CDI-320

# Installation & Start-up Manual

## CDI-320 Allen-Bradley™ Connection

CDI320-US-04
EFFECTIVE 4/1/94

ABB Drives

ABB

# CDI-320
# Allen-Bradley Connection
# For ACS 500, ACH 500, Series B

# **Installation & Start-up Manual**

CDI-320-US-04

EFFECTIVE 4/1/94

## General Safety Instructions

*Warnings* in this manual appear in either of two ways:

- *Dangerous voltage warnings*, preceded by a Dangerous Voltage symbol, indicate the presence of voltages which may cause death or serious injury. These warnings describe procedures to avoid death or serious injury.
- *General warnings*, preceded by a General Warning symbol, indicate situations or conditions which may cause death or serious injury. These warnings describe procedures to avoid death or serious injury.

---

**CAUTIONS** inform you of situations or conditions which will damage machinery or cause additional motor-operation down-time if you do not take suggested steps to correct or address such situations or conditions.

---

***Note:*** *Notes provide you with additional and useful information. Although less urgent than cautions and warnings, notes are important and should not be ignored.*

## Warning Symbols

For your own safety please pay special attention to instructions containing these symbols:

This warning symbol indicates the presence of dangerous voltage. This symbol informs you of high voltage conditions, situations, and locations that may cause death or serious injury if you do not follow precautions and proper steps.

This warning symbol indicates a general warning.

This warning symbol indicates an electrostatic discharge hazard.

## Warnings, Cautions, and Notes

**WARNING!** Your drive contains dangerous voltages when connected to the line power. Always check that the ACS 500 is safe, after disconnecting the power, by measuring the DC bus voltage and line input voltage. Failure to check voltages could cause death or serious injury. Only a qualified electrician should carry out the electrical installation.

Note that the Motor Control Card of the ACS 500 is at DC bus voltage potential.

The DC bus capacitors contain dangerous DC voltage levels (1.35 x $V_{IN}$). After disconnecting the supply, wait at least five minutes after the display readout on the control panel has disappeared before taking any measurements.

Dangerous external control voltages may be present on the relay outputs of the Control Interface Card and Option Cards.

**CAUTION**: Electrostatic Discharge (ESD) can damage electronic circuits. Do not handle any components without following the proper ESD precautions.

# Table of Contents

## Chapter 1 – Introduction



## Chapter 2 – Quick Installation



## Chapter 3 – Hardware Installation

## Chapter 4 – Software Installation



## Chapter 5 – Programming

## Chapter 6 – Trouble Shooting



## Appendix – A

**This page intentionally left blank.**

# Chapter 1 – Introduction

This chapter describes the purpose and contents of this manual, describes the intended audience, explains conventions used in this manual, and lists related publications.

## How To Use This Manual

The purpose of this manual is to provide you with the information necessary to install, start-up, and program the CDI-320 software interface from Allen-Bradley PLCs to ABBs ACS 500™ AC drives.

*Chapter 1 – Introduction* the chapter you are reading now, introduces you to the contents of this manual, and to the conventions being used through out the manual.

*Chapter 2 – Quick Installation* gives technical information about the operator panel.

*Chapter 3 – Hardware Installation* gives instructions of how to install the 1771-DMC coprocessor into the I/O rack, and how to do the wiring. The wiring diagrams include both the drive interface, and the programming terminal interface.

*Chapter 4 – Software Installation* gives an in-depth description of how to download the CDI-320 software into the coprocessor module. This chapter also teaches how to use the downloader software running on an IBM-PC or compatible.

*Chapter 5 – Programming* describes how to access the 1771-DMC module from the PLC ladder logic.

*Chapter 6 – Trouble Shooting* explains entry level diagnostics for finding out the causes and corrections to the most typical communication problems.

*Appendix – A* includes examples of how to use the coprocessor module from a ladder-logic program running on a PLC5 platform.

## Intended Audience

The audience for this manual has:

- Knowledge of standard electrical wiring practices, electronic components, and electrical schematic symbols.
- Minimal knowledge of ABB product names and terminology.
- Previous experience in installing, operating, and programming the ACS 500™ or ACH 500™ drives.

The audience for this manual will install, start-up, and diagnose the serial communication link to ACS/ACH 500 drives from a coprocessor module. The audience will also program the ACS 500 drives for the serial communication network.

The programming for the drives is described in the communication manual for the drives, and is not repeated within this document. The ACS/ACH 500 drives need to be updated for CDI 300 communication for use with the CDI 320 software module.

## Conventions Used In This Manual

Listed below are some terms used in this manual. These terms are defined here to help you understand their meanings and applications throughout this manual. A more complete list of conventions used for the ACS 500 drives is in the Installation manual for the ACS 500.

### Parameter

A parameter is a sub-set of a Group, selected through the Control Panel keys. Parameters in this manual often are expressed as a number, a decimal (.), another number, a decimal, and another number. The first number at the left represents the Main. The number between the decimals represents the Group, for example, 20.2 (Start/Stop). The number at the right represents a Parameter within that group, for example, 4 (Brake Chopper).

In this manual, Parameter 4 in Group 20.2 is expressed as Parameter 20.2.4.

### RS-485

RS-485 is a standard electrical communication interface. This standard specifies the physical interface and voltages for serial communication network. The RS-485 does not specify what kind of protocol is being used on the network. Modbus and CDI-300 are two examples of many possible protocols that can use the physical RS-485 link.

**Terminal Block** A terminal block is a group of wire connections on a drive. This manual expresses specific terminal blocks and connections as a letter, usually X, a number, a colon (:), and another number. The letter and number to the left of the colon represent the name of the terminal block, for example, X25. The number to the right of the colon represents the terminal connection, for example 16, on the terminal block. In this manual, a terminal connection numbered 16, located on a terminal block named X25, is expressed as X25:16.

**PLC** A PLC is a Programmable Logic Controller. PLCs are especially suitable to handle a large quantity of discreet I/O, and to modify it based upon software. PLCs can be programmed with a separate programming software.

**Coprocessor** 1771-DMC coprocessor module is a self-contained small computer. The coprocessor is capable of executing compiled code under a real-time operating system control.

**CDI 300 network** CDI 300 network is a serial and masterless communication network designed for standard drives. This ABB network can be used for connecting drives to various external control systems.

**Dataset** A piece of data being transferred on CDI 300 network. Each dataset has user defined contents.

**RAM** Read and Write Memory. This is a memory that can store data which can be retrieved at a later time.

**Non-volatile memory** Non-volatile memory keeps its contents even if the power is disconnected from it. Examples of non-volatile memory include EEPROM and Flash-ROM.

**Block transfer** A way of transferring a block of data between an AB PLC and its intelligent I/O devices.

**Local Rack** The I/O rack where the PLC resides. Communication between the PLC and the local rack have a minimum time delay.

**Remote Rack** Any I/O rack, which is connected to the PLC using a Remote I/O communication link. Communication between the PLC and the modules in any remote rack will take longer to execute, than to similar devices in the local rack.

## Warranty and Liability Information

The warranty for your ABB drive covers manufacturing defects. The manufacturer carries no responsibility for damage due to transport or unpacking.

In no event and under no circumstances shall the manufacturer be liable for damages and failures due to misuse, abuse, improper installation, or abnormal conditions of temperature, dust, or corrosives, or failures due to operation above rated capacities. Nor shall the manufacturer ever be liable for consequential and incidental damages.

The period of manufacturer's warranty is 12 months, and not more than 18 months, from the date of delivery.

Extended warranty may be available with certified start-up. Contact your local distributor for details.

Your local ABB Drives company or distributor may have a different warranty period, which is specified in their sales terms, conditions, and warranty terms.

If you have any questions concerning your ABB drive, contact your local distributor or ABB Drives office.

The technical data and specifications are valid at the time of printing. ABB reserves the right to subsequent alterations.

## Related Publications

For related information about the drive, refer to the ABB *ACS 500 Adjustable Frequency AC Drives 2 to 350 HP Programming Manual Including Application Macros (ACS 500-05)* and the *ACS 501 with Option Pack Users Manual (ACS 500-08)*. For the information about the CDI 300 network, refer to the ABB *CDI 300 Installation and Start-up Manual*.

# Chapter 2 – Quick Installation

This chapter has a short, quick installation guide for installing a new 1771 DMC module to an Allen-Bradley PLC I/O rack. This chapter includes both hardware installation, and the software configuration for the module. For detailed description, see *Chapter 3 – Hardware Installation* and *Chapter 4 – Software Installation*. For programming the PLC, please refer to *Chapter 5 – Programming*.

This chapter presumes that the audience has some prior knowledge of A-B PLC terminology, and has previous experience in using them.

## Hardware installation

The 1771-DMC module is an intelligent I/O module, which can be installed either into the local, or remote I/O rack. If there is a need for doing direct message communication with the drives, the module must be placed to the first slot right from a later type of PLC5s. Also the coprocessor and the PLC must be connected together through the coprocessor expansion port.

### Unit installation

If the unit has a battery on it, remove the battery for over 5 minutes to clear the memory of the unit completely.

Follow the steps listed below during the installation:

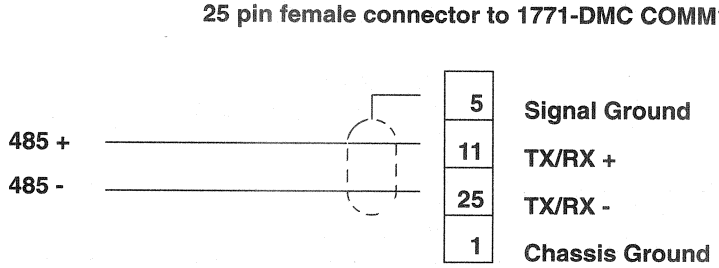
- Configure the COMM1 port for RS-485. This is done using the dip-switches on the module. The dip-switch settings are different in different models, so verify the settings from the user manual for your module.
- Place the battery into the DMC module
- Place the module into the I/O rack
- Wire the module to the drives

## Wiring

The 1771-DMC is connected to ABB ACS-500 drives using the CDI-300 network. This network is a daisy-chained multidrop network requiring a twisted shielded pair cable connection. Three wires plus the shield are required for a successful installation.

The following diagram shows how to connect the CDI-300 network to the 1771-DMC module.

*Figure 2-1*




If the 1771-DMC module is at the end of the network, a terminating resistor of 120 Ω, πW needs to be placed between the 485+ and 485- lines close to the 1771-DMC module. This resistor is terminating the network. Proper use of termination resistor will improve the noise tolerance of the network.

For further details of how to wire the drives, see the *CDI-300 Installation and Start-up guide*.

## Software Installation

The 1771-DMC module has a built-in RS-485 port. This port is able to communicate with the ACS-500 drives using the software package CDI 320 developed by ABB Drives. This package needs to be downloaded into the device from an IBM-PC or compatible computer.

The downloaded file is stored in a battery backed up RAM. The battery backed up RAM will keep its memory even if the power is disconnected from the module. If the battery is low or removed, and the power is disconnected, a memory loss can happen.

The PC needs to be connected using the serial cable described in *Figure 3-2, "Programming port connection, 9-pin," on page 3-5* or *Figure 3-3, "Programming port connection, 25-pin," on page 3-5*. The downloading is done using the following steps:

- Connect the serial cable into the `COM1:` serial port.
- Change the default directory to the one where all the files from the CDI-320 diskette have been installed. If using the `A:` drive type
  - `A:`
  - `CD \`
- Start up the downloader by typing `COPRO`
- Verify the connection by pressing the return key. The coprocessor should respond with a '`$`'sign
- Download the following three files to the module.
  - Press `[F1]`, type `STARTUP`, press `[ENTER]`
  - Press `[F1]`, type `CDI320`, press `[ENTER]`
  - Press `[F1]`, type `CDI320BX`, press `[ENTER]`
- Set the internal clock to the correct time by typing `SETIME [ENTER]`, and by typing in the correct time.
- Start the module by typing `STARTUP`, press `[ENTER]`.

The module will automatically restart itself each time the power is supplied to the module. There is no need to repeat the downloading steps, unless the battery is removed from the module.

If the time is not setup correctly, the CDI-320 application will not start when the coprocessor is powered up.

**This page intentionally left blank.**

# Chapter 3 – Hardware Installation

This chapter describes in detail the hardware installation for the 1771-DMC module when used with ACS-500 drives. This includes the hardware setups for the module, cable connections for the CDI-300 network, and cable connections for the downloader software package.

## Introduction

The control coprocessor is an independent processor expanding the capability of programmable logic controllers. The coprocessor can execute compiled C programs to perform tasks including communication to external devices via the asynchronous serial port or the optional Ethernet port.

The ABB ACS-500 drive connection is done using a user program running on the processor module, which allows the built in serial port to communicate directly with the drives.

### Module types

There are presently three different models of the coprocessor. These differ by the amount of the built in memory and on the optional Ethernet port. The different module types are listed on *Table 3-1 "Coprocessor models"*.

*Table 3-1 Coprocessor models*

| Description | Number | Memory | Ethernet port |
|---|---|---|---|
| Main module | 1771-DMC | 256 kBytes | No |
| Main module | 1771-DMC1 | 1 MBytes | Yes |
| Main module | 1771-DMC4 | 4 MBytes | Yes |

Models 1771-DMC1 and 1771-DMC4 can be used directly for communication with the ABB ACS 500 drives. The 1771-DMC can also be used, but requires a memory expansion to be installed.

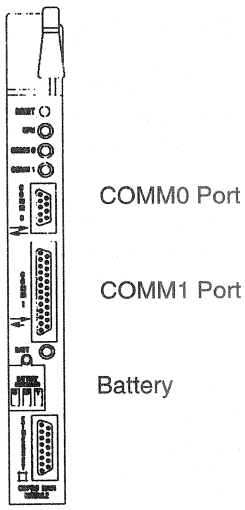
An optional 1 Mbyte SIMM Memory expansion needs to be installed into the 1771-DMC module.

## Hardware overview

*Figure 3-1 "Coprocessor hardware"* shows the major hardware elements for the coprocessor. The CDI-320 software module uses only the COMM1 port for communication.

The COMM0 port is used for downloading the software modules into the unit. The optional Ethernet port is not used by the CDI-320 module.

*Figure 3-1 Coprocessor hardware*




This module has the following elements on the front.

- COMM0 port. This is a 9-pin, optically isolated serial communication port. This port is used for connecting the IBM-PC or compatible computer for downloading the application software. This is an RS-232 port.
- COMM1 port. This is a 25-pin, optically isolated serial communication port. This is the main port where the ACS-500 drives are connected. There are no built-in termination resistors on the module.
- Battery. This battery provides the power to the module, which allows it to keep the application inside when the power is disconnected from the I/O Rack. If the battery is removed, or is low, when the power to the I/O rack is removed, the module will lose its memory, and the CDI-320 application will need to be reloaded.
- LEDs. These four leds provide status information from the module. For details see the User Manual provided with the coprocessor.

## Module Installation

The CDI-320 software module operates with all the A-B PLCs that use the 1771 I/O series, and are capable of communicating using Block Transfers. These module include PLC2s, PLC3, PLC5s, and PLC5/250s.

The co-processor module can be installed into three different locations in the I/O system. These include Direct-Connect mode with a direct memory access capability, and Stand-alone Mode in either the local or remote I/O rack.

### Direct-Connect Mode

The Direct Connect mode is available for only the latest series of A-B PLCs. In this mode, the co-processor module is located side by side with the PLC module. The coprocessor expansion port in the PLC needs to be connected to the expansion port on the co-processor.

This direct communication mode can be used to do message transactions with ABB Drives. These messages include the option of doing complete parameter uploads, downloads, and unlimited parameter reads and writes to the drives.

### Stand-alone Mode, Local I/O Rack

In the stand-alone mode, the co-processor module is located in the I/O Rack. The PLC can communicate with the processor using the Block Transfer Reads and Writes.

When the module is located in the local I/O rack, the communication between the PLC and the co-processor is not restricted by the Remote I/O (RIO) communication speed.

In stand-alone mode, the coprocessor module running the CDI-320 software can communicate with all the PLC models using the Block Transfer Reads and Writes.

### Stand-alone Mode, Remote I/O Rack

If the module is located in a remote I/O rack, the communication between the module and the PLC is done using the Block Transfer. Compared to the Local I/O Rack mode, the communication is slower, due to the delays caused by the Remote I/O communication link.

Otherwise the operation is identical to the Local I/O Rack operation.

### Switch Configuration

The 1771-DMC module has dip switches that must be set up before installing the 1771-DMC module into the I/O Chassis. Since there are several versions of the 1771-DMC available, please refer to the user manual for details about setting the switches.

Required settings include the setting of the COMM1 port for RS-485 communication mode. There are no switches needed for setting the COMM0 port for RS-232 communication mode.

### Battery

Install the battery to the unit as described in the User Manual.

### Wiring

To use the CDI-320 software module, two different serial connections to the module are needed. One serial cable is needed for the COMM0 port, and is used only for downloading the executable files into the module. The other connection is from the COMM1 port down to the drives.

### Configuration cable

For downloading the executable from the PC, a serial cable is needed. *Figure 3-2 "Programming port connection, 9-pin"* shows the connections with the 9-pin serial port, and *Figure 3-3 "Programming port connection, 25-pin"* shows the connections with the 25-pin serial port on the PC.

*Figure 3-2 Programming port connection, 9-pin*

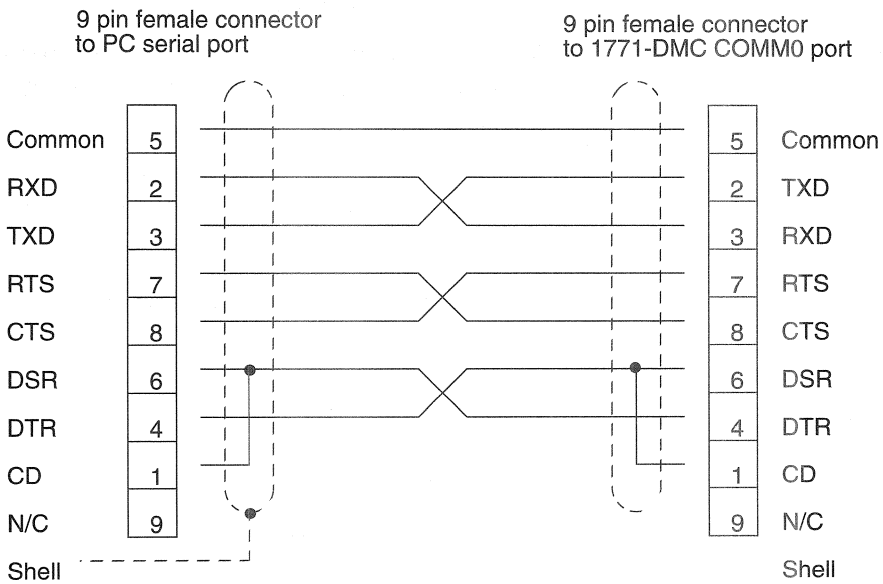
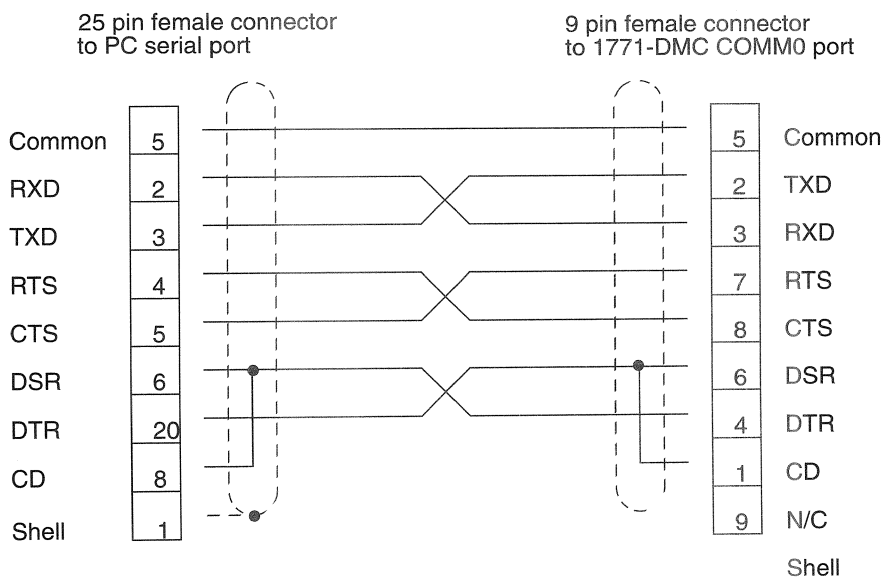



*Figure 3-3 Programming port connection, 25-pin*



## Drive connection

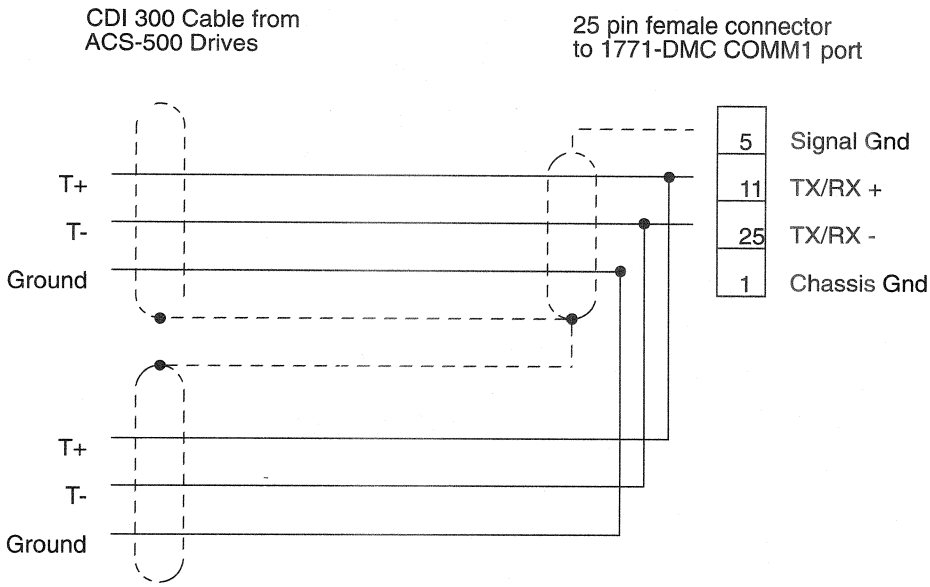
The drives are connected to the coprocessor with the Common Drives Interface network, CDI-300. This is a twisted shielded pair network, based upon the RS-485 industrial standard.

The wiring to the drives is described in detail in the CDI-300 manual for ACS-500 drives. Also the recommended cable is described there. ABB Drives is recommending the use of Belden 9842 dual twisted pair shielded cable or compatible. The cable must be twisted, have a shield, and the recommended wave impedance is 120 Ω.

The wiring to the coprocessor from the drives is shown in *Figure 3-4 "Connection to CDI-300 network"*. This does not show the terminating resistor.

If the coprocessor is the very last station on the daisy chained network, there needs to be a 120 Ω πW resistor placed in between pins 11 and 25.

*Figure 3-4 Connection to CDI-300 network*



# Chapter 4 – Software Installation

This chapter describes in detail the software module installation for the 1771-DMC module when used with ACS-500 drives.

## Introduction

The 1771-DMC module is a self-contained computer with a real-time operating system (OS-9), a RAM disk, and communication ports.

The co-processor module is able to execute multiple different programs. These programs need to be downloaded into the module using a serial interface from an IBM PC or compatible. Once the files have been transferred, they will remain on the 1771-DMC RAM disk until erased. The module can also lose its memory if the battery is dead or removed, and the power is removed from the I/O Rack.

Things needed for downloading the files:

- A 1771-DMC module mounted in a 1771 rack
- An IBM PC or compatible computer with a standard serial port
- DOS based configuration utility (`COPRO.EXE`)
- The serial RS-232 cable as described in *Figure 3-2 "Programming port connection, 9-pin"* or *Figure 3-3 "Programming port connection, 25-pin"*

## Software Installation

The following chapters describe how to load all the files into the co-processor module. In these chapters it is presumed, that the coprocessor is solely used for the CDI-300 network communication.

ABB Drives does not support the use of the co-processor for other purposes while running the CDI-300 communication software.

The following steps should be executed in series.

### Clear the Memory

This step is needed if there are other software packages already downloaded in the coprocessor module.

The simplest way to clear the memory is to remove the battery from the module, and to disconnect the power to the I/O Rack for 5 minutes.

### Connect the cable

To download the CDI-320 application to the module, the co-processor needs to be connected to an IBM-PC or compatible.

The cable to be used is shown in the *Figure 3-2 "Programming port connection, 9-pin"* for the 9-pin serial port, and in the *Figure 3-3 "Programming port connection, 25-pin"* for the 25-pin serial port.

The cable is connected to the COMM0 port in front of the co-processor module. Quick verification of the connection can be done using the COMM0-LED on the front. This led has the following colors:

- Solid Green, the cable is not connected
- Black, the cable is connected to the IBM-PC and the PC has power on
- Red, the co-processor is sending data on COMM0 port
- Green / Red, the co-processor is both receiving and sending data on COMM0 port

If the led goes from green to black when the serial cable is connected to the COMM0 port, the cable is properly connected.

### Starting the Installation

Copy all the files from the distribution diskette to one directory on your hard-drive. A way to do this is to type the following line:

- `XCOPY A:\*.* C:\CDI-320\*.*`

This will copy all the files to the directory of `C:\CDI-320`. This copying needs to be done only once. To start the program type:

- `C:`
- `CD \CDI-320`
- `COPRO`

### *COPRO Options*

The COPRO is both a terminal emulation software, and a downloader especially designed for the 1771-DMC coprocessor module. This software can be used with both standard PC serial ports, COM1: and COM2:, and with user selected baud and parity options.

As a default, the COPRO program uses the following options:

- COM1 port
- 9600 Baud
- 8 Data bits
- No parity
- 1 Stop bit

All of these can be changed by using command line options while starting up the program. Only the options which are different from the default need to be specified.

A complete selection of options is in *Table 4-1 "Command line options"*. Typically there is no need to select any of these options.

*Table 4-1 Command line options*

| Option | Default value | Legal values | Description |
|---|---|---|---|
| /P | /P1 | /P1 /P2 | COMM1: or COMM2: port selection |
| /B | /B9600 | /B1200 /B2400 /B4800 /B9600 | Baud rate |
| | /8 | /7 /8 | Number of data-bits |
| | /N | /N /O /E | Parity, None, Odd, Even |
| | /1 | /1 /2 | Number of stop bits |

Normally the serial port only needs to be selected.

A list of available command line switches can be displayed by using the? command.

- COPRO?

### *Verify communication*

To verify the communication, follow the steps listed below:

- Type [Return], the coprocessor should type '$' back to the screen. Also the COMM0 led should blink to green and red (very fast).
  - If no return, check
    - That the cable is properly connected
    - That the cable is wired correctly
    - That the baud rate, parity, and the serial port selections are correct
- Type "DIR" and [Return] using the software. The co-processor will respond by listing all the downloaded modules. This list should be empty.
  - If the list is not empty, clear the coprocessor memory as described in *Paragraph "Clear the Memory"*.

### Download the software modules

The CDI-320 application consists of three different files. These are:

- `STARTUP` - the system auto-start-up script file
- `CDI320` - the main drive application module
- `CDI320BX` - the block transfer manager module

All of these files are located on the release diskette along with the `COPRO.EXE` file. All of these must be downloaded into the 1771-DMC RAM disk.

To download the files follow these steps after verifying the communications:

- Press `[F1]`
- At the prompt, type `STARTUP` and `[Return]`. This will download the `STARTUP` file
- After the download is complete, press `[F1]`
- At the prompt, type `CDI320` and `[Return]`. This will download the `CDI320` file
- After the download is complete, press `[F1]`
- At the prompt, type `CDI320BX` and `[Return]`. This will download the `CDI320BX` file
- Type `SETIME` and `[Return]`. At the prompt enter a valid date and time

---

The `SETIME` command sets the internal date and time clock on the coprocessor. The date and time must be valid, otherwise the coprocessor will not execute user applications.

---

### Start the downloaded modules

To start the downloaded modules, type `STARTUP` from the OS-9 command prompt '`$`'.

This start-up is needed only once. After this, all the files are automatically started when the power is connected to the co-processor module.

After starting up the modules, the OS-9 prompt is no longer accessible. If there is a need to access the OS-9 prompt after the modules have been activated, see the *Paragraph "Terminate the Application Tasks"*.

### Exit the Downloader

Press [F10] (Quit) to exit to DOS.

### Terminate the Application Tasks

The application can be stopped by turning the Terminate bit high on the Module Control Word. This is described in more detail in *"Chapter 5 – Programming"*. This is needed only if there is a need to change the downloaded files in the co-processor.

# Chapter 5 – Programming

This chapter describes in detail the ladder logic interface programming for the CDI-320 software module. This chapter is intended for people who will write the ladder logic file interfacing with the CDI-320 software.

## Introduction

The CDI-320 software has been designed to access all features of the CDI-300 network. The module is capable of

- Sending drive control information down to the ACS-500 drives using the dataset services
- Receiving actual value feedback information back from the drives using the dataset services
- Sending and Receiving message transactions to and from the drives. This message communication is possible only if the coprocessor has a direct memory connection with the A-B PLC.

The dataset services are accessed using the Block Transfer commands. Due to the limitations of the block transfers, the data transfer is segmented into blocks.

## Overview

The block transfers are used for four different functions:

- Configure the coprocessor for CDI-300 communication
- Read the coprocessor status for CDI-300 communication
- Modify the outgoing dataset values to the drives
- Read the received dataset values from the drives

All of these operations are done using the Block Transfer Read and Write services.

Due to the limited size of Block Transfer, the data-transfer has been paged into blocks of data. This allows the PLC to access all the necessary information to and from the co-processor.

The Block Transfer 'space' has been divided into Blocks. These blocks are described in table *Table 5-1 "Block Transfer Overview"*.

*Table 5-1 Block Transfer Overview*

| Block Number | R / W | Description |
|---|---|---|
| 1 | R/W | Dataset data corresponding to configuration table entry #1 |
| 2 | R/W | Dataset data corresponding to configuration table entry #2 |
| ... | ... | ... |
| 128 | R/W | Dataset data corresponding to configuration table entry #128 |
| 129 | R | Status block |
| 130 | W | Module configuration block |
| 131 | W | Dataset configuration Block 1. 16 datasets |
| 132 | W | Dataset configuration Block 2. 16 datasets |
| 133 | W | Dataset configuration Block 3. 16 datasets |
| 134 | W | Dataset configuration Block 4. 16 datasets |
| 135 | W | Dataset configuration Block 5. 16 datasets |
| 136 | W | Dataset configuration Block 6. 16 datasets |
| 137 | W | Dataset configuration Block 7. 16 datasets |
| 138 | W | Dataset configuration Block 8. 16 datasets |

Blocks marked R/W are both readable and writeable. Blocks marked as R are only readable, while the blocks marked as W are only writeable.

The blocks can be written freely in random. The blocks are identified by the first word in the block transfer write data-area.

The readable blocks are read in sequence. On each one of the Block Transfer Read block executions, the co-processor returns one of the readable blocks. Also in this case, the first word will be used to identify the block number, whose data is returned after the block number word.

Every Block Transfer Read and Block Transfer Write must have the length as 64. The Block Transfers will time-out and generate an error with any other length.

## Block structure

Each one of the blocks have their own internal structure. All of these structures are described below.

### Blocks 1 – 128

These blocks contain the dataset data. If the dataset is shorter than 16 words, the data is located at the beginning of the block, with the rest of the block unused.

Only the blocks which are transmitted need to be modified using the Block Transfer Writes.

### Block 129 - Status Block

The status block is a read-only block, containing general network status information, and also some diagnostic counters. Most of these diagnostic counters are intended for ABB Drives internal use only, and are not fully documented.

*Table 5-2 Status Block*

| Word | Data Type | Description |
|---|---|---|
| 1 | Word | 129<br>BTR Block number |
| 2 | Packed Boolean | Module status word,<br>*Table 5-3 "Module Status Word"* |
| 3 | Packed Boolean | Active station bitmap, stations 0–15 |
| 4 | Packed Boolean | Active station bitmap, stations 16–31 |
| 5 – 8 | String | The 'new' device name for this node. |
| 9 – 18 | Word | Reserved diagnostics counters |
| 19 | Word | Characters transmitted |
| 20 | Word | Characters received |
| 21 | Word | Bad characters received |
| 22 – 25 | Word | Reserved diagnostics counters |

Word one always identifies this block as the Module Status Block with the value of 129. The Module Status Word (word 2 in the status block) is a bit packed boolean word described in *Table 5-3 "Module Status Word"*.

The active station bitmaps in words 3 and 4 have an individual bit high, if the corresponding station is active on the CDI-300 link.

The 'new' device name is valid when the Node Name Changed flag is set in the Module Status Word. This string shows the new name for the coprocessor on the CDI-300 link, if it is being changed by an external tool. This string could be read from the status block, and stored into the PLC configuration file, so that the next time the coprocessor is initialized, the new name is remembered.

Diagnostic counters in words 19, 20, and 21 show the cumulative count of the transmitted, received, and bad (parity error) characters on the CDI-300 link for the coprocessor module.

*Table 5-3 Module Status Word*

| Bit | Meaning | Description |
|---|---|---|
| 0 | BA Active | The BA is active on the network |
| 1 | Node Active | The coprocessor is being polled by the BA |
| 2 | Datasets configured | The dataset configuration has been completed |
| 3 | Node Name Changed | The coprocessor has been renamed from the CDI-300 link |
| 4 | Config Error | The module configuration block is invalid |
| 5 | Config Error | One or more dataset configuration blocks is invalid |

### Block 130 - Module Configuration Block

This configuration block contains the general CDI-300 operating parameters and the module control word.

*Table 5-4 Module Configuration Block*

| Word | Data Type | Description |
|---|---|---|
| 1 | Word | 130<br>BTW Block Number |
| 2 | Word | Module Control Word<br>*Table 5-5 "Module Control Word"* |
| 3 | Word | 1 – 31<br>CDI-300 network station number |
| 4 – 7 | String | The user defined device name for the coprocessor on the CDI-300 network |

*Table 5-5 Module Control Word*

| Bit | Meaning | Description |
|---|---|---|
| 0 | Enable Link | The rising edge of the bit configures the CDI-300 link, and enables the communication. The falling edge of this bit shuts down CDI-300 communication. |
| 1 | Configure Datasets | The rising edge of this bit causes the dataset configuration to be read from the dataset configuration blocks |
| 2 | Faulted | This faulted status bit is copied to the CDI-300 link status bit 7 |
| 3 | Node Name Accepted | The rising edge of this bit will clear the Node Name Changed bit in the Module Status Word. The PLC program should monitor the Node Name Changed bit, and copy the new node name from the Module Status Block into the Module Config Block as required. After reading the new name the Node Name Accepted bit should be set until the Node Name Changed flag clears. |
| 15 | Terminate | Setting this bit will cause the CDI-320 application tasks to terminate. This provision is to permit access to the OS-9 shell to download new or updated application modules. **This bit must be clear for normal operation** |

### *Blocks 131 – 138 Dataset configuration Blocks*

These configuration blocks contain the dataset configuration data. Each block has 16 entries for a total of 128 possible datasets. To minimize the amount of memory required, the dataset number is specified in the table rather than being implied by the table entry number. This permits a wide range of dataset numbers to be configured using a small table.

All dataset configuration table entries are initialized to 0 at start-up to eliminate the need to write config blocks for unused table entries.

Each Dataset Configuration Block has three consecutive words for each one of the datasets. These words have the following meanings:

- Dataset number [1 – 128]. 0 disables this entry
- Output dataset size in bytes [1 – 32]. 0 disables this entry. For the incoming datasets, the length must be between of 1 and 32. The actual value has no difference. The value of 32 is a good choice for incoming dataset length.
- Dataset transmission interval in milliseconds. [100, 200, …, 25500]. 0 identifies this as a read set.

The entries correspond to the actual dataset data blocks in blocks 1 – 128. The first entry in the dataset configuration table defines the actual dataset number for the dataset data block number 1, the second for the dataset data block number 2 etc.

*Table 5-6 Dataset Configuration Blocks*

| Word | Data Type | Description | | |
|---|---|---|---|---|
| 1 | Word | 131 – 138<br>BTW Dataset Configuration block number | | |
| 2 – 4 | Word | DS Number | DS Size | DS Interval |
| 5 – 7 | Word | DS Number | DS Size | DS Interval |
| 8 – 10 | Word | DS Number | DS Size | DS Interval |
| 11 – 13 | Word | DS Number | DS Size | DS Interval |
| 14 – 16 | Word | DS Number | DS Size | DS Interval |
| 17 – 19 | Word | DS Number | DS Size | DS Interval |
| 20 – 22 | Word | DS Number | DS Size | DS Interval |
| 23 – 25 | Word | DS Number | DS Size | DS Interval |
| 26 – 28 | Word | DS Number | DS Size | DS Interval |
| 29 – 31 | Word | DS Number | DS Size | DS Interval |
| 32 – 34 | Word | DS Number | DS Size | DS Interval |
| 35 – 37 | Word | DS Number | DS Size | DS Interval |
| 38 – 40 | Word | DS Number | DS Size | DS Interval |
| 41 – 43 | Word | DS Number | DS Size | DS Interval |
| 44 – 46 | Word | DS Number | DS Size | DS Interval |
| 47 – 49 | Word | DS Number | DS Size | DS Interval |

Once all the needed datasets are configured, all the rest of the entries should be left to the value of zero.

The dataset size is defined in bytes. One word is two bytes. For the use of ACS 500 drives, multiply the number of parameters transmitted by two. This is the length of the dataset in bytes.

### Blocks 1 – 128 structure

The blocks 1 – 128 which are used for both the incoming and outgoing dataset data have their own internal structure. This structure differs between the outgoing datasets and the incoming datasets.

#### Outgoing datasets

The outgoing dataset data contains $n + 1$ words. The $n$ here stands for the number of parameters to be sent out using the dataset. The first word is the block identification number, followed by the words containing the actual parameter values.

*Table 5-7 Outgoing dataset data block structure*

| Word | Value | Description |
|---|---|---|
| 1 | 1 – 128 | BTW Block number |
| 2 – 9 | | Parameter value for the dataset |

#### Incoming datasets

The incoming dataset data block contains $n + 2$ words. The $n$ here stands for the number of parameters being sent by ACS 500 drive using the dataset service. The first word is the block identification number, followed by a word specifying the length of the received dataset. After this is the received parameter data.

The length field can be used for communication loss detection. When the dataset is not received, the length field is set to zero. The datavalues remain in their last received value.

*Table 5-8 Incoming dataset data block structure*

| Word | Value | Description |
|---|---|---|
| 1 | 1 – 128 | BTR Block number |
| 2 | 0 – 8 | Received dataset length, zero if not received |
| 3 – 10 | | Parameter value read |

### Device Configuration

The CDI-320 module requires configuration information for the CDI-300 network. This configuration must be written into the module every time the power is turned on.

The configuration is done in the following steps, which will be explained in more detail later:

- Shut down the CDI-300 communication
- Define the CDI-300 node-number and the device name
- Define all the outgoing datasets into the Dataset configuration blocks
- Define all the incoming datasets into the Dataset configuration blocks
- Restart the CDI-300 communication
- Redefine the dataset configuration

### Stop the CDI-300 communication

This is done by writing a value of zero to the Module Control Word, clearing the bit 0, Enable CDI-300. To do this, perform a block transfer write to the coprocessor module, using 7 words with the following contents:

*Table 5-9 Module configuration example*

| Word | Value | Description |
|---|---|---|
| 1 | 130 | Identify the write to Module Configuration Block |
| 2 | 0 | Module Control Word |
| 3 | (1 – 31) | The station number on CDI-300 link |
| 4 – 7 | String | The coprocessor name on the CDI-300 link |

In this example, the first two words are fixed, while words 3 through 7 depend on the application.

This write also defines the CDI-300 node-number and the device name on the link.

### Define datasets

Both the incoming and outgoing datasets are defined to the coprocessor module by writing the correct data into the Dataset configuration blocks 131–138. An example where the gateway is sending out dataset 10, with 4 words, with a 500ms transmission interval, and is receiving datasets 12, and 13 is in *Table 5-10 "Dataset Configuration example"*.

*Table 5-10 Dataset Configuration example*

| Word | Value | Description |
|---|---|---|
| 1 | 131 | Identify the write to Dataset Configuration Block |
| 2 | 10 | Write dataset number |
| 3 | 8 | 8 bytes for the length, 4 words |
| 4 | 500 | Transmission interval, 500ms |
| 5 | 12 | Read dataset number |
| 6 | 32 | Read dataset maximum length |
| 7 | 0 | Identify this as an incoming dataset |
| 8 | 13 | Read dataset number |
| 9 | 32 | Read dataset maximum length |
| 10 | 0 | Identify this as an incoming dataset |

The rest of the Dataset configuration blocks should be zeroes.

With this configuration, writes to block 1 will send out the values for dataset 10, and block transfer reads from blocks 2 and 3 will return data for the incoming datasets 12 and 13.

### Restart CDI-300 network

After the dataset and module configuration has been completed, the CDI-300 network application needs to be restarted on the coprocessor module. This is done by doing two block transfer writes to the coprocessor module.

The first write will configure the CDI-300 network, while the second write will cause the dataset configuration to be read.

*Table 5-11 Restart CDI-300*

| Word | Value | Description |
|---|---|---|
| 1 | 130 | Identify the write to Module Configuration Block |
| 2 | 1 | Module Control Word |
| 3 | (1 – 31) | The station number on CDI-300 link |
| 4 – 7 | String | The coprocessor name on the CDI-300 link |

*Table 5-12 Set Dataset configuration*

| Word | Value | Description |
|---|---|---|
| 1 | 130 | Identify the write to Module Configuration Block |
| 2 | 3 | Module Control Word |
| 3 | (1 – 31) | The station number on CDI-300 link |
| 4 – 7 | String | The coprocessor name on the CDI-300 link |

The station number and the coprocessor name should be the same as in *Table 5-9 "Module configuration example"*.

### Dataset Transmission

The CDI-320 application software is sending out to the CDI-300 network all the dataset values which it has been configured to be sending out. The time interval for this transmission is specified by the user in the Dataset configuration block.

The values being sent can be modified by doing a Block Transfer write to the block, which contains the outgoing dataset.

The block transfer writes can be done freely in a random order. The first data word in a block transfer write will always specify the Dataset block number where the write takes place.

To modify the data in dataset 10 on the example configuration shown above, do the following Block Transfer Write to the coprocessor module.

*Table 5-13 Modify outgoing dataset example*

| Word | Value | Description |
|---|---|---|
| 1 | 1 | Identify the write to Dataset data block |
| 2 | User specified | First word of the dataset |
| 3 | User specified | Second word of the dataset |

Word 1 on the Block Transfer Write identifies that this write will write to the first dataset data block.

## Dataset Reception

The coprocessor module will return in sequence the data for each one of the readable Blocks, one per each one of the Block Transfer Reads being executed by the host PLC.

The block is identified by the first word in each of these blocks. This block id number corresponds to the block numbers shown in *Table 5-1 "Block Transfer Overview"*. It is not possible to read in random the readable blocks, as it is possible to write the blocks in random.

In addition to the incoming, configured dataset blocks, also the Status Block is read as a part of this cyclical block transfer read operation.

## Module Status

The Module Status is the block number 129. Its contents are described in *Table 5-2 "Status Block"*. The Module Status Block is returned as one of the readable, incoming blocks, which are returned in sequence for the Block Transfer Read commands executed from the host PLC.

## Message Services

The CDI-320 software module can send out unsolicited messages on the CDI-300 network.

Message Service access is not available on the first release of the CDI-320 software. The availability of these features will be introduced at a later time. The following information is provided for information only.

These messages can be used to:

- Read one or more parameters within one group
- Write any parameter value in the drive
- Upload all the parameters in binary format from the drive
- Download all the parameters in binary format back to the drive
- To write the present setups into the EEPROM

These options are available only if the coprocessor is connected to a PLC5 coprocessor expansion port.

The message feature is a powerful way of accessing data directly on the drive. If the message communication is to be used, great care must be taken to verify that the message communication is programmed properly.

The message communication is designed to operate in the background. The message communication does not interfere with the ongoing process data transfer using the datasets.

### Overview

The message communication is done in four steps:

- Pack the necessary information for an outgoing message into the PLC integer table
- Tell the coprocessor to send a message out to a drive
- Wait for the reply message to return
- Use the received data

A PLC3 write message is use to activate the sending of an unsolicited message on the CDI-300 network. This write message is sent directly to the coprocessor using the coprocessor expansion port.

The outgoing data is contained in an integer file, and the reply data back from the drives is also in an integer file. The file number, length, and the starting element to be used is told to the coprocessor in the PLC3 write message.

The completion status of the message can be read from the coprocessor using a PLC3 read message. The reply data is automatically placed into the indicated PLC numerical file.

### MSG Write Data Block

The MSG write ladder logic instruction sends a block of data to the 1771-DMC. The MSG instruction control block specifies where the data block is located in the PLC memory (integer file and element number).

The MSG block needs to send out a `PLC3 Word Range Write` command. The Destination Data Table Address must be within "00" to "31", the quotation marks are required. The Port Number is `3A`.

The data to be sent to the coprocessor using a MSG block should contain the following information:

*Table 5-14 MSG Write Data Block*

| Word | Data Type | Description |
|---|---|---|
| 1 | Word | Command buffer file number. (Integer file) |
| 2 | Word | Command buffer offset. (Word number within the file) |
| 3 | Word | Reply buffer file number. (Integer file) |
| 4 | Word | Reply buffer offset. (Word number within the file) |
| 5 | Word | Reply buffer size in words. The reply buffer must be large enough to accept the whole reply message |

The MSG write data block basically tells the coprocessor where to find the outgoing command data (File number in word 1, and the starting word number in word 2), and where to place the returning reply data (File number in word 3, the starting element number in word 4, and the maximum length of the buffer in word 5).

The coprocessor will access the information from the specified file, and will modify the reply buffer with the data from the drives.

### Command Buffer

The command buffer is a part of an integer file. The file number and the first word used is defined in the MSG Write Data Block, words one and two. See *Table 5-14 "MSG Write Data Block"*.

The command buffer can be placed in the same integer file as other data.

The command buffer has an internal format described in *Table 5-15 "Command Buffer Format"*. The command data is described in paragraph *Message Formats on page 15*.

*Table 5-15 Command Buffer Format*

| Word | Data Type | Description |
|---|---|---|
| 1 | Word | CDI 300 Command number [2 – 63] |
| 2 | Word | Destination station number [0 – 31] |
| 3 | Word | Command length in bytes |
| 4 ... N | Word | Command data |

The coprocessor will take the information from the command buffer, send it out on CDI 300 network, and return the reply data back to the reply buffer.

### Reply Buffer

The reply buffer is a block of words in an integer file where the reply data from the CDI300 command is placed. The reply buffer is a part of an integer file. The file number and the first word used is defined in the MSG Write data block, words three and four. Word five is used to limit the maximum length that the coprocessor is allowed to use. See *Table 5-14 "MSG Write Data Block"*.

The reply buffer can be placed in the same integer file as other data.

The reply buffer has an internal format described in *Table 5-16 "Reply Buffer Format"*. The reply data is described in paragraph *Message Formats on page 15*.

*Table 5-16 Reply Buffer Format*

| Word | Data Type | Description |
|---|---|---|
| 1 | Word | Reply length in bytes |
| 2 | Word | Reply data |

If the reply data received from the CDI300 network is larger than the defined reply buffer length, an error will be indicated in the MSG Read Data Block, and the reply will be discarded.

### MSG Read Data Block

The MSG Read instruction reads a block of data from the 1771-DMC. The data block is stored in a file and referenced by the MSG instruction control block.

The MSG block needs to send out a `PLC3 Word Range Read` command. The Destination Data Table Address must be within "00" to "31", the quotation marks are required. This number must match with the command MSG number. The Port Number is `3A`.

The MSG Read Data Block is used to 'poll' from the coprocessor the status of a pending message transmission. The MSG Write Data Block will initiate a message communication between the coprocessor and the drives. Depending on the command number and the network load there will be a time delay before a reply data is received.

The MSG Read Data Block can also be used to determine the completion status for the message communication.

*Table 5-17 MSG Read Data Block*

| Word | Data Type | Description |
|---|---|---|
| 1 | Word | Message channel status word |

There is only one word of data returned with the MSG Read Data Block. This word has the following internal format.

*Table 5-18 Message channel status word*

| Bit | Position | Description |
|---|---|---|
| CDI 300 Status | 0 – 7 | CDI300 command completion status code. 0 is success, <>0 is a communication error. |
| CDI 300 Error | 8 | The message or reply was not transferred successfully. There has been an error on CDI 300 link. See CDI 300 status for completion status. |
| Size Error | 9 | The reply buffer is too small to contain the reply data. Reply Error will also be set. |
| PLC Data Table Error | 10 | An error occurred accessing the PLC data table. Either Command Error or Reply Error will be set to indicate the data table access which failed. |
| Memory Error | 11 | An error occurred allocating memory for the command. Command Error will also be set. |
| Reply Error | 12 | An error occurred receiving the command or writing the reply data into the reply buffer. Other status bits may provide further details for the error. |
| Command Error | 13 | An error occurred reading the command data from the command buffer or sending the command. Other status bits may provide further details for the error. |

| Bit | Position | Description |
|---|---|---|
| Channel Busy | 14 | Set when a message is in progress, cleared on completion or error. |
| | 15 | Unused, always 0 |

A PLC Data Table Error may cause the PLC to fault. The coprocessor module will continue to operate on-line.

## Message Formats

Messages on the CDI 300 link are communication transactions between two stations. The message services are intended for programming of devices or for doing software maintenance, like uploading of parameter settings.

The message communication is designed to work without interfering the dataset process data transfer. This allows the use of messages even in a working system.

The messages are always done in command-response cycle. In this transaction, the following things happen:

- A sends a command message to B. The command is identified with a command number. Command data is passed from A to B.
- B receives the command, and processes it.
- B send a reply message back to A. The reply may or may not contain data with it.
- A receives the reply, and if necessary, uses the received reply data.

An example of message transaction is a read command. Here the command data contains the parameter addressees to be read, while the reply data contains the actual values of the parameters being read.

On ACS and ACH 500 drives, there are some messages which have been defined. These are described in detail in the following paragraphs.

### Identify 0x02

Identify message can be used to identify the stations on the CDI 300 network. This command will return from each one of the stations the following information in text format:

- Device type identification
- Device firmware version identification
- User defined device name

Command number: 0x02

Command data: NONE

Reply data: 29 bytes of data with

10 bytes for device identification

11 bytes for firmware version

8 bytes for user defined device name

### Set Node Name 0x03

This command can be used for changing the user defined device name. This name is completely for customer use.

If a name has been changed there is also a need to issue a Backup command. This command will store the new name into non-volatile memory. Without the Backup command, the new name will be lost after a power up.

Command number: 0x03

Command data: 8 bytes of data defining the new name

Reply data: NONE

### Read Parameter 0x15

The read parameter command can be used for reading one or multiple parameters from an ACS/ACH 500 drive. The read is limited to parameters residing within one group.

The parameter address format is described in the CDI300 manual for the drives. The address is in the column of Address in *Appendix A – Parameter List*. The leading zeroes are compulsory.

Command number: 0x15

Command data: First and last parameter address to be read, terminated by 0x00

Reply data: n words of data. Each word corresponds to the actual parameter value being read.

Example: To read the operational data values from Output Frequency to DC bus voltage, the command data is:

0.00.001°0.00.006°

Here the ° stands for a byte with a value of 0 (NULL).

The reply data contains 6 words of data, one word per each parameter being read.

### Write Parameter 0x16

The write parameter command can be used for modifying parameters in the drive. The write is limited to one parameter value at a time.

The parameter address format is described in the CDI300 manual for the drives. The address is in the column of Address in the Appendix A – Parameter List. The leading zeroes are compulsory.

Command number: 0x16

Command data: The parameter address to be read twice, followed by the new value in word.

Reply data: NONE

Example: To change the EXT REF1 MINIMUM to 10 Hz, the command data is:

1.02.003°1.02.003°LM

Here the ° stands for a byte with a value of 0 (NULL).

L is the least significant byte of 1000 = 238

M is the most significant byte of 1000 = 3

The written values are not written into non-volatile memory, unless explicitly told to do so. The storage to non-volatile memory is activated by the Backup command.

### Upload 0x10

Upload can be used for reading all parameter values from the drive up to the coprocessor module. This reading will take a complete copy of the parameter listing, and store it in the PLC for a later restoring to the same or other drive which has the same firmware revision.

This uploading is done using one command, and the reply contains the whole application set-point listing.

Command number: 0x10

Command data: NONE

Reply data: n bytes of data containing in binary format the parameter settings for the drive.

The uploaded parameter listing is in binary format, and is not intended to be modified in any way. The data can be used for a later download command to restore the previously uploaded set-point values.

### Download 0x11

Download can be used for restoring to the drive previously uploaded parameter settings. This command will download a binary block of data to the drive. The binary block of data contains all the parameter values within it.

To keep the new set-points valid after the drive is being powered down, a separate Backup command needs to be sent down to the drives. This will write the changed values into non-volatile memory.

The length must be exactly the same as returned by the Upload command. The only data that can be downloaded is the data received using an earlier Upload command.

Command number: 0x11

Command data: n bytes of data containing in binary format the parameter settings for the drive. This data must have been previously read with the Upload command.

Reply data: NONE

### Backup 0x12

The Backup command is used for storing any changes made to the parameter settings in the drive to non-volatile memory. This command needs to be issued after:

- Set Node Name
- Write Parameter and
- Download

commands. If the Backup command is not issued, all the changes made to the parameter settings in the drive will be lost after the power is disconnected from the drive.

# Chapter 6 – Trouble Shooting

This chapter describes the first entry level of trouble shooting for the CDI-320 software package. This chapter is intended for people who sillbe programming the PLC communicating with the 1771-DMC coprocessor module.

## Fault Diagnostics

This chapter concentrates on problems and possible remedies for the serial communication with the CDI-320 software module. For other general fault diagnostics with the ACS 500 or ACH 500 drives, consult the appropriate product manual.

The communication problems can be caused by multiple sources. Some of these include:

- Loose connections or incorrect wiring
- Bad grounding
- Duplicate station or dataset numbers
- Incorrect setup of the drives
- Incorrect programming of the coprocessor

This chapter will list some possible communication problems, how to identify them, and will list some possible corrections.

## Downloading the CDI 320

The downloading of the three modules that complete the CDI 320 software requires that the module is wired correctly to the PC, and that there is no other application running actively in the module. Also the internal clock has to be set to an accurate date and time.

### *Wrong wiring*

If the wire from the coprocessor is wired correctly, the COMM0 light will be green when the PC has power, and it should be blinking to red when the module is sending data back to the PC.

If the wire is not correct, the COMM0 LEDwill be black. If the LED is green, the wire is most likely correct.

### *Downloading failed*

If unable to download the application to the module, it might be caused by another downloaded application. To verify that there is nothing else in the co-processor memory, the best way is to clear its memory.

To do this, disconnect the battery on the coprocessor. After this either pull the coprocessor out from the rack, or disconnect the power to the I/O rack for over one minute.

Reconnect the power and follow the downloading instructions earlier in this manual.

### Coprocessor does not start

This problem is indicated when the coprocessor is powered up, there is a start-up sequence on the connected configuration terminal, and any kind of error message occurs.

Also there is no green light on the COMM1 port, even when the coprocessor is connected to an active network.

In this case try the following:

- type `DIR` on the terminal. You should see only the following three files:
  - STARTUP
  - CDI320
  - CDI320BX
- If there are more files, clear the memory, and reload all the modules.
- Set the time to the coprocessor using the `SETIME` command and answer with a valid time. The coprocessor does not start-up if the time is incorrect.

### Block Transfer Fails

If the coprocessor starts up without problems the Block transfer reads and writes should be working from the PLC. If there are problems, check the following items:

*Wrong slot*

If the slot addressing is wrong, the BTR and BTW will fail immediately. Check that the Rack number, Group number, and the Module number are correct. Also verify that the slot addressing is correct (∫, 1, or 2-slot).

*Incorrect length*

If the block transfer fails with a time-delay of about 5 seconds, check that the BTR and BTW lengths are exactly 64 words. **Only** 64 word BTR and BTW is supported.

*Incorrect order*

The coprocessor code support BTR and BTW only when they are executed in sequence. If there are two BTRs in row, or two BTWs in row, the second one will fail, and will give a 5 second time out delay before failing.

## No Communication

If the coprocessor code works without problems, the problem might be in configuration or wiring.

### BA off-line

On the CDI-300 network, one of the drives must be assigned a station number of zero. If this is not the case, or if drive number zero is off-line, the whole communication is stopped.

If there is no BA on line, the `COMM1` port is black, and the `GOOD MSG COUNTER` and `BAD MSG COUNTER` do not advance. Turn the BA on line, or assign one of the drives a station number of zero.

### Configuration

If the communication does not work but there seems to be information being transmitted on the wire, the configuration might be incorrect on the coprocessor or on the drive. There is some communication on the network if the COMM1 light blinks on the coprocessor module.

Ensure that the coprocessor is on line. This can be seen from the station on line bits in the Module Status Block.

Check that the drive configuration is correct on the outgoing datasets and on the incoming datasets.

Check that the coprocessor is configured correctly to send and receive the intended datasets.

Check that the coprocessor ladder file starts up the coprocessor properly, with the Module Control Word having sequential values of 0, 1, and 3. See the ladder logic example in the *Appendix – A*.

## Other problems

Other problems might be caused by incorrect wiring, or grounding on the system. Please refer to the fault diagnostics in the "Installation and Start-Up Manual" for the CDI-300 network.

## Summary

The problems described here cover the most usual problems on starting up the CDI 320 module.

If after basic troubleshooting, the problem continues, contact ABB technical support (800) 243-4384.

**This page intentionally left blank.**

# Appendix – A

This appendix shows an example ladder logic file for interfacing with the coprocessor module from a PLC5.

## Introduction

The following example is designed to show an example interface from a PLC5 processor to the coprocessor module. This example is designed to be used with the ACS 500 series AC drives.

With the ACS 500 drives, the maximum size of the 'dataset' is 8 words. This is used in the following example.

The Ladder file does the following things in sequence after a power-up, or after the processor is placed into the Run-state:

- Deactivate the coprocessor on the CDI-300 network
- Activate the coprocessor on the CDI-300 network
- Send to the coprocessor both the incoming and outgoing dataset configuration
- Use the downloaded configuration

After these steps, the ladder file will continuously write the outgoing dataset data values to the coprocessor, and at the same time, read the coprocessor status and the received dataset data values back.

The ladder file has one BTW block and one BTR block. These are executed in sequence, BTW - BTR - BTW - BTR .... The initialization timer in the beginning has two functions. It provides a short start up time delay for the coprocessor to initialize, and it also ensures that on the power up the coprocessor is configured correctly in all the proper steps.

This ladder file uses multiple data files for its operation. In this example there are multiple files used to improve the readability of the example, and also to make it easily expandable. The BTW and BTR instructions can not be used with an indirect addressing: therefore, there are buffer tables to contain both the outgoing and incoming data using the block transfer commands.

The files in use for the outgoing data and for the configuration are:

- N20

  This file is the BTW data buffer area. All the outgoing blocks to the coprocessor are first transferred to this buffer, before sending the data out.

- N21

  This file contains the Module Configuration Block in its first 7 elements (N21:0 – N21:6). See *Table 5-4, "Module Configuration Block"* for details. This block is used to control the coprocessor status.

- N22

  This file contains the dataset configuration data as shown in the *Table 5-6, "Dataset Configuration Blocks"*. The ladder file is designed to send to the coprocessor 1 – 32 different dataset configuration records. These can be sent as one (1 – 16) or two (1 – 32) Dataset configuration blocks. The first block starts at N22:0, and the second block at N22:50.

- N23

  This file contains the outgoing dataset information. The ladder file is designed to send down to the drives datasets with a maximum length of 8 words. The dataset numbers are defined in the dataset configuration block data in the file N22.

  The dataset data is in sequence. The first dataset data is in N23:10 – N23:19. The word N23:10 has the dataset block id 1, 2, .... The words N23:11 to N23:8 contain the actual data. N23:9 is unused. The second dataset data starts from N23:20. The third dataset from N23:30.

The files in use for the incoming data are:

- N30

  This file is the BTR data buffer, where the returned data is placed from the coprocessor. The first word, N30:0 is always the read block index, which is used to copy the received data in to the correct data-area.

- N31

  This file contains the Status Block from the coprocessor. See *Table 5-2, "Status Block"* for details. This information can be used for diagnostics, and for finding out which stations are presently on line on the CDI-300 network.

- N32

  This file contains the received datasets in sequence. The first received dataset is stored to N32:0 – N32:9. The N32:0 is the dataset index (1 – 64) indicating which dataset is received. This index points to the entry number in the Dataset Configuration Block table, where the actual dataset number is defined.

  Word N32:1 is the received length of the dataset. 0 indicates that the sending station is either off-line, or the dataset is not configured.

  The words N32:2 – N32:9 contain the actual received data.

  The second dataset is stored into N32:10 – N32:19. The third one is after this.

The CMP and FAL statements before the BTW statement are used for transferring the outgoing information from the datafiles into the BTW buffer area. This code uses the FAL instruction instead of the COP instruction, for the FAL is executed only once on the rising edge of the input pin, while the COP instruction is excuted on every program scan, when the incoming condition is true. If the BTW takes longer than one program scan, the COP instruction will waste the processor time compared to the FAL instruction.

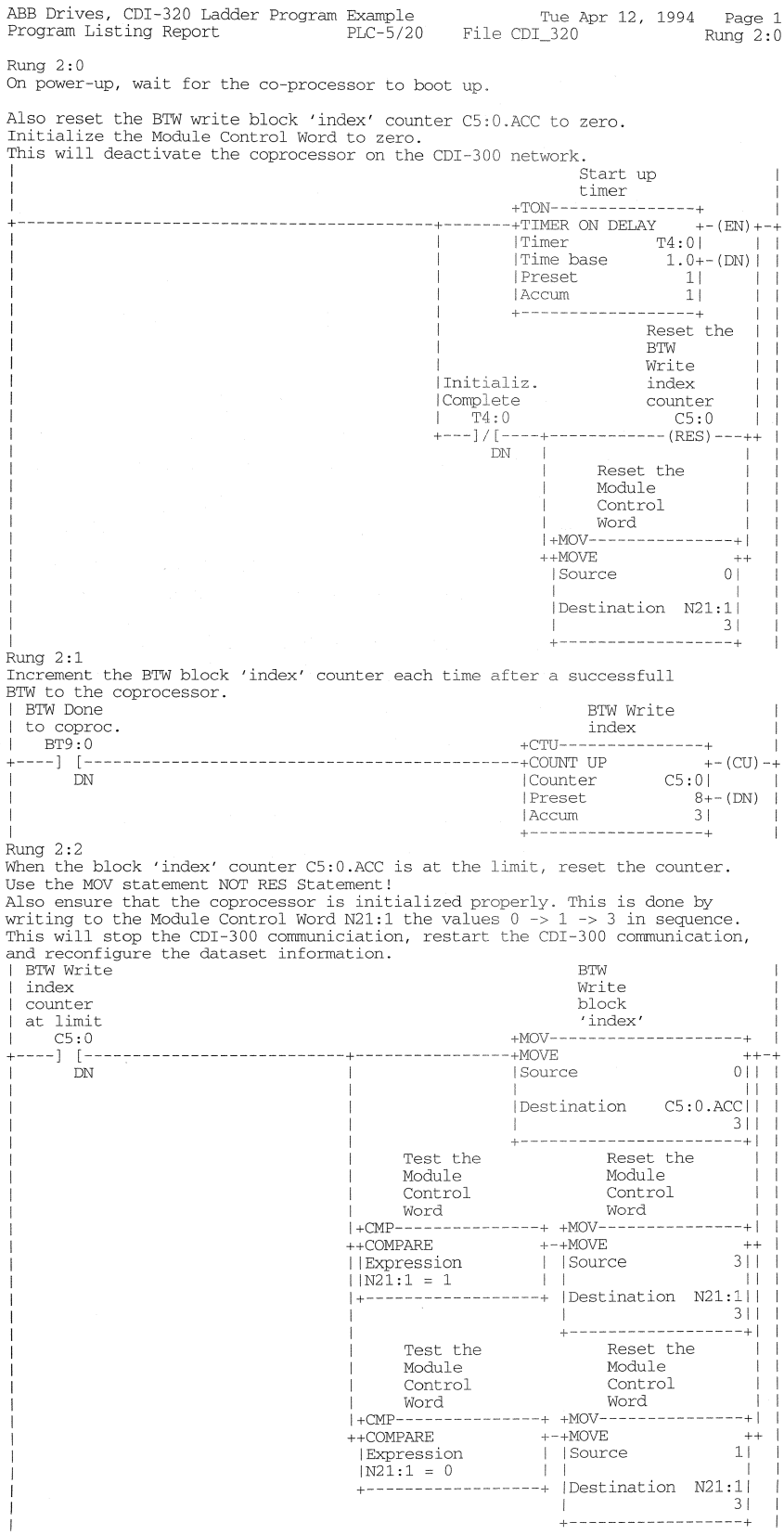
The C5:0.ACC is used to 'loop' through all the outgoing blocks. Timer T4:0 is used to allow a time for the coprocessor to initialize after the power up, and also to ensure that the Module Control Word N21:1 is written to the coprocessor in sequence with the values of 0, 1, and 3. The timer starts from the accumulated value of zero when the PLC goes to the run state.

The CMP FAL instructions after the BTR statement are used to move the received data block to the correct container area.

The application should modify only the N22 (and N21) for sending data to the coprocessor, and N31 and N32 for receiving data back from the coprocessor. Because the coprocessor expects that it sees one BTR after a BTW instruction, it is not adviseable to do direct BTWs or BTRs from the middle of ladder file.

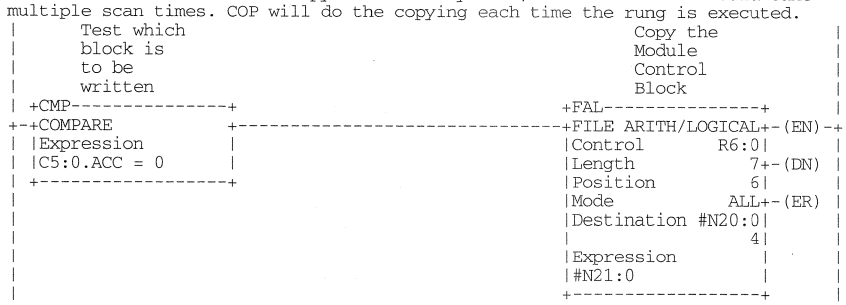
The PLC interface should be done in its own ladder file, as shown in this example.

## Ladder listing



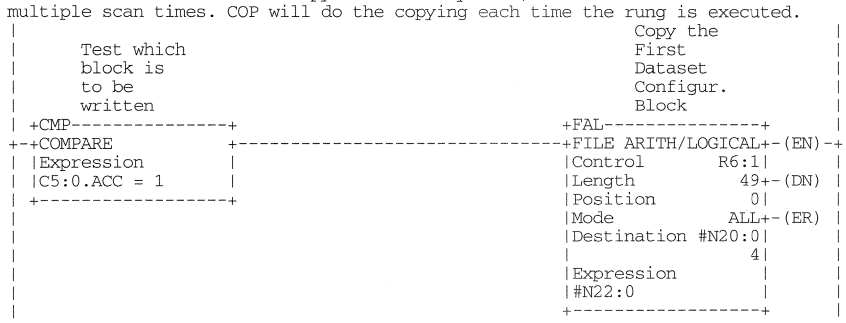
Rung 2:3
Before writing the Module Control Block (130) to the coprocessor, copy the data from N21:0 to the BTW data buffer area starting at N20:0.

Use FAL to ensure that the copy is done only once, even if the BTW would take multiple scan times. COP will do the copying each time the rung is executed.



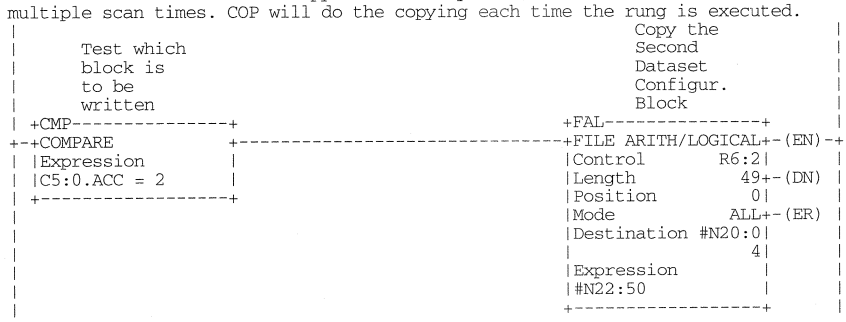

Rung 2:4
Before writing the Dataset Configuation Block (131) to the coprocessor, copy the data from N22:0 to the BTW data buffer area starting at N20:0.

Use FAL to ensure that the copy is done only once, even if the BTW would take multiple scan times. COP will do the copying each time the rung is executed.

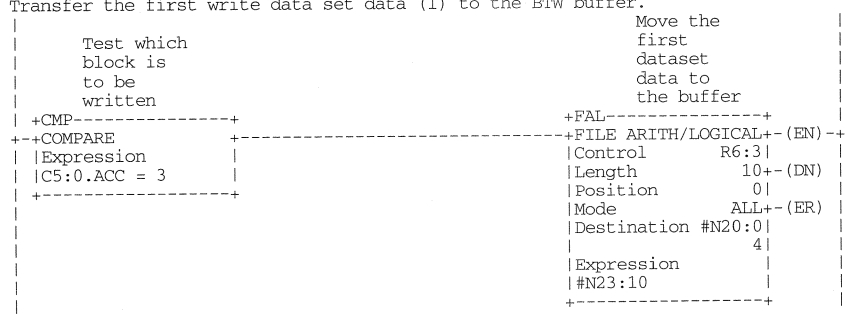


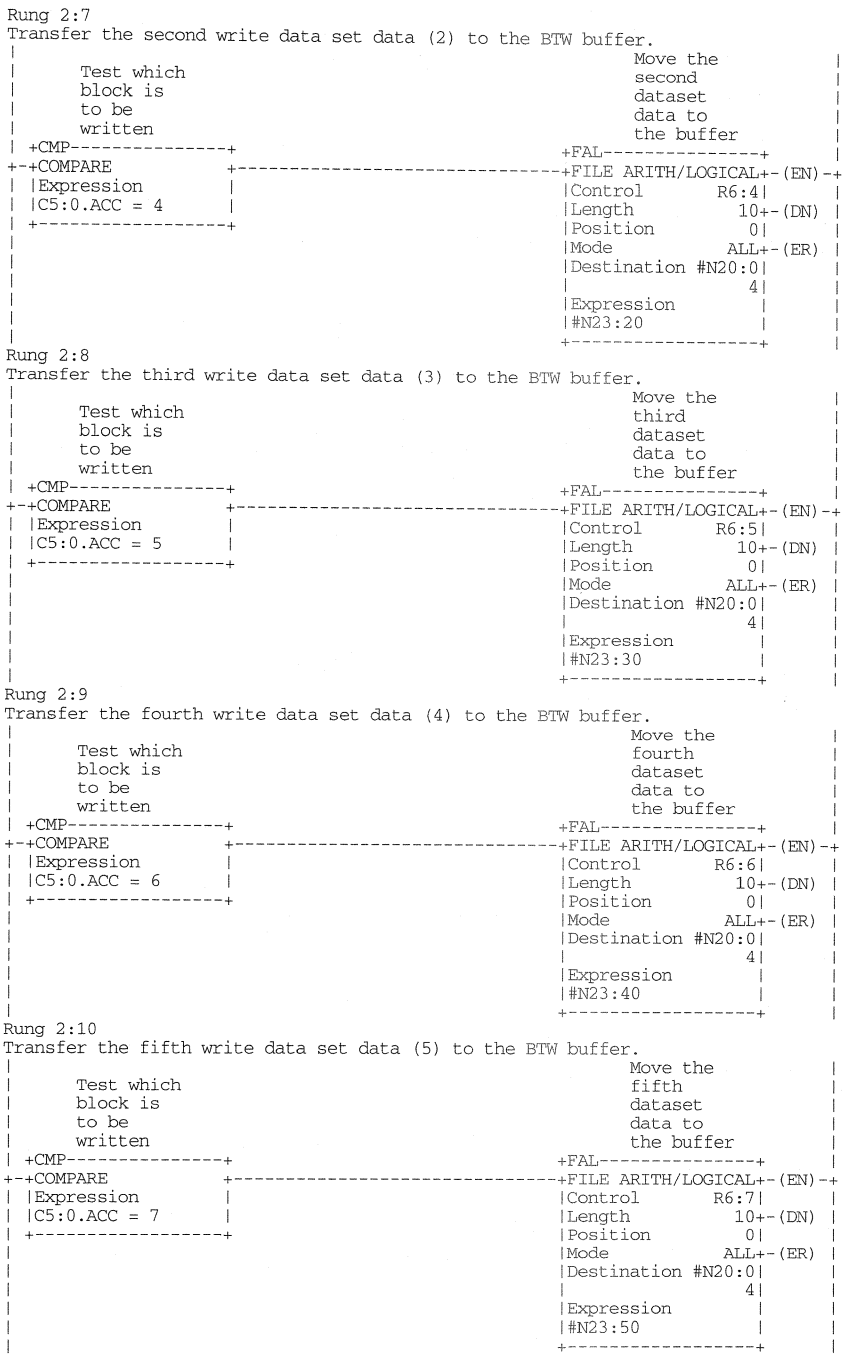

Rung 2:5
Before writing the Dataset Configuation Block (132) to the coprocessor, copy the data from N22:50 to the BTW data buffer area starting at N20:0.

Use FAL to ensure that the copy is done only once, even if the BTW would take multiple scan times. COP will do the copying each time the rung is executed.




Rung 2:6
Transfer the first write data set data (1) to the BTW buffer.

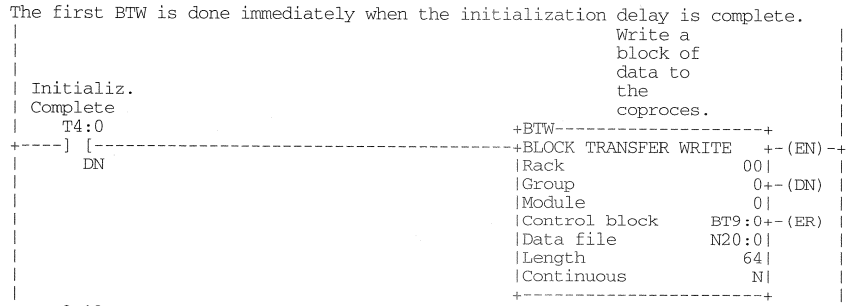



Rung 2:7
Transfer the second write data set data (2) to the BTW buffer.
Test which block is to be written
CMP
COMPARE
Expression
C5:0.ACC = 4
Move the second dataset data to the buffer
FAL
FILE ARITH/LOGICAL
Control R6:4
Length 10
Position 0
Mode ALL
Destination #N20:0 4
Expression #N23:20
EN
DN
ER
Rung 2:8
Transfer the third write data set data (3) to the BTW buffer.
Test which block is to be written
CMP
COMPARE
Expression
C5:0.ACC = 5
Move the third dataset data to the buffer
FAL
FILE ARITH/LOGICAL
Control R6:5
Length 10
Position 0
Mode ALL
Destination #N20:0 4
Expression #N23:30
EN
DN
ER
Rung 2:9
Transfer the fourth write data set data (4) to the BTW buffer.
Test which block is to be written
CMP
COMPARE
Expression
C5:0.ACC = 6
Move the fourth dataset data to the buffer
FAL
FILE ARITH/LOGICAL
Control R6:6
Length 10
Position 0
Mode ALL
Destination #N20:0 4
Expression #N23:40
EN
DN
ER
Rung 2:10
Transfer the fifth write data set data (5) to the BTW buffer.
Test which block is to be written
CMP
COMPARE
Expression
C5:0.ACC = 7
Move the fifth dataset data to the buffer
FAL
FILE ARITH/LOGICAL
Control R6:7
Length 10
Position 0
Mode ALL
Destination #N20:0 4
Expression #N23:50
EN
DN
ER

Rung 2:11
Write all the blocks in sequence to the coprocessor. The first word of the BTW data block is the block index.

Chain the BTW and the BTR together in sequence. After a BTW do always one BTR.

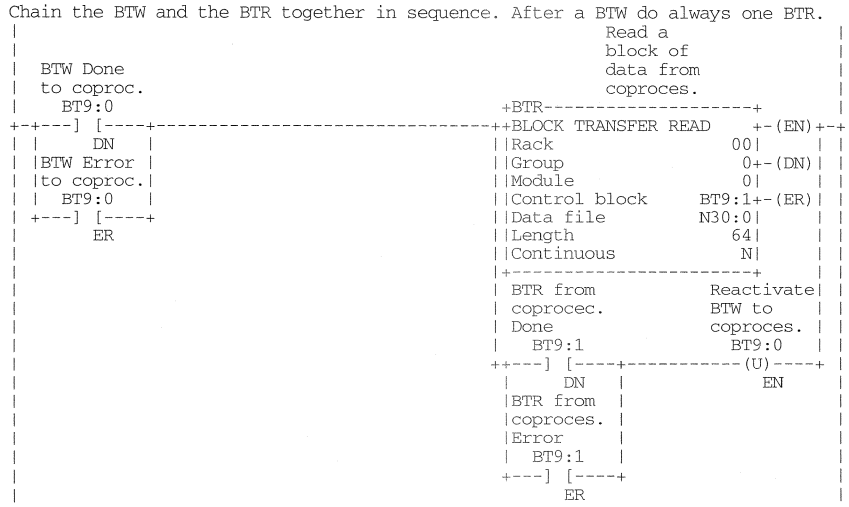
The first BTW is done immediately when the initialization delay is complete.



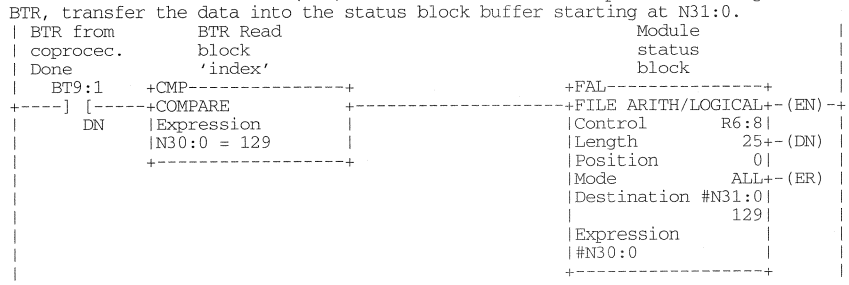

Rung 2:12
Read all the blocks in sequence from the coprocessor. The first word of the BTR data block is the block index.

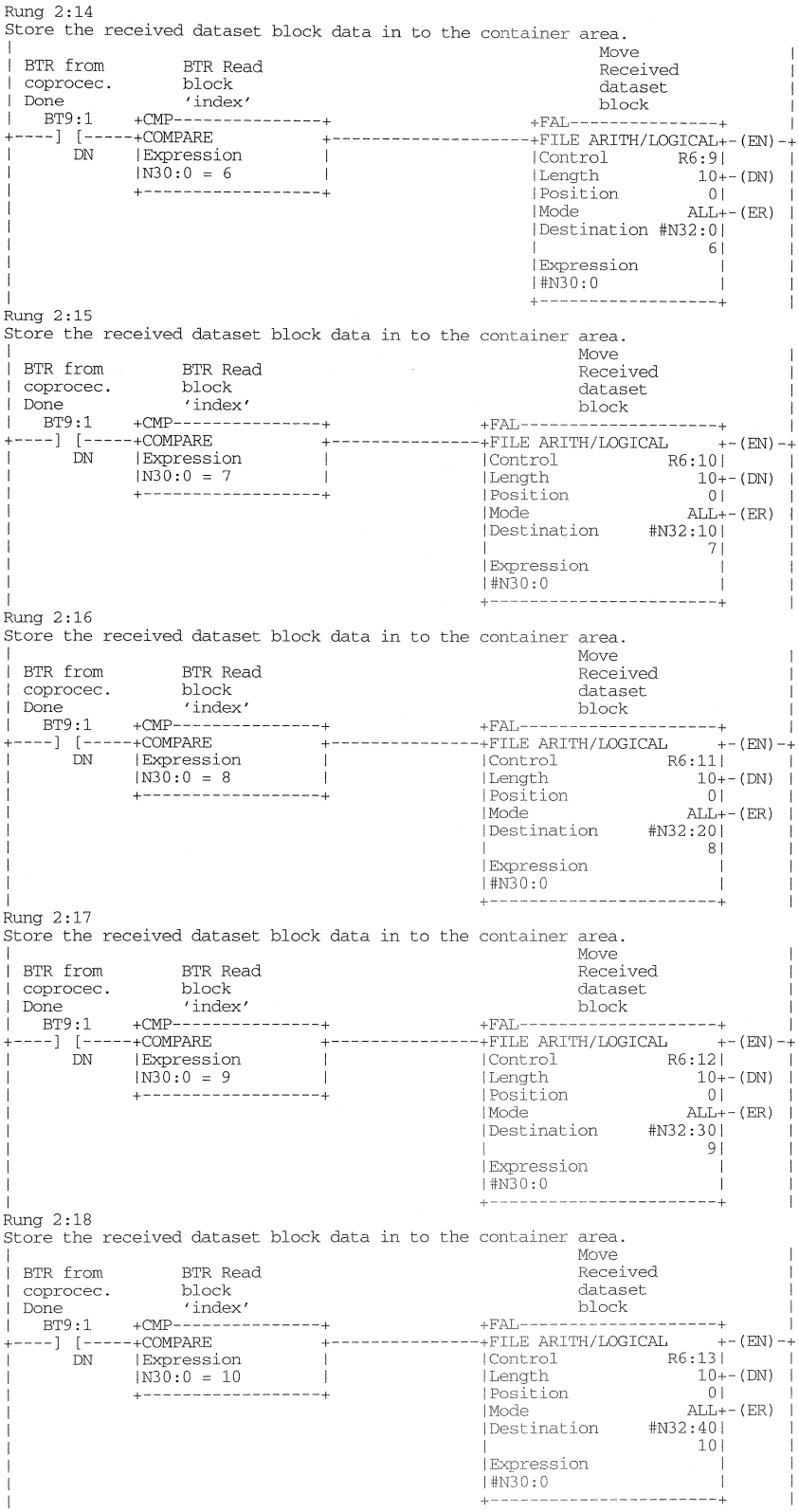
Once the BTW is completed, do the BTR. When the BTR is complete, restart the BTW by setting the BT9:0.EN bit to zero.

Chain the BTW and the BTR together in sequence. After a BTW do always one BTR.




Rung 2:13
If the Module Status Block (129) was returned from the coprocessor using the BTR, transfer the data into the status block buffer starting at N31:0.




Rung 2:14
Store the received dataset block data in to the container area.
BTR from coprocec. Done
BTR Read block 'index'
Move Received dataset block
BT9:1
CMP
COMPARE
Expression
N30:0 = 6
FAL
FILE ARITH/LOGICAL
Control R6:9
Length 10
Position 0
Mode ALL
Destination #N32:0
6
Expression
#N30:0
EN
DN
ER
Rung 2:15
Store the received dataset block data in to the container area.
BTR from coprocec. Done
BTR Read block 'index'
Move Received dataset block
BT9:1
CMP
COMPARE
Expression
N30:0 = 7
FAL
FILE ARITH/LOGICAL
Control R6:10
Length 10
Position 0
Mode ALL
Destination #N32:10
7
Expression
#N30:0
EN
DN
ER
Rung 2:16
Store the received dataset block data in to the container area.
BTR from coprocec. Done
BTR Read block 'index'
Move Received dataset block
BT9:1
CMP
COMPARE
Expression
N30:0 = 8
FAL
FILE ARITH/LOGICAL
Control R6:11
Length 10
Position 0
Mode ALL
Destination #N32:20
8
Expression
#N30:0
EN
DN
ER
Rung 2:17
Store the received dataset block data in to the container area.
BTR from coprocec. Done
BTR Read block 'index'
Move Received dataset block
BT9:1
CMP
COMPARE
Expression
N30:0 = 9
FAL
FILE ARITH/LOGICAL
Control R6:12
Length 10
Position 0
Mode ALL
Destination #N32:30
9
Expression
#N30:0
EN
DN
ER
Rung 2:18
Store the received dataset block data in to the container area.
BTR from coprocec. Done
BTR Read block 'index'
Move Received dataset block
BT9:1
CMP
COMPARE
Expression
N30:0 = 10
FAL
FILE ARITH/LOGICAL
Control R6:13
Length 10
Position 0
Mode ALL
Destination #N32:40
10
Expression
#N30:0
EN
DN
ER

## *Datafiles*

The following listing shows a 'snap-shot' of the datafiles used with the ladder file above.

The files have been described in the Introduction at the beginning of this chapter.

```
ABB Drives, CDI-320 Ladder Example                    Wed Apr 13, 1994   Page 1
Data Table Report          PLC-5/20     File CDI_320        Data Table File N20:0

  Address     0      1      2      3      4      5      6      7      8      9
N20:0           4      0   3000      0      0      0      0      0      0      0
N20:10          0      0      0      0      0      0      0      0      0      0
N20:20          0      0      0      0      0      0      0      0      0      0
N20:30          0      0      0      0      0      0      0      0      0      0
N20:40          0      0      0      0      0      0      0      0      0      0
N20:50          0      0      0      0      0      0      0      0      0      0
N20:60          0      0      0      0
```

```
ABB Drives, CDI-320 Ladder Example                    Wed Apr 13, 1994   Page 2
Data Table Report          PLC-5/20     File CDI_320        Data Table File N21:0

  Address     0      1      2      3      4      5      6
N21:0         130      3     20  12599  14129  11588  19779
```

```
ABB Drives, CDI-320 Ladder Example                    Wed Apr 13, 1994   Page 3
Data Table Report          PLC-5/20     File CDI_320        Data Table File N22:0

  Address     0      1      2      3      4      5      6      7      8      9
N22:0         131     20      4   1000     21      4   1000     22      4   1000
N22:10         23      4   1000     24      4   1000     10      4      0     11
N22:20          4      0     12      4      0     13      4      0     14      4
N22:30          0      0      0      0      0      0      0      0      0      0
N22:40          0      0      0      0      0      0      0      0      0      0
N22:50        132      0      0      0      0      0      0      0      0      0
N22:60          0      0      0      0      0      0      0      0      0      0
N22:70          0      0      0      0      0      0      0      0      0      0
N22:80          0      0      0      0      0      0      0      0      0      0
N22:90          0      0      0      0      0      0      0      0      0
```

```
ABB Drives, CDI-320 Ladder Example                    Wed Apr 13, 1994   Page 4
Data Table Report          PLC-5/20     File CDI_320        Data Table File N23:0

  Address     0      1      2      3      4      5      6      7      8      9
N23:0           0      0      0      0      0      0      0      0      0      0
N23:10          1      0   1000      0      0      0      0      0      0      0
N23:20          2      0   2000      0      0      0      0      0      0      0
N23:30          3      0   2000      0      0      0      0      0      0      0
N23:40          4      0   3000      0      0      0      0      0      0      0
N23:50          5      0   4000      0      0      0      0      0      0      0
```

ABB Drives, CDI-320 Ladder Example — Wed Apr 13, 1994 Page 6
Data Table Report — PLC-5/20 — File CDI_320 — Data Table File N30:0

| Address | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| N30:0 | 7 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N30:10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N30:20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N30:30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N30:40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N30:50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N30:60 | 0 | 0 | 0 | 0 | | | | | | |

ABB Drives, CDI-320 Ladder Example — Wed Apr 13, 1994 Page 7
Data Table Report — PLC-5/20 — File CDI_320 — Data Table File N31:0

| Address | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| N31:0 | 129 | 7 | 511 | 16400 | 0 | 0 | 0 | 0 | 9 | 0 |
| N31:10 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 31188 | -4016 |
| N31:20 | 14 | 0 | 2104 | 0 | 0 | | | | | |

ABB Drives, CDI-320 Ladder Example — Wed Apr 13, 1994 Page 8
Data Table Report — PLC-5/20 — File CDI_320 — Data Table File N32:0

| Address | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| N32:0 | 6 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N32:10 | 7 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N32:20 | 8 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N32:30 | 9 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N32:40 | 10 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Message Example

The following listing shows an example code for doing message communication between the CDI-320 software and an ACS 500 or ACH 500 drives.

This information is provided for reference only. The message communication is not available for the first release of the product.

```
ABB Drives, CDI-320 Message example                    Tue May 10, 1994   Page 1
Program Listing Report                PLC-5/20    Addr 2                Rung 4:0

Rung 4:0
Before doing any messages, wait for the station to come on-line on the CDI-300
network.

This can be seen from the Module Status Word, bit 1.
| Module                                                                    |
| Status Wrd                                                                |
| Node                                                                      |
| active on                                                                 |
| CDI-300                                                                   |
|   N31:1                                          +TON---------------+     |
+----] [-------------------------------------------+TIMER ON DELAY    +-(EN)-+
|       1                                          |Timer         T4:2|     |
|                                                  |Time base      1.0+-(DN) |
|                                                  |Preset           5|     |
|                                                  |Accum            5|     |
|                                                  +------------------+     |
Rung 4:1
Build a toggle bit for testing the messaging. 30 seconds on, 30 seconds off.
|  T4:2  T4:4                                      +TON---------------+     |
+--] [---]/[-------------------------------------------+TIMER ON DELAY    +-(EN)-+
|    DN    DN                                      |Timer         T4:3|     |
|                                                  |Time base      1.0+-(DN) |
|                                                  |Preset           5|     |
|                                                  |Accum            2|     |
|                                                  +------------------+     |
Rung 4:2
Build a toggle bit for testing the messaging. 30 seconds on, 30 seconds off.
| Activate                                                                  |
| message                                                                   |
| one                                                                       |
|   T4:3                                           +TON---------------+     |
+----] [-------------------------------------------+TIMER ON DELAY    +-(EN)-+
|      DN                                          |Timer         T4:4|     |
|                                                  |Time base      1.0+-(DN) |
|                                                  |Preset           5|     |
|                                                  |Accum            0|     |
|                                                  +------------------+     |
Rung 4:3
Send the message control block to the coprocessor.

This block tells where the data is, and where to return the reply data.
After the message write is done, set the pending bit on the reply area, so that
the read message will be active during the read.
|                                                        Message             |
| Activate  |Message                                     one                 |
| message   |one is not                                  write               |
| one       |pending                                     block               |
|    T4:3      N40:6                           +MSG--------------------+     |
+----] [--------]/[-------------------------+--+SEND/RECEIVE MESSAGE   +-(EN)+-+
|      DN         14                         |  |Control block    MG10:0+-(DN)| |
|                                            |  |                       +-(ER)| |
|                                            |  +-----------------------+     | |
|                                            |Message   |          |Message   | |
|                                            |one       |Set the   |one       | |
|                                            |write     |bit only  |pending   | |
|                                            |done      |once      |bit       | |
|                                            |  MG10:0       B3        N40:6   | |
|                                            +---] [-------[ONS]-------(L)----+ |
|                                                  DN          0         14     |
```

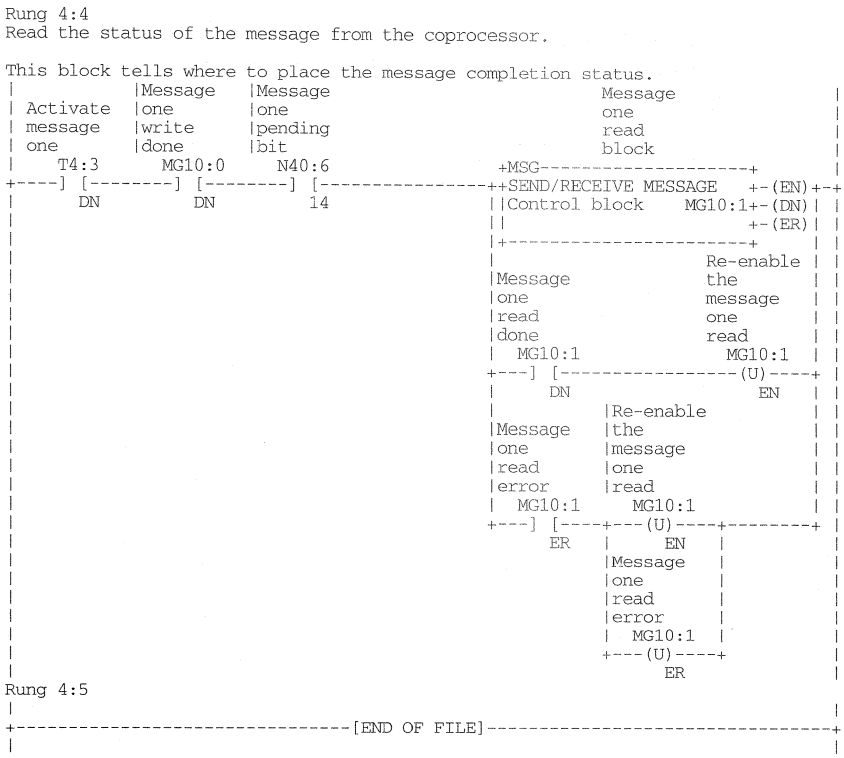


ABB Industrial Systems Inc.
Standard Drives Division
16250 West Glendale Drive
New Berlin, WI 53151
Telephone: (414) 785-3416
Fax: (414) 785-0397

ABB Industry Oy
P.O. Box 184
00381 Helsinki
FINLAND
Telephone: +358-0-5641
Telefax: +358-0-564 2681
Telex: +57-12440502 str sf

Printed in U.S.A.

CDI-320-US-04
EFFECTIVE 4/1/94