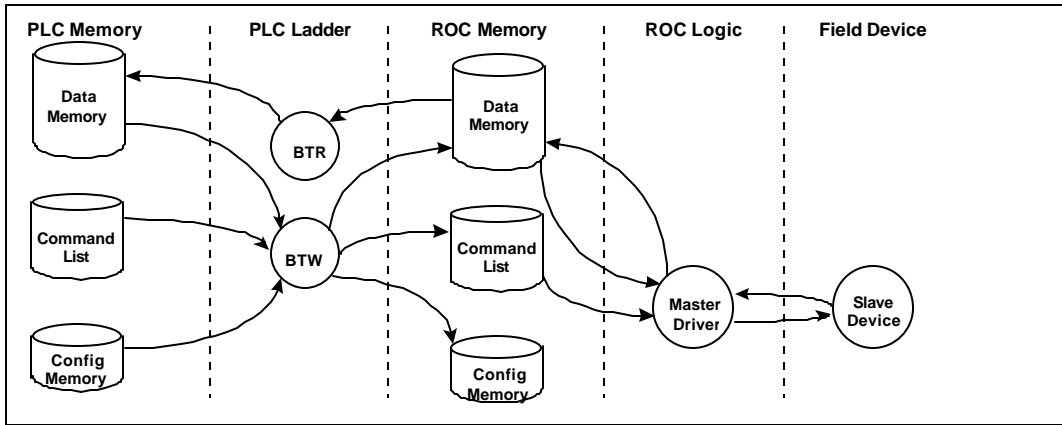
- 1) Select the correct processor program and F3 for Off-line programming
- 2) F1 for Processor Functions
- 3) F1 for Change Processor  
 Modify the processor here if necessary (Note the ROC will only work with 5/02 or greater processors)
- 4) F5 for Configure I/O  
 Select *1746-BAS module for SLC 5/02 or greater*, or enter 13106 for module code
- 5) F9 for SPIO Config when the correct slot is highlighted
- 6) F5 Advanced Setup
- 7) F5 for M0 file length - type in 64 and Enter
- 8) F6 for M1 file length - type in 64 and Enter

Esc out and save configuration

**1.4 Data Flow**

**1.4.1 General concepts**

In developing a solid understanding of the module’s operation, it is important to understand the movement of data in between the ladder logic, the module and the Master drivers.



The following discussion covers the flow of data in the different stages. Further discussion is available in later sections on the flow of data under the different operating modes of the ports.

### 1.4.2 Reading data from the module

The module maintains a 4000 block of data memory. This memory contains:

1. The results of Master port transactions
2. Slave port Status data
3. Module Revision information
4. Master port Status data

During the transfer of data from the module to the PLC, the ladder logic is able to gain access to this information.

### 1.4.3 Writing data to the module

The module, depending on the configuration of the ports, requires three basic types of data in order to operate correctly. The three types of memory, which can be transferred to the module, are as follows:

1. Configuration Data. This data contains all of the parameters necessary for the module to configure the serial ports and to set up the data transfers between the module and the ladder logic.
2. Command List. This set of data contains all of the parameters the module required to encode valid commands which will be transmitted out the Master port to Fisher ROC slave devices. Up to 20 Command List blocks can be sent to the module for a total of 100 commands.
3. Data Memory. This type of memory is moved to the module to provide the data values necessary for the Master port to service write requests (i.e., the data written to the slaves).

### 1.4.4 Master Port Driver

Under normal applications, the Master port is used primarily to issue read commands to slave devices, thereby acting as a data gatherer and then transferring the data which has been read to the ladder logic.

The module uses the Command List entries to encode valid Fisher ROC commands. As each command is executed, the module scans for the next entry in the Command List. If the Master port is issuing a read command, the results of the read are deposited in the Data Memory. If the Master port is issuing a write command, data from the Data Memory is written to the slave device.

For every command, which the module executes, the status of the command can be found in the Master Error Table. This table can be located anywhere in the Data Memory block and is read back into the ladder logic as part of the regular data transfer process.

## 1.5 ROC Support of Fisher ROC Functionality

The ROC module supports several Fisher ROC Function Codes used for data transfer. The following table documents the Function Codes and the point types that are supported (See Section 6 for more details).

Function Code	Description	Read / Write
8	Set New Time and Date	Write
10	Send Data From Configurable Opcode Tables	Read
24	Store and Forward	Read / Write
120	Send Pointers for Alarm, Event, and History Logs	Read
128	Send Archived Daily and Hourly Data for the Currently Selected Day and Month	Read
130	Send Archived Hourly and Daily Data for Specified History Point Starting at Specified History Pointer	Read
180	Send Specified Parameters	Read
181	Set Specified Parameters	Write



## 2 Getting Going - A Step by Step Approach

Installation of the 3100/3150-ROC module is easily accomplished. Installation into a system requires only a few steps. Following is a step-by-step procedure for getting an application operational:

Step	Example	User Application
1. Identify Rack position	Rack 0 Group 2 Slot 0	Rack : ____ Group : ____ Slot : ____
2. Identify PLC Data Files usage	BT Buffers: N7 BT Control: N7 Config File: N7 Data File : N10	BT Buffers: N ____ BT Control: N ____ Config File: N ____ Data File : N ____
3. Ladder Logic	Example on disk and in Appendix (Several examples to choose from)	Select the example closest to your application and modify as needed
4. Modify Logic for rack position	<u>PLC</u> BTR - Rung 2:0 BTW - Rung 2:1 <u>SLC</u> I:x.0 addresses O:x.0 addresses M0:x addresses M1:x addresses	Modify these instructions as needed based on the required rack position. Be sure to configure the slot in the SLC
5. Modify Logic for Data file usage	N7 and N10 is used as data space for the module	Create files and change references from N7 and N10
6. Install card in rack	Power down rack and install module	Power down and install module
7. Connect a comm cable to the front of the module	Decide on cable type needed for application	
9. Apply power to system and place PLC in RUN	Monitor the status files and the LEDs on the front of the module	

Once the hardware has been installed and the necessary programming has been downloaded to the processor, the system is ready (Presuming all other system components are safely ready).

### 3 Ladder Logic Overview

Data transfers between the processor and the ProSoft Technology module occur using the Block Transfer commands, in the case of the PLC, and M0/M1 data transfer commands, in the case of the SLC. These commands transfer up to 64 physical registers per transfer. The logical data length changes depending on the data transfer function.

The following discussions and Sections details the data structures used to transfer the different types of data between the ProSoft Technology module and the processor. The term 'Block Transfer' is used generically in the following discussion to depict the transfer of data blocks between the processor and the ProSoft Technology module. Although a true Block Transfer function does not exist in the SLC, we have implemented a pseudo-block transfer command in order to assure data integrity at the block level. Examples of the PLC and SLC ladder logic are included in the Appendix.



In order for the ProSoft Technology module to function, the PLC/SLC must be in the RUN mode, or in the REM RUN mode. If in any other mode (Fault/PGM), the module will stop all communications until block transfers resume.

#### 3.1 Operational Overview

On power up the module moves a 255 into Word 1 of the BTR data file. This is a signal that the module needs to receive configuration data before proceeding any further. Once the configuration is received the module will begin transferring data to and from the processor depending upon how many Read and Write block counts have been configured. Once these are completed, the module will then transfer the command blocks if any have been configured.

#### 3.2 Ladder Logic

The flow of the ladder logic is somewhat predefined by the way the module has been programmed. The expected flow of the ladder logic should be as follows:

##### Read Rung

1. Read Data from the Module. In the case of the PLC the module data will be transferred into the BTR Buffer. In the case of the SLC the module data will be accessed directly out of the M1 file
2. Decode the BTR Block ID number. Depending on the value of the BTR Block ID, copy the module data into the correct location in the ladder logic data table
3. Move the BTW Block ID Number from Word 1 of the BTR Buffer into Word 0 of the BTW Buffer. In the case of the SLC the transfer will actually be from Word 1 of the M1 file to Word 0 of the M0 file. The BTW Block ID number should be manipulated if necessary to assure that data is not overwritten in the module (The LIM test branch does this in the example logic)
4. Test for Event Initiated Commands and module configuration

##### Write Rung

1. Decode the BTW Block ID number and depending on the value move either data values, Command List values or Configuration values to the BTW buffer (M0 file in the SLC)
2. If the configuration transfer is enabled, then clear the configuration enable bit
3. In an Event Initiated Command is enabled, then clear the enable bit
4. Execute the BTW transfer. In the PLC this will be done by enabling the BTW instruction. In the SLC, this will be done by setting the Transfer Done bit (an Output bit has been assigned to this function in the design of the module)

## 4 Writing to the Module

This section provides reference level details on the transfer of data from the PLC/SLC processor to the ROC module. This type of transfer allows the ladder logic to send configuration, command list and data to the module.

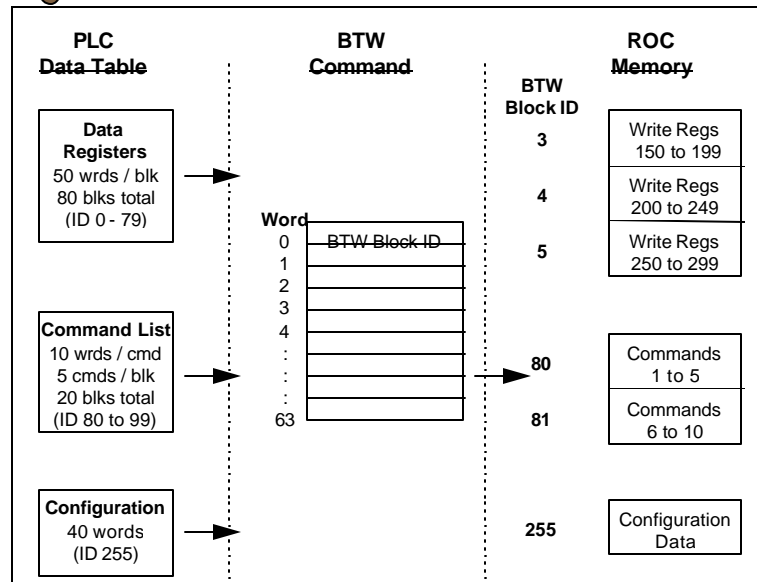
### 4.1 Block Transferring to the Module

Data transfer to the module from the processor is executed through the Block Transfer Write function. The different types of data, which are transferred, require slightly different data block structures, but the basic data structure is:

Word	Name	Description								
0	BTW Block ID	A block page identifier code. This code is used by the ProSoft module to determine what to do with the data block. Valid codes are: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>BTW Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-79</td> <td>Module Data Memory</td> </tr> <tr> <td>80-99</td> <td>Command List</td> </tr> <tr> <td>255</td> <td>Module Communication Configuration</td> </tr> </tbody> </table>	BTW Code	Description	0-79	Module Data Memory	80-99	Command List	255	Module Communication Configuration
BTW Code	Description									
0-79	Module Data Memory									
80-99	Command List									
255	Module Communication Configuration									
1 to 63	Data	The data to be written to the module. The structure of the data is dependent on the Block ID code. The following sections provide details on the different structures.								



Although the full physical 64 words of the data buffer may not be used, the BTW and M0 lengths must be configured for 64 words, otherwise module operation will be unpredictable.



**Data transfer from PLC to ROC:** Data values and Command List entries are 'paged' into the ROC module. The data type and location being written into corresponds to the BTW Block ID number. The BTW Block ID number is controlled by the ROC module, as discussed later in this section.

## 4.2 Communications Configuration [ BTW Block ID 255 ]

The ProSoft Technology firmware communication parameters must be configured at least once when the card is first powered up, and any time thereafter when the parameters must be changed.

### Power Up

On power up, the module enters into a logical loop waiting to receive configuration data from the processor. While waiting, the module sets the second word of the BTR buffer (the BTW Block ID) to 255, telling the processor that the module must be configured before anything else will be done. The module will continuously perform block transfers until the communications configuration parameters block is received. Upon receipt, the module will begin execution of the command list if present, or begin looking for the command list from the processor.

### Changing parameters during operation

Changing values in the configuration table can be done at any time. The module does not accept any of the changes until the 're-configuration' process is initiated. This can be accomplished in several ways, including:

1. Cycle power to the rack
2. Press the reset pushbutton on the module (3100 only)
3. Move 255 into BTW Block ID position (See example logic when B3/0 is set)

During this process, the 'CFG' LED will toggle, giving a visual indication that the module has received the configuration block.



Transferring the Communications Configuration Parameters to the module will force a reset of the communication port, as well as dropping DTR for 200 ms pulses to reset any attached hardware.

The configuration data block structure, which must be transferred from the processor to the module, is as follows:

BTW Buffer	Data Addr	Name	Example Value
0		BTW Block ID	255
		<b>Port 1 Config</b>	
1	N[ ]:0	Port Configuration Word	0 - Master
2	N[ ]:1	Port Unit Addr	1
3	N[ ]:2	Baud Rate	5
4	N[ ]:3	RTS to TxD Delay	0
5	N[ ]:4	RTS Off Delay	0
6	N[ ]:5	Response Timeout	0
7	N[ ]:6	Intercharacter Delay	0
8	N[ ]:7	Spare	0
9	N[ ]:8	Spare	0
10	N[ ]:9	Spare	0
		<b>Port 2 Config</b>	
11	N[ ]:10	Port Configuration Word	0 - Master
12	N[ ]:11	Port Unit Addr	1
13	N[ ]:12	Baud Rate	5
14	N[ ]:13	RTS to TxD Delay	0
15	N[ ]:14	RTS Off Delay	0
16	N[ ]:15	Response Timeout	0
17	N[ ]:16	Intercharacter Delay	0
18	N[ ]:17	Spare	0
19	N[ ]:18	Spare	0
20	N[ ]:19	Spare	0

(Cont'd)

System Configuration			
21	N[ ]:20	Read Block Cnt	3
22	N[ ]:21	Write Block Cnt	1
23	N[ ]:22	Cmd Block Cnt	2
24	N[ ]:23	Slave Err Ptr	100
25	N[ ]:24	Master Error Ptr	120
26	N[ ]:25	BT Delay Cntr	0
27	N[ ]:26	Floating Point Offset	0
28	N[ ]:27	Read Block ID Start	0
29	N[ ]:28	Write Block ID Start	0
30	N[ ]:29	Spare	0

**Port 1 and 2 Configuration**

Data Addr	Name	Description																		
N[ ]:0 N[ ]:10	Port Configuration Word	<p>This register contains several communication configuration parameters encoded into the word. These are as follows:</p> <p><b>Protocol Mode:</b> The port's protocol mode is selected by these bits:</p> <p><b>Bits</b> <b>210</b> 000 Fisher ROC Master</p> <p><b>Unused Bits:</b> All unused bits must be set to 0.</p> <p><b>Stop Bits:</b> The number of stop bits to be used is defined as follows:</p> <p><b>Bits</b> <b>13 12</b> 0 0 One stop bit 0 1 Two stop bits 1 x Invalid Port Configuration</p> <p><b>Parity:</b> The parity mode to be used by the module is defined by this word as follows:</p> <p><b>Bits</b> <b>15 14</b> 0 0 No parity 0 1 Odd parity 1 0 Even parity 1 1 Invalid Port Configuration</p>																		
N[ ]:1 N[ ]:11	Master Unit Address	The value entered in this register is used as the Fisher ROC Master Unit address. Valid values range from 0 to 255.																		
N[ ]:2 N[ ]:12	Baud Rate	<p>The baud rate at which the port is to operate. The available configurations are as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Baud Rate</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>300 Baud</td> </tr> <tr> <td>1</td> <td>600 Baud</td> </tr> <tr> <td>2</td> <td>1200 Baud</td> </tr> <tr> <td>3</td> <td>2400 Baud</td> </tr> <tr> <td>4</td> <td>4800 Baud</td> </tr> <tr> <td>5</td> <td>9600 Baud</td> </tr> <tr> <td>6</td> <td>19200 Baud</td> </tr> <tr> <td>7</td> <td>38400 Baud</td> </tr> </tbody> </table> <p>The module's two ports are limited to an upper baud rate of either 19200 or 38400 baud. The module <b>cannot</b> be configured with one port at 19200 and the other at 38400. If an attempt is made to configure the module in this fashion, a Port Configuration Error will be returned.</p>	Value	Baud Rate	0	300 Baud	1	600 Baud	2	1200 Baud	3	2400 Baud	4	4800 Baud	5	9600 Baud	6	19200 Baud	7	38400 Baud
Value	Baud Rate																			
0	300 Baud																			
1	600 Baud																			
2	1200 Baud																			
3	2400 Baud																			
4	4800 Baud																			
5	9600 Baud																			
6	19200 Baud																			
7	38400 Baud																			

Data Addr	Name	Description
N[ ]:3 N[ ]:13	RTS to TXD Delay	<p>This value represents the time in <u>1 ms increments</u> to be inserted between asserting RTS, and the actual transmission of data. The delay, if greater in duration than the hardware time delay associated with CTS, will override the CTS line until the time-out is complete.</p> <p>This configurable parameter is useful when interfacing with modem based devices, anytime line noise must be allowed to subside before data is transmitted, or if data transmissions must be slowed down. Valid values range from 0 to 65535 (0xffff).</p>
N[ ]:4 N[ ]:14	RTS Off Delay	<p>The value in this word represents the number of <u>1 ms time delay increments</u> inserted after the last character is transmitted and before RTS is dropped. The module automatically inserts a one character width Off Delay, assuring that RTS does not drop until after the last character has been completely sent. Unless working under unusual conditions, this value will normally be configured with a value of 0. Valid value range from 0 to 65535 (0xffff).</p>
N[ ]:5 N[ ]:15	Message Response Timeout	<p>This register represents the message response timeout period in 1 ms increments. This is the time which a port configured as a Master will wait before re-transmitting a command if no response is received from the addressed slave. The value is set depending on the expected slave response times. The allowable range of values is 0 to 65535(0xffff). If a zero value is entered, the module will default to a one second timeout value (1000 ms).</p>
N[ ]:6 N[ ]:16	Inter-character Timeout	<p>This register is used in situations where the end of message character timeout delay must be extended beyond the normal 3.5 character widths. The value entered represents the number of 1 ms intervals of 'no transmission' which will be counted prior to accepting a message. This parameter will be useful in satellite or packet radio installation where a data transmission may be split between two packets. Increasing this value beyond the system's packet handling time will eliminate timeout errors. Valid values range from 0 to 65535 (0xffff)</p>
N[ ]:7 N[ ]:17	Spare	
N[ ]:8 N[ ]:18	Spare	
N[ ]:9 N[ ]:19	Spare	

**System Configuration**

Data Addr	Name	Description
N[ ]:20	Read Data Block Count	<p>This value represents the number of 50 word data blocks which are to be transferred from the ROC Module to the processor. The blocks returned from the module start at block 0 and increment from there. The maximum block count is 80.</p> <p>As an example, a value of 5 will return BTR Block ID data blocks 0, 1, 2, 3, and 4, or module registers 0 to 249.</p> <div data-bbox="846 512 1300 575" style="border: 1px solid black; padding: 2px;"> <p>If a value greater than 80 is entered, a System Configuration Error is activated</p> </div>
N[ ]:21	Write Data Block Count	<p>This value represents the number of 50 word data blocks which are to be transferred from the processor to the ROC Module. The module will use this value to return a BTW Block ID Number to the processor. The ladder logic can use this value to determine which data to move to the ROC via the Block Transfer Write. The maximum block count is 80.</p> <p>As an example, if a value of 5 is entered, the ROC will return BTW Block ID numbers 0, 1, 2, 3, and 4 to the ladder logic (See Section 4.2).</p> <div data-bbox="846 898 1300 961" style="border: 1px solid black; padding: 2px;"> <p>If a value greater than 80 is entered, a System Configuration Error is activated</p> </div>
N[ ]:22	Command Block Count	<p>This value represents the number of 50 word Command Blocks which are to be transferred from the processor to the ROC Module. This value will be 0 if the module will not be configured with a Master port. See the discussion in Section 4.1 for details on the number of Command Blocks needed. The maximum block count is 20.</p> <div data-bbox="857 1163 1312 1226" style="border: 1px solid black; padding: 2px;"> <p>If a value greater than 20 is entered, a System Configuration Error is activated</p> </div>

Data Addr	Name	Description
N[ ]:23	Slave Error Block Pointer	<p>This value represents the relative starting position in the module's data table within which the Fisher ROC Slave Error Data Block is placed. The Slave Error Table is a 20 word block containing Slave port status and several communication counters. The error data can be placed anywhere in the module's data space (0 to 3999). The contents of the Error Table can then be obtained as part of the regular Register Data.</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>If a value greater than 3980 is entered, a System Configuration Error is activated</p> </div> <p><b>ROC Module Memory</b></p> <p>Block ID 0 to 79 Address: 0 to 3999</p>
N[ ]:24	Master Error Block Pointer	<p>This value represents the relative starting position in the module's data register table within which the Master Error Data Block is placed. The error block (120 words in length) can be placed anywhere in the module's data space (0 to 3999). The contents of the Error Table can then be obtained as part of the regular Register Data.</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>If a value greater than 3880 is entered, a System Configuration Error is activated</p> </div> <p><b>ROC Module Memory</b></p> <p>Block ID 0 to 79 Address: 0 to 3999</p>



Data Addr	Name	Description
N[ ]:25	Block Transfer Delay Counter	This is an empirical value used by the module to balance the amount of time the module spends block transferring and the amount spent handling port communications. The value entered is used as a loop counter in the module, where each time through the loop the count is incremented. When the count equals the Block Transfer Delay Counter a Block Transfer sequence is initiated. The range on this value is 0 to 255.  <u>Example:</u> In Master Mode applications with the module in a remote rack, the frequency of command execution can be improved by entering a value of 75-150. The value must be determined empirically.
N[ ]:26	Spare	
N[ ]:27	Read Block ID Start	This value determines the starting BTR Block ID number, which will be returned from the module. As an example, if the ladder logic needs to receive Blocks 2 through 5 from the module, the parameter should be configured with a '2' and the Read Block Count should be set to '4'. Valid values range from 0 to 79.
N[ ]:28	Write Block ID Start	This value determines the starting BTW Block ID number, which the module will return to the ladder logic. As an example, if the ladder logic needs to write into Blocks 4 through 5 in the module, this parameter should be set to '4' and the Write Block Count should be set to '2'. Valid values range from 0 to 79.

### 4.3 Writing Into Module Data Memory [ BTW Block ID Codes 0-79 ]

Writing into the ROC register data space is accomplished using a Block Transfer Write with BTW Block ID codes from 0 to 79 followed by 50 words of data.

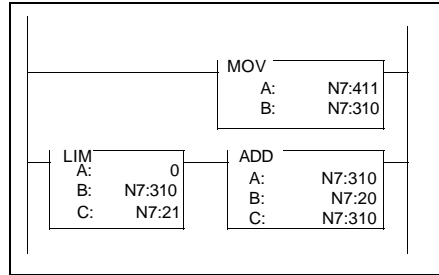


Care must be exercised with memory layout to assure that ROC read and write commands do not overwrite data being moved in from the processor ladder logic. Fisher ROC data **cannot** be moved into a 50 word block that is also updated by the processor. The ladder logic examples in the Appendix address this concern.

#### 4.3.1 Ladder Logic to Write Data to Module

The ladder logic required to move data to the module is a simple series of EQU-COP branches, or it can be implemented using indirect addressing. The way that we have implemented the transfer to the module in all of our example ladder logic (See Appendix and Application Notes) is through a two step process, where:

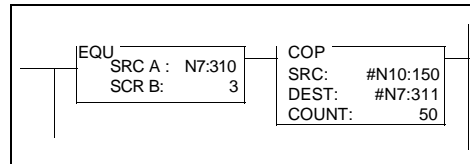
- Step 1: During the BTR process, the module will 'feed' the ladder logic a BTW Block ID Number in the second word of the BTR Data Buffer. Ladder logic is implemented to accept this value, condition it if needed, and then to move the value to the actual BTW Block ID location. The ladder logic to do this is shown below:



**Setting up the BTW Block ID Number**

Located at the bottom of the BTR rung (Rung 0), this logic moves the BTW Block ID Number being received from the module and offsets it by the Read Block Count (N7:20) in order to assure that PLC data does not overwrite the data being returned from the module to the PLC. See logic in Appendix for implementation details.

Step 2: During the processing of the BTW rung, the ladder logic will test for the value in the BTW Block ID register and based on the value, copy data from the data table into the BTW Block Transfer buffer. This process requires that every BTW Block ID which will be processed be accounted for with a branch of logic. An example of the ladder logic required follows:



**Test BTW Block ID and move data to BTW Buffer**

This branch, located in the BTW rung (rung 1) is an example of the logic that must be implemented for each data block to be move to the module. See logic in Appendix for implementation example.

**4.3.2 Block Transfer Data Structure**

The structure of the block transfer buffer when writing data to the module is shown below:

Word	Name	Description										
0	BTW Block ID	<p>The block identifier number allows the ROC Module to decode which '50 word page' in the module's 4000 word data space the data is to be written. The data space to be written into can be determined by multiplying the BTW Block ID by 50. The result is the first word of the 'page'. As an example:</p> <table border="1"> <thead> <tr> <th>BTW Block ID</th> <th>Data Space</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0 to 49</td> </tr> <tr> <td>1</td> <td>50 to 99</td> </tr> <tr> <td>10</td> <td>500 to 549</td> </tr> <tr> <td>20</td> <td>1000 to 1049</td> </tr> </tbody> </table> <p>By paging the different data blocks into the module the processor can control the module data memory contents.</p>	BTW Block ID	Data Space	0	0 to 49	1	50 to 99	10	500 to 549	20	1000 to 1049
BTW Block ID	Data Space											
0	0 to 49											
1	50 to 99											
10	500 to 549											
20	1000 to 1049											
1 to 50	Data	The data to be written to the module.										

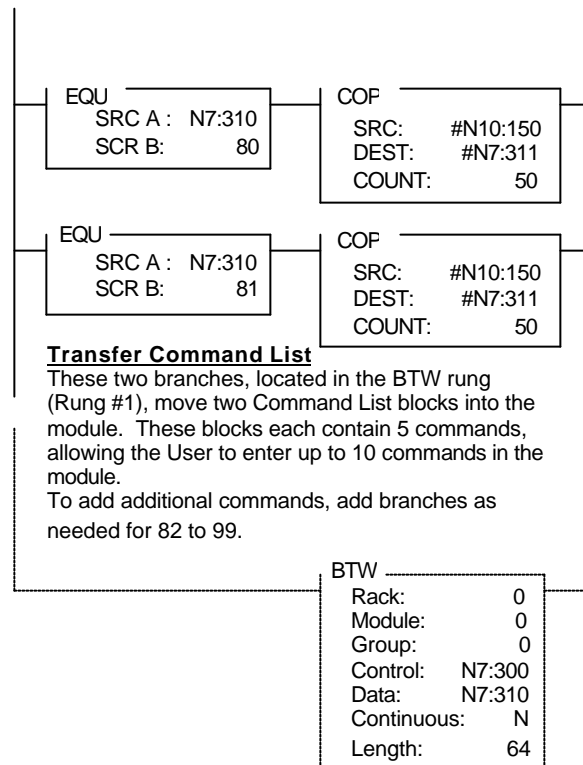
## 4.4 Command List Configuration - Master Mode [ BTW Block ID Codes 80-99 ]

A ROC Master port establishes communications and performs various communications functions based on the data, which the user has placed in the command list. The command list consists of up to 100 individually configured command data blocks (10 words reserved per command), which are shared between the two available ports (in the case when the module is configured with two Master ports).

### 4.4.1 Command List Ladder Logic

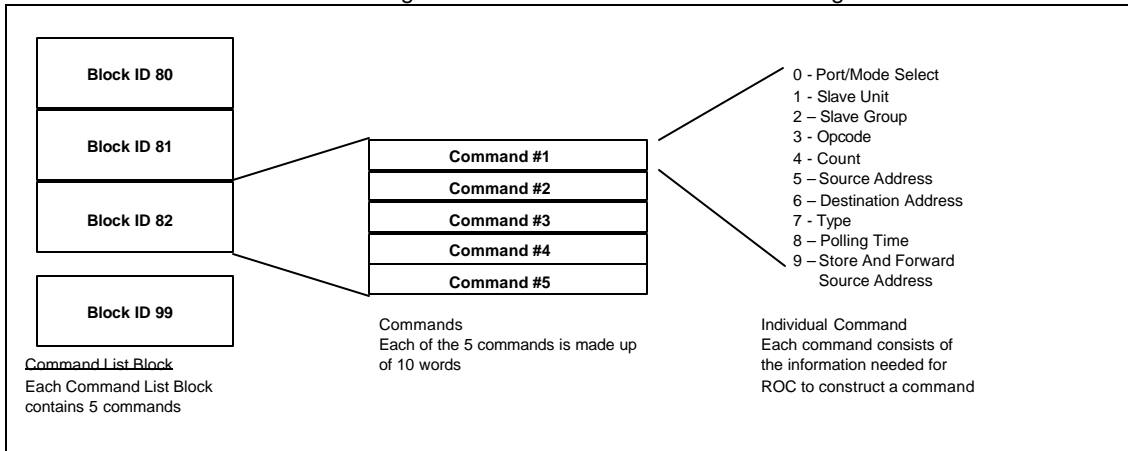
This list, entered into the processor Data Table, is transferred to the module's memory using BTW Block ID codes 80-99 with each code representing a 50 word block, or 5 commands.

An example of the ladder logic to move the commands to the module is as follows:



### 4.4.2 Command List Structure

The structure of the block containing the Command List is shown in the diagram below:



See Section 6 for details on configuring Fisher ROC Commands

Name	Description												
Port/Mode Select	<p>The Port/Mode Select parameter allows the application to select which port the ROC Module will use to execute the command, and whether the command will be performed continuously or under direct ladder logic control (Control). Valid values are:</p> <table border="1"> <thead> <tr> <th>Port/Mode</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable Command</td> </tr> <tr> <td>1</td> <td>Port 1 Continuous Command</td> </tr> <tr> <td>2</td> <td>Port 2 Continuous Command</td> </tr> <tr> <td>9</td> <td>Port 1 Control Command</td> </tr> <tr> <td>10</td> <td>Port 2 Control Command</td> </tr> </tbody> </table> <p><u>Continuous: Opcodes 8 and 181</u> Write commands enabled as continuous will be executed every time the module's Command List is scanned.</p> <p><u>Control Command Mode</u> In the Control Command Mode, the command will only be executed when the Command Enable Bit (see Section 4.5) transitions from 0 to 1. The command is executed once per transition (i.e., the module performs some one-shot logic to assure that the command only executes one). To clear the one-shot in the module, the Command Enable Bit must change state from 1 back to 0.</p>	Port/Mode	Description	0	Disable Command	1	Port 1 Continuous Command	2	Port 2 Continuous Command	9	Port 1 Control Command	10	Port 2 Control Command
Port/Mode	Description												
0	Disable Command												
1	Port 1 Continuous Command												
2	Port 2 Continuous Command												
9	Port 1 Control Command												
10	Port 2 Control Command												
Slave Unit	The slave Unit represents the Fisher ROC slave unit address of the destination slave station. Addresses should be entered in the decimal form.												
Slave Group	The slave group represents the Fisher ROC slave group address of the destination slave station. Addresses should be entered in the decimal form.												

Name	Description																
Opcode	<p>The opcode entered in the table tells the ROC Module what command to execute. The different choices are detailed in Section 6, but in an overview they are as follows:</p> <table border="0" data-bbox="792 331 1354 625"> <thead> <tr> <th><u>Opcode</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>8</td> <td>Set New Time and Date</td> </tr> <tr> <td>10</td> <td>Send Data From Configurable Opcode Tables</td> </tr> <tr> <td>120</td> <td>Send Pointers for Alarm, Event, and History Logs</td> </tr> <tr> <td>128</td> <td>Send Archived Daily and Hourly Data for the Currently Selected Day and Month</td> </tr> <tr> <td>130</td> <td>Send Archived Hourly and Daily Data for Specified History Point Starting at Specified History Pointer</td> </tr> <tr> <td>180</td> <td>Send Specified Parameters</td> </tr> <tr> <td>181</td> <td>Set Specified Parameters</td> </tr> </tbody> </table>	<u>Opcode</u>	<u>Description</u>	8	Set New Time and Date	10	Send Data From Configurable Opcode Tables	120	Send Pointers for Alarm, Event, and History Logs	128	Send Archived Daily and Hourly Data for the Currently Selected Day and Month	130	Send Archived Hourly and Daily Data for Specified History Point Starting at Specified History Pointer	180	Send Specified Parameters	181	Set Specified Parameters
<u>Opcode</u>	<u>Description</u>																
8	Set New Time and Date																
10	Send Data From Configurable Opcode Tables																
120	Send Pointers for Alarm, Event, and History Logs																
128	Send Archived Daily and Hourly Data for the Currently Selected Day and Month																
130	Send Archived Hourly and Daily Data for Specified History Point Starting at Specified History Pointer																
180	Send Specified Parameters																
181	Set Specified Parameters																
Count	<p>For commands 8, 10, 120, 128, and 130 count is the number of bytes the Fisher ROC command is to read or write. For commands 180 and 181 count represents the number of data points. See Section 6 for a detailed discussion on the byte lengths to be specified for the different commands.</p>																
Source Address	<p>The value represents the register addresses, for both read and write commands, from which data will be obtained. When issuing a command, the Source Register Address is the register location in the module where the command will begin getting the command parameters to send to the slave. The parameters need to start at this address and be placed in sequential words.</p>																
Destination Address	<p>The value represents the register addresses, for read commands, to which the data received will be written. When issuing a command, the Destination Address is the register location in the module where the command will begin placing the data from the slave. The data will be placed in the data table one value per integer. See section 6.1.1 for a detailed explanation.</p>																
Type	<p>The Type field is relevant only during the 180 read command. The Stripping of the TLP from the data is available. <b>In order to use the stripping feature, all data types requested in one command must be of the same size. (i.e. all floating point, all integers or all bytes)</b></p> <table border="0" data-bbox="792 1287 1386 1392"> <thead> <tr> <th><u>Type</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Default value. Strips TLP from data and stores data only.</td> </tr> <tr> <td>1</td> <td>Stores TLP and data as received.</td> </tr> </tbody> </table>	<u>Type</u>	<u>Description</u>	0	Default value. Strips TLP from data and stores data only.	1	Stores TLP and data as received.										
<u>Type</u>	<u>Description</u>																
0	Default value. Strips TLP from data and stores data only.																
1	Stores TLP and data as received.																
Polling Time Preset	<p>The Polling Time Preset value allows each command to have a configurable execution frequency. In the module, a timer is maintained for each command. Once per second the timer is decremented, until it reaches zero. When the timer reaches zero, the command is enabled for execution, and the timer is reset to the Polling Timer Preset value. The resolution of the polling timer is 1 second. Valid values are 0 to 65535 (0xffff).</p>																
Store And Forward Source Address	<p>The value represents the register addresses, for both read and write commands, from which the Store and Forward Communication Path header will be obtained. The contents of the 10 words starting at this address will be used as the Communication Path for ROC to ROC Communications.</p>																

### 4.4.3 Editing the Command List

Entering the Command List is a matter of entering the correct values into the PLC data table. Using the ladder logic programming software, enter the values necessary to setup one or more valid commands.



#### **Hints to Make Life Easier**

When first setting up the Command List we recommend that you start out with one command. This one command will allow the module to begin transmitting if all else is OK (i.e., ladder logic, cable is connected, etc.). Once the module is transmitting, then attempt to communicate with the slave, then enter any other commands needed.

An example of a command list is shown below. Note that the commands can be entered in rows and that once the column definitions are understood, reviewing the Command List is very easy.

	0	1	2	3	4	5	6	7	8	9
	<u>PORT</u>	<u>SLV</u>	<u>SLV</u>	<u>OP</u>		<u>SRC</u>	<u>DEST</u>		<u>POLL</u>	<u>STORE</u>
	<u>NUM</u>	<u>UNIT</u>	<u>GROUP</u>	<u>CODE</u>	<u>CNT</u>	<u>ADDR</u>	<u>ADDR</u>	<u>TYPE</u>	<u>TIME</u>	<u>FWD</u>
N7:50	1	25	2	180	1	0	50	0	0	0
N7:60	1	25	2	10	3	10	60	0	0	0
N7:70	2	26	2	180	1	20	80	0	0	0
N7:80	10	26	2	181	4	30	90	0	0	0

#### **Example Command List**

*An example of multiple message configuration data blocks is shown in the following table.*

### 4.5 Command Control Mode - Master Mode

Under some special operating conditions, it may be necessary for the ladder logic to be able to closely coordinate and control the execution of commands in the Command List. To accommodate this requirement, the ROC module supports something called the Command Control Mode.

When configured in the Command Control Mode, the ladder logic is able to provide Command Enable control on a per Command List entry basis. In addition, when used in conjunction with the Command Done Bits (See Section 5.2), the ladder logic is able to effectively one-shot each command if desired.

#### 4.5.1 The BTW Block Structure

The structure of the Enable bits as they are written to the module in the BTW Block Transfer buffer is as follows:

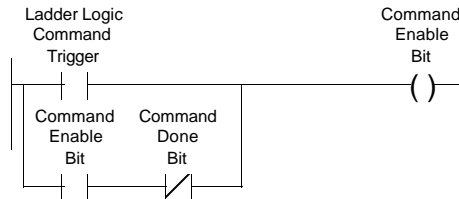
Word	Name	Description
0	BTW Block ID	The Command Enable bits are moved to the module during every BTW transaction. Therefore, all valid BTW Block ID numbers can be used here
1-50	Data	Module data and Command List, as outlined above

Word	Name	Description														
51-56	Cmd Enable Bits	<p>These registers contain Command Enable Bits for each command in the command list, up to the first 96 commands. The Enable Bits are bit mapped into the words depending on their relative position in the Command List. The mapping within the words is as follows:</p> <table border="1"> <thead> <tr> <th>Word</th> <th>Cmnds</th> </tr> </thead> <tbody> <tr> <td>51</td> <td>1 to 16</td> </tr> <tr> <td>52</td> <td>17 to 32</td> </tr> <tr> <td>53</td> <td>33 to 48</td> </tr> <tr> <td>54</td> <td>49 to 64</td> </tr> <tr> <td>55</td> <td>65 to 80</td> </tr> <tr> <td>56</td> <td>81 to 96</td> </tr> </tbody> </table> <p>Example: Word 51 bit 0 is Command #1 Enable</p>	Word	Cmnds	51	1 to 16	52	17 to 32	53	33 to 48	54	49 to 64	55	65 to 80	56	81 to 96
Word	Cmnds															
51	1 to 16															
52	17 to 32															
53	33 to 48															
54	49 to 64															
55	65 to 80															
56	81 to 96															

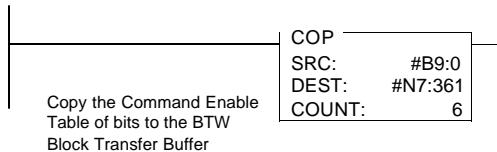
### 4.5.2 Controlling the Commands

When a command is configured in the Command Control Mode, and when the module detects the Command Enable bit changing state from 0 to 1, the module will attempt to execute the command (Three attempts will be made to execute the command). If the command is successfully sent, the Command Done bit will be set. If an error occurs during the sending process, the Command Error bit will be set.

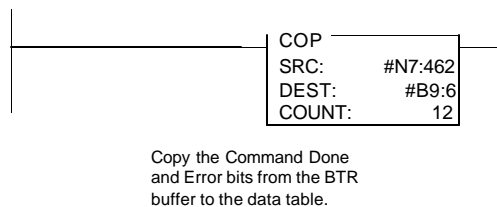
As example of the ladder logic, which might be implemented to control a command, would appear in structure as follows:



The simplest implementation would be to maintain a Binary table of Command Enable Bits, which is, copied to the BTW Buffer every transaction. The following branch of logic can be added to the BTW rung (transfer data to module):



The Command Done and Error bits could then be copied into the same Binary File and referenced in ladder logic after being transferred. The following instruction can be added to the BTR rung (read data from module) accomplish this:



### 4.5.3 Example Command List

Commands can be controlled through configuration of the Command Enable.

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
	<b>PORT</b>	<b>SLV</b>	<b>SLV</b>	<b>OP</b>		<b>SRC</b>	<b>DEST</b>		<b>POLL</b>	<b>STORE</b>
	<b>NUM</b>	<b>UNIT</b>	<b>GROUP</b>	<b>CODE</b>	<b>CNT</b>	<b>ADDR</b>	<b>ADDR</b>	<b>TYPE</b>	<b>TIME</b>	<b>FWD</b>
N7:50	9	25	2	180	1	0	50	0	0	0
N7:80	10	25	2	181	4	30	90	0	0	0

**Example Command List**

An example where the command in N7:50 is configured as a Control Command Mode for Port 1 while the N7:60 command is configured for Port 2.



## 5 Reading from the Module

This section provides reference level details on the transfer of data from the PLC/SLC processor to the ROC module. This type of transfer allows the ladder logic to send configuration, command list and data to the module.

### 5.1 Transferring data from the module [ BTR Block ID 0 to 79 ]

When the Master port driver reads data from a slave or when a Host writes to the Slave port driver, the resulting data is placed into the ProSoft module's data space (Addresses 0 to 3999). This Module Data space is the same block of memory that the PLC/SLC can write into per the above discussion.

The transfer of data from the ProSoft Technology module to the processor is executed through the Block Transfer Read function. The following sections detail the handling of the read data.



Although the full physical 64 words of the data buffer may not be used, the BTR and M1 lengths must be configured for a length of 64 words, otherwise module operation will be unpredictable

#### 5.1.1 The Read Data Block Structure

The BTR buffer definition is:

Word	Name	Description																																												
0	BTR Block ID	<p>The ladder logic uses this value to determine the contents of the data portion of the BTR buffer. With some conditional testing in ladder logic, the data from the module can be placed into the PLC/SLC data table.</p> <div style="text-align: center;"> <table border="1" style="margin: auto;"> <thead> <tr> <th colspan="2">BTR Buffer</th> <th colspan="2">BTW Buffer</th> </tr> <tr> <th>Word</th> <th></th> <th>Word</th> <th></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>BTR Block ID</td> <td>0</td> <td>BTW Block ID</td> </tr> <tr> <td>1</td> <td>BTW Block ID</td> <td>1</td> <td></td> </tr> <tr> <td>2</td> <td></td> <td>2</td> <td></td> </tr> <tr> <td>3</td> <td></td> <td>3</td> <td></td> </tr> <tr> <td>4</td> <td></td> <td>4</td> <td></td> </tr> <tr> <td>:</td> <td></td> <td>:</td> <td></td> </tr> <tr> <td>:</td> <td></td> <td>:</td> <td></td> </tr> <tr> <td>:</td> <td></td> <td>:</td> <td></td> </tr> <tr> <td>63</td> <td></td> <td>63</td> <td></td> </tr> </tbody> </table> <p>An arrow points from the BTR Buffer word 1 to the BTW Buffer word 0.</p> </div> <p>The relationship between the BTR Block ID number and the register table can be put into an equation:  <math display="block">\text{Starting Register Address} = \text{Block ID Number} * 50</math>                     Valid codes are between 0 and 79.</p>	BTR Buffer		BTW Buffer		Word		Word		0	BTR Block ID	0	BTW Block ID	1	BTW Block ID	1		2		2		3		3		4		4		:		:		:		:		:		:		63		63	
BTR Buffer		BTW Buffer																																												
Word		Word																																												
0	BTR Block ID	0	BTW Block ID																																											
1	BTW Block ID	1																																												
2		2																																												
3		3																																												
4		4																																												
:		:																																												
:		:																																												
:		:																																												
63		63																																												
1	BTW Block ID	<p>The module returns this value to the processor to be used to enable the movement of register data and command list blocks to the module. The BTW Block ID number is developed by the module based on the parameters entered in parameters 21 and 22 of Block 255. This value is intended to only be a suggestion and to ease the ladder logic programming requirements. If it is desired to develop a different data transfer series, this may be easily accomplished in ladder logic.</p> <p>Valid codes are:</p> <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;"><u>BTW Code</u></th> <th style="text-align: left;"><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0-79</td> <td>Module Data</td> </tr> <tr> <td>80-99</td> <td>Command List</td> </tr> <tr> <td>255</td> <td>Module Configuration</td> </tr> </tbody> </table>	<u>BTW Code</u>	<u>Description</u>	0-79	Module Data	80-99	Command List	255	Module Configuration																																				
<u>BTW Code</u>	<u>Description</u>																																													
0-79	Module Data																																													
80-99	Command List																																													
255	Module Configuration																																													

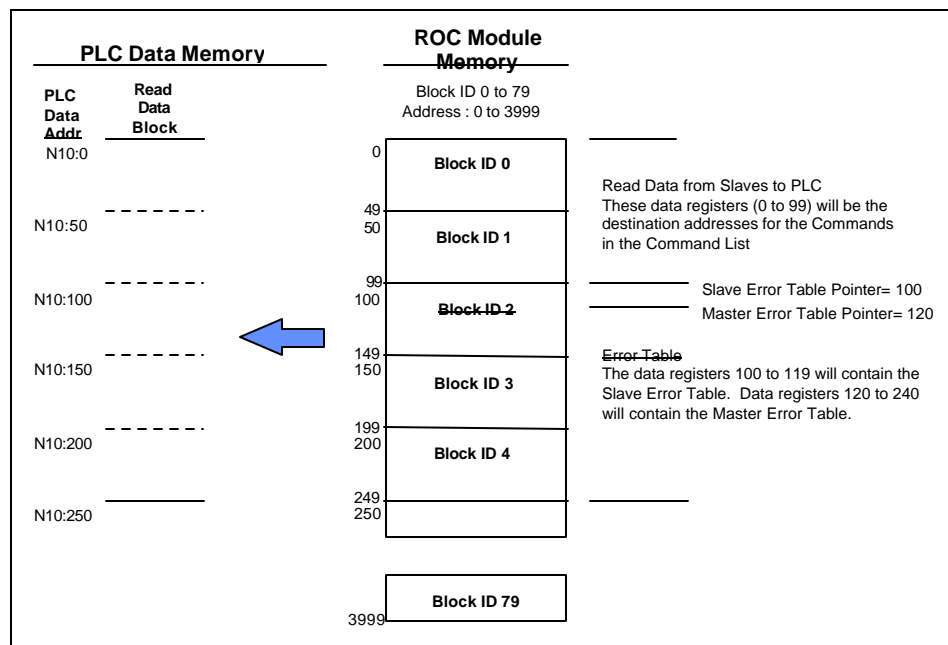
Word	Name	Description
2 to 51	Data	The contents of the module's Register Data space (0 - 3999). This data will contain data received from the slaves, data moved from the processor, and the Slave and Master Error Tables. The values will be 16 bit register values, and should be placed into integer files. Note that the user application ladder logic controls the placement and use of the data registers.
52 to 63	Command Done and Error Bits	See Section 5.2

### 5.1.2 Moving the data from the module to the processor

Data that has been read from the slave devices is deposited into a 4000 word register table in the module. This table is addressed starting at 0 and going up to 3999.

The data register table is transferred from the module to the ladder logic through a paging mechanism designed to overcome the 64 physical word limit of the BTR instruction. The paging mechanism is outlined in the discussion above, but the important thing to understand is the relationship between the page numbers (BTR Block ID numbers) and the register addresses in the module.

The diagram also shows the layout for an example application. Note the number of blocks returned from the module to the ladder logic is determined by the value entered in the System Configuration 'Read Block Cnt' register. In this example we have assumed a Read Block Count value of 5.

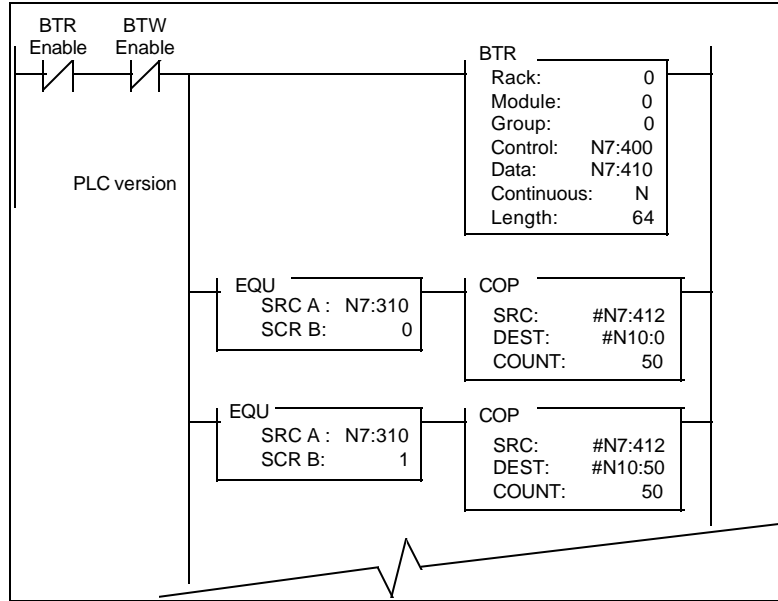


**Read Data Blocks from ROC Module**

*Note that this diagram assumes a Read Block Count value of 5, therefore returning Registers 0 to 249 from the module. This value can be altered as needed depending on the application.*

### 5.1.3 Ladder Logic to Read Module Data

The ladder logic must be programmed to look at the BTR buffer, decode several words, and then take action. The following is an example of such ladder logic:



**Example ladder to transfer data from module**  
 This logic shows a method for moving data from the module to the PLC data table.

### 5.1.4 Slave Error Code Table

The Slave Error Table contains the system information for the ROC module.



The Slave Error Code Table is initialized to zero on power up, and every time the module receives the 255 configuration data block.

The Slave Error Table is a 20 word block. The location of the Error Table is determined by the Slave Error Table Pointer parameter in the Configuration Block. The structure of the data block is as follows:

#### Port 1 Status Codes

Word	Example Addr	Name	Description
0	N10:100	Not Used	
1	N10:101	Not Used	
2	N10:102	Not Used	
3	N10:103	Not Used	
4	N10:104	Not Used	

#### Port 2 Status Codes

Word	Example Addr	Name	Description
5	N10:105	Not Used	
6	N10:106	Not Used	
7	N10:107	Not Used	
8	N10:108	Not Used	
9	N10:109	Not Used	

**System Information**

<b>Word</b>	<b>Example Addr</b>	<b>Name</b>	<b>Description</b>
10-11	N10:110 N10:111	Product Name (ASCII)	These two words represent the product name of the module in an ASCII representation. In the case of the ROC product, the letters ' ROC ' should be displayed when placing the programming software in the ASCII data representation mode.
12-13	N10:112 N10:113	Revision (ASCII)	These two words represent the product revision level of the firmware in an ASCII representation. An example of the data displayed would be '1.00' when placing the programming software in the ASCII data representation mode.
14-15	N10:114 N10:115	Operating System Rev (ASCII)	These two words represent the module's internal operating system revision level in an ASCII representation.
16-17	N10:116 N10:117	Production Run Number (ASCII)	This number represents the 'batch' number that your particular chip belongs to in an ASCII representation.
18-19	N10:118 N10:119	Spare	

All counters in the Slave Error Table will rollover to 0 after reaching 65535

**5.1.5 Master Error Code Table**

The ROC Module monitors the status of all Master port commands. This status is communicated to the processor in the form of a Master Error Code Table, the position of which is controlled by the Master Error Table Pointer in the Communication Configuration setup. Each Master command will generate an Error Code for use by the user.

The Master Error Code Table is initialized to zero on power up, and every time the module receives the 255 configuration data block.

The Error Code Table is a 120 word block. The relationship between the placement of the error codes within the Error Table and the commands is according to the command's relative position in the command list.

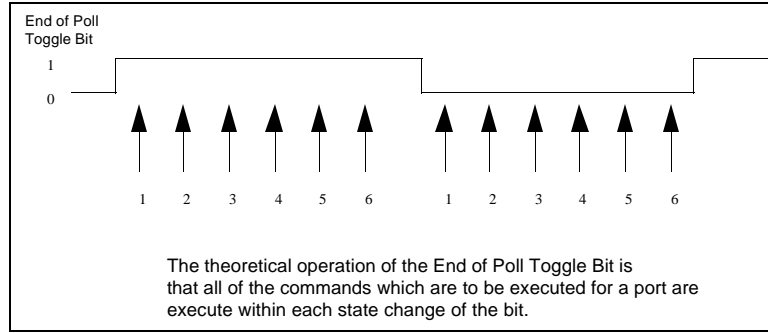
The simplest method for obtaining the Master Error Status Table is to locate it at the end of the application's data map and then read it back into the PLC/SLC data table as part of the regular data. The structure of the Master Error Table is as follows:

<b>Word</b>	<b>Description</b>
0	Command List End of Poll Status
1	Command #1 Error Status
2	Command #2 Error Status
-	
98	Command #98 Error Status
99	Command #99 Error Status
100-120	Future

Where:

**Command List End Of Poll Status:** This register provides an indication of when the Master has completed one cycle through the Command List. A bit in the word will be toggled each time the command list has been completed. The status is indicated for each master port as follows:

<b>Bit</b>	
0	Master Port 1
1	Master Port 2



**Command Error Status:** The Error Status Codes, either received from the slaves, or generated by the module, are placed in the table. See the next section for the meaning of the error codes. The values will be 16 bit values, and should be placed into an integer file. Note that the user application ladder logic controls the placement and use of these registers.

<b>Error Status Table Example</b>										
<b>Master Error Table Pointer = 120</b>										
	<u>Wrd</u> <u>0</u>	<u>Wrd</u> <u>1</u>	<u>Wrd</u> <u>2</u>	<u>Wrd</u> <u>3</u>	<u>Wrd</u> <u>4</u>	<u>Wrd</u> <u>5</u>	<u>Wrd</u> <u>6</u>	<u>Wrd</u> <u>7</u>	<u>Wrd</u> <u>8</u>	<u>Wrd</u> <u>9</u>
N10:120	0	0	8	0	0	0	0	0	0	0
N10:130	0	0	0	0	0	0	0	0	0	0
N10:140	0	0	0	0	0	0	0	0	0	0
N10:150	0	0	0	0	0	0	0	0	0	0
N10:160	0	0	0	0	0	0	0	0	0	0
N10:170	0	0	0	0	0	0	0	0	0	0
N10:180	0	0	0	0	0	0	0	0	0	0
N10:190	0	0	0	0	0	0	0	0	0	0
N10:200	0	0	0	0	0	0	0	0	0	0
N10:210	0	0	0	0	0	0	0	0	0	0
N10:220	0	0	0	0	0	0	0	0	0	0
N10:230	0	0	0	0	0	0	0	0	0	0

*These registers correspond to the registers used in the sample program for PLC-5 in the back of this manual. Your application may require your own specific program. In this case an error code of 8 was generated for command 2 -- all other commands were executed without any errors. Column 0 is used to identify that a master port has reached the end of the command list, and is starting at the top of the Command List*

### 5.1.6 Error Status Codes

The Error Codes returned in the Master Error Code Table reflect the outcome of the commands executed by the module. Note that in all cases, if a zero is returned, there was not an error. Valid Error Status Codes are as follows:

<b>Code</b>	<b>Name</b>	<b>Description</b>
0	All OK	The module is operating as desired.
1	Illegal Function	An illegal function code request is being attempted.
2	Bad Data Address	The address, or the range of addresses, covered by a request from the master are not within allowed limits.
3	Bad Data Value	The value in the data field of the command is not allowed.

Code	Name	Description
4	Incomplete Response Detected	This error indicates that an incomplete response was received to a master query. Often this will indicate that the slave device may be responding too quickly or that there may be excessive noise on the line.
6	Module Busy	The module busy status code is returned when a write command from the master has not yet been completed when a second write command is received.
8	Timeout Error	Communications with the addressed slave have been unsuccessful due to a lack of response from the slave. The Master port will attempt a command three times before moving onto the next command.
10	Buffer Overflow	The receive buffer has overflowed and reset the character count to 0. If this condition occurs try reading fewer parameters at one time.
16	Port Configuration Error	If this value is returned from the module, one or both of the serial ports have been misconfigured. To determine the exact source of the problem, verify the following: <ul style="list-style-type: none"> <li>- Parity Configuration</li> <li>- Stop Bit Configuration</li> <li>- Baud Rate Configuration</li> <li>- Start Input Register Address</li> <li>- Start Output Register Address</li> <li>-</li> </ul>
18	System Configuration Error	If this error is returned from the module, one of the system configuration parameters has been detected out of range. To determine the source, verify the following: <ul style="list-style-type: none"> <li>- Read Block Count &lt;= 80</li> <li>- Write Block Count &lt;=80</li> <li>- Command Block Count &lt;= 20</li> <li>- Slave Error Pointer &lt;= 3850</li> <li>- Master Error Pointer &lt;= 3880</li> <li>-</li> </ul>
20 + ROC Error Code	Error Codes Returned From ROC Slave	The error code returned from the ROC slave will be offset by 20. If an error of 2 is returned from the slave then the error will be stored in the module as 22.
254	Checksum Error	The slave determined that the message checksum was in error, and therefore discarded the message.
255	TX Hardware Timeout	A transmit timeout condition has occurred indicating that the module was not able to transmit the command. Verify that the RTS-CTS jumper on the port is still connected.

## 5.2 Decoding Command Done and Command Error Bits - Master Mode

The Command Done and Command Error bits are returned for use in the ladder logic program during every data block transfer (BTR Block ID 0 to 79). These bits can be used by the ladder logic to keep track of command execution or to disable commands when a command is configured in the Command Control Mode (See Section 4.5).

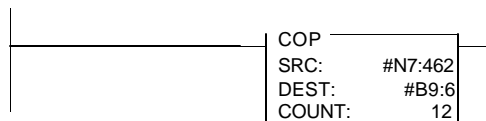
### 5.2.1 The Block Structure

The structure of the Done and Error bits as they are returned in the BTR Block Transfer buffer is as follows:

Word	Name	Description														
0	BTR Block ID	When the BTR Block ID value is between 0 and 79 the BT Buffer contains Command Done and Command Error bits, as outlined below.														
1	BTW Block ID	Same as above description.														
2-51	Data	Module data, as outlined above.														
52-57	Cmd Done Bits	<p>These registers contain Done Bit flags for each command in the command list, up to the first 96 commands. The Done Bits are bit mapped into the words depending on their relative position in the Command List. The mapping within the Done Bits is as follows:</p> <table border="0"> <thead> <tr> <th><u>Word</u></th> <th><u>Cmds</u></th> </tr> </thead> <tbody> <tr> <td>52</td> <td>1 to 16</td> </tr> <tr> <td>53</td> <td>17 to 32</td> </tr> <tr> <td>54</td> <td>33 to 48</td> </tr> <tr> <td>55</td> <td>49 to 64</td> </tr> <tr> <td>56</td> <td>65 to 80</td> </tr> <tr> <td>57</td> <td>81 to 96</td> </tr> </tbody> </table> <p>Example: Word 52 bit 0 is Command #1</p>	<u>Word</u>	<u>Cmds</u>	52	1 to 16	53	17 to 32	54	33 to 48	55	49 to 64	56	65 to 80	57	81 to 96
<u>Word</u>	<u>Cmds</u>															
52	1 to 16															
53	17 to 32															
54	33 to 48															
55	49 to 64															
56	65 to 80															
57	81 to 96															
58-63	Cmd Error Bits	<p>These registers contain Error Bit flags for each command in the command list, up to the first 96 commands. The Error Bits are bit mapped into the words depending on their relative position in the Command List. The mapping within the Done Bits is as follows:</p> <table border="0"> <thead> <tr> <th><u>Word</u></th> <th><u>Cmds</u></th> </tr> </thead> <tbody> <tr> <td>58</td> <td>1 to 16</td> </tr> <tr> <td>59</td> <td>17 to 32</td> </tr> <tr> <td>60</td> <td>33 to 48</td> </tr> <tr> <td>61</td> <td>49 to 64</td> </tr> <tr> <td>62</td> <td>65 to 80</td> </tr> <tr> <td>63</td> <td>81 to 96</td> </tr> </tbody> </table> <p>Example: Word 52 bit 0 is Command #1</p>	<u>Word</u>	<u>Cmds</u>	58	1 to 16	59	17 to 32	60	33 to 48	61	49 to 64	62	65 to 80	63	81 to 96
<u>Word</u>	<u>Cmds</u>															
58	1 to 16															
59	17 to 32															
60	33 to 48															
61	49 to 64															
62	65 to 80															
63	81 to 96															

### 5.2.2 Ladder Logic

A simple rung of logic can be entered to move the Done and Error bits from the BTR buffer to the PLC/SLC data table. An example follows:



Copy the Command Done and Error bits from the BTR buffer to the data table.

## 6 Fisher ROC Command Configuration

The ProSoft Technology ROC Fisher ROC Master communication driver supports several data read and write commands. When configuring a Master port, the decision on which command to use is made depending on the type of data being addressed, and the level of Fisher ROC support in the slave equipment.

### 6.1 Fisher ROC Commands

The ROC module supports a command subset of the Fisher ROC Specification. The following sections detail the different commands supported by the module.

Opcode	Command	Master Driver Comments
8	Set New Time and Date	<p><u>Count</u>: Number of bytes to be sent. Valid count values are 6 – 9.</p> <p><u>Source Addr</u>: Starting word address in the module where the write data is stored. The data starting at this address will be sent to the slave when an opcode 8 is written. Enter a 0 to output parameter data from internal database address 0.</p>
10	Send Data From Configurable Opcode Tables	<p><u>Count</u>: Number of bytes to be sent. Valid count value is 3.</p> <p><u>Source Addr</u>: Starting word address in the module where the write data is stored. The data starting at this address will be sent to the slave when an opcode 10 is written. Enter a 0 to output parameter data from internal database address 0.</p> <p><u>Dest Addr</u>: Starting word address in the module's Register Memory in which the received data should be placed, starting with word 0.</p>
120	Send Pointers for Alarm, Event, and History Logs	<p><u>Count</u>: Number of bytes to be sent. Valid count value is 0.</p> <p><u>Source Addr</u>: N/A.</p> <p><u>Dest Addr</u>: Starting <u>word address</u> in the module's Register Memory in which the received data should be placed, starting with word 0.</p>
128	Send Archived Daily and Hourly Data for the Currently Selected Day and Month	<p><u>Count</u>: Number of bytes to be sent. Valid count value is 3.</p> <p><u>Source Addr</u>: Starting word address in the module where the write data is stored. The data starting at this address will be sent to the slave when an opcode 128 is written. Enter a 0 to output parameter data from internal database address 0.</p> <p><u>Dest Addr</u>: Starting word address in the module's Register Memory in which the received data should be placed, starting with word 0.</p>



Opcode	Command	Master Driver Comments
130	Send Archived Hourly and Daily Data for Specified History Point Starting at Specified History Pointer	<p><u>Count</u>: Number of bytes to be sent. Valid count value is 4.</p> <p><u>Source Addr</u>: Starting word address in the module where the write data is stored. The data starting at this address will be sent to the slave when an opcode 128 is written. Enter a 0 to output parameter data from internal database address 0.</p> <p><u>Dest Addr</u>: Starting word address in the module's Register Memory in which the received data should be placed, starting with word 0.</p>
180	Send Specified Parameters	<p><u>Count</u>: Number of data points to be read. Count must be greater than 0.</p> <p><u>Source Addr</u>: Starting word address in the module from which the TLP data to be sent to the slave should be read. The first word is Point type, the second is Point/Logic number and the third is Parameter number. Enter a 0 to output parameter data from internal database address 0.</p> <p><u>Dest Addr</u>: Starting word address in the module's Register Memory in which the received data should be placed, starting with word 0.</p> <p><u>Type</u>: Controls TLP stripping (See Section 4 for details). Set to 0 for stripping and 1 for no stripping.</p>
181	Set Specified Parameters	<p><u>Count</u>: Number of points to be written. Count must be greater than 0.</p> <p><u>Source Addr</u>: Starting word address in the module from which the TLP data to be sent to the slave should be read. The first word is Point type, the second is Point/Logic number and the third is Parameter number. Enter a 0 to output parameter data from internal database address 0.</p> <p><u>Dest Addr</u>: Starting word address in the module from which the data to be written to the module is obtained. The first word corresponds to the TLP in the first location of Source Addr.</p> <p><u>Type</u>: Type represents the number of byte in the data type. All of the data written with one command must be of the same type. Valid values are 4=floating-point, 2=integer, and 1=byte.</p>

### 6.1.1 Opcode 180 and 181 Examples

The first command example is using an opcode 180 to read 2 data points. The second writes data to the same two data points.

The commands are shown below:

	0 <u>PORT</u> <u>NUM</u>	1 <u>SLV</u> <u>UNIT</u>	2 <u>SLV</u> <u>GROUP</u>	3 <u>OP</u> <u>CODE</u>	4 <u>CNT</u>	5 <u>SRC</u> <u>ADDR</u>	6 <u>DEST</u> <u>ADDR</u>	7 <u>TYPE</u>	8 <u>POLL</u> <u>TIME</u>	9 <u>STORE</u> <u>FWD</u>
N7:50	1	25	2	180	2	0	50	0	0	0
N7:60	1	25	2	181	2	0	100	2	0	0

The TLP for the 2 points are stored the module starting at address 0, set by the SRC ADDR of the command. In this example we will assume that N10:0 maps to address 0 of the module database. This example is set up for reading and writing to Point type 3 (Analog Input), logical numbers 0 and 1, and parameter 2 (Scan Period). The value in N10:0 is the point type for the first data point. N10:1 holds the logical number and N10:2 contains the parameter number for the first data point. N10:3 – N10:5 contain the TLP values for the second point.

N10:0	N10:1	N10:2	N10:3	N10:4	N10:5	N10:6	N10:7	N10:8	N10:9
3	0	2	3	1	2	0	0	0	0

For opcode 180 the received data will have the TLP stripped out and the data will be stored in the module database starting at address 50 (DEST ADDR). If type is set to 1, then the TLP will be stored with each data value. This would take 3 words for the TLP and the data will be stored immediately following.

For opcode 181 the same TLP will be used as in the previous example because the SRC ADDR is also set to address 0. The data values to write to the ROC must be stored in the module database. The data will be fetched starting from the address set by DEST ADDR. In this example the starting address is 100.

N10:100	N10:101	N10:102	N10:103	N10:104	N10:105	N10:106	N10:107	N10:108	N10:109
500	1000	0	0	0	0	0	0	0	0

The data value of 500 will be written to the first data point and 1000 to the second.

### 6.1.2 Point Type Support

The point types supported by the Fisher ROC may contain mixed data types. In the ROC module the point data is treated as an array of words. Byte data will be placed in the Least Significant byte of a word. Word data will be presented as words. Floating point data will be presented as two consecutive words.

*ROC Opcode 120 Example*  
*Byte 0 is at the destination address*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
INT0	INT1	INT0	INT1	INT0	INT1	INT0	INT1	INT0	INT1
Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	Byte 16	Byte 17	Byte 18	Byte 19
INT0	INT1	0	BIN0	0	BIN0	0	BIN0	0	BIN0
Byte 20	Byte 21	Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29
INT0	INT1	INT0	INT1	0	BIN0	0	BIN0	0	BIN0
Byte 30	Byte 31	Byte 32	Byte 33	Byte 34	Byte 35				
0	BIN0	0	BIN0	0	BIN0				

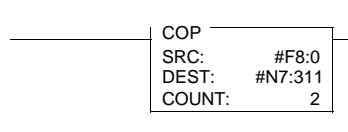
AC = ASCII    INT = INT16    FLP = Floating Point    BIN = Binary

### 6.2 Floating Point Support

The movement of floating point data between the ROC module and other devices is easily accomplished as long as the device supports IEEE 754 Floating Point format. This IEEE format is a 32-bit single precision floating point format.

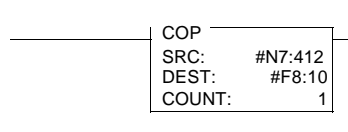
The programming necessary to move the floating point data is to take advantage of the COP command that exists in the PLC and the SLC. The COP command is unique in the PLC/SLC data movement commands in that it is an untyped function, meaning that no data conversion is done when moving data between file types (i.e., it is an image copy not a value copy).

The structure of the COP command to move data from a Floating Point file into an integer file (something you would do to move floating point values to the module) is as follows:



*This command will move one floating point value in two 16 bit integer images to the integer file. For multiple floating point values simply increase the count field by a factor of 2 per floating point value.*

The structure of the COP command to move data from an Integer file to a Floating Point file (something you would do to receive floating point values from the module) is as follows:



*This command will move two 16 bit integer registers containing one floating point value image to the floating point file. For multiple values simply increase the count field.*

### 6.3 Store And Forward

The Store and Forward function is achieved by setting an address value of greater than 0 in location 9 of a command. This address is the starting word of the communication path for ROC to ROC communications. A value of 0 disables Store and Forward. When Store and Forward is used the module wraps the opcode 24 and communication path information around the command before it is sent out of the communication port.

*Example: The Store and Forward Source Address of a command is N11:40. Communication Path data is stored as the following:*

Unit	Group	Unit	Group	Unit	Group	Unit	Group	Unit	Group
N11:40	N11:41	N11:42	N11:43	N11:44	N11:45	N11:46	N11:47	N11:48	N11:49

## 7 Diagnostics and Troubleshooting

Several hardware diagnostics capabilities have been implemented using the LED indicator lights on the front of the module. The following sections explain the meaning of the individual LEDs for both the PLC and the SLC platforms.

### 7.1 3100 PLC Platform LED Indicators









The PLC platform ROC product is based on the ProSoft CIM hardware platform. The following table documents the LEDs on the 3100-ROC hardware and explains the operation of the LEDs.

ProSoft CIM Card		
ACTIVE	↘ ↘	FLT
CFG	↘ ↘	BPLN
ERR1	↘ ↘	ERR2
TXD1	↘ ↘	TXD2
RXD1	↘ ↘	RXD2

ProSoft CIM	Color	Status	Indication
ACT	Green	Blink (Fast)	<u>Normal state</u> : The module is operating normally and successfully Block Transferring with the PLC
		On	The module is receiving power from the backplane, but there may be some other problem
		Off	The module is attempting to Block Transfer with the PLC and has failed. The PLC may be in the PGM mode or may be faulted
FLT	Red	Off	<u>Normal State</u> : No system problems are detected during background diagnostics
		On	A system problem was detected during background diagnostics. Please contact factory for technical support
CFG	Green	Off	<u>Normal state</u> : No configuration related activity is occurring at this time
		Blink	This light blinks every time a Module Configuration block (ID = 255) is received from the processor ladder logic
		On	The light is on continuously whenever a configuration error is detected. The error could be in the Port Configuration data or in the System Configuration data. See Section 4 for details
BPLN	Red	Off	<u>Normal State</u> : When this light is off and the ACT light is blinking quickly, the module is actively Block Transferring data with the PLC
		On	Indicates that Block Transfers between the PLC and the module have failed.( Not activated in the initial release of the product)
ERR1 ERR2	Amber	Off	<u>Normal State</u> : When the error LED is off and the related port is actively transferring data, there are no communication errors
		Blink	Periodic communication errors are occurring during data communications. See Section 4 to determine the error condition
		On	This LED will stay on under several conditions: ?? CTS input is not being satisfied ?? Port Configuration Error ?? System Configuration Error ?? Unsuccessful comm on ROC slave Recurring error condition on ROC master
Tx1 Tx2	Green	Blink	The port is transmitting data.
Rx1 Rx2	Green	Blink	The port is receiving data

## 7.2 3150 SLC Platform LED Indicators

The following table documents the LEDs on the 3150-ROC hardware and explains the operation of the LEDs.

COMMUNICATIONS			
	ACT		FAULT
	CFG		BPLN
	PRT1		ERR1
	PRT2		ERR2

LED Name	Color	Status	Indication
ACT	Green	Blink (Fast)	<u>Normal state</u> : The module is operating normally and successfully Block Transferring with the SLC
		On	The module is receiving power from the backplane, but there may be some other problem
		Off	The module is attempting to Block Transfer with the SLC and has failed. The SLC may be in the PGM mode or may be faulted
FLT	Red	Off	<u>Normal State</u> : No system problems are detected during background diagnostics
		On	A system problem was detected during background diagnostics. Please contact factory for technical support
CFG	Green	Off	<u>Normal state</u> : No configuration related activity is occurring at this time
		Blink	This light blinks every time a Module Configuration block (ID = 255) is received from the processor ladder logic
		On	The light is on continuously whenever a configuration error is detected. The error could be in the Port Configuration data or in the System Configuration data. See Section 4 for details
BPLN	Red	Off	<u>Normal State</u> : When this light is off and the ACT light is blinking quickly, the module is actively Block Transferring data with the SLC
		On	Indicates that Block Transfers between the SLC and the module have failed
ERR1 ERR2	Amber	Off	<u>Normal State</u> : When the error LED is off and the related port is actively transferring data, there are no communication errors
		Blink	Periodic communication errors are occurring during data communications. See Section 4 to determine the error condition
		On	This LED will stay on under several conditions: ?? CTS input is not being satisfied ?? Port Configuration Error ?? System Configuration Error ?? Unsuccessful comm on ROC slave ?? Recurring error condition on ROC master
PRT1 PRT2	Green	Blink	The port is communicating, either transmitting or receiving data

## 7.3 Troubleshooting - General

In order to assist in the troubleshooting of the module, the following tables have been put together to assist you. Please use the following to help in using the module, but if you have additional questions or problems please do not hesitate to contact us.

The entries in this section have been placed in the order in which the problems would most likely occur after powering up the module.

Problem Description	Steps to take
BPLN light is on (SLC)	<p>The BPLN light comes on when the module does not think that the SLC is in the run mode (i.e., SLC is in PGM or is Faulted). If the SLC is running then verify the following:</p> <ul style="list-style-type: none"> <li>?? Verify the SLC Status File to be sure the slot is enabled</li> <li>?? The Transfer Enable/Done Bits (I/O Bits 0 for the slot with the module) must be controlled by the ladder logic. See Section 2.x for details or the example ladder logic in the Appendix.</li> <li>?? If the ladder logic for the module is in a subroutine file verify that there is a JSR command calling the SBR</li> </ul>
CFG light does not clear after power up (no ERR LED)	<p>The 255 BTW Block ID number is not being detected by the module. This could be due to a Block Transfer failure (PLC) or to an error in the ladder logic preventing the 255 value from being moved to the BTW buffer</p>
CFG light does not clear after power up (w/ ERR LED)	<p>If the BPLN light has been cleared, then several of the Port and System configuration values are value checked by the module to be sure that legal entries have been entered in the data table. Verify the Error Status Table for an indication of a configuration error.</p>
CFG light toggles	<p>Under normal conditions, the CFG LED will clear immediately after receipt. If the CFG light toggles, this usually indicates that the logic condition which places the 255 Block ID value in the BTW buffer is not being cleared. Check the ladder logic to be sure that the condition moving the 255 value is not held true.</p>
Module is not transmitting	<p>Presuming that the processor is in run, verify the following:</p> <ul style="list-style-type: none"> <li>?? CTS input is not satisfied (check RTS/CTS jumper)</li> <li>?? Check Error Status codes for 255 code. If so see next problem</li> <li>?? If in slave mode, verify the slave address being requested from the Host</li> <li>?? If in master mode, verify the command list configuration and that the Command List is being moved into the module (i.e., check the Command Block Cnt and associated ladder logic)</li> </ul>
Error Code 255 in Status Table	<p>This is caused by only one thing, a missing CTS input on the port. If a cable is connected to the port, then verify that a jumper has been installed between the RTS and CTS pins. If so then there may be a hardware problem.</p>
Overwriting data blocks	<p>This condition normally occurs when it is forgotten that the BTW Block ID value is being manipulated by the module, and that it always starts at 0. Please verify that the configuration of the module (Read and Write Block Counts) is not causing data from the PLC/SLC to overwrite data being returned from the module. A simple method for verifying this is to perform a histogram on the BTW Block ID register.</p>
Data swapping is occurring (3100 only)	<p>Under several circumstances data swapping in the module has occurred. This swapping has always been associated with the 8/16 pt jumper on the back of the card. Please verify that the jumper is in the 8pt position</p>
New configuration values are not being accepted by the module	<p>In order for new values to be moved to the module a Block Transfer Write with a Block ID of 255 must be transmitted to the module. The 'User Config Bit' in the example logic accomplishes this. In the example logic the bit must either be set in the data table manually or the module must be powered down/reset.</p> <p>In order to download the configuration upon transitioning from PGM to RUN, simply add a run to set the 'User Config Bit' based on the First Scan Status Bit (S1:1/15)</p>

Problem Description	Steps to take
Error Codes being returned in locations with no commands (Master Configuration)	<p>Be sure that the Command Block Count configuration value is setup correctly. There should be one branch of logic in the Write Rung corresponding to each Command Block to be written (i.e., a Command Block Count of 2 should have two branches of logic to handle BTW Block IDs 80 and 81).</p> <p>If the Command Block Count configuration value exceeds the number of branches in logic, the Command List is inadvertently being duplicated. To resolve the issue, either add more branches of logic or reduce the Command Block Count value to match the number of BTW logic branches.</p>
RX1 or RX2 on continuously (3100 only)	<p>The TX and RX LEDs on the module are tied to the hardware state of the ports (i.e., are not controlled directly by firmware). When the RX LED is on continuously is normally indicates that the polarity of the cable connection to the port is swapped.</p> <p>This is particularly true in RS-485 and RS-422 modes.</p>

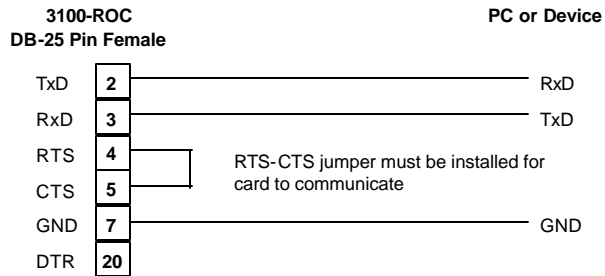
## 8 Cable Connection Diagrams

The following diagrams show the connection requirements for the ports on the 3100 and 3150 modules.

### 3100-ROC Module

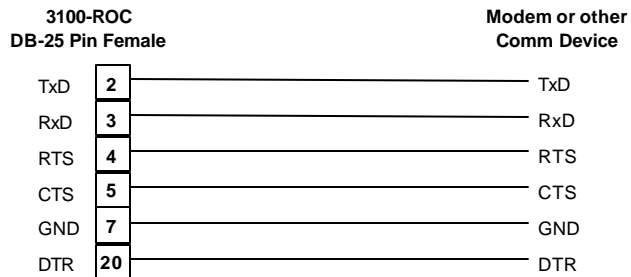
#### **RS-232 w/ No Hardware Handshaking**

Port Connection with another communication port



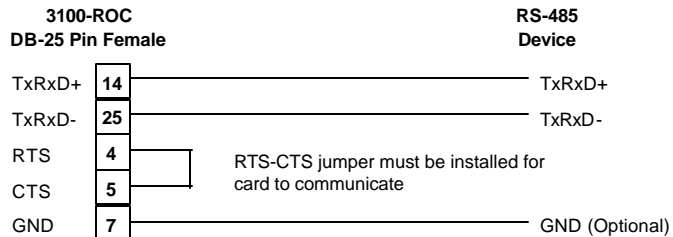
#### **RS-232 w/ Hardware Handshaking**

Port Connection with a modem or other similar device



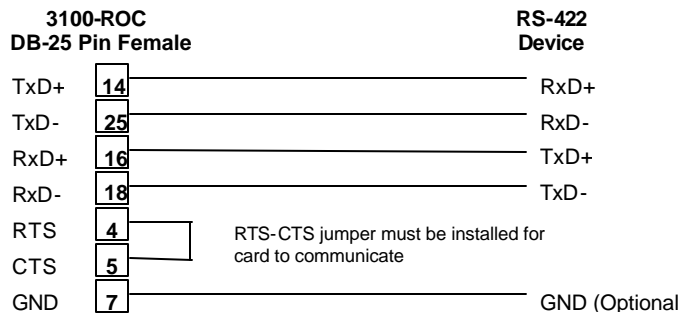
#### **RS-485/2-Wire Connection**

The jumper on the module must be set in the RS-485 position for all 2-wire applications



#### **RS-422/4-Wire Connection**

The jumper on the module must be set in the RS-422 position for all 4-wire applications

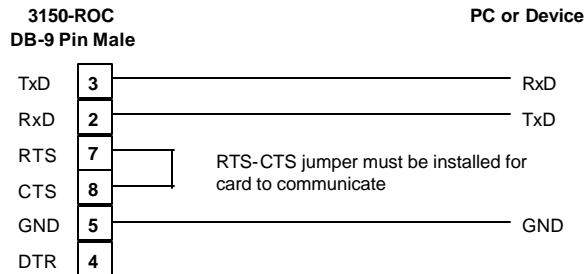




### 3150-ROC Module

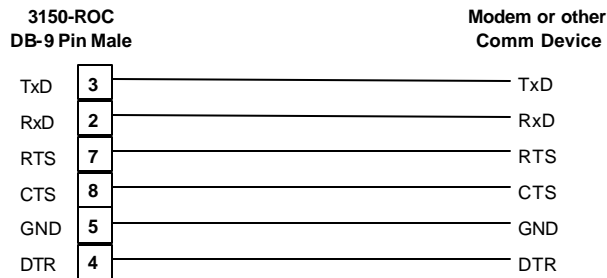
#### RS-232 w/ No Hardware Handshaking

Port Connection with another communication port



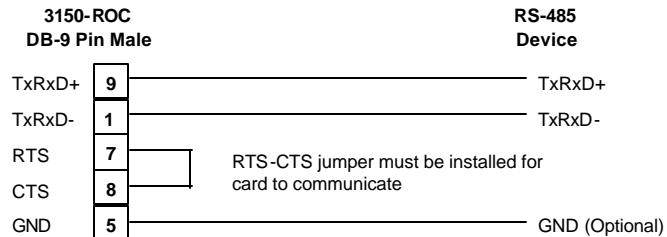
#### RS-232 w/ Hardware Handshaking

Port Connection with a modem or other similar device



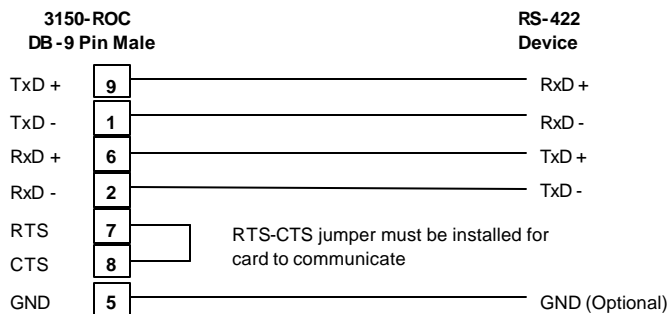
#### RS-485/2-Wire Connection

The jumper on the module must be set in the RS-485 position for all 2-wire applications



#### RS-422/4-Wire Connection

The jumper on the module must be in the RS-422 position for all 4-wire applications



#### RS-485 and RS-422 Tip

If communication in the RS-422/RS-485 mode does not work at first, despite all attempts, try switching termination polarities. Some manufacturers interpret +/- and A/B polarities differently.

## A Support, Service and Warranty

### Technical Support

ProSoft Technology survives on its ability to provide meaningful support to its customers. Should any questions or problems arise, please feel free to contact us at:

**Factory/Technical Support**  
ProSoft Technology, Inc.  
9801 Camino Media, Suite 105  
Bakersfield, CA 93311  
(661) 664-7208  
(800) 326-7066  
(661) 664-7233 (fax)

E-mail address: [prosoft@prosoft-technology.com](mailto:prosoft@prosoft-technology.com)

Web Site: <http://www.prosoft-technology.com>

FTP Site <ftp://ftp.prosoft-technology.com>

Before calling for support, please prepare yourself for the call. In order to provide the best and quickest support possible, we will most likely ask for the following information (you may wish to fax it to us prior to calling):

1. Product Version Number
2. Configuration Information
  - Communication Configuration
  - Master Command List
  - Jumper positions
3. System hierarchy
4. Physical connection information
  - RS-232, 422 or 485
  - Cable configuration
5. Module Operation
  - Block Transfers operation
  - LED patterns

An after-hours answering system (on the Bakersfield number) allows pager access to one of our qualified technical and/or application support engineers at any time to answer the questions that are important to you.

### Module Service and Repair

The ROC card is an electronic product, designed and manufactured to function under somewhat adverse conditions. As with any product, through age, misapplication, or any one of many possible problems, the card may require repair.

When purchased from ProSoft Technology, the module has a one year parts and labor warranty according to the limits specified in the warranty. Replacement and/or returns should be directed to the distributor from whom the product was purchased. If you need to return the card for repair, it is first necessary to obtain an RMA number from ProSoft Technology. Please call the factory for this number and display the number prominently on the outside of the shipping carton used to return the card.

### General Warranty Policy

ProSoft Technology, Inc. (Hereinafter referred to as ProSoft) warrants that the Product shall conform to and perform in accordance with published technical specifications and the accompanying written materials, and shall be free of defects in materials and workmanship, for the period of time herein indicated, such warranty period commencing upon receipt of the Product.

This warranty is limited to the repair and/or replacement, at ProSoft's election, of defective or non-conforming Product, and ProSoft shall not be responsible for the failure of the Product to perform specified functions, or any other non-conformance caused by or attributable to: (a) any misapplication of misuse of the Product; (b) failure of Customer to adhere to any of ProSoft's

specifications or instructions; (c) neglect of, abuse of, or accident to, the Product; or (d) any associated or complementary equipment or software not furnished by ProSoft.

Limited warranty service may be obtained by delivering the Product to ProSoft and providing proof of purchase or receipt date. Customer agrees to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to ProSoft, and to use the original shipping container or equivalent. Contact ProSoft Customer Service for further information.

**Limitation of Liability**

EXCEPT AS EXPRESSLY PROVIDED HEREIN, PROSOFT MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, WITH RESPECT TO ANY EQUIPMENT, PARTS OR SERVICES PROVIDED PURSUANT TO THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER PROSOFT OR ITS DEALER SHALL BE LIABLE FOR ANY OTHER DAMAGES, INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION IN CONTRACT OR TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), SUCH AS, BUT NOT LIMITED TO, LOSS OF ANTICIPATED PROFITS OR BENEFITS RESULTING FROM, OR ARISING OUT OF, OR IN CONNECTION WITH THE USE OR FURNISHING OF EQUIPMENT, PARTS OR SERVICES HEREUNDER OR THE PERFORMANCE, USE OR INABILITY TO USE THE SAME, EVEN IF PROSOFT OR ITS DEALER'S TOTAL LIABILITY EXCEED THE PRICE PAID FOR THE PRODUCT.

Where directed by State Law, some of the above exclusions or limitations may not be applicable in some states. This warranty provides specific legal rights; other rights that vary from state to state may also exist. This warranty shall not be applicable to the extent that any provisions of this warranty is prohibited by any Federal, State or Municipal Law that cannot be preempted.

**Hardware Product Warranty Details**

Warranty Period : ProSoft warranties hardware product for a period of one (1) year.

Warranty Procedure : Upon return of the hardware Product ProSoft will, at its option, repair or replace Product at no additional charge, freight prepaid, except as set forth below. Repair parts and replacement Product will be furnished on an exchange basis and will be either reconditioned or new. All replaced Product and parts become the property of ProSoft. If ProSoft determines that the Product is not under warranty, it will, at the Customer's option, repair the Product using current ProSoft standard rates for parts and labor, and return the Product freight collect.

## B Product Specifications

The 3100/3150-ROC ("Fisher ROC Communication Module") product family allows Allen-Bradley 1771 and 1746 I/O compatible processors to easily interface with other Fisher ROC protocol compatible devices as a Fisher ROC Master.

The ROC product includes the following standard features:

### General Specifications

- ? Two fully configurable serial ports, each capable of supporting Fisher ROC Master. Available configurations include:
 

<u>Port Configurations</u>	<u>Port 1</u>	<u>Port 2</u>
Master-Master	Master	Master
- ? Support for the storage and transfer of up to 4000 registers to the PLC /SLC data tables
- ? Support movement of binary, integer, ASCII, and floating point data types
- ? Memory mapping is completely user definable through data table configuration
- ? RS-232C handshaking for SCADA radio/modem applications
- ? RS-422/RS-485 compatible for multi-drop applications with up to 32 slaves per port
- ? Satellite and Packet Radio support with a configurable Inter-character Timeout available per port
- ? Software configuration (From processor ladder logic)
 

Slave Addr	:	0 to 255
Parity	:	None, odd, or even,
Stop Bit	:	1 or 2
Baud Rate	:	300 TO 38,400
RTS to TxD	:	0-65535 ms, 1 ms resolution
RTS Off	:	0-65535 ms, 1 ms resolution
Timeout	:	0-65535 ms, 1 ms resolution
- ? Response time  
The Fisher ROC Master protocol driver is written in Assembly and in a compiled higher level language. As such, the interrupt capabilities of the hardware are fully utilized to minimize delays, and to optimize the product's performance

### Fisher ROC Master Specifications

- ? Protocol modes:
  - RTU mode with CRC-16 error checking
- ? Supported Fisher ROC Function codes:
 

8	Set New Time and Date
10	Send Data From Configurable Opcode Tables
24	Store and Forward
120	Send Pointers for Alarm, Event, and History Logs
128	Send Archived Daily and Hourly Data for the Currently Selected Day and Month
130	Send Archived Hourly and Daily Data for Specified History Point Starting at Specified History Pointer
180	Send Specified Parameters
181	Set Specified Parameters
- ? Supports up to 100 Command List entries, each individually configurable with the following parameters:
  - Port/Mode Selection
  - Slave Unit
  - Slave Group
  - Opcode
  - Number of values to transfer
  - Source data address
  - Destination data address
  - Swap type
  - Polling Time
- ? Command Control Mode

Allows individual command execution control to be done in ladder logic enabling a list of commands to be executed based on events in the PLC/SLC

- ? Individual command 'Done' and 'Error' bits available
- ? Individual Command Error Statue codes returned to the ladder processor

**Hardware Specifications**

- ? Backplane Current Load :
  - 3100 : 0.65 A
  - 3150 : 0.15 A at 5 V
  - 0.04 A at 24 V
- ? Operating Temperature : 0 to 60 °C
- ? Storage Temperature : -40 to 85 °C
- ? Connections :
  - 3100 : 2 - DB25 Female Connectors
  - 3150 : 2 - DB9 Male Connectors

## C Jumper Configurations

### Hardware Overview

When purchasing the ROC product, there are two available configurations. These choices are as follows:

	ProSoft Cat Num	Description
	<b>PLC</b>	<b>SLC</b>
Module provided by ProSoft	3100	3150

When purchasing the module from ProSoft Technology, the jumper configurations will have been factory set to default positions for testing prior to shipment.

### Module Jumper Configurations

The following section details the available jumper configurations for the 1771 and 1746 platform solutions. As needed, differences between the module based solutions and the firmware based solutions are highlighted.

#### 3100 for the 1771 Platform

Following are the jumper positions for the ProSoft Technology 3100-ROC module:

Jumper	3100
JW1	N/A
JW2	N/A
JW3	N/A
JW4	Flash Pgm/Run Mode
JW5	8 Pt
JW6	Not Used
JW7	Enabled
JW8	Port 2 RS232/422/485 config
JW9	Port 1 RS232/422/485 config

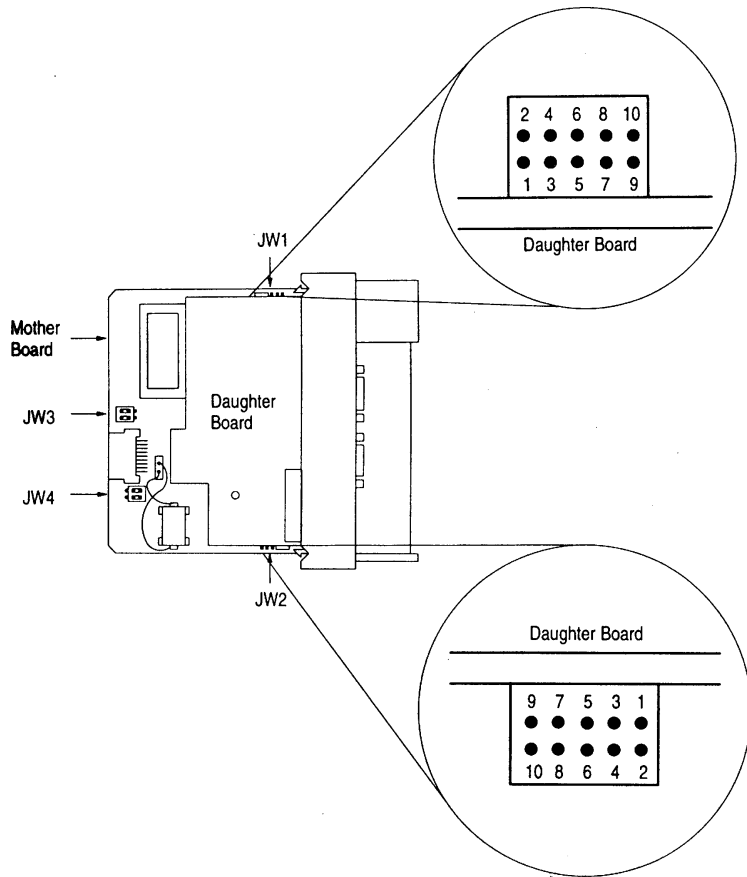
- JW4 Flash Pgm/Run Mode Select** **Run Position**  
 The position of this jumper should only be changed if needing to reprogram the ROC FLASH memory. This will only need to be done if the module is to be upgraded in the field to a later version of firmware.
- JW5 Backplane 8/16 point** **8 Point**  
 The module should be operated in the 8 point configuration unless specifically directed otherwise by the factory.
- JW7 Battery Enable / Disable** **Enabled**  
 This jumper should be placed in the Enabled position when the module is powered up. Although not critical to the operation of the module, this will back up some data registers in the module during a power failure or reset.
- JW8/9 RS Configuration for Port 1 and 2** **RS-232,422,485**  
 The default from factory is RS-232, but all options are supported by the ROC firmware

#### 3150 for the 1746 Platform

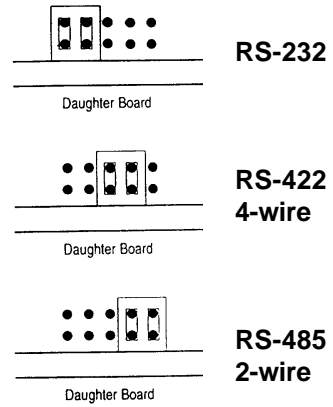
Following are the jumper positions for the ProSoft Technology 3150-ROC module :

Jumper	3150-ROC
JW1	As Needed
JW2	As Needed
JW3	N/A
JW4	N/A

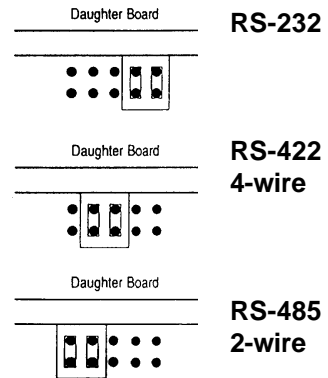
- JW1/2 RS configuration for port 1 and 2** **RS-232 Position**  
 The default from factory is RS-232, but RS-422 and RS-485 are supported by the firmware and hardware. See the following diagram:



**Jumper JW1 Settings**



**Jumper JW2 Settings**



## **D Product Revision History**

03/08/00	Revision 1.00 - 1 - Initial release of product
08/16/00	Revision 1.04 – 5 - Incorporates changes from byte oriented data to word oriented data. The type function was modified to specify stripping of TLP from data or storing raw TLP and data. Integers and floating-point numbers are automatically swapped in order to be stored correctly in the PLC. Changed Store and Forward to use a pointer to the Communication Path header data.

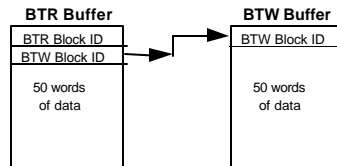


## E Read, Write and Command Block Count Values usage

As part of the configuration process, the User is able to configure several parameters in the Communication Configuration Data Block, which have a strong impact on how the module transfers data with the PLC/SLC processor.

### Overview

As shown in Section 4 and 5 of the manual the BTR buffer contains the BTR and the BTW Block ID numbers. The BTR Block ID is used to identify the data contents, while the BTW Block ID is used by the ladder logic to determine which data to move to the module. Diagrammatically, the relationship is as follows:



### Configuration Parameters

Three parameters which are important to the transfer of data are:

**Read Block Count:** This value represents the number of 50 word data blocks which are to be transferred from the ROC Module to the processor. The blocks returned from the module start at the value entered in the Read\_Block\_Start register and increments from there

**Write Block Count:** This value represents the number of 50 word data blocks which are to be transferred from the processor to the ROC Module.

**Command Block Count:** This value represents the number of 50 word Command Blocks which are to be transferred from the processor to the ROC Module.

These values are used by the module in order to determine how the BTW and BTR Block ID Codes are to be manipulated. Part of the functionality that the module provides is to control the incrementing and resetting of the BTR and BTW Block ID codes. This was done in the interest of limiting the amount of ladder logic required to support the module.

### Module Operation

As a result of the configuration parameters entered, the module will cycle through the range of BTW and BTR Blocks. The cycle is based on the following equations:

#### BTW Block ID

```

if ( BTW Block ID >= Write_Block_Cnt ) then BTW Block ID = 80
elseif( BTW Block ID >= 80 + Command_Block_Cnt) then BTW Block ID = Write_Block_Start
else BTW block ID = BTW block ID + 1
  
```

#### BTR Block ID

```

if ( BTR Block ID >= Read_Block_Cnt ) then BTR Block ID = Read_Block_Start
else BTR block ID = BTR block ID + 1
  
```

As an example, assume that we are configured with the following values:

```

Read_Block_Cnt      4
Write_Block_Cnt     1
Command_Block_Cnt   2
Read_Block_Start    1
Write_Block_Start    0
  
```

These configuration values would lead to the following cycle of Block ID codes:

BTW Block ID	BTR Block ID
0	1
80	2
81	3
0	4
80	1
81	2
0	3

Note that there is no implicit relationship between the absolute value in the BTW and the BTR Block ID.

## F Example Ladder Logic

The following example logic has been provided to assist you in developing applications more effectively. These examples are provided on our FTP site <ftp://ftp.prosoft-technology.com>.

### **Master Mode Examples**

#### **Example #1: Master Mode - Basic Application**

ROC5EX1M	PLC 5
ROC3EX1M	SLC 5/03

#### **Example #2: Master Mode w/ Command Control**

ROC5EX2M	PLC 5
ROC3EX2M	SLC 5/03