ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

User's Manual

I/O Logic Controller
(Cat. No. 1771-DR)

Price: $25.00

Important User Information

Because of the variety of uses for the solid state equipment described herein, and because of the differences between it and electromechanical equipment, you must satisfy yourself as to its acceptability for each of your applications. In no event will Allen-Bradley Company be responsible or liable for indirect or consequential damages that may result from installation or use of this equipment.

The illustrations, charts, and layout examples shown in this manual are intended solely to help you understand the text, not to guarantee operation. Because of the many variables and requirements associated with any particular installation, Allen-Bradley Company will not assume responsibility for actual use based upon illustrations of applications.

No patent liability is assumed by Allen-Bradley Company with respect to use of information, circuits, equipment, or software described in this text.

Table of Contents

Chapter	Title	Page

4 Module/Processor Communication

Chapter 1 Using This Manual

Chapter objectives

Read this chapter to familiarize yourself with this manual. It tells you how to use the manual properly and efficiently.

What this manual contains

Chapter	Title	Topics Covered
1	Using this manual	Manual's purpose, audience and contents
2	Introducing the I/O Logic Controller	Module description, features and hardware components
3	Configuring and installing your module	Keying, power requirements, wiring and installation
4	Module/Processor communication	Word and file parameters of block transfer data
5	Troubleshooting	Troubleshooting guide
Appendices		
A	Block transfer timing	Instructions for determining block-transfer timing
B	Hex input conversion table	
C	Block-transfer ladder diagram examples	Examples of block-transfer programming

Audience

We assume that you know how to:

- program and operate an Allen-Bradley programmable controller.
- program block-transfer instructions.

If you don't know how to do either of these, read the user's manual for your processor. Refer to Allen-Bradley Industrial Computer Group Publication Index (publication SD 499) for a complete list of publications.

Definitions of major terms

To make this manual easier for you to read and understand, we avoid repeating product names where possible. We refer to:

- programmable controllers, as the "processor".
- the I/O Logic Controller, as the "controller".

For a list of PC words and their definitions, contact your Allen- Bradley sales engineer or distributor for publication PCGI-7.2.

Warnings and cautions

You will see three types of precautionary statements as you read this manual: Important, CAUTION and WARNING.

- **Important** tells you about specific areas of concern when operating your equipment.
- **CAUTION** tells you when equipment may be damaged if procedures are not followed properly.
- **WARNING** tells you when people may be injured if procedures are not followed properly.

Related publications

Consult the Allen-Bradley Industrial Computer Group Publication Index (SD 499) if you would like more information about your modules or programmable controllers. This index lists all available publications for Allen-Bradley programmable controller products. Also refer to Programmable Controller Grounding and Wiring Guidelines (publication number 1770-4.1) if necessary.

Chapter summary

This chapter told you how to use this manual efficiently. The next chapter introduces you to the I/O Logic Controller.

Chapter 2 Introducing the I/O Logic Controller

Chapter objectives

This chapter describes:

- example applications of the controller.
- functions of the controller.
- compatible Allen-Bradley processors and I/O devices.
- status indicators, fuses, terminals and specifications.

Example applications

Use the I/O Logic Controller Module (Cat. No. 1771-DR) in the following applications:

- high speed presence sensing
- high speed canning, bottling, packaging
- conveyor systems for material sorting
- counting high speed events for a selected time period

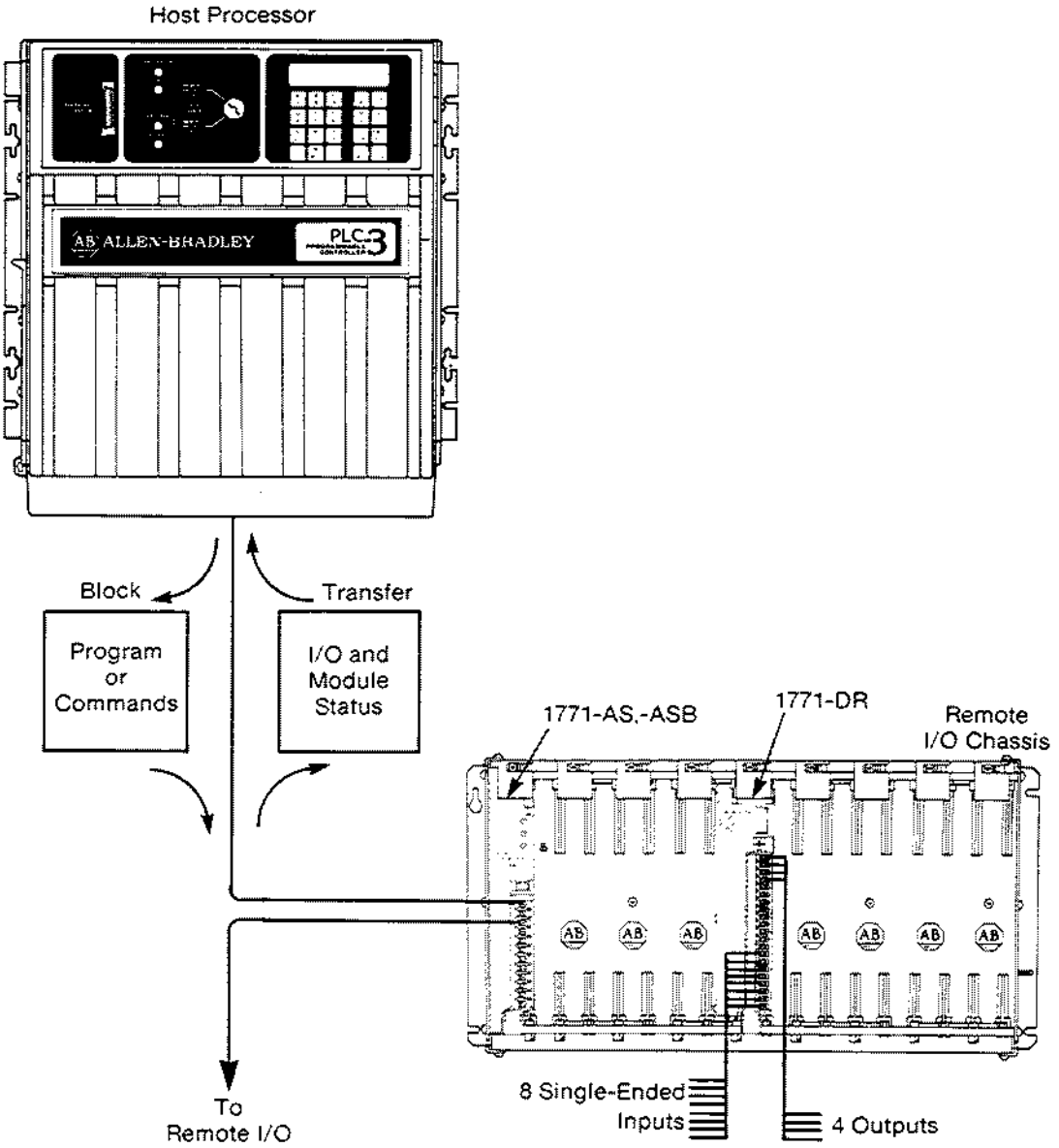
Module functions

The I/O Logic Controller Module monitors inputs and controls outputs based on the program downloaded from the host processor. It reports I/O status and self-diagnostics back to the host processor (figure 2.1). The controller offers the following functions:

- monitors up to eight discrete inputs.
- accepts input pulses within programmable ranges from 50us to 10ms.
- controls up to four discrete outputs.
- provides response times of less than one millisecond for all four outputs at the same time when the time constant of its input filter is 50 us.
- processes block-transfers to or from the host processor,
  - return I/O status, accumulated count values, detected programming errors and self diagnostics to the host processor.
  - receive commands, programmed inputs or programs from the host processor.
- masks selected inputs so the host processor can substitute programmed bits in their place.
- forces and/or disables selected outputs upon command.
- accepts on-line changes for synchronization with other events supervised by the host processor.

Module functions (continued)

Figure 2.1
Block diagram of I/O Logic Controller



Compatible processors

You can use the module with any Allen-Bradley processor that uses block transfer programming in both local and remote 1771 I/O systems. Processors that are compatible with the module include:

- Mini-PLC-2 (cat no. 1772-LN3)
- Mini-PLC-2/05 (cat. no. 1772-LS, -LSP)
- Mini-PLC-2/15 (cat. no. 1772-LV)
- Mini-PLC-2/16 (cat. no. 1772-LX,-LXP)

Compatible processors (continued)

- Mini-PLC-2/17 (cat. no. 1772-LW,-LWP)
- PLC-2/20 (cat. no 1772-LP1, -LP2)
- PLC-2/30 (cat. no. 1772-LP3)
- PLC-3 (cat. no 1775-L1, -L2)
- PLC-3/10 (cat. no. 1775-LP4, -LP8)

Compatible I/O devices

Typical 12 to 24 V DC 2-wire or 3-wire input switches that you can use with the module include:

- DC proximity switches (A-B Bulletin 871C)
- photoelectric sensors (A-B Bulletin 880L)
- DC limit, float, pushbutton or selector switches (A-B Bulletin 800, 801 and 840)

Typical output devices that the module can control include:

- DC relays (A-B Bulletin 700)
- DC solenoids
- indicators (A-B Bulletin 800)

Module description

The I/O Logic Controller Module is an intelligent I/O module that offers high speed response independent of the host processor's scan time.

Status indicators

The module has three LED status indicators on the left LED display (figure 2.2).

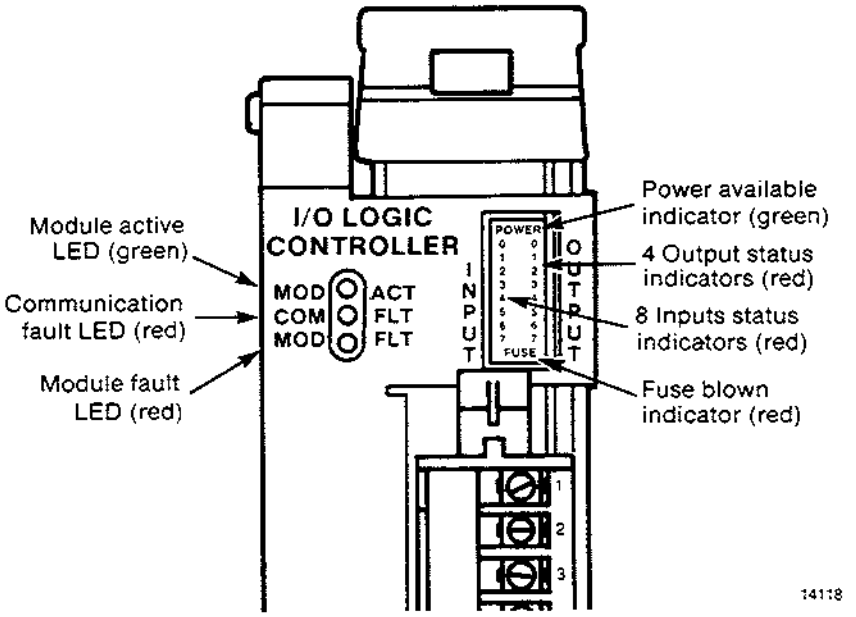
- MOD ACT (green) - indicates module is up and running.
- COMM FLT (red) - indicates a communication fault.
- MOD FLT (red) - indicates a hardware fault.

The module also has 14 LED status indicators on the right LED display (figure 2.2).

- 4 red output status indicators (one for each output) light when the corresponding output circuitry is energized.
- 8 red input status indicators (one for each input) light when the corresponding input circuitry is energized.
- Power (green) indicates the modules output circuits are functioning.
- Fuse (red) indicates one or more output fuses have blown.

Status indicators *(continued)*

Figure 2.2
LED status indicators



Output fuses

Your module has four 3A rectifier fuses (one per output) located on the right printed circuit board.

Terminal identification

Figure 3.1 shows each terminal on the module.

Specifications

Number and type of inputs
- 8 single-ended
- Current sinking

Input voltage range and logic state
- Logic 0: 0 to 4.0 V DC
- Logic 1: 10.0 to 24.0 V DC

Input current
- 6 mA @ 12 V DC
- 12 mA @ 24 V DC

Input isolation
- 1500 V DC

Digital input filtering
- Range: 50 to 9999 us
- Selected by input pairs

Input frequency
- 1 KHz

Number and type of outputs
- 4 single-ended
- Current sourcing

Output voltage range (customer supply)
- 5 to 24 V DC

Output current rating (per output)
- Continuous: 2 A
- Surge: 4 A for 10 ms

VA rating
- 48 W per output
- 192 W per module

On-state voltage drop
- 0.25 V DC per output

Off-state leakage current
- 1 mA per output

Output isolation
- 1500 V DC

Output fusing (per output)
- 3 A recitifier fuse (Littelfuse 322002, Buss GBB003 or equivalent)

Backplane current
- 1.1 A @ 5 V DC

Environmental conditions
- Operating temperature 0° to 60° C (32° to 140° F)
- Storage temperature -40° to 85° C (-40° to 185° F)
- Relative humidity 5% to 95% (without condensation)

Module location
- Any 1771 I/O chassis I/O group

Keying
- Left connector: between 10 and 12 between 30 and 32
- Right connector: between 10 and 12 between 34 and 36

Wiring arm
- (Cat. No. 1771-WG)

Chapter summary

This chapter described the I/O Logic Controller Module, its functions and applications, and the processors and I/O devices with which it is compatible. The next chapter tells you how to configure and install the module.

Chapter 3 Installing Your Module

Chapter objectives

This chapter tells you how to:

- key the module
- power controller input circuitry and output devices
- make wiring arm connections
- install the module.

Keying

Plastic keying bands are shipped with each I/O chassis. These bands ensure that only a selected type of module can be placed in a particular I/O chassis module slot. They also help to align the module with the backplane connector.

Each module is slotted at its rear edge. The position of the keying bands must correspond to these slots to allow insertion of the module. Place the keying bands on the upper backplane connectors between the numbers at the right of the connectors.

- Left connector
  between 10 and 12
  between 30 and 32
- Right connector
  between 10 and 12
  between 34 and 36

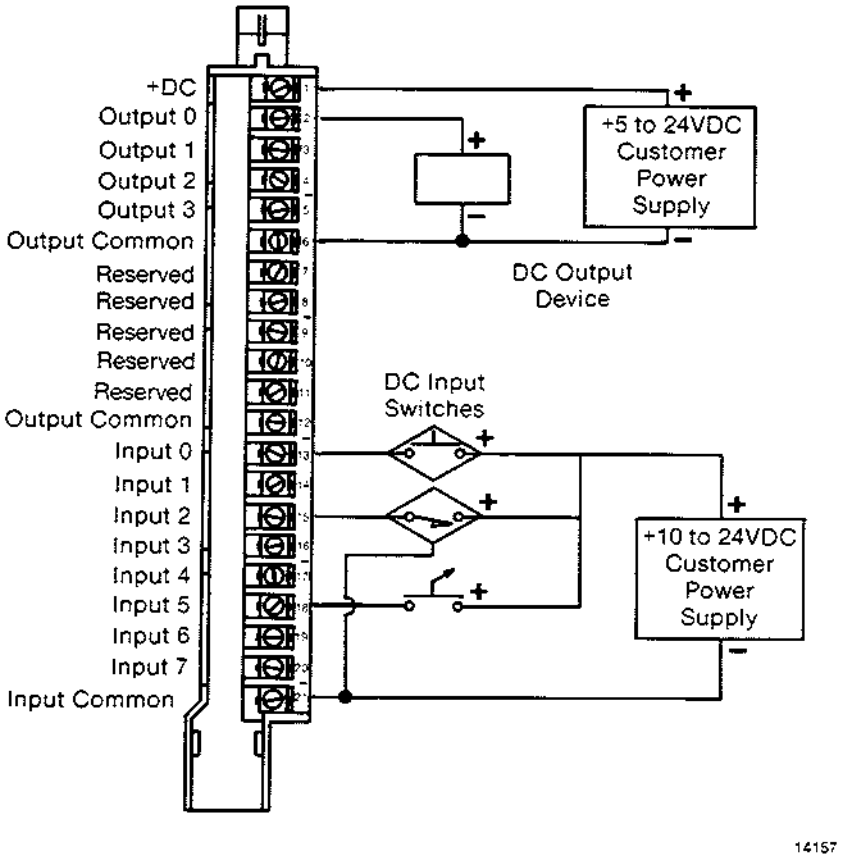
Power requirements

Normally you must provide a minimum of two external power supplies: one to power input circuitry and one to power output devices. You can, however, use one supply if common voltage for both inputs and outputs is acceptable and isolation between inputs and outputs is not necessary.

Input power supply

Input circuitry does not require the direct hookup of a power supply. Connect the (+) side of the supply to the input device and the (-) side to input common, terminal 21 of the wiring arm (figure 3.1).

Figure 3.1
Connection diagram for input/output devices and power supplies



Output power supply

To power the four outputs (figure 3.1) connect a 5 to 24 V DC supply to terminal 1 and terminal 6 of the wiring arm.

Wiring-arm connections

Refer to figure 3.1 for connection diagrams for input and output devices. There are two output commons associated with outputs 0 through 3 (terminals 6 and 12). One 21 point swing arm (cat. no. 1771-WG) provides all the connection terminals.

Installing your module

Now that you've determined the power requirements, keying, and wiring for your module, you can use the following procedure to install it.

Refer to Programmable Controller Grounding and Wiring Guidelines (pub. no. 1770-4.1) for proper grounding and wiring methods to install your module.

WARNING: Remove power from the 1771 I/O chassis backplane and wiring arm before removing or installing an I/O module.

- Failure to remove power from the backplane or wiring arm could cause module damage, degradation of performance or injury.
- Failure to remove power from the backplane could cause injury or equipment damage due to possible unexpected operation.

WARNING: Install the module in the I/O chassis so that both halves of the module are in the same I/O group when you select 2-slot addressing. Failure to observe this rule will result in faulty module operation and/or damage to the module circuitry with possible injury to personnel.

When installing your module in an I/O chassis:

1. Turn off power to the I/O chassis.
2. Open the module locking latch on the I/O chassis and insert the module into the slot keyed for it.
3. Firmly press to seat the module into its backplane connector.

Installing your module (continued)

CAUTION: Do not force the module into the backplane connector. You can damage the connector and/or the module. If you can't seat the module with firm pressure, check alignment and keying.

4. Snap the chassis latch over the top of the module to secure its positon.
5. Connect the wiring arm to the module.

Response to an external fault

Your module operates independent of the host processor except for downloading programs or commands and reporting status. If a processor fault or I/O communications fault occurs, the module either continues or stops operation depending on how you set the last state switch in the module's I/O chassis.

If you set the last state switch to turn outputs off, the module stops operating and turns its outputs off.

If you set the last state switch to hold outputs in last state, the module continues operating.

Electrostatic discharge

Electrostatic discharge can damage integrated circuits or semiconductors in your module if you touch backplane connector pins. Avoid electrostatic damage by observing the following precautions:

- Touch a grounded object to discharge yourself before handling the module.
- Do not touch the backplane connector or connector pins.
- When not in use, keep the module in its static-shield bag.

CAUTION: Electrostatic discharge can degrade performance or damage the module. Handle as stated above.

Chapter summary

This chapter described the keying, power requirements, wiring and installation of your module. In the next chapter you will read about module/processor communication.

Chapter 4 Module/Processor Communication

Chapter objectives

This chapter describes the format of block-transfer-read and write data and includes information on programming your module.

Block-transfer

The controller module and the host processor communicate through block-transfer programming. Processors that use block-transfer programming are listed below, along with the associated user manual. Refer to the latest edition of the manual for a detailed description of block-transfer.

Processor	Publication Number
Mini-PLC-2	1772-6.8.4
Mini-PLC-2/05	1772-6.8.6
Mini-PLC-2/15	1772-6.8.2
Mini-PLC-2/16	1772-6.5.7
Mini-PLC-2/17	1772-6.5.6
PLC-2/20	1772-6.8.1
PLC-2/30	1772-6.8.3
PLC-3, -3/10	1775-6.4.1

Block-transfer-write data

You write data from the processor to the module in blocks. You can send a maximum of 64 words in one block-transfer operation. The number of words you send to the module determines how many outputs it controls and the number of input branches to define those outputs. A hex code file resides in the host processor's data table and represents the logic program of the controller. You start the hex code file with an Initialization Command (0001). The initialize command:

- prepares the internal flags for programming.
- clears the memory of the controller.
- stops any I/O processing.
- de-energizes the outputs.

Block-transfer-write data (continued)

You end the hex code file with an Execute Command (0003). The Execute Command follows the configuration data and causes the module to begin execution of the most recent program downloaded if no program errors are detected. Between the Initialize Command and Execute Command is the file for the controller's configuration and logic program.

You download this program to the controller using a block-transfer-write command.

Rung format

You must translate equivalent rungs of ladder logic to hex files for your logic controller using the following format:

1. Output instruction
2. Output select
3. Preset count for counters (if applicable)/time base for timers (if applicable)
4. Accumulated count for counters (if applicable)
5. Input branch
6. Input branch (if applicable)
7. Input branch (if applicable)

Input and Output instructions

The controller's instruction set includes the following input and output instructions.

Input instructions

—] [— Examine on	True when input switch is closed (10-24 V DC at the wiring arm)
—]/[— Examine off	True when input switch is open (0-4 V DC at the wiring arm)

Output instructions

—( )— Output energize	Turns on when at least one input branch is true.
—(L)— Output latch	Turns on when at least one input branch is true and remains on until reset by an unlatch instruction having the same output address.

Input and Output instructions (continued)

—(U)— Output unlatch	Turns off a latch instruction having the same output address when at least one input branch is true. It will remain off until set by a latch instruction. When the input conditions for both the latch/unlatch instructions are true, the unlatch instruction overrides the latch instruction.
—(UCT)— Up counter	Counts false-to-true rung transitions. A true input branch must go false before the instruction recognizes another false-to-true transition. You must enter a preset and an initial accumulated value (0 - 9999). The output energizes when the accumulated value equals or exceeds the preset. The accumulated value will increment to a maximum value of 9999 and then rollover to zero. If the accumulated value rolls to zero and is less then the preset value, the output will de-energize. The up counter must be reset by the counter reset instruction (RCT) having the same output address.
—(DCT)— Down counter	Counts down with the same characteristics as an up counter.
—(RCT)— Reset counter	Resets the up or down counter (UCT or DCT) with the same output address. It also de-energizes the output (if the preset value does not equal zero) and resets the accumulated value of the counter to zero when at least one input branch is true.
—(TMR)— Timer	Starts timing when at least one input branch changes from false to true. It remains enabled unless it is reset by the Timer Reset Command (TRS) with the same address or until the time has elapsed. The output energizes when the timer "times out". You must enter a preset between 1 ms and 10 seconds.

Input and Output instructions *(continued)*

—(TMR)-(TEC)— Timed Event Counter	The timed event counter is a combination of a counter (TEC) and a timer (TMR). Both instructions have independent branch input conditions, but the same output address. The counter increments when at least one input branch makes a false-to-true transition while the timer is running. The same input branch must become false before another independent branch becomes true to increment the counter. The output energizes and counting is suspended when the timer "times out". The accumulated count value (max 9999) is transferred to the host processor in the module's status block (words 5 through 8 depending on which outputs are selected as counters). The (TEC) instruction latches its accumulated value until reset when the (TMR) instruction is re-enabled. You must program a preset between 1 ms and 10 seconds for the (TMR) instruction. The (TMR) instruction is retentive. Once an input branch goes from false-to-true, the timer will continue timing until the preset is reached or the Timer Reset (TRS) with the same address activates.
—(TRS)— Timer reset	Resets the accumulated time value of the (TMR) timer having the same output address and de-energizes the output when at least one input branch of its rung is true.

Input and Output instructions (continued)

WARNING: Up counter (UCT), down counter (DCT), counter reset (RCT), timer (TMR) and timer reset (TRS) instructions used with the I/O Logic Controller do not operate like the timers and counters associated with Allen-Bradley processors. Refer to figures 4.4, 4.5, 4.6 and the accompanying text for descriptions of operation.

Output select

You select the output using the form FF XX, where XX = 00 through 03. The four outputs are numbered sequentially from 00 to 03.

Preset value

Use a preset value when you use UCT and/or DCT instructions. The maximum count for a preset is 9999.

Accumulated value

A counter accumulated value is generated when you use UCT, DCT or TEC instructions. The accumulated value is programmable from 0000 to 9999 BCD and is returned to the host processor's status block.

- Word 5 - BCD count for output 0
- Word 6 - BCD count for output 1
- Word 7 - BCD count for output 2
- Word 8 - BCD count for output 3

Time

You must enter a time base when you use a TMR instruction. Time base values range from 0.001 seconds (one millisecond) to 10.00 seconds. Enter 0000 for 10.00 seconds.

Program formats for various output instructions

The following tables show the hex code formats for the various output instructions:

OTE,OTL,OTU,RCT,TEC,TRS
1. Output instruction
2. Output select
3. Input branch
4. Input branch (if applicable)

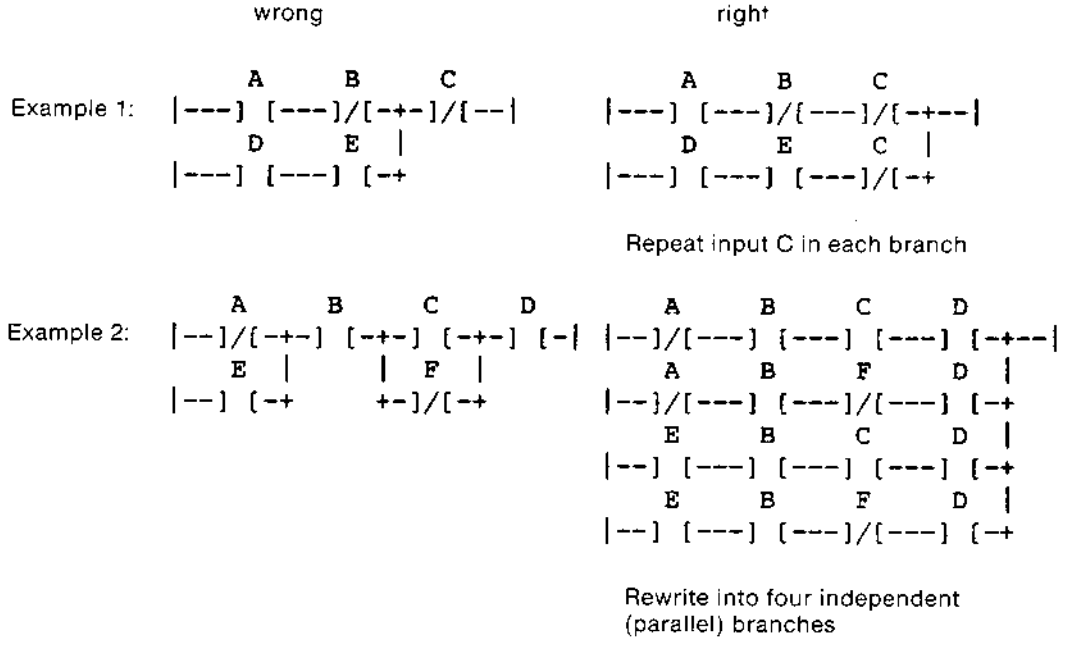
UCT,DCT
1. Output instruction
2. Output select
3. Preset value
4. Accumulated value
5. Input branch
6. Input branch (if applicable)

TMR
1. Output instruction
2. Output select
3. Preset Time
4. Input branch
5. Input branch (if applicable)

Input branch conditions

When defining the input branch conditions in the hex file, you must first convert branches containing common terms to independent (parallel) branches (figure 4.1).

Figure 4.1
Converting branches containing common terms to parallel branches



Input branch conditions (continued)

Each word in the hex code file for an independent input branch is divided into an upper and lower byte.

The upper byte represents the number of unique inputs in an independent (parallel) branch. You can program up to eight (8) inputs per branch. The lower byte represents the inputs in a branch which are low true logic (examine off).

Refer to "Appendix B - Hex input conversion table" to convert rung input conditions to a hex code equivalent. Example 1 in figure 4.1 shows the ladder logic rewritten to repeat input C in each branch. Example 2 shows the ladder logic rewritten into four independent (parallel) branches. Follow these steps to convert to a hex equivalent:

1. Define all the inputs used in the first branch (ABCD). Referring to Appendix B, ABCD converts to 0F.
2. Define all the inputs used in the first branch that are low true logic (examine off). Input A is the only examine off. Referring to Appendix B, A converts to 01. Branch one ($\overline{A}$BCD) then converts to 0F01.

$$\overline{A}BCD = 0F01$$

3. Define all the inputs used in the remaining three branches in the same way:

$$\overline{A}BD\overline{E} = 2B21$$
$$BCD\overline{E} = 1E00$$
$$BDE\overline{F} = 3A20$$

The following diagram shows the upper and lower bytes of the fourth independent branch of example 2 ($BDE\overline{F}$ = 3A20).

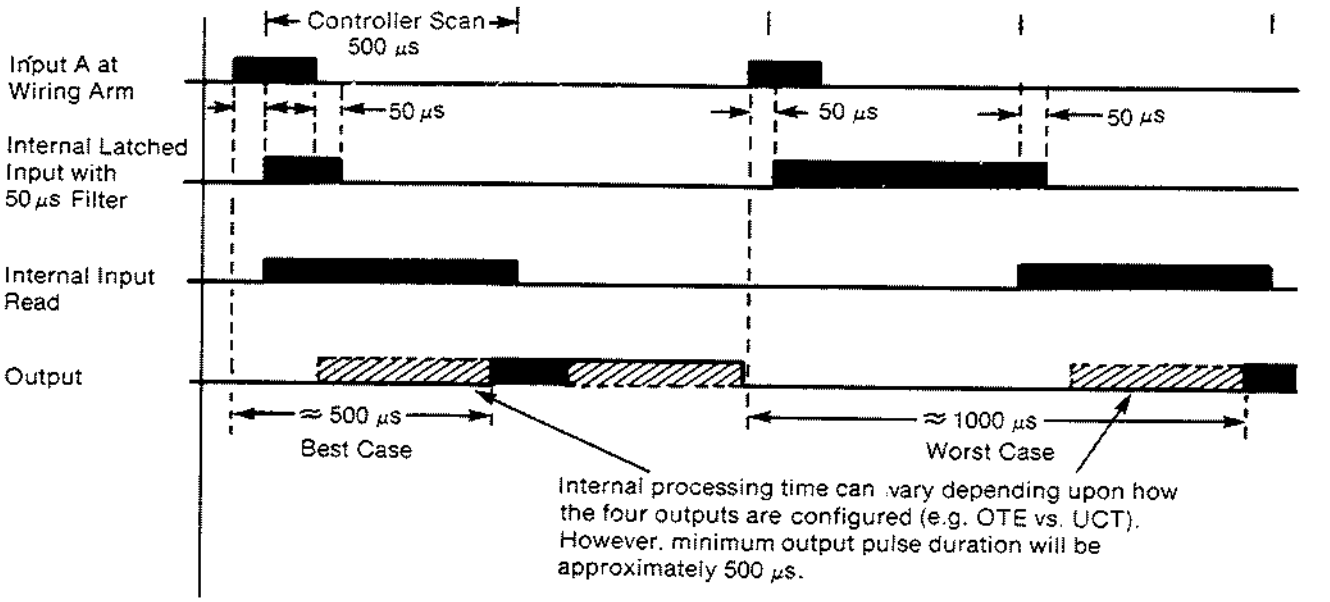
	Inputs used								Inputs with low true logic (examine off)							
Bit	17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
Input	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	H	G	F	E	D	C	B	A	H	G	F	E	D	C	B	A
	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0
Hex Equivalent			3			A						2			0	

Response time

You can select any time constant (50 µs to 9999 µs) for controlling your module's response time. Refer to figures 4.2 and 4.3 for timing diagrams for 50 µs and 600 µs input filters. Response time is the time required to sense an input, process the data and control an output. Response times for some typical time constants are:

Time constant	Maximum input frequency	Maximum response time
50 µs	1000 Hz	1.0 ms
100 µs - 499 µs	1000 Hz	1.5 ms
2.0 ms - 2.499 ms	200 Hz	3.5 ms
4.5 ms - 4.999 ms	100 Hz	6.0 ms
7.0 ms - 7.499 ms	67 Hz	8.5 ms
9.5 ms - 9.999 ms	50 Hz	11.0 ms

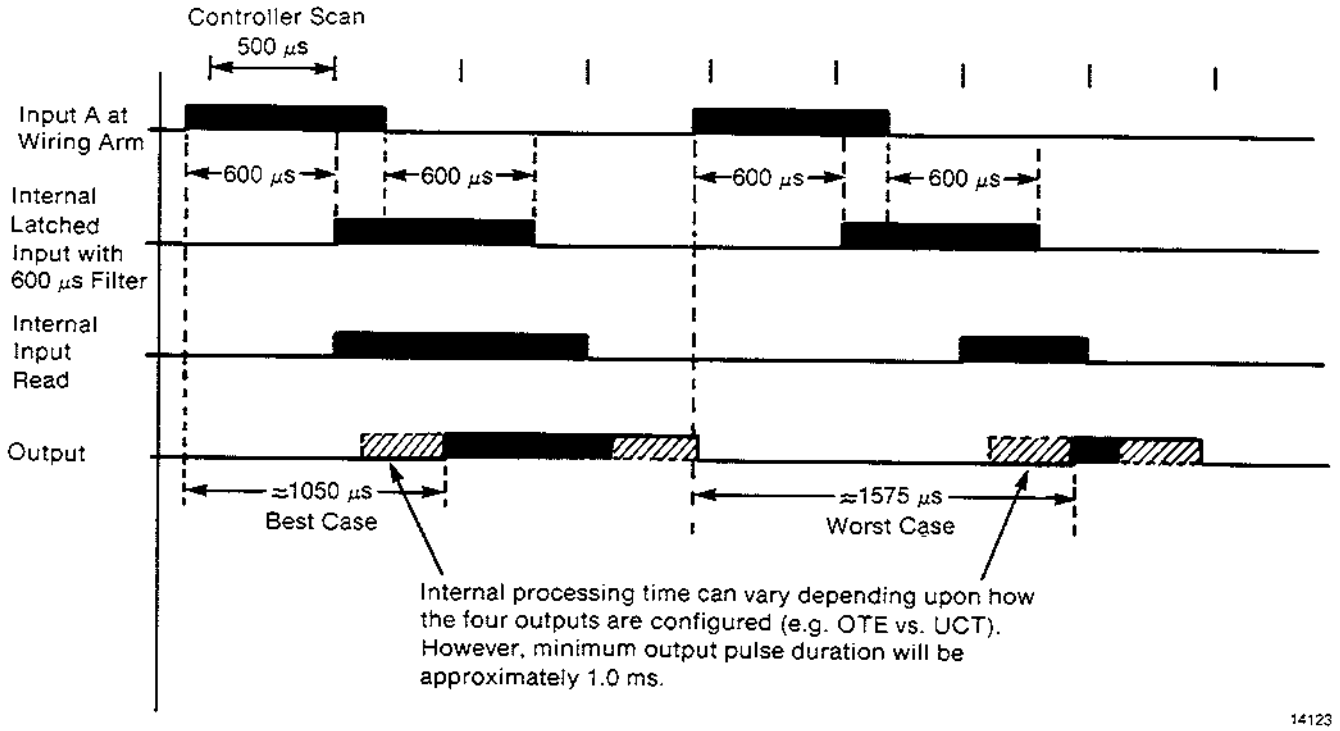
Figure 4.2
Timing diagram for a 50 µs input filter



14122

Response time *(continued)*

Figure 4.3
Timing diagram for a 600 µs input filter



You select time constants by entering BCD numbers in storage words in the host processor's data table for transfer to the controller. You must assign the same time constant to pairs of inputs (0 and 1, 2 and 3, 4 and 5, 6 and 7). If an application requires less than 8 inputs, you must still allocate 5 words (one word for set input filter instruction 0300 and 4 words for input pairs) in the hex code file for input filter constants.

Following is the programming format in the hex code file for the input filters:

Set input filter	0050
Inputs 0 and 1	0050 (50 microseconds)
Inputs 2 and 3	1000 (one millisecond)
Inputs 4 and 5	9999 (9.999 milliseconds)
Inputs 6 and 7	0000 (default value of 500 microseconds)

Response time (continued)

If no input filter value is entered at power-up the module will default to 500 μs. Use this instruction only if you want to change from the 500 μs default value.

WARNING: Up counter (UCT), down counter (DCT), counter reset (RCT), timer (TMR) and timer reset (TRS) instructions used with the I/O Logic Controller do not operate like the timers and counters associated with Allen-Bradley processors. Refer to figures 4.4, 4.5, 4.6 and the accompanying text for descriptions of operation.

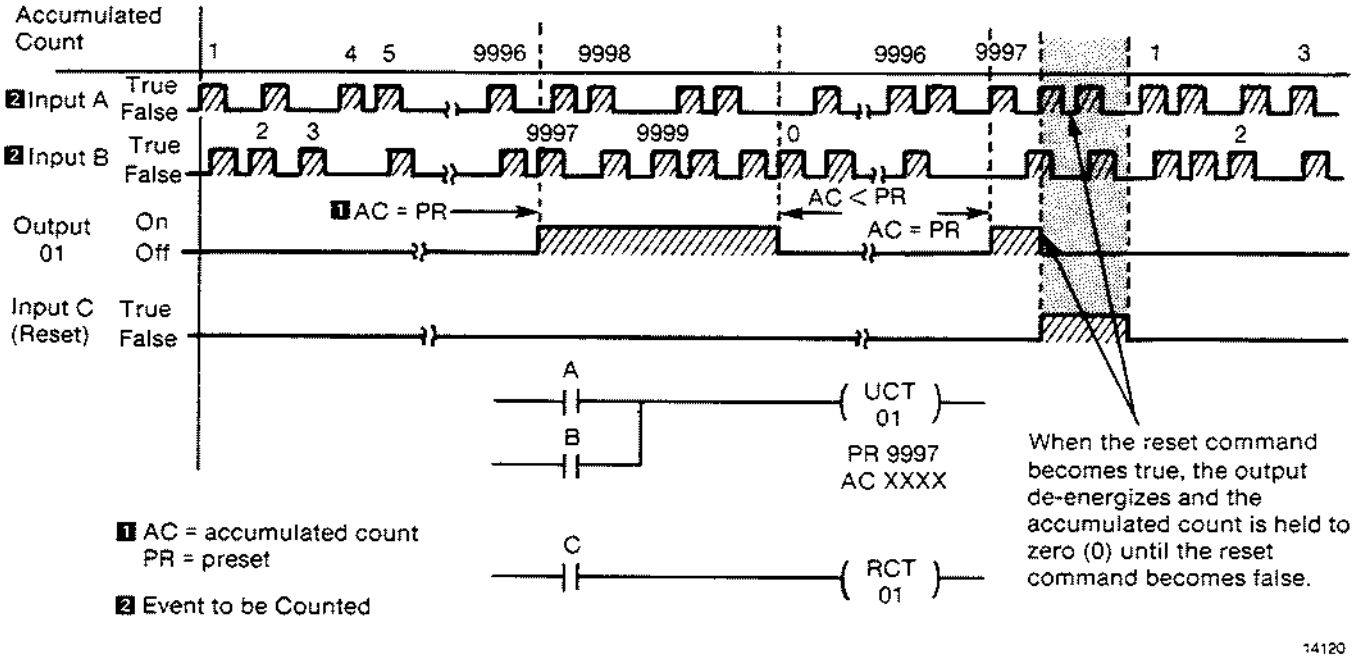
Up counter instruction (0013)

The up-counter instruction (UCT) increments its accumulated value for each false-to-true transition of an input branch (figure 4.4). Because only the false-to-true transition causes a count to be made, the input branch (rung condition) that has caused the accumulated value to increment by one must go from true-to-false before the next count is registered. When the accumulated value reaches the preset value, the output is turned on. UCT and DCT instructions for the same output must have the same preset and the same accumulated value programmed.

Unlike the timer instruction, the UCT instruction continues to increment its accumulated value after the preset value has been reached. If the accumulated value goes above 9999, it will roll to zero and, if the accumulated value is less than the preset, the output will de-energize.

Up counter instruction (0013) *(continued)*

Figure 4.4
Functional diagram for an up-counter with reset



Counter reset instruction (0015)

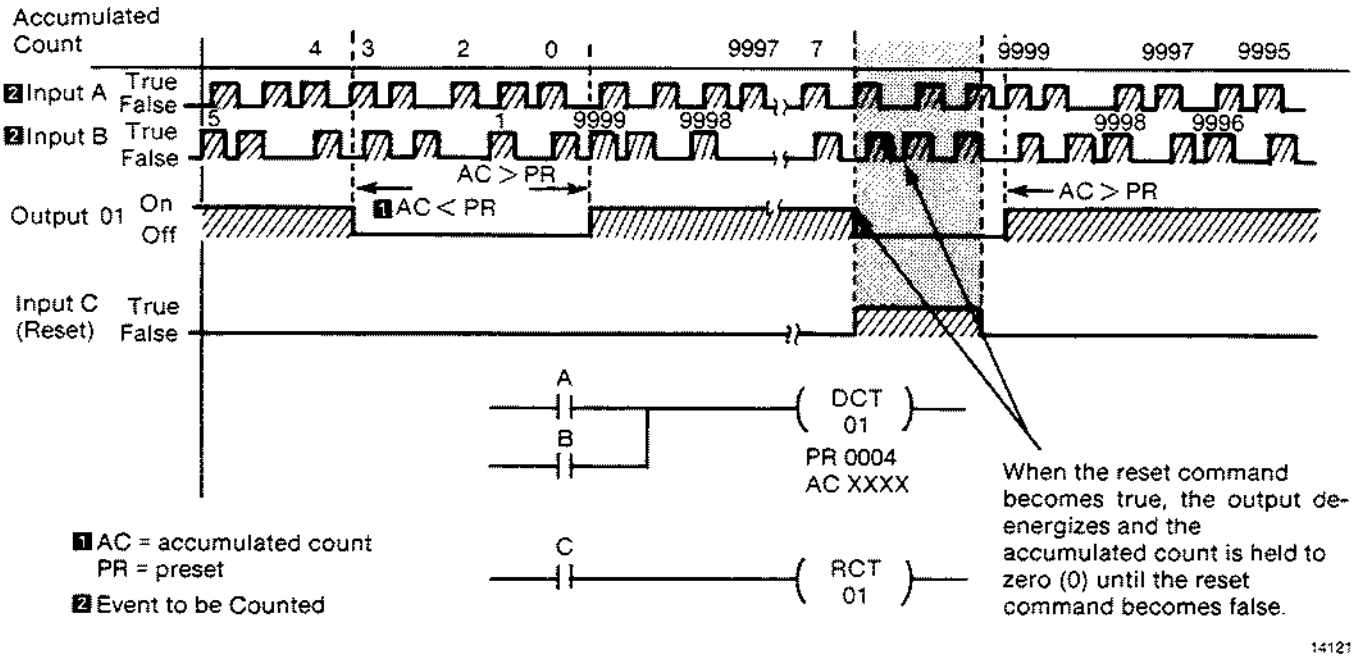
The counter reset instruction (RCT) is an output instruction that resets the UCT or DCT accumulated count value to zero and de-energizes the output if its accumulated value is less then the preset value. If the preset is zero the output is always energized. In order for the counter to be reset, the RCT instruction must have the same output designation as the counter.

Down counter instruction (0014)

The down-counter (DCT) instruction subtracts one from its accumulated value for each false-to-true transition of an input branch (figure 4.5). Because only the false-to-true transition causes a count to be made, the input branch (rung condition) that has caused the accumulated value to decrement by one must go from true-to-false before the next count is registered. The output energizes whenever the accumlated count equals or exceeds the preset value. If the accumulated value is less than the preset value, the output de-energizes. UCT and DCT instructions for the same output must have the same preset and the same accumulated value programmed.

Unlike the timer instruction, the DCT instruction continues to decrement its accumulated value after the preset has been reached. If the accumulated value goes below zero, it will roll to 9999 , and if the accumulated value is greater than or equal to the preset, the output will energize. Normally, the down-counter instruction is paired with the up-counter instruction to form an up/down counter using the same output.

Figure 4.5
Functional diagram for a down-counter with reset

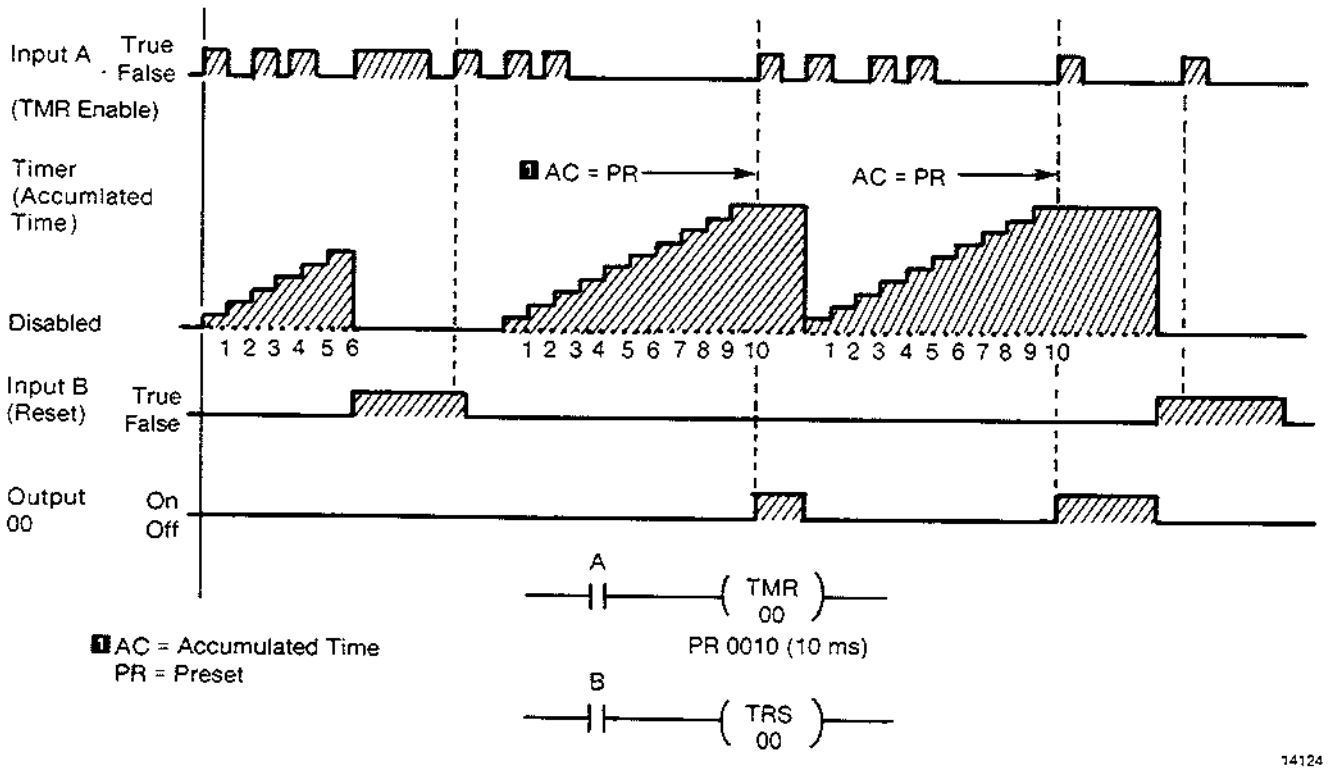


Timer instruction (0017)

Use the timer instruction (TMR) to turn on a device oince a programmed time interval has timed out (figure 4.6). Once the input branch conditions make a false-to-true transition, the timer begins to count one millisecond time base intervals. The timer continues to count one millisecond time base intervals regardless of changing input branch conditions. The timer will only discontinue timing if it is reset by the (TRS) timer reset instruction. The accumulated time is reset at this time. The timer is re-enabled on the next false-to-true transition of its input condition (as long as the TRS is no longer true).

When the accumulated time equals the programmed preset time (1 ms to 10 seconds), the timer stops incrementing its accumulative time and energizes the output. If a false-to-true transition of the input occurs after the output has been energized, the output is de-energized and the accumulated time is reset to zero and begins timing again.

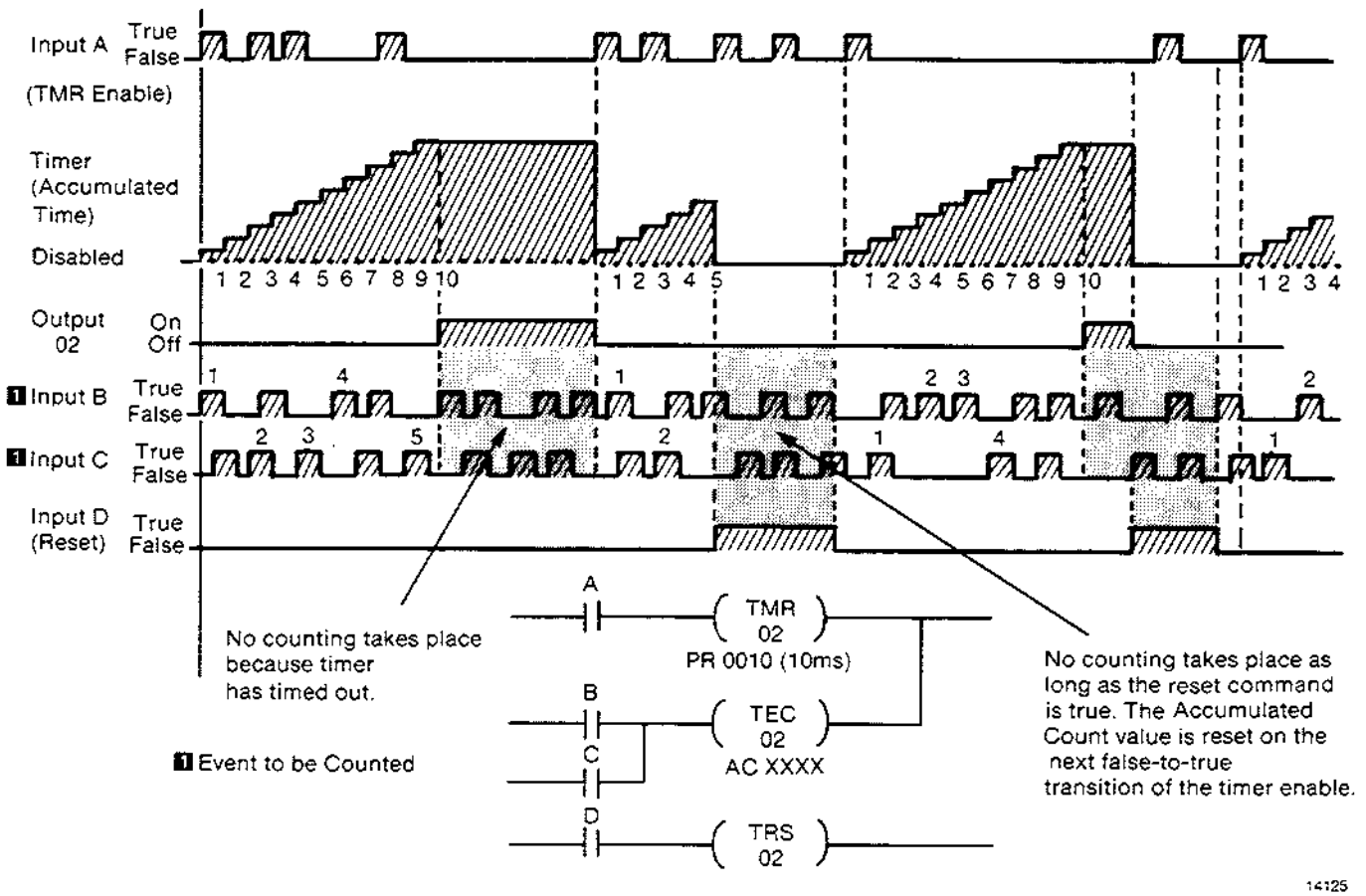
Figure 4.6
Functional diagram for a timer with reset



Timed event counter instruction (0016)

The timed event counter (TEC) instruction (figure 4.7) counts events or pulses over a selected time interval while the timer is timing. This instruction is enabled when the input branch conditions make a false-to-true transition of the timer portion of the instruction. No counting takes place during the periods the timer is not timing. The timer will stop timing when it is reset by the timer reset (TRS) instruction. The accumulated time is reset at this time. The timer is re-enabled on the next false-to-true transition of its input condition as long as the reset condition is no longer true. When the accumulated time equals the preset time (1 ms to 10 seconds), the timer stops incrementing its accumulated time and energizes the output. If a false-to-true transition of the input occurs after the output is energized, the output de-energizes and the accumulated time resets to zero and begins timing again.

Figure 4.7
Functional diagram for a timed event counter with reset



Timer reset instruction (0018)

The timer reset instruction (TRS) is an output instruction that resets the timer accumulated time to zero and de-energizes the output. In order for the timer to be reset, the TRS instruction must have the same output designation as the timer.

Block-transfer-read data

Refer to figure 4.8 for the block-transfer-read data format.

Figure 4.8
Format of block-transfer-read data

Bit / Word	17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
1	Power Up	Data Valid	Prog. Error	H/W Fault		Fuse Blown		Boolean Mode								
2	Hardware Inputs 7	6	5	4	3	2	1	0					Hardware Outputs 3	2	1	0
3	Composite Inputs								Logical Outputs							
4	Binary value specified with SETID Command															
5	BCD				Count				Output				0			
6	BCD				Count				Output				1			
7	BCD				Count				Output				2			
8	BCD				Count				Output				3			
9	Error Word Pointer								Error Code							
10	Communication Processor Series/Rev.								Control Processor Series/Rev.							

Word 1 - Status flags

The first status word contains the status flags in the upper byte and zeroes in the lower byte.

Word	Bit	Description
1	10	When set to one, the Boolean Mode bit indicates that the programming format is in terms of ladder logic expressions.
1	12	When set to one, the fuse blown bit indicates that one or more of the 3A rectifier fuses protecting the output circuitry has blown.
1	14	When set to one, the hardware fault bit indicates that the controller has a hardware problem.
1	15	When set to one, the program error bit indicates that a program error in the hex file transferred to the controller has occurred. This bit remains set until an INIT command is received or power to the rack is recycled.
1	16	When set to one, the data valid bit indicates that the data transferred to the controller using block-transfer is valid (no program errors).
1	17	When set to one, it indicates that the controller has passed its power up diagnostics routine and is ready to receive a program.

Word 2 - Hardware outputs and inputs

Word 2, bits 00 through 03 represent the state of outputs 0 through 3. When set to one, the bit indicates that the output at the corresponding terminal on the 1771-WG wiring arm is supplying voltage to the device.

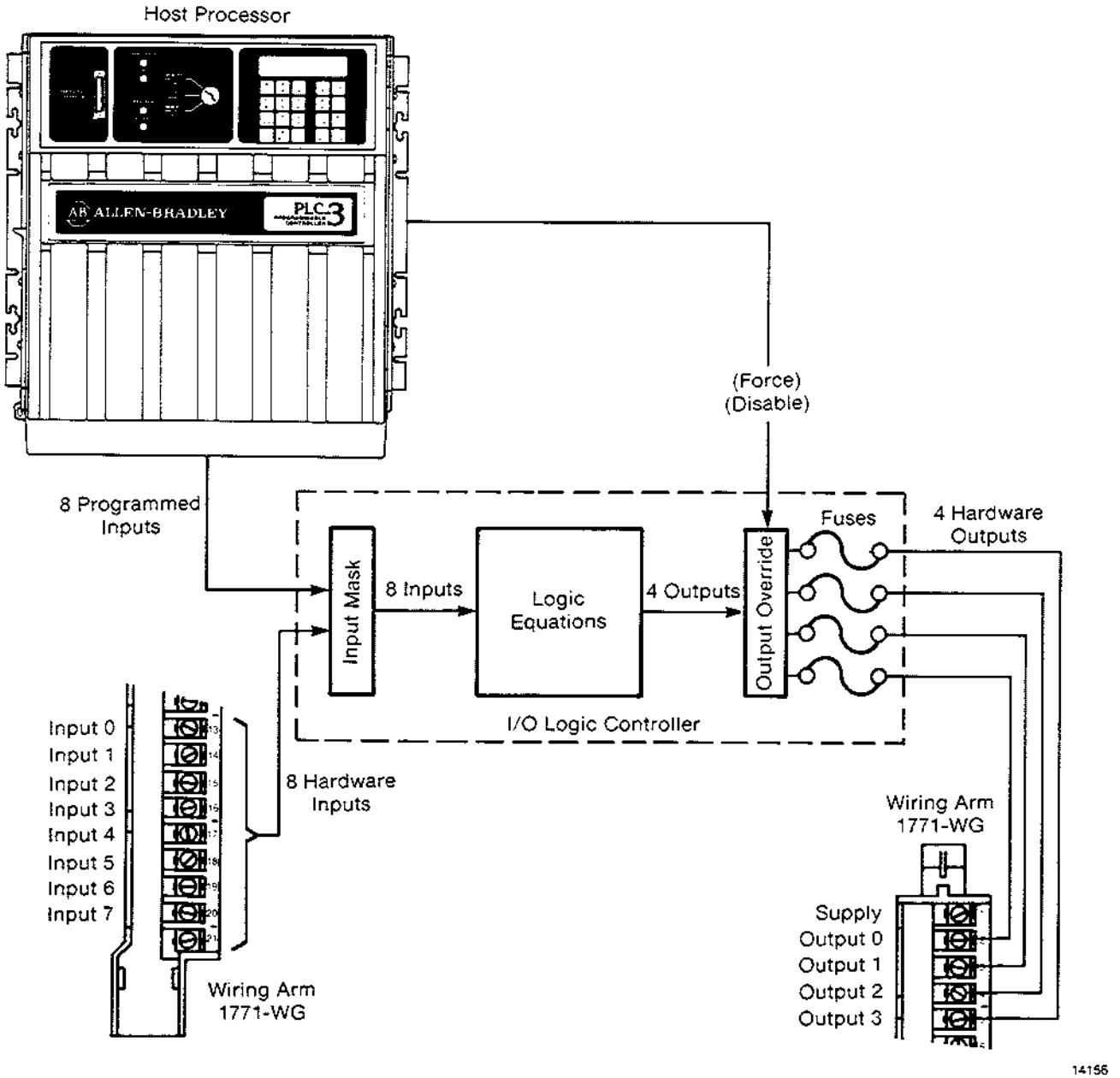
When reset to zero, the bit indicates that the output at the corresponding terminal on the 1771-WG wiring arm is not supplying voltage to the device. Refer to figure 4.9 - I/O Block diagram.

Word 2, bits 10 through 17 represent the state of inputs 0 through 7. When set to one, the bit indicates that the input is true (10 to 24 V at the corresponding terminal on the 1771-WG wiring arm). When reset to zero, the bit indicates that the input is false (0 to 4 V at the corresponding terminal on the 1771-WG wiring arm).

Word 3 - Logical outputs and composite inputs

Word 3, bits 00 through 03 represent the logical state of outputs 0 through 3 without being conditioned by the Disable output command and/or the Force output command. Refer to Figure 4.9 - I/O Block diagram.

Figure 4.9
I/O block diagram



Word 3 - Logical outputs and composite inputs (continued)

When set to one, the bit indicates that the logic equation of the output is true. When reset to zero, the bit indicates that the logic equation of the output is false.

Word 3, bits 10 through 17 represent the logical state of inputs used in the logic equations.

When set to one, the bit indicates that the input will be used in the logic equations as high true logic (on). When reset to zero, the bit indicates that the input will be used in the logic equations as low true logic (off).

Word 4 - Binary value specified with SETID command

Use SETID (0004) when you need more than 64 words for your program. The address is returned in the fourth status word and is used to link multiple block-transfers if more than 64 words are required. Also use this command as a block identifier in the configuration file. This helps you to identify which program within the file you are currently using.

Words 5, 6, 7 and 8 - Binary coded decimal (BCD) counts

Use words 5 through 8 for BCD counts when outputs 0, 1, 2 and 3 are defined as up counter (UCT), down counter (DCT) or timed event counter (TEC) instructions. The value of the accumulated count (max 9999) is returned to the processor.

- Word 5 = BCD count for output 0
- Word 6 = BCD count for output 1
- Word 7 = BCD count for output 2
- Word 8 = BCD count for output 3

Word 9 - Error code and error word pointer

Word 9, bits 10 through 17 (error word pointer) points to the word in the hex file where a programming error has occurred.

Word 9, bits 00 through 07 (error code) specifies what type of programming error has occurred. Refer to the following section titled "Programming errors".

Word 10 - Control processor series/revision and Communication processor series/revision

Word 10, bits 00 through 07 specifies the current series/revision of the control processor on the controller module.

Control Processor		
Series	Revision	Hex code
A	A	21
A	B	22
A	C	23
A	D	24
B	A	41
B	B	42
B	C	43
B	D	44

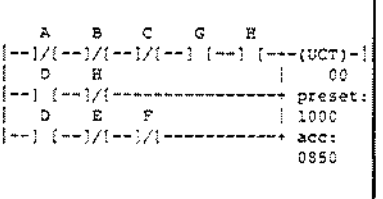
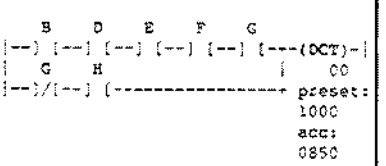
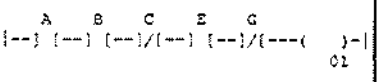
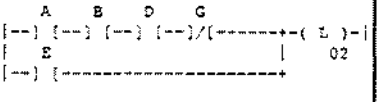
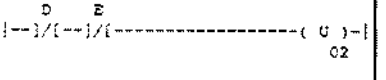
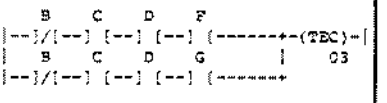
Word 10, bits 10 through 17 specifies the current series/revision of the communication processor on the controller module.

Communication Processor		
Series	Revision	Hex code
A	A	21
A	B	22
A	C	23
A	D	24
B	A	41
B	B	42
B	C	43
B	D	44

The value 2121 in status word 10 indicates series A and revision A for both the control and communication processors.

Programming example

The following programming example shows the various output commands, hex codes and associated ladder logic:

Parameter	Hex code	Equivalent ladder logic
Initialize	0001	
UCT	0013	
Output 0	FF00	
Preset	1000	
Accumulated	0850	
Input branch	C707	
Input branch	8880	
Input branch	3830	
DCT	0014	
Output 0	FF00	
Preset	1000	
Accumulated	0850	
Input branch	7A00	
Input branch	C040	
RCT	0015	
Output 0	FF00	
Input branch	0100	
OTE	0010	
Output 1	FF01	
Input branch	5744	
OTL	0011	
Output 2	FF02	
Input branch	4B40	
Input branch	1000	
OTU	0012	
Output 2	FF02	
Input branch	1818	
TMR	0017	
Output 3	FF03	
Time	9125	
Input branch	8181	
TEC	0016	
Output 3	FF03	
Input branch	2E02	
Input branch	4E02	
TRS	0018	
Output 3	FF03	
Input branch	8000	
Set input filters	0030	
Inputs 0 and 1	0000	default value of 500 us
Inputs 2 and 3	9999	9.999 milliseconds
Inputs 4 and 5	0850	850 microseconds
Inputs 6 and 7	2500	2.5 milliseconds
Execute	0003	
Total words in hex file		44 words
Maximum allowable file length		64 words

Programming errors

Word 1, bit 15 (program error status bit) of the read block transfer file is set if any program errors exist in the hex file that is transferred to the controller.

Word 9, bits 00 through 07 inform you of the type of problem. Bits 10 through 17 point to the incorrect word in the hex file.

If an error occurs during programming, the Execute Command (0003) is not processed and all outputs are de-energized. Additional commands are not accepted until an Initialized Command (0001) is processed. If a program is executing when a command generates an error, the program will halt execution and all outputs are de-energized.

Programming boundaries

You can assign the same output address to any one of the following groups of similar instructions without having a programming error occur (Table 4-1). For example, you can assign the same output address to each of three rungs whose outputs are an up counter (UCT), down counter (DCT) and reset counter (RCT) and another output address to each of two rungs whose outputs are latch (L) and unlatch (U) without causing a programming error. However, if you define an output as an up counter (UCT) and as a latch (L) a programming error occurs. The module sets an error bit and stops operating when it detects a programming error.

You can use any one of four output addresses, 00-03, to define a rung. The maximum number of rungs of equivalent ladder logic can vary from four (using only output energize instructions) to twelve (using only counter instructions). You can assign the same output address to any of the groups in table 4-1, only once.

Table 4-1
Programming boundaries

-( )-	-(L)- -(U)-	-(UCT)- -(DCT)- -(RCT)-	-(TMR)- -(TRS)-	-(TMR)-(TEC)- -(TRS)-

Following is a list of possible errors and their associated error codes which are displayed in the lower byte of the ninth status word (refer to figure 4.8):

Error code	Error
01	You assigned an invalid output command. A valid command has an upper byte of zeroes. The lower byte must be a valid BCD number which is an output command.
02	Program error. Command must fit within block-transfer file length.
03	You entered an output again as the same function or as a different function outside of the programming boundaries. Refer to Table 4-1,"Programming boundaries", above.
04	You entered a command other than an on-line command after the Execute Command (0003), to program an output.
05	Invalid command
07	Incorrect upper byte for this command.
08	You assigned an invalid output address.
09	Invalid command
10	This word contains an invalid BCD digit.
11	You must assign the same preset and accumulated values to UCT and DCT instructions having the same address.
12	You assigned an invalid filter time constant.
13	Invalid command
14	Invalid command
15	Invalid command
16	Invalid command
17	Invalid command
18	Invalid command

Commands

There are two types of commands in your controller instruction set.

- Configuration data commands
- On-line commands

Commands (continued)

Use the configuration data commands to initially configure or program the controller. You can use all the controller's commands as configuration data commands. Use the on-line commands to change your program after it has been block-transferred to the controller. You can do this without causing any program errors or impeding I/O processing. Following is a list of the configuration data and on-line commands and their hex codes.

Configuration data commands	
NOP	(0000)
INIT	(0001)
HALT	(0002)
EXEC	(0003)
SETID	(0004)
END	(0005)
OTE	(0010)
OTL	(0011)
OTU	(0012)
UCT	(0013)
DCT	(0014)
RCT	(0015)
TEC	(0016)
TMR	(0017)
TRS	(0018)
SETIF	(0030)
DISABLE OUTPUT	(0031)
FORCE OUTPUT	(0032)
MASK INPUT	(0033)

On-line commands	
NOP	(0000)
HALT	(0002)
EXEC	(0003)
SETID	(0004)
END	(0005)
DISABLE OUTPUT	(0031)
FORCE OUTPUT	(0032)
MASK INPUT	(0033)

Configuration data commands

The following configuration data commands are described under "Output instructions" earlier in this section:

OTE	(0010)	RCT	(0015)
OTL	(0011)	TEC	(0016)
OTU	(0012)	TMR	(0017)
UCT	(0013)	TRS	(0018)
DCT	(0014)		

Configuration data commands (continued)

Following is a list of the remaining configuration data commands:

NOP (0000) - Reserves one word in the hex file for adding an instruction to the file at a later date without changing the values in the rest of the file. This command can also edit or eliminate current logic that may no longer be required.

INIT (0001) - Clears out the present controller program and sets all parameters to the default state. It also stops all I/O processing and de-energizes all outputs.

HALT (0002) - Use this command to immediately stop all I/O processing. Outputs remain in their last state regardless of the input conditions. An Execute Command (0003) must be sent to the module to resume I/O processing.

EXEC (0003) - Normally follows reconfiguration data and causes the module to begin execution of the most recent program downloaded from the processor. It also stops processing of any remaining data in the block-transfer data block.

SETID (0004) - This command when processed by the controller returns a BCD or binary value in the fourth status word of the block-transfer-read. You need additional ladder logic to pass this value from the fourth status word to the block-transfer-write file address. You can also use this command as a block identifier in the configuration file to help you identify which program is in current use when you are using multiple programs. Also use this command to link multiple block-transfers together if more than 64 words are required to define a program. The SETID command uses the following programming format:

0004	SETID Command
XXXX	Address value

END (0005) - Module executes the program up to but not beyond the END instruction. You can use this command to debug parts of your program without executing the complete program. All words following this command are not processed by the module including the Execute Command (0003).

SETIF (0030) - Provides the means to change the input filter constant for a given input pair. Four data words follow this command to allow you to program individual filter constants (one word for each input pair).

Configuration data commands (continued)

Disable Outputs (0031) - You can use this command during installation to verify operation of the module without requiring connections to output devices. The output of the logic equations is verified in the status block (word 3). Use the mask input command with the disable output command to provide inputs to the logic equations. This means you do not need an actual hardwired input device. The disable output command disables selected outputs and controls the outputs regardless of the state of the input conditions. In the default condition all outputs are enabled. The disable output command uses the following programming format:

0031	Disable Output Command
FF0X	Where X is the hex code for the outputs that are disabled

Hex	Disabled Outputs
F	None
E	0
D	1
C	0,1
B	2
A	0,2
9	1,2
8	0,1,2
7	3
6	0,3
5	1,3
4	0,1,3
3	2,3
2	0,2,3
1	1,2,3
0	0,1,2,3

Force Output (0032) - Use the force output command during initial installation to verify proper operation of an output device. You can use this command at any time (before a valid configuration file) as long as no program errors exist. The force output command forces the selected outputs on or off overriding any other status. It uses the following programming format:

0032	Force Output Command
0X0X	Where X in the upper byte selects the output or combination of outputs to be forced.
	Where X in the lower byte selects whether the output or combination of outputs are forced ON or OFF.
	Use 0000 to unforce all commands.

Configuration data commands *(continued)*

Upper Byte (Outputs to be forced)		Lower Byte (ON/OFF)		
Hex Code	Outputs	Hex Code	Outputs ON	Outputs OFF
0	None	0	None	0,1,2,3
1	0	1	0	1,2,3
2	1	2	1	0,2,3
3	0,1	3	0,1	2,3
4	2	4	2	0,1,3
5	0,2	5	0,2	1,3
6	1,2	6	1,2	0,3
7	0,1,2	7	0,1,2	3
8	3	8	3	0,1,2
9	0,3	9	0,3	1,2
A	1,3	A	1,3	0,2
B	0,1,3	B	0,1,3	2
C	2,3	C	2,3	0,1
D	0,2,3	D	0,2,3	1
E	1,2,3	E	1,2,3	0
F	0,1,2,3	F	0,1,2,3	None

Mask Inputs (0033) - Masks selected inputs from the wiring arm so the processor can substitute programmed bits. Use the Mask Input Command when an input from the processor is required for an enable or interlock. The Initialize Command (0001) sets the default to have all inputs coming from the wiring arm. Once an input is programmed to come from the processor data table, it remains that way until a new Initialize Command is entered or a new Mask Input Command is entered.

Use of the Mask Input Command changes the overall system response time due to processor scan time, block-transfer time etc. Refer to Appendix A for examples of block-transfer timing. The Mask Input Command uses the following programming format:

0033 Mask Inputs Command

XXYY XX in the upper byte selects which inputs are coming from the processor data table. Refer to Appendix B - Hex Input Conversion Table for the appropriate hex code.

YY in the lower byte selects the logical value of the inputs used in the logic equations.

Configuration data commands *(continued)*

Example:

Inputs from processor (upper byte)									Logical value (lower byte)								
Bit	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**Bit**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
Input Mask	7 0	6 0	5 0	4 1	3 0	2 0	1 0	0 1	Input Logic	7 0	6 0	5 0	4 0	3 0	2 0	1 0	0 1
Hex Equi-valent		1				1					0				1		

A (1) in each Mask field selects that input to originate from the host processor.	A (1) in each Logic field indicates that the input from the host processor is true (ON).
A (0) in each Mask field selects that input to originate from the wiring arm.	A (0) in each Logic field indicates that the input from the host processor is false (OFF).

In our example above inputs 0 and 4 (underlined) come from the processor. Input 0 is true (on) and input 4 is false (off).

On-line commands

The following configuration data commands are also on-line commands. Refer to the descriptions above found under "Configuration data commands".

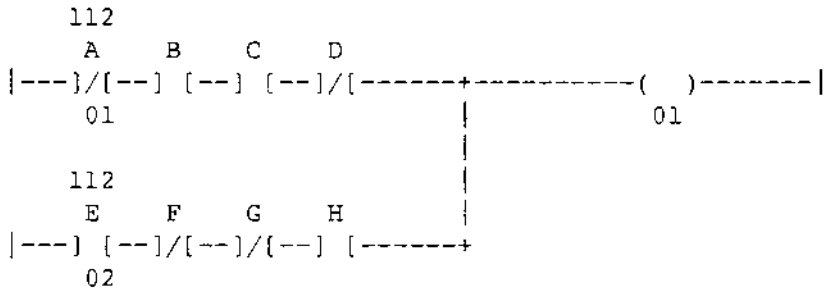
NOP	END
HALT	DISABLE OUTPUT
EXEC	FORCE OUTPUT
SETID	MASK INPUT

Mask input programming examples for PLC-2 and PLC-3 family processors

The following examples show how to use the mask input feature and the associated ladder logic.

This mask input example uses two data files. The first data file contains the configuration data to program the controller. The second data file contains the mask input command and the end of program command.

PLC-2 family processors



Controller logic

Inputs A (0) and E (4) come from inputs connected to the host processor located in rack 1, module group 2, terminal 1 and rack 1, module group 2 and terminal 2, respectively.

Parameter	Hex Code	Octal Address in PLC-2 processor	
Initialize	0001	224	Config-uration file
OTE	0010	225	
Output 1	FF01	226	
Input branch	0F09	227	
Input branch	F060	230	
Mask Input	0033	231	
Inputs from processor	1101	232	
Execute	0003	233	
Mask Input	0033	400	On-line data file
Inputs from processor	1110	401	
End of program	0005	402	

PLC-2 family processors *(continued)*

The following support logic in a PLC-2 processor will perform the mask input example.

```
LADDER DIAGRAM DUMP
                                             START
        ! 150                                                                        131  !
Rung  1 +-[G]----------------------------------------------------------------------(PUT)-+
        ! 400                                                                        224  !
        !                                                                                 !
        ! Data  Power-                                                                    !
        ! valid  up                                                                       !
        !  bit   bit                                                                      !
        ! 212    212    151                                                               !
Rung  2 +-]/[---] [-+-[G]-------------------------------------------------------------(PUT)-+
        !   16    17! 224                                                            224  !
        !Pushbutton !                                                                     !
        !  to load  !                                                                     !
        !new program!                                                                     !
        ! 112       !                                                                     !
        +-] [-------+                                                                     !
        !   00                                                                            !
        ! Read  Write                                                                     !
        ! done  enable                                                                    !
        ! 110    010                                           +------------------+  010  !
Rung  3 +-]/[---]/[--------------------------------------------+ BLOCK XFER READ  +--(EN)--+
        !   07     06                                          !DATA ADDR:     030!    07 !
        !                                                      !MODULE ADDR:   100!       !
        ! Read                                                 !BLOCK LENGTH:   10!  110  !
        ! done                                                 !FILE:    200-  211+--(DN)--+
        ! 110    Buffer file                                   +------------------+    07 !
        !                                                      +------------------+  034  !
Rung  4 +-] [--------------------------------------------------+ FILE TO FILE MOVE+--(EN)--+
        !   07                                                 !COUNTER ADDR:  034!    17 !
        !                                                      !POSITION:      001!       !
        !                                                      !FILE LENGTH:   010!  034  !
        !                                                      !FILE A:  200-  211+--(DN)--!
        !                                                      !FILE R:  212-  223!    15 !
        !Write  Read                                           !RATE PER SCAN: 010!       !
        ! done  enable                                         +------------------+       !
        ! 110    010                                           +------------------+  010  !
Rung  5 +-]/[---]/[--------------------------------------------+ BLOCK XFER WRITE +--(EN)--+
        !   06     07                                          !DATA ADDR:     031!    06 !
        !                                                      !MODULE ADDR:   100!       !
        !                                                      !BLOCK LENGTH:   08!  110  !
        !                                                      !FILE:    224-  233+--(DN)--+
        !                                                      +------------------+    06 !
        ! 112                                                                        401  !
Rung  6 +-] [-------------------------------------------------------------------------( )--+
        !   01                                                                        00  !
        ! 112                                                                        401  !
Rung  7 +-]/[-------------------------------------------------------------------------( )--+
        !   02                                                                        04  !
```

Rung descriptions for mask input PLC-2 family processor example program

Rung 1 - Unconditionally loads the mask input data file starting address into the block-transfer-write (BTW) instruction. Length of the file must not exceed the BTW length.

Rung 2 - Configuration file starting address is loaded into the BTW instruction conditioned by the power-up bit (212/17) being true and the data-valid bit (212/16) being false or a pushbutton being on.

Rung 3 - Both the read done bit (110/07) and write enable bit (010/06) being false are used to condition the block-transfer-read (BTR) so that different block lengths for reads and writes can be used.

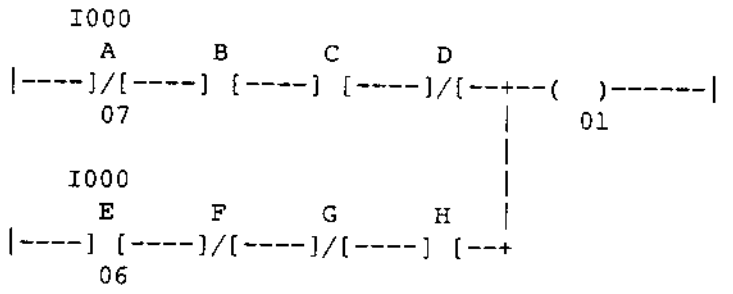
Rung 4 - Use a file-to-file move to buffer the read data when making any data comparisons.

Rung 5 - Both the write done bit (110/06) and read enable bit (010/07) being false are used to condition the BTW so that different block lengths for reads and writes can be used.

Rung 6 - Used for Mask Input Command. Input (112/01) is used to condition bit 401/00 (Input 0) which is passed from the host processor.

Rung 7 - Used for Mask Input Command. Input (112/02) is used to condition bit 401/04 (Input 4) which is passed from the host processor.

PLC-3 family processors



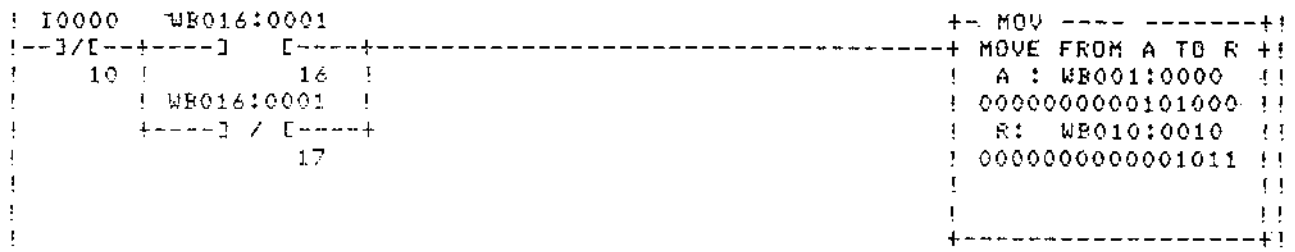
Controller logic

PLC-3 family processors (continued)

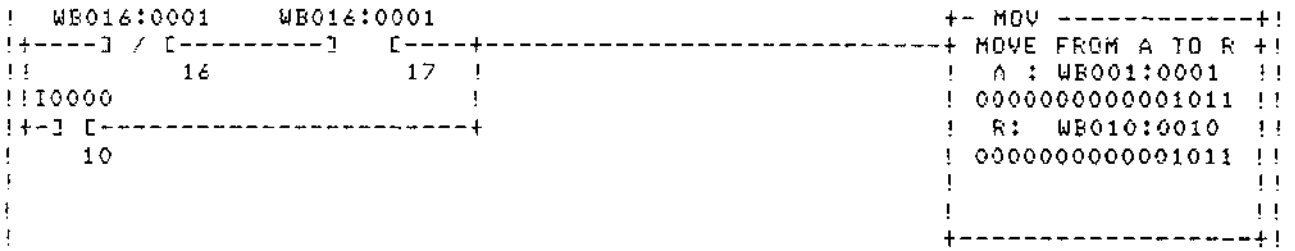
Inputs A (0) and E (4) come from the host PLC-3 processor rack 0, module group 0, terminal 7 and rack 0, module group 0 and terminal 6, respectively.

Parameter	Hex Code	Address in PLC-3	
Initialize	0001	B15:11	Config-uration file
OTE	0010	B15:12	
Output 1	FF01	B15:13	
Input branch	0F09	B15:14	
Input branch	F060	B15:15	
Mask input	0033	B15:16	
Inputs from PLC-3	1101	B15:17	
Execute	0003	B15:18	
Mask input	0033	B15:40	On-line data file
Inputs from PLC-3	1110	B15:41	
End of program	0005	B15:42	

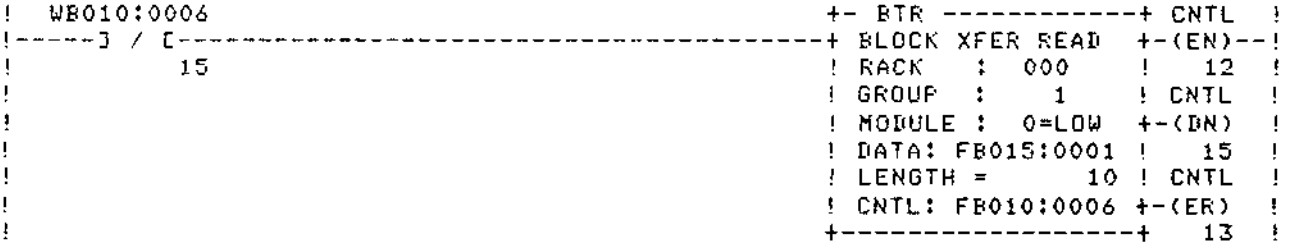
RUNG NUMBER RM0



RUNG NUMBER RM1



RUNG NUMBER RM2



PLC-3 family processors
(continued)

```
                                RUNG NUMBER RM3

!  WB010:0006                                          +- MVF ------------+ C0001 !
!------]   [-----------------------------------------+ FILES FROM A TO R+-(EN)--!
!           15                                       !                  !   12  !
!                                                    !  A : FB015:0001  ! C0001 !
!                                                    !  R : FB016:0001  +-(DN)  !
!                                                    !  COUNTER  : C0001!   15  !
!                                                    !POS/LEN =   0/  10! C0001 !
!                                                    !   MODE =  10/SCAN+-(ER)  !
!                                                    +------------------+   13  !

                                RUNG NUMBER RM4

!  WB010:0006                                          +- BTW ------------+ CNTL  !
!------] / [-----------------------------------------+ BLOCK XFER WRITE +-(EN)--!
!           05                                       ! RACK   :  000     !   02  !
!                                                    ! GROUP  :    1     ! CNTL  !
!                                                    ! MODULE :  0=LOW   +-(DN)  !
!                                                    ! DATA: FB015:0011  !   05  !
!                                                    ! LENGTH =        8 ! CNTL  !
!                                                    ! CNTL: FB010:0006 +-(ER)  !
!                                                    +------------------+   03  !

                                RUNG NUMBER RM5

! I0000                                                                WB015:0041  !
!---] [-------------------------------------------------------------------(    )----!
!    07                                                                    00  !

                                RUNG NUMBER RM6

! I0000                                                                WB015:0041  !
!--]/[--------------------------------------------------------------------(    )----!
!    06                                                                    04  !

                                RUNG NUMBER RM7

!                                                                                  !
!-----------------------------------------------------------------------------(EOP) !
!                                                                                  !

                START=WB015:0000
  WORD #          0       1       2       3       4       5       6       7
   00000       0000    0000    0000    0000    0000    0000    0000    0000
   00008       0000    0000    0000    0001    0010    FF01    0F09    F060
   00016       0033    1101    0003    0000    0000    0000    0000    0000
   00024       0000    0000    0000    0000    0000    000D    0000    0000
   00032       0000    0000    0000    0000    0000    000C    0000    0000
   00040       0033    1110    0005    0000    0000    0000    0000    0000
   00048       0000    0000    0000    0000    0000    0000    0000    0000
   00056       0000    0000    0000    0000    0000    0000    0000    0000
   00064       0000    0000    0000    0000    0000    0000    0000    0000
   00072       0000    0000    0000    0000    0000    0000    0000    0000
   00080       0000    0000    0000    0000    0000    0000    0000    0000
   00088       0000    0000    0000    0000    0000    0000    0000    0000
   00096       0000    0000    0000    0000    0000
```

Rung descriptions for PLC-3 family processor example program

Rung 0 - The on-line data file address (stored at B1:0 in binary format) is loaded into the block-transfer-write BTW instruction (at B10:10) if pushbutton I000/10 is off and either the power-up bit (B16:1/17) is off or the data valid bit (B16:1/16) is on.

Rung 1 - If the power-up bit (B16:1/17) is on and the data valid bit (B16:1/16) is off or the pushbutton I000/10 is on, the starting address of the configuration file (stored at B1:1 in binary format) is loaded into the BTW instruction (at B10:10).

Rung 2 - The read done bit (B10:6/15) is false and causes a block-transfer-read (BTR) to be enabled.

Rung 3 - BTR data is buffered at B16:1 to B16:11.

Rung 4 - The write done bit must be false to enable a BTW.

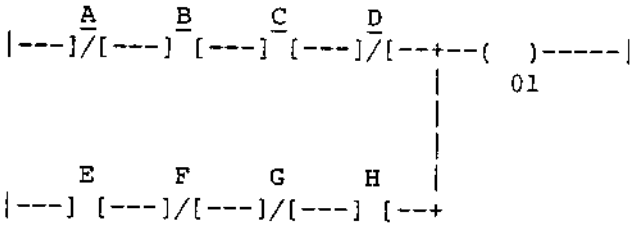
Rung 5 - Input I000/7 conditions B15:41/0, which controls input A through the mask input command.

Rung 6 - Input I000/6 conditions B15:41/4, which controls input E through the mask input command.

SETID programming examples

The following examples show how to use the SETID feature and the associated ladder logic.

PLC-2 family processors

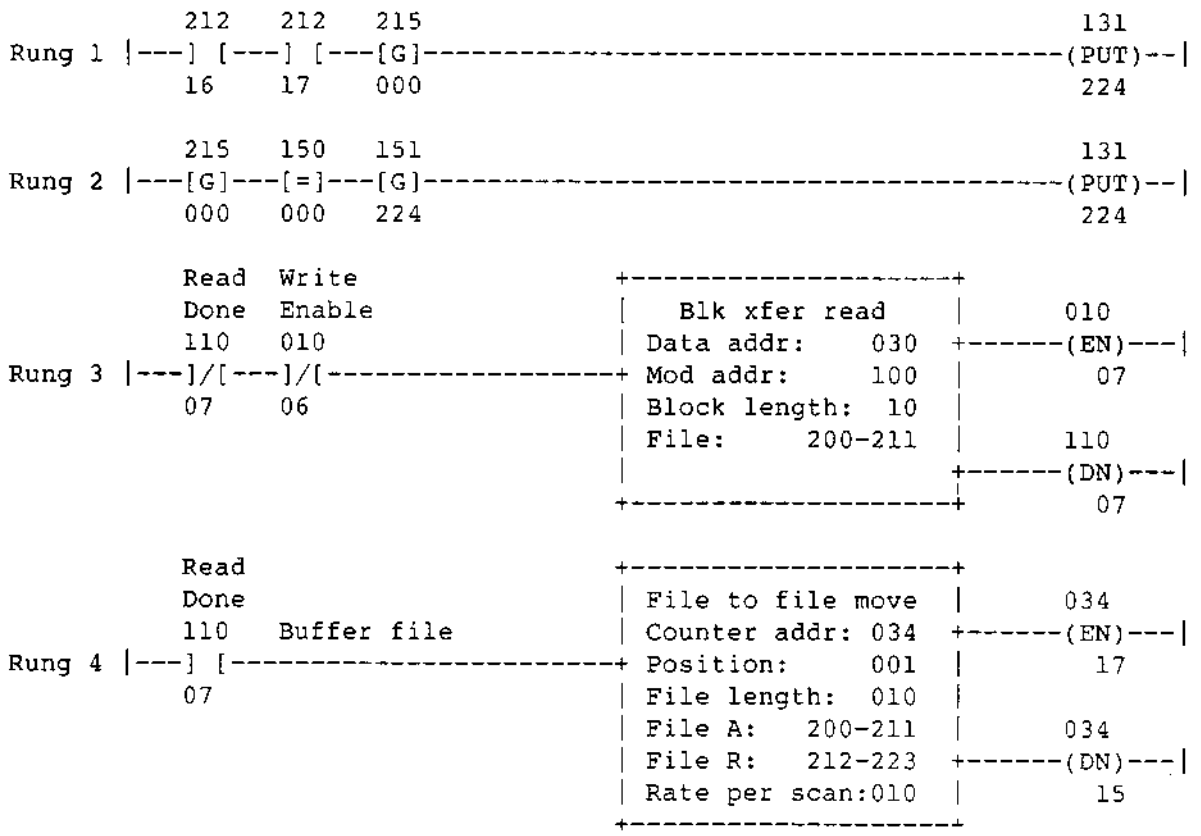


Controller logic

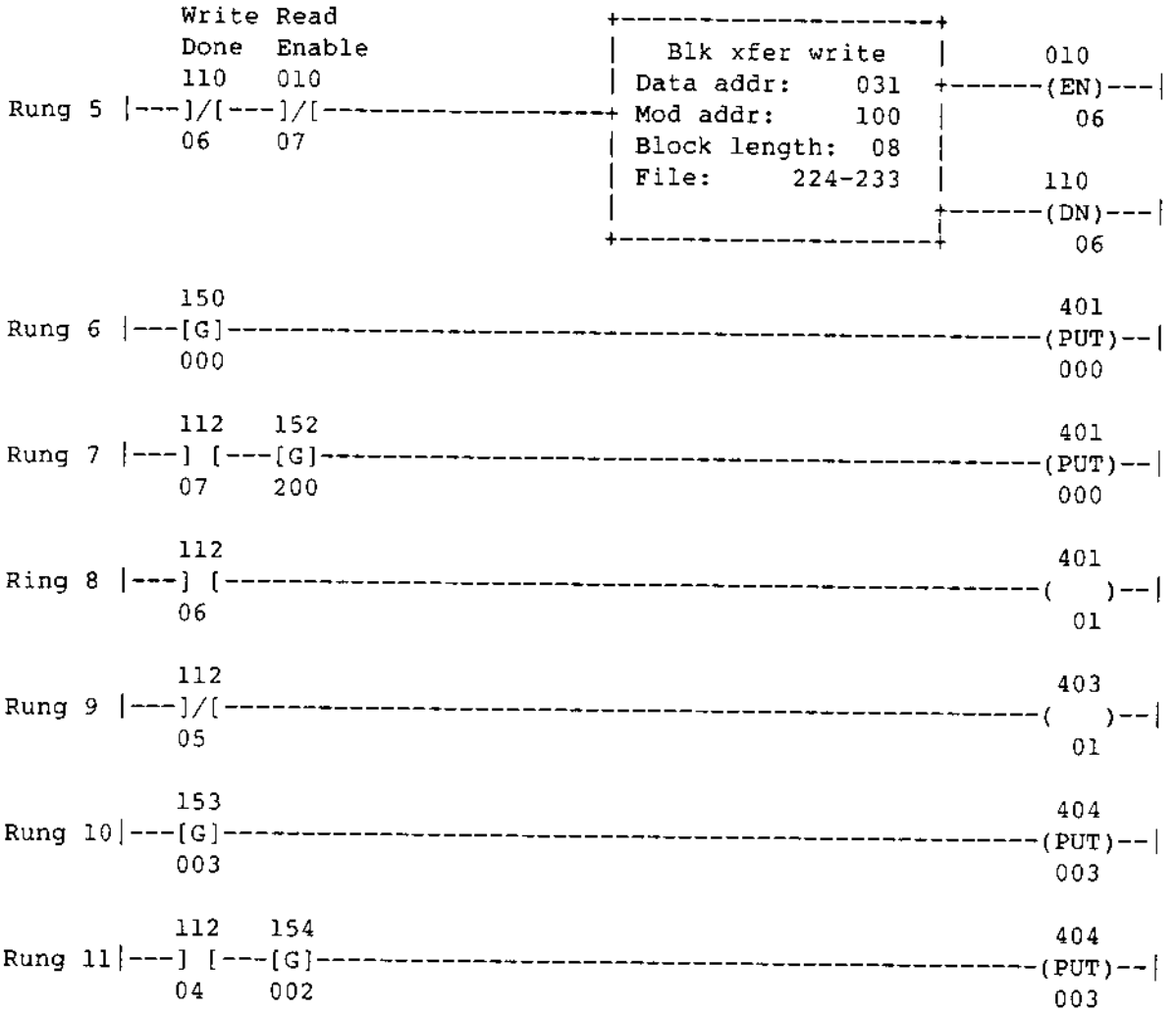
PLC-2 family processors *(continued)*

Parameter	Hex Code	Octal Address in PLC-2 processor	
Initialize	0001	224	Config-uration file
OTE	0010	225	
Output 1	FF01	226	
Input branch	0F09	227	
Input branch	F060	230	
SETID	0004	231	
Binary value	0400	232	
Execute	0003	233	
Force output	0032	400	On-line file
No outputs forced	0000	401	
Disable output	0031	402	
No outputs disabled	FF0F	403	
Halt/Exec	0003	404	
NOP	0000	405	
END	0005	406	

You must enter the following support logic in your PLC-2 processor to perform the SETID instruction.



PLC-2 family processors
(continued)



Rung descriptions for the PLC-2 family processor SETID program

Rung 1 - Loads the on-line data file starting address (stored at 215) into the block-transfer-write (BTW) instruction if the data valid bit (212/16) and power-up bit (212-17) are both on. The length of the second file must not exceed the length of the BTW.

Rung 2 - At power-up, the fourth status word (215) is zero. The configuration file located at (224) is loaded into the BTW instruction.

Rung 3 - The read done bit (110/07) and write enable bit (010/06) condition the block-transfer-read (BTR) so that different block lengths for reads and writes can be used.

Rung 4 - Use a file-to-file move to buffer the read data when making any data comparisons.

Rung 5 - The write done bit (110/06) and read enable bit (010/07) condition the BTR so that different block lengths for reads and writes can be used.

Rung 6 - Disables the force output command by loading zero into word 401.

Rung 7 - When PB1 (112/07) is true, this activates the force output command but output 1 is forced OFF by loading 0200 into word 401.

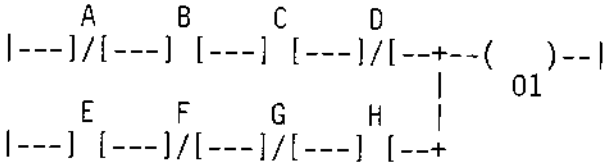
Rung 8 - When PB2 (112/06) and PB1 (112/07) are true, output 1 is forced on by the Force Output Command.

Rung 9 - When PB3 (112/05) is off, output 1 is disabled because the value FF0F (no outputs disabled) is loaded into word 403. When PB3 (112/05) is on, output 1 is enabled.

Rung 10- Unconditionally loads the execute command (0003) into word 404.

Rung 11- When PB4 (112/04) is on, the halt command is loaded into word 404 causing all outputs to freeze in their current state.

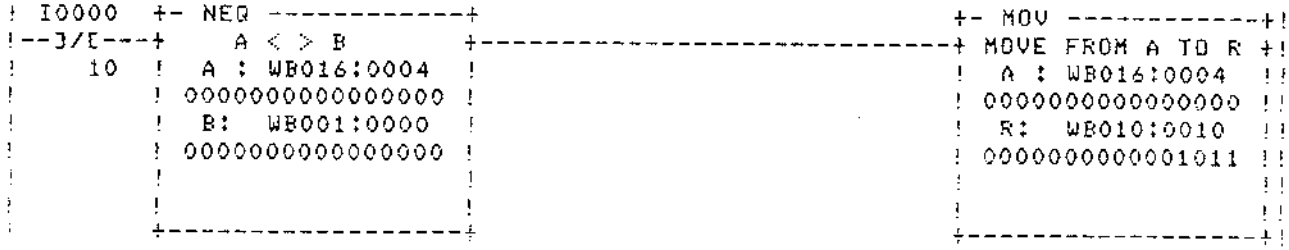
PLC-3 family processors



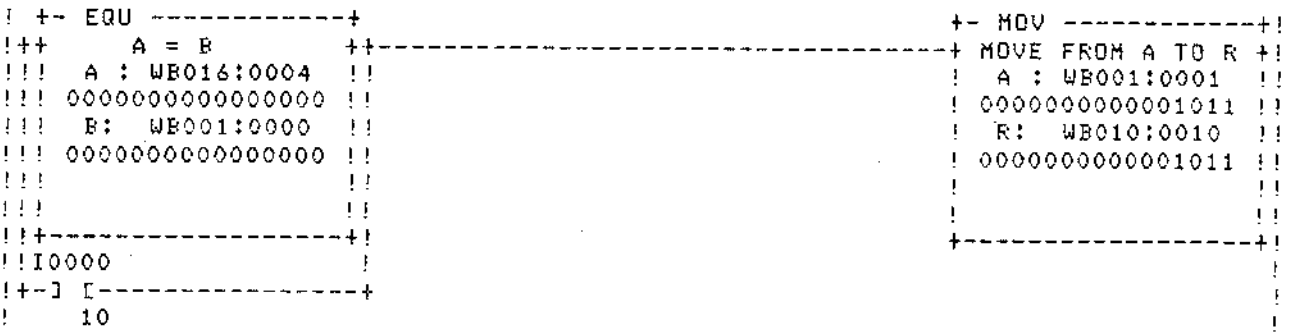
Controller logic

Parameter	Hex Code	Address in PLC-3	
Initialize	0001	B15:11	Configuration file
OTE	0010	B15:12	
Output 1	FF01	B15:13	
Input branch	0F09	B15:14	
Input branch	F060	B15:15	
SETID	0004	B15:16	
Binary value	0400	B15:17	
Execute	0003	B15:18	
Force output	0032	B15:40	On-line file
No outputs forced	0000	B15:41	
Disable output	0031	B15:42	
No outputs disabled	FF0F	B15:43	
Halt/Exec	0003	B15:44	
NOP	0000	B15:45	
END	0005	B15:46	

RUNG NUMBER RM0



RUNG NUMBER RM1



PLC-3 family processors
(continued)

RUNG NUMBER RM2

```
!  WB010:0006                                          +- BTR ------------+ CNTL  !
!-----] / [-----------------------------------------+ BLOCK XFER READ  +-(EN)--!
!           15                                       ! RACK    :  000     !   12  !
!                                                    ! GROUP   :    1     ! CNTL  !
!                                                    ! MODULE  :  0=LOW   +-(DN)  !
!                                                    ! DATA: FB015:0001  !   15  !
!                                                    ! LENGTH =       10 ! CNTL  !
!                                                    ! CNTL: FB010:0006 +-(ER)  !
!                                                    +------------------+   13  !
```

RUNG NUMBER RM3

```
!  WB010:0006                                          +- MVF ------------+ C0001 !
!-----]   [-----------------------------------------+ FILES FROM A TO R+-(EN)--!
!           15                                       !                  !   12  !
!                                                    !  A : FB015:0001  ! C0001 !
!                                                    !  R : FB016:0001  +-(DN)  !
!                                                    !  COUNTER  : C0001!   15  !
!                                                    !POS/LEN =   0/  10! C0001 !
!                                                    !   MODE =  10/SCAN+-(ER)  !
!                                                    +------------------+   13  !
```

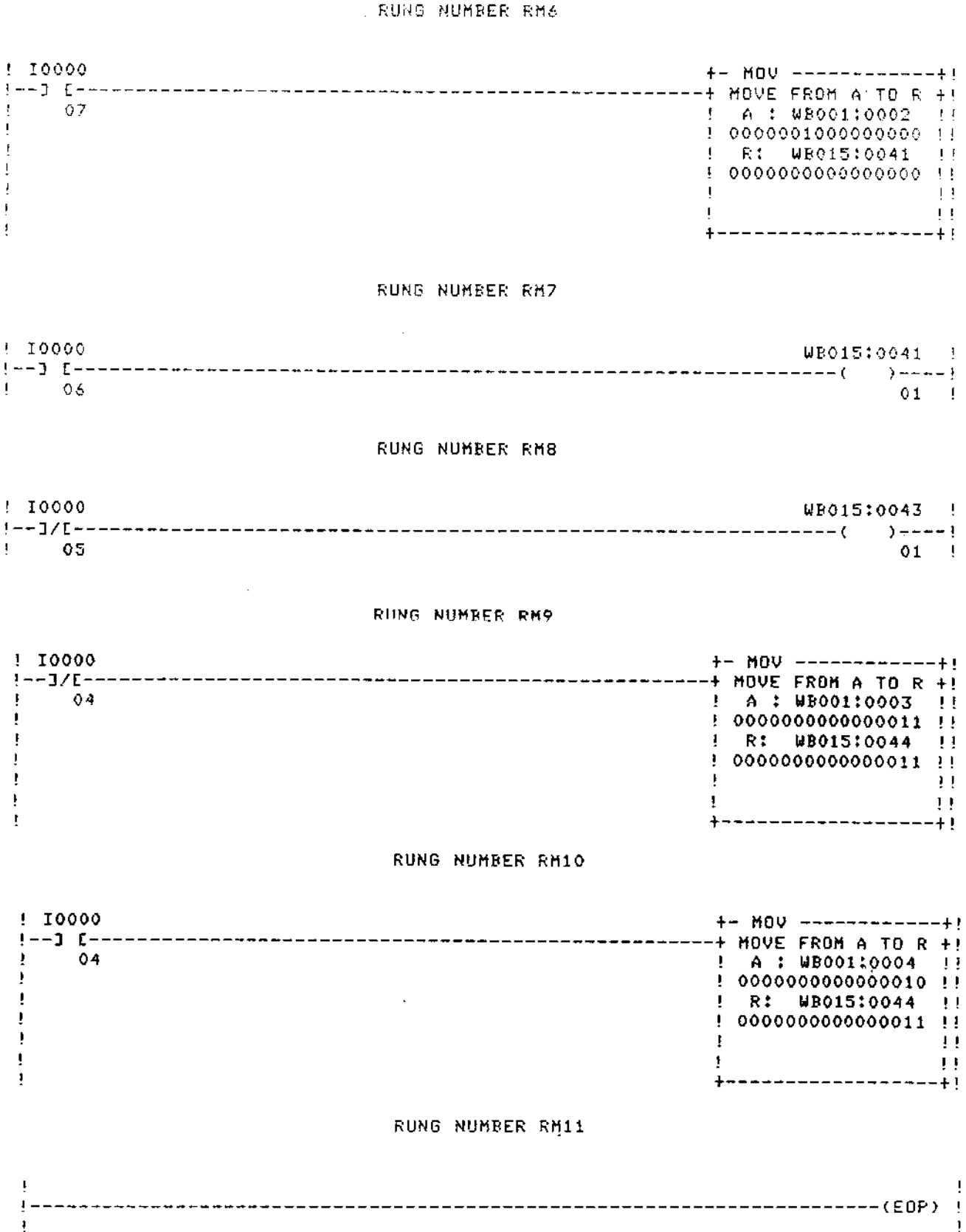
RUNG NUMBER RM4

```
!  WB010:0006                                          +- BTW ------------+ CNTL  !
!-----] / [-----------------------------------------+ BLOCK XFER WRITE +-(EN)--!
!           05                                       ! RACK    :  000     !   02  !
!                                                    ! GROUP   :    1     ! CNTL  !
!                                                    ! MODULE  :  0=LOW   +-(DN)  !
!                                                    ! DATA: FB015:0011  !   05  !
!                                                    ! LENGTH =        8 ! CNTL  !
!                                                    ! CNTL: FB010:0006 +-(ER)  !
!                                                    +------------------+   03  !
```

RUNG NUMBER RM5

```
! I0000                                                    +- MOV ------------+!
!--]/[-----------------------------------------------------+ MOVE FROM A TO R +!
!    07                                                    !  A : WB001:0000  !!
!                                                          ! 0000000000000000 !!
!                                                          !  R:  WB015:0041  !!
!                                                          ! 0000000000000000 !!
!                                                          !                  !!
!                                                          !                  !!
!                                                          +------------------+!
```

PLC-3 family processors
(continued)



PLC-3 family processors
(continued)

START=W9015:0000

WORD #	0	1	2	3	4	5	6	7
00000	0000	0000	0000	0000	0000	0000	0000	0000
00008	0000	0000	0000	0001	0010	FF01	0F09	F060
00016	0004	0028	0003	0000	0000	0000	0000	0000
00024	0000	0000	0000	0000	0000	0000	0000	0000
00032	0000	0000	0000	0000	0000	0000	0000	0000
00040	0032	0000	0031	FF0F	0003	0000	0005	0000
00048	0000	0000	0000	0000	0000	0000	0000	0000
00056	0000	0000	0000	0000	0000	0000	0000	0000
00064	0000	0000	0000	0000	0000	0000	0000	0000
00072	0000	0000	0000	0000	0000	0000	0000	0000
00080	0000	0000	0000	0000	0000	0000	0000	0000
00088	0000	0000	0000	0000	0000	0000	0000	0000
00096	0000	0000	0000	0000	0000			

Rung descriptions for PLC-3 family SETID programming example

Rung 0 - The on-line data file starting address (stored at B16:4 in binary format) is loaded into the block-transfer-write (BTW) instruction if the address (B16:4) is not zero and the configuration file select pushbutton (I000/10) is not on. The length of this file must not exceed the length of the BTW or the BTW length must be changed.

Rung 1 - At power-up, the fourth status word (B16:4) is zero. The starting address of the configuration file is loaded into the BTW. If pushbutton I000/10 is on, the configuration file address will also be loaded. Block-transfer addresses are stored in binary.

Rung 2 - The BTR done bit (B10:6/15) off is the only BTR condition. This allows the necessary false-to-true transition of PLC-3 block transfers.

Rung 3 - When the BTR is done, the status words are stored in buffer file (FB16:1).

Rung 4 - The BTW done bit (B10:6/5) off is the only BTW condition. This causes a false-to-true transition of the rung.

Rung 5 - If I000/07 is off, zeroes are loaded into B15:41 which contains the force output data. This disables all forces.

Rung descriptions for PLC-3 family SETID programming example (continued)

Rung 6 - If I000/7 is on, the force output command causes output 1 to be forced off by loading 0200 into B15:41.

Rung 7 - If I000/6 is on and I000/7 is on, output 1 is forced on by energizing B15:41.

Rung 8 - If I000/5 is off, B15:43/1 is on and all outputs are disabled because FF0F is stored in B15:43. If I000/5 is on, B15:43/1 is off. Output 1 is enabled by FF0D stored in B15:43.

Rung 9 - If I000/4 is off, the execute command (0003) is stored in the on-line data file in word B15:44.

Rung 10- If I000/4 is on, the halt command (0004) is loaded into the on-line data file in word B15:44.

Chapter summary

In this chapter we described the format of block-transfer read and write data. We also described the various commands used to program and configure your controller and gave you programming examples.

Chapter 5 Troubleshooting

Chapter objectives

In this chapter you will read how to troubleshoot your controller using the ACTIVE (green) and FAULT (red) indicators, block-transfer rungs in your ladder program and diagnostic bits in word 1 of the read-data file.

Troubleshooting table

The following table lists problems indicated by LED changes, possible causes and recommended action.

- ● LED ON
- o LED OFF

Indication	Description	Recommended Action
● Module active o Comm. fault o Module fault	Normal operation. Module should operate when the processor goes into the run mode and you transfer a logic program.	
o Module active o Comm. fault o Module fault	Module is not receiving DC power from the chassis backplane.	Check chassis power supply(ies). Make sure the controller is properly seated in the I/O chassis.
o Module active o Comm. fault ● Module fault	Module has detected a hardware fault in its power-up routine or a run time error occurred during normal operation.	Power-up the I/O chassis containing the controller. Return the module for repair if the module fault LED remains lit when you restore power.
● Module active ● Comm. fault o Module fault	Module has detected that communications to the processor or adapter has been lost. [1] Block-transfer error (out of sequence, time-out error, check-sum error)	Check cable connection on the remote I/O link. Replace adapter module, power supply, remote scanner or controller. Check block-transfer programming.

[1] Outputs are de-energized if a rack fault occurs with the last state switch set to "OFF". Controller will continue to execute the current program if the last stateswitch is set to "ON".

Causes of block-transfer errors

Look at the block-transfer rungs in the ladder diagram program. You have a block-transfer error when you see one or both of the following:

- The block-transfer error bits are intensified (PLC-3 processors).
- The enable and done bits of block-transfer instructions do not intensify or they remain intensified. They should alternately turn on (intensify) and turn off.

Block-transfer errors occur if:

- the module's location (rack, group, slot) in the I/O chassis does not match the rack, group and slot of the block-transfer instructions in the ladder program.
- the block lengths of block-transfer-read and write instructions are not equal (PLC-2 family processors). If they are different lengths, you must not enable the read and write instruction in the same scan.
- your conditioning instructions in the block-transfer rungs do not allow the rungs to turn off and on.
- you're using a PLC-2/20 or PLC-2/30 processor with remote I/O and you do not set the scanner for block-transfer operation.
- you're using a PLC-3 processor and you do not create block-transfer data files.

Errors indicated by diagnostic bits

Examine the diagnostic bits by displaying the read block of the block-transfer-read instruction. Refer to the programming manual of your processor for the procedure.

The upper byte of the first read-data word contains the diagnostic bits.

If this bit is set:	Then:
14	the controller has a hardware fault.
15	a program error in the hex file has been transferred to the controller.
16	the data transferred to the controller using block-transfer is valid (no program errors).
17	the controller has passed its power-up diagnostics routine and is ready to receive a program.

Chapter summary

This chapter showed you how to diagnose problems with your controller. It also explained the causes of block-transfer errors and the meaning of the diagnostic bits.

Appendix A Block-transfer timing

Block-transfer Timing for PLC-2 Family Processors

The time required for a block-transfer-read or -write operation for PLC-2 family processors depends on:

- the system scan time(s)
- the number of words to be transferred
- the I/O configuration
- the number of enabled block-transfer instructions in the ladder diagram program during any program scan

A block-transfer module performs only one block-transfer operation per I/O scan regardless of whether both read and write operations are requested. When done, the module toggles from one operation to the other in each program scan.

For a worst case calculation of the time between block transfers, assume that the number of enabled block-transfer instructions during any program scan is equal to the number of block-transfer modules in the system. Also assume that the controller is transferring 64 words in a write operation and ten words in the alternate read operation.

The method of calculating the worst case time between block transfers is covered for the following cases: PLC-2/30 remote and local systems, a PLC3 system, and a Mini-PLC-2/15 controller.

PLC-2/30 (PLC-2/20) Remote System

The system scan time for a remote PLC-2/30 or PLC-2/20 system is the sum of the processor scan time, the processor I/O scan time (between processor and remote distribution panel), and the remote distribution panel I/O scan time. The remote distribution panel can process only one block-transfer operation per remote distribution panel scan.

You can calculate the worst case time between transfers under normal operating conditions in three steps.

1. Calculate the system values that are determined by the system configuration.
   - Program Scan (PS) = (5 ms/1K words) x (number of program words)
   - Processor I/O Scan (PIO) = (0.5 ms/rack number) x declared rack numbers)

PLC-2/30 (PLC-2/20) Remote System (continued)

- Remote Distribution I/O Scan (RIO) = (7 ms/chassis) x (number of chassis)
- Number of Words Transferred (W) = 64 words for one write operation, 10 words for one read operation

2. Calculate the block-transfer time for a write operation (TW) and for a read operation (TR).

   - TW = (PS + PIO + 2 RIO + 0.5W + 13) ms
   - TR = (PS + PIO + 2 RIO + 0.5W + 4) ms

These equations are valid for up to 10,000 cable feet between the remote distribution panel and remote I/O chassis for a baud rate of 57.6kBd, or 5,000 cable feet at 115kBd.

3. Calculate the worst case system time (ST) between transfers.

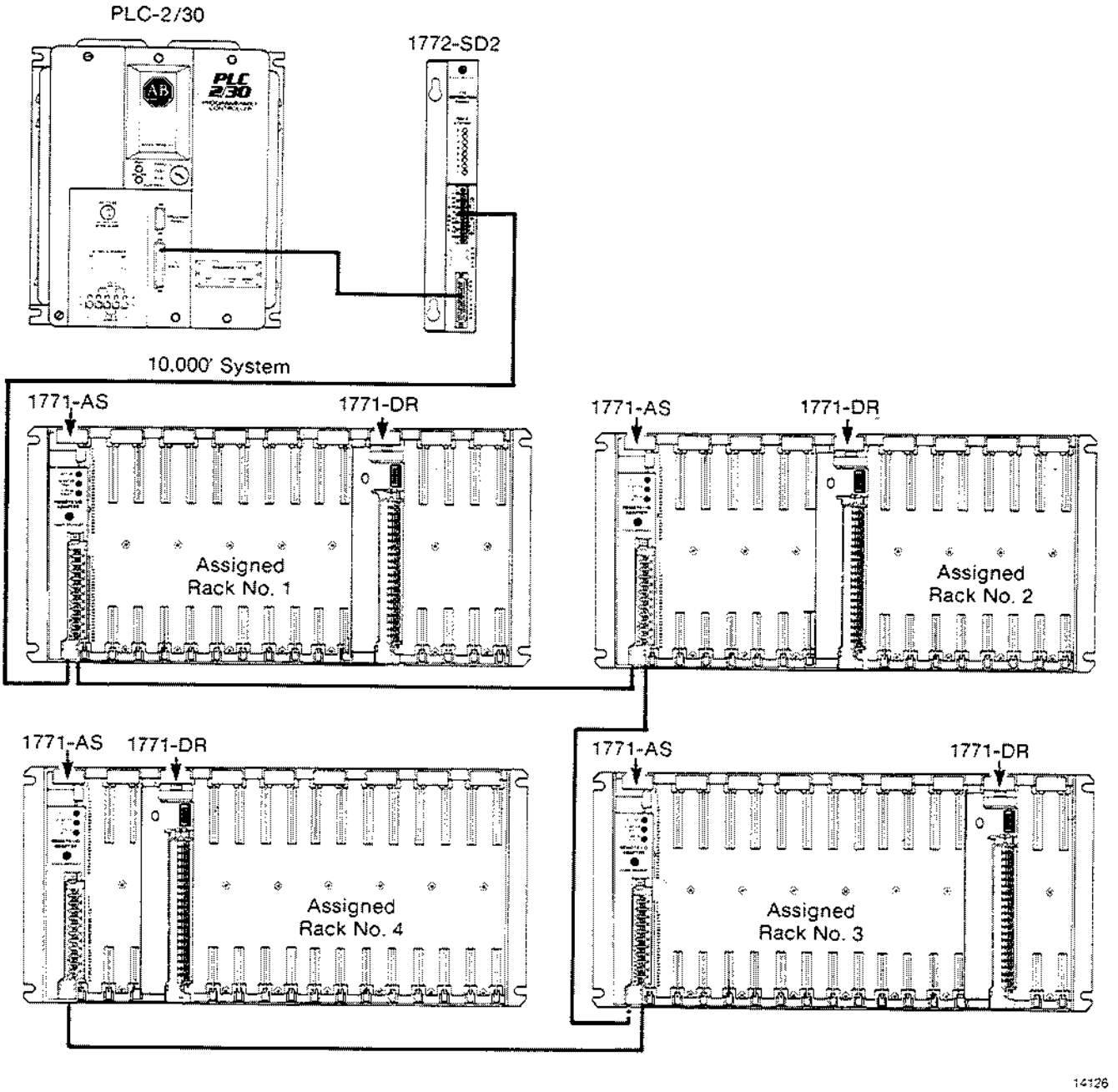
   - ST = Sum of transfer times of all block-transfer modules in a system taken worst case (read or write)

Example 1

A PLC-2/30 programmable controller is controlling 4 I/O chassis in a remote configuration with 1 assigned rack number per chassis (figure A.1). A controller is located in each chassis. Assume that 10 words are transferred in each read operation, 64 words are transferred in each write operation, and that the ladder diagram program contains 4K words. There are no other block-transfer modules in the system.

PLC-2/30 (PLC-2/20) Remote System (continued)

Figure A.1
PLC-2/30 Remote System Example



We want to find the worst case time between two consecutive block-transfer-read operations from the same module in this system.

PLC-2/30 (PLC-2/20) Remote System (continued)

Solution:

- Program length = 4K words (K = 1,024)
- Number of chassis = 4 (1 assigned rack number/chassis)
- Number of block-transfer words = 10 words (read) or 64 words (write)

1. Calculate the system values.
   - Processor Scan Time (PS) = (5 ms/1K words) x (4K words) = 20 ms
   - Processor I/O Scan Time (PIO) = (0.5 ms/rack number) x (4 rack numbers) = 2 ms
   - Remote Distribution I/O Scan Time (RIO) = (7 ms/chassis) x (4 chassis) = 28 ms
   - Number of Words Transferred = 10 (read) or 64 (write)
2. Calculate the block-transfer times for a write operation and for a read operation.
   - TW = (PS + PIO + 2(RIO) + 0.5W + 13) ms
     = (20 + 2 + 2(28) + 0.5(64) + 13) ms
     = 123 ms (write)
   - TR = (PS + PIO + 2(RIO) + 0.5W + 4) ms
     = (20 + 2 + 2(28) + 0.5(10) + 4) ms
     = 87 ms (read)
3. Calculate the worst case system time (ST) between 2 consecutive block-transfer-read operations.
   - ST = 4TW + 4TR
     = 4(123) + 4(87)
     = 840 ms

This is the worst case time between two consecutive block-transfer-read operations in the 4-chassis remote configuration described in example 1 (1 enabled controller in each chassis).

PLC-2/30 Local System

The system scan time for a local PLC-2/30 system is the program scan time plus the processor I/O scan time. Each block-transfer module is updated during a program scan.

The calculation of the worst case time between transfers can be done in three steps.

1. Calculate the system values that are determined by the system configuration.
   - Program Scan (PS) = (5 ms/1K words) x (number of program words)
   - Processor I/O Scan (PIO) = (1 ms/rack number) x (number of declared rack numbers)
   - Number of words transferred (W) = 10 (read) or 64 (write)
2. Calculate the block-transfer time (T) for the read or write operation.
   - T = 0.08 ms/word x number of words transferred

The same equation is used for either read or write transfer times.

3. Calculate the worst case system time (ST) between transfers.
   - ST = PS + PIO + T(1)(read) + T(2)(read) + T(3)(read) + ...
     PS + PIO + T(1)(write) + T(2)(write) + T(3)(write) + ...

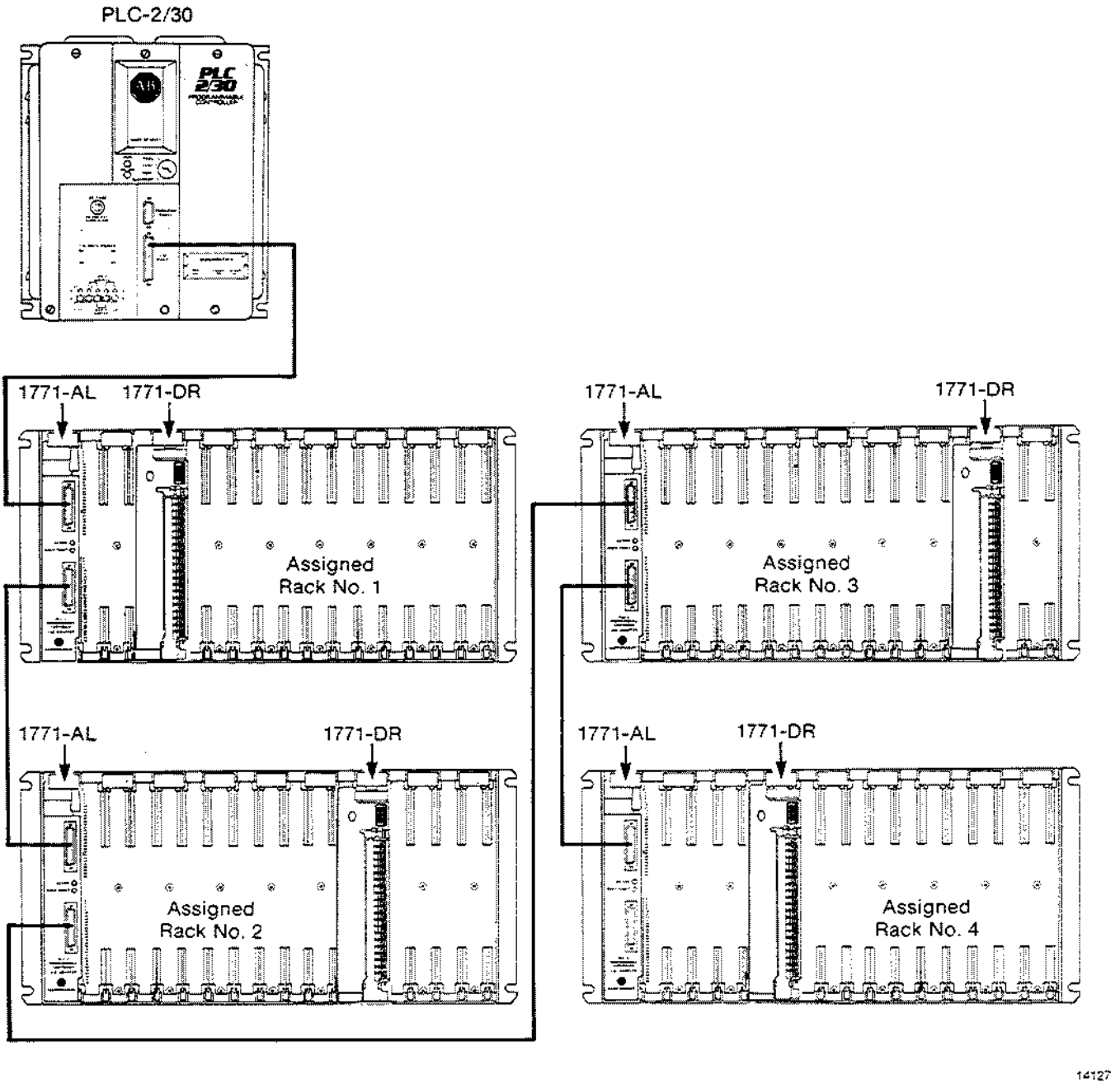
     = 2(PS + PIO) + T(1)(read) + T(2)(read) + T(3)(read) + ...
     T(1)(write) + T(2)(write) + T(3)(write) + ...

Example 2

A PLC-2/30 programmable controller is controlling four I/O racks in a local configuration. Assume 1 block-transfer module per chassis and 1 assigned rack number per chassis (figure A.2).

PLC-2/30 Local System *(continued)*

Figure A.2
PLC-2/30 Local System Example



Solution:

- Program length = 4K words
- Number of chassis = 4 (1 assigned rack number per chassis)
- Number of block-transfer words, W = 10 (read) or 64 (write)

PLC-2/30 Local System (continued)

1. Calculate the system values.
   - Processor Scan Time (PS) = (5 ms/1K words) x 4K words) = 20 ms
   - Processor I/O Scan Time (PIO) = (0.5 ms/rack number) x (4 rack numbers) = 2 ms
   - Number of Words Transferred (W) = 10 (read) or 64 (write)
2. Calculate the block-transfer times (T) for the read and write operation.
   - T = 0.08 ms/word x 10 words = .80 ms (read)
   - T = 0.08 ms/word x 64 words = 5.12 ms (write)
3. Calculate the worst case system time (ST) between 2 consecutive block-transfer-read operations.

The module toggles to a read operation in the scan following completion of the write operation and vice versa.

- ST = PS + T(1) + T(2) + T(3) + T(4)(writes)
  = PS + T(1) + T(2) + T(3) + T(4)(reads)
- ST = 2PS + 2PIO + 4T(read) + 4T(write)
  = 2(64) + 2(10) + 4(.80) + 4(5.12)
  = 128 + 20 + 3.2 + 20.48
  = 171.68 ms

This the worst case time between two consecutive block-transfer-read operations in the 4-chassis local configuration described in example 2 (one enabled controller in each chassis).

Mini-PLC-2/15 Controller

The Mini-PLC-2/15 scan is 15 ms for 1K program. Its I/O scan time is 5 ms. Each block-transfer module is updated during a program scan.

You can calculate the worst case time between transfers in two steps.

The facts are:

- Processor scan time (PS) = 15 ms/1K words
- Processor I/O scan time (PIO) = 5 ms
- Number of words transferred (W) = 10 (read) or 64 (write)

Mini-PLC-2/15 Controller *(continued)*

1. Calculate the block-transfer time (T) for the read and write operation.

   - T = 0.08 ms/word x number of words transferred

The same equation is used for read and write transfer times.

2. Calculate the worst case system time (ST) between two block-transfer- read operations.

   - ST = PS + PIO + T(read) + PS + PIO + T(write)

Example 3

A Mini-PLC-2/15 programmable controller is communicating with one controller in its I/O chassis. The ladder diagram program contains 2K words.

Solution:

The facts are:

- Program length = 2K words
- Processor scan time (PS) = (15 ms/1K words) x (2K words) = 30 ms
- Processor I/O scan time (PIO) = 5 ms
- Number of words transferred (W) = 10 (read), (64 write)

1. Calculate the block-transfer time (T) for the read and write operation.

   - T = 0.08 ms/word x 10 words (read)
     = 0.80 ms (read)

   - T = 0.08 ms/word x 64 words (write)
     = 5.12 ms (write)

2. Calculate the worst case system time (ST) between two consecutive block-transfer-read operations.

   - ST = PS + PIO + T(read) + PS + PIO + T(write)
     = 30 + 5 + .80 + 30 + 5 + 5.12
     = 75.92 ms

This is the worst case time between two consecutive block-transfer-read operations for the Mini-PLC-2/15 controller.

Block-transfer Timing for PLC-3 Family Processors

The execution time required to complete a block-transfer-read or -write operation with a PLC-3 family processor depends on the number of:

- words of user program
- active I/O channels on the scanner
- I/O chassis entries in the rack list for the channel
- I/O channels on the scanner that contain block-transfer modules
- block-transfer modules on the channel (if the I/O chassis containing a block-transfer module appears more than once in the I/O chassis rack list, count the module once each time the chassis appears in the rack list)

The typical time required for the controller to complete a block- transfer-read/-write (bidirectional) depends on the program scan and the scanner scan as follows:

Time [read/write] = program scan + 2(scanner scan)

Program Scan: The program scan is approximately 2.5 ms per 1K words of user program when using examine on/off and block instructions.

Scanner Scan: The time required for the scanner to complete a read- or write-block transfer depends on the number of other block-transfer modules on the same scanner channel that are enabled simultaneously.

Block-transfer times typically are similar regardless of the type of block- transfer module, the number of words transferred, or whether a read or write operation is requested.

A block-transfer I/O channel is a channel that contains one or more block- transfer modules located in any chassis connected to the channel.

An I/O chassis can appear more than once in a rack list of I/O chassis. Count the chassis and the block-transfer module(s) that it contains as often as it is listed.

The procedure for calculating block-transfer timing for a PLC-3 processor is given here followed by a worksheet and an example calculation:

Block-transfer Timing for PLC-3 Family Processors (continued)

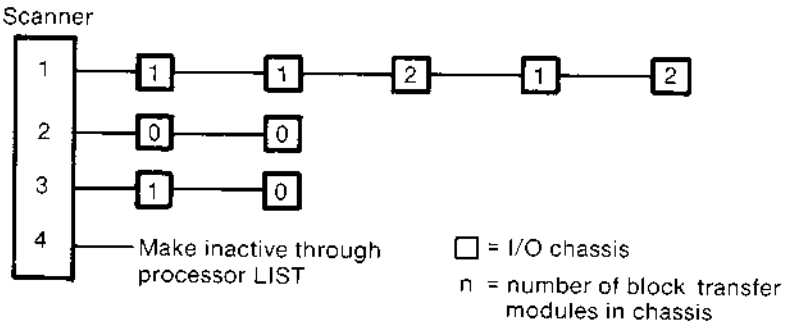
1. Determine the number of active I/O channels on the scanner and the number of I/O channels with block-transfer modules. Show the number of:
   - block-transfer modules in each I/O chassis
   - block-transfer I/O channels
   - I/O chassis entries in the rack list for each block-transfer I/O channel
   - active I/O channels per scanner
2. Determine the nominal block-transfer time.
3. Compute the approximate scanner time for each block-transfer channel.
4. Compute the controller read-/write-block-transfer time.

Example Computation

An example computation to determine the block-transfer timing with a PLC-3 family processor follows. The example is based on these facts:

- user program contains 20K words
- channel 1 contains five I/O channels with a total of seven block-transfer modules including one controller module
- channel 2 contains two I/O chassis with no block-transfer modules
- channel 3 contains two I/O chassis with one controller module
- channel 4 is made inactive through processor LIST

1. Diagram the chassis connected in series to each channel (up to four) of your scanner module. Then, fill in the information called for below. Example values have been added.



Block-transfer Timing for PLC-3 Family Processors *(continued)*

Description	Number	Ch1	Ch2	Ch3	Ch4
Active I/O channels	3				
Block-transfer I/O channels	2				
Block-transfer modules on each I/O block-transfer channel		7	0	1	0
I/O chassis on each block-transfer I/O channel (I/O chassis in rack list)		5	0	2	0

2. Determine a time from the table. Example values have been added.

Number of Active I/O Channels

Active I/O channels containing one or more block-transfer modules		1	2	3	4
	1	40	52	54	58
	2		67	68	76
	3			98	99
	4				123
	Time (ms)				

Number of active I/O channels: 3

Number of active I/O channels containing one or more block-transfer module: 2

Time from table: 66 ms

Block-transfer Timing for PLC-3 Family Processors (continued)

3. Compute the scanner times for each block-transfer channel. Example values have been added.

(CT = Channel Time).

CT = [Time] x [# BT modules] + [# I/O chassis -1] x 9 ms
(table) on BT channel on BT channel

CT1 = [68] x 7 + [5-1] x 9
= [68] x [7] + [4] x 9
= 476 + 36
= 512 ms

CT2 = Not a block-transfer channel

CT3 = [68] x [1] + 1 x 9
= 68 + 9
= 77 ms

CT4 = Not an active channel

4. Compute the controllers read-/write-block transfer time. Example values have been added.

Program Scan:

Time (program) = 2.5 ms/1K words x 20K words
= 2.5 x 20
= 50 ms

Scanner Scan:

Time (read or write) = 512 ms for channel 1 and 77 ms for channel 3 (from Step 3).

Read/Write

Time (controller module in channel 1) = Program scan + 2 [Scanner scan]
= 50 + 2 [512]
= 50 + 1024
= 1074 ms
= 1.1 seconds

Time (controller module in channel 3) = Program scan + 2 [Scanner scan]
= 50 + 2 [77]
= 204 ms

Appendix B Hex input conversion table

Input	Hex
A	01
B	02
C	04
D	08
E	10
F	20
G	40
H	80
AB	03
AC	05
AD	09
AE	11
AF	21
AG	41
AH	81
BC	06
BD	0A
BE	12
BF	22
BG	42
BH	82
CD	0C
CE	14
CF	24
CG	44
CH	84
DE	18
DF	28
DG	48
DH	88
EF	30
EG	50
EH	90
FG	60
FH	A0
GH	C0
ABC	07
ABD	0B
ABE	13
ABF	23
ABG	43
ABH	83
ACD	0D
ACE	15
ACF	25
ACG	45
ACH	85
ADE	19
ADF	29
ADG	49
ADH	89
AEF	31
AEG	51
AEH	91
AFG	61
AFH	A1
AGH	C1

Input	Hex
BCD	0E
BCE	16
BCF	26
BCG	46
BCH	86
BDE	1A
BDF	2A
BDG	4A
BDH	8A
BEF	32
BEG	52
BEH	92
BFG	62
BFH	A2
BGH	C2
CDE	1C
CDF	2C
CDG	4C
CDH	8C
CEF	34
CEG	54
CEH	94
CFG	64
CFH	A4
CGH	C4
DEF	38
DEG	58
DEH	98
DFG	68
DFH	A8
DGH	C8
EFG	70
EFH	B0
EGH	D0
FGH	E0
ABCD	0F
ABCE	17
ABCF	27
ABCG	47
ABCH	87
ABDE	1B
ABDF	2B
ABDG	4B
ABDH	8B
ABEF	33
ABEG	53
ABEH	93
ABFG	63
ABFH	A3
ABGH	C3
ACDE	1D
ACDF	2D
ACDG	4D
ACDH	8D
ACEF	35
ACEG	55
ACEH	95

Input	Hex
ACFG	65
ACFH	A5
ACGH	C5
ADEF	39
ADEG	59
ADEH	99
ADFG	69
ADFH	A9
ADGH	C9
AEFG	71
AEFH	B1
AEGH	D1
AFGH	E1
BCDE	1E
BCDF	2E
BCDG	4E
BCDH	8E
BCEF	36
BCEG	56
BCEH	96
BCFG	66
BCFH	A6
BCGH	C6
BDEF	3A
BDEG	5A
BDEH	9A
BDFG	6A
BDFH	AA
BDGH	CA
BEFG	72
BEFH	B2
BEGH	D2
BFGH	E2
CDEF	3C
CDEG	5C
CDEH	9C
CDFG	6C
CDFH	AC
CDGH	CC
CEFG	74
CEFH	B4
CEGH	D4
CFGH	E4
DEFG	78
DEFH	B8
DEGH	D8
DFGH	E8
EFGH	F0
ABCDE	1F
ABCDF	2F
ABCDG	4F
ABCDH	8F
ABCEF	37
ABCEG	57
ABCEH	97
ABCFG	67
ABCFH	A7

Input	Hex
ABCGH	C7
ABDEF	3B
ABDEG	5B
ABDEH	9B
ABDFG	6B
ABDFH	AB
ABDGH	CB
ABEFG	73
ABEFH	B3
ABEGH	D3
ABFGH	E3
ACDEF	3D
ACDEG	5D
ACDEH	9D
ACDFG	6D
ACDFH	AD
ACDGH	CD
ACEFG	75
ACEFH	B5
ACEGH	D5
ACFGH	E5
ADEFG	79
ADEFH	B9
ADEGH	D9
ADFGH	E9
AEFGH	F1
BCDEF	3E
BCDEG	5E

Input	Hex
BCDEH	9E
BCDFG	6E
BCDFH	AE
BCDGH	CE
BCEFG	76
BCEFH	B6
BCEGH	D6
BCFGH	E6
BDEFG	7A
BDEFH	BA
BDEGH	DA
BDFGH	EA
BEFGH	F2
CDEFG	7C
CDEFH	BC
CDEGH	DC
CDFGH	EC
CEFGH	F4
DEFGH	F8
ABCDEF	3F
ABCDEG	5F
ABCDEH	9F
ABCDFG	6F
ABCDFH	AF
ABCDGH	CF
ABCEFG	77
ABCEFH	B7
ABCEGH	D7

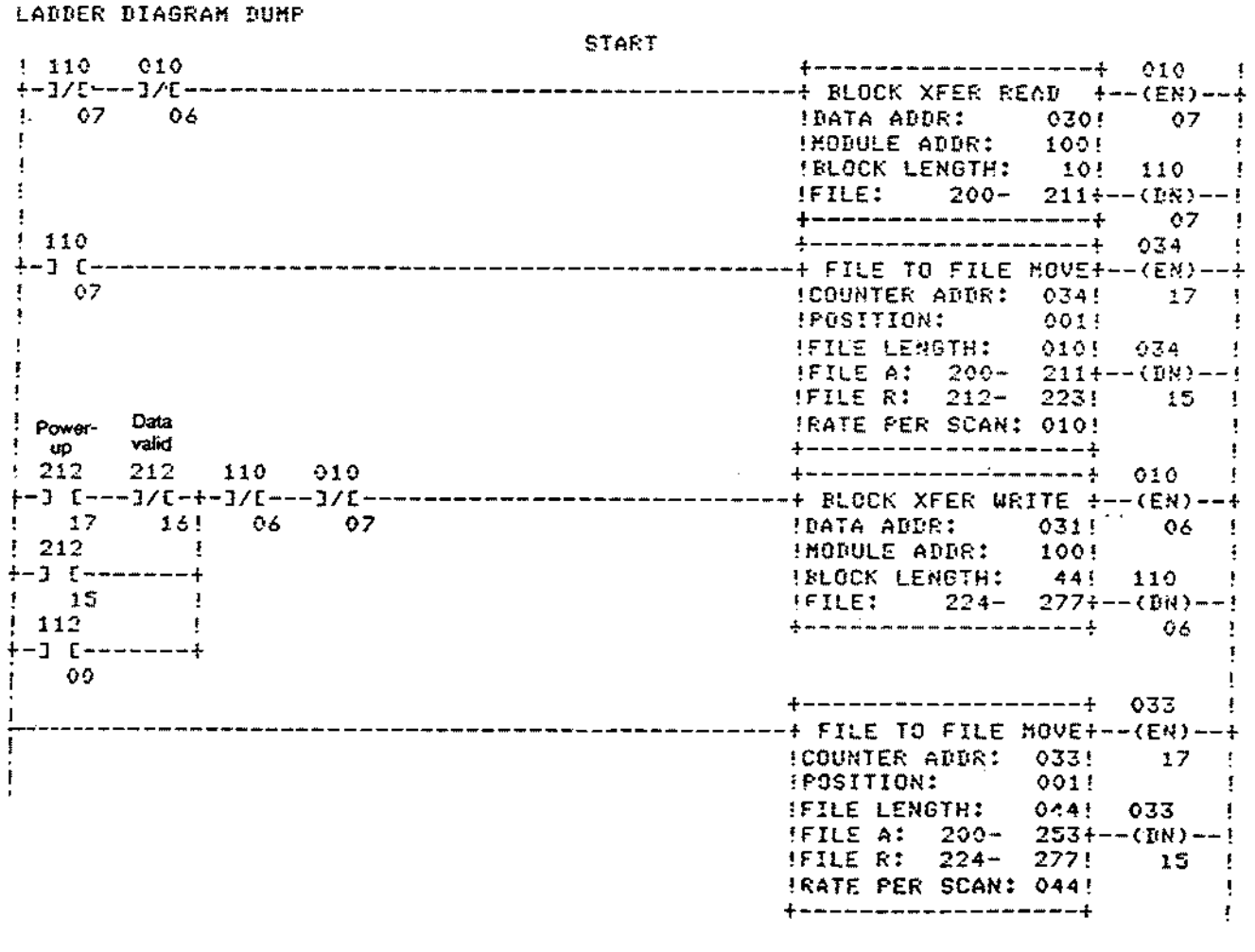
Input	Hex
ABCFGH	E7
ABDEFG	7B
ABDEFH	BB
ABDEGH	DB
ABDFGH	EB
ABEFGH	F3
ACDEFG	7D
ACDEFH	BD
ACDEGH	DD
ACDFGH	ED
ACEFGH	F5
ADEFGH	F9
BCDEFG	7E
BCDEFH	BE
BCDEGH	DE
BCDFGH	EE
BCEFGH	F6
BDEFGH	FA
CDEFGH	FC
ABCDEFG	7F
ABCDEFH	BF
ABCDEGH	DF
ABCDFGH	EF
ABCEFGH	F7
ABDEFGH	FB
ACDEFGH	FD
BCDEFGH	FE
ABCDEFGH	FF

Appendix C Block-transfer ladder diagram examples

Read-only block-transfer for PLC-2 family processors

Figure C.1 shows example rungs for a read-only block-transfer operation. Use this example to optimize your block-transfer timing. This example shows the transfer of configuration data only, with no on-line commands.

Figure C.1
Example Read-only block-transfer program for PLC-2 family processors



Read-only block-transfer for PLC-2 family processors *(continued)*

Figure C.1 (continued)

Data address: 030/031

This is the first possible address in the timer/counter area of the data table. Use the first available timer/counter address for your first block-transfer module data address.

Module address: 100

The module is located in rack 2, I/O group 2, slot 0. Two slot modules are addressed in slot 0 of the particular group.

Block length: 010/044

The controller has default block lengths of 10 words for the read instruction and 64 words for the write instruction. To optimize block-transfer timing, the actual values for the block lengths should be programmed in the read and write block transfer instructions. When different block lengths are programmed for the read and write instructions, they must not be enabled in the same scan.

File: 200/224

This is the address of the first word of the read/write file. Use a file-to- file move to buffer your read data. Use addresses 212 through 223 when making data comparisons.

This example is a read only operation. Use it to increase the processor's update time of the module's status.

Rung descriptions

Rung 1 - The block-transfer-read (BTR) instruction is enabled if the read-done bit (110/07) and write-enable bit (010/06) are off.

Rung 2 - Use a file-to-file move to buffer the read data. The BTR data is buffered if the BTR done bit (110/07) is on. Use addresses 212 through 223 when making data comparisons.

Rung 3 - Bit 212/17 is on after the first BTR. Bit 212/16 is off only until the first block-transfer-write (BTW) command. As a result, one BTW occurs to initialize the module and then all block-transfers are read instructions.

If the programming error bit (212/15) is on, and the BTW done bit and BTR enable bits are both off, a BTW takes place. If the power-up bit (212/17) is on and the data valid bit (212/16) is off, no program exists in the controller. A BTW takes place if the BTW done bit is off and the BTR enable bit is off.

The BTW instruction can be manually selected by pushbutton 112/00 as long as the BTR enable (010/07) and the BTW done (110/06) are both off.

Rung 4 - This rung is for display purposes only.

Rung descriptions (continued)

WARNING: When the block lengths of bidirectional block-transfer instructions are set to unequal values, do not enable the rung containing the alternate instruction until the done bit of the first transfer is set. If you enable them in the same scan:

- the number of words transferred may not be the number intended
- invalid data could be operated upon in subsequent scans
- output devices could be controlled by invalid data

Unexpected and/or hazardous machine operation could occur. Damage to equipment and/or personal injury could result.

Rung descriptions
(continued)

Figure C.2
Example values entered in the read- and write-data files (PLC-2 family processors)

Counter addr: 033 File A: 200- 253	hexadecimal data monitor File to file move Position: 001	File length: 044 File R: 224- 277
Position	**File A data**	**File R data**
001	C100	0001
002	130E	0013
003	130E	FF00
004	0000	1000
005	8500	0850
006	0000	C707
007	0000	8880
008	0000	3830
009	0000	0014
010	2121	FF00
011	C100	1000
012	130E	0850
013	130E	7A00
014	0000	C040
015	0850	0015
016	0000	FF00
017	0000	0100
018	0000	0010
019	0000	FF01
020	2121	5744
021	0001	0011
022	0013	FF02
023	FF00	4B40
024	1000	1000
025	0850	0012
026	C707	FF02
027	8880	1818
028	3830	0017
029	0014	FF03
030	FF00	9125
031	1000	8181
032	0850	0016
033	7A00	FF03
034	C040	2E02
035	0015	4E02
036	FF00	0018
037	0100	FF03
038	0010	8000
039	FF01	0030
040	5744	0000
041	0011	9999
042	FF02	0850
043	4B40	2500
044	1000	0003
	read-data file	**write-data file**

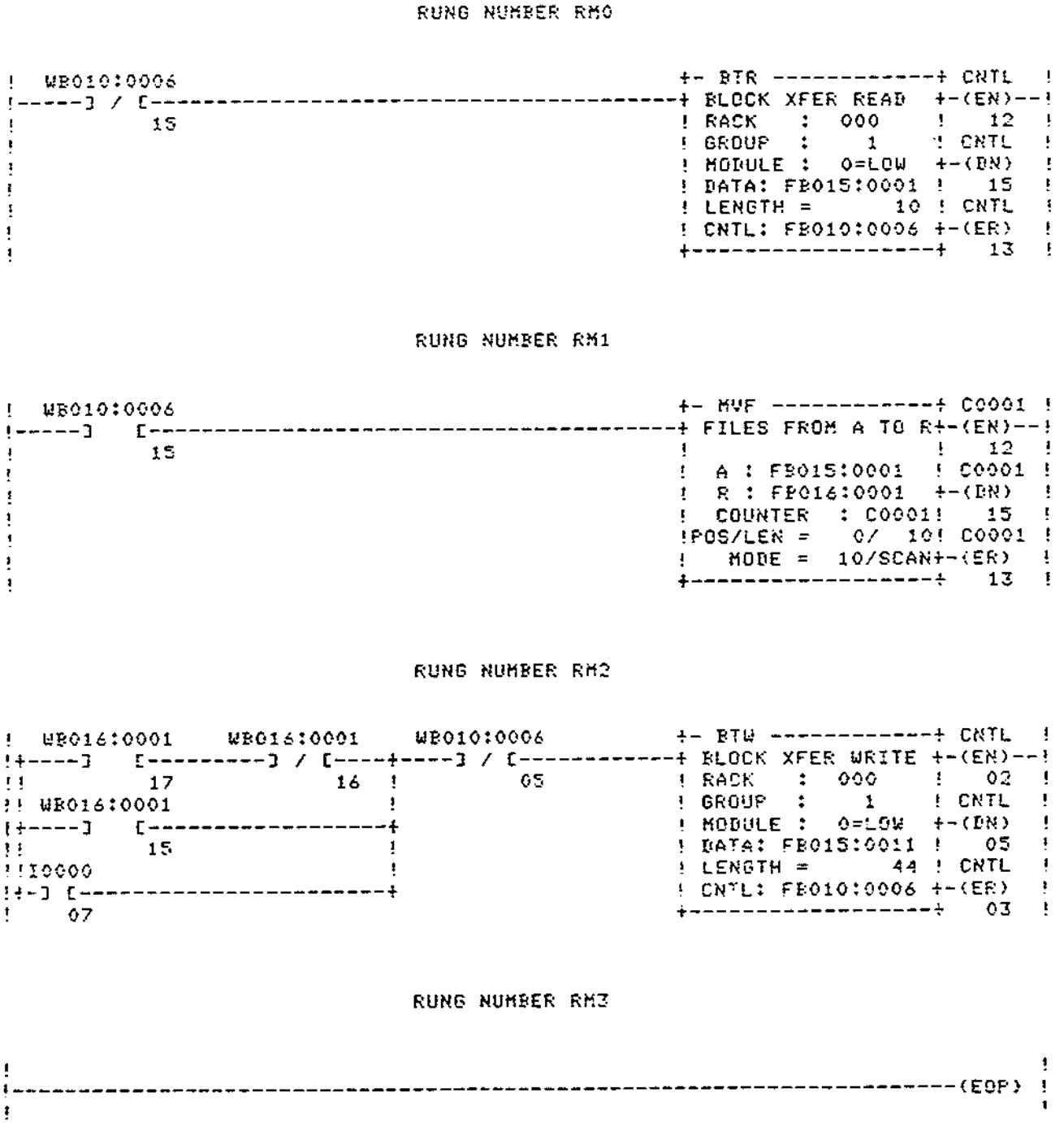
Rung descriptions
(continued)

Position	Read data file	
001	C100	Indicates that the module is functioning properly and that the programming format is in terms of ladder expressions.
002	130E	The upper byte (13) indicates that the is true (10 to 24 V at inputs 0, 1 and 4). The lower byte (0E) indicates that 5 to 24 V is present at outputs 1, 2 and 3.
003	130E	The upper byte indicates that all inputs for the controller are originating from the wiring arm. The lower byte (0E) indicates the state the outputs should be in is based upon the logic equations resident on the controller. Outputs 1,2 and 3 are energized based upon the logic equations.
004	0000	Address pointer not used.
005	0850	Indicates that the accumulated count (UCT, DCT, TEC) for output 0 is currently 850.
006	0000	Indicates that the accumulated count (UCT, DCT, TEC) for output 1 is zero or that output 1 is currently not being used as a counter instruction.
007	0000	Indicates that the accumulated count (UCT, DCT, TEC) for output 2 is zero or that output 2 is currently not being used as a counter instruction.
008	0000	Indicates that the accumulated count (UCT, DCT, TEC) for output 3 is zero or that output 3 is currently not being used as a counter instruction.
009	0000	Indicates that there are no program errors to report.
010	2121	The upper byte (21) indicates that the Communication Processor is Series A, Revision A and the lower byte (21) indicates that the Control Processor is also at Series A, Revision A.
Position	**Write data file**	
001 through 044		Represents the converted hex file of the ladder logic expressions in the programming example. (Refer to "Programming Example" in Chapter 4.)

Read-only block-transfer for PLC-3 family processors

Figure C.3 shows example rungs for a read-only block-transfer operation. Use this example to optimize your block-transfer timing.

Figure C.3
Example Read-only block-transfer program for PLC-3 family processors



Read-only block-transfer for PLC-3 family processors
(continued)

START=WB015:0000

WORD #	0	1	2	3	4	5	6	7
00000	0000	0000	0000	0000	0000	0000	0000	0000
00008	0000	0000	0000	0001	0013	FF00	1000	0850
00016	C707	8880	3830	0014	FF00	1000	0850	7A00
00024	C040	0015	FF00	0100	0010	FF01	5744	0011
00032	FF02	4B40	1000	0012	FF02	1818	0017	FF03
00040	9125	8181	0016	FF03	2E02	4E02	0018	FF03
00048	8000	0030	0000	9999	0850	2500	0003	0000
00056	0000	0000	0000	0000	0000	0000	0000	0000
00064	0000	0000	0000	0000	0000	0000	0000	0000
00072	0000	0000	0000	0000	0000	0000	0000	0000
00080	0000	0000	0000	0000	0000	0000	0000	0000
00088	0000	0000	0000	0000	0000	0000	0000	0000
00096	0000	0000	0000	0000	0000			

Rung 0 - The block-transfer-read (BTR) instruction is conditioned by the BTR done bit (B10:6/15). This causes the necessary false-to-true transition of the instruction.

Rung 1 - The BTR data is buffered into B16:1 when the BTR done occurs.

Rung 2 - If the power-up bit (B16:1/17) is on and the data valid bit (B16:1/16) is off, no program exists in the controller. A block-transfer write (BTW) takes place if the BTW done bit is off. The power-up bit is on and the data valid bit is off only until the first BTW command. As a result, one BTW occurs to initialize the module and then all block-transfers are read instructions.

If a programming error occurs in the module bit B16:1/15 is energized. This enables a BTW instruction if the BTW enable bit is off.

The BTW instruction can be manually selected by pushbutton (I0000/7) if the BTW done bit is on.

Section Page

ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

Programmable Controller Division
747 Alpha Drive, Cleveland, Ohio 44143

Publication 1771-6.5.35 — October, 1986

P/N 955101-45