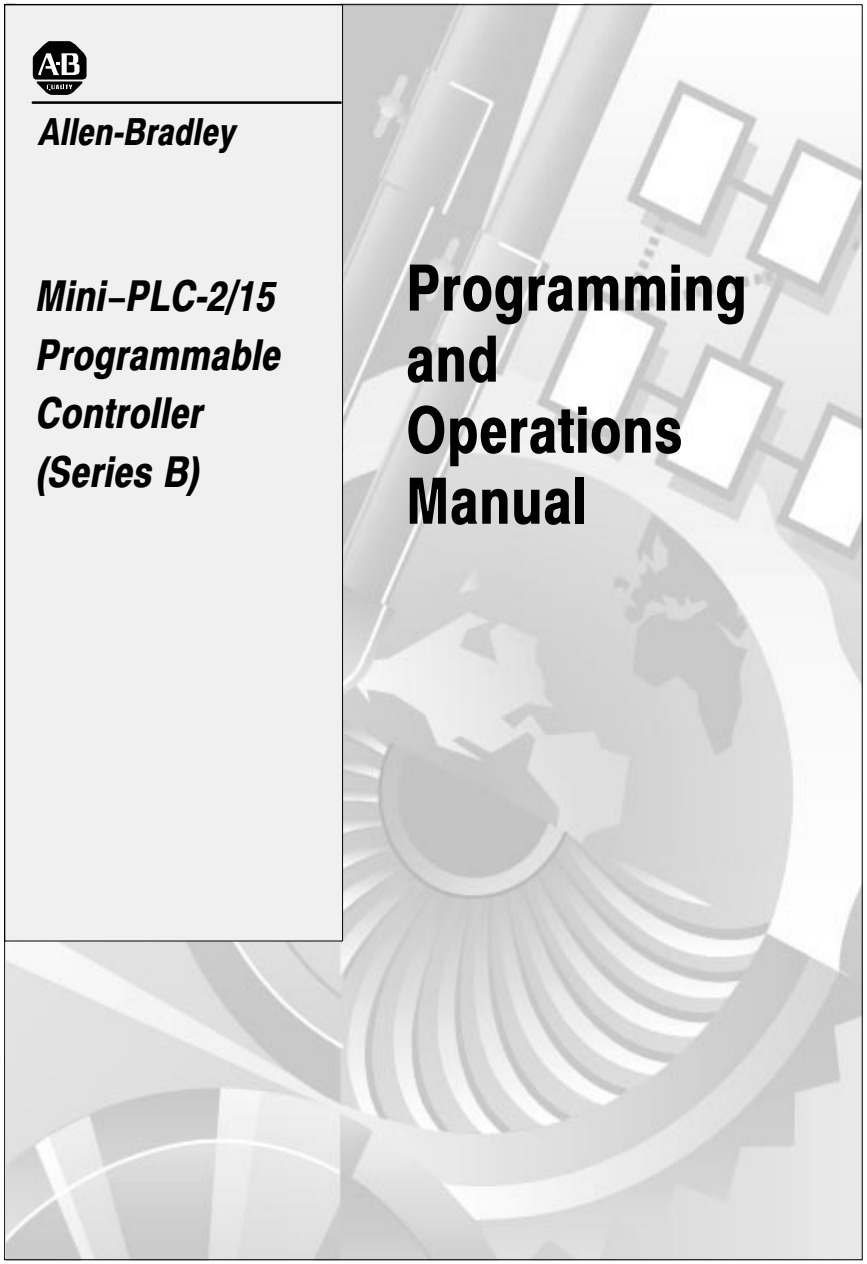


Allen-Bradley

Mini–PLC-2/15 Programmable Controller (Series B)

Programming and Operations Manual

Table of Contents

Chapter 1

Introduction

NOTE: Read this chapter before you use the series B Mini-PLC-2/15 programmable controller (cat. no. 1772-LV). It will tell you how to use this manual properly and efficiently.

Purpose

The Mini-PLC-2/15 programmable controller has been revised to meet customer needs when programming. An additional EAF EPROM hardware feature has been added to meet your programming needs. If you choose to take advantage of this hardware feature, consult your local Allen-Bradley Distributor or Sales Representative for additional product information.

To The Reader

This manual focuses on the Mini-PLC-2/15 system. It is divided into three major sections: Operations, Programming, and a Quick Reference.

The Programming section informs you about basic theory concerning the hardware features and programming techniques available to you when using this system.

The Operations section informs you step by step about each programming function.

The Quick Reference section acts as a guide so you can minimize production down time.

This manual is procedure oriented. This means that it will tell you how to operate your Mini-PLC-2/15 system. our training center educates you about the Allen-Bradley technology. If you are a new user and are unfamiliar with our technology we suggest that you contact our training center:

Allen-Bradley Training Center
6880 Beta Drive
Highland Heights, Ohio 44143
Phone: (216) 646-6777

Vocabulary

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

- Series B Mini-PLC-2/15 programmable controller as “the controller” or “the processor.”
- Execute Auxiliary Function as “EAF”
- Programmable Read Only Memory as “PROM”
- Erasable Programmable Read Only Memory as “EPROM”

A glossary section located in the back of this manual clarifies technical terms.

Manual Design

Each page consists of headings, text, and illustrations.

- Headings in the left margin describe the contents of the text.
- Text in the right margin defines an instruction, technique, or an operating principle.
- Illustrations display operating features of the controller, or they show how each program appears on the industrial terminal.

Conventions

The term “syntax,” is used throughout the entire manual. it is used to describe the arrangement of an instruction on a rung.

Words in [ ] denote the key name or key symbol.

Words in ( ) denote information that you must provide. For example, an address value.

Data table word address values are reported in octal values.

Related Publications

Our Publication Index (publication SD499) is a guide to further inform you about products related to our series B Mini-PLC-2/15 programmable controller. Consult your local Allen-Bradley distributor or sales representative for information regarding this publication or any needed information.

Chapter 2

An Introduction to Programmable Controllers

Objectives

This chapter reviews general fundamentals common to our programmable controllers (PC's). When you are finished, you will have read several important concepts that will help you understand this manual. You'll be able to:

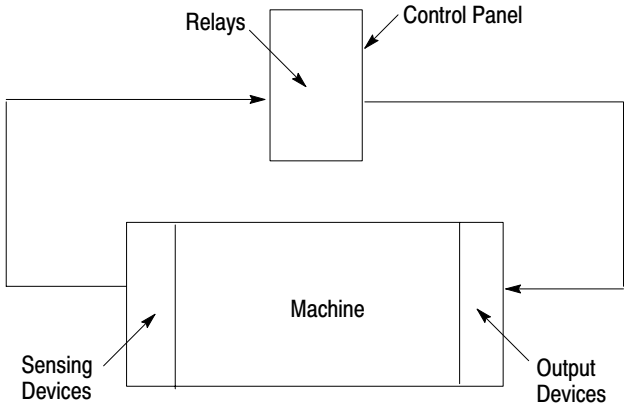
- Describe what a programmable controller does.
- List and describe the functions of the four major sections of a programmable controller.
- Describe how the four major sections of a programmable controller interact.
- Give an example of a simple program.

If you can do all this now, then turn directly to chapter three.

Traditional Controls

You are probably familiar with the traditional methods of machine control (Figure 2.1). Sensing devices located on the machine detect changes in the machine's condition. For instance, a part arriving at a work station would contact and close a limit switch, the sensing device. As a result, an electrical circuit is completed and a signal is sent to the control panel.

Figure 2.1
Traditional Methods of Machine Control

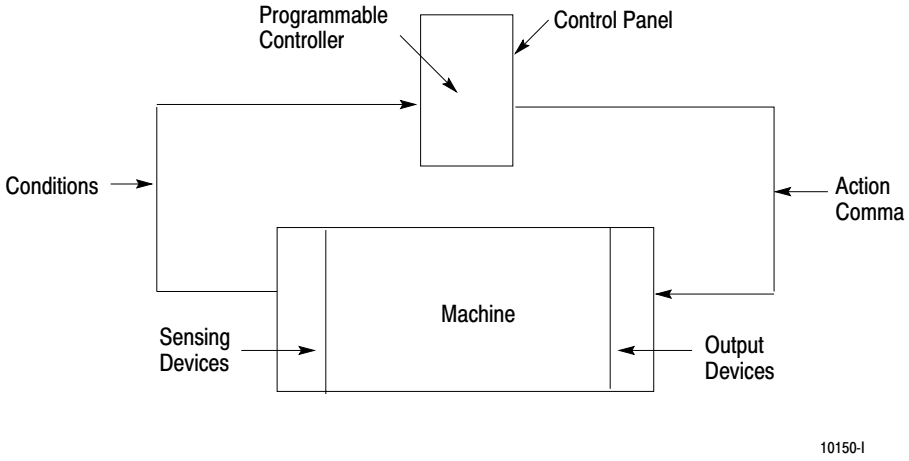


At the control panel, the electrical signal enters a bank of relays or other devices, such as solid state modules. Circuits within the control panel open or close causing additional electrical signals to be sent to output devices at the machine. For example, a relay energized by the limit switch may complete another circuit energizing the output device, a clamp, which secures the part at the work station.

Programmable Control

Programmable controllers can perform many of the functions of traditional controls (Figure 2.2). Sensing devices and output devices are located at the machine and perform the same jobs. The field wiring between the machine and the control panel provides electrical paths from the sensing devices to the control panel, and from the control panel to the output devices.

Figure 2.2
Machine Control with a Programmable Controller



However, inside the control panel you'll find a programmable controller rather than relays or discrete solid state devices. Instead of wiring those devices and relays together to produce a desired response, you simply tell your programmable controller how you want it to respond to the same conditions. you do this with a program.

Programming is telling your programmable controller what you want it to do. A program is nothing more than a set of instructions you give the programmable controller telling it how to react to different conditions at the machine.

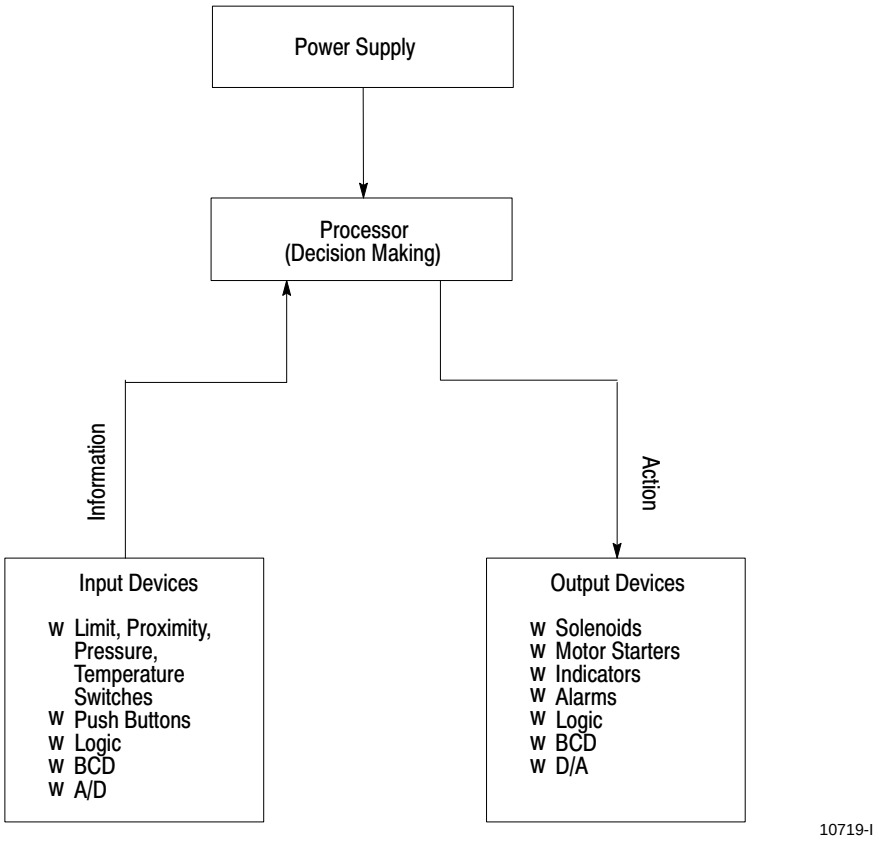
The Four Major Sections

Let's take a closer look at a typical programmable controller system. it usually consists of four major sections:

- Power supply
- Input section (connects to input devices)
- Output section (connects to output devices)
- Processor section

Figure 2.3 shows these sections

Figure 2.3
The Four Major Sections of a Programmable Controller



Power Supply

The power supply provides a low level DC voltage source for the electronic circuitry of the programmable controller. It converts the higher level line voltages to low level logic voltages required by the programmable controller's electronic circuitry.

Input Section

The input section serves four very important functions:

- Termination
- Indication
- Conditioning
- Isolation

Termination

The input section provides terminals for the field wiring coming from the sensing devices on the machine.

Indication

The input section of most modules also provides a visual indication of the state of each input terminal with indicators. The indicator is on when there is a voltage applied to its terminal. It is off when there is no voltage applied to its terminal. Since the indicator reveals the status of its terminal, it's usually called an input status indicator.

You should also notice another important characteristic of input indicators. They are only associated with terminals used for wiring sensing devices to the input section. The terminal that's used to provide a ground for the sensing circuits has no indicator.

Conditioning

Another function of the input section is signal conditioning. The electrical power used at the machine is usually not compatible with the signal power used within the programmable controller. Therefore, the input section receives the electrical signal from the machine and converts it to a voltage compatible with the programmable controller's circuitry.

Isolation

The input section isolates the machine circuitry from the programmable controller's circuitry. Isolation helps to protect the programmable controller's circuitry from unwanted and dangerous voltage levels that may occur occasionally at the machine or in the plant's wiring system.

Output Section

The output section has functions similar to those of the input section:

- Termination
- Indication
- Conditioning
- Isolation

Termination

The output section provides terminals for the field wiring going to the output devices on the machine.

Indication

The output section of most modules provides a visual indication of the state of each output device with indicators.

The output status indicator will be on when the output device is energized. A common term applied to either input status indicators or output status indicators is I/O status indicators. I/O stands for either input or output.

In addition, the output section of modules with fuses has blown fuse indicators. Typically, each output circuit is fused in the output section. Groups of these fuses will have a blown fuse indicator associated with them. When one of the fuses in the group opens, the blown fuse indicator will be lit.

Conditioning

The output section conditions the programmable controller's signals for the machine. that is, it converts the low-level DC voltages of the programmable controller to the type of electrical power used by the output devices at the machine.

Isolation

The output section isolates the more sensitive electronic circuitry of the programmable controller from unwanted and dangerous voltages that occasionally occur at the machine or the plant's wiring system. There are situations where additional external protection may be required.

Processor Section

The four major section of a programmable controller is the processor. The processor section might be called the "brains" of the programmable controller because it is divided into halves that serve functions similar to your brain. One half is the Central Processing Unit (CPU), the other is memory.

Central Processing Unit

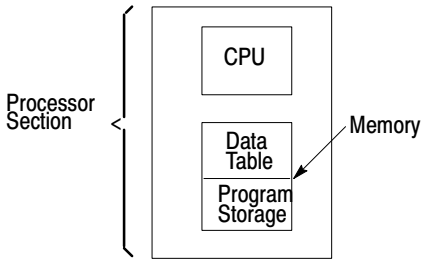
The Central Processing Unit (CPU) is divided into two sections:

- Decision area - makes decisions about what the machine is to do.
- Memory - storage area.

Memory

The programmable controller's memory serves three functions:

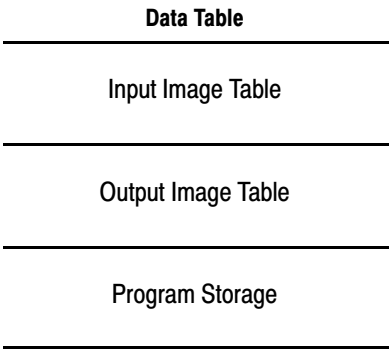
- Stores important information (or data) that the CPU may need to make its decisions.
- Stores sets of instructions called a program.
- Stores messages.



Data Table

The area of memory, where data is controlled and utilized, is called the data table. The data table is divided into several smaller sections according to the type of information to be remembered. These smaller sections are called:

- Input image table
- Output image table
- Processor work areas (2)
- Timer/Counter accumulated values and internal storage
- Timer/Counter preset values and internal storage



At this time, we will only discuss the input and output image tables. Chapter 4 discusses the remaining areas.

Image Tables

The input image table reflects the status of the input terminals. The output image table reflects the status of the output terminals.

Each image table is divided into a number of smaller units called bits. A bit is the smallest unit of memory. A bit is a tiny electronic circuit that the CPU can turn on or off. Bits in the image table are associated with a particular I/O terminal in the input or output section.

When the CPU detects a voltage at an input terminal, it records that information by turning the corresponding bit on. Likewise, when the CPU detects no voltage at an input terminal, it records that information by turning the corresponding bit off. If, while executing your program, the CPU decides that a particular output terminal should be turned on or off, it records that decision by turning the corresponding bit on or off. In other words, each bit in the I/O image tables corresponding to the on or off status of an I/O terminal.

When people who work with PCs talk about turning a bit on, they use the term “set.” For example, “The CPU sets the bit.” means “turns it on”. On the other hand, they use the term “reset” when they talk about turning the bit off. For example, “The CPU reset the bit.”

Picture memory as a page that has been divided into many blocks. Each block represents one bit. you now know that each bit is either on or off. We could show the state of each bit by writing “on” or “off” into each block. However, there is an easier way. We can agree that the numeral one (1) means on and that the numeral zero (0) means off.

We can easily and quickly show the status of each bit by writing one (1 or zero (0) into the appropriate block. Most people who work with PCs show bit status in this way. Frequently, you’ll hear them use expressions like, “The CPU responded by writing a one into the bit when the limit switch closed.” Of course, the CPU didn’t really write a one into memory, it simply set the bit by turning it on.

If you heard the expression, “The CPU wrote a zero into that bit location,” what actually happened? If you said the CPU merely reset the bit by turning it off, you’re right. Remember,

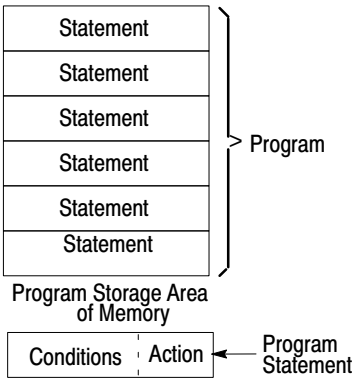
When the I/O device is:	The bit status is said to be:
on	on 1 set
off	off 0 reset

Program Storage

The other major area of memory, program storage, takes up the largest portion of memory. You’ll recall that this is where your instructions to the programmable controller are stored. You’ll also recall that this set of instructions is called a program.

Program Language

A program is made up of a set of statements. Each statement does two things. First, it describes an action to be taken. For instance, it might say, “Energize motor starter number one.” Second, it describes the conditions that must exist in order for the action to take place.



For example, you may want the action to take place, “Whenever a certain limit switch closes.” So your condition could be, “If limit switch number two is closed,...” The action would be, “energize motor starter number one.” The entire statement would then read, “If limit switch number two is closed, then energize motor starter number one.” Therefore, when limit switch number two at the machine is closed, the programmable controller would energize the motor starter. When the condition is not met, however, the action, “energize the motor starter” is not taken. thus, when limit switch number two opens, the programmable controller responds by de-energizing the motor starter because that action is implied in the statement.

A program is made up of a number of similar statements. Typically, there is one statement for each output device on the machine. Each statement first lists the conditions that must be met and second, states the action to be taken.

Instructions

Each condition is represented by a specific instruction; therefore, each action is represented by a specific instruction. These instructions tell the CPU to do something with the information stored in the data table.

Some instructions tell the CPU to read what’s written in the image table. When the CPU is instructed to read from an image table, it examines a specific bit to see if a certain I/O device is on or off.

Other instructions tell the CPU to write information into the image table. When the CPU is instructed to write into the output image table, it writes a one or a zero into a specific bit. The corresponding output device will turn on or off as a result.

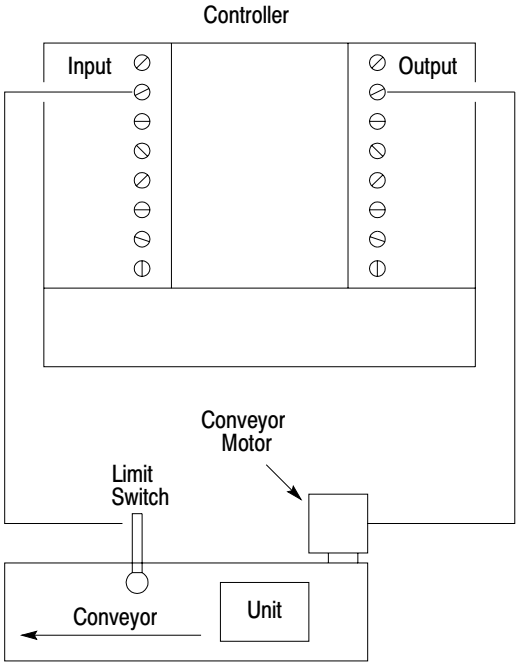
PC Control Sequence

Let’s look at a simple example to see the sequence of events that take place in controlling a machine with a programmable controller (Figure 2.4). Suppose you are making a unit. This unit would be carried to the work area by the motor driven conveyor. The limit switch will detect when the part has arrived at the work area. When that happens, we want the conveyor to de-energize so work can be done on the part.

Notice how the limit switch and motor are wired to the programmable controller. The limit switch, wired to terminal 02, is normally-closed. The arriving part will open the switch. Therefore, the program statement controlling the conveyor motor must read, “If there is voltage at input terminal 02 (limit switch), then energize output terminal 02 (conveyer motor).” The conveyor motor is wired to output terminal 02.

NOTE: Figure 2.4 is for demonstration purposes only. We do not label associated wiring, a motor starter, or an emergency stop button.

Figure 2.4
A Simplified Example of a Machine with a Programmable Controller



Since the limit switch is wired normally-closed, the conveyor motor will run until the arriving part opens the switch. At that time, the condition for energizing the motor will no longer be met. Therefore, the motor will be de-energized.

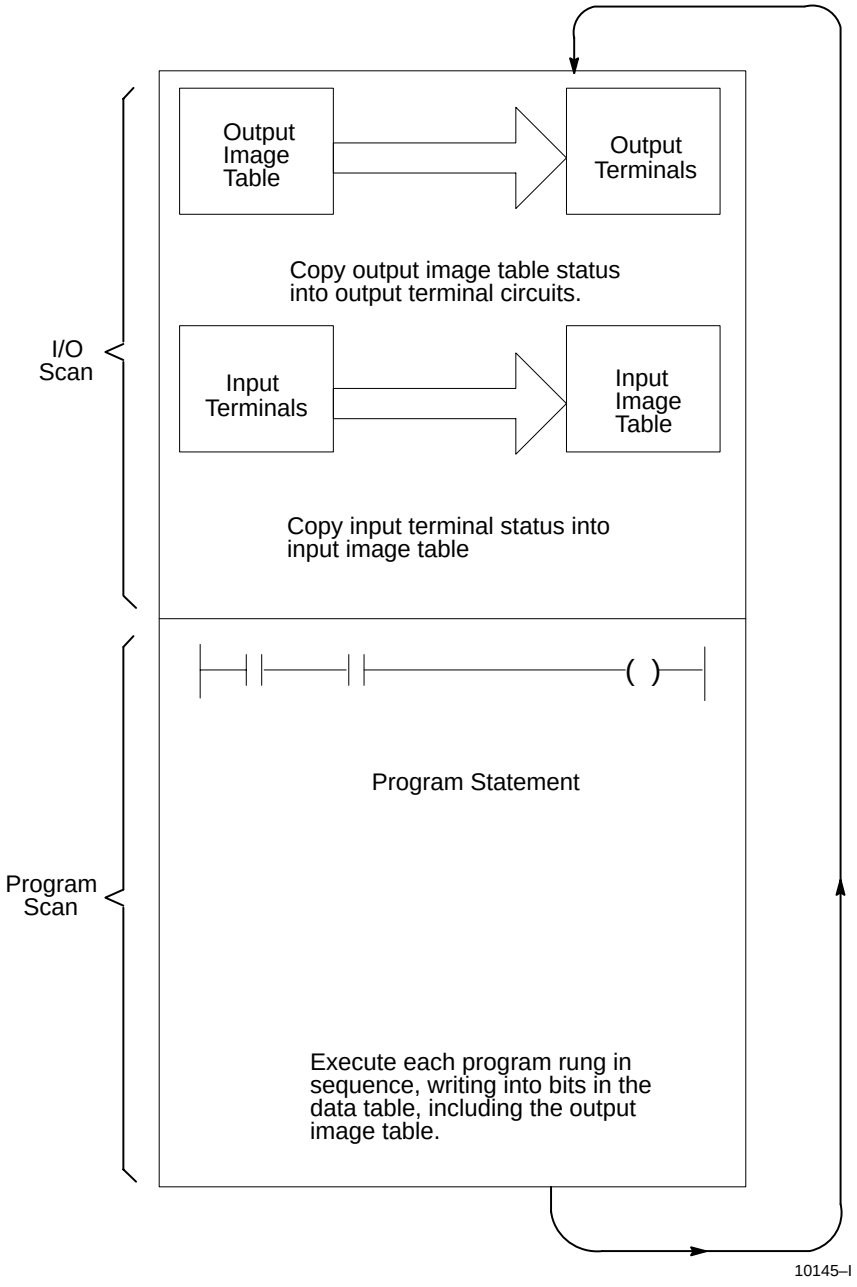
When the condition is met, we say it is true. When the condition is not met, we say it is false. There may be more than one condition that has to be met before an action can be executed. When all the conditions in that set of conditions are true, the action is executed and we say the statement is true. When one or more of the conditions are false, the action is not executed and we say the statement is false.

Scan Sequence

Upon power up, the CPU begins the scan sequence (Figure 2.5) with the I/O scan. During the I/O scan, data from input modules is transferred to the input image table. Data from output image table is transferred to the output modules.

- Next the CPU scans the program. It does this statement by statement. Each statement is scanned in this way:
  - First, for each condition, the CPU checks, or “reads,” the image table to see if the condition has been met.
  - Second, if the set of conditions has been met, the CPU writes a one into the bit location in the output image table corresponding to the output terminal to be energized. On the other hand, if the set of conditions has not been met, the CPU writes a zero into that bit location, indicating that the output terminal should not be energized.

Figure 2.5
Scan Sequence



The program in this example has only one statement, “If there is no voltage at input terminal 02, then energize output terminal 02.” The condition, “If there is not voltage at input terminal 02,” is really an instruction to the CPU to examine bit 02 in the input image table to see if it is off. the action portion of our program, “then energize output terminal 02,” is really another instruction telling the CPU to turn on bit location 02 in the output image table if the condition has been met.

Our program could been written this way

If (Condition)	Then (Action)
Input bit 02 is off	Turn output bit 02 on

In our example, the CPU reads a 0 at input bit location 02 and knows that the condition has been met. The CPU then carries out the action instruction by writing a 1 into output bit location 02.

If there were more statements in the program, the CPU would continue in the same manner scanning each statement and executing each instruction until it reached the end of the program. Statement by statement, the CPU would first read specific image table bits to see if the proper set of conditions were met. Then, the CPU would respond by writing a 0 or a 1 into an output bit as directed by the program. After reading and executing all program statements, the CPU scan the output image table and energizes or de-energizes output terminals. The CPU then goes to the input modules to update the input image table.

Now the entire process is repeated. in fact, it's repeated over and over again, thousands of times a minute. Each time, the CPU starts by sensing the status of the input terminals during the input image table scan. if an input device has changed states since the last scan, the CPU will change the state of the corresponding bit to reflect the new state. Next, the CPU scans the program and sets or resets output bits. Finally, the CPU scans the output image table and orders each output terminal on or off according to the state of its corresponding bit in the output image table.

When our example begins the CPU is energizing output terminal 02 because output bit 02 is on.

When the part is conveyed to the work station, it trips the limit switch. The closed limit switch applies a voltage to input terminal 02. The CPU scans the input image table, senses this voltage, and responds by writing a 1 (on) into bit 02 in the input image.

The CPU then scans our program. Our program states that "if (condition) input bit 02 is off, then (action) turn output 02 on." The CPU examines input image table bit 02 and discovers that input bit 02 is on. Since the condition is not true, the CPU writes a 0 (off) into output image table bit 02.

Finally, when the CPU next scans the output image table, it sees the zero in output bit 02 and responds by de-energizing output terminal 02. The conveyor will stop after the part closes the limit switch.

Conclusion

Now that you have read the basic concepts to our programmable controllers, you can proceed to chapter 3. Chapter 3 explains the specific hardware features of the Mini-PLC-2/15 system.

Chapter 3

Mini-PLC-2/15 System: An Overview

Objectives

This chapter focuses on the complete Mini-PLC-2/15 system. In this chapter you will read about:

- Major components
- General features
- Hardware features
- Optional features

This chapter is a synopsis of our Mini-PLC-2/15 Assembly and Installation Manual, publication 1772-803.

Major Components

A complete programmable controller system consists of the following major components:

- A series B Mini-PLC-2/15 processor module
- An I/O chassis
- A system power supply
- I/O modules (up to 16 modules)
- Industrial Terminal System (cat. no. 1770-T3)

General Features

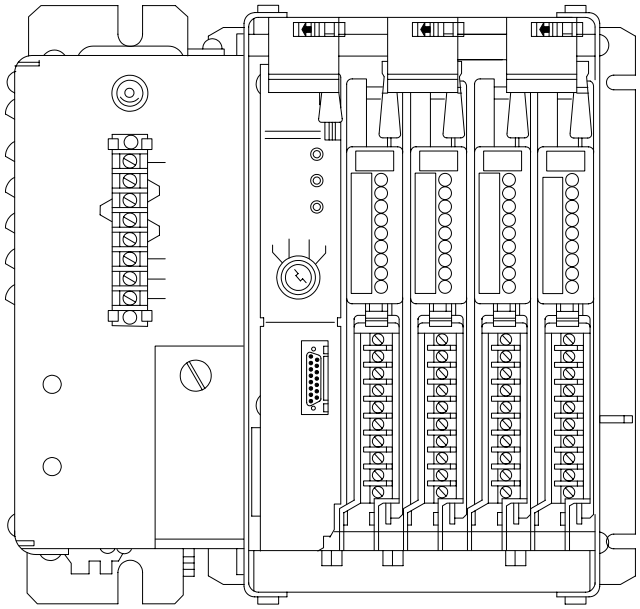
The system provides the following features:

- 2K CMOS RAM memory
- 488 timers and counters
- 1920 word capacity data table
- Ladder diagram and functional block instruction set
- Four function arithmetic capabilities
- EAF instruction capability
- Remote mode selection
- On-line programming
- Block transfer capability
- 70 message storage
- Data highway compatibility

Hardware Features

This system comes equipped wit the following hardware features: Refer to Figure 3.1 for their location.

Figure 3.1
Mini-PLC-2/15 Programmable Controller



18404

System Status Indicators

These indicators are located on the front panel of the controller and power supply. They indicate error conditions and are labeled as:

Processor

Indicator: red

State: On indicates that the processor is unable to scan the program and I/O.

Off indicates no error.

Response: The processor will cease to operate and outputs will either be disabled or held in their last state in accordance with the I/O chassis switch setting.

Memory

Indicator: Red

State: On indicates errors in either:

- Memory data
- Parity
- EPROM

Off indicates no error.

Blinking indicates an error in EPROM.

Response: The processor will halt all operations and disable all outputs.

RUN

Indicator: Green

State: On indicates that the output devices respond to your program when the processor is in run or run program modes.

Off indicates that the processor is in the program or test modes, either wit the keyswitch or using the remote mode select function.

Response: On- The processor will begin operations.

Off - When the processor is in the test mode, the program is executed while the outputs are disabled. When the processor is in the program mode and all outputs are off, then the program will not execute.

BATTERY LOW

Indicator: Red

State: On and Off blinking indicates a low battery.

Response: The battery low bit, 02700, will cycle on and off when a battery low voltage condition is detected (the processor can be in any mode). The battery will continue to provide memory backup for about one week after the indicator begins to flash.

DC ON

Indicator: Red

State: On indicates that 5.1V DC is present and within the required tolerances.

Response: On - The processor will begin operations. Off - The processor will not begin operations.

Mode Select Switch

This is keylock switch located on the front of the processor module. This switch has four positions to indicate the processor's mode of operation:

PROG

Function: You can enter or edit program instructions.

Response: All output are disabled. Program instructions are not executed.

TEST

Function: You can test your program without enabling outputs.

Response: All outputs are disabled. Program instructions are executed.

RUN

Function: Your program is continuously being scanned and executed.

Response: Programmed instructions control the outputs. Program changes can not be made.

RUN PROG (remote mode selection)

Function: Lets you select the desired mode without having to turn the keyswitch.

Advantage: When the keyswitch is in the RUN/PROG position, you can enter instructions from the industrial terminal that will place the series B processor into any one of the remote modes of operation:

- Remote Program - identical to the switch-selected program mode. The program scan and I/O scan will be halted. All outputs are disabled. Going into this mode from remote run/program will reset an I/O fault and clear a memory parity error.
- Remote Test - identical to the switch-selected test mode. The program instructions are executed, but all outputs are disabled.
- Remote Run/Program - identical to the switch-selected run mode. Whenever the keyswitch is turned to the RUN/PROG position the processor automatically enters this mode.
- On-Line Data Change - similar to the run mode, except that you can change data table values associated with instructions.
- On-Line Programming - allows you to make program changes while the program is running and controlling the outputs.

WARNING: Do not use the on-line programming feature of the Mini PLC-2/15 when the 1770-T3 industrial terminal is connected to the processor through a series A Communications Adapter Module (cat. no. 1771-KA). Unpredictable machine operation could result and cause damage to your equipment, and/or injury to your personnel.

Key Sequence:

- Remote program mode - [SEARCH] [5] [9] [2]
- Remote test mode - [SEARCH] [5] [9] [1]
- Remote run/program mode - [SEARCH] [5] [9] [0]
- On-line data change - [SEARCH] [5] [1]
- On-line programming - [SEARCH] [5] [2]

Power Supply

Purpose: Provides regulated 5.1V DC power to the processor and I/O modules.

Provides 5V DC power to the memory circuitry of the processor module.

Hardware: System Power Supply Module (ct. no. 1771-P1)

Function: Monitors the incoming AC voltage for the below levels:

- 98 to 132V AC for 120V AC operations
- 196 to 250V AC for 220 or 240V AC operations

Power Cable

Purpose: Connects the system power supply module and battery pack to the I/O chassis.

Hardware:

- Cat. no. 1771-CL I/O Power Cable (1 ft/30.5cm)
- Cat. no. 1771-CM I/O Power Cable (5 ft/1.5m)

AC Input Fuse

Purpose: Guards against overcurrent conditions on the AC input line.

Sizes:

- 1.0 amp fuse for 120V operations
- 0.5 amp fuse for 220 or 240V operations

Terminal Strip

Purpose: Provides wire connections for the power supply module.

Hardware: Terminals L1 and L2 label the AC input connections.

Battery Backup

Purpose: Provides battery backup power for the processor's memory.

Power Supply: Two alkaline D size battery cells or one D size lithium battery cell.

Function: Guards against the loss of memory if the:

- AC power line fails
- Power supply fails

Transport Cable

Purpose: Allows the controller to be moved from the I/O chassis without memory loss.

Function: It electrically connects the battery pack of the system power supply to the processor module for transporting.

Hardware: Cat. no. 1772-CD transport cable (2 ft/6 cm)

> **CAUTION:** Do not remove the processor module or install the processor module when the system's power is on. This could alter memory content and you must re-enter your program.

Switch Group Assembly

Purpose: Determines output response to a malfunction detected by the controller.

Location: Left side of the I/O chassis backplane.

Quantity: 8 per switch group assembly. Switches 2-8 are not used with this controller.

Response:

- On indicates that the outputs are left in their last state when a fault is detected. machine operation can continue after fault detection.

- Off indicates that the outputs are de-energized when a fault is detected.

WARNING: Switch number 1 should be set to OFF for most applications. This allows the processor to turn controlled devices OFF when a fault is detected. If this switch is set ON, machine operation can continue after fault detection and damage to equipment and/or injury to personnel could result.

NOTE: Use the tip of a ballpoint pen to set the switch.

Optional Features

Data Highway Compatibilities

Purpose: To provide communication between two or more processors or other computerized equipment.

Connections: With the optional Communication Adapter Module (cat. no. 1771-KA) the processors can be connected to the Allen-Bradley Data Highway System.

WARNING: Do not use the on-line programming feature when the industrial terminal is connected to a series A Communication Adapter Module (cat. no. 1771-KA). Unpredictable machine operation could result and cause damage to your equipment and/or injury to your personnel.

EAF Instructions

Purpose: Provides additional specific application instructions.

Hardware: Optional EAF EPROMs are available through your local Allen-Bradley Distributor or Sales Representative.

Function: Each EAF EPROM provides its own unique set of instructions for your application needs. Some instructions include:

AF1 EPROM - Advance Math

- Addition
- Subtraction
- Multiplication
- Division
- Square Root
- Average
- Standard Deviation
- BCD to binary conversion
- Binary to BCD conversion

AF2 EPROM - Process Functions

- Square root
- Integrator
- BCD to binary conversion with scaling
- Binary to BCD conversion with scaling

AF3 EPROM - File Diagnostic Instructions

- File Search
- File Diagnostic

AF4 EPROM - Log, Powers, Trigonometry

- Addition
- Subtraction
- Multiplication
- Division
- Square root
- BCD to binary conversion
- Binary to BCD conversion
- Log base 10
- Natural log
- $y +^{x}$
- $e +^{x}$
- 1/x
- Sine x
- Cosine x

NOTE: Refer to individual product data publications for your needs by contacting your local Allen-Bradley Distributor or Sales Representative.

Industrial Terminal

Purpose: To program your controller you need the Industrial Terminal System series B (cat. no. 1770-T3).

Function: With your industrial terminal you can:

- Enter
- Monitor
- Edit
- Troubleshoot

your program.

Also, you can interface with the processor by:

- Generating reports
- Interfacing peripheral devices

Keyboard

Function: The detachable keyboard houses PROM memory, a sealed touchpad, and a keytop overlay.

There are three keytop overlays:

- PLC-2 family: for use with any PLC-2 family processor.
- PLC: for use with any PLC family processor.
- Alphanumeric: for alphanumeric characters and graphic characters generation.

In the Quick Reference section there is a list of commands with their description to aid you in your programming functions.

Peripheral Equipment

Purpose: Optional auxiliary hardware which serves as a support function to enable you to store or maintain your programs on a magnetic medium or in report form.

Description: There are peripheral devices available to you.

Examples are:

- Silent 700 Data Terminal
- Data Cartridge Recorder (cat. no. 1770-SB)
- HC High Speed Bidirectional Printer (cat. no. 1770-HC)

NOTE: For further information concerning our peripheral equipment contact your local Allen-Bradley Distributor or Sales Representative.

Chapter 4

Memory Organization

Objectives

This chapter describes:

- Hardware and its relationship to your program
- Memory and its components

In chapter 2 we described in general terms the processor's memory section. Now we want to give you detailed concepts of the memory's organization and its structure. Understanding these concepts will aid you in programming your processor.

Introduction

Before we explain memory organization and its structure, there are some vocabulary definitions that will clarify this chapter:

Bit: the smallest unit of information that memory is capable of retaining.

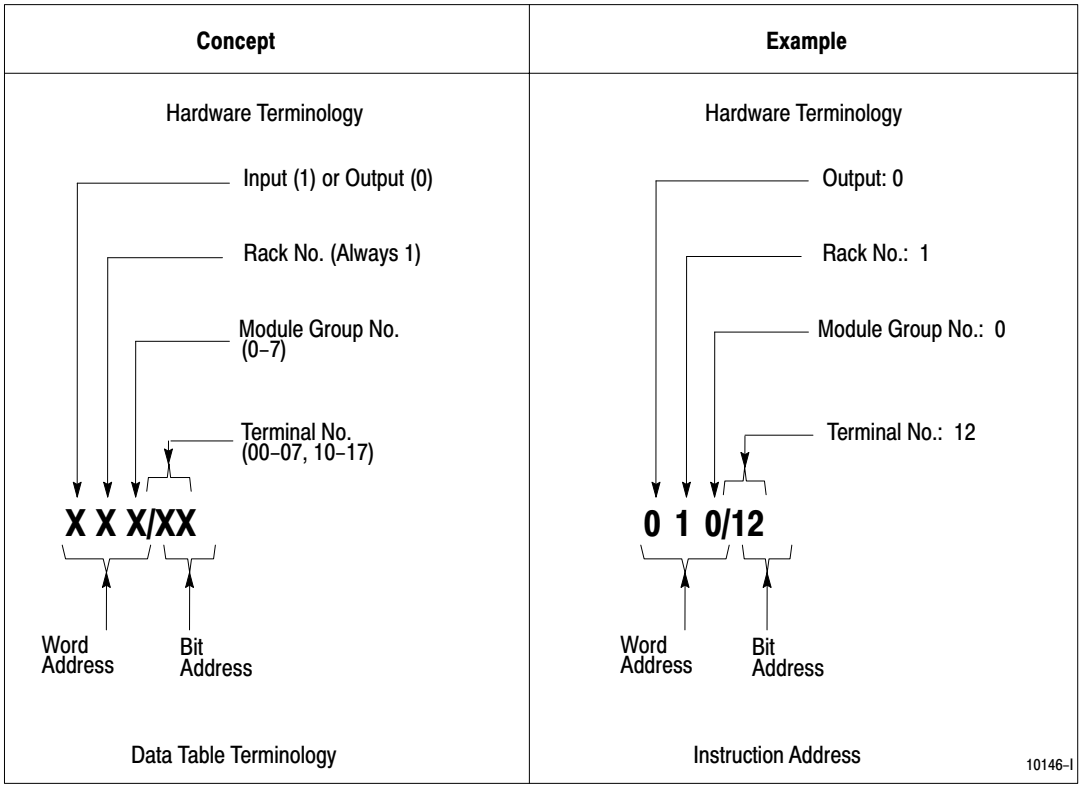
Byte: a group of 8 bits.

Word: a group of 16 bits.

Hardware vs. Your Program

The chart below and Figure 4.1 represent how the hardware of your system relates to the input and output image tables. Understanding these two illustrations will further your programming abilities.

Figure 4.1
Word Address Equals Memory Bits



Hardware	vs.	Your Program
I/O Terminal		Bit
Module Group		Word
Module Slot		Byte
One Rack		Eight Words
If the terminal has voltage (on state)		A specific bit will be on and a 1 will be written in memory
If the terminal has no voltage (off state)		A specific bit will be off and a 0 will be written in memory.

Now we will show you how to calculate the input and output image tables' areas and how these values compare with the hardware of your system.

Remember: 1 rack - 8 words
You can only have one rack in this system.

Therefore: 8 words/rack x 16 bits - 128 I/O

Conclusion: 128 I/O is the combined amount of usable bits utilized in the input image table and/or the output image table.

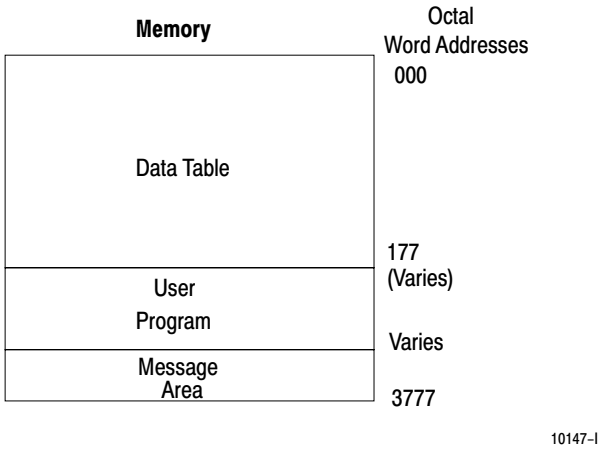
Memory

Memory is divided into three major sections: data table, user program, and a message storage area. These areas store input status, output status, your program instructions, and messages.

Figure 4.2 shows these areas with their corresponding octal addresses. We will describe these areas in detail so you will gain programming flexibility using your system.

NOTE: Octal is referred to as a base eight numbering system. It is defined in the glossary.

Figure 4.2
The Areas of Memory



Data Table

When we ship your processor, Allen-Bradley sets the memory for specific addresses. We call this type of data table organization, **factory configured**. Figure 4.3 shows memory structure with a factory configured data table. When we explain specific concepts about the different areas of memory, we will refer to a factory configured data table.

The data table area is a major part of memory. it is divided into six sections which includes the input and output image tables. (These two areas were described in chapter 2). The processor controls and utilizes words stored in the data table. The input devices coupled with the control logic from your program determines the status of the output devices. Input devices are limit switches, pushbutton switches, pressure switches, etc...Output devices are solenoids, motor starters, alarms, etc... Transfer of input data from input devices and the

transfer of output data to output devices occur during I/O scan. If the status of the output instruction changes in the program then the on/off status of the output devices update during the I/O scan to reflect the change.

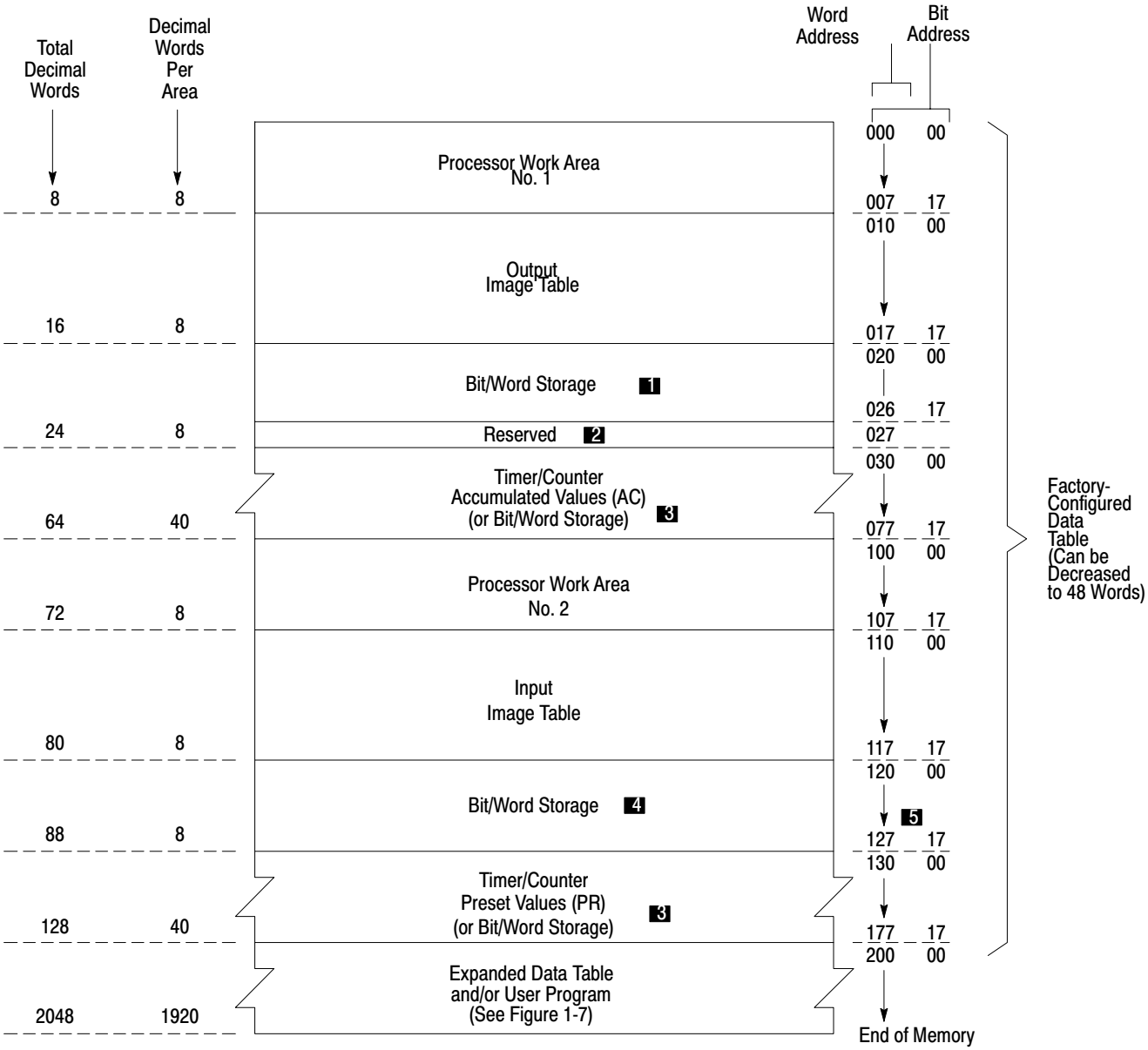
To utilize the data table to its fullest capacity certain facts must be understood:

- The processor automatically reserves the first 128 words in the memory for the data table.
- You can increase the data table size in two word increments up to 256 words. Then you can increase in blocks of 128 words.
- When the data table is set to 256 words, you can program up to 104 timer/counter instructions. These instructions are explained in chapter 5 section C.
- The data table can be changed in size from 48 words to 1,920 words using the industrial terminal.

Adjusted Data Table

You can adjust the data table size from the factory adjusted size of 128 words. This type of action is called **reconfiguration**. Using the 1770-T3 industrial terminal the data table can be adjusted in size from 48 words to 1920 words. Expanding the data table provides additional timers/counters and space for files (see chapter 5 for timers/counters and chapter 6 for file information), but your program storage and message areas will be reduced.

Figure 4.3
Data Table Organization, Factory Configured



1 May not be used for accumulated values.

2 Not available for bit/word storage. Bits in this word are used by the processor for battery low condition, message generation, EPROM transfer and data highway.

3 Unused timer/counter memory words can reduce data table size and increase user program area.

4 May not be used for preset values.

5 Do not use word 127 for block transfer data storage.

10148–I

Data Table Areas

There are six areas (Figure 4.3) making up the data table. They are:

- Input image table
- Output image table
- Processor work areas (2)
- Timer/Counter accumulated values and internal storage
- Timer/Counter preset values and internal storage

Chapter 2 discusses the input and output image tables. We will now discuss the remaining areas. Keep in mind that we are referring to a factory configured data table.

Processor Word Areas

Purpose: The processor uses these 16 words (addresses 000-007 and 100-107) for its internal control functions.

Description: There are two processor work areas. They are located at addresses 000-007 and 100-107. You cannot access these memory locations. Their word addresses are not available for addressing.

NOTE: The term address is defined in chapter 5. Remember, all addresses are base eight values.

Accumulated Values and Internal Storage

Purpose: Stores accumulated values of timer or counter instructions. This area also stores data by words and/or bits from your program instructions. (Addresses 030-077).

Description: Each timer or counter instruction uses two words of memory. One word is stored in the accumulated value area, the other is the preset value area. The preset value is 100* above the accumulated value. Therefore, a timer/counter having an address of 030 automatically has its preset value stored at address 130.

Preset Values and Internal Storage

Purpose: Stores preset values (PR) of timer or counter instructions. This area also stores data by words and/or bits from your program (addresses 130- 177).

Description: The preset value is the number of timed intervals or events to be counted. When the accumulated value equal the preset value (AC = PR), a status bit is set and can be examined to turn on or off an output device.

User Program

This is the second major part of memory. It is divided into three areas:

- Main ladder diagram program
- Subroutine area
- Data highway instructions

The user program area begins at word address 200.

Main Ladder Diagram Program

Purpose: Your program is a group of ladder diagram and functional block instructions used to control an application.

NOTE: The term ladder diagram is defined in chapter 5.

Description: Refer to chapter 8 section A.

Subroutine Area

Purpose: Used to jump to a defined ladder diagram area. This will allow you to perform ladder diagram subroutines.

Description: Refer to chapter 6 section B.

Data Highway Instruction

Purpose: Allows you to link up to 64 different stations for data gathering. A station can be defined as either another processor, computer, or a communication device.

Description: Programmed in a special format. Refer to our Publication index (publication SD499) for the list of appropriate publications.

Message Storage

This is the third major part of memory. You are able to print out messages in hard copy form. You can store up to 70 messages using the 1770-T3 industrial terminal.

Message storage follows the end statement of your program and is limited by the number of unused words remaining in memory. Each word stores two message characters. A character is any alpha or numerical figure (this includes blank spaces).

Messages can be written to display current data table information such as the number of parts rejected in a production run for a particular time period. You can write your program to display messages when a pushbutton switch or industrial terminal key is activated.

Address 027 controls messages 1-6. You designate control words which store your messages in groups of 8. Your control words must be arranged in consecutive order.

Report generation is a function of your message control words. Reserve bit addresses 02710 thru 02717 for this automatic report generation function to determine status of this function. These bit addresses should not be used for

any other functions if you want to achieve maximum flexibility within your program.

When you enter the report generation message [M][S][,][0][RETURN] the terminal displays the prompt: MESSAGE CONTROL WORDS (Y DIGITS REQUIRED): where "Y" is the required 3 or 4 digits of a word address for the selected data table size. You must enter the beginning word address of the message control word file. The Industrial Terminal then calculates and displays the ending address. You can locate the message control word file anywhere in the data table **except** in Processor work areas and in the Input Image table (i.e., do a SEARCH 50 to display the number of racks). When using EPROMs, Memory Write Protect is active: The message control word file must be placed in the areas of Data Table4 which can be changed (010_8- 177_8). Once you choose the start address, the Industrial Terminal displays a table which shows the message numbers associated with each message control word.

NOTE: You must verify that the message control word location does not coincide with a block transfer location or a timer or counter preset location.

Chapter 5

Fundamental Instruction Set

Programming Logic: Objectives

This chapter describes fundamental programming and editing techniques common to the controller.

In this chapter you will read sections A through E concerning:

- Programming Logic
- Relay Type Instructions
- Timer and Counter Instructions
- Data Manipulation Instructions
- Arithmetic Instructions

NOTE: Refer to the operations section of this manual for example instructions concerning chapters 5 and 6.

Section A
Programming Logic

Introduction

In this section will you read about the instructions needed to write a program, and how to define the needed conditions before the action takes place.

A program is a list of instructions that guides the controller. These instructions can examine or change the status of bits in the memory of the controller. The status of these bits determines the operation of your output devices.

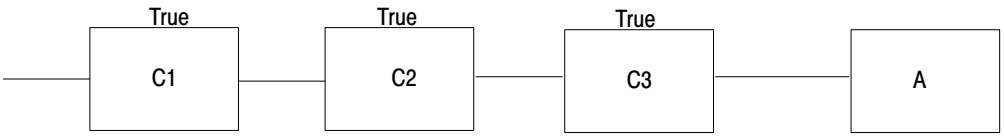
When you write a program you specify the things you want done in your application and the conditions that must be met before those things are done. For example, if you want a solenoid energized when a limit switch is closed, you would specify:

Condition: If limit switch is closed

Action: Energize solenoid

Programming logic differs from relay logic in an important way. Programming logic is only concerned with whether or not conditions have been met. These conditions may be open or closed input or output devices. We must have a continuous or unbroken path of true logic conditions for an action to be taken. The number of conditions is not important. There can be none, one, or many conditions preceding an output action.

Perhaps an example might make this more clear:

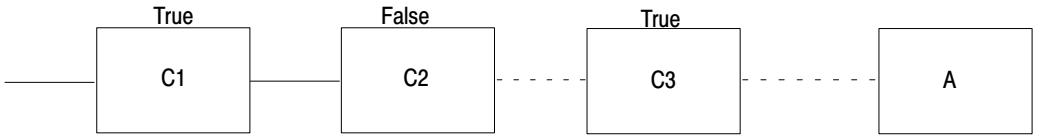


Here, a **series** of conditions, (C1, C2, C3) must be true before an action is performed.

C1 =	Input switch 1. When the switch is on, this condition is true. This switch turns on a conveyer belt.
C2 =	Input sensor 1. When the sensor is off, this condition is true. This sensor detects if the temperature in the factory is below 40oC.
C3 =	Input sensor 2. When the sensor is on, this condition is true. This sensor detects the presence of a part of the conveyer belt.
A =	The part will be drilled
_ =	The path of conditions is continuous, that is, all conditions are true.

When C1, C2, and C3 are true, then a continuous path is made to a particular action. In this case, the continuous path causes the part to be drilled.

When the path of conditions is continuous, we say that the rung is true. When the path of conditions is not continuous, we say the rung is false.



Here the path of conditions is not continuous because condition 2 is false. Therefore, the action is not performed. We say the rung is false.

Hardware Review

Recall that input and output devices are connected, via field wiring, to the controller's input and output terminals. Furthermore, each of these terminals correspond to a memory bit that reflects the state of that device.

Set vs. Reset

As a review, if the device goes on, then we say the corresponding bit in memory is set to a 1. If the device goes off, we say the corresponding bit in memory is

reset to a 0. (From this point on, set means turned to the on-condition or 1. Reset means turned to the off-condition or 0.)

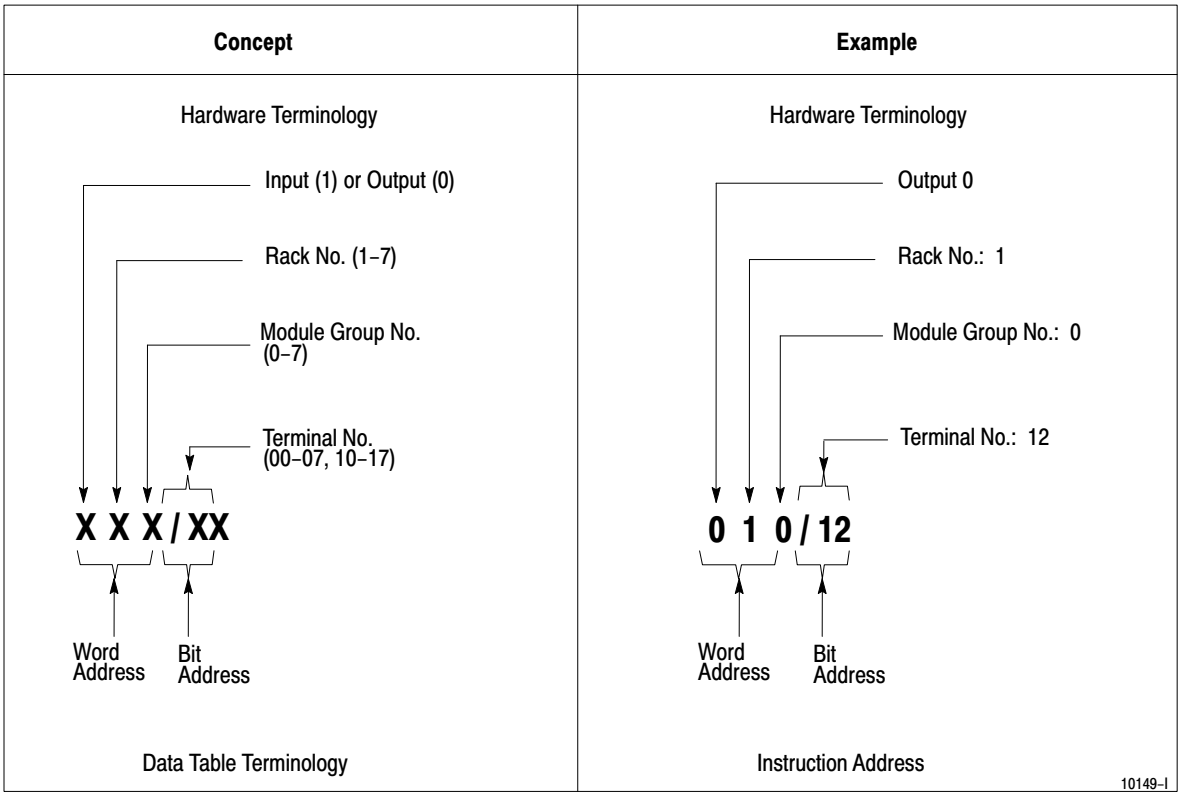
If the device :	The a bit in memory is:
on	set
off	reset

Address

Recall that the controller scans the status of your inputs and controls your output devices. It does not go to the input or output terminals to see if outputs are on or off. Rather, it checks the status of the input and output devices by scanning corresponding bits in memory. The controller uses addresses to refer to memory bits.

Each input and output bit has a five-character address (Figure 5.1).

Figure 5.1
Instruction Address Terminology



Reading from left to right:

- The first number denotes the type of word corresponding to a module:
  - 0 output
  - 1 input

NOTE: Remember, there is only 1 rack in a Mini-PLC-2/15 system.

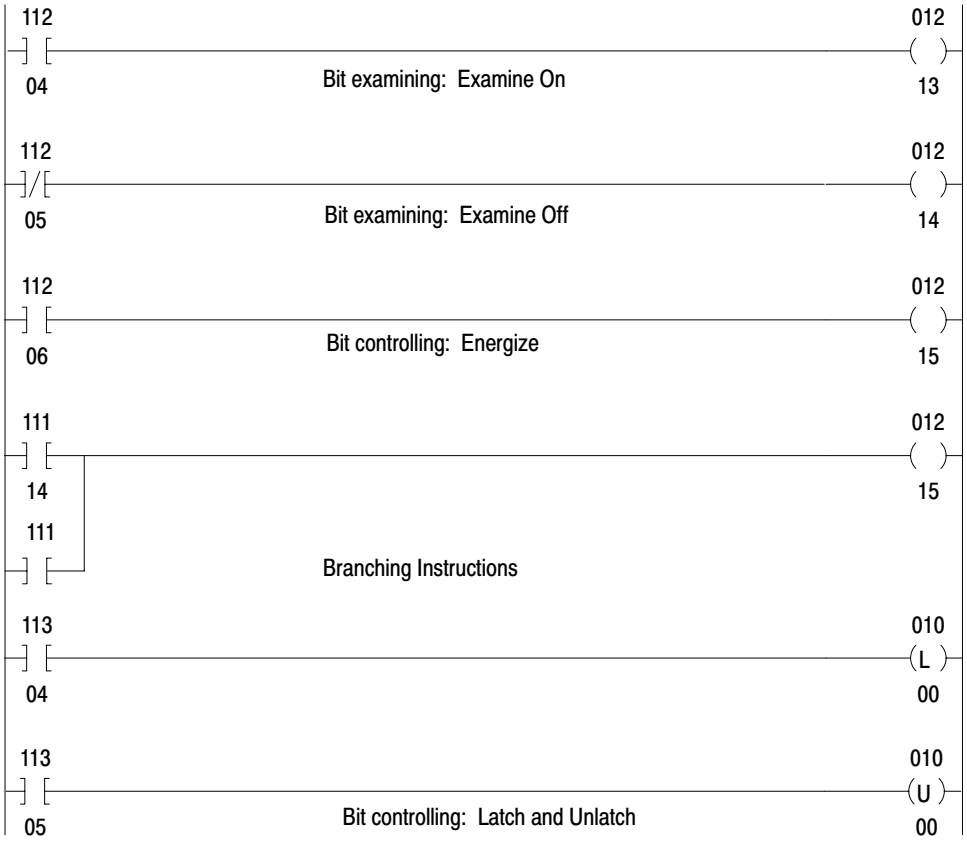
- The second number denotes an I/O rack ad it always is a 1.
- The third number denotes a module group. This number will range from 0-7.
- The fourth and fifth numbers denote a terminal designation:
  - 00-07 left slot of the module group
  - 10-17 right slot of the module group

Section B
Relay Type Instructions

Introduction

You can use six relay type instructions to write a program (Figure 5.2). We will refer to these instructions as relay type instructions because of their similarities to relay symbols. These instructions react to changes of input to change certain electric control circuits. There are three kinds of relay type instructions: bit examining, bit controlling, and branch instructions.

Figure 5.2
Relay Type Instructions



Bit Examining Instructions

Exami ne On

Symbol: -| |-

Purpose: This instruction tells the controller to examine a bit at a specified memory location.

Syntax: Programmed at the condition side of the rung.

Function: Determines the instruction condition. The instruction condition becomes:

True

- If the controller detects that a bit in memory is set.

False

- If the controller detects that a bit in memory is reset.

Examine Off

Symbol: -|/|-

Purpose: This instruction tells the controller to examine a bit at a specified memory location.

Syntax: Programmed at the condition side of the rung.

Function: Determines the instruction condition. The instruction condition becomes:

True

- If the controller detects that a bit in memory is reset.

False

- If the controller detects that a bit in memory is set.

Bit Controlling Instructions

Energize

Symbol: -( )-

Purpose: This instruction tells the controller to set or reset a specified memory bit.

Syntax: Programmed at the output side of the rung.

Function: Controls a specific bit based on the rung condition. when the preceding rung conditions are:

True

- The energize instruction set a specified bit.

False

- The energize instruction resets a specified bit.

Latch

Symbol: -(L)-

Purpose: This instruction tells the controller to set a specified memory bit. It is used with the unlatch instruction.

Syntax: Programmed at the output side of the rung. This is a retentive instruction. Retentive means that once the rung condition goes false, the latch bit remains set until reset by an unlatch instruction.

Function: Controls a specific bit based on the rung condition. When the rung conditions are:

True

- The latch instruction sets a specified bit.

False

- No action is taken.

NOTE: If power is lost, and back-up battery power is maintained, all latch bits will remain on. when all power is off, all outputs associated with the latch bits will be off.

Unlatch

Symbol: -(U)-

Purpose: This instruction tells the controller to reset a specified bit in memory. It is used with the latch instruction.

Syntax: Programmed at the output side of the rung; used with the latch instruction. This is a retentive instruction.

Function: Controls a specific bit based on the rung condition. When the rung conditions are:

True

- The unlatch instruction resets the specified bit.

False

- No action is taken.

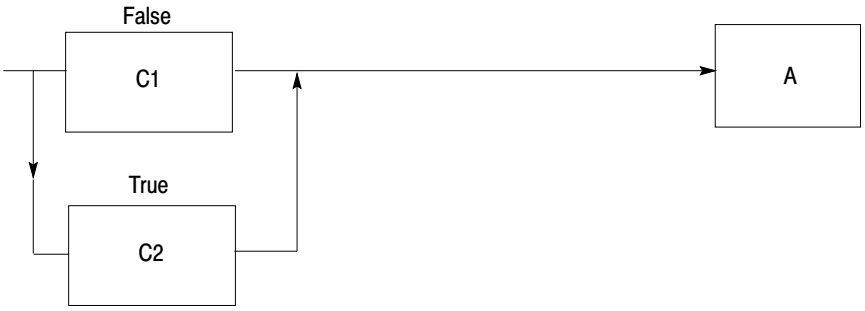
NOTE: The conditions for the unlatch instruction must be different than the conditions that precede the latch instruction.

Latch and unlatch instructions should be used in pairs with each other. They should also be used to control a store bit which controls an actual output.

Branching Instructions

So far you've only looked at rungs having a series of instructions. You use branching instructions when you want several parallel sets of conditions to make an output action possible. A program with branching says, "If this set of conditions is true, or if that set of conditions is true, perform the following action." Branching allows two or more paths to reach the same output destination.

The rung below uses parallel logic:



Here two conditions are parallel. As long as one of the conditions (C1 **or** C2) is true, a continuous path to the action exists. Therefore, the action is performed.

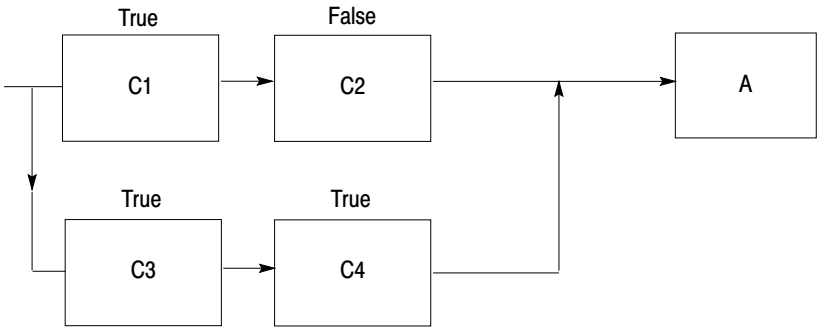


Figure 5.2 shows a program rung with branching, as it would appear by the 1770-T3 terminal display. You create a branch by using two different branch instructions. These are the branch start and branch end instructions.

Nested Branching

The rung below shows a nested branch.

Creating nested branches is not possible because the branch end instruction completes a branch group. But the above rung shows a single branch group with two branch end instructions. Above, the examine-on instruction with the address 11011 is actually a branch group within a branch group.

The rung below achieves the same result, but avoids nested branching:

Section C
Timers and Counters

Introduction

Timer and counter instructions are output instructions internal to the controller. They provide many of the capabilities available with timing relays and solid state timing/counting devices. Usually conditioned by examine instructions, timers and counters keep track of timed intervals or counted events according to the logic continuity of the rung. You can program a maximum of 488 internal timers and/or counters.

Timer/Counter Theory

Each timer or counter instruction has two 3-digit values. Each value requires one word of data table memory. These 3-digit values are:

- Accumulated Value (AC)

  Storage: Begins at word address 030.

  Function: Timers - number of elapsed timed intervals.
  Counters - number of counted events.
  Both - upper 4 its of accumulated word (14-17) are the status bits.

- Preset Value (PR)

  Storage: Always 100_8 words greater than its corresponding AC value.

  Function: Denotes the number of timed intervals or events to be counted. When the accumulated value equals the preset value, AC=PR, a status bit is set and can be examined to turn an output device on or off.

Timer Instructions

Introduction

A timer counts elapsed time-base intervals and store this count in the accumulated value word. Timer instructions have three time bases: 1.0 second, 0.1 second, or 0.01 second.

Two bits in the accumulated value word are status bits:

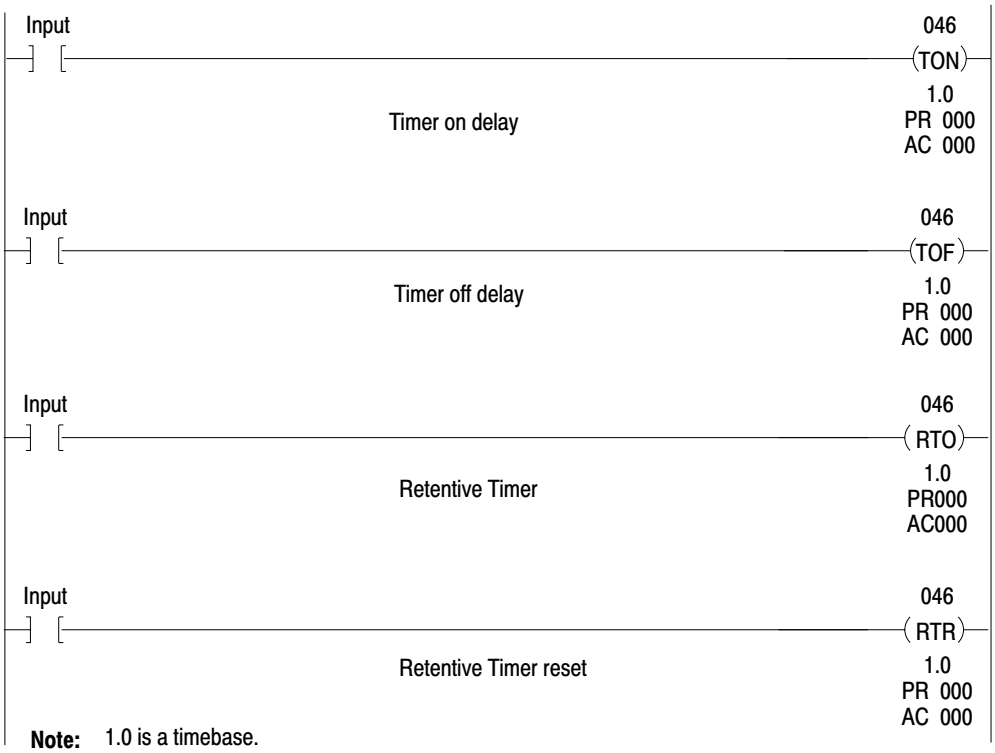
- Bit 15 is the timed bit. It is set either on or off wen the timer has timed out. The settings on or off depend on the type of timer instruction used.
- Bit 17 is the enable bit. It is set when rung conditions are true and is reset when rung conditions are false.

There are four types of timer instructions available with the controller:

- Timer on-delay
- Timer off-delay
- Retentive timer on-delay
- Retentive timer reset

We will look at these timers in detail. Chapter 9 illustrates programming applications for timers and counters. Figure 5.3 shows the timer instructions and Figure 5.4 shows the counter instructions.

Figure 5.3
Timer Instructions with Their Default Values



Note: 1.0 is a timebase.

CAUTION: Allowances should be made for conditions which could be created by the use of the jump instruction. Jumped program rungs are not scanned by the processor so that input conditions are not examined and outputs that are controlled by these rungs remain in their last state. Timers and counters cease to function. Critical rungs should be reprogrammed outside the jumped section in the program zone.

Timer On-Delay Instruction

Symbol: -(TON)-

Purpose: Can be used to turn a device on or off once an interval is timed out.

Syntax: Programmed as an output instruction.

Function: When the rung condition becomes:

True

- Timer cycle begins
- Timer increments its AC value.
- Bit 15 is set when AC=PR and the timer stops timing.
- Bit 17 is set.

False

- Accumulated value resets to 000.
- Bits 15 and 17 are reset.

Timer Off Delay Instruction

Symbol: (TOF)-

Purpose: Can be used to turn a device on or off once an interval in timed out.

Syntax: Programmed as an output instruction.

Function: When the rung conditions becomes:

True

- Bit 15 is set.
- Bit 17 is set.
- Accumulated value resets to 000.

False

- Timer cycle begins.
- Timer increments its AC value.
- Bit 15 resets when the AC=PR and the timer stops timing.
- Bit 17 is reset.

Retentive Timer Instruction

Symbol: -(RTO)-

Purpose: Similar to the TON instruction. The AC value is retained through false rung conditions.

Syntax: Programmed as an output instruction.

Function: When the rung condition becomes:

True

- Timer begins counting time-base intervals.
- Bit 15 is set when AC=PR and the timer stops timing.
- Bit 17 is set.

False

- Accumulated value is retained.
- Bit 15 - no action is taken.
- Bit 17 is reset.

NOTE: The RTO instruction retains its AC value when the:

- Rung condition turns false.
- Mode select switch is changed to the PROG position.
- Power outage occurs and memory backup is maintained.

Retentive Timer Reset Instruction

Symbol: -(RTR)-

Purpose: Resets the accumulated value and timed bit of the retentive timer.

NOTE: Give this instruction the same word address as its corresponding RTO instruction.

Function: When the rung condition becomes:

True

- RTR instruction resets the accumulated value of the RTO instruction.
- Bits 15 and 17 and reset.

False

- No action is taken.

Counter Instructions

Introduction

A counter counts the number of events that occur and stores this count in its accumulated value word. An event is defined as a false-to-true transition. Counter instructions have no time base.

The upper four bits in the accumulated value (AC) word are status bits:

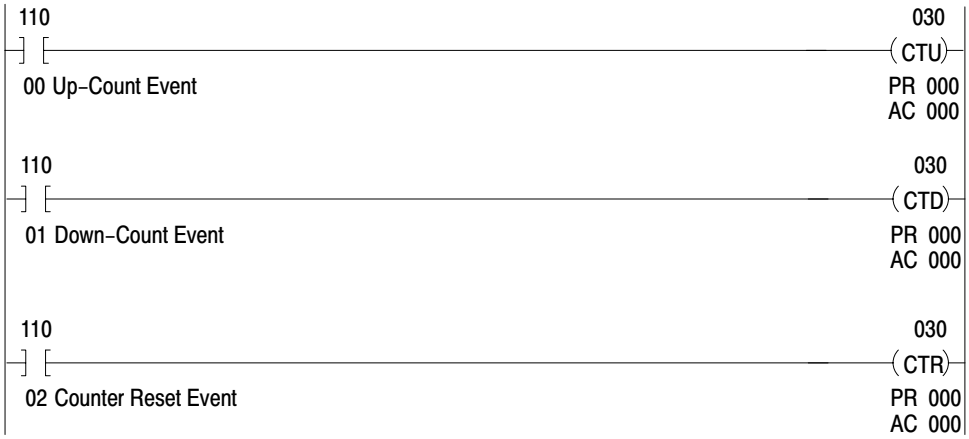
- Bit 14 - Overflow/underflow bit. it is set to one when the AC value of the CTU instruction exceeds 999 or when the AC value of the CTD instruction falls below 000.
- Bit 15 - Count complete bit. it is set to on when the AC value ≥PR value.
- Bit 16 - Enable bit for CTD instruction. It is set on when the rung condition is true.
- Bit 17 - Enable bit for CTU instruction. It is set on when the rung condition is true.

There are three types of counter instructions available with the controller:

- Up counter
- Down counter
- Counter reset

We will look at these counters in detail.

Figure 5.4
Counter Instructions with Their Default Values



Up Counter Instruction

Symbol: -(CTU)-

Purpose: Increments its accumulated value for each false-to-true transition of the rung condition.

Syntax: Programmed as an output instruction.

Function: When the rung condition becomes:

True

- Accumulated value increments by 1.
- Bit 14 is set on if the AC >999.
- Bit 15 is set on when AC ≥PR. Incrementing of the accumulated value can continue after the preset value is reached.
- Bit 17 is set and stays set until the rung goes false.

False

- Accumulated value is retained.
- Bit 14 - no action is taken.
- Bit 15 is retained if it was set.
- Bit 17 is reset.

The CTU retains its AC value when:

- You change the mode select keyswitch to the PROG position.
- The rung condition turns false.
- A power outage occurs and memory backup is maintained.

NOTE: Bit 14 of the accumulated value word is set when the accumulated value either overflows or underflows. When a down counter preset is reset to 000, the underflow bit 14 will not be set when the count goes below 0.

Down-Counter Instruction

Symbol: -(CTD)-

Purpose: Decrements its accumulated value from 999 for each false-to-true transition of the rung condition. This indicates an underflow condition.

Syntax: programmed as an output instruction.

Function: When the rung condition becomes:

True

- Accumulated value decrements by 1.
- Bit 14 is set when AC≤000.
- Bit 15 is reset when AC≤PR; counting can continue.
- Bit 16 is set and stays set until the rung goes false.

False

- Accumulated value is retained.
- Bit 14 is retained if it was set.
- Bit 15 is retained if it was set.
- Bit 16 is reset.

Counter Reset Instruction

Symbol: -(CTR)-

Purpose: Resets the up counter or down counter instructions' accumulated value and status bits to 0.

Syntax: Programmed as an output instruction. The AC and PR values are displayed.

Function: When the rung condition becomes:

True

- Accumulated value of the specified counter is reset to 000.
- Status bits (14, 15, 16, 17) are reset.

False

- No action is taken.

Section D
Data Manipulation Instructions

Introduction

In this section you will read how data is transferred or compared when it is stored in the data table.

To transfer or to compare stored data located in the data table use the following data manipulation instructions:

- Get
- Put
- Less Than
- Equal To
- Get Byte
- Limit Test

Transfer Instructions

Get

Symbol -|G|-

Purpose: Accesses 16 bits of data from one word location in the data table. It does not determine rung logic continuity.

Syntax: Programmed in the condition area of the ladder diagram rung (Figure 5.5). It can be located at the beginning a rung or with one or more conditions preceding it.

Function: Always accesses the word to which it is addressed. It displays a decimal number beneath the instruction. The lower 12 bits (bits 0-13) of the specified word contain the data.

Figure 5.5
Get and Put

Put

Symbol: -(PUT)-

Purpose: Receives 16 bits of data from the immediately preceding get instruction and stores the data at the specified data table word location. Used with a get instruction to form a data transfer rung.

Syntax: Programmed in the output side of the ladder diagram rung (Figure 5.5). This instruction can have the same address as other instructions in the program. It is always programmed with a get instruction.

Function: Transfers an image of the 16 bits of one data table word to another data table word when the rung is true.

NOTE: The put instruction acts only upon true rung conditions. There should be no instructions between the get and put instructions. Position all conditions before the get instruction.

Compare Instructions

Equal

Symbol: -|=|-

Purpose: Compares the data in your specified address with data stored at another address in memory. It determines the rung condition.

Syntax: Programmed after the get instruction in the condition side of the ladder diagram rung (Figure 5.6).

Function: The rung condition becomes:

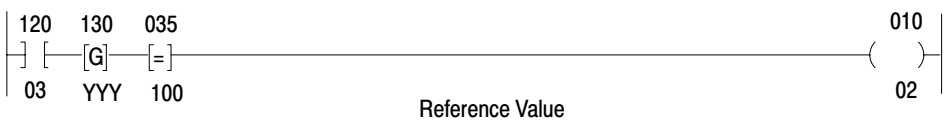
True

- If there is equality.

False

- If there is not equality.

Figure 5.6
Equal To Comparison



When YYY = 100, GET/EQU comparison is true and 01002 is energized.

Less Than

Symbol: -|<|-

Purpose: Compares the data in your specified address with the data stored at another address in memory. It determines the rung condition.

Syntax: Programmed after the get instruction in the condition side of the ladder diagram rung (Figure 5.7).

Function: The rung condition becomes:

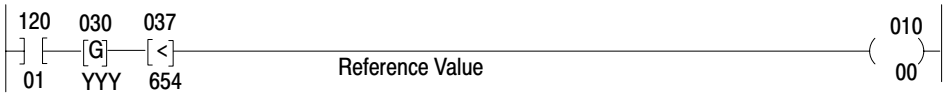
True

- If the get value is less than the reference values stored in the less than instruction.

False

- If the get value is not less than the less than value.

Figure 5.7
Less Than Comparison



When YYY < 654, GET/LES comparison is true and 01000 is energized.

Get Byte

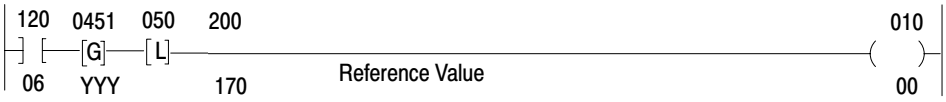
Symbol: -|B|-

Purpose: Accesses 1 byte (instead of 1 word) from one word location of the data table.

Syntax: Can be programmed with a limit test instruction located at the condition area of the ladder diagram (Figure 5.8). The data is shown in octal form.

Function: Used with a put instruction to transfer either the upper or lower byte to the lower byte of the put address.

Figure 5.8
Get Byte/Limit Test Comparison



When 170<YYY<200, comparison is true and 01005 is energized.

Limit Test

Symbol: -|L|-

Purpose: Checks to see if a byte value is between two reference byte values in the limit test instruction.

Syntax: Programmed with a get byte instruction located at the condition area of the ladder diagram (Figure 5.8).

NOTE: Do not place compare instructions between the get byte and limit test instruction. The get byte and limit test instructions only work with octal values.

Function: The rung condition becomes:

True

- If the specified byte value is between the two reference values.

False

- If the specified byte value is outside the reference values.

Get Byte/Put

Symbol: Figure 5.9

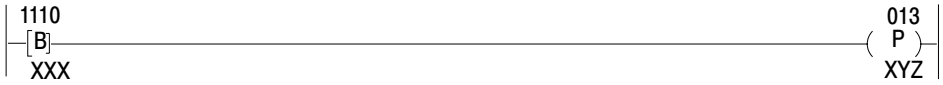
Purpose: Duplicates eight bits of data from the get byte instruction to the lower byte of the put instruction.

Function: The value in the get byte instruction is displayed in octal form. The value in the put instruction is displayed in hexadecimal form.

Syntax: Figure 5.9

NOTE: Do not use the upper byte of the put address for storage because it will be a random value.

Figure 5.9
Get Byte/Put Test Comparison



Section E
Arithmetic Instructions

Introduction

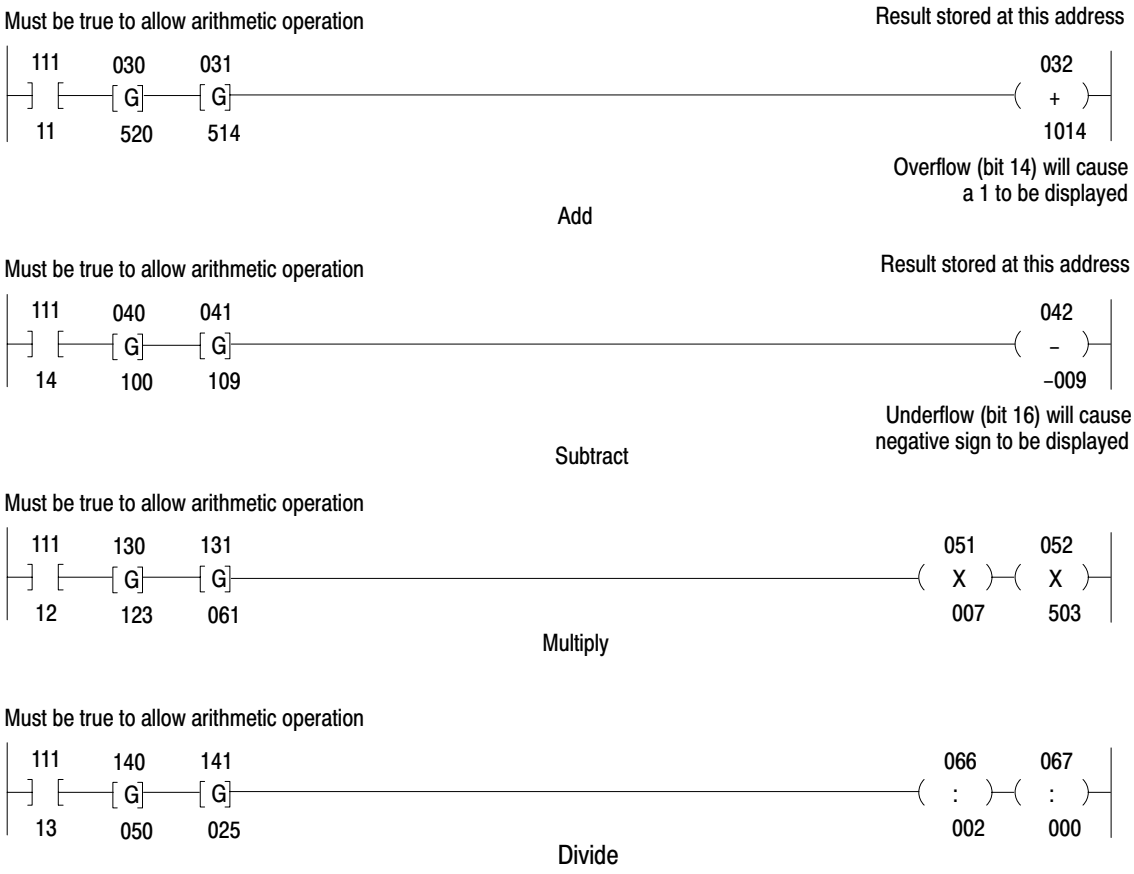
You can do 3-digit arithmetic operations using your controller. The basic operations used are:

- Addition
- Subtraction
- Multiplication
- Division

See Figure 5.10.

Additional functions are available by purchasing EAF PROM instructions through your local Allen-Bradley Distribution or Sales Representative.

Figure 5.10
Arithmetic Instructions



NOTE: The controller performs arithmetic and data manipulation operations with 3-digit BCD (binary coded decimal) values.

Addition

Symbol: -(+)-

Purpose: Reports the sum of two values stored in the get instruction words.

Syntax: Programmed in the output position of the ladder diagram rung. Your sum is stored in the add instruction word address.

Function: When the sum exceeds 999, the overflow bit (bit 14) in the add instruction word is set. When you are in the run, test, or run program mode, the overflow condition is displayed on the industrial terminal screen as a “1” preceding the sum.

NOTE: If an overflow value (4 digits) is used for subsequent comparisons or other arithmetic operations, inaccurate results could occur. Your processor

performs arithmetic and data manipulation operations only with 3-digit BCD values.

Subtraction

Symbol: -(-)-

Purpose: Reports the difference between two values stored in the get instruction words. The second get word value is subtracted from the first get word value.

Syntax: programmed in the output position of the ladder diagram rung. Your difference is stored in the subtract instruction word address.

Function: When the difference is a negative number, the underflow bit (bit 16) in the subtract instruction word is set. When you are in the run, test, or run program mode, the negative sign will appear on the industrial terminal screen preceding the difference.

NOTE: Use only positive values. If a negative BCD value is used for subsequent operation, inaccurate results could occur. The processor only compares, transfers and computes the absolute BCD value.

Multiplication

Symbol: -(x)-(x)-

Purpose: Reports the product of two values stored in the get instruction words.

Syntax: Programmed in the output position of the ladder diagram. Your product is stored in two multiplication instruction word addresses. If the product is less than 6 digits, leading zeros will appear in the product.

NOTE: For good documentation habits we recommend using consecutive word addresses.

Division

Symbol: -(:)-(:)-

Purpose: Reports the quotient of two values stored in the get instruction words.

Syntax: Programmed in the output position of the ladder diagram rung. your quotient is stored in two divide instruction word addresses.

NOTE: For good documentation habits we recommend using consecutive word addresses. Quotient is expressed as a decimal, accurate to 3 decimal places. Any remaining data is truncated. Although division by 0 is undefined mathematically, the division of a number including 0 : 0 will give the result of

000.000. This differs from the PLC-2/20 and PLC-2/30 controllers where 0 : 0 = 1.000.

Chapter 6

Advanced Instruction Set

Objectives

This chapter describes advanced programming techniques common to the controller. In this chapter you will read sections A through E concerning:

- Scan Theory
- Program Control Instructions
- Jump Instructions and Subroutine Programming
- Advance Data Manipulation
- Block Transfer Instructions

Section A
Scan Theory

Introduction

In this section you will read:

- Scan Function
- Scan Time

Scan Function

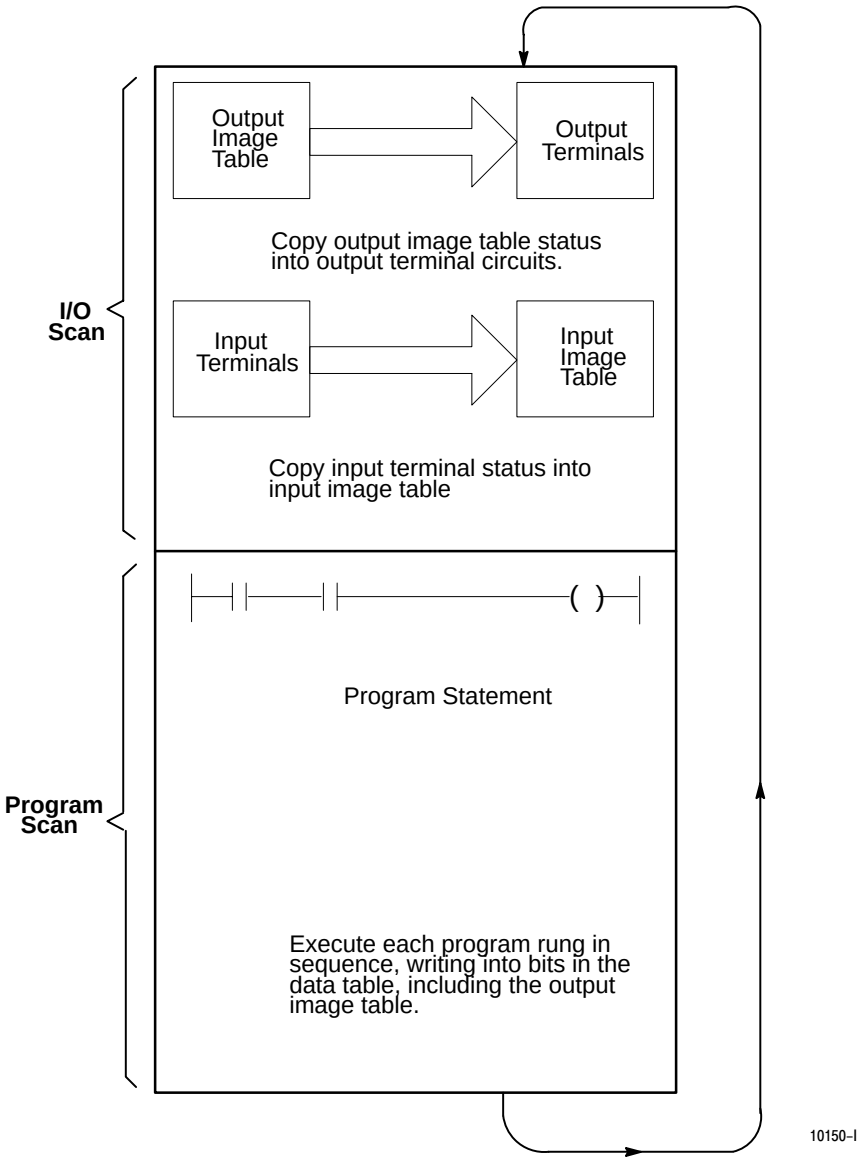
In order for the processor to implement your program, it must evaluate the action that it takes based on monitoring the status of input conditions. In addition, it must control the status of output devices in accordance with the program logic. Every instruction in your program requires an execution time. Execution times vary greatly depending upon the instruction, the amount of data to be operated on, and whether the instruction is true or false.

As a review from chapter 2, there are two types of scan functions (Figure 6.1):

- I/O scan
- Program scan

Upon power up, the CPU begins the scan sequence with the I/O scan. During the I/O scan, data from input modules is transferred to the input image table. Data from output image table is transferred to the output modules.

Figure 6.1
Scan Sequence



Next, the CPU scans the program. It does this statement by statement. Each statement is scanned in this way:

- First, for each condition, the CPU checks, or “reads,” the image table to see if the condition has been met.

- Second, if the set of conditions has been met, the CPU writes a one into the bit location in the output image table corresponding to the output terminal to be energized. On the other hand, if the set of conditions has not been met, the CPU writes a zero into the bit location, indicating that the output terminal should not be energized.

NOTE: When your processor is in the test mode, all outputs are not active. When your processor is in the run mode, all outputs are active.

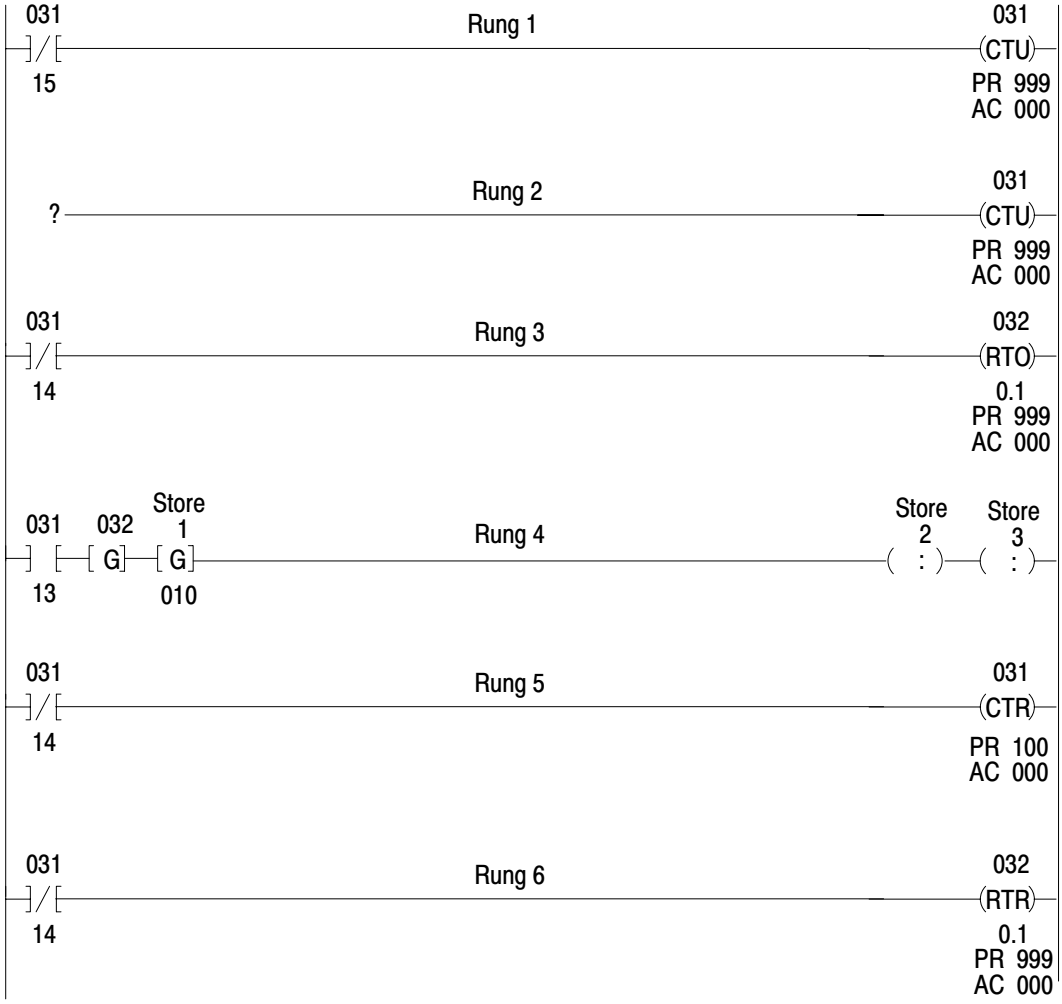
Scan Time

Scan time is the amount of time it takes the processor to monitor and update inputs and outputs, and to execute instructions in memory in accordance with your program. The scan is performed serially; first the I/o image table is updated, (other parts of the data table are not scanned), then the user program area is scanned.

There are two ways to measure I/o scan time:

- Program the rungs in Figure 6.2. The operations section of this manual provides general instruction on how to program rungs.
- Add the execution values for each instruction by using Table 6.A. The sum of these values is the I/O scan time.

Figure 6.2
Rungs for Measuring Scan Time



Here is an explanation of each rung:

Rung 1: The count increments its accumulated value each time this rung is true.

Rung 2: This rung enables the counter to increment on the next scan. If we did not have this rung, the counter would always be true and it would not increment. Remember: Counters increment only on false to true transitions.

Rung 3: The timer times in tenths of seconds when we are counting. This value is displayed on the industrial terminal screen.

Rung 4: The actual scan time is displayed beneath store 2 and store 3 in milliseconds.

Rung 5: An input device is controlling the counter.

Rung 6: An input device is controlling the timer.

Table 6.A
Approximate Execution Time Per Scan
(in average microseconds)

Instruction Name	Symbol	Instruction True	Instruction False
Examine on,Examine Off	-| |-,-|/|-	10	5
Output Energize	-( )-	19	19
Output Latch	-(L)-	19	15
Output Unlatch	-(U)-	19	15
Get	-[G]-	27	-
Put	-(PUT)-	22	15
Equal	-(=)-	22	5
Less Than	-(<)-	31	5
Get Byte	-|B|-	11	-
Limit Test	-|L|-	23	5
Counter Reset	-(CTR)-	23	15
Retentive Timer Reset	-(RTR)-	24	16
Timer On-delay	-(TON)-	140	60
Retentive Timer On-delay	-(RTO)-	140	48
Timer Off-delay	-(TOF)-	145	70
Up Counter	-(CTU)-	130	110
Down Counter	-(CTD)-	135	115
Add	-(+)-	48	15
Subtract	-(-)-	80	19

Instruction Name	Symbol	Instruction True	Instruction False
Multiply	-(x)-(x)-	615	60
Divide	-(:)-(:)-	875	60
Add to any of the above when its address is 400_8 or greater		27	27
Master Control Reset	-(MCR)-	23	20
Zone Control Last State[1]	-(ZCL)-	83	28
Branch Start		18	13
Branch End		18	13
End, Temporary End	T.END	27	27
Subroutine Area	SBR	27	27
Immediate Input Update	-[I]-	140	-
Immediate Output Update	-(IOT)-	170	33
Label	LBL	19	-
Return	-(RET)-	28	15
Jump to Subroutine	-(JSR)-	160	50
Jump	-(JMP)-	170	50
Block Transfer Read	BLOCK X-FER 1	150	135
Block Transfer Write	BLOCK X-FER 0	150	135
Sequencer Load	SEQ 2	650	200
Sequencer Input	SEQ 1	790	200
Sequencer Output	SEQ 0	730	200
File-to-word Move	FILE 12	470	200
Word-to-file Move	FILE 11	910	280

Instruction Name	Symbol	Instruction True	Instruction False
File-to-file Move	FILE 10	470	200

[1]When a rung which contains a ZCL instruction is false, the execution time of each instruction between the start fence and end fence is 17 microseconds per word.

If the scan time is over 130ms a watch dog timer (internal alarm system) will automatically timeout and the processor will shut down.

The time required for the processor to execute some instructions can be quite long. Repeated use of instructions with long execution times could cause the watch dog timer to time out. Therefore, the watch dog timer reset automatically every time the processor executes any one of the following instructions:

- File-to-file move
- File-to word move
- Word-to file move
- Sequencer input
- Sequencer output
- Return
- Temporary end
- Subroutine area

These instructions will be discussed later in this chapter.

Section B
Program Control Instructions

Introduction

Certain applications may need programming techniques designed to override a group of non-retentive outputs or update I/O ahead of the usual I/O scan time. The program control instructions satisfies this need.

Program control instructions are divided into two categories:

- Output Override
- Immediate I/O Update

The table below illustrates specific instructions for these categories:

Program Control Instructions

Output Override	Immediate Update I/O
Master Control Reset	Immediate Input Update
Zone Control last State	Immediate Output Update

The output override, or zone type instructions, operate similar to a hardwired master control relay in that they can affect a group of outputs in the user program. But these instructions not a substitute for a hardwired master control relay, which provides emergency I/O power shutdown.

Output Override Instructions

Master Control Reset

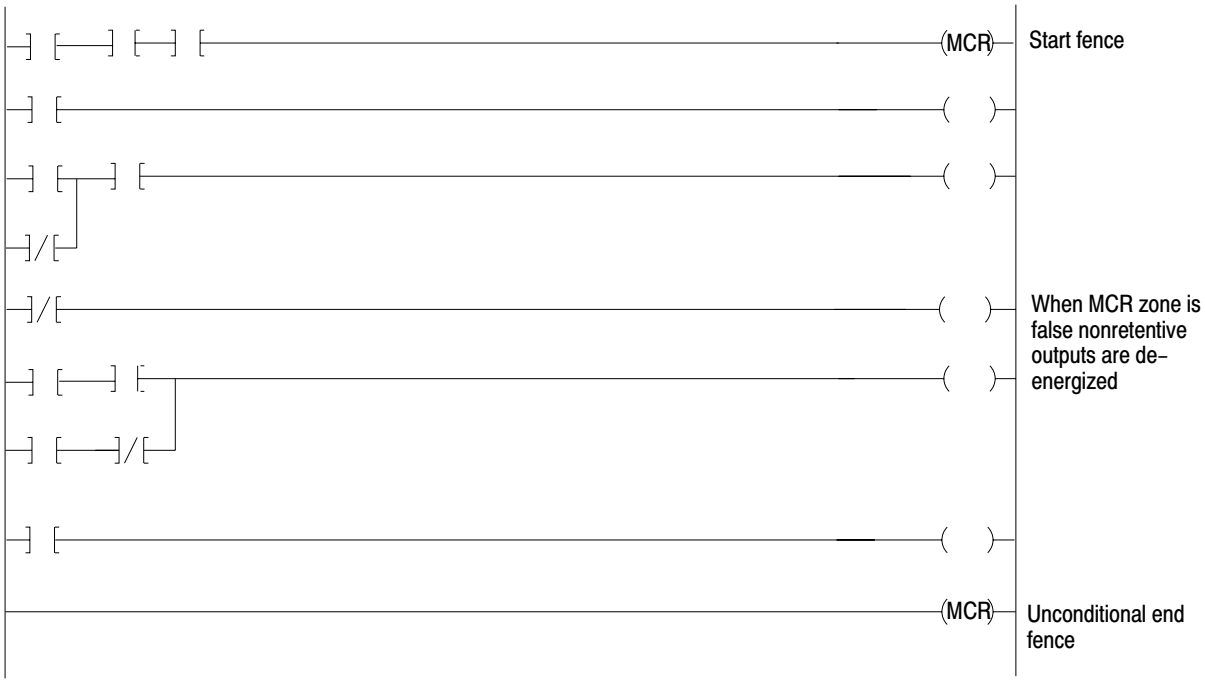
Symbol: -(MCR)-

Purpose: Controls a group of outputs.

Syntax: Two instructions are required: (Figure 6.3)

- To begin the zone (start fence)
- To end the zone (end fence)
- Program the start fence with a set of input conditions.
- Program the end fence unconditionally.
- **Do not** next MCR zones with other MCR or ZCL zones.
- Each zone must be separate and complete.

Figure 6.3
MCR Programming



Function: If the start fence becomes:

True

- Each rung condition controls their output instructions.

False

- All non-retentive output instructions within the zone area de-energize by the MCR zone.

NOTE: Latch/unlatch instructions should not be placed within an MCR zone, because the MCR zone maintains retentive instructions in the last active state when the start fence goes false.

Zone Control Logic

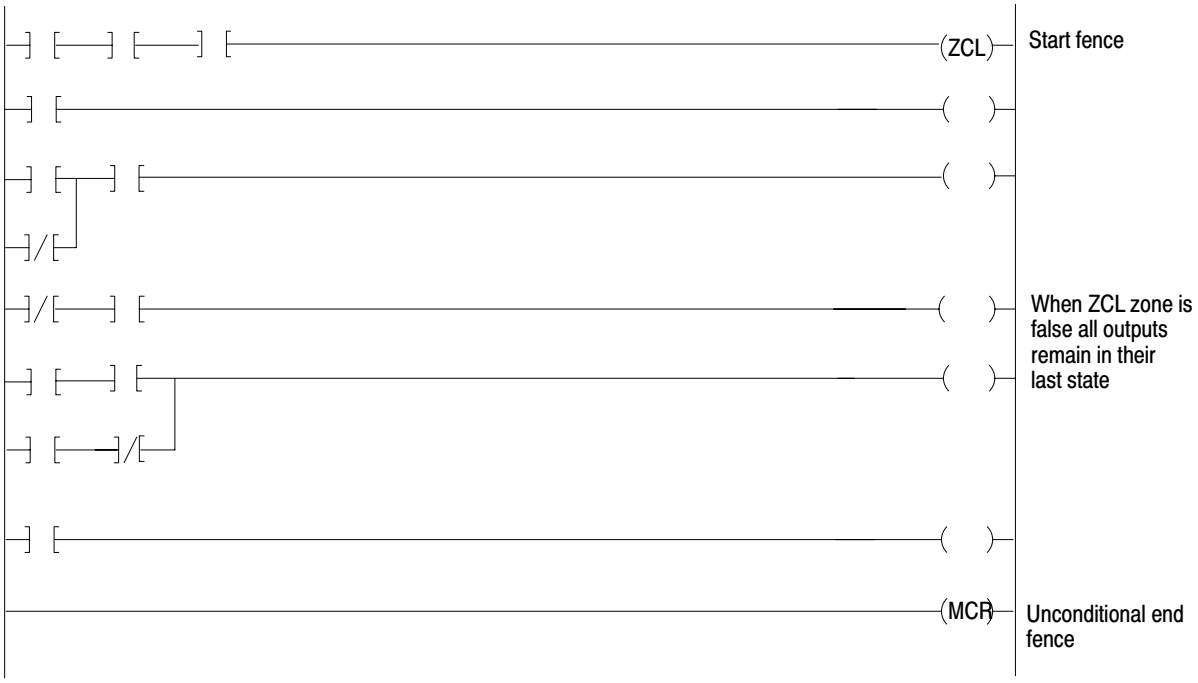
Symbol: -(ZCL)-

Purpose: Allows control of one or a group of outputs in more than one manner in the same program.

Syntax: Two instructions are required (Figure 6.4):

- Program the first ZCL instruction with input conditions to begin the zone (start fence).
- Program the second ZCL instruction unconditionally to end the zone (end fence).
- Do not nest ZCL zones with other ZCL or MCR zones.
- Each zone must be separate and complete.

Figure 6.4
ZCL Programming



Function: When the rung becomes:

True

- All output instructions within the zone act according to the logic conditions preceding them.

False

- All output instructions within the zone remain in their last state; regardless of the I/O rack last state switch setting or changes in logic on the input side of the rung. These same outputs may now be controlled by another zone program. Only one zone may control a set of outputs at one time.

Immediate Update I/O Instructions

Purpose: Interrupts the program scan to update I/O data before the normal I/O update sequence.

Function: Used where I/O modules interface with I/O devices that operate in a shorter time period than the processor scan.

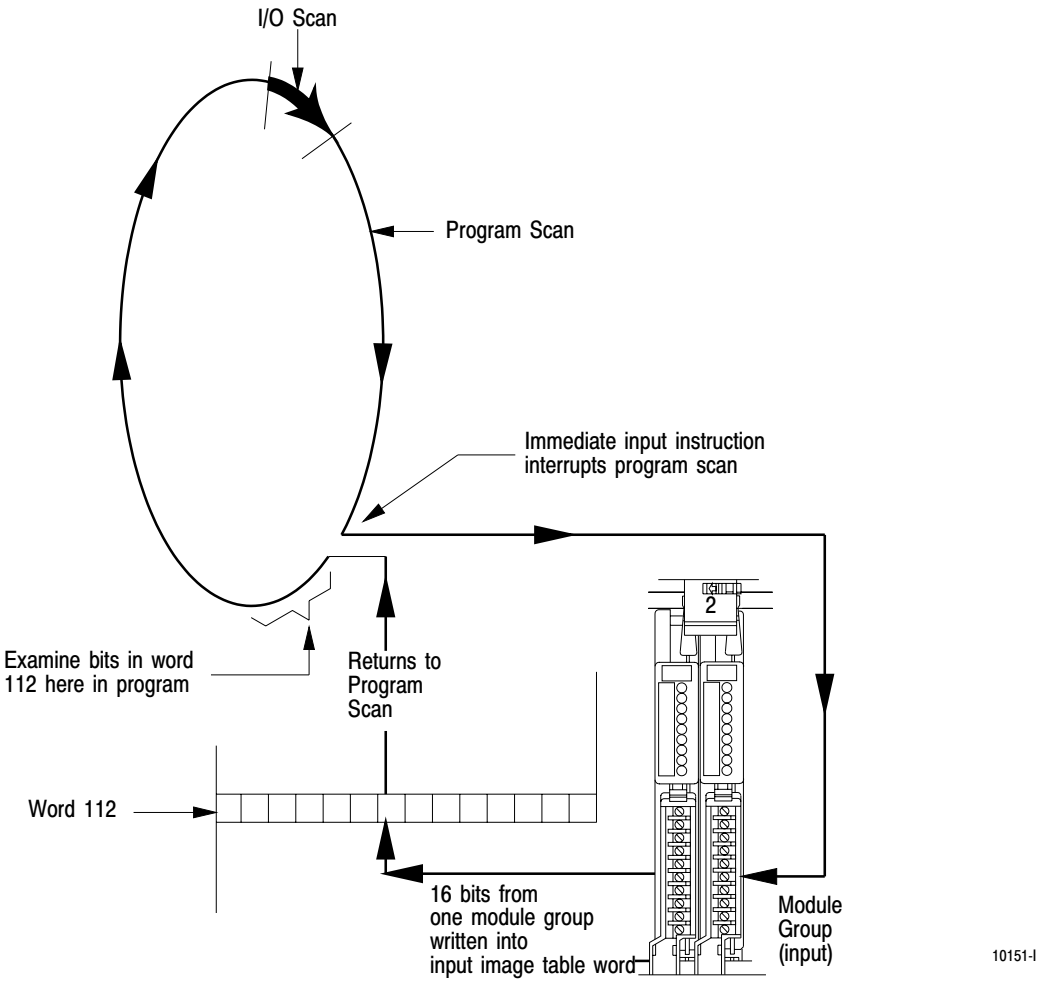
Immediate Input Update

Symbol: -|I|-

Purpose: Updates one input image table word from one input module group in advance of the normal I/O scan sequence.

Syntax: Programmed at the condition side of the logic rung just before inputs in the module group are examined in the program (Figure 6.5).

Figure 6.5
Immediate Input Instruction



Function: The instruction is always considered logic true and execution takes place whether or not other rung conditions allow logic continuity.

Immediate Output Update

Symbol: -(IOT)-

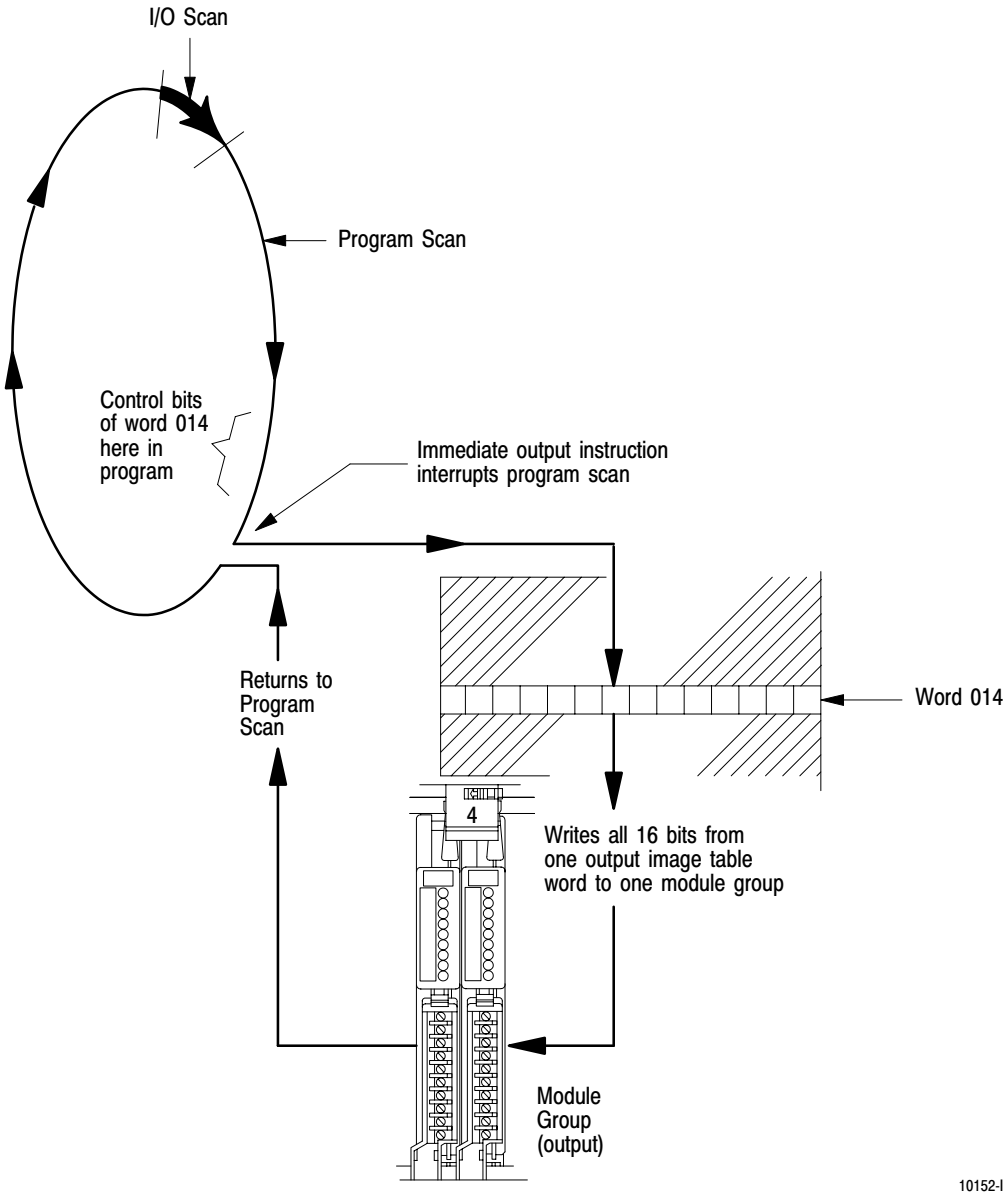
Purpose: Transfers one word from the output image table to the output module group.

Syntax: Programmed at the output side of the logic rung. When programming only assign output image table bit addresses (Figure 6.6).

Function: Execution occurs when logic continuity is established.

NOTE: To avoid loss of production time use these instructions only when absolutely necessary.

Figure 6.6
Immediate Output Instruction



Section C
Jump Instructions and Subroutine Programming

Introduction

You are capable of reducing scan time by using instructions that selectively jump over portions of a program. These instructions are:

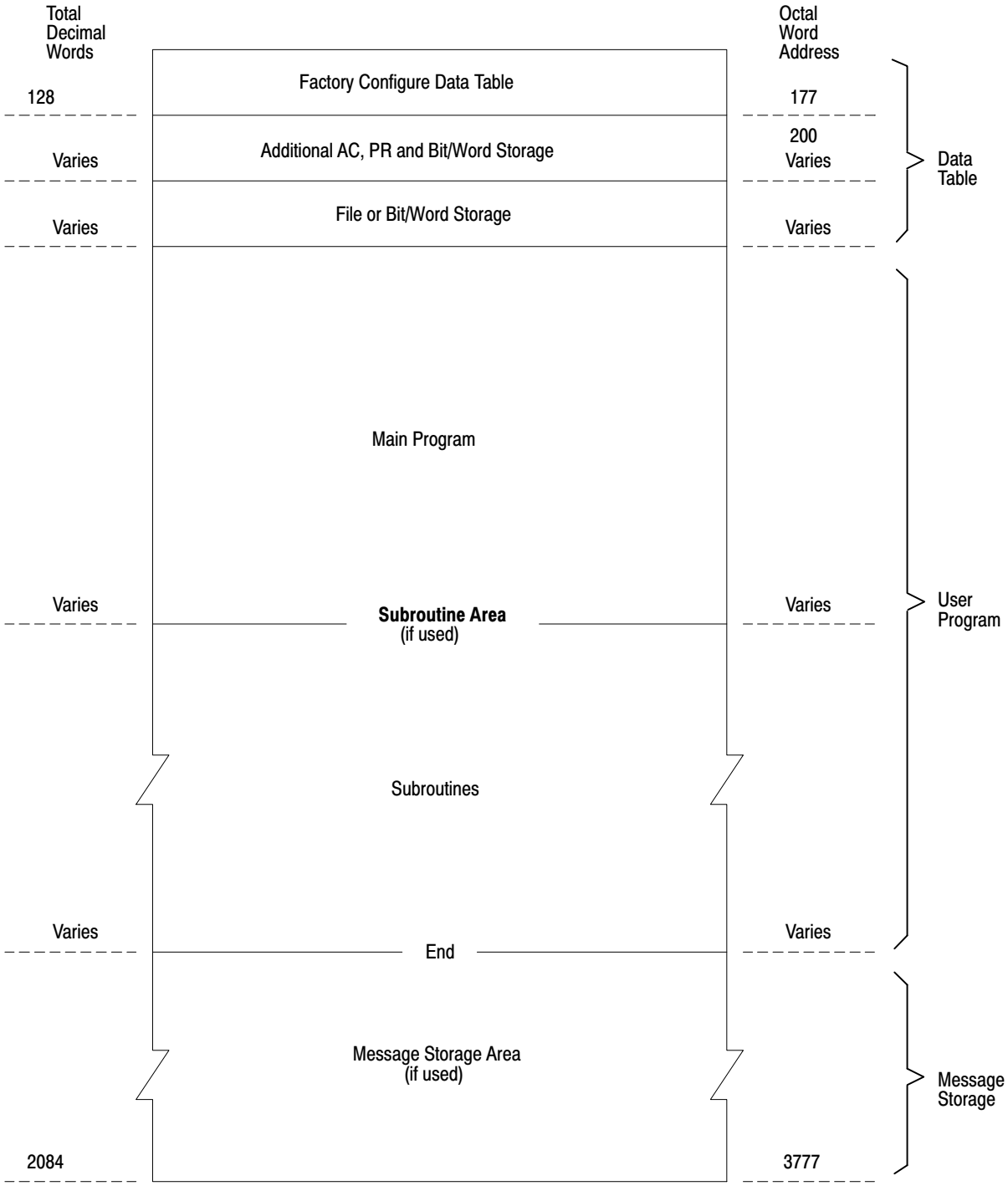
- Jump
- Label
- Jump to subroutine
- Return

This section describes how jump instructions and subroutine programming direct the path of the program scan through the main program and the subroutine area.

What is a Subroutine Area?

The subroutine area is located in the memory between the main program and the message store areas (Figure 6.7). This area acts as the end of program statement for the main program. It allows storage of small programs that are to be accessed periodically. Subroutine areas are not scanned unless you program the processor to jump to this area.

Figure 6.7
Subroutine Area



10153–I

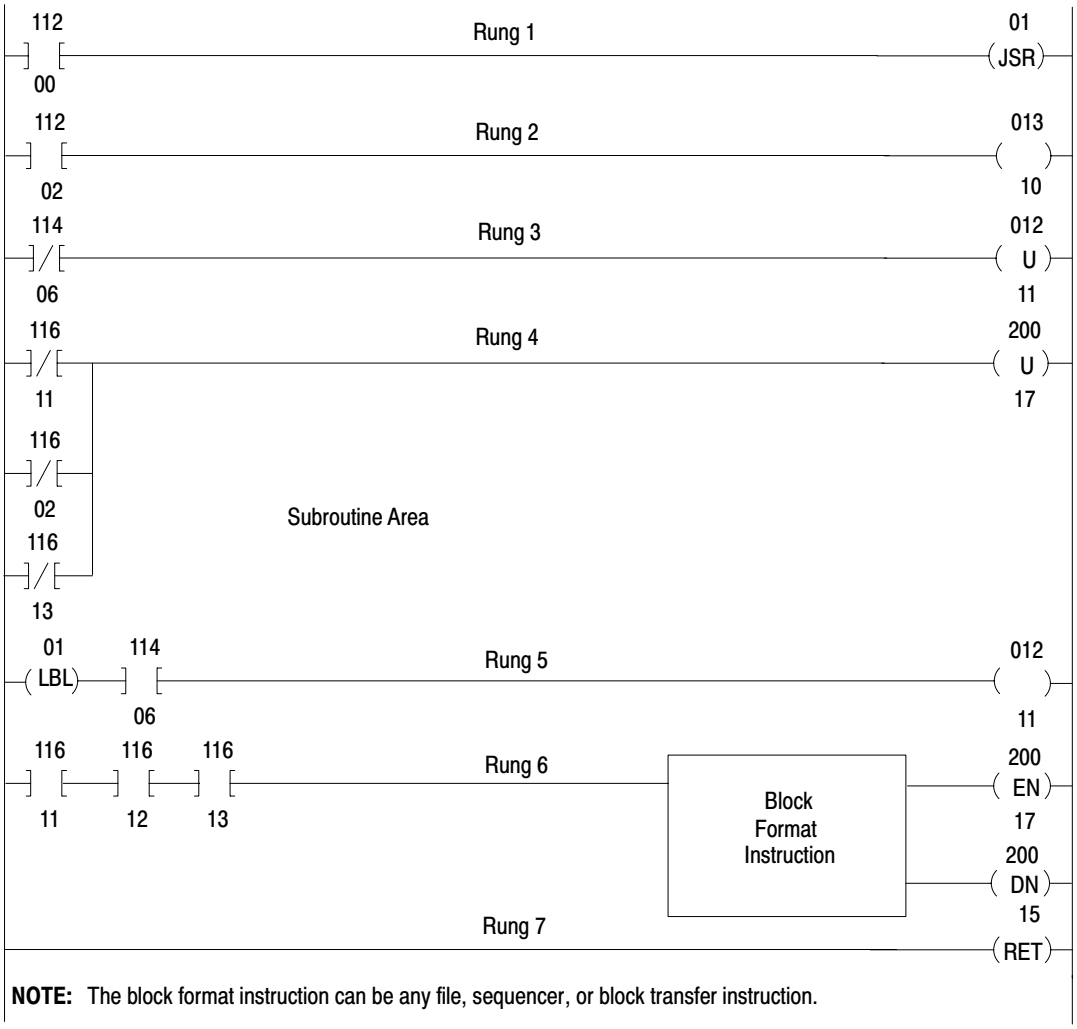
A maximum of eight subroutines can be programmed in the subroutine area. Each subroutine begins with a label instruction and (when you want to exit to your main program) ends with a return instruction. We will discuss jump, label, and return instructions later in this section.

Subroutine Area Instruction

Symbol: SBR

Purpose: Serves as an end of program statement for the main program (Figure 6.8).

Figure 6.8
Advance Data Instruction Format (General)



NOTE: The block format instruction can be any file, sequencer, or block transfer instruction.

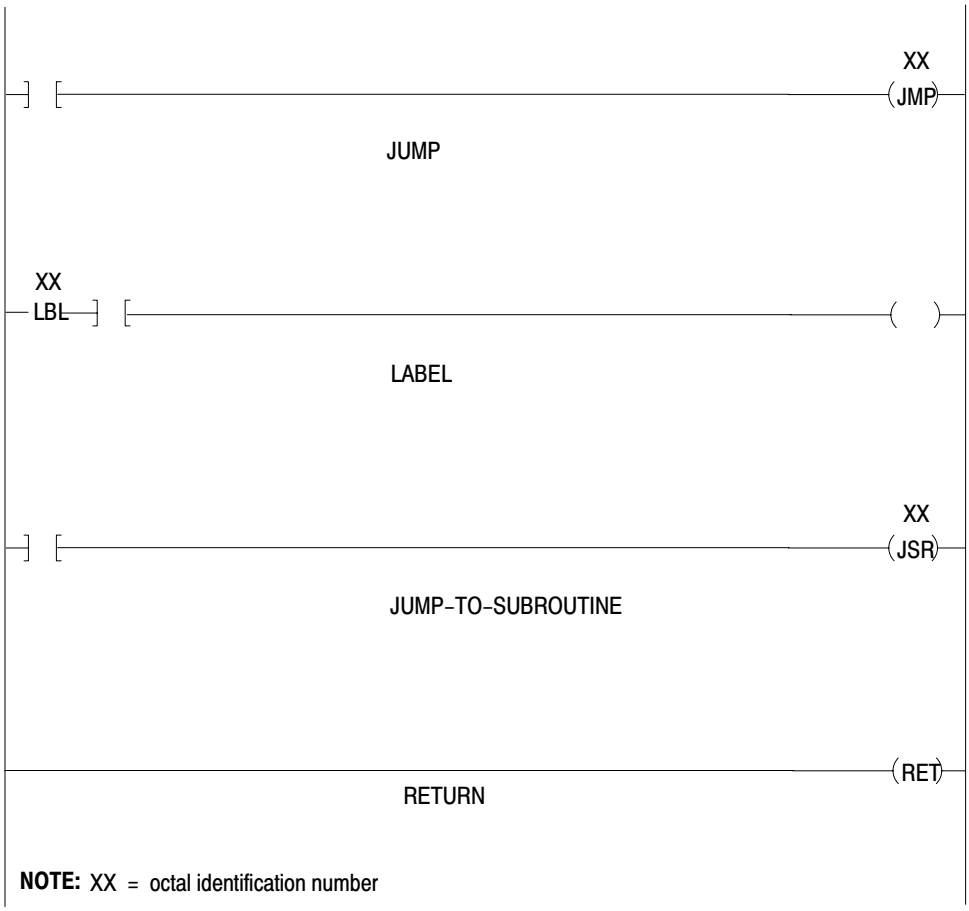
Syntax: Actual programming techniques are described in the operations section, chapter 9.

Here we will state general programming facts:

- Uses one word of memory.
- Processor does not scan the instruction until you program a jump to subroutine instruction.
- Up to eight subroutines can be programmed if you do not program any jump instructions.
- **Do not** next subroutine programs by inserting a jump to subroutine instruction in the subroutine area.

Figure 6.9 illustrates the next group of instructions.

Figure 6.9
Jump Instructions



Jump

Symbol: -(JMP)-

Purpose: Used with a label instruction to instruct the processor to jump forward in the main program to the label instruction with the same identification number. Executes in the main program.

Syntax: Programmed as an output instruction. Do not program in an area where the jump instruction crosses the boundary between the main program and subroutine area, or vice-versa.

Function: Execution takes place only on true conditions.

> **CAUTION:** Allowances should be made for conditions which could be created by the use of the jump instruction. Jumped program rungs are not scanned by the processor so that input conditions are not examined and outputs that are controlled by these rungs remain in their last state. Timers and counters cease to function. Critical rungs should be reprogrammed outside the jumped section in the program zone.

Label

Symbol: -(LBL)-

Purpose: Target for the jump and jump to subroutine instructions.

Syntax: Programmed as the first condition instruction in the rung. If conditions precede a label instruction, they will be ignored by the processor during a jump operation. Do not program with a program control instruction.

Function: Always true.

NOTE: There are 8 labels available. Each label can only be defined once (using an octal identifier), but can be the target of multiple jump or jump to subroutine instructions. Octal identifiers are labeled from 00-07.

WARNING: Do not place a label instruction in a ZCL or MCR zone. When jumping over a start fence, the processor will execute the program from the label to the end fence as if the start fence had been true, i.e. outputs controlled by the rungs. The start fence may have been false intending that all outputs within the zone be controlled by the output override instruction, i.e. off for MCR or last state for ZCL instructions. Unpredictable machine operation could occur with possible damage to equipment and/or injury to personnel.

Jump to Subroutine

Symbol: -(JSR)-

Purpose: Used with label instruction to instruct the processor to jump from the main program to the label instruction having the same identification number in the subroutine area. Executes the subroutine.

Syntax: Programmed as an output instruction. This instruction must always cause the processor to cross the boundary from the main program to the subroutine area.

Function: Execution takes place only on true conditions.

Return

Symbol: -(RET)-

Purpose: Terminates a subroutine and returns the processor to the main program.

Syntax: Programmed as an output instruction without an identification number in the subroutine area. It is usually programmed unconditional. Refer to Figure 6.9. Every subroutine must have a return instruction.

Function: Returns the processor to the instruction immediately following the jump to subroutine instruction. The main program continues to operate.

SECTION D
Advance Data Manipulation

This section describes ways to transfer file data to another designated area. In this section you will read about:

- Files
- Data monitor mode display
- Data transfer file instructions
- Sequencer instructions

Chapter 10 demonstrates the programming techniques when using advanced data manipulation instructions.

What is a File?

A file is a group of consecutive data table words used to store information. A file is defined by a counter and a starting word address. the counter has two functions:

- Defines the file length with its preset value.
- Points to a particular word in a file with its accumulated value.

The counter address is the file instruction's address. The processor uses this address to search for the instruction.

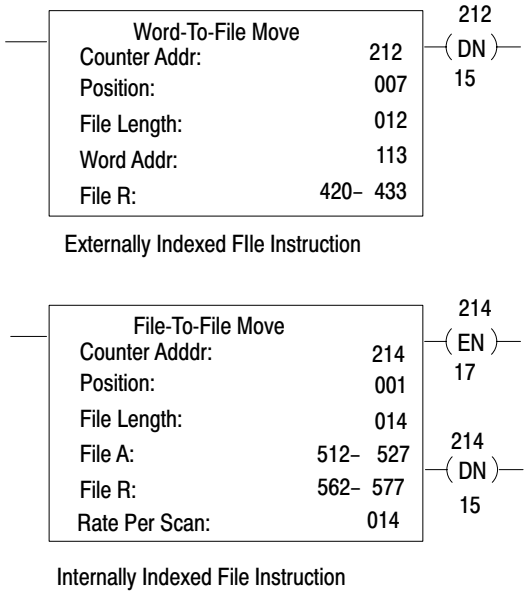
A file can be between 1 and 999 words in length. The address of word 1 defines the address of the file. When displayed, the words of a file are designated consecutively by positions 001-999 according to the length of the file.

The word address defines:

- The location in the data table to which or from which the data will be moved.
- This word address can be manipulated by ladder diagram logic.

There are two types of file instructions. Those that have an externally indexed counter and those that have an internally indexed counter. The two file instructions that have an externally indexed counter are: Word-to-file move, and File-to-word move. The file instruction that has an internally indexed counter is: File-to-file. Figure 6.10 shows the difference in format between these two types of file instructions.

Figure 6.10
Types of file Instructions



Externally Indexed

Externally indexed means that you assign an accumulated value to the counter. The counter address is the same address as the file instruction in the accumulated value area of the data table. The counter address hold the accumulated value. The accumulated value points to the file's position value. The position value is the accumulated value and it represents the specific word location within the file.

Another characteristic of the externally indexed file instruction is that it only has a done bit. This is bit 15. the done bit is automatically entered from the counter address. It is set when the operation is complete and remains set as long as the rung condition is true.

Internally Indexed

Internally indexed means that the accumulated value of the counter is internally incremented. You again assign an accumulated value to the counter.

When you look at Figure 6.10 notice that a value for rate per scan is needed. The rate per scan defines the number of words which will be operated upon during one scan. For example, suppose you have a file that contains twelve words. If you assign the value of 004 for the rate per scan that means that the instruction will execute four words per scan at a time. Therefore, the entire operation will be completed in three scans.

Another characteristic of the internally indexed file instruction is that it has a done bit and an enable bit. The done bit is bit 15 and the enable bit is bit 17. These two bits are automatically entered from the counter address. The enable bit is set when the rung logic goes from a false to true transition; the done bit is set when the file instruction is completed.

Modes of Operation

There are three modes of operation based on the rate per scan. They are:

- Complete
- Distributed complete
- Incremental

We will discuss each mode of operation.

Complete Mode

In the complete mode, the rate per scan is equal to the file length value and the entire file is operated upon in one scan. For example, if there are 12 words in your file and your rate per scan value is 12 then all 12 words will be operated upon during one scan.

For each false-to-true transition of the rung condition, the instruction is enabled, the accumulated value of the file counter is internally indexed from the first to the last word of the file. As the accumulated value points to each word, the operation defined by the file instruction is performed. After the instruction has operated on the last word, the done bit (bit 15) is set. When the rung condition goes false, both the done and enable bits are reset and the counter resets to position 001. If the rung was enabled for only one scan, the done bit would come on during the scan and remain set for one additional scan.

Distributed Complete Mode

In the distributed complete mode, the rate per scan is less than the file length value and the entire file is operated over several program scans. For example, if there are 12 words in your file and your rate per scan value is 3, then 3 words will be operated upon during each scan. Therefore, it would take 4 scans to execute the entire file instruction operation.

For each true rung condition, the instruction is enabled. The number of words equal to the rate per scan is operated upon during one scan. The process is repeated over a number of scans until the entire file has been operated upon. Once the file instruction is enabled it remains enabled for the number of scans necessary to complete the operation. The rung could become repeatedly false and true during this time without interrupting the operation of the instruction.

At the time of completion, if the rung is true, the enable bit (bit 17) and the done bit (bit 15) are both set. If the rung is false, the enable bit is reset after the

last group of words is operated upon. At the same time, the done bit is set and stays set for one scan. During the next scan the done bit is reset, and the counter is reset to position 001.

Incremental Mode

In the incremental mode, the rate per scan is equal to 0. This means that upon each false-to-true transition one word is operated upon per scan, then the counter increments to the next position. When the rung is true the enable bit (bit 17) is set. After the last word in the file has been operated upon, the done bit (bit 15) is set. When the rung goes false, the done and enable bits are reset (after the last word has been operated upon), and the counter is reset to position 001. If the rung remains true for more than one scan, the operation does not repeat. The operation only occurs in the scan in which the false-to-true transition occurs.

To change from one mode to another use Table 6.B to determine the values:

Table 6.B
Changing Modes

To change:	Enter the Rate per Scan Value:
Complete to Distribute Complete	001 thru 006
Distributed Complete to Incremental	000
Distributed Complete to Complete	007
Incremental to Complete	007

Data Monitor Mode Display

Once you establish your file data, you'll want to edit, load, or monitor your file data. To do these functions the processor has a data monitor mode. This mode lets you access your file in two ways: either by displaying binary values or hexadecimal values (Figure 6.11).

The binary data monitor display lets you manipulate one word at a time by displaying each bit using binary digits. The hexadecimal monitor display lets you manipulate 4 digits which represents word values. The industrial terminal can automatically convert your data from one number system to the other when the alternate display is selected.

Three sections divide the data monitor display. They are identified as (Figure 6.11):

- **Header:** located at the top of the screen and contains information pertaining to its corresponding file instruction. For example: counter, file, word addresses, and file length.
- **File Section:** located in the center of the screen and displays the data stored in a file. The column labeled POSITION refers to each word's position in the file. FILE A DATA represents the original file, and FILE R DATA represents the new file.
- **Command Buffer:** located at the bottom center of the screen and is used to enter or change file data. It is always displayed in the program mode.

Figure 6.11
Data Monitor Displays

HEXADECIMAL DATA MONITOR
FILE TO FILE MOVE

Header
COUNTER ADDR: 200 POSITION: 001 FILE LENGTH: 007
FILE A: 400–406 FILE R: 500–506

File Section

POSITION	FILE A DATA	FILE R DATA
001	0000	0000
002	0000	0000
003	0000	0000
004	0000	0000
005	0000	0000
006	0000	0000
007	0000	0000

Command Buffer → DATA: 0000

BINARY DATA MONITOR
SEQUENCER OUTPUT

Header
COUNTER ADDR: 200 STEP: 008 SEQUENCER LENGTH: 009
FILE: 600–610

OUTPUT ADDR:		011	013
DATA:	11110000	11000011	1111100 00011000
MASK ADDR:		211	212
DATA:	11111111	11111111	11111111 11000000

File Section

STEP	WORD 1		WORD 2	
001	00000000	00000000	00000000	00000000
002	00000000	00000000	00000000	00000000
003	00000000	00000000	00000000	00000000
004	00000000	00000000	00000000	00000000
005	00000000	00000000	00000000	00000000
006	00000000	00000000	00000000	00000000

Command Buffer → DATA: 00000000 00000000

Data Transfer File Instructions

There are three types of data transfer file instructions:

- File to file move
- File to word move
- Word to file move

Refer to Figure 6.12 while you are reading about each file instruction.

Figure 6.12
File Instructions

Key Sequence	1770–T3 Display	Instruction Notes
FILE 10	File to File Move Counter Addr: 030 Position: 001 File Length: 001 File A: 110- 110 File R: 110- 110 Rate per Scan 001 030 EN 17 030 DN 15	Output Instruction. Modes: Complete, Distributed and Incremental. Counter is internally incremented by the instruction. Requires 5 words of user program.
FILE 11	Word to File Move Counter Addr: 030 Position: 001 File Length: 001 Word Addr: 010 File R: 110- 110 030 DN 15	Output instruction. Counter must be externally indexed by user program. Data is transferred every scan that rung is true. Requires 4 words of user program.
FILE 12	File to Word Move Counter Addr: 030 Position: 001 File Length: 001 File A: 110- 110 Word Addr: 010 030 DN 15	Same as word–to–file.

NOTE: Numbers shown are default values. Numbers in shaded areas must be replaced by user-entered values. The number of 0 default address digits initially displayed (3 or 4) will depend on the size of the data table.

Here is an explanation of each value:

Counter Address:	Address of the instruction in the accumulated value area of data table.
Position:	Current word being operated upon (accumulated value of counter).
File Length:	Number of words in file (preset value of the counter).
File A:	Starting address of source file.
File R:	Starting address of destination file.
Word Address:	Address of source word or destination word outside of file.
Rate per Scan:	Number of data words moved per scan.

File to File Move

Symbol: FILE 10

Purpose: Duplicates and transfers your designated file to another file address that you identified. The original file remains intact.

Syntax: Programmed as an output instruction; requires 5 words of the user program area.

Function: The counter is incremented internally by the instruction.

> **WARNING:** The counter address for the file-to-file move instruction should be reserved for that instruction. Do not manipulate the counter accumulated or preset word. Changes to these values could result in unpredictable machine operation or a run-time error. Damage to equipment and/or injury to personnel could occur.

When the rung becomes:

True

- The data is transferred from the original file to your designated file.

False

- No action is taken.

Word to File Move

Symbol: FILE 11

Purpose: Duplicates and transfers the data of a word from the data table to a specified word within a file.

Syntax: Programmed as an output instruction; requires 4 words of the user program area.

Function: Your program must externally index the counter.

When the rung goes:

True

- Data from a designated word address in the data table is transferred to the selected position in the file.

False

- No action is taken.

File to Word Move

Symbol: FILE 12

Purpose: Duplicates and transfers the data of a word within your designated file to a specified word elsewhere in the data table.

Syntax: Programmed as an output instruction; requires 4 words of the user program area.

Function: Your program must externally index the counter. When the rung becomes:

True

- The data of a word is transferred from your file to your designated word address.

False

- No action is taken

WARNING: The counter address for the word-to-file move and file-to-word move instructions should be used only for the intended instruction and the corresponding instructions which manipulate the accumulated value. Do not inadvertently manipulate the preset or accumulated word. Changes to these values could result in unpredictable machine operation or a run time error. Damage to equipment and/or injury to personnel could occur.

Sequencer Instructions

There are three sequencer instructions:

- Sequencer input
- Sequencer output
- Sequencer load

These instructions either transfer information from the data table to output word addresses, compares I/O word information with information stored in tables, or transfers I/o word information into the data table. Understanding and applying these concepts will give you flexibility with your programs.

Refer to Figure 6.13 while you are reading about each sequence instruction.

Figure 6.13
Sequencer Instructions

Key Sequence	1779-T3 Display	Instruction Notes
SEQ 0	Sequencer Output Counter Addr: 030 Current Step: 001 Seq Length: 001 Words per Step: 1 File: 110- 110 Mask: 010- 010 Output Words 1: 010 2: 3: 4: 030 EN 17 030 DN 15	Output instruction. Increments, then transfers data. Same data transferred each scan that the rung is true. Counter is indexed by the instruction. Unused output bits can be masked. Requires 5–8 words of your program.
SEQ 1	Sequencer Input Counter Addr: 030 Current Step: 000 Seq Length: 001 Words per Step: 1 File: 110- 110 Mask: 010- 010 Input Words 1: 010 2: 3: 4:	Input instruction. Compares input data with current step for equality. Counter must be externally indexed by your program. Unused input bits can be masked. Requires 5–8 words of your program.
SEQ 2	Sequencer Load Counter Addr: 030 Current Step: 000 Seq Length: 001 Words per Step: 1 File: 110- 110 Output Words 1: 010 2: 3: 4: 030 EN 17 030 DN 15	Output instruction. Increments, then loads data. Counter is indexed by the instruction. Does not mask. Requires 4–7 words of your program.

Here is an explanation of each value:

Counter Address:	Address of the instruction in accumulated value area of data table.
Current Step:	Position in sequencer table (accumulated value of counter).
Seq Length:	Number of steps (preset value of the counter).
Words per Step:	Width of sequencer table.
File:	Starting address of sequencer table.
Mask:	Starting address of mask file.
Output Words:	Words controlled by the instruction.
Load Words:	Words fetched by the instruction.
Input Words:	Words monitored by the instruction.

Sequencer Input

Symbol: SEQ 1

Purpose: Compares input data to stored data for equality.

You can compare up to 64 inputs.

Syntax: programmed as an input instruction. May be programmed with a sequencer output instruction.

You mask the unused input bits (chapter 10 explains the term mask).

The counter is externally controlled by the ladder diagram logic.

This instruction requires 5-8 words of your program.

Function: You must externally index the counter in your program.

If the rung condition becomes:

True

- Action is taken.

False

- No action is taken.

> **WARNING:** The counter address of the sequencer input instruction should be used only for the intended instruction and the corresponding instructions which manipulate the accumulated value. Do not inadvertently manipulate the preset or accumulated word. Changes to these values could result in unpredictable machine operation or a run-time error. Damage to equipment and/.or injury to personnel could occur.

Sequencer Output

Symbol: SEQ 0

Purpose: Controls consecutive outputs for every step of the sequencer.

Controls up to 64 outputs simultaneously.

Syntax: Programmed as an output instruction. Can be used with a sequencer input instruction or another input instruction.

You mask the unused output bits (chapter 10 explains mask).

This instruction requires 5-8 words of the user program area.

Function: Outputs are controlled upon execution of the instruction then and the counter increments to the next step.

When the rung becomes:

True

- The counter increments to the next step and that data will be outputted every scan that the rung remains true.
- When AC=PR, the done bit is set.

False

- Outputs remain in their last state unless changed by instructions elsewhere in your program.

> **WARNING:** The counter address of the sequencer output instruction should be reserved for that instruction. Do not manipulate the counter preset or accumulated word. Changes to these values could result in unpredictable machine operation or a run-time error. Damage to equipment and/or injury to personnel could occur.

Sequencer Load

Symbol: SEQ 2

Purpose: Places data into the sequencer file that you established in the data table for this instruction.

Syntax: Programmed as an output instruction.

You can **not** mask any unused bits.

This instruction requires 4-7 words of the user program area.

Function: A false-to-true rung transition enables the instruction. When the rung becomes:

True

- The instruction increments to the next step and executes the instruction. (Loads the data).

False

- No action is taken.

Use this instruction for:

- Machine diagnostics - If when the actual sequence of an operation becomes mismatched with the desired sequence of operation as contained in the sequencer input instruction, a fault signal can be enabled by the user program.

- Teach sequential operation - The I/O conditions representing the desired operation can be loaded into the sequencer input tables as the machine is manually stepped through the control cycle.

WARNING: The counter address of the sequencer load instruction should be reserved for that instruction. Do not manipulate the counter accumulated or preset word. Changes to these values could result in unpredictable machine operation or a run-time error. Damage to equipment and/or injury to personnel could occur.

SECTION E
Block Transfer Instructions

Introduction

Block transfer refers to a set of instructions and a programming technique used to transfer many words of data in one I/O scan. You can transfer data either from intelligent 1771 I/O modules to the processor's data table or from the processor's data table to the intelligent 1771 I/O modules.

There are two types of block transfer instructions:

- Block transfer read
- Block transfer write

We will discuss these instructions later in this chapter.

These instructions can transfer from 1 to 64 words depending on the particular type of intelligent I/O module.

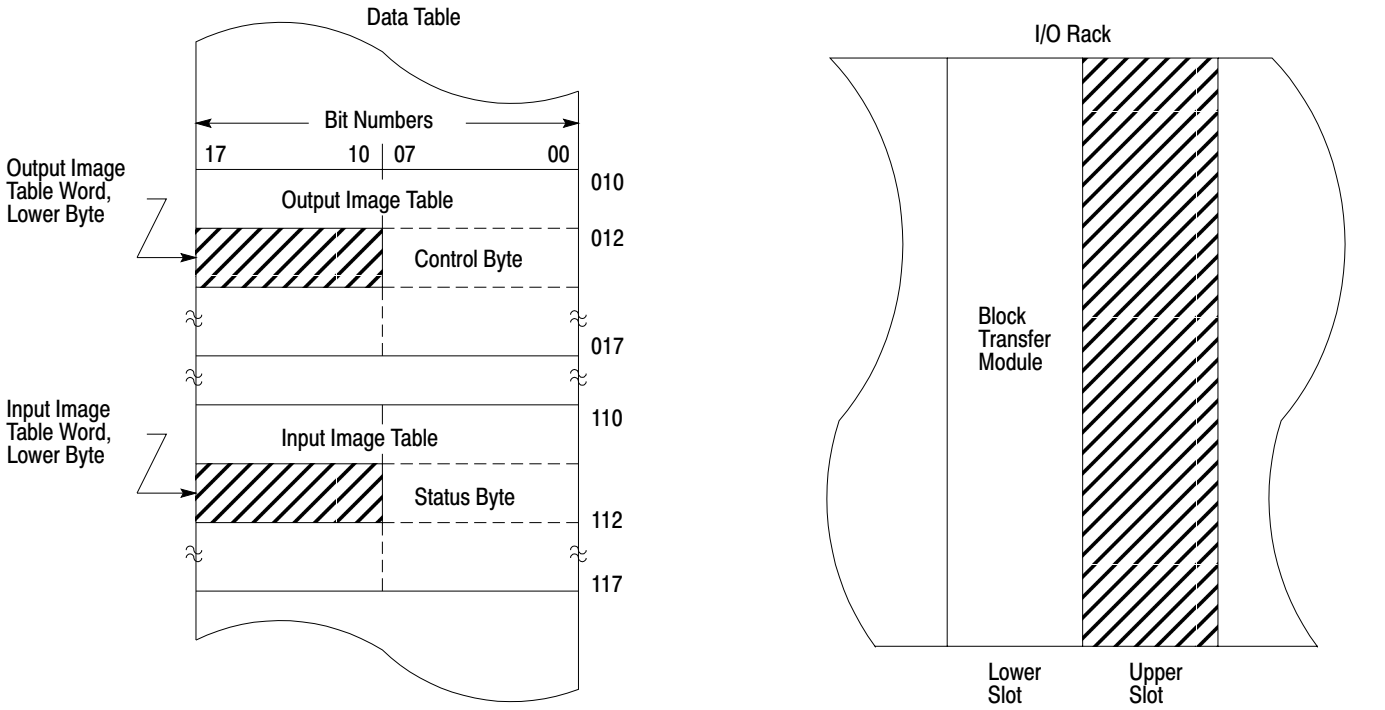
Basic Operation

The processor uses two I/o image table bytes to communicate with block transfer modules. The byte corresponding to the module's address in the output image table (control byte) contains the read or write bit for initiating the transfer of data. The byte corresponding to the module's address in the input image table (status byte) is used to signal the completion of the transfer.

NOTE: Do not use word 127 for data storage.

Whether the upper or lower byte of the I/O image table word is used depends on the position of the module in the chassis' module group. If the module is installed using a left slot or dual slot then the lower byte is used. Address the left slot as slot 0, address the upper slot as slot 1 (Figure 6.14).

Figure 6.14
Image Table Byte Relationship vs Module Position



The lower byte of the I/O image table words are used when the module is in the lower slot and vice versa.

10154–I

The block transfer read or write operation is initiated in the program scan and completed in the I/O scan as follows:

Program Scan - When the rung goes true, the instruction is enabled. The number of words to be transferred by the read or write bit that controls the direction of transfer are set by a bit pattern in the output image table byte.

I/O Scan - The processor requests a transfer by sending the output image table byte data to the block transfer module during the scan of the output image table. The module signals that it is ready to transfer. The processor then interrupts the I/O scan and scans the timer/counter accumulated area of the data table, looking for the address of the module that is ready to transfer. The module address is stored in BCD at a word address in the same manner as an accumulated value of a timer is stored. The module address was entered by the programmer when entering the block instruction parameters. (The word address at which the module address is stored is called the data address of the instruction. See Table 6.C.)

Once the module address is found, the processor locates the address of the file to which (or from which) the data will be transferred. The file address is stored in BCD at an address 1008 above the address containing the module address. This is done in the same manner that the processor locates the preset value of a timer in a word address 1008 above the accumulated value address.

Table 6.C
Timer/Counter Block Transfer Analogy

Timer/Counter	Equals	Block Transfer Analog
Address of Accumulated Value		Data Address of Instruction
Accumulated Value in BCD		Module Address in BCD
Address of Preset Value		100_8 Above Data Address
Preset Value in BCD		File Address in BCD

After locating the file address in the data table, the processor then duplicates and transfers the file data consecutively one word at a time until complete, starting at the selected file address.

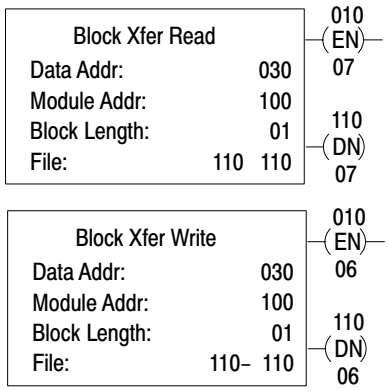
At the completion of the transfer, a done bit for the read or write operation is set in the input image table byte as a signal that a valid transfer has been completed.

Block Transfer Syntax

The format of a block transfer read and a block transfer write instruction with default values is shown in Figure 6.15. An example of a program using these instructions will be illustrated in chapter 11.

Let's look at the instruction:

Figure 6.15
Block Transfer Format



NOTE: Numbers shown are default values. Numbers in shaded areas must be replaced by user–entered values. The number of default address digits initially displayed (3 or 4) will depend on the size of the data table.

Here is an explanation of each value:

Data Address	:	First possible address in the timer/counter accumulated value area of data table.
Module Address	:	RGS for R = rack, G = module group, S = slot number.
Block Length	:	Number of words to be transferred. (00 can be entered for default value or for 64 words).
File	:	Address of first word of the file.
Enable bit –(EN)–	:	Automatically entered from the module address. Set on when rung containing the instruction is true.
Done bit –(DN)–	:	Automatically entered from the module address. Remains on for 1 program scan following successfule transfer.

There are several parts to the instruction that need to be explained. They are:

- Data Address
- Module address
- Block length
- Module address
- Block length
- File address
- Enable bit
- Done bit

Data Address and Module Address

The data address is used to store the module address of the block transfer module. The data address must be assigned the first available address in the timer/counter accumulated area of the data table starting at word address 0308. When more than one block transfer module is used, consecutive data addresses must be assigned ahead of address for timer and counter instructions.

The module address is stored in BCD by R=rack, G=module group and S=slot number (concepts from chapter 4). When block transfer is performed, the processor searches the timer/counter accumulated area of the data table for a match of the module address.

The boundary word data bits can be set manually using bit manipulation [SEARCH] [5][3], or by get/put transfer. The get/put transfer can be

programmed by assigning the get and put instructions to the address immediately following the last block transfer data address. The value of the get instruction is reset to 000 when programmed.

Defining the Block Transfer Data Address Area

When the block transfer instructions are used, the first word and consecutive words of the timer/counter accumulated area of the data table must be reserved for block transfer data addresses.

Block transfer data addresses should be separated from the addresses of timer and counter instructions by inserting a boundary. When the processor sees this boundary word, it will not search further for block transfer data. In addition, the processor is prevented from finding other BCD values that could, by chance, be in the same configuration as the rack, group and slot numbers found in block transfer data addresses.

Block Length

The block length is the number of words that the module will transfer. it depends on the type of module and the number of channels connected to it. The number of words requested by the instruction (block length value) must be a valid number for the module: i.e. from 1 up to the maximum for the module. The block length can also be set at the default value of the module. This is useful when programming bidirectional block transfers. See chapter 11.

The block length heading of the instruction will accept any value from 00 to 63 whether or not the number is valid for a particular module. A value of 00 is entered for the default value and/or for a block length of 64.

The block length is stored in binary in the byte corresponding to the module's address in the output image table.

File Address

The file address is the first word of the file to which (or from which) the transfer will be made. The file address is stored 1008 words above the data address of the instruction. When the file address is entered into the instruction block, the industrial terminal computes and displays the ending address based on the block length.

When reserving an area for a block transfer file, an appropriate address must be selected to ensure that block transfer data will not write over assigned timer/counter accumulated or preset values. The file address cannot exceed address 3577.

Done and Enable Bits

The read and write bits are the enable bits for block transfer modules. Either one (or both for a bidirectional transfer) is set in the program scan when the rung containing the block transfer instruction is true.

The done bit is set in the I/O scan that the words are transferred, provided that the transfer was initiated and successfully completed. The done bit remains set for only one additional program scan.

Block transfer will be requested in each program scan that the read and/or write bits remain set. The read and/or write bits are turned off when the rung containing the instruction goes false.

Block Programming Instructions

Block Transfer Read

Symbol: BLOCK
X-FER 1

Purpose: Reading information from a 1771 I/O module to the processor's input image table in one I/O scan.

Syntax: Programmed as an output instruction. This instruction requires two words of your program.

Function: Acts upon false-to-true transitions. When the rung goes:

True

- Data is transferred.

False

- No action is taken.

Block Transfer Write

Symbol: BLOCK
X-FER 0

Purpose: Writing information from the processor's output image table to the 1771 I/O module in one I/O scan.

Syntax: Programmed as an output instruction. This instruction requires two words of your program.

Function: Acts upon false-to-true transitions when the rung goes:

True

- Data is transferred.

False

- No action is taken.

Buffering Data

The purpose of block transfer data buffering is to allow the data to be validated before it can be used. Data that is read from the block transfer module and transferred to data table locations must be buffered. Data that is written to the module does not need to be buffered because block transfer modules perform this function internally.

Transferred data is buffered to ensure that both the transfer and the data are valid. As an example, readings from an open-circuited temperature sensor (invalid data) could have a valid transfer from an analog input module to the data table. The processor examines the data-valid bit and/or the diagnostic bit which is contained in the transferred data to determine whether or not the data is valid. The block transfer done bit is set if the transfer is valid.

The data-valid bit and/or the diagnostic bit differs for each block transfer module. Some modules set one or both for the entire file of words transferred, while others set a data-valid bit or diagnostic bit in each word. Refer to the

respective user's manual for the block transfer module to determine the correct usage of the diagnostic and/or data valid bit(s).

One technique of buffering data is to store the transferred data in a temporary buffer file. If the data in the buffer is valid, it is immediately transferred to another file in the data table where it can be used. If invalid, it is not transferred but written over in the next transfer. Another technique uses only one file. The technique prevents invalid data from being operated upon by preconditioning the rungs that would transfer data out of a file one word at a time. Diagnostic and/or data-valid bits are examined in these rungs.

Data can be moved from storage word-by-word using get/put transfers. Or, the entire buffer file can be moved at once using a file-to-file move instruction. The choice depends on the kinds of diagnostic and/or data-valid bits and the objectives of the user program. Generally, when one diagnostic bit is contained in each word, a get/put transfer is used. When one is set for the entire file, a file-to-file move instruction is used. In either case, the diagnostic bits are examined as conditions for enabling the file move or word transfer.

The example in Figure 6.16 shows the memory map and ladder diagram rungs for buffering 3 words of data that are read from the block transfer module. The data is read and buffered in the following sequence:

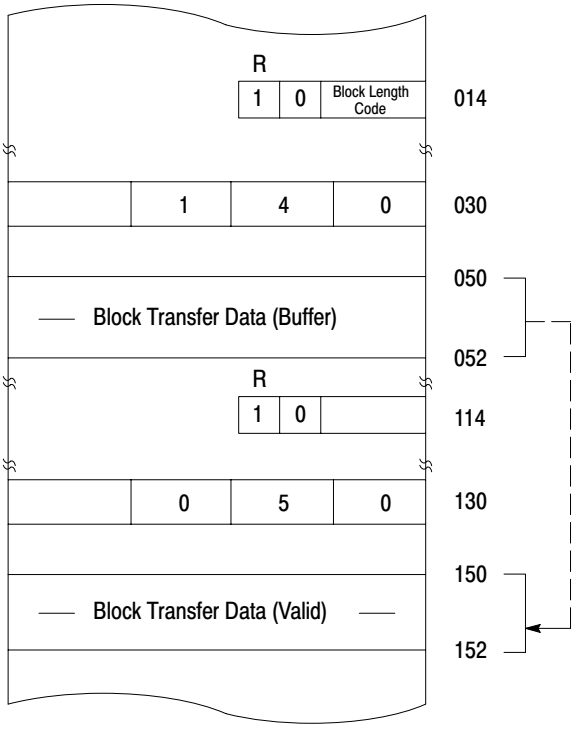
1. When rung 3 goes true, bit 01407 (the block transfer enable bit) will be turned on and block transfer will be requested. This latches on storage bit 01000 in rung 4.

2. Block transfer will be enabled during the program scan. The transfer will be performed during an interruption of the next I/O scan. Data from the module will be loaded into words 050-052. when block transfer is complete, bit 114/07 (the block transfer done bit) is set in the input image table. This indicates block transfer was successfully performed. The processor then continues with the I/O scan and program scan.

3. During the program scan, rung 1 will be true because bit 01000 is still latched on and bit 11407 is on because block transfer was performed. This will turn bit 01002 on. in rung 2, bit 01000 is then unlatched.

4. In rung 5, bit 01002 is still on and a diagnostic bit is examined to ensure the data read from the module is valid. Assuming the data is valid, the diagnostic bit will be on and the data will be transferred from word 050 to 150. In rungs 6 and 7, the data in words 051 and 052 will be transferred to words 151 and 152, respectively, if the diagnostic bits are on.

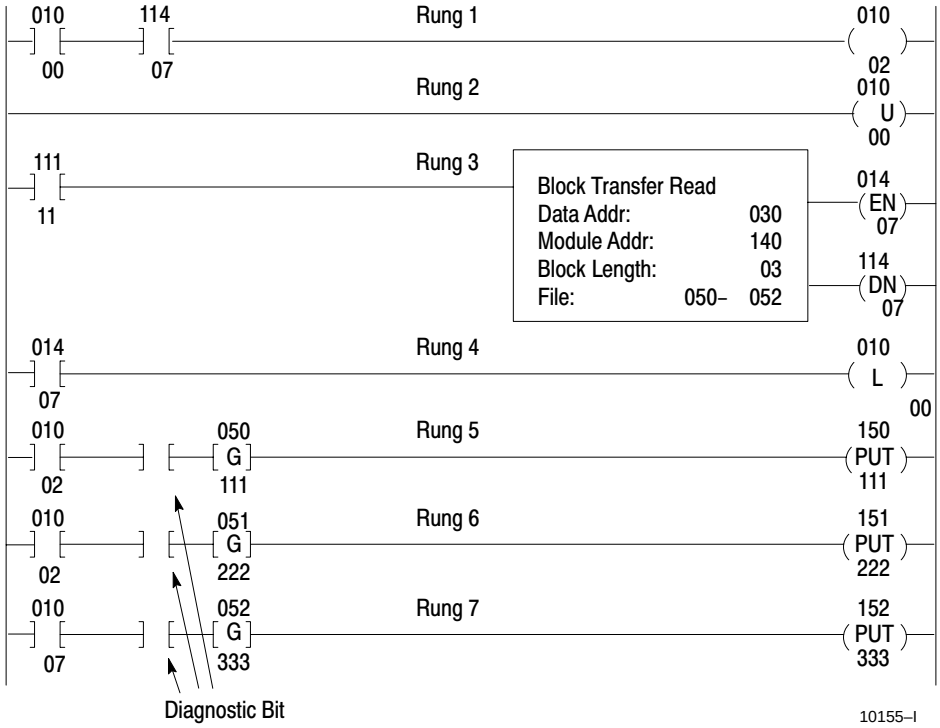
Figure 6.16
Buffering Data



Data in the buffer file 050–052 will be moved to 150–152 when:

A. Done Bit 114/07 is set (Valid transfer)

B. Diagnostic Bit is TRUE for each word to be moved in rungs 5–7 (valid data)



Chapter 7

Operations Overview

To the Reader

This section is the operations section. it will get you started in programming by giving you step by step directions for relay type instructions and editing functions. This section is not designed to give detailed instructions for all of our commands located within the Industrial Terminal (cat. no. 1770-T3). We designed a Quick Reference section to let you familiarize yourself with the commands.

Application examples for instructions such as timers, counters, arithmetical functions, data manipulation, and advanced program instructions are located in this operations section of the manual.

Before you begin, we suggest that you have the latest firmware revisions. Contact your local Allen-Bradley distributor or sales representative for the latest firmware revisions.

Conventions

We know that your action will be pressing each key. To avoid giving redundant directions, each page of this section is divided into two columns:

- KEY : Tells you what key or keys to press.
- DISPLAY : Tells you the controller's action.

Also, unless we indicate otherwise:

- Words in [ ] denote the key name or key symbol.
- Words in ( ) denote information that you must provide. For example, an address value.
- Data table word addresses are reported in octal values.

Let's Begin

Industrial terminal key Symbols

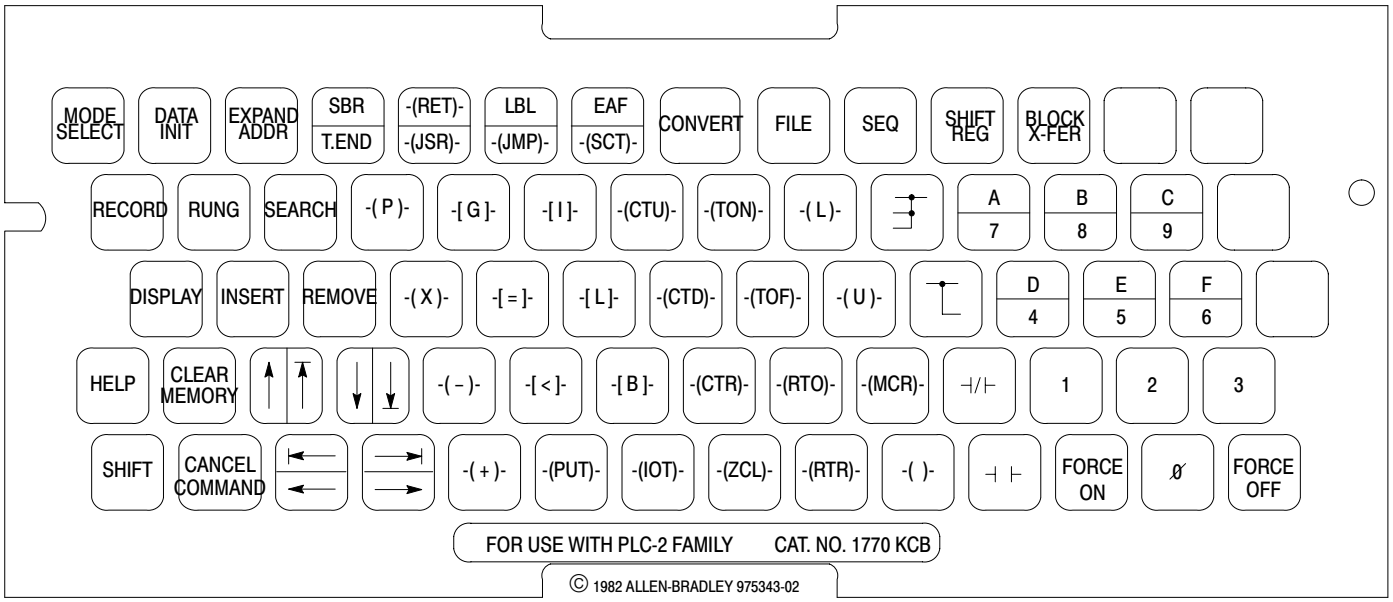
There are not numbered keys greater than 9. To display numbers which are greater than 9 press the individual keys. For example:

To display: 1011 press individually: 1011

Some keys have two symbols occupying one key (Figure 7.1). To display the top section of each key use your shift key before the desired symbol. For example:

A	Press To display 7
7	[Shift] To display A

Figure 7.1
PLC-2 Family Keytop Overlay

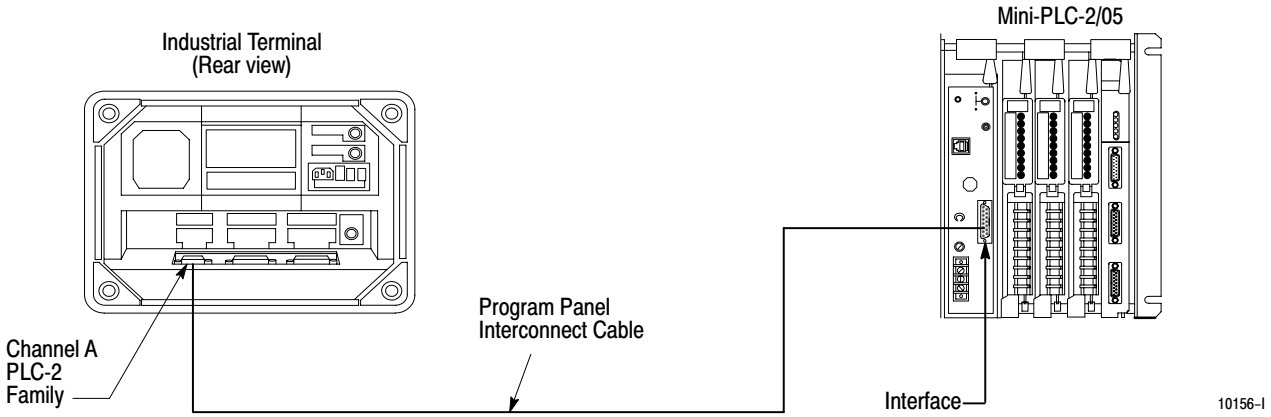


10291–I

Industrial terminal Installation

Before you start to program your controller make sure all of your peripheral equipment is installed properly. Follow these basic instructions to connect the industrial terminal to the controller. Refer to Figure 7.2 when following these instructions.

Figure 7.2
Industrial Terminal Installation



10156–I

5. Connect the AC power cord of the industrial terminal to the incoming AC power source.

6. Connect one end of the PLC-2 Program Panel Interconnect Cable (cat. no. 1772-TC) to CHANNEL A at the rear of the industrial terminal.

7. Connect the other end of the cable to the socket labeled INTERFACE at the front of the controller.

8. Place the PLC-2 Family Keytop Overlay (cat. no. 1770-KCB) onto the keyboard.

9. Turn the power switch on the front of the industrial terminal to the ON position.

10. Select your desired processor mode by turning the keyswitch to PROG.

Refer to our Publication Index (publication SD499) for additional literature information regarding our peripheral equipment.

WARNING: Use only Allen-Bradley authorized programming devices to program Allen-Bradley programmable controllers. using unauthorized programming devices may result in unexpected operation, possibly causing equipment damage and/or injury to personnel.

Chapter 8

Programming Fundamental Instructions

Objectives

This chapter shows how to program the fundamental instructions.

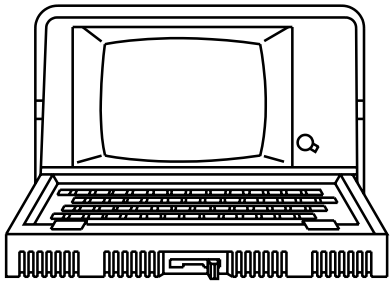
In this chapter you will read how to:

- Enter rungs using relay type instructions.
- Insert an instruction into an existing rung.
- Remove an instruction from an existing rung.
- Edit an existing rung.

System Start Up

Power up your industrial terminal by turning the ON/OFF switch clockwise. The switch is located to the right of the screen (Figure 8.1).

Figure 8.1
Industrial Terminal



10697-I

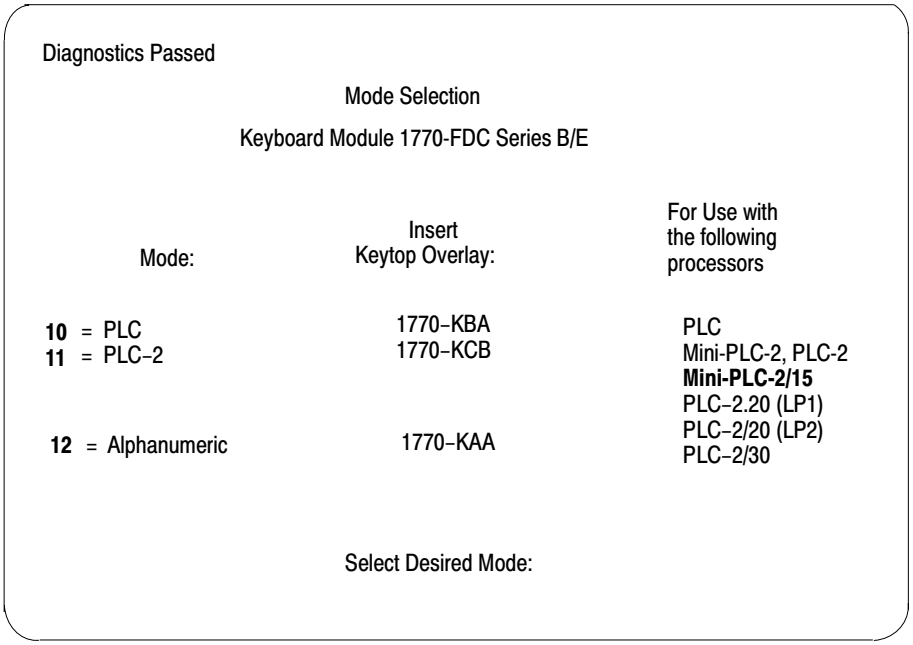
After a short while, the following display will appear:

Figure 8.2 illustrates the screen's display which you will see before you begin to insert your program. Press the keys 11 to start programming.

NOTE: The processor mode select switch must be in the PROG keyswitch position before you are able to insert your program.

Now you are able to insert your program. Proceed to section A.

Figure 8.2
Press the Keys 11 to begin Programming



Section A
Relay Type Instructions

Objectives

This section demonstrates how to enter each relay type instruction. Each rung is not a program. Do not use our examples using on-line production equipment. These examples are used only for demonstration purposes.

In this section you will read:

- How to enter relay type instructions.
- How to enter branch instructions.
- How to enter bit controlling instructions.

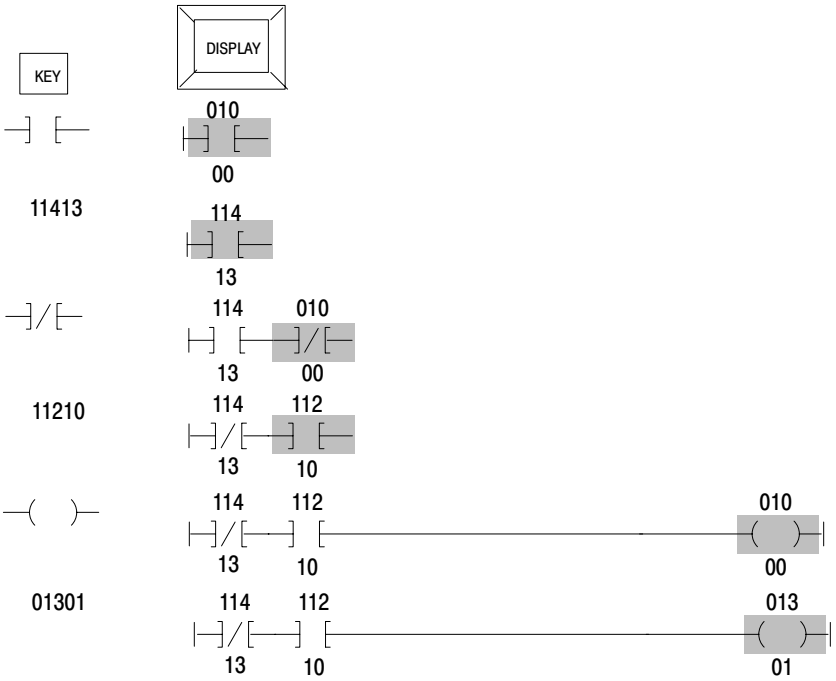
If you can do this, proceed to chapter 9.

Relay Type Instructions

Examine On, Examine Off, Energize

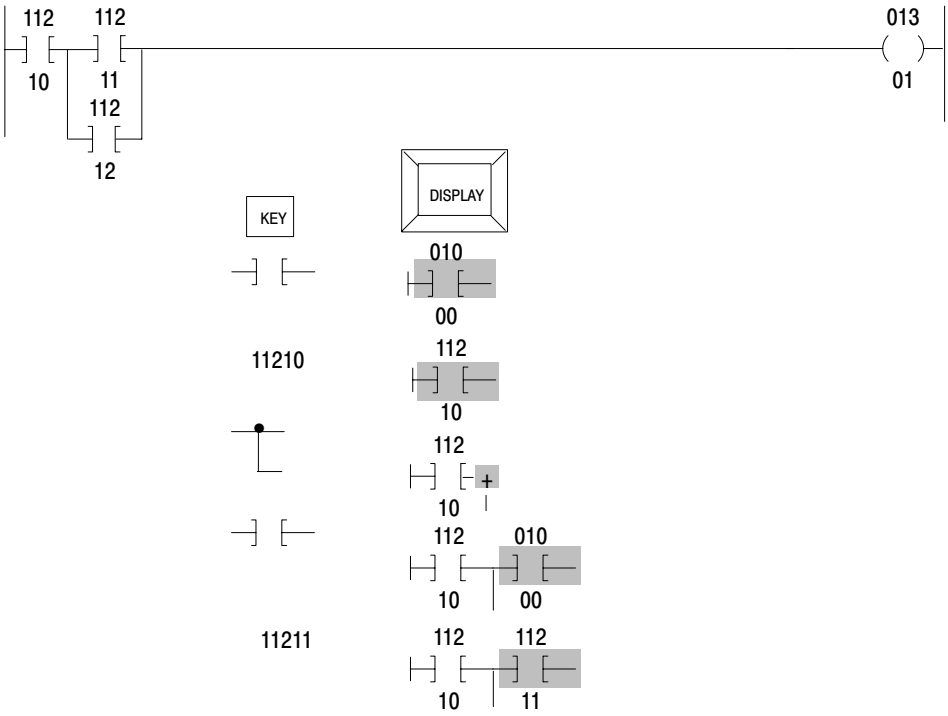
We will begin by entering this rung:

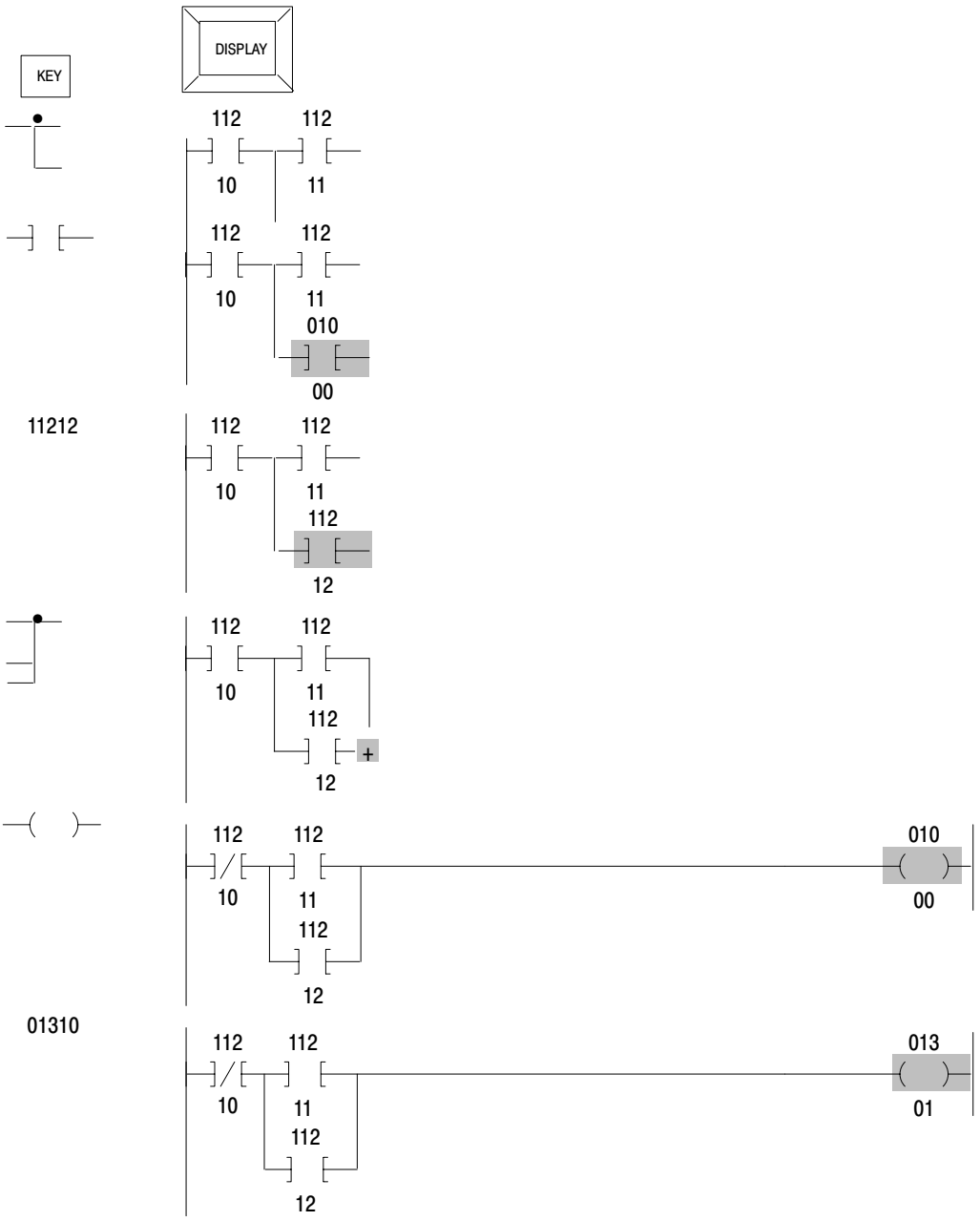
NOTE: The cursor is located where you see the intensified part of the rung.



Branch Instructions

We will enter this rung:

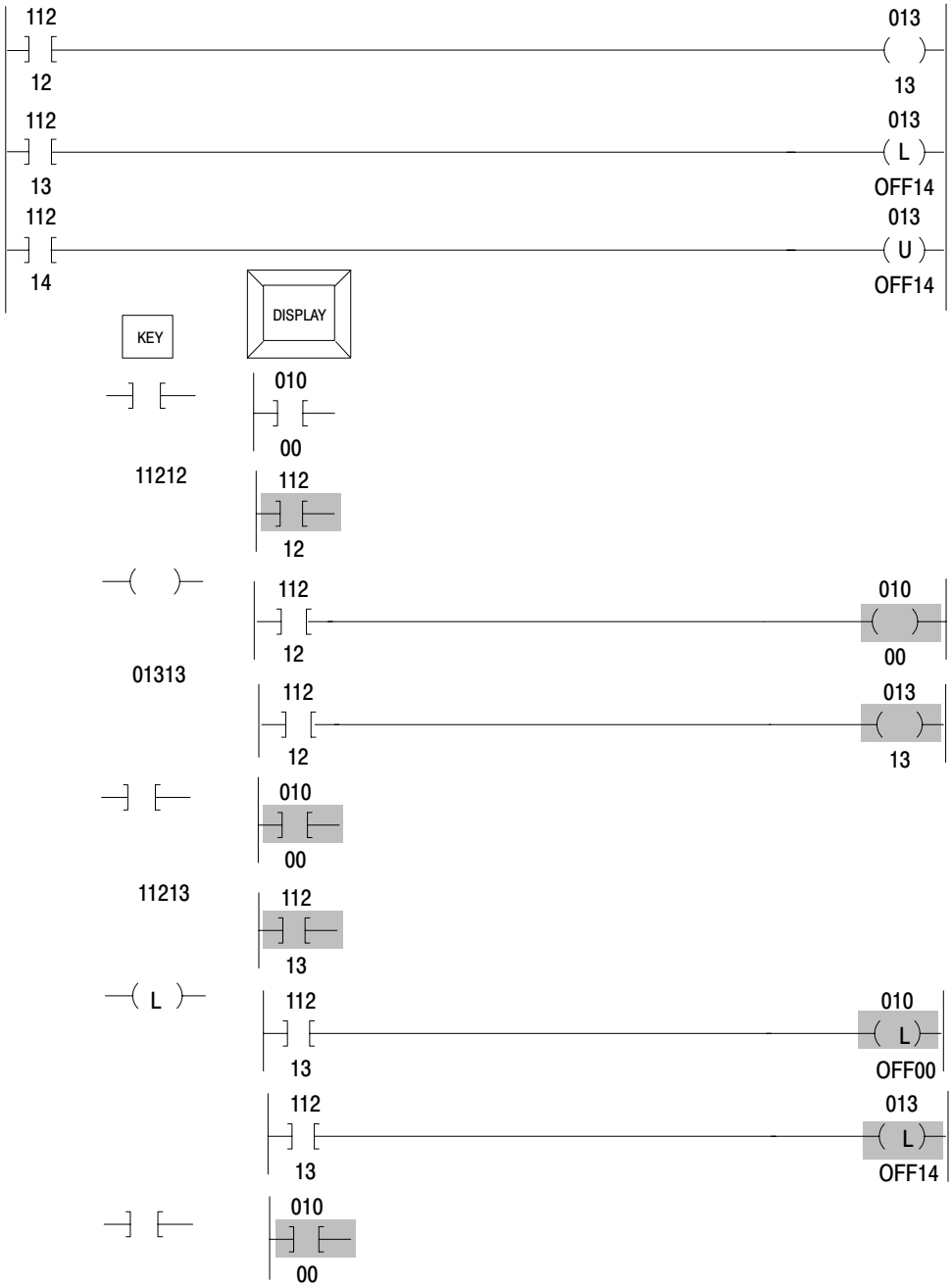


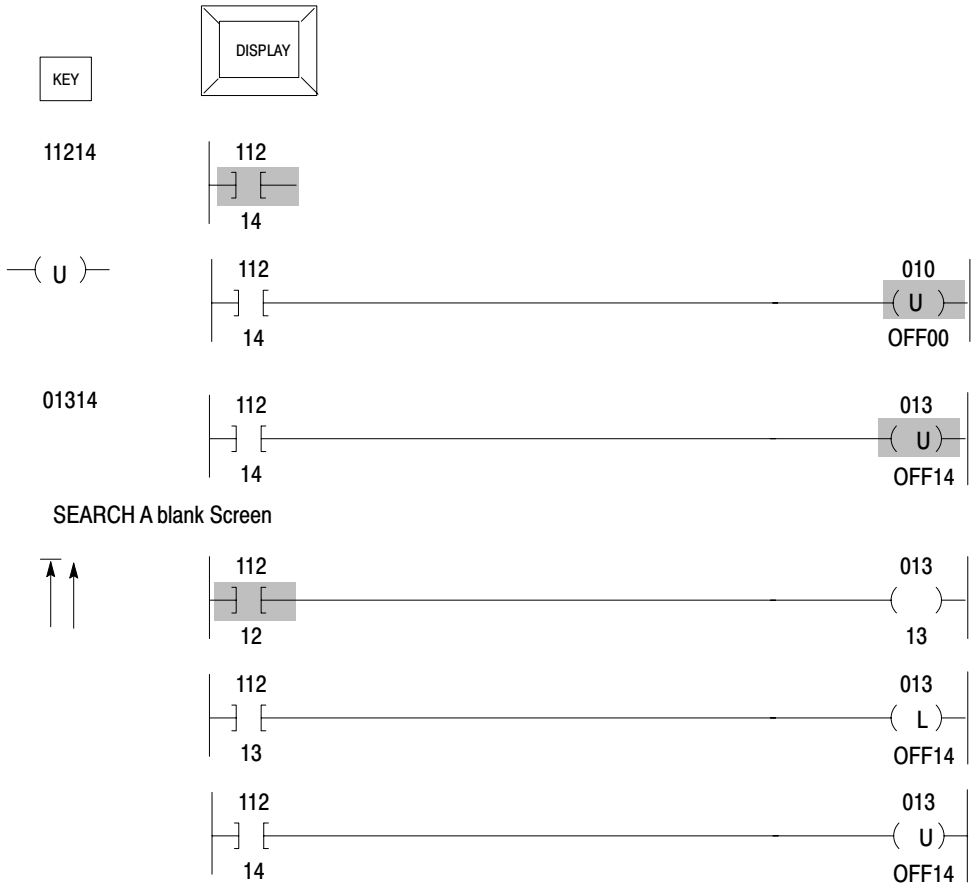


Bit Controlling Instructions

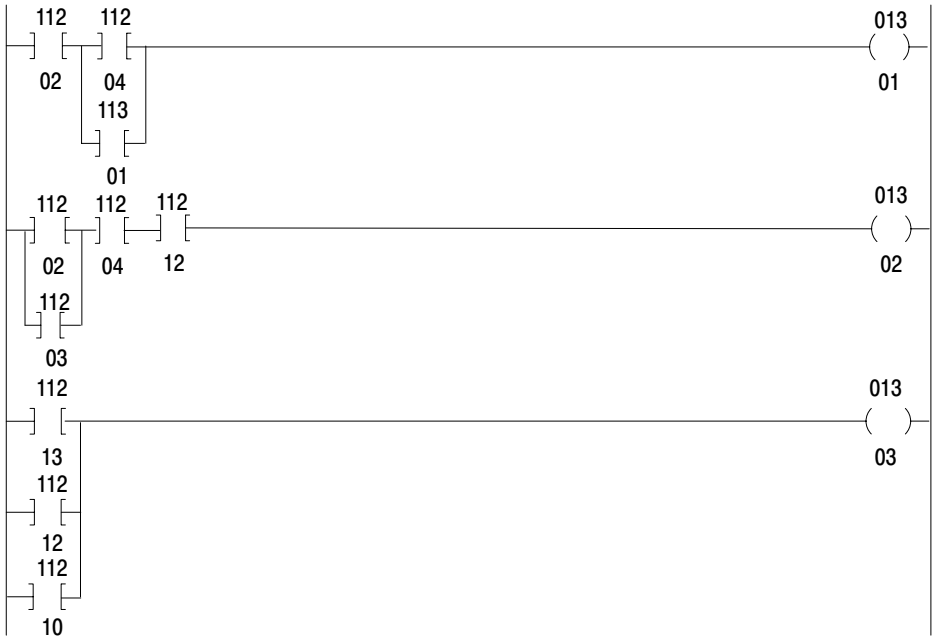
Energize, Latch, Unlatch

This is your final result:





Now practice inserting the following rungs:



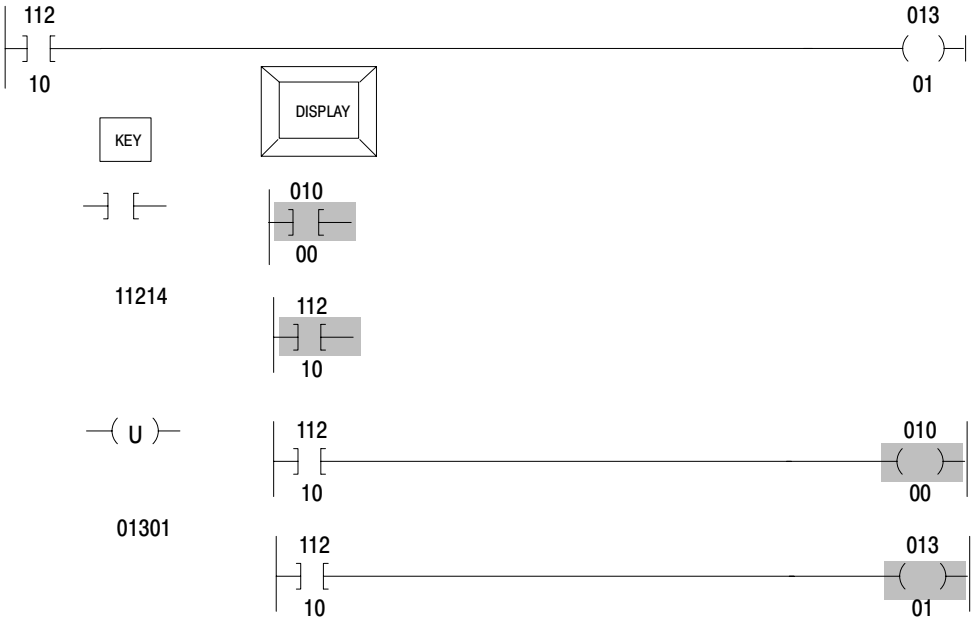
Section B
Editing Your Instructions

Objectives

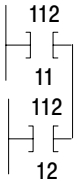
This section demonstrates:

- How to enter instructions.
- How to remove instructions from a rung.
- How to add rungs to your program.
- How to remove rungs from your program.

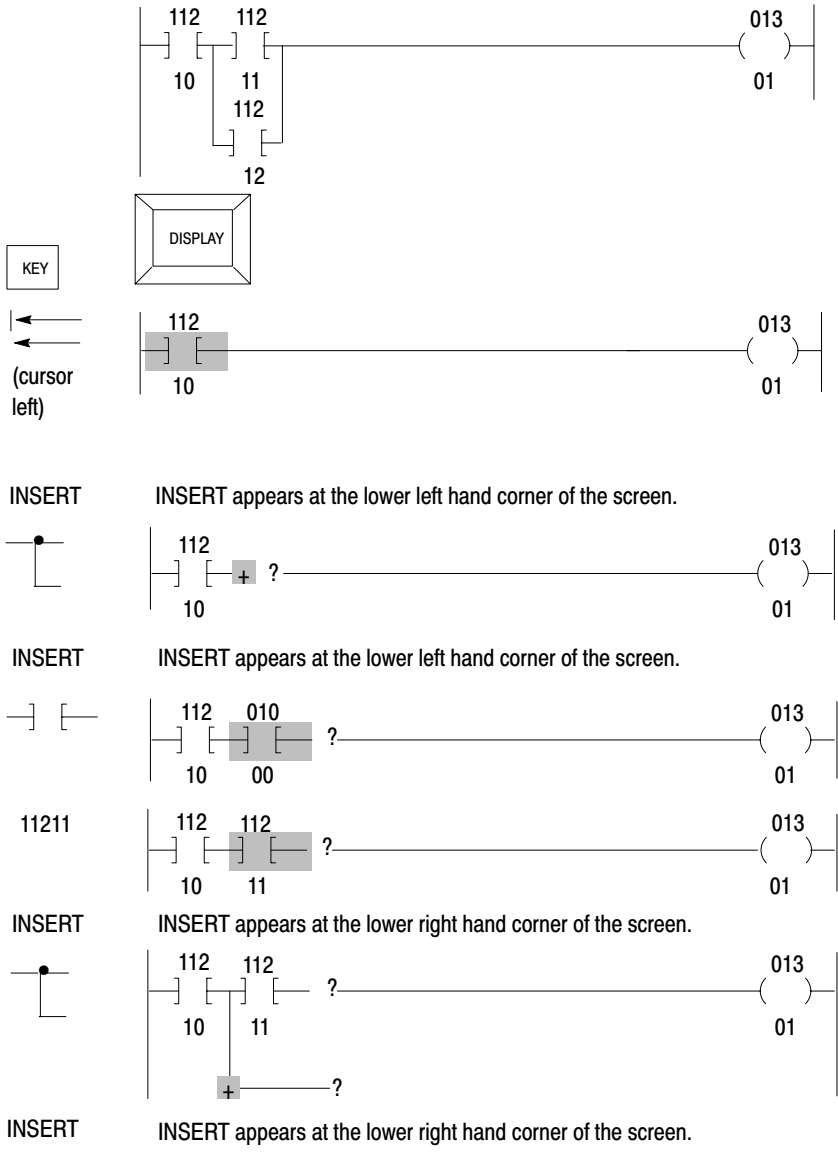
We will begin this section by entering this rung:

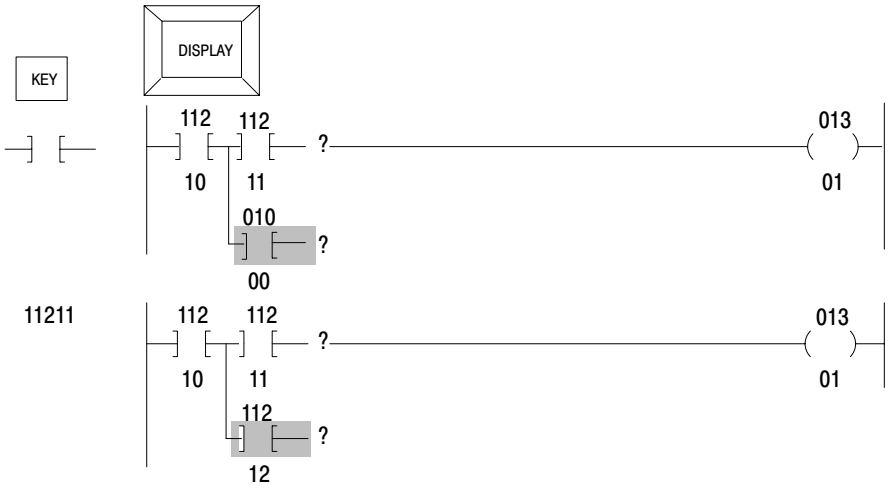


Now insert this branched instruction:



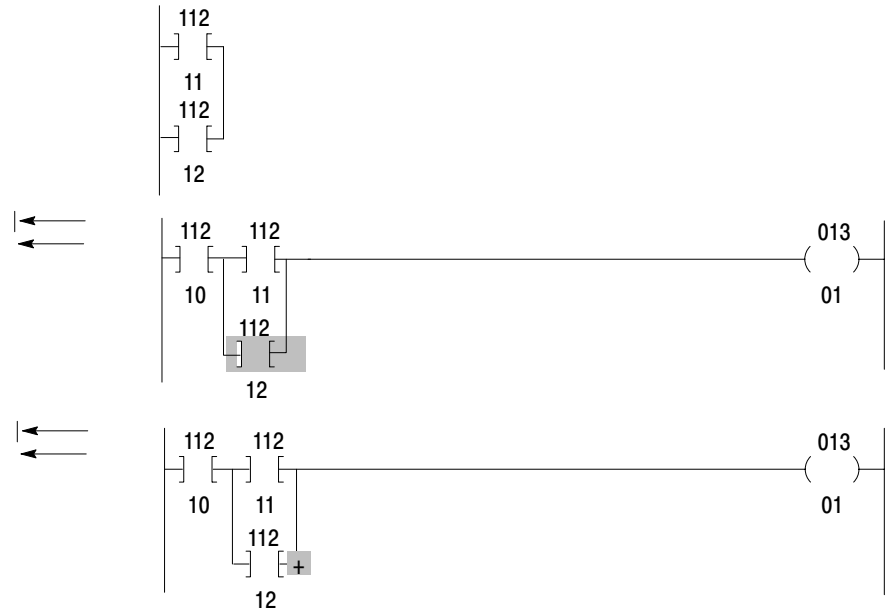
When you are finished it should look like this:



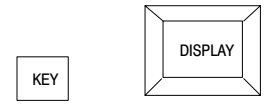


INSERT INSERT appears at the lower right hand corner of the screen.

Now we will remove this ranched instruction:

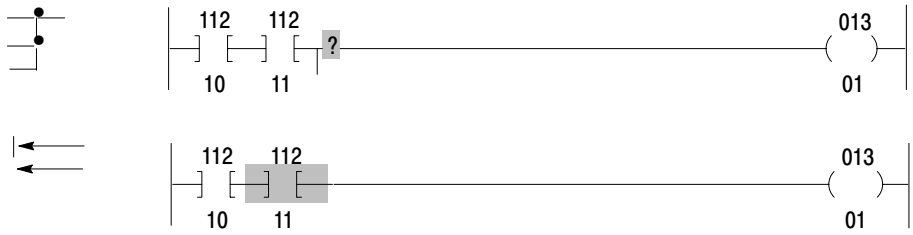


REMOVE REMOVE appears at the lower left hand corner of the screen.

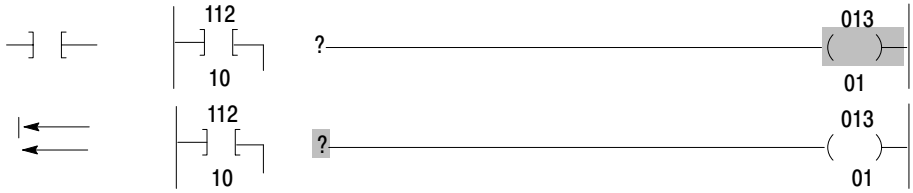


REMOVE REMOVE appears at the lower left hand corner of the screen.

REMOVE REMOVE appears at the lower left hand corner of the screen.



REMOVE REMOVE appears at the lower left hand corner of the screen.

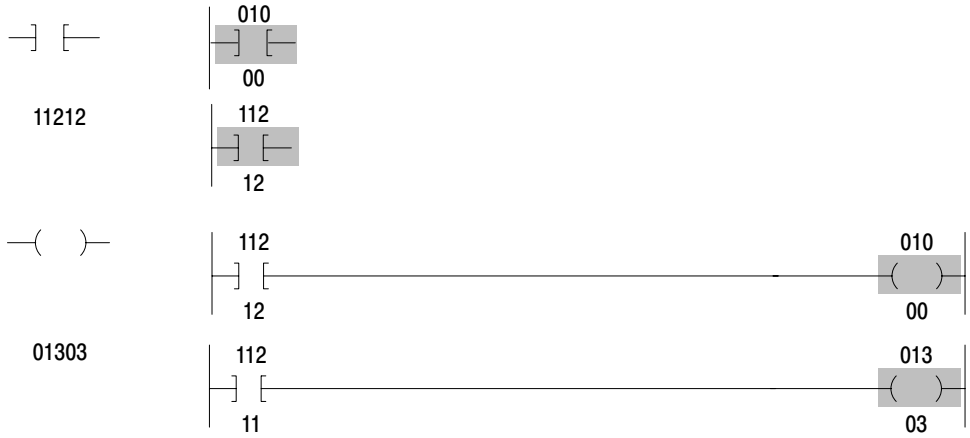


REMOVE REMOVE appears at the lower left hand corner of the screen.

Now we will add two new rungs to this existing rung:

You are done inserting the first new rung.

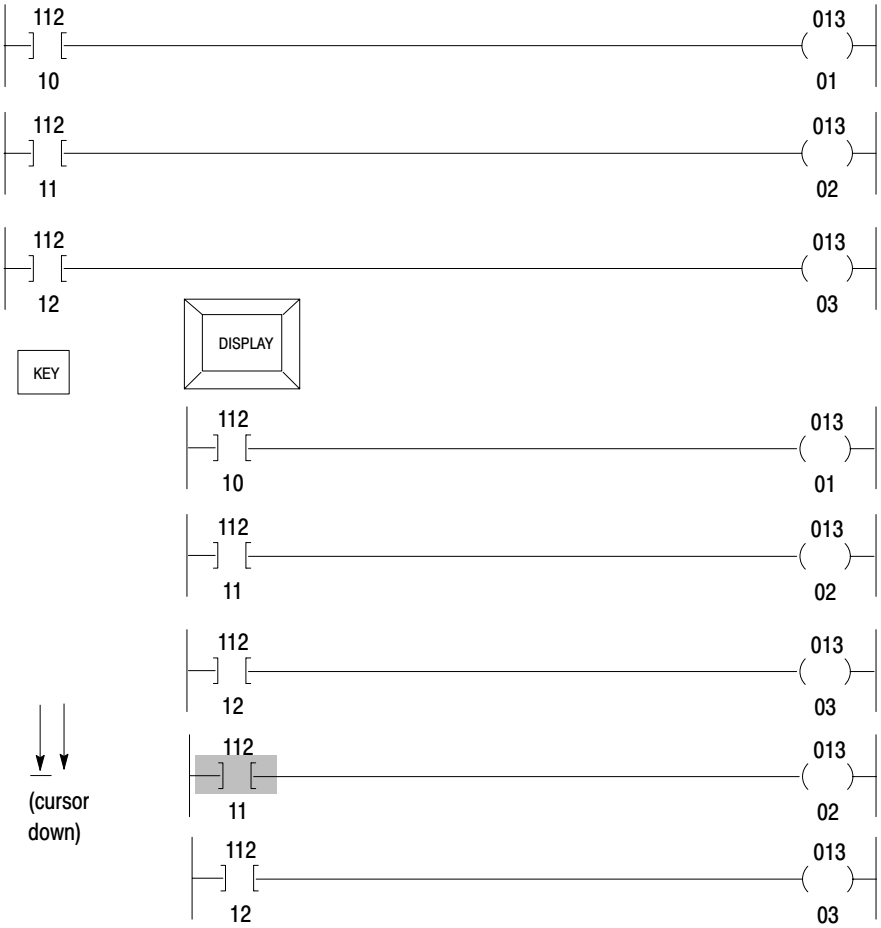
Let's insert the second rung:



This completes the insertion of the second rung.

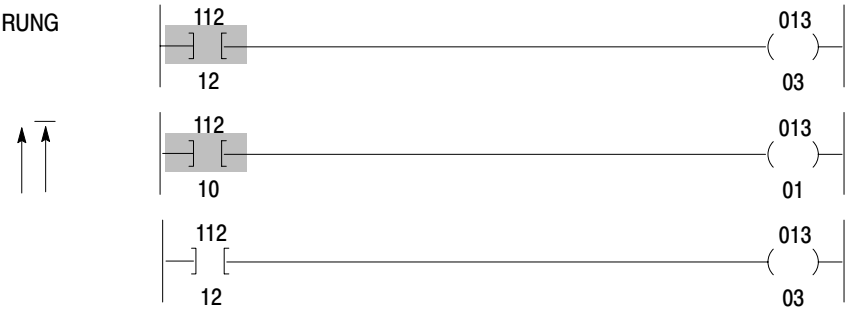
This is the complete program.

To remove the middle rung from the existing program.



REMOVE — REMOVE appears at the lower left hand corner of the screen.

RUNG

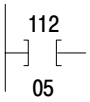


You removed the middle rung!

Practic by completing the exercie below:
Start by entering:

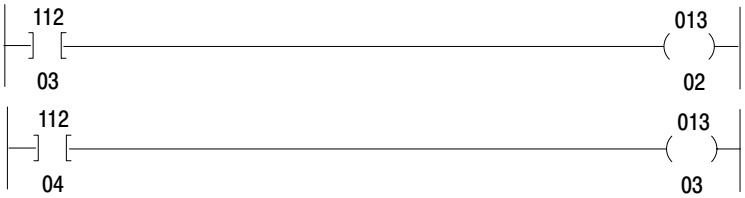
Then enter:

Now remove:



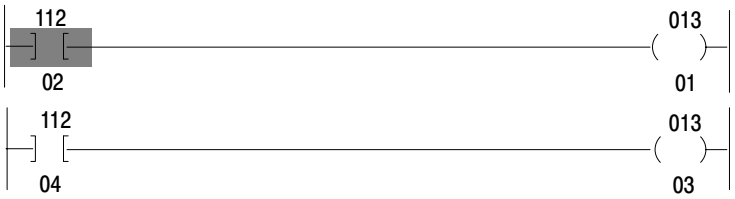
and

Add the following two rungs to your program:



Finally remove this rung:

Your final result should be:



Chapter 9

Programming Applications

Objectives

This chapter illustrates generic programming applications which demonstrate a suggested use for the following instructions:

- Timer/counters
- Arithmetic
- Data manipulation
- Program control

Do not program these examples using on-line production equipment. These examples are used only for demonstration purposes.

Application One

Refer to Figure 9.1.

This application illustrates the conversion of temperature from Celsius to Fahrenheit.

Suppose that a thermocouple is connected to a thermocouple input module which records the Celsius temperature of a motor bearing. For the operator's ease we would like to convert the recorded Celsius temperature in the data table to Fahrenheit values for display. This temperature must maintain certain range values for your application.

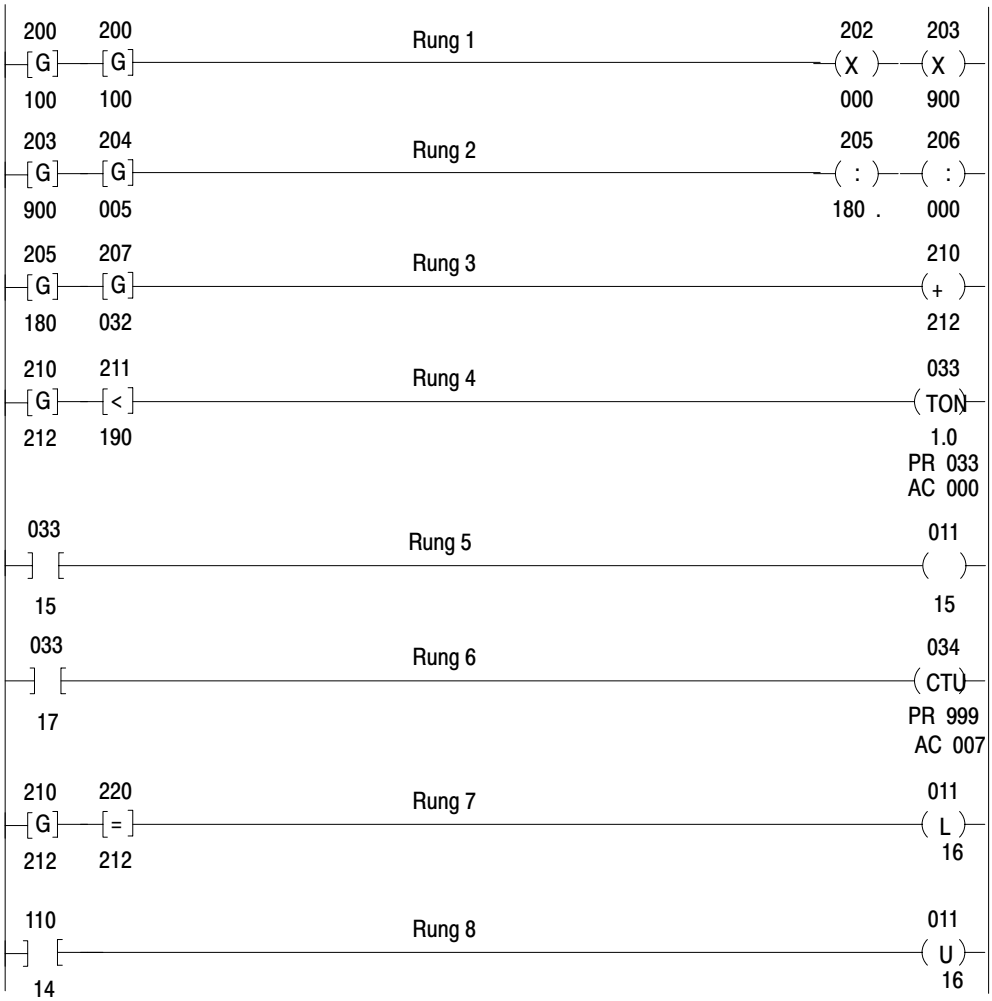
You would like to:

- Monitor the temperature between 87^0 to 100^0C
- Count the times the value falls below 190^0F
- Count the times the values stay at 212^0F

Now we will look at each rung:

Formula: F = (9/5 C) + 32

Figure 9.1
Converting Temperature Values



Rung 1: The get instruction at address 200 multiplies the temperature 100^0C by 9 and the result, 900 is stored in address 203.

Rung 2: The get instruction at address 203 divides 5 into 900 and stores the quotient, 180, in address 205.

Rung 3: The get instruction at address 207 adds 32 to the value 180 which is located at get address 205. The sum of 212 is stored at address 210. Thus 100^0C = 212^0F.

Rung 4: If the displayed temperature is less than 180^0F, the timer initiates timing for 3 seconds.

Rung 5: If 3 seconds have elapsed, an output at address 01115 will energize a heating device which will bring the temperature back into the desired range.

Rung 6: Counter 034 counts the number of times the value falls below 190^0F. Therefore, when rung 4 is true the counter increments.

Rung 7: When the temperature equals 212^0F, latch 11014 enables an alarm or an annunciator device.

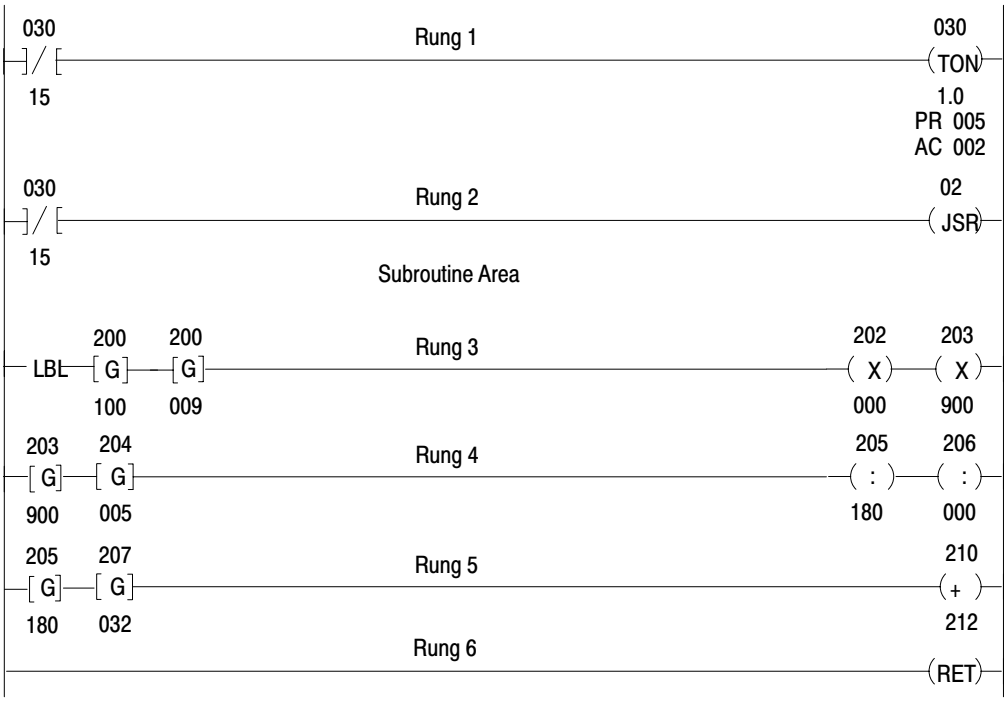
Rung 8: To unlatch the alarm, an operator would press a pushbutton connected to address 01116 that would shut the alarm off.

Application Two

Refer to Figure 9.2.

This application is similar to application one, but we are only recording the converted temperature reading every five seconds. There is an explanation of each rung:

Figure 9.2
Recording Temperature Values Every 5 Seconds



Rung 1: When rung 1 is true, the timer starts timing.

Rung 2: The JSR instruction jumps to the subroutine area label instruction when the timer's accumulated value reaches 5 seconds.

Rungs 3 thru 5: Convert Celsius temperature to Fahrenheit temperature exactly as in application one.

Rung 6: The return instruction signals the processor to return to the main program area.

> **CAUTION:** Allowances should be made for conditions which could be created by the use of the jump instruction. Jumped program rungs are not scanned by the processor so that input conditions are not examined and outputs that are controlled by these rungs remain in their last state. Timers and counters cease to function. Critical rungs should be reprogrammed outside the jumped section in the program zone.

Application Three

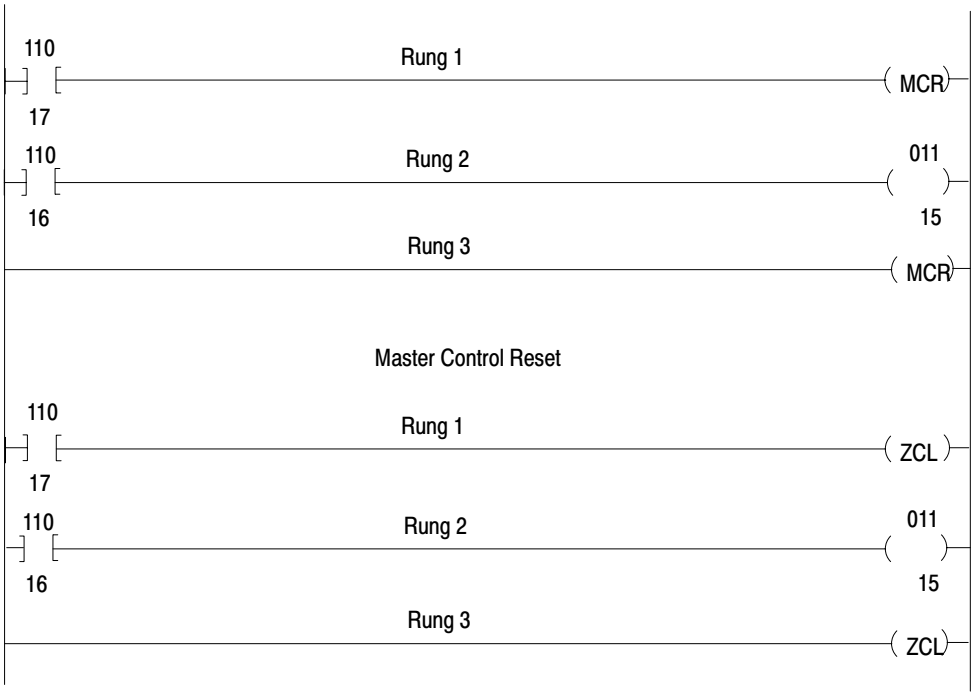
Refer to Figure 9.3.

This application illustrates the program control instructions, master control reset (MCR) and zone control last state (ZCL).

Common applications such as varying either packaged size or recipe ingredient use these instructions; packaging a product in two different sizes or converting a food product to a dietary food product by changing its sugar content would be examples of applications.

Before you program these two instructions you must think about how you would want your outputs to react when changing your process.

Figure 9.3
Program Control Instructions



Using the MCR instruction, rung logic would denote:

- If address 11017 is true, then address 01115 will work normally.
- If address 11017 is false, then address 01115 will be reset.

Using the ZCL instruction, rung logic would denote:

- If address 1117 is true, then address 01115 will work normally.
- If address 11017 is false, then address 01115 will be held in its last state.

Chapter 10

Block Format Instructions

Objectives

In this chapter you will read sections A thru C concerning:

- How to expand the data table.
- How to enter a file instruction.
- How to load data into the hexadecimal data monitor display.
- How to edit your file data.
- How to document a sequencer input and output instruction.
- How to enter a sequencer input and output instruction.
- How to load data into the binary data monitor display.

Section A
File Instruction Programming

Objectives

In this section you will read:

- How to expand the data table.
- How to enter a file instruction.
- How to load data into a hexadecimal data monitor display.

File Instructions

As you recall from chapter 6, there are three types of file instructions:

- File to file move
- File to word move
- Word to file move

We will show you how to enter the file to file move instruction, but the syntax is the same for each file instruction.

Before you begin:

Turn the keyswitch to the program position (PROG).

As we stated in chapter 4, you must expand the data table to provide additional space for files. To do this:

KEY	DISPLAY
SEARCH	No display is on the screen
50	DATA TABLE CONFIGURATION

NUMBER OF 128–WORD D.T. BLOCKS	02
NUMBER OF INPUT/OUTPUT RACKS	2
NUMBER OF T/C (if applicable)	104
DATA TABLE SIZE	256

The above chart shows a factory configured data table.

The following chart will help you adjust your data table size:

Enter	Data Table Size
01	128
02	256
03	384
04	512
05	640
10	1280
15	1920

After you have adjusted the data table, press [CANCEL COMMAND] and we'll continue.

NOTE: Other industrial terminal commands are summarized in the Quick Reference section of this manual.

This is your end result:

KEY

DISPLAY

11205

010
00
112
05

FILE — Screen does not change.

HELP

File Instructions
10 = FILE TO FILE MOVE
11 = WORD TO FILE MOVE
12 = FILE TO WORD MOVE

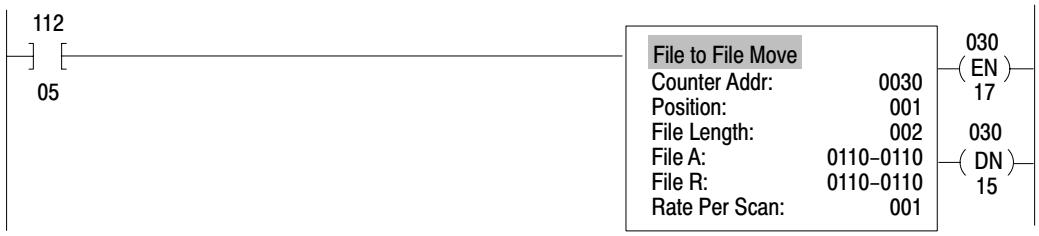
NOTE: FILE A = SIOURCE 1 FILE;
FILE R = RESULT FILE

SELECT THE DESIRED INSTRUCITON:

We will select the file to file move instruction:

10

(This is our desired instruction)

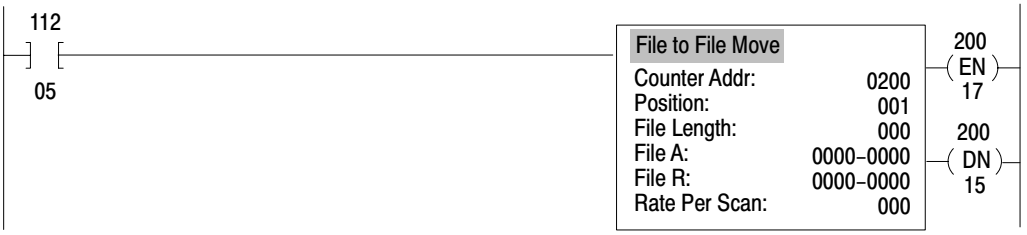


Notice that the cursor is now on the first digit of the counter address. Also, the above display shows all default values.

Let’s fill in each instruction value.

0200

Note: When you expand the data table you must keyp in a 4 place counter address.



KEY DISPLAY

Now the cursor is on the first digit of the file length.

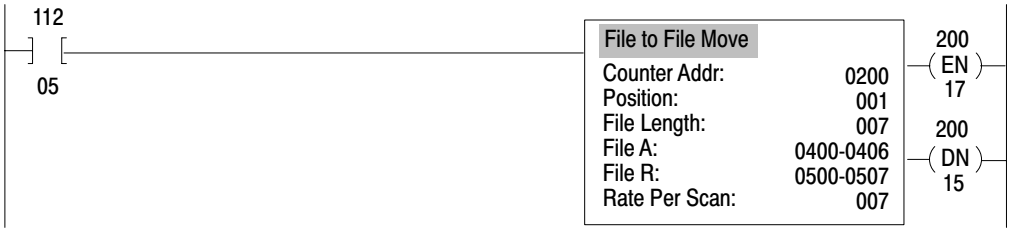
The cursor moved to the first digit of file A.

The cursor moved to the first digit of file R.

Finally, the cursor is on the first digit of the rate per scan.

You can now proceed to add data to your file by using the data monitor display. **Do not** press [CLEAR MEMORY].

DATA MONITOR DISPLAY



Position your cursor on the words FILE TO FILE MOVE. Use the arrow keys to move your cursor.

For this example we will use the hexadecimal data monitor display.

KEY	DISPLAY
DISPLAY	The screen does not change
1	

```
----------------------------------------------------------------------------------

                          HEXADECIMAL DATA MONITOR
                            FILE TO FILE MOVE

COUNTER ADDR:  200          POSITION:  001          FILE LENGTH:  007
FILE A:  400-406                                    FILE R:  500-506

POSITION                    FILE A DATA             FILE R DATA
001                         0000                    0000
002                         0000                    0000
003                         0000                    0000
004                         0000                    0000
005                         0000                    0000
006                         0000                    0000
007                         0000                    0000


                            DATA :  0000
```

NOTE: If you wanted to enter binary information press [DISPLAY][0].

We will now enter data in position 001.

KEY

1257

DISPLAY

```
                                HEXADECIMAL DATA MONITOR
                                  FILE TO FILE MOVE
COUNTER ADDR:   200             POSITION:  001          FILE LENGTH:   007
FILE A:   400-406                                       FILE R:   500-506

POSITION                        FILE A DATA             FILE R DATA
001                             0000                    0000
002                             0000                    0000
003                             0000                    0000
004                             0000                    0000
005                             0000                    0000
006                             0000                    0000
007                             0000                    0000

                                DATA :   1257
```

NOTE: If you made a mistake you can correct it by moving the cursor to the incorrect number and then pressing the correct number key. Then proceed...

INSERT

```
                                HEXADECIMAL DATA MONITOR
                                  FILE TO FILE MOVE
COUNTER ADDR:   200             POSITION:  001          FILE LENGTH:   007
FILE A:   400-406                                       FILE R:   500-506

POSITION                        FILE A DATA             FILE R DATA
001                             1257                    0000
002                             0000                    0000
003                             0000                    0000
004                             0000                    0000
005                             0000                    0000
006                             0000                    0000
007                             0000                    0000

                                DATA :   0000
```

This information (1257) now appears in position 001 of file A. The cursor at the bottom of the screen is now at 1257.

We will now add information to position 002.

KEY

0721

DISPLAY

HEXADECIMAL DATA MONITOR
FILE TO FILE MOVE

COUNTER ADDR: 200
FILE A: 400–406

POSITION: 001

FILE LENGTH: 007
FILE R: 500–506

POSITION	FILE A DATA	FILE R DATA
001	1257	0000
002	0000	0000
003	0000	0000
004	0000	0000
005	0000	0000
006	0000	0000
007	0000	0000

DATA : 0721

The cursor is on the value 1.

KEY

INSERT

DISPLAY

HEXADECIMAL DATA MONITOR
FILE TO FILE MOVE

COUNTER ADDR: 200
FILE A: 400–406

POSITION: 001

FILE LENGTH: 007
FILE R: 500–506

POSITION	FILE A DATA	FILE R DATA
001	1257	0000
002	0721	0000
003	0000	0000
004	0000	0000
005	0000	0000
006	0000	0000
007	0000	0000

DATA : 0000

Proceed by loading each position of file A with following data:

POSITION 003 0879
POSITION 004 0162
POSITION 005 1982
POSITION 006 9715
POSITION 007 5761

Now file A is loaded.

KEY	DISPLAY
Press a corresponding input address 11205: using an Input/Output Simulator (cat. no. 1790–DP). Notice that the data in file A transferred to file R.	HEXADECIMAL DATA MONITOR FILE TO FILE MOVE COUNTER ADDR: 200 FILE A: 400–406 POSITION: 001 FILE LENGTH: 007 FILE R: 500–506 POSITION 001 002 003 004 005 006 007 FILE A DATA 1257 0721 0879 0162 1982 9715 5761 FILE R DATA 1257 0721 0879 0162 1982 9715 5761 DATA : 0000

NOTE: You do not have to enter data in each position. you can skip position numbers.

Do not clear your controller's memory. We will use this data to demonstrate new concepts in section B.

Section B
Editing a File

Objectives

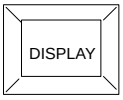
In this section you will read:

- How to edit your file's data in the hexadecimal data monitor display.

Let's Begin

Your screen should look like this:

KEY



DISPLAY

The screen does not change

1

HEXADECIMAL DATA MONITOR
FILE TO FILE MOVE

COUNTER ADDR: 200
FILE A: 400–406

POSITION: 001

FILE LENGTH: 007
FILE R: 500–506

POSITION	FILE A DATA	FILE R DATA
001	1257	1257
002	0721	0721
003	0879	0879
004	0162	0162
005	1982	1982
006	9715	9715
007	5761	5761

DATA : 0000

Notice that you now see a command buffer at the bottom of the screen. it is labeled DATA: 1257. This is also the same number in FILE A at POSITION 001.

Press [↓↓] until the cursor is at POSITION 004.

We will change 0162 to 0281.

0281

HEXADECIMAL DATA MONITOR
FILE TO FILE MOVE

COUNTER ADDR: 200
FILE A: 400–406

POSITION: 001

FILE LENGTH: 007
FILE R: 500–506

POSITION	FILE A DATA	FILE R DATA
001	1257	1257
002	0721	0721
003	0879	0879
004	0162	0162
005	1982	1982
006	9715	9715
007	5761	5761

DATA : 0281

KEY

INSERT

DISPLAY

```
                                HEXADECIMAL DATA MONITOR
                                  FILE TO FILE MOVE
COUNTER ADDR:   200             POSITION:  001
FILE A:   400-406
                                                        FILE LENGTH:   007
                                                        FILE R:   500-506
POSITION
001                             FILE A DATA             FILE R DATA
002                             1257                    1257
003                             0721                    0721
004                             0879                    0879
005                             0281                    0162
006                             1982                    1982
007                             9715                    9715
                                5761                    5761
                                DATA :   0281
```

Notice that FILE R's POSITION has not changed.

Not practice by changing the data in POSITION 006 of file A to 7777.

Start by moving your cursor to POSITION 006.

NOTE: If you wanted to change the data in FILE R you would follow the same procedure. To move your cursor over to FILE R, press [SHIFT][→], then follow the same procedure.

If you understand sections A and B, proceed to section C and we will show you how to document and program a sequencer input and output instruction.

Section C
Documenting A Sequencer Instruction

Objectives

In this section you will read:

- How to document a sequencer input and output instruction.
- How to enter a sequencer input and output instruction.
- How to load data into the binary data monitor display.

Good documentation is the key to any successful programming operation. Read the areas titled "Mask" and "Programming Limitations" before you read our bottle filling application example. After the application example, we will demonstrate how to program sequencer input and output instructions.

Mask

A special programming technique called a “mask” is used with the sequencer instructions. By masking your bits you will be able to conserve bits and use them for storage.

Definition

A mask is a means of selectively screening out data. The purpose of the mask is to allow unused bits in the specific instruction to be used independently. For example, the number of output bits required for sequential operation can be any integer up to 64. When fewer than 64 outputs are required, masking allows the unused output terminals of the module that is controlled by the sequencer instruction to be used to control output devices which are independent to the sequencer operation.

A 0 in a mask bit location prevents the instruction from operating on the data in the corresponding bit location. A 1 in a mask bit location allows the corresponding bit to be operated. When all the output data bits are relevant to the instruction, use a mask of all 1s.

Other instructions can control a mask in the user program. If a changing mask is required for different steps in the sequencer operation,use a get-put or file-to-file move.

> **WARNING:** When choosing a mask word address, be sure that the next 1, 2, or 3 consecutive word addresses are not already assigned. Other data written into a mask could cause unpredictable machine operation. This could cause damage to your equipment and/or injury to your personnel.

Programming Limitations

Sequencer instructions are powerful tools when programming your operations. But, like all good tools, there are some limitations:

- Two events can not run simultaneously.
- If one sequencer instruction is out of order, then your process stops. You cannot continue to a different process.
- The logic of a sequencer instruction is usually programmed using “AND” logic. You can use “OR” logic but you will use more memory space.
- A sequencer load instruction can be programmed alone but used only in the incremental mode.

- When programming a sequencer input with a sequencer output instruction, the counter address for both instructions may be the same.

Bottle Filling Application

This application starts when a bottle is placed on a conveyor, it ends when the bottle is filled and ready for the next sequence of operations. This application is totally automated.

Again, we emphasize that this application is only for demonstration purposes. **Do not** try to program this application using your on-line production equipment.

Documenting Your Program

Before we illustrate how to program the bottle filling application, here is a list of tasks that would lead you to practice good documentation habits.

Task 1: Write out the sequence of operation that would explain the production process.

Task 2: Make two lists: one for input devices and one for output devices.

Task 3: Complete the sequence worksheets which are located in this chapter. They are figures 10.2 and 10.2. Additional worksheets are available through your local Allen-Bradley distributor or sales representative.

Task 4: Write out your processor program using the sequencer instructions.

Task 5: Program your processor. Test out the program then place your worksheets and all related information in a notebook for future reference.

Task 1: Sequence of Operation

1. Four bottles are placed at the beginning of a moving conveyor.
2. The bottles are at station one ready to be filled.
3. Each bottle actuates a photocell indicating that each bottle is present.
4. One fill tube is inserted into each bottle.
5. The file tubes fill each bottle for 3 seconds.
6. The file tubes are removed from each bottle.
7. A solenoid moves the bottles to the next station.

Task 2: List Your Devices

Input Devices

The input devices and their abbreviations are:

Input Device	Abbreviation	Comment
Photocell	PC	Bottle in Place
Fill tube extended	LS1	Limit Switch
Fill tube retracted	LS2	Limit Switch
Automated	Auto	Type of Operation
Timer	Timer	3 Seconds

Output Devices

The output devices and their abbreviations are:

Output Device	Abbreviation	Comment
Conveyor motor	CM	Initializing motion
Conveyor motion forward	CMF	
Fill tube motor	FTM	
Fill tube forward	FTF	
Fill tube filling	FTS	Fluid starts to flow
Fill tube reverse	FTR	
Solenoid	SOL	Moves the bottles off the conveyor

Task 3: Completing Your Worksheets

Figure 10.1 and Figure 10.2 illustrate completed sequencer worksheets. Notice that the first step of the sequencer output is the last step of your operation. Table 10.A describes each step. To aid you in understanding this documentation concept, read Table 10.A while looking at each figure.

Table 10.A

NOTE: Read this table from left to right while looking at Figure 10.1 and Figure 10.2.

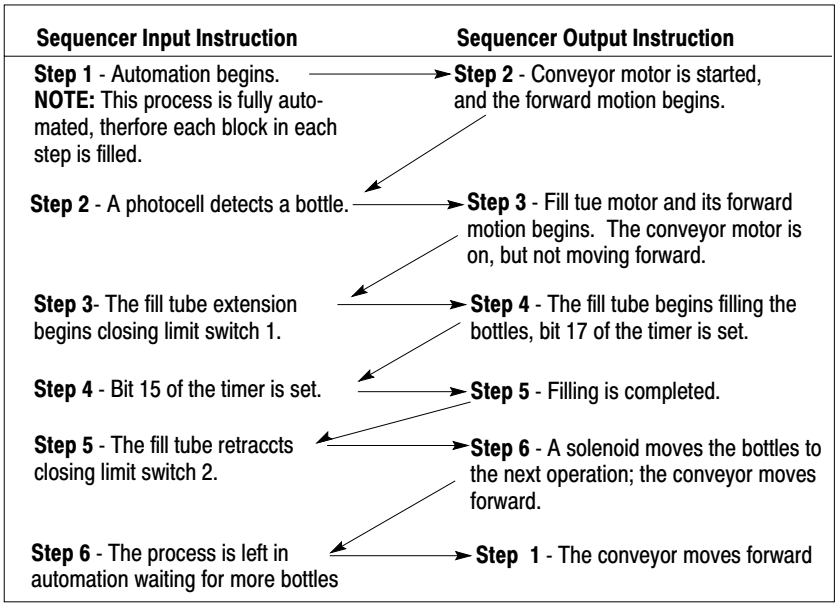


Figure 10.1
Completed Sequencer Input Worksheet

ALLEN-BRADLEY
Programmable Controller
Data Table MAP (128-word)
(Publication 5048 - November, 1983)

PAGE 1 OF 2

PROJECT NAME Bottle Filling Applications PROCESSOR Mini-PLC-2/15 Series B

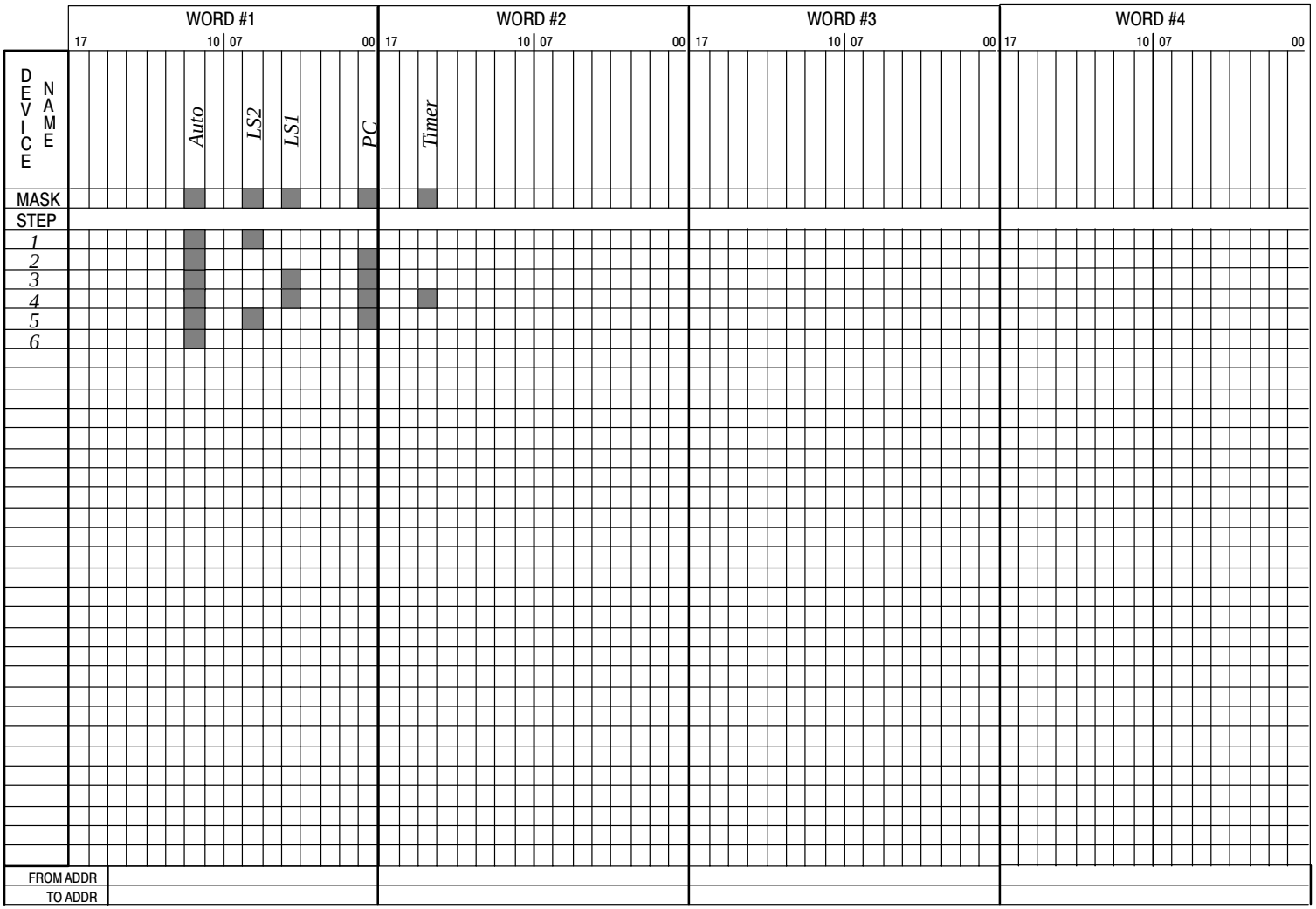
DESIGNER Engineer DATA TABLE ADDR _____ TO _____

SEQUENCER Input

COUNTER ADDR: 200 FILE 400 TO 413 SEQ LENGTH: 006

WORD ADDR: 110 Timer 200

MASK ADDR: 070 071



Note: A filled-in box means that each device is actuated

10145-I

Figure 10.2
Completed Sequencer Output Worksheet

ALLEN-BRADLEY
Programmable Controller
Data Table MAP (128-word)
(Publication 5048 - November, 1983)

PAGE 2 OF 2

PROJECT NAME Bottle Filling Applications PROCESSOR Mini-PLC-2/15 Series B

DESIGNER Engineer DATA TABLE ADDR _____ TO _____

SEQUENCER Output

COUNTER ADDR: 200 FILE 600 TO 613 SEQ LENGTH: 006

WORD ADDR: 012

MASK ADDR: 075

DEVICE NAME	WORD #1 17–10	WORD #1 07–00	WORD #2 17–10	WORD #2 07–00	WORD #3 17–10	WORD #3 07–00	WORD #4 17–10	WORD #4 07–00
		ETS ETF SOL FTR FTM CMT CM	Timer					
MASK		■ ■ ■ ■ ■ ■ ■ ■	■					
STEP								
1		■ ■						
2		■ ■						
3		■ ■						
4		■ ■						
5		■ ■						
6		■ ■						
FROM ADDR								
TO ADDR								

Note: A filled-in box means that each device is actuated

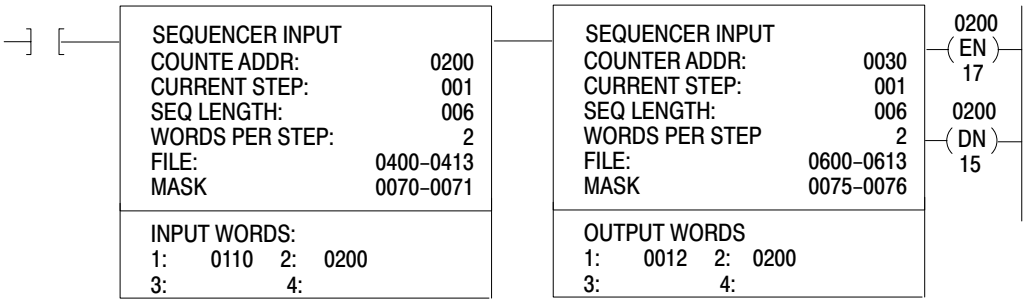
10148-I

Task 4: Processor Instruction Program

Figure 10.3 is an example of a program rung which represents your worksheets.

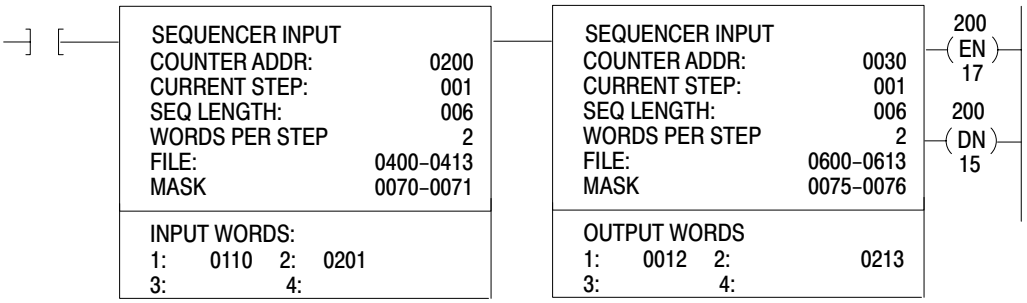
Figure 10.3
Program Rung Example

Automation

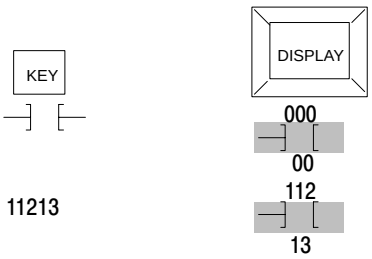


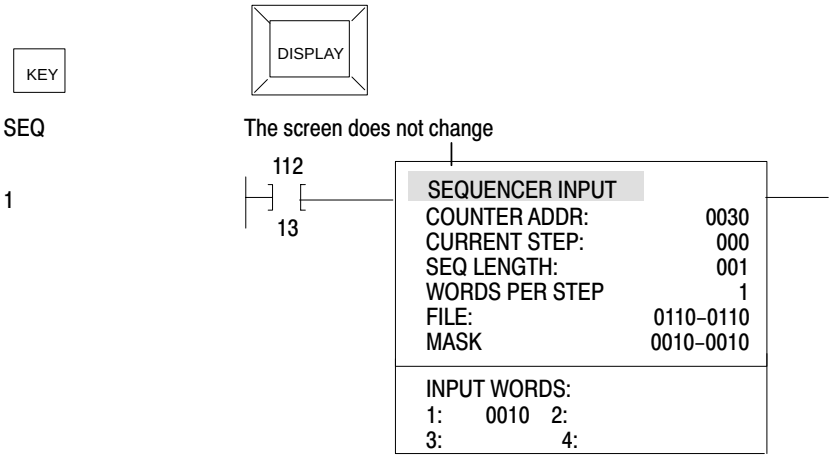
Task 5: Programming Your Processor

Start by expanding your data table. Refer to page 10-2 for the data table size values or press [SEARCH][5][0]. After you adjust the data table press [CANCEL COMMAND]. You are now ready to insert your program:



This is your finished rung.



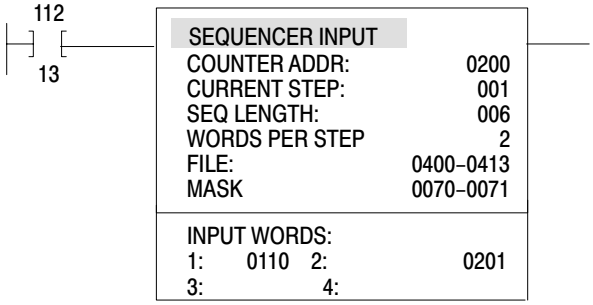


Notice that the words SEQUENCER INPUT are flashing, the cursor is on the first digit of counter address, and the default values are shown.

Insert the following values. your cursor will move automatically throughout the block instruction. The values are:

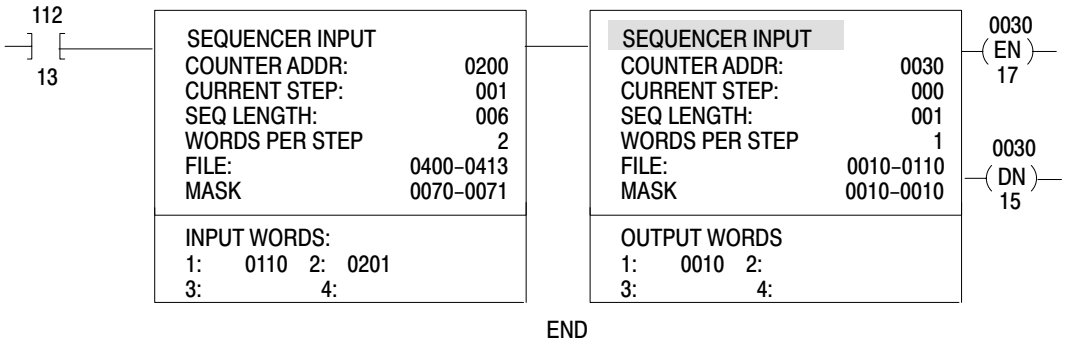
CURRENT ADDR: 0200
CURRENT STEP: 001
SEQ LENGTH: 006
WORDS PER STEP: 2
FILE: 0400
MASK: 0070
INPUT WORDS:
1:0110 2:0201

Your completed block instruction should look like this:



We will continue to insert data for the sequencer output instruction.

KEY	DISPLAY
SEQ	The screen does not change
0	

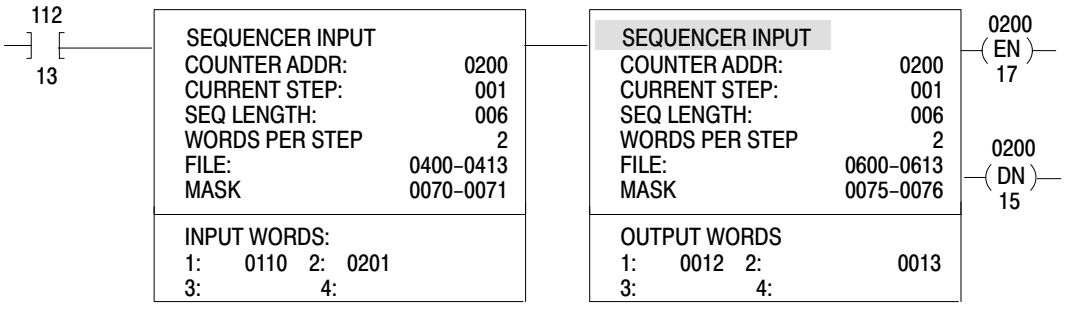


The words SEQUENCER OUTPUT are flashing, the cursor is on the first digit of the counter address, and the default values are shown.

Insert the following values. Your cursor will move automatically throughout the block instruction. The values are:

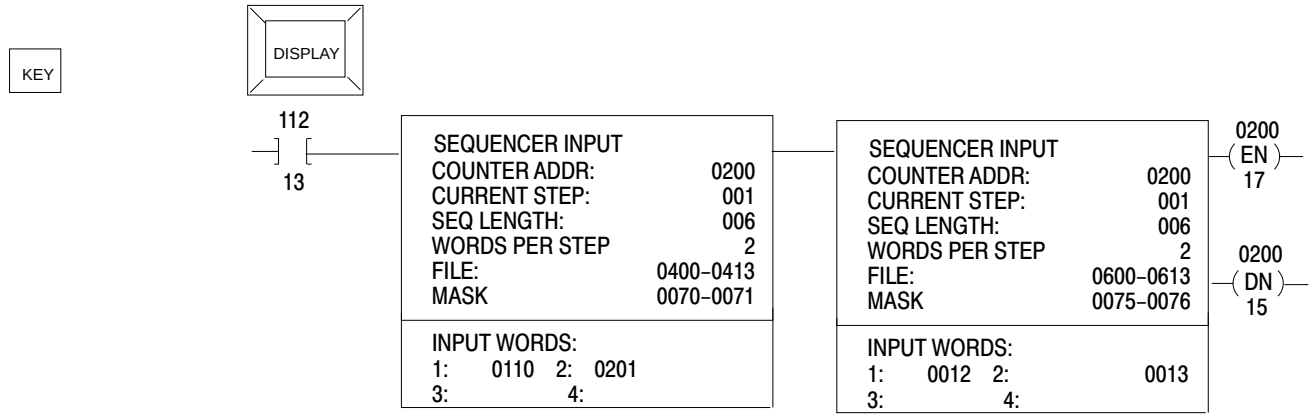
CURRENT ADDR:	0200
CURRENT STEP:	001
SEQ LENGTH:	006
WORDS PER STEP:	2
FILE:	0400
MASK:	0075
INPUT WORDS:	
1:0110 2:0201	

Your completed block instruction should look like this:



Let's continue and load your data into the binary data monitor mode for each sequencer instruction. You will get your data from your worksheets (Figure 10.1 and Figure 10.2). A filled in block means that a 2 will be inserted in a corresponding bit position.

Start by positioning your cursor on the words SEQUENCER INPUT. Use the cursor control keys to move the cursor.



Let's go on...

DISPLAY The screen does not change.

0

BINARY DATA MONITOR
SEQUENCER INPUT

COUNTER ADDR: 02004 STEP: 001 SEQUENCER LENGTH: 006

FILE: 600–610

INPUT ADDR: 110 201
DATA: 00000000 00000000 00000000 00000000

MASK ADDR: 070 071
DATA: 00000000 00000000 00000000 00000000

STEP	WORD 1		WORD 2	
001	00000000	00000000	00000000	00000000
002	00000000	00000000	00000000	00000000
003	00000000	00000000	00000000	00000000
004	00000000	00000000	00000000	00000000
005	00000000	00000000	00000000	00000000
006	00000000	00000000	00000000	00000000

DATA: 00000000 00000000

PROGRAM MODE

KEY

DISPLAY

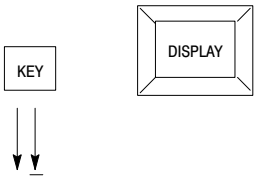
0000001000000000

```
                                BINARY DATA MONITOR
                                SEQUENCER INPUT
COUNTER ADDR:  200                  STEP:  001                  SEQUENCER LENGTH:  006
                                FILE:  900–913

INPUT ADDR:  110                                                    201
DATA:          00000000   00000000   00000000   00000000

MASK ADDR:  070                                                     071
DATA:          00000000   00000000   00000000   00000000

STEP          WORD 1                          WORD 2
001        00000000   00000000   00000000   00000000
002        00000000   00000000   00000000   00000000
003        00000000   00000000   00000000   00000000
004        00000000   00000000   00000000   00000000
005        00000000   00000000   00000000   00000000
006        00000000   00000000   00000000   00000000
                          DATA:  00000010   00000000
PROGRAM MODE
```

INSERT

```
                                BINARY DATA MONITOR
                                SEQUENCER INPUT
COUNTER ADDR:  200                  STEP:  001                  SEQUENCER LENGTH:  006
                                FILE:  0400–0413

INPUT ADDR:  110                                                    201
DATA:          00000000   00000000   00000000   00000000

MASK ADDR:  070                                                     071
DATA:          00000000   00000000   00000000   00000000

STEP          WORD 1                          WORD 2
001        00000010   00000000   00000000   00000000
002        00000000   00000000   00000000   00000000
003        00000000   00000000   00000000   00000000
004        00000000   00000000   00000000   00000000
005        00000000   00000000   00000000   00000000
006        00000000   00000000   00000000   00000000
                          DATA:  00000000   00000000
PROGRAM MODE
```

Data is transferred to step 001.



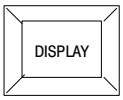

```
                                  BINARY DATA MONITOR
                                  SEQUENCER INPUT
COUNTER ADDR:  200                STEP:  001              SEQUENCER LENGTH:  006

                                  FILE:  400-413

                INPUT ADDR:  110                                  201
                DATA:        00000000  00000000  00000000  00000000

                MASK ADDR:   070                                  071
                DATA:        00000000  00000000  00000000  00000000

                STEP          WORD 1                WORD 2
                001        00000010  00000000  00000000  00000000
                002        00000000  00000000  00000000  00000000
                003        00000000  00000000  00000000  00000000
                004        00000000  00000000  00000000  00000000
                005        00000000  00000000  00000000  00000000
                006        00000000  00000000  00000000  00000000

                                  DATA:  00000000  00000000
PROGRAM MODE
```

The cursor is on the first digit of DATA.

0000001000000001

```
                                  BINARY DATA MONITOR
                                  SEQUENCER INPUT
COUNTER ADDR:  200                STEP:  001              SEQUENCER LENGTH:  006

                                  FILE:  400-413

                INPUT ADDR:  110                                  201
                DATA:        00000000  00000000  00000000  00000000

                MASK ADDR:   070                                  071
                DATA:        00000000  00000000  00000000  00000000

                STEP          WORD 1                WORD 2
                001        00000010  00000000  00000000  00000000
                002        00000000  00000000  00000000  00000000
                003        00000000  00000000  00000000  00000000
                004        00000000  00000000  00000000  00000000
                005        00000000  00000000  00000000  00000000
                006        00000000  00000000  00000000  00000000

                                  DATA:  00000010  00000001
PROGRAM MODE
```

KEY



INSERT

BINARY DATA MONITOR
SEQUENCER INPUT

COUNTER ADDR: 200 STEP: 001 SEQUENCER LENGTH: 006

FILE: 400–413

INPUT ADDR: 110 201
DATA: 00000000 00000000 00000000 00000000

MASK ADDR: 211 212
DATA: 00000000 00000000 00000000 00000000

STEP	WORD 1		WORD 2	
001	00000010	00000000	00000000	00000000
002	00000010	00000001	00000000	00000000
003	00000000	00000000	00000000	00000000
004	00000000	00000000	00000000	00000000
005	00000000	00000000	00000000	00000000
006	00000000	00000000	00000000	00000000

DATA: 00000010 00000001

PROGRAM MODE

BINARY DATA MONITOR
SEQUENCER INPUT

COUNTER ADDR: 200 STEP: 001 SEQUENCER LENGTH: 006

FILE: 400–413

INPUT ADDR: 110 201
DATA: 00000000 00000000 00000000 00000000

MASK ADDR: 211 212
DATA: 00000000 00000000 00000000 00000000

STEP	WORD 1		WORD 2	
001	00000010	00000000	00000000	00000000
002	00000010	00000001	00000000	00000000
003	00000000	00000000	00000000	00000000
004	00000000	00000000	00000000	00000000
005	00000000	00000000	00000000	00000000
006	00000000	00000000	00000000	00000000

DATA: 00000000 00000000

PROGRAM MODE

Data is transferred to step 002.

Continue adding your data:

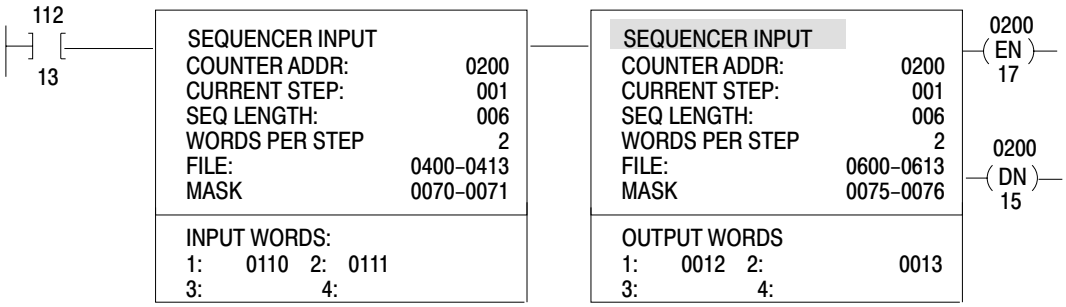
003: 00000010 00010000
004: 00000010 00010001
005: 00000010 01000001
006: 00000010 00000000

To add data to WORD 2 press [SHIFT] [⟶].

Data for word 2 step 004 is: 00100000 00000000

Press [CANCEL COMMAND] to display your rung.

Position your cursor on the words SEQUENCER OUTPUT.

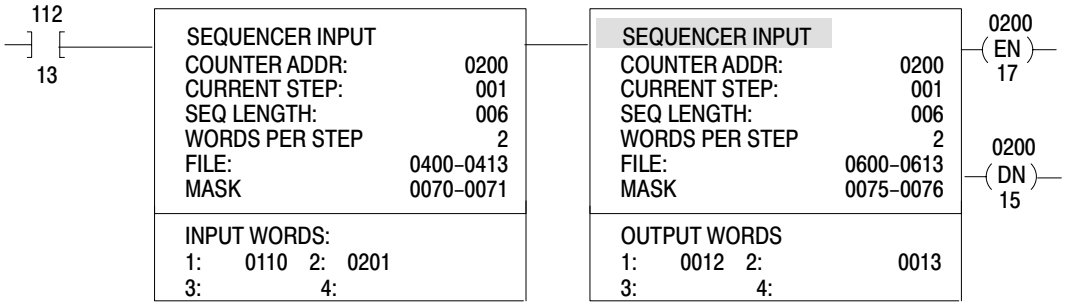


Load your data from Figure 10.2 into the binary monitor mode exactly like you did for the sequencer input instruction.

A completed display for the sequencer output instruction looks like:

```
                          BINARY DATA MONITOR
                            SEQUENCER INPUT
COUNTER ADDR:  200                  STEP:  001          SEQUENCER LENGTH:  006
                                    FILE:  600-613

INPUT ADDR:  012                                              013
DATA:          00000000    00000000    00000000    00000000

MASK ADDR:  075                                               076
DATA:          00000000    00000000    00000000    00000000

STEP          WORD 1                    WORD 2
001      00000010  00000000    00000000  00000000
002      00000010  00000001    00000000  00000000
003      00000010  00000001    00000000  00000000
004      00000010  00000001    00000000  00000000
005      00000010  00000001    00000000  00000000
006      00000010  00000000    00000000  00000000

                          DATA:  00000010  00000000
PROGRAM MODE
```

Press [CANCEL COMMAND] to see your ladder diagram rung.



You've completed programming our sample application.

Chapter 11

Special Programming Techniques

Objectives

This chapter describes special programming techniques. In this chapter you will read sections A through C concerning:

- Help directories
- On-Line data change
- On-line programming
- Data initialization key
- Block transfer
- One-shot
- Leading edge
- Trailing edge
- Manual restart
- Cascading timers

Section A
Special Programming Aids

Objectives

In this section you will read:

- Help directories
- On-line data change
- On-line programming
- On-line programming procedure
- Data initialization key

Help Directories

Help directories have been developed as an aid in using the industrial terminal. They list the several functions or instructions common to a single multipurpose key such as the [SEARCH] or [FILE] key. A master help directory is also available which lists the eight function and instruction directories for the Mini-PLC-2/15 processor and the key sequence to access them. The master help directory is displayed by pressing [HELP]. You can press the [HELP] key any time during a multi-key sequence. The remaining keys in the sequence can be pressed then without having to press [CANCEL COMMAND].

NOTE: If a particular function or instruction directory or an item in a directory is not available with the Mini-PLC-2/16 processor, the industrial terminal will display a “FUNCTION NOT AVAILABLE WITH THIS PROCESSOR” message if the key sequence is pressed.

On-Line Data Change

You can change (while the processor is in the run/program mode) the lower 12 bits of a word or word instruction. this excludes arithmetic and put instructions, or certain data of a block instruction. To do this, position the cursor on the appropriate instruction and press [SEARCH][5][1]. The message “ON-LINE DATA CHANGE, ENTER DIGITS FOR; (required Information)” will be displayed near the bottom of the screen. The new digits will be displayed in a command buffer as they are entered. Use the [→] and [←] cursor control keys as needed. After the new data is displayed, press [INSERT]to enter the data into memory.

To terminate this function, press [CANCEL COMMAND].

> **WARNING:** When the address of an instruction whose data is to be changed duplicates the address of other instructions in user program, the consequences of the change for each instruction should be thoroughly explored beforehand. This is to guard against unexpected machine operation which could result in damage to equipment and/or injury to personnel.

NOTE: When the memory write protect is activated by the EPROM back-up memory, on-line data change will not be allowed for addresses above 177. If attempted, the industrial terminal will display the error message: MEMORY PROTECT ENABLED.

On-line Programming

On-line programming allows you to make changes to the user program during machine operation when the processor is in the run/program mode and memory write protect is not active.

> **WARNING:** The task of on-line programming should be assigned only to an experienced programmer who understands the nature of Allen-Bradley programmable controllers and the machinery being controlled. Proposed on-line changes should be checked and rechecked for accuracy; and all possible sequences of machine operation resulting from the change should be assessed in advance. Be absolutely certain that the change must be done on-line and that the change will solve the problem without introducing additional problems. Notify personnel in the machine area before changing machine operation on-line.

To minimize the chances of error, maintain accurate data table assignments sheets and use the data initialization key described later in this section.

On-line Programming Procedure

The changes to your program that you can make in the on-line programming mode include the following:

- Insert an instruction
- Remove an instruction
- Insert a rung
- Remove a rung
- Change an instruction or instruction address

The on-line programming mode is accessible from the industrial terminal by pressing the key sequence [SEARCH][5][2]. The processor keyswitch must be in the RUN/PROG position. The heading, “ON-LINE PROGRAMMING” will appear in the top right-hand corner of the screen highlighted in reverse video.

The procedure for programming on-line in run/program mode is similar to the procedure for editing in program mode with the exception that the following three keys have a special purpose in on-line programming:

- [RECORD]
- [CANCEL COMMAND]
- [DATA INIT]

Use the [RECORD] key to enter a change to your program. Once pressed, the changed program is active.

Use the [CANCEL COMMAND] key to abort any on-line programming operation prior to pressing the [RECORD] key. It restores the ladder diagram display and program logic to its original state prior to the on-line programming operation. You can also use it to terminate the on-line programming mode.

The [DATA INIT] key will be discussed next.

Data Initialization Key

You must enter two types of information when programming the following instructions:

- Get
- Equal to
- Less than
- Timers
- Counters
- Files
- Sequencers

The two types of information needed are the **instruction** and **operating parameters.**

NOTE: Operating parameters are used only for file and sequencer instructions.

The data stored at the instruction address is divided into two sections: status bits (bits 14-17) and BCD values (bits 00-13). During program execution, these bits are constantly changing to reflect current states and values of program instructions. Therefore, when programming on-line, a decision must be made whether to use the current data or enter new data.

Use the [DATA INIT] key when adding an instruction containing new data. Do not use it when adding an instruction that will use the data at a pre-assigned address.

The [DATA INIT] key performs two functions in the on-line programming mode:

- It allows entry of BCD data values (stored at the instruction address).
- It assures that the status bits are cleared to 0000.

Use the [DATA INIT] key when programming a data instruction whose address is not currently being used in the program. If the [DATA INIT] key were not used, data at the new address (possibly remaining from previous programming) may interfere with proper machine operation when you insert the new instruction into the program.

> **WARNING:** When the address of a new instruction duplicates the address of other instructions in the program, the [DATA INIT] key should not be used without first assessing the consequences. Pressing the [DATA INIT] key will zero out the status bits stored at the existing instruction address. This may interfere with desired machine operation. Damage to equipment and/or injury to personnel could result.

To look for a specific instruction, press [SEARCH] (key sequence of the instruction). To look for a specific address, press [SEARCH] (key sequence of the address). This will help you to determine addresses currently used in your program.

To locate all addresses (excluding those associated with examine on and examine off instructions and those contained within files) press [SEARCH][8] (key sequence of the address). The address entered is the word address. For the output energize, latch and unlatch instructions, the industrial terminal will locate all of the bit addresses associated with the word address.

The message “SEARCH FOR” and the entered key sequences will be displayed at the bottom of the screen. The message “EXECUTING SEARCH” will appear temporarily. The industrial terminal will begin to search for the address and/or instruction from the cursor’s position. It will look past the temporary end and subroutine area boundaries to the END statement. Then it will continue searching from the beginning of the program to the point where the search began.

If found, the rung containing the first occurrence of the address and/or instruction will be displayed as well as the rungs after it. if the [SEARCH] key is pressed again, the next occurrence of the address and/or instruction will be displayed. When it cannot be located or all addresses and/or instructions have been found, a “NOT FOUND” message will be displayed.

If the instruction is found in the subroutine area or past the temporary end instruction, the area in which it is found will be displayed in the lower right hand corner of the screen.

Press [CANCEL COMMAND] at any time to terminate this function. All other keys are ignored during the search.

In summary, use [DATA INIT] to:

- Enter a data instruction with an unused address.
- Enter new data.
- Clear the status bits of an already used address. Press [DATA INIT] after the instruction key(s) and before you enter the address.

Section B
Block Transfer

Objectives

As a review from chapter 6, we will list the characteristics common to both block transfer instructions:

- Output instruction
- Block length depends on the type of I/O module.
- Request is made in the program scan.
- I/O scan is interrupted for the transfer.
- Entire file is transferred in 1 scan.
- Done bit remains on for 1 program scan after a valid transfer.
- Uses 2 words of the user program area.
- Key sequence:

  Block transfer read - BLOCK X-FER 1
  Block transfer write - BLOCK X-FER 0

- **Do not** use word 127 for block transfer data storage.

Programming Technique

You enter these two instructions exactly like the sequencer or file instructions described in chapter 10. The following is a synopsis of how to enter these instructions:

KEY	DISPLAY
BLOCK X–FER	No display..
1	BLOCK XFER READ DATA ADDR: 303 MODULE ADDR: 100 BLOCK LENGTH: 01 FILE: 0110–0110

NOTE: If you cannot remember the number identifier for these instructions then press [BLOCK X-FER][HELP]

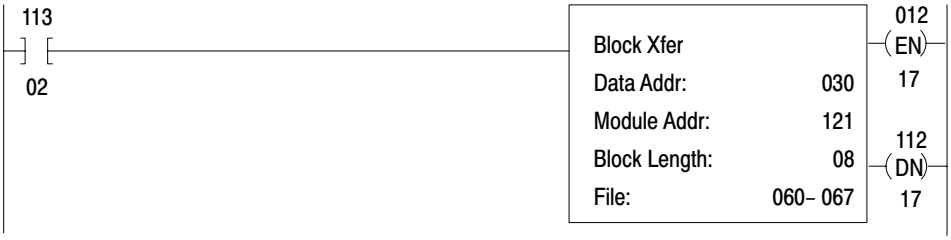
File in the instruction's parameters. These are our parameters:

DATA ADDR:	030
MODULE ADDR:	121
BLOCK LENGTH:	08
EILE:	0060-0067

Here is an explanation of the operation cycle after we entered our values:

- I/O module occupies rack number 1, module group 2, slot 1.
- Enable it is automatically set at the address 01217.
- Done bit is automatically set at the address 11217.
- Figure 11.1 shows a sample rung. Do not program this rung as practice when using on-line production equipment.

Figure 11.1
Sample Rung



Here is a explanation to help you understand the sample rung:

1. During the program scan when input switch 11302 is true, the read enable bit 01217 is set to 1.

2. In the next scan of the output image table, the data in 01217 is sent to the module.

3. The I/O module responds (ready for transfer).

4. The processor interrupts the output image table scan and begins to search the timer/counter's accumulated area of the data table.

5. The processor locates the address 121 in word address 030 and locates the file address 060, 100 above 030 in word address 130.

6. The processor transfers the data from the I/o module to processor's data table word address 060-067.

7. The done bit (11217) is set. This completes the transfer.

8. The processor completes the I/O scan.

NOTE: This same discussion applies when programming multiple write instructions of different block lengths to one module.

Under certain conditions, it may be desirable to transfer part of a file rather than the entire file. For example, a processor could be programmed to read the first two or three channels of an analog input module periodically but read all channels less frequently. To do this, two or more block transfer read instructions would be used: one for each desired block length starting at the same first word. The read instructions would have the same module address, data address and file address. The size of the file would equal the larger block length.

When two or more block transfer instructions have a common module address, careful programming is required to compensate for the following possible situations:

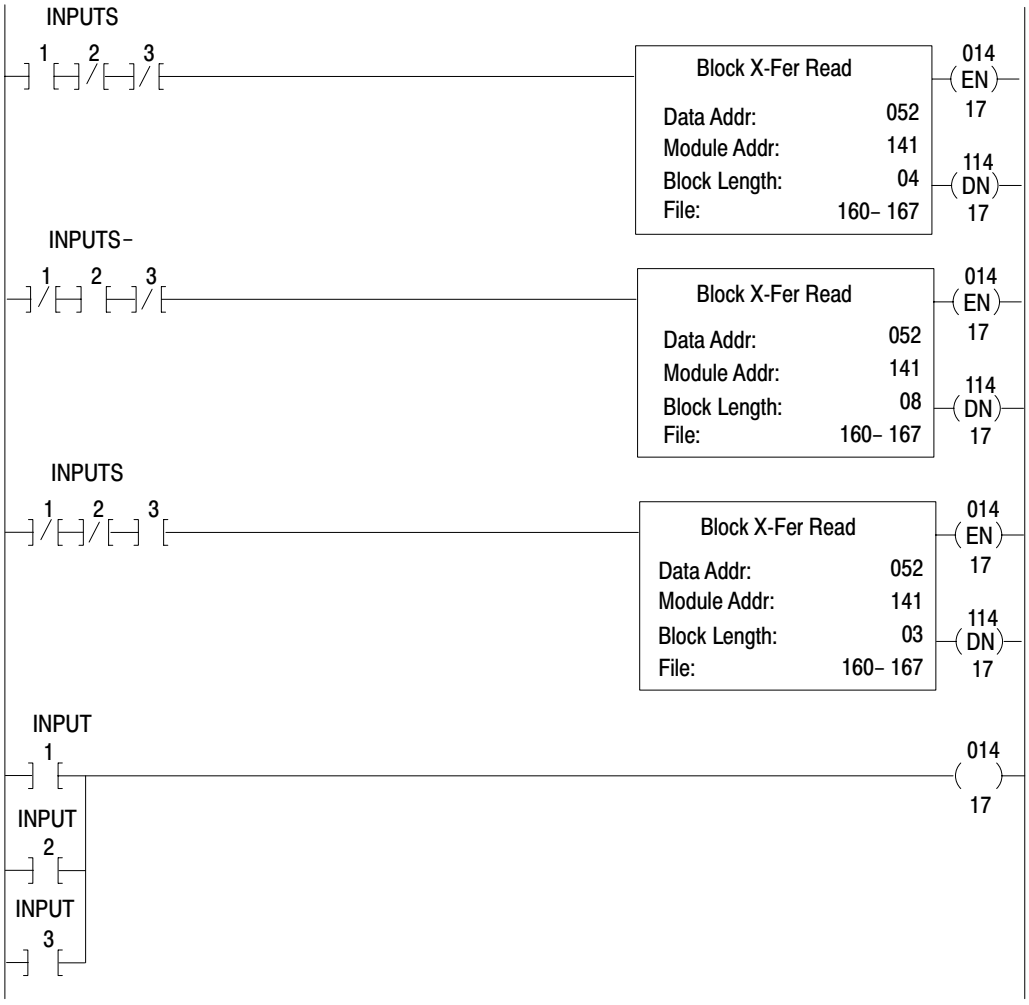
During any program scan, data in the output image table byte can be changed alternately by each successive block transfer instruction having a common module address. The enable bit can be turned on or off alternately according to the true or false condition of the rungs containing these instructions. The on or off status of the last rung will govern whether the transfer will occur.

Secondly, the block length can be changed alternately in accordance with the block lengths of the enable instructions. The block length of the last enabled block transfer instruction having a common module address will govern the number of words transferred.

Refer to the user's manual for additional information on the block transfer module of interest or consult our Publication Index (publication SD499).

The programming example shown in Figure 11.2 shows how multiple reads of different block lengths from one module can be programmed. when any one of the input switches is closed, the rung is enabled and the block length is established. The last rung enables the block transfer instruction regardless of the previous changes in status of the enable bit. The examine off instructions prevent more than one of the block transfer instructions from being energized in the same scan.

Figure 11.2
Programming Multiple Reads from One Module



WARNING: When programming multiple writes (or reads) to the same module, programming errors could prevent the desired transfer from taking place or limit the number of words transferred. Invalid data could be sent to an analog output device (or could be operated upon in subsequent scans) resulting in unpredictable machine operation. Damage to equipment and/or injury to personnel could occur.

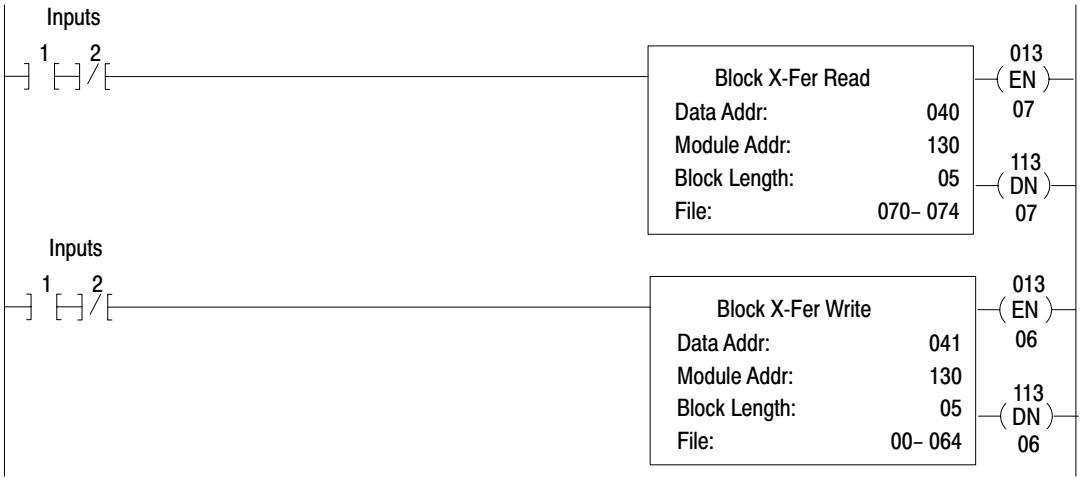
Bidirectional Block Transfer

Definitions: Bidirectional block transfer is the sequential performance of block transfer read and block transfer write operations. The order of operation is generally determined by the I/O module.

Syntax: Two rungs of our program are required, one containing the block transfer read instruction, the other containing the block transfer write instruction. When both instructions are given the same module address, the pair are considered as bidirectional block transfer instructions.

Operations: Figure 11.3 shows a sample program for using a bidirectional block transfer technique. Do not enter this sample program using on-line production equipment.

Figure 11.3
Bidirectional Block Transfer



The operation's cycle for the above program is similar to the operation's cycle using one block transfer instruction. However, understanding your entered values is essential. Therefore, we will not discuss each part of the instruction.

Data Address and Module Address - The module address is stored in BCD in the data address of the read and write instructions. In this example, the module address is 130: rack 1, module group 3, slot 0.

Two data addresses must be used. In this example they are 040 and 041. Both contain the module address. For bidirectional operation, each data address word also contains an enable bit; bit 16 for a read operation (in 040). When the processor searches the data addresses in the timer/counter accumulated area of the data table, it finds two consecutive data addresses both containing the same module address. The read bit is set high in one data address (in 040). the write bit is set high in the other (in 041). When the processor finds a match of the module address and the enable bit (read bit or write bit) for the desired direction of transfer, it then locates the file address to which (or from which) the data will be transferred.

File Address - Generally two file addresses are required: one to receive data transferred from the module, the other containing data to be transferred to the module. In this example, they are 060 and 070. the consecutive storage locations containing the file addresses in BCD are found in the preset area of the data table at addresses 140 and 141. They are found 1008 above the corresponding consecutive data addresses in the accumulated area of the data table.

Block Length - The block lengths of the read and write instructions can be set equal or unequal to each other up to any value not exceeding the default (maximum) block length of the module. If the default value is used, it instructs the module to control the number of words transferred. Although the default value varies from one kind of module to another, it can be entered into the instruction block as the number 00 for all block transfer modules.

Equal Block Lengths - When the block lengths are set equal or when the default block length is specified by the programmer, the following considerations are applicable. (Our example shows equal block lengths):

- Both the read and write instructions could and should be enabled in the same scan (separate but equal input conditions).
- The module decides which operation will be performed first when both instructions are enabled in the same scan.
- The alternate operation will be performed in a subsequent scan.
- Transferred data should not be operated upon until the done bit is set.

Unequal Block Lengths - Consult the user's manual for the block transfer module of interest for programming guidelines when setting the block lengths to unequal values. Our Publication Index (publication SD499) lists the specific manuals.

> **WARNING:** When the block lengths of the bidirectional block transfer instructions are set to unequal values, the rung containing the alternate instruction must not be enabled until the done bit of the first transfer is set. If they are enabled in the same scan, the number of words transferred may not be the number intended, invalid data could be operated upon in subsequent scans or analog output devices could be controlled by invalid data. Unexpected machine operation could occur with possible damage to equipment and/or injury to personnel.

Section C
Special Program Techniques

Objectives

In this section you will read:

- How to program a one-shot
- How to program a leading edge one-shot
- How to use a manual restart
- How to program cascading timers

One-Shot

The one-shot programming technique is used for certain applications to set a bit on for one program scan only. There are two types of one-shots that you can program:

- Leading edge
- Trailing edge

Leading Edge One-Shot

The leading edge one-shot programming technique sets a bit for one scan when its input condition has made a false-to-true transition. The false-to-true transition represents the leading edge of the input pulse. The programming for a leading edge one-shot is shown in Figure 11.4.

Figure 11.4
Leading Edge One-Shot

When bit 11204 makes a transition, the one-shot bit (bit 25300) is set on for one scan. The length of time bit 11204 remains on does not affect the one-shot bit due to the next two rungs. Bit 01114 will be latched on when bit 11204 is on or bit 01114 will be unlatched when 11204 is off. During the next scan, either set of conditions will prevent bit 25300 from being set on. The one-shot bit is set on for another scan only when bit 11204 makes a true-to-false and then a false-to-true transition.

Trailing Edge One-Shot

A trailing edge one-shot programming technique sets a bit for one scan when its input condition has made a true-to-false transition. The true-to-false transition represents the trailing edge of the input pulse. Programming for a trailing edge one-shot is shown in Figure 11.5.

Figure 11.5
Trailing Edge One-Shot

When bit 11204 goes true, bit 01114 is latched on. As soon as bit 11204 makes a true-to-false transition, the one-shot bit (bit 25300) is set on and bit 01114 is unlatched. Bit 15300 will remain on for only one scan. The input bit 11204 must make a false-to-true transition then a true-to-false transition to set the one-shot bit for another scan.

Manual Restart

You can control start-up manually when using the EPROM back-up memory. Anytime that an EPROM to RAM memory transfer occurs, bit 02701 in the data table will be reset by the processor. This allows a machine's start switch to be programmed in either of two ways as shown in Figure 11.6 and Figure 11.7.

The technique shown in Figure 11.6 can be used when your program does not contain any MCR instruction. The technique shown in Figure 11.7 is an alternative if an unused octal identifier (chapter 6 section B) is available.

Figure 11.6
Manual Restart Using a JMP Instruction

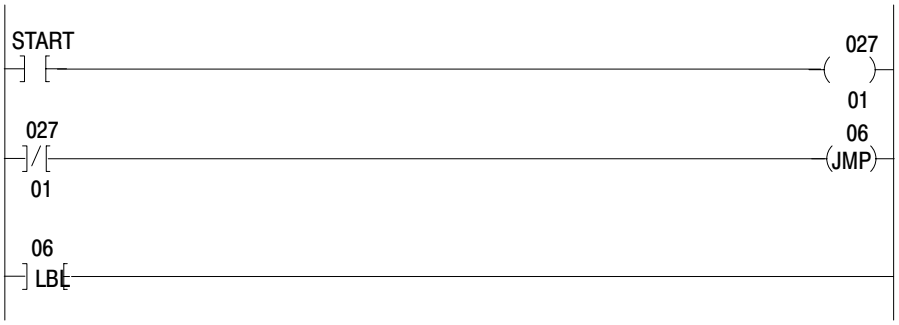
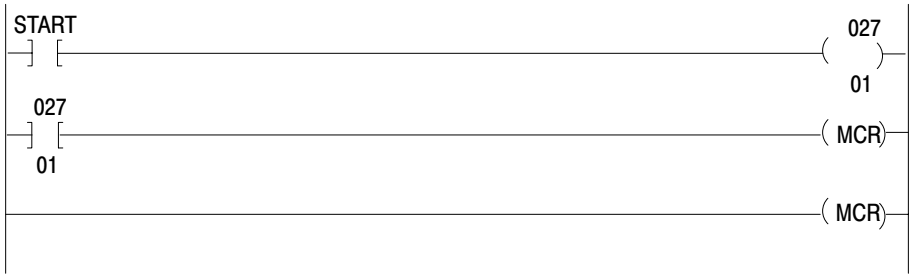


Figure 11.7
Manual Restart Using an MCR Instruction



The values in the data table at start-up will depend on whether or not the memory was retained by back-up battery. If a battery was used, the data table will contain the values which existed when power was removed. If no battery was used, the values programmed into EPROM will be transferred into the data table at power-up.

Start-up conditions can be summarized for manual start-up (using one of the suggested programming techniques) and for automatic start-up (where neither technique is used) as follows:

a. From initial start-up conditions: remove the battery from the system

b. From in-process conditions at time of power loss: maintain battery back-up.

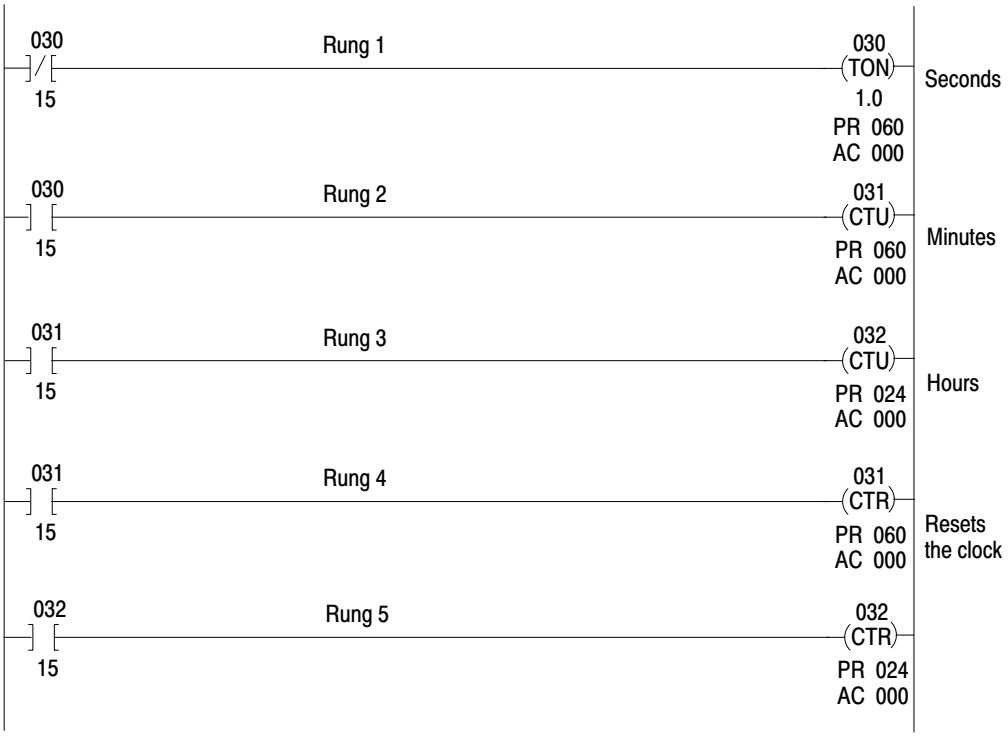
CAUTION: The battery should not be replaced during AC power loss. Your application program would be lost. Replace the memory back-up battery only with power applied to the system.

An EPROM to RAM transfer will not take place upon power-up when the battery is operating and the memory content matches EPROM content. A transfer will take place if any alteration of memory content occurred while the battery was being changed. If a transfer occurred (memory was altered), the data table will contain the values programmed into the EPROM. If the transfer did not occur (memory was not altered), the data table will contain the values which existed at the time the system power was removed. There is not way of determining whether the transfer will occur if the battery is replaced during a power loss.

Cascading

Cascading is defined as a program technique that extends the ranges of timers and/or counter instructions beyond the maximum values that may be accumulated. Figure 11.8 illustrates a 24-hour clock program. Again we emphasize not to enter these instructions using your on-line production equipment. This clock program is not accurate. Do not use it as a real time clock device.

Figure 11.8
24 Hour Clock



A synopsis of the operation's cycle is:

Rung 1: When the conditions are true the timer will start.

Rung 2: When AC=PR (accumulated value equals preset value) of the timer, counter 031 increments.

Rung 3: When AC=PR of counter 031, counter 032 increments.

Rung 4: When AC=PR of counter 031, the -(CTR)031- resets counter 031's accumulated value.

Rung 5: When AC=PR of counter 032, the -(CTR)032- resets counter 032's accumulated value.

Chapter 12

Run Time Errors

Objectives

In this chapter you will read:

- What are run time errors?
- Diagnosing a run time error.
- Causes of run time errors.

What are Run Time Errors?

Run time errors are errors that occur while the processor executes your program, and are only apparent during this time. These errors result from improper programming techniques.

For example, it is possible to program a series of instructions in which the processor cannot properly perform the operation. Or it is possible to program paired instructions, such as a jump/label, with improper syntax.

In the run or test modes, if a run-time error occurs, your processor will halt program operations and the processor and memory indicators will illuminate a red color. In the run/program mode you get no indication because the processor is automatically switched to the program mode.

Diagnosing a Run Time Error

The following are steps on how to diagnose run time errors:

1. Connect your industrial terminal to the processor.
2. Turn on the industrial terminal and notice the message, RUN-TIME ERROR. If the industrial terminal is already connected, then your ladder diagram will be replaced by the display showing the run-time error message.
3. Turn the keyswitch to the PROG position.
4. Press [1][1] to display the instruction that caused the error.
5. Correct the run time error by editing your program. (Refer to Table 12.A.)
6. Restart your processor.

Table 12.A
Possible Causes of Run Time Errors

Instruction	Cause
Jump	Jumping from the main program into the subroutine area or vice versa.
	Jumping backwards.
	Omitting the label instruction corresponding to the jump instruction.
	Jumping over a temporary end instruction.
Label	Multiple placement of the same label identification number.
	Removing a label instruction but leaving its reference, the jump or jump to subroutine instructions.
Jump to subroutine	To begin your main program.
	To jump forward in your main program.
	Use in the subroutine area.
	Omitting a return instruction.
	Omitting a corresponding label instruction.
	Jumping over a temporary end instruction.
Return	Processor does not find a return instruction from the subroutine area.
	Using a return instruction outside the subroutine area.
Files	AC>PR
	Duplicating counter's address.
	Manipulating the counter's accumulated value by means of external programming equipment or data highway hardware.
Sequencer	File address is out of range.
	Preset value equals 0.
	AC>PR
Block transfer	Giving the module address a non-existent I/O rack number.
	Incorrect block length value.

Chapter 13

Troubleshooting Aids

Objectives

Troubleshooting aids are useful during start-up of your operations and when troubleshooting your Mini-PLC-2/15 programmable controller system. They are:

- Bit manipulation function
- Bit monitor function
- force on and force off functions
- Temporary end instruction
- ERR message display

We will discuss each troubleshooting aid.

Bit Manipulation Function

Bit manipulation allows the status of the displayed bits to be selectively changed or forced. It is useful in setting initial conditions in the data of word instructions.

Bit manipulation can function when the processor is in program mode. When in test, or run/program, the user program may override the bit status in the next scan.

The [→] and [←] keys can be used to cursor over to any bit. With the cursor on the desired bit, you can change its status by pressing the [1] or [0] key. Bit manipulation also allows the forcing of image table bits.

To terminate this function, press [CANCEL COMMAND].

> **WARNING:** If it is necessary to change the status of any data table bit, be sure that the consequences of the change are thoroughly understood. If not, unpredictable machine operation could occur directly or indirectly as a result of changing the bit status. Damage to equipment and/or injury to personnel could result.

Bit Monitor Function

Bit monitor lets you display the status of all 16 bits of any data table word. It can function when the processor is in any mode. By pressing [SEARCH][5][3] (key sequence of word address), the status of all 16 bits of the desired word will be displayed. While the cursor is in the word address filed, use the [→] and [←] keys to change address digits.

The status of the 16 bits in the next highest or next lowest word address also can be displayed by pressing the [↑] or [↓] keys, respectively. Bit monitor also can display the status of force conditions.

Force Functions

There are two types of force functions:

- Force on
- Force off

You can use the force functions to selectively force an input bit or output bit on or off. The processor must be in the test or run/program mode.

The force functions determine the on/off status of input bits and output bits by overriding the I/O scan. You can use an input bit on or off regardless of the actual state of the corresponding input device. However, forcing an output bit will cause the corresponding output device to be on or off regardless of the rung logic or the status of the output image table bit.

NOTE: When in test mode, the processor will hold outputs in their last state regardless of attempts to force them on, even through the output bit instructions will be intensified.

Forcing functions can be applied in either of two ways using:

- Bit manipulation/monitor display of an I/O word
- Ladder diagram display of user program

By pressing [SEARCH][5][3] (key sequence of address), you can display the bit status and force status of the 16 corresponding input bits or output terminals of the desired word. use the [→] and [←] keys to cursor to the desired bit. Or, in the ladder diagram display, forcing can be applied by placing the cursor on an examine or energize instruction. In either case, after positioning the cursor, you can use any one of the following key sequences for placing or removing a forced condition:

[FORCE ON] [INSERT]

[FORCE OFF] [INSERT]

[FORCE ON] [REMOVE]

[FORCE OFF] [REMOVE]

You can remove all force on or all force off functions at once in ladder diagram display by pressing either of the following key sequences:

[FORCE ON] [CLEAR MEMORY]

[FORCE OFF] [CLEAR MEMORY]

The on or off status of a forced bit will appear beneath the bit instruction in the rung.

In all processor modes, a “FORCED I/O” message is displayed near the bottom of the screen when bits are forced on or off. In every mode except the program mode, the forced status “ON” or “OFF” is displayed below each forced instruction.

NOTE: The on or off status of latch/unlatch instructions is also displayed below the instruction. However, this is displayed only in program mode.

All force functions are cleared when you:

- Disconnect the power cable or communications cable of the industrial terminal or processor.
- Lose AC power throughout your plant.
- Press [MODE SELECT].

> **WARNING:** When an energized output is being forced off, keep personnel away from the machine area. Accidental removal of force functions will instantly turn on the output device. Injury to personnel near the machine could result.

The industrial terminal displays a complete list of bit addresses that are forced on or off by pressing:

[SEARCH] [FORCE ON]

[SEARCH] [FORCE OFF]

If all the bits forced on or off cannot be displayed at one time, use the [SHIFT] [↑] and [SHIFT] [↓] keys to display additional forced bits.

Press [CANCEL COMMAND] to terminate this display.

Temporary End Instruction

You can use the temporary end instruction to test or debug a program up to the point where it is inserted. It acts as a program boundary because instructions below it in user program are not scanned or operated upon. Instead, the processor immediately scans the I/O image table followed by user program from the first instruction to the temporary end instruction.

When the temporary end instruction is inserted, the rungs below it, although visible and accessible, are not scanned. Their content can be edited, if desired. The displayed section of user program made inactive by the temporary end instruction will contain the message “INACTIVE AREA” in the lower right-hand corner of the screen.

You can insert the temporary end instruction in either of two ways:

- Cursor to the last rung of the main program to be kept active. Position the cursor on the output instruction. Press [INSERT][T.END]
- Cursor to the first rung of the main program to be made inactive. Position the cursor on the first instruction in the rung. Press [INSERT][<-][T.END].

To remove this instruction, position the cursor on it and press [REMOVE][T.END].

To enter a rung after the temporary end instruction, place the cursor on the temporary end instruction. Press [INSERT] [RUNG] and then enter the new rung.

Attempting to use the temporary end instruction in any of the following ways will either be prevented by the industrial terminal or result in a run-time error.

- Using more than one temporary end instruction at a time.
- Using the instruction in the subroutine area.
- Inserting or removing the instruction during on-line programming.
- Placing the instruction in the path of jump or jump to subroutine instructions.

ERR Message for an Illegal Opcode

An illegal opcode is an instruction code that the processor does not recognize. It causes the processor to fault and is displayed as an ERR message in the ladder diagram rung in which it occurs. The 4-digit hex value of the illegal opcode is displayed above the ERR message by the 1770-T3 industrial terminal.

The illegal opcode ERR message should not be confused with ERR messages caused when a 1770-T1 or -T2 industrial terminal is connected to a processor that was programmed using a 1770-T3 industrial terminal. Those ERR messages do not contain the 4-digit hex value and do not cause a processor fault.

If an illegal opcode should occur you can compare the rung containing the illegal opcode with the equivalent rung in a hard copy printout of the program. you must either replace the error with its correct instruction, replace the instruction, or remove it. The ERR message due to an illegal opcode cannot be removed directly. Instead, remove and replace the entire rung. You should identify and correct the cause of the problem in addition to correcting the ERR message.

Appendix A

Quick Reference Section

This section reminds you of what you have read in this manual. Tables illustrate:

Table A.A
General Program Information

Instruction Name	Key Symbol	Words of Memory Required
Examine On	–| |–	1
Examine Off	–|/|–	1
Output Energize	–( )–	1
Output Latch	–(L)–	1
Output Unlatch	–(U)–	1
Branch Start		1
Branch End		1
Timer On–Delay [1]	–(TON)–	3
Timer Off–Delay [1]	–(TOF)–	3
Retentive Timer On–Delay [1]	–(RTO)–	3
Retentive Timer Reset [1]	–(RTR)–	3
Up Counter [1]	–(CTU)–	3
Down Counter [1]	–(CTD)–	3
Counter Reset [1]	–CTR)–	3
Get	–|G|–	1
Put	–(PUT)–	1
Less than	–|<|–	1
Equal to	–|–=|–	1
Get Byte	–|B|–	1
Limit Test	–|L|–	1
Add	–(+)–	1
Subtract	–(–)–	1
Multiply	–(X)–(X)–	2
Divide	–(÷)–(÷)–	2
Master Control Reset	–(MCR)–	1
Zone Control Last State	–(ZCL)–	1
Immediate Input Update	–|I|–	1
Immediate Output Update	–(IOT)–	1
Subroutine Area Instruction	SBR	1
Label	–(LBL)–	1
Jump	–(JMP)–	1
Jumpt to Subroutine	–(JSR)–	1
Return	–(RET)–	1
File–to–File Move [2]	FILE 10	5
File–to–Word Move[2]	FILE 12	4
Word–to–File Move[2]	FILE 11	4
Sequencer Input[2]	SEQ 1	5–8
Sequencer Output[2]	SEQ 0	5–8
Sequencer Load[2]	SEQ 2	5–8
Block Transfer Read	BLOCK XFER 1	2
Block Transfer Write	BLOCK XFER 0	2

Instruction Name	Key Symbol	Words of Memory Required
Temporary End	T. END	1

[1] Timer and counter instructions use two words from the data table and one word from user program.
[2] File and sequence instructions use a varied amount from the data table. It depends on the size of your file. Values can range from 000–999 words

Table A.B
Instruction Execution Values

Note: These values are approximate (in average microseconds) exectuion values per scan.

Instruction Name	Symbol	Instruction True	Instruction False
Examine on, Examine off	–||–,–|/|–	10	5
Output Energize	–()–	19	19
Output Latch	–(L)–	19	15
Output Unlatch	–(U)–	19	15
Get	–(G)–	27	–
Put	–(PUT)–	22	15
Equal	–(=)=	22	5
Less Than	–(<)–	31	5
Get Byte	–|B|–	11	–
Limit Test	–|L|–	23	5
Counter Reset	–(CTR)–	23	15
Retentive Timer Reset	–(RTR)–	24	16
Timer On–delay	–(TON)–	140	60
Retentive Timer On–delay	–(RTO)–	140	48
Retentive Timer On–delay	–(TOF)	145	70
Up Counter	–(CTU)–	130	110
Down Counter	–(CTD)–	135	115
Add	–(+)–	48	15
Subtract	–(–)–	80	19
Multiply	–(x)–(x)–	615	60
Divide	–(÷)–(÷)–	875	
Add to any of the above when its address is 400_8 or greater		27	27
Master Control Reset	–(MCR)–	23	20
Zone Control Last State	–(ZCL)–	28	81+
Branch Start		18	13
Branch End		18	13
End, Temporary End	T.END	27	27

Instruction Name	Symbol	Instruction True	Instruction False
Subroutine ARea	SBR	27	27
Immediate Input Update	–[I]–	140	–
Immediate Output Update	–(IOT)–	170	33
Label	LBL	19	–
Return	–(RET)–	28	15
Jump To Subroutine	–(JSR)–	160	50
Jump	–(JMP)–	170	50
Block Transfer Read	BLOCK X–FER 1	150	135
Block Transfer Write	BLOCK X–FER 0	150	135
Sequencer Load	SEQ 2	650	200
Sequencer Input	SEQ 1	790	200
Sequencer Output	SEQ 0	730	200
File–to–word Move	FILE 12	470	200
Word–to–file Move	FILE 11	910	280
File–to–file Move	FILE 10	470	200

Table A.C
Relay Type Instructions

NOTE: You can assign input and output address, XXXXX, to any location in the data table, excluding the processor work areas. The word address is displayed above the instruction and the bit number below it. To enter a bit address larger than 5 digits, press [EXPAND ADDR] after the instruction key and then enter the bit address. Use a leading zero if necessary.

Key Symbol	Instruction Name	1770–T3 Display	Rung Conditions
–||–	Examine On	XXX –||– XX	When the addressed memory bit is on, the instruction is true.
–|/|–	Examine Off	XXX –|/|– XX	When the addressed memory bit is off, the instruction is true.
–( )–	Energize	XXX –( )– XX	When the rung is true, the addressed memory bit is set.[1]] If the bit controls an output device that output device will be on.
–(L)–	Latch	XXX –(L)– ON XX or OFF	When the rung is true, the addressed memory bit is latched on and remains on until it is unlatched. [1]] The output latch instruction is initially off when entered, as indicated below the instruction. it can be preset on by pressing a [1] after entering the bit address. An on will then be indicated below the instruction in program mode. An unlatch instruction will always override a latch instruction, even if the latch rung is true.
–(U)–	Unlatch	xxx –(U)– ON XX or OFF	When the rung is true, the addressed bit is unlatched. [1]] If the bit controls an output device, that device is de–energized. On or off will appear below the instruction indicating the status of the bit in Program mode only.
	Branch Start		This instruction begins a parallel logic path and is entered at the beginning of each parallel path.
	Branch End		This instruction ends two or more parallel logic paths and is used with branch start instructions.

[1]] These intructions should not be assigned input image table addresses because input image table words are reset each I/O scan.

Table A.D
Timer Instructions

NOTE: The timer word address, XXX, is assigned to the timer accumulated areas of the data table. To determine which addresses are valid accumulated areas, the most significant digit in the word address must be an even number.

The time base, TB, is user–selectable and can be 1.0, 0.1, or 0.01 second. Preset values, YYY, and accumulated values, ZZZ, can vary from 000 to 999.

Bit 15 is the timed bit. Bit 17 is the enable bit.

The word address displayed will be 3 or 4 digits long depending on the data table size. When entering the word address, ust a leading zero if necessary.

Key Symbol	Instruction Name	1770–T3 Display	Rung Condition	Status Bit
–(TON)–	Timer On Delay	XXX –(TON)– TB PR YYY AC ZZZ	When the rung is true, the timer begins to increment the accumulated value at a rate specified by the time base. When the rung is false, the timer resets the accumulated value to 000.	When the rung is true: Bit 15 – set when AC = PR Bit 17 – set When the rung is false: Bits 15 and 17 – reset
–(TOF)–	Timer Off Delay	XXX –(TOF)– TB PR YYY AC ZZZ	When the rung is true, the timer resets the accumulated value to 000. When the rung is false, the timer begins to increment the accumulated value.	When the rung is true: Bit 15 – set Bit 17 – set Whenthe rung is false: Bit 15 – resets when AC = PR bit 17 reset
–(RTO)–	Retentive Timer	XXX –(RTO)– TB PR YYY AC ZZZ	When the rung is true, the timer begins to incremetn the acumulated value. When false, the accumualted value is retained.	When the rung is true: Bit 15 – set when AC = PR bit 17 – set When the rung is false: Bit 15 – no ation is taken. Bit 17 – reset
–(RTR)–	Retentive TImer Reset	XXX –(RTR)– PR YYY AC ZZZ	XXX – Word address of the retentive timer it is resetting. The preset and accumulated values are automatically entered by the industrial terminal. WHen the rung is true, the accumulated value and status bits are reset.	When the rung is true: Bit 15 and 17 – reset When the rung is false: No action is taken.

Table A.E
Conter Instructions

NOTE: The counter word address, XXX, is assigned to the counter accumulated areas of the data table To determine which addresses are valid accumulated areas, the most significant digit in the word address must be an even number.

Bit 14 is the overflow/underflow bit.
BIt 15 is the count complete bit.
BIt 16 is the enable bit the for CTD instruction.
Bit 17 is the enable bit for the CTU instruction.

The word address displayed will be 3 or 4 digits long depending on the data table size. WHen entering the word address, use a leading zero if necessary.

Key Symbol	Instruction Name	1770–T3 Display	Rung Condition	Status Bit
–(CTU)–	Up Counter	XXX –(CTU)– PR YYY AC ZZZ	Each time the rung goes true, the accumulated value is incremented one count. The counter will continue counting after the preset value is reached. The accumulated value can be reset by the CTR instruction	When the rung is true: BIt 14 –= set if AC>999 Bit 15 – set when AC≥PR Bit 17 – set When the rung is false: Bit 14 and 15 – retained if it ws set BIt 17 – reset
–(CTR)–	Counter Reset	XXX –(CTR)– PR YYY AC ZZZ	XXX – Word address of the CTU is resetting The preset and accumulated values are automatically entered by the industrial terminal. When the rung is ture, the CTU, or CTD accumulated value and status bits are reset to 000.	When the rung is true: Bit 14, 15, 16, 17 – reset When the rung is false: No action is taken.
–(CTD)–	Down Counter	XXX –(CTD)– PR YYY AC ZZZ	Each time the rung goes true, the accumulated value is decreased by one count.	When the rung is true: Bit 14 – set when AC <000 Bit 15 – set when AC<PR Bit 16 – set When the rung is false: Bit 154 and 15 – retained if it was set Bit 16 – reset

Table A.F
Data Manipulation Instructions

NOTE: Date manipulation instructions operate upon BCD valuessss and/or 16 bit data table. THe word address XXX, is displayed above the instruciton; the BCD vlue or data operated uon YYY, is displayed beneath it. The data is stored in the lower 12 bits of the wword address and can be any value from 000 to 999 BCD, excet as noted.

Word address displayed will be either 3 or 4 digits depending upon the data table size. When entering the word address, use a leading zero if necessary.

Key Symbol	Instruction Name	1770-T3 Display	Rung Conditions
-[G]-	Get	XXX -[G]- YYY	The get instruction is used with other data manipulation or arithmetic instructions. When the rung is true, all 16 bits of the get instruction are duplicated and the operation of the instruction following it is performed.
-(PUT)-	Put	XXX -(PUT)- YYY	The put instruction should be preceded by the get instruction. When the rung is true, all 16 bits at the get instruction address are transferred to the put instruction address.
-[<]-	Less Than	XXX -[<]- YYY	The less than instruction should be preceded by a get instruction. 3-digit BCD values at the get and less than word addresses are comapred. If the logic is true, the rung is enabled.
-[=]-	Equal To	XXX -[=]- YYY	The equal to instruction should be preceded by a get instruction.
-[B]-	Get Byte	XXXD -[B]- YYY	D - Designates the upper or lower byte of the word. 1 = uper byte, 0 = lower byte. YYY - Octal value from 000 to 377 stored in the upper or lower byte of the word address. The get byte instruction should be followed by a limit test instruction. A duplicate of the designated byte is made and comared with the upper and lower limits of the limit test instruction.
-[L]	Limit Test	XXX AAA -[L]- - BBB	AAA - Upper limit of limit test, an octal value from 000 to 377. BBB - Lower limit of limit test, an octal value from 000 to 377. The limit test instruction should be preceded by a get byte instruction. Compares the value at the get byte instruction with the values at the limit test instruction. If found to be between or equal to the limits, the rung is enabled.

Table A.G
Arithmetic Instructions

NOTE: Arithmetic instructions operate on BCD values in the data table. THe word address XXX is displayed above the instruction; the BCD value YYY which is the result of the arithmetic operation, is displayed beneath it. The BCD value is stored in the lower 12 bits of the word address and can be any value from 000 to 999.

Displayed word addresses will be 3 0r 4 digits depending on the data table size. When entering the word address, use a leading zero if necessary.

Key Symbol	Instruction Name	1770–T3 Display	Rung Conditions
–(+)–	Add	XXX –(+)– YYY	The add instruc tion is an output instruction. It is always preceded by two get instructions which store the BCD values to be added. When the sum exceeds 999, bit 14 is set. A1 is displayed in front of the result YYY.
–(–)–	Subtract	XXX –(–)– YYY	The subtract instruction is an output instruction. It is always preceded by two get instructions. The value in the second get address is subtracted from the value in the first. When the difference is negative, bit 16 is set and a minus sign is displayed in front of the result YYY.
–(x)–	Multiply	XXX –(x)– YYY	The multiply instruction is an output instruciton. It is always preceded by two get instructions which store the values to be multiplied. Two word addresses are required to store the 6 digit product.
–(÷)–	Divide	XXX –(÷)– YYY	The divide instruction is an output instruciton. It is always preceded ny two get instructions. The vlue of the first is divided by the value of the second. Two word addresses are required to store the 6 digit quotient. Its decimal point is placed automatically by the industrial terminal.

Table A.H
Program Control Instructions

NOTE: The MCR and ZCL boundary instructions have no word address.

The word addresses, XXX, of the immediate input and output instructions are limited to the input and output image tables respectively.

Displayed word addresses will be 23 or 4 digits long, depending on data table size. When entering the word address, use a leading zero if necessary.

Key Symbol	Instruction Name	1770–T3 Display	Explanation and Rung Conditions
–(MCR)–	Master Control Reset	–(MCR)–	Two MCR instructions are required to control a group of outputs. The first MCR instruction is programmed with input conditions to begin the zone. THe second MCR instruction is programmed unconditionally to end the zone. When the MCR rung is true, each rung condition controls their output instruction. When the first MCR rung is false, all non–retentive bits in the zone are reset. **WARNING:** Do not overlap MCR zones, or nest with ZCL zones. Do not jump to a label in MCR zones.
–(ZCL)–	Zone Control Last State	–(ZCL)–	Two ZCL instructions are required to control a group of outputs. The first ZCL instruction is programmed with input conditions to begin the zone. The second ZCL instruction is programmed unconditionally to end the zone. When the ZCL rung is ture, all output instructions within the zone act according to the logic conditions preceding them. When the first ZCL rung is false, outputs in the zone will remain in their last state. **WARNING:** Do not overlap ZCL zones, or nest with MCR zones. Do not jump to a label in ZCL zones.
–[I]–	Immediate Input	XXX [I]	Processor interrupts proram scan to update input image table with data from the corresponding module group. It is updated before the n ormal I/O scan and executed each program scan.
–(IOT)–	Immediate Output	XXX –(IOT)–	When rung is true, the processor interupts program scan to update a module group with data from its corresponding output image table word address. It is updated before the normal IO scan and executed each program scan when the rung is true. Can be programmed unconditionally.

Table A.I
Jump/Subroutine Instructions

Key Symbol	Instruction Name	1770–T3 Display	Explanation and Rung Conditions
SBR T. END	Subroutine Area	SUBROUTINE AREA	Establishes the boundary between main program and subroutine area. Subroutine area is not scanned unless directed to do so by a JSR instruction.
–(LBL)–	Label	XX –(LBL)–	This condition instruction is the target destination for JMP and JSR instructions.
–(JMP)–	Jump	XX –(JMP)–	XX – two digit octal identificatio nmber, 00–07.
–(JSR)–	Jump To Subroutine	XX –(JSR)–	When rung is true, processor jumps to referenced label in subroutine area. Same as LBL with which it is used.
–(RET)–	Return	–(RET)–	No identification number. Can be used unconditionally. Returns the processor to the instruction immediately following the JSR in the main program that initiated the jump to subroutine.

Figure A.1
File Instructions

Key Sequence	1770-T3 Display	Instruction Notes
FILE 10	File To File Move Counter Addr: 030 Position: 001 File Length: 001 File A: 110-110 File R: 110-110 Rate Per Scan 001 030 (EN) 17 030 (DN) 15	Output instruction. Modes: Complete, istributed and Incremental. Counter is internally incremented by the instruction. Requires 5 words of user program.
FILE 11	Word To File Move Counter Addr: 030 Position: 001 File Length: 001 Word Adddress: 010 File R: 110-110 030 (DN) 15	Output instruction. Counter must be externally indexed by user program. Data is transferred every scan that rung is true. Requires 4 words of user program.
FILE 12	File To Word Move Counter Addr: 030 Position: 001 File Length: 001 File A: 110-110 Word Address: 010 030 (DN) 15	Same as word-to-file.

NOTE: Numbers shown are default values. Numbers in shaded areas must be replaced by user-entered values. The number of default address digits initially displayed (3 or 4) will depend on the size of the data table.

Here is an explanation of each value:

Counter Address : Address of the instruction in the accumulated value area of data table.

Position : Curent word being operated upon. (Accumulated value of counter.)

File Length : Number of words in file (preset value of the counter).

File A : Starting address of source file.

File R : Starting address of destination file.

Word Address : Address of source word or destination word outside of file.

Rate per scan : Number of data words moved per scan.

NOTE: Access the Data Monitor Display as follows:
Enter all instruction parameters. Press key sequence:
[DISPLAY] [0] for the binary monitor mode;
[DISPLAY] [1] for the hexadecimal monitor mode.

Figure A.2
Sequencer Instructions

Key Sequence	1770–T3 Display	Instruction Notes
SEQ 0	Sequencer Output Counter Addr: 030 Current Step: 001 Seq Length: 001 Words Per Step 1 File: 110–110 Mask: 010–010 Output Words 1: 010 2: 3: 4: 030 EN 17 030 DN 15	Output instruction. Increments, then transfers data. Same data transferred each scan that the rung is true. Counter is indexed by the instruction. Unused output bits can be masked. Requires 5–9 words of your program.
SEQ 1	Sequencer Input Counter Addr: 030 Current Step: 000 Seq Length: 001 Words Per Step: 1 File: 110–110 Mask: 010–010 Input Words 1: 010 2: 3: 4:	Input instruction. Compares input data with current steps for equality. Counter must be externally indexed by your program. Requires 5–8 words of your program.
SEQ 2	Sequencer Load Counter Addr: 030 Current Step: 001 Seq Length: 001 Words Per Step: 1 File: 110–110 Input Words 1: 010 2: 3: 4: 030 EN 17 030 DN 15	Outupt instruction. Increments, then loads data. Counter is indexed by the instruction. Does not mask. Requires 4–7 words of your program.

NOTE: Numbers shown are default values. Numbers in shaded areas must be replaced by your entered values. The number of default address digits initially displayed (3 or 4) will depend on the size of the data table.

Here is an explanation of each value:

Counter Address : Address of the instruction in the accumulated value area of data table.
Position : Position in sequencer table (accumulated value of counter).
Seq Length : Number of steps (preset value of the counter).
Words per Step : Width of sequencer table.
File : Starting address of source file.
Mask : Starting address of mask file.
Output Words : Words controlled by the instruction.
Load Words : Words fetched by the instruction.
Input Words : Words monitored by the instruction.

NOTE: Access the Data Monitor Display as follows:

Enter all instruction parameters. Press key sequence:
[DISPLAY] [0] for the binary monitor mode;
[DISPLAY] [1] for the hexadecimal monitor mode.

Figure A.3
Block Transfer Instructions

Key Sequence	1770–T3 Display	Instruction Notes
BLOCK XFER 0	Block Xfer Write Data Addr: 030 Module Addr: 001 Block Length: 001 File: 110–110 010 (EN) 06 110 (DN) 06	Output instruction. Block length depends on kind of module. Entire file transferred in one scan. Data read from I/O module must be buffered. Uses two words of yser program for each instruction.
BLOCK XFER 1	Block Xfer Read Data Addr: 030 Module Addr: 001 Block Length: 001 File: 110–110 010 (EN) 06 110 (DN) 06	Set block lengths equal or to default value for module. Same module address used for read and write instructions Enable read and write instructions in same scan. Order of operation determined by the module. Refer to the module user's manual.
BLOCK XFER 0 BLOCK XFER 1	Enter both instruction blocks for bidirectional blcok transfer.	

NOTE: Numbers shown are default values. Numbers in shaded areas must be replaced by your entered values. The number of default address digits initially displayed (3 or 4) will depend on the size of the data table.

Here is an explanation of each value:

Data Address : First possible address in accumulated value area of data table.

Module Address : RGS for R = rack, G = module group, S = slot number.

Block Length : Number of words to be transferred. (00 can be entered for default value)

File : Address of first word of the file, 100_8 above the data address.

NOTE: Access the Data Monitor Display as follows:

Enter all instruction parameters. Press key sequence:
[DISPLAY] [0] for the binary monitor mode;
[DISPLAY] [1] for the hexadecimal monitor mode.

Industrial Terminal Commands

This section describes commands available to you when using the 177–T3 industrial terminal. This section contains commands referring to:

- Data table configuration
- Clear memory functions
- Editing functions
- Search functions
- Help directories
- Troubleshooting aids
- Report generation
- Contact histogram functions
- Alphanumeric/graphic keytop definitions
- Industrial terminal control codes

Table A.J
Data Table Configuration

Function	Key Sequence	Mode	Description
Data table configuration	[SEARCH] [5] [0] [Numbers]	Program	If the number of 128–word sections is 1 or 2, enter this number, and the number of timers/counters. If the number of 128–word sections is 3 or greater, enter only this number. The industrial terminal will calculate and display the data table size in decimal.
Processor memory layout	[SEARCH] [5] [4]	Any	Displays the number of words in the data table area, user program area, message area and unused memory.
Either	[CANCEL COMMAND]		To terminate.

Table A.K
Clear Memory Functions

Function	Key Sequence	Mode	Description
Data table clear	[SEARCH] [7] [7] (Start Address) (End Address) [CLEAR MEMORY]	Program	Display a start address and an end address field. Start and end word addresses determine boundaries for data table clearing. Clears the data table within and including addressed boundaries.
User program clear	[CLEAR MEMORY] [8] [8]	Program	Position the cursor at the desired location in the program. Clears user program from the position of the cursor to the first boundary: i.e. temporary end, subroutine area or end statement. Does not clear data table or messages.
Partial memory clear	[CLEAR MEMORY] [9] [9]	Program	Clears user program and clear messages from position of the cursor. Does not clear data table.
Total memory clear	[CLEAR MEMORY] [9] [9]	Program	Position the cursor on the first instruction of the program. Clears user program and messages. Does not clear data table, unless the cursor is on the first program instruction.

NOTE: When Memory write protect is active, memory cannot be cleared except for data table addresses 010–177 with a programmed EPROM installed.

Table A.L
Editing Functions [1]

Function	Key Sequence	Mode	Description
Inserting a condition instruction	[INSERT] (Instruction) (Address) or [INSERT] [←] (Instruction) (Address)	Program	Position the cursor on the instruction that will precede the instruction to be inserted. Then press key sequence. Position the cursor on the instruction that will follow the instruction to be inserted. The press key sequence.
Removing a condition instruction	[REMOVE] (Instruction)	Program	Position the cursor on the instruction to be removed and and press the key sequence.
Inserting a rung	[INSERT] [RUNG]	Program	Position the cursor on any instruction in the preceding rung and press the key sequence. Enter instructions and complete the rung.
Removing a rung	[REMOVE] [RUNG]	Program	Position the cursor anywhere on the rung to be removed and press the key sequence. **NOTE:** Only addresses corresponding to output energize, latch and unlatch instructions are cleared to zero.
Change data of a word or block instruction	[INSERT] (Data)	Program	Position the cursor on the word or block instruction whose data is to be changed. Press the key sequence.
Change the address of a word or block instruction	[INSERT] (First Digit) [←] (Address)	Program	Position the cursor on a word or block instruction with data and press [INSERT]. Enter the first digit of the first data value of the instruction. Then use the [<]and [–>] key as needed to cursor up to the word address. Enter the appropriate digits of the word address.
On–line programming	[SEARCH] [5] [2]		Initiates on–line programming.
Replace an instruction or Change address of an instruction without data	[Instruction] (Address)	Program	Position the cursor on the instruction to be replaced or whose address is to be changed. Press the desired instruction key (of key sequence) and the required address(es).
On–Line Data Change	[SEARCH] [5][1] (Data) [RECORD] [CANCEL COMMAND]	Run/Program	Position the cursor on the word or block instruction whose data is to be changed. Press the key sequence. Cursor keys can be used. Press [RECORD] to enter the new data into memory. To terminate on–line data change.
All editing functions	[CANCEL COMMAND]	Program Run/Program	Aborts the operation at the current cursor position.

These functions can also be used during on–line programming.
[1]When bit address exceeds 5 digits, press the [EXPAND ADDR] key before entering address and enter a leading zero if necessary.

Table A.M
Search Functions

Function	Key Sequence	Mode	Description
Locate first rung of program	[SEARCH] [↑]	Any	Positions cursor on the first instruction of the program.
Locate last rung or program area	[SEARCH] [↓]	Any	Positions cursor on the temporary end instruction,subroutine area boundary, or the end statement depending on the cursor's location. Press key sequence again to move to the next boundary.
Locate first instruction of current rung	[SEARCH] [←]	Program	Position cursor on first instruction of the current rung.
Move cursor off screen	[SEARCH] [→]	Test,Run or Run/Program	Moves cursor off screen to left.
Locate output instruction of current rung	[SEARCH] [→]	Any	Positions cursor on the output instruction of the current rung.
Locate rung without an output instruction	[SHIFT] [SEARCH]	Any	Locates any rung left incomplete due to an interruption in programming.
Locate specific instruction	[SEARCH] [Instruction key] (Address)	Any	Locates instruction searched for. Press [SEARCH] to locate the next occurrence of instruction.
Locate specific word address	[SEARCH][8] (Address)	Any	Locates this address in the program (excluding –[]– and – [/]– instructions and addresses in files). Press [SEARCH] to locate the next occurrence of this address.
Single rung display	[SEARCH] [DISPLAY]	Any	Displays the first rung of a multiple rung display by itself. Press key sequence again to view multiple rungs.
Remote Mode Select: run/program	[SEARCH] [5][9][0]	Run/Program	Places the processor in run/ program mode.
Remote test	[SEARCH] [5][9][1]		Places the processor in remote test mode.
Remote program	[SEARCH] [5][9][2]		Places the processor in remote program mode.

Table A.N
Help Directories

Function	Key Sequence	Mode	Description
Help directory	[HELP]	Any	Displays a list of the keys that are used with the [HELP] key to obtain further directories.
Control function directory	[SEARCH] [HELP]	Any	Provides a list of all control functions that use the [SEARCH] key.
Record function directory	[RECORD] [HELP]	Any	Provides a list of functions that use the [RECORD] key.
Clear memory directory	[CLEAR MEMORY] [HELP]	Program	Provides a list of all functions that use the [CLEAR MEMORY] key.
Data monitor directory	[DISPLAY] [HELP]	Any	Provides the choice of data monitor display accessed by the [DISPLAY] key.
File instruction directory	[FILE][HELP]	Any	Provides a list of all instructions that use the [FILE] key.
Sequencer instruction directory	[SEQ][HELP]	Any	Provides a list of all instructions that use the [SEQ] key.
Block transfer directory	[BLOCK XFER] [HELP]	Any	Provides a list of all instructions that use the [BLOCK XFER] key.
All directories	[CANCEL COMMAND]	Any	To terminate.

Table A.O
Troubleshooting Aids

Function	Key Sequence	Mode	Description
Bit monitor	[SEARCH] [5][3] [Address]	Any	Displays the on/off status of all 16 bits at specified word address and corresponding force conditions if they exist.
	[↑] or [↓]		Displays the status of 16 new bits at the next lowest or highest word address, respectively.
Bit manipulation	[SEARCH] [5][3]	Test or Run/Program	Displays the on/off status of all 16 bits at specified word address and corresponding force conditions if they exist.
	[<–] or [–>]		Moves cursor to the bit to be changed.
	[1] or [0]		Enter a 1 to set bit on or a 0 to reset a bit.
	See FORCING below		Forcing or removing forces from input bits or output devices.
Either of above	[CANCEL COMMAND]		To terminate.
Force On	[FORCE ON] [INSERT]	Test or Run/Program	Position the cursor on the image table bit to be forced on and press the key sequence. The input bit or output device will be forced on.
Removing a Force On	[FORCE ON] [REMOVE]	Test or Run/Program	Position the cursor on the image table bit whose force on is to be removed and press the key sequence.
Removing all Force On	[FORCE ON] [CLEAR MEMORY]	Test or Run/Program	Position cursor anywhere in program and press key sequence.
Force Off	[FORCE OFF] [INSERT]	Test or Run/Program	Position the cursor on the image table bit to be forced off and press the key sequence. The input bit or output device will be forced off.
Removing a Force Off	[FORCE OFF] [REMOVE]	Test or Run/Program	Position the cursor on the image table bits whose force off is to be removed and press the key sequence.
Removing all Force Off	[FORCE OFF] [CLEAR MEMORY	Test or Run/Program	Position the cursor anywhere in program and press key sequence.
Forced address display	[SEARCH] [FORCE ON] or [SEARCH] [FORCE OFF]	Any	Displays a list of the bit addresses that are forced on and forced off. The [SHIFT] [ ] and [SHIFT] [ ] keys can be used to display addition forces.
	[CANCEL COMMAND]		To terminate.
Inserting a temporary end instruction	[INSERT] [ ] [T.END] or	Program	Position the cursor on the on the instruction that will follow the temporary end instruction. The remaining rungs, although displayed and accessible, are not scanned.

Function	Key Sequence	Mode	Description
	[INSERT] [T.END]		Position the cursor on the instruction that will precede the temporary end instruction. The remaining rungs, although displayed and accessible, are not scanned.
Removing a temporary end instruction	[REMOVE] [T.END]	Program	Position cursor on temporary end instruction and press key sequence.

NOTE: When in test mode, the processor will hold outputs off regardless of attempts to force them on.

Table A.P
Report Generation Commands

Command	Key Sequence	Description
Enter report generation function	[RECORD][DISPLAY] or	Puts industrial terminal into report generation function.
	Set baud rate, Message Code Keys)	Same (entered from a peripheral device).
Message store	[M][S][,] (Message Number) [RETURN]	Stores message in processor memory. Use [ESC] to end message.
Message print	[M] [P] [,] (Message Number) [RETURN]	Prints message exactly as entered.
Message report	[M][R][,}(Message Number) [RETURN]	Prints message with current data table values or bit status.
Message delete	[M][D][,] (Message Number) [RETURN]	Removes message from processor memory.
Message Index	[M][I][RETURN]	Lists messages used and the number of words in each message.
Automatic report generation	[SEARCH][4}[0} or [M][R][RETURN]	Allows messages to be printed through program control.
		Same (entered from a peripheral device).
Exit automatic report generation	[ESC] or [CANCEL COMMAND]	Terminates automatic report generation.
		Same (entered from a peripheral device).
Exit report generation function	[ESC] or [CANCEL COMMAND]	Returns to ladder diagram display.
		Same (entered from a peripheral device).

NOTE: [CANCEL COMMAND] can only be used if the function was entered by a command from a peripheral device.

Table A.Q
Contact Histogram Functions

Function	Key Sequence	Mode	Description
Continuous contact histogram	[SEARCH][16] (Bit Address) [DISPLAY]	Run Run/Program or Test	Provides a continuous display of the on/off history of the addressed bit in hours, minutes and seconds. Can obtain a hardcopy printout of contact histogram by connecting a peripheral device to Channel C and selecting proper baud rate before indicated key sequence.
Paged Contact histogram	[SEARCH][17] 9Bit Address) [DISPLAY] [DISPLAY]	Run Run/Program or Test	Display 11 lines on/off history of the addressed bit in hours, minutes and seconds. Displays the next 11 lines of contact histogram. Can obtain a hard copy printout of contact histogram by connecting peripheral device to Channel C and selecting proper baud rate.
Either	[CANCEL COMMAND]		To terminate.

Table A.R
Alphanumeric/Graphic Key Definitions

Key	Function
[LINE FEED]	Moves the cursor down one line in the same column.
[RETURN]	Returns the cursor to the beginning of the next line.
[RUB OUT]	Deletes the last character or control code that was entered.
[REPT LOCK]	Allows the next character that is pressed t be repeated continuously until [REPT LOCK] is pressed again.
[SHIFT]	Allows the next key pressed to be a shift character.
[SHIFT LOCK]	Allows all subsequent keys pressed to be shift characters until [SHIFT] or [SHIFT LOCK] is pressed.
[CTRL]	Used as part of a key sequence to generate a control code.
[ESC]	Terminates the present function.
[MODE SELECT]	Terminates all functions and returns the mode select display to the screen.
Blank Yellow Keys	Space keys. Move the cursor one position to the right.

Table A.S
Industrial Terminal Control Codes

Control Code Key Sequence	Function
[CTRL][P] [Column #][;] [Line #][A]	Positions the cursor at the specified column and line number. [CTRL][P][A] will position the cursor at the top left corner of the screen.
[CTRL][P][F]	Moves the cursor one space to the right.
[CTRL][P][U]	Moves the cursor one line up in the same column.
[CTRL][P][5][C]	Turns cursor on.
[CTRL][P][4][C]	Turns cursor off.
[CTRL][P][5][G]	Turns on graphics capability.
[CTRL][P][4][G]	Turns off graphics capability.
[CTRL][P][5][P]	Turns Channel C outputs on.
[CTRL][P][4][P]	Turns Channel C outputs off.
[CTRL][I]	Horizontal tab that moves the cursor to the next preset 8th position.
[CTRL][K]	Clears the screen from cursor position to end of screen and moves the cursor to the top left corner of the screen.
Key Sequence	**Attribute**
[CTRL][P][0][T]	Attribute 0 = Normal Intensity
[CTRL][P][1][T]	Attribute 1 = Underline
[CTRL][P][2][T]	Attribute 2 = Intensify
[CTRL][P][3][T]	Attribute 3 = Blinking
[CTRL][P][4][T]	Attribute 4 = Reverse Video
Any three attributes can be used at one time using the following key sequence: [CTRL][P][Attribute #][;][Attribute #][;][Attribute #][T]	

Appendix B

Glossary

AC Input Module
I/O module which converts various AC signals originating at user devices to the appropriate logic level for use with the processor.

AC Output Module
I/O module which converts the logic levels of the processor to a usable output signal to control a user's AC load.

Active
A green diagnostic indicator on Allen-Bradley PC hardware that, when illuminated, signifies normal communication exists between the processor and a remote I/O chassis.

Acquisition
A function which obtains information from PC memory locations or data files for use in data manipulation or data handling.

Address
A location in the processor's memory; usually used in reference to the data table.

Alternating Current (AC)
Current in which the charge-flow periodically reverses direction.

Application
Any machine or process which is monitored and controlled by a PC by means of a user program.

Arithmetic Capability
The ability to do addition, subtraction, multiplication and division, with the PC processor.

Backplane
A printed circuit card located in the back of a chassis. It has sockets into which specific modules fit for interconnection.

Battery Low
1) A condition caused when the memory backup battery voltage drops low enough to warrant battery replacement. 2) An indicator which signals this condition.

Binary
A numbering system using only the digits 0 and 1. Also called "base 2."

Binary Coded Decimal (BCD)
A method used to express individual decimal digits (0 thru 9) in 4-bit binary notation; e.g., the number 23 is represented as 0010 0011 in the BCD notation.

BCD	DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Binary Image
An exact copy of the states (“1” and “0”) of all bits in PC memory, including the data table, program, data files, and message areas.

Binary Word
A related grouping of ones and zeroes having meaning assigned by position, or numerical value in the binary system of numbers.

Bit
1) An acronym for binary digit; the smallest unit of information in the binary numbering system. Represented by the digits 0 and 1. 2) The smallest division of a PC memory word.

Bit Manipulation
The process of controlling and monitoring individual special-purpose data table bits through user-programmed instructions in order to vary application functions.

Bit Rate
The rate at which binary digits, or pulses representing them, pass a given point in a communication line.

Bit Storage
A single bit in any unused data table word which may be individually energized or de-energized without directly controlling any output. However, any storage bit may be monitored as often as necessary in the user’s program in order to control various outputs indirectly.

Block Diagram
A simplified schematic drawing.

Branch
A parallel logic path within a user program rung.

Byte
A sequence of binary digits usually operated upon as a unit. (The exact number depends on the system.) In Allen-Bradley PCs, this is the smallest complete unit of information that can be transmitted, and is made up of 8 data bits plus a parity bit.

Cascading
A programming technique that extends the ranges of timer and/or counter instructions beyond the maximum values that may be accumulated. This is accomplished by means of other PC instructions.

Cassette Recorder
A peripheral device for transferring information between PC memory and magnetic tape. In the record mode it is used to make a permanent record of a program existing in the processor's memory. In the playback mode it is used to enter a previously recorded program into the processor's memory.

Cassette Tape
A magnetic recording tape permanently enclosed in a protective housing. The Philips type cassette is most common. Cassettes for PC use should contain computer grade tape.

Catalog Number (cat. no.)
A designation for a specific Allen-Bradley PC unit with the major product line. It is a 4-digit number generally followed by a dash and 2 or three alphanumeric characters.

Central Processing Unit (CPU)
Another term for processor. It includes the circuits controlling the interpretation and execution of the user-inserted program instructions stored in the PC memory.

Chassis
The piece of hardware that the processor and its associated modules set into when operating.

Character
One symbol of set of elementary symbols, such as a letter of the alphabet or a decimal numeral.

Clear
To return a memory to a non-programmed state, usually represented as "0" or off.

CMOS
Acronym for Complementary Metal Oxide Semiconductor circuitry. An integrated circuit family which has high threshold logic and low power consumption, thus making it especially useful in remote applications where supplying power becomes expensive.

The type of memory used in the Mini-2/15 programmable controller.

Code
A system of symbols (bits) for representing data.

Coding
The preparation of a set of instructions or symbols which, when used by a programmable controller, have a special external meaning.

Color Code
A color system for component identification by use of solid colors, tracers, braids, surface printing, etc.

Compare Function
A user-programmed instruction which equates numerical values for “equal” or “less than” relationships in order to vary operation sequence or application.

Compatibility
The ability of various specified units to replace one another, with little or no reduction in capability.

Control
1) A unit, such as a PC or relay panel, which operates an industrial application. 2) To cause a machine or process to function in a predetermined manner. 3) To energize or de-energize a PC output or set to “1” (on) or reset to “0” (off) a data table bit, by means of user-programmed instructions.

Core Memory
A type of memory used to store information in ferrite cores. Each may be magnetized in either polarity to represent a logic “1” to “0”. This type of memory is non-volatile.

Counter
In relay-panel hardware, an electro-mechanical device which can be wired and preset to control other devices according to the total cycle of one on and off function. in pc, a counter is internal to the processor, which is to say it is controlled by a user-programmed instruction. A counter instruction has greater capability than any hardware counter. Therefore, most PC applications do not require hardware counters.

Cursor
A means for indicting on a CRT screen the point at which data entry or editing will occur. The intensified element may be at constant high intensity or flashing (alternate high intensity and normal intensity). If flashing, additional data may be necessary to complete the instruction.

Cursored Rung
The top rung in a multiple rung display. it is the only rung which can be altered by means of the program panel in this display mode. (However, any other rung can be moved to this cursored rung position through scrolling.)

Cycle
1) A sequence of operations that is repeated regularly. 2) The time it takes for one such sequence to occur.

Data
A general term for any type of information.

Data Files
Groups of application data values stored after, and separate from, the user program area in the PC memory in the data table. These Files are manipulated by, and used with, the user's program as the application requires formula changes.

Data Highway
A single-cable, differential,half-duplex, serial data link which provides communication among multiple stations which are separate PCs, computers, and data terminals. It eliminates the need for separate, independently wired data links. Whether communication or not, all stations may function independently.

Data Link
Equipment, especially transmission cables and interface modules, which permits the transmission of information.

Data Manipulation
The process of altering and/or exchanging data between storage words through user-programmed PC instructions in order to vary application functions.

Data Table
A major portion of PC memory which is monitored and controlled through both PC instructions and processor electronics.

Data Transfer
The process of exchanging data between PC memory areas through user-programmed PC instructions in order to vary application functions.

Debugging
Process of detecting, locating, and correcting mistakes in hardware or software.

Diagnostic Program
A user-inserted test program to help isolate hardware malfunctions in the programmable controller and application equipment.

Direct Current (DC)
An electric current which flows in only one direction.

Downtime
The time when a system is not available for production due to required maintenance.

Edit
To deliberately modify the user program in the PC memory.

Electrical-optical Isolator
A device which couples input to output using a light source and detector in the same package. it is used to provide electrical isolation between input circuitry, output circuitry, and processor circuitry. Same as opto-electrical.

Enable
To cause a particular function to occur by means of preconditions with PC program logic.

Enclosure
A surrounding case designed to provide a degree of protection for equipment against a specified environment and to protect personnel against accidental contact with the enclosed equipment.

Energize
This instruction sets a data table bit to “1” (on) if the preconditions in its rung are true. The bit is reset to “0” (off) if the preconditions are false.

Examine Off
This instruction is a true precondition if its addressed data table bit is off (“0”). It is false if the bit is on (“1”).

Examine On
This instruction is a true precondition if its addressed data table bit is ON (“1”). It is false if the bit is off (“0”).

Execution
The performance of a specific operation such as would be accomplished through processing one instruction, a series of instructions, or a complete program.

Execution Time
The total time required for the execution of one specific operation.

Extended Data Comparison
A user-programmed on-line application diagnostic routine. At various steps in

the application cycle, certain data files are compared to the data table. whenever a discrepancy is detected, a printed report describes the type of problem and its location.

False
As related to PC instructions, a disabling logic state. Fault
Any malfunction which interferes with normal application operation.

Feedback
The signal or data sent to the PC from a controlled machine or process to denote its response to the command signal.

Fence Codes
Special user-programmed PC instructions which control and delimit specific program areas such as fault zones and conditional ignore zones.

Force Off Function
A feature which allows the user to de-energize, independent of the PC program, any input or output by means of the program panel.

Force On Function
A feature which allows the user to energize, independent of the PC program, any input or output by means of the program panel.

Get Byte Instruction
A PC instruction which accesses either the upper or lower 8-bit byte of an addressed 16 bit data table word. This instruction functions only with limit test instructions.

Hard Contact
Any type of physical switch contacts. Contrasted with electronic switching devices, such as triacs and transistors.

Hard Copy
Any form of printed document such as ladder diagram program listing, paper tape, or punched cards.

Hardware
The mechanical, electrical and electronic devices which compose a programmable controller and its application.

Image Table
An area in PC memory dedicated to I/O data. Ones and zeroes (“1” and “0”) represent on and off, respectively, conditions. During every I/O scan, each input controls a bit in the input image table; each output is controlled by a bit in the output image table.

Immediate Input Instruction
A PC instruction that immediately transfers input data from selected input

modules to the associated 16-bit word in the input image table without waiting for the normal I/O scan.

Input Devices
Devices such as limit switches, pressure switches, push buttons, etc., that supply data to a programmable controller. These discrete inputs are two types those with common return and those with individual returns (referred to as isolated inputs). Other inputs include analog devices and digital encoders.

Instruction
A command or order that will cause a PC to perform one certain prescribed operation. The user enters a combination of instructions into PC memory to form a unique application program.

Interfacing
Interconnecting a PC with its application devices, and data terminals through various modules and cables. Interface modules convert PC logic levels into external signal levels, and vice versa.

I/O Chassis
Same as chassis.

I/O Module
The printed circuit assembly that interfaces between the user devices and the PC.

I/O Rack
Same as rack.

I/O Scan Time
The timer required for the PC processor to monitor all inputs and control all outputs. The I/O Scan repeats continuously.

Keying
Keying bands installed in various backplane module sockets in order to ensure that only certain modules can be inserted into designated sockets.

Ladder Diagram
An industry standard for representing control logic relay systems.

Ladder Diagram Programming
A method of writing a user's PC program in a format similar to a relay ladder diagram.

Language
A set of symbols and rules for representing and communicating information (data) among people, or between people and machines.

Latch Instruction
A PC instruction which causes a bit to stay on, regardless of how briefly the

instruction is enabled. (It can only be turned off by an unlatch instruction in a separate rung.)

Latching Relay
A relay constructed so that it maintains a given position by mechanical or electrical means until released mechanically or electrically.

LED
Acronym for Light-Emitting Diode.

LED Display
An illuminated visual readout composed of lead alphanumeric character segments.

Limit Switch
A switch which is actuated by some part or motion of a machine or equipment to alter the electrical circuit associated with it.

Limit Test Instruction
This PC instruction accesses two user-programmed values which are the upper and lower limits for testing a number accessed by a get byte instruction. This get byte-limit test combination is a true precondition if the number is between the upper and lower limits.

Line
1) A component part of a system used to link various sub-systems located remotely from the processor. 2) The source of power for operation, e.g., 120V AC line.

Load
1) The power delivered to a machine or apparatus. 2) A device intentionally placed in a circuit or connected to a machine or apparatus to absorb power and convert it into the desired useful form. 3) To insert data and memory storage.

Load Resistor
A resistor connected in parallel with a high impedance load so that the output circuit driving the load can provide a t least the minimum current required for proper operation.

Logic
A means of solving complex problems through the repeated use of simple functions which define basic concepts. Three basic logic functions are AND, OR, and NOT.

Logic Diagram
A drawing which represents the logic functions and/or, not, etc.

Logic Family
Group of digital integrated circuits sharing a basic circuit design with standardized input-output characteristics.

Magnetic Core Memory
(See core memory.)

Malfunction
Any incorrect functioning within electronic, electrical, or mechanical hardware. (See fault.)

Manipulation
The process of controlling and monitoring data table bits or words by means of the user's program in order to vary application functions.

Master Control Relay
A mandatory hardwired relay which can be de-energized by any hardwired series-connected emergency stop switch. Whenever the master control relay is de-energized, its contacts must open to de-energize all application I/O devices.

> **WARNING:** The master control relay must never be replaced by MCR fence codes.

MCR Instructions
User-programmed fence codes for MCR zones.

MCR Zones
user program areas in which all non-retentive outputs can be turned OFF simultaneously. Each MCR zone must be delimited and controlled by MCR fence codes (MCR instructions).

> **WARNING:** MCR FENCE CODES must never replace the master control relay hardware.

Memory
A grouping of circuit element which has data storage and retrieval capability.

Memory Module
A processor memory storage module consisting of memory storage and capable of storing a finite number of words.

Message
A meaningful combination of alphanumeric characters which establishes the content and format of a report. it must be entered into PC memory by means of a data terminal keyboard.

Mode
A selected method of operation (e.g., run, test, or program).

Module
An interchangeable “plug-in” item containing electronic components which may be combined with other interchangeable items to form a complete unit.

Module Group
Adjacent I/O modules which relate 16 I/O terminals to a single 16-bit image table word. A Mini-PLC-2 module grouping contains 2 modules, each with 8 I/O terminals.

Monitor
1) CRT display package. 2) To observe an operation.

Non-Retentive Output
An output which is continuously controlled by a single program rung. Whenever the rung changes state (true or false), the output turns on or off. (Contrasted with a retentive output which remains in its last state (on or off) depending on which of its two rungs, latch or unlatch, was last true.)

Non-Volatile Memory
A memory that is designed to retain its information while its power supply is turned off without requiring a backup power source.

Octal Numbering System
One which uses a base eight, e.g., the decimal number 324 would be written in octal notation as 5048. Only the digits 0 thru 7 are used.

Off-Delay Timer
1) In relay-panel application, a device in which the timing period is initiated upon de-energization of its coil. 2) In PC, an instruction which turns off one or more outputs (by means of other rungs) after a programmed time delay. While the preconditions of the off-delay timer rung are true, the outputs are energized. Whenever the rung goes false, the time delay is started, and the outputs are de-energized at its completion.

On-Delay Timer
1) In relay panel applications, a device in which the timing period is initiated upon energization of its coil. 2) In PC, an instruction which turns on one or more outputs (by means of other rungs) after a programmed time delay. While the preconditions of the on-delay timer rung are false, the outputs are de-energized. whenever the rung goes true the time delay is started, and the outputs are energized at its completion.

On-Line Data Change
This feature allows the user to change various data table values through the program panel while the application is operating normally.

On-Line Operation
Operations where the programmable controller is directly controlling the machine or process.

Output
Information transferred from PC image table words through output modules to control output devices.

Output Devices
Devices such as solenoids, motor starters, etc., that receive data from the programmable controller.

Parity
A method of testing the accuracy of binary numbers used in recorded, transmitted, or received data.

Parity Bit
An additional bit added to a binary word to make the sum of the number of “1s” in a word always even or odd.

Parity Check
A check that tests whether the number of “1s” in an array of binary digits is odd or even.

PC
Abbreviation for programmable controller.

Peripheral Equipment
Units which may communicate with the programmable controller, but are not apart of the progammable controller; e.g., cassette recorder, tape reader, terminal or computer.

Power Supply
In general a device which converts AC line voltage to one or more DC voltages. 1) A PC power supply provides only the DC voltages required by the electronic circuits internal to the PC. 2) A separate power supply, installed by the user, to provide any DC voltages required by the application input and output devices.

Process
1) Continuous and regular production executed in a definite uninterrupted manner. 2) A PC application which primarily requires data comparison and manipulation. The PC monitors the input parameters in order to vary the output values. (As generally contrasted with a machine, a process does not cause mechanical motion.)

Processor
A unit in the programmable controller which scans all the inputs and outputs in a predeteremined order. The Processor monitors the status of the inputs and outputs in response to the user programmed instructions in memory, and it energizes or de-energizes outputs as result of the logical comparisons made through these instructions.

Program
A sequence of instructions to be executed by the PC processor to control a machine or process.

Program Panel
A device for inserting, monitoring, and editing a program in a programmable controller.

Program Scan Time
The time required for the PC processor to execute all instructions in the program once. The program scan repeats continuously. The program monitors inputs and controls outputs through the input and output image tables.

Programmable Controller
A solid state control system which has a user programmable memory for storage of instructions to implement specific functions such as
I/O control logic, timing, counting, arithmetic, and data control logic, timing, counting, arithmetic, and data manipulation. A PC consists of central processor, input/output interface, memory, and programming device which typically uses relay-equivalent symbols. PC is purposely designed as an industrial control system which can perform functions equivalent to a relay panel or a wired solid state logic control system.

PROM
Acronym for Programmable Read Only memory. A type of ROM that requires an electrical operation to generate the desired bit or word pattern. In use, bits or words are accessed on demand, but not changed.

Protected Memory
Storage (memory) locations reserved for special purposes in which data cannot be entered directly by the user.

Put Instructions
A PC instruction which is used with a preceding get instruction to transfer data from one data table word to another. The on and off states of all bits in the word addressed by get are duplicated in the word addressed by put.

Rack
A PC chassis that contains modules (e.g., I/O rack or processor rack).

Rack Fault
1) A red diagnostic indicator which illuminates to signal a loss of communication between the processor and any Remote I/o chassis. 2) The condition which is based on the loss of communication.

RAM
Acronym for Random Access Memory. RAM is a type of memory that can be accessed (read from) or loaded (written into) depending on the particular addressing and operation codes generated internally in the PC.

Read
1) Accessing of data from a storage device such as memory, magnetic tape, etc. 2) Transfer of data between devices, such as between a peripheral device and a computer.

Read/Write Memory
A memory in which data can be placed (write mode) or accessed (read mode). The write mode destroys previous data, read mode does not alter stored data.

Remote I/O PC
A type of programmable controller in which some or all of the I/O chassis are mounted in separate enclosures. Any remote I/O chassis may be located either close to the PC processor or as far away as 5000 or 10,000 cable feet, depending on the specific hardware.

Remote Mode Selection
A feature which allows the user to select or change processor modes by means of a program panel from as far away as 5000 cable feet.

Report
An application data display or printout containing information in a user-designed format. Reports include operator messages, part records, production lists, etc. Initially entered as messages, reports are stored in a memory area separate from the user's program.

Report Generation
The printing or displaying of user-formatted application data by means of a data terminal. Report generation can be initiated by means of either the user's program or a data terminal keyboard.

Retentive Output
An output that remains in its last state (on or off) depending on which of its two program rungs (one containing a latch instruction, the other an unlatch) was the last to be true. The retentive output remains in its last state while both rungs are false. It also remains in its last state if power is removed from, then restored to, the PC (Contrasted with a non-retentive output which is continuously controlled by single rung.)

Retentive Timer
A PC instruction which accumulates the amount of time, whether continuous or not, that the preconditions of its rung are true, and controls one or more outputs (by means of other rungs) after the total accumulated time is equal to the programmed time. Whenever the rung is false, the accumulated time is retained. Moreover, if the outputs have been energized, they remain on. Also, the accumulated time and energized outputs are retained if power is removed from, then restored to, the PC. A separate rung, containing a retentive timer reset instruction, must be programmed in order to reset the accumulated time to zero and turn off the outputs.

Revision
A firmware change which does not greatly affect unit or module function. In Allen-Bradley PCs this is indicated by a slash and a letter following the series letter (e.g., series B/C).

ROM
Acronym for Read Only memory. A ROM is a solid state digital storage memory whose contents cannot be altered by the PC.

Routine
A sequence of PC instructions which monitors and controls a specific application function.

Rung
A grouping of PC instructions which controls one output or storage bit. This is represented as one section of a logic ladder diagram.

Scan Time
The time necessary to completely execute the entire PC program one time.

Scanner Module
A basic PC processor module which provides I/O data communication between the processor and the I/o chassis. As it scans the I/O racks, the states (“1” and “0”) of the image table bits monitor and control the states (on and off) of the I/O module terminals.

Scrolling
A multiple rung display function which allows all displayed rungs to be moved up or down, adding the next (or preceding) rung at the bottom (or top) of the display. As determined by the user, the display may be changed either one rung at a time or continuously.

Search Function
A PC programming equipment feature which allows the user to quickly display and/or edit any instruction in the PC program.

Sequencer
A controller which operates an application through a fixed sequence of events. (See mechanical drum programmer). In contrast, a PC functions according to varying I/O patterns.

Serial Operation
Type of information transfer within a programmable controller whereby the bits are handled sequentially rather than simultaneously, as they are in parallel operation. Serial operation is slower than parallel operation for equivalent clock rate. However, only one channel is required for serial operation.

Series
A hardware design change which affects the form, fit, and/or function of a unit or module. In Allen-Bradley PCs, this is indicated by the word series and a letter following the catalog number (e.g., cat. no. 1774-DL2, series B).

Solenoid
A electromagnet with a movable core, or plunger, which, when it is energized, can move a small mechanical part a short distance.

State
The logic “0” or “1” condition in PC memory or at a circuit’s input or output.

Static
Refers to a state in which a quantity does not change appreciably within an arbitrarily long time interval.

Storage
Synonymous with memory.

Switching
The action of turning on and off a device.

System
A collection of units combined to work as a larger integrated unit having the capabilities of all the separate units.

Terminal
Any fitting attached to a circuit or device for convenience in making electrical connections.

Terminal Address
An Allen-Bradley 5-digit number which identifies a single I/O terminal. It is also related directly to a specific image table bit address.

Thumbwheel Switch
A rotating numeric switch used to input numeric information to a controller.

Timer
In relay-panel hardware, an electromechanical device which can be wired and present to control the operating interval of other devices. In PC, a timer is internal to the processor, which is to say it is controlled by a user- programmed instruction. A timer instruction has greater capability than a hardware timer. Therefore, most PC applications generally use timer instructions.

Tolerance
A specified allowance for error from a desired or measured quantity.

True
A related to PC instructions, an enabling logic state.

Unlatch Instruction
A PC instruction which causes an output to stay off, regardless of how briefly the instruction is enabled. (It can only be turned on by a latch instruction in a separate rung.)

Variable Data
Numerical information which can be changed during application operation. It includes timer and counter accumulated values, thumbwheel settings, and arithmetic results.

Volatile Memory
A memory that loses its information if the power is removed from it and if it does not have a backup power source.

Voltage
The term most often used in place of electromotive force, potential, potential difference, or voltage drop. It describes the electric pressure that exists between two points and is capable of producing a flow or current when a closed circuit is connected between the two points.

Word
A grouping or a number of bits in a sequence that is treated as a unit.

Word Length
The number of bits in a word, in PC literature these are generally only data bits. One PC word - 16 data bits.

Word Storage
An unused data table word which may be used to contain numerical information without directly controlling any outputs. Any storage word may be monitored as often as necessary by the user program.

Work Area
A portion of the data table reserved for specific processor functions.

Write
The process of leading information into memory.

ZCL Instructions
User-programmed fence for ZCL zones.

ZCL Zones
Distinct program areas which control the same outputs, through separate rungs, at different times. Each ZCL zone is delimited and controlled by ZCL fences (ZCL instructions). for any grouping of outputs, the user's program must enable only one ZCL zone at any time. (If all ZCL zones are disabled, the outputs would remain in their last states.)

3-Digit Address
This identifies a specific 16-bit word in the first 51210 words (000 thru 777) of an Allen-Bradley PC data table.

4-Digit Address
This identifies a specific 16-bit word in the second 512 words (address 1000 thru 1777) of an Allen-Bradley PC data table. it also identifies a byte.

5-Digit Address
This identifies a specific bit in an Allen-Bradley PC data table. it also identifies an I/O terminal directly related to an image table bit. It also identifies a byte.

5-Digit Code
The user-programmed address in any Allen-Bradley PC instruction which can monitor or control a single data table bit. Whenever a 5-digit code refers to a 5-digit address in an image table, it also identifies the corresponding terminal address on a specific I/O module.

7-Segment Display
A device which can exhibit alphanumeric characters. The individual segments can be enabled in various combinations to display all decimal numerals (0 thru 9) as well as many alphabetical characters. Several displays may be combined to exhibit multi-digit numbers.

8-Bit Word
This word size is used by certain PCs of limited capability. However, Allen-Bradley PCs use 16-bit words which provide a more powerful instruction set and greater memory capacity.

16-Bit Word
The number of bits in one word.

Symbols

A

B

C

D

E

Rockwell Automation

Allen-Bradley

Allen-Bradley, a Rockwell Automation Business, has been helping its customers improve productivity and quality for more than 90 years. We design, manufacture and support a broad range of automation products worldwide. They include logic processors, power and motion control devices, operator interfaces, sensors and a variety of software. Rockwell is one of the worlds leading technology companies.

Worldwide representation.

Argentina • Australia • Austria • Bahrain • Belgium • Brazil • Bulgaria • Canada • Chile • China, PRC • Colombia • Costa Rica • Croatia • Cyprus • Czech Republic • Denmark • Ecuador • Egypt • El Salvador • Finland • France • Germany • Greece • Guatemala • Honduras • Hong Kong • Hungary • Iceland • India • Indonesia • Ireland • Israel • Italy • Jamaica • Japan • Jordan • Korea • Kuwait • Lebanon • Malaysia • Mexico • Netherlands • New Zealand • Norway • Pakistan • Peru • Philippines • Poland • Portugal • Puerto Rico • Qatar • Romania • Russia–CIS • Saudi Arabia • Singapore • Slovakia • Slovenia • South Africa, Republic • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • United Arab Emirates • United Kingdom • United States • Uruguay • Venezuela • Yugoslavia

Allen-Bradley Headquarters, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414 382-2000 Fax: (1) 414 382-4444

Publication 1772-6.8.2 – March, 1984
Supersedes Publication 1772–804 March, 1984

PN 955094–68