MicroMentor

Understanding and Applying Micro Programmable Controllers

Understanding and Applying Micro Programmable Controllers

MICROMENTOR

MicroMentor

Understanding and Applying
Micro Programmable Controllers

Solid state equipment has operational characteristics differing from those of electromechanical equipment. “Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls” (Publication SGI-1.1, Allen-Bradley Company) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

In no event will the Allen-Bradley Company be responsible or liable for indirect or consequential damage resulting from the use or application of this equipment.

The examples and diagrams in this book are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, the Allen-Bradley Company cannot assume responsibility or liability for actual use based on the examples and diagrams.

 Printed in USA

TABLE OF CONTENTS

Page

Preface vii

Chapter 1 —
Introduction to PLCs

1.0 History of PLCs 2
1.1 Why Use a PLC? 4
1.2 Traditional PLC Applications. 5

Chapter 2 —
The Micro PLC

2.0 Development of the Micro PLC 8
2.1 What Makes a Micro PLC a Micro? . . . 10
2.2 Capabilities Overview 13
2.3 Micro PLC Applications 14

Chapter 3 —
Micro PLC Operation

3.0 Components Overview. 18
3.1 Inputs 18
3.2 Outputs 20
3.3 Central Processing Unit – CPU 21
3.4 Types of Application Memory 22
3.5 Data, Memory and Addressing. 23
3.6 Operating Cycle 24
3.7 Power Supplies 26
3.8 Programming Devices 28
3.9 Operator Interfaces 30

Page

Chapter 4 — Ladder Logic Fundamentals

4.0 Programming Languages 34
4.1 Electrical Ladder Diagrams 35
4.2 Ladder Logic Programs 36
4.3 Ladder Logic Instructions 39
4.4 Combining Instructions 44
4.5 Program Execution 49

Chapter 5 — How to Apply a Micro PLC

5.0 What is a Potential Control Application? 52
5.1 What are the Application's Requirements? 53
5.2 Selecting a Control Method 60
5.3 What are the PLC Specifications? 66
5.4 Program Development Procedures. . . . 70
5.5 Installation Requirements 80

Page

Chapter 6 — Commissioning and Troubleshooting

6.0 Commissioning. 84
6.1 Troubleshooting Overview. 86
6.2 Finding the Problem 87
6.3 Troubleshooting the PLC. 88
6.4 Troubleshooting I/O. 88
6.5 Program Troubleshooting 91
6.6 Faults. 92
6.7 Safety. 92
6.8 Troubleshooting Model. 93

Chapter 7 — Application Examples

7.0 Introduction. 96
Basic Logic
– 7.1 OR Circuit 96
– 7.2 AND Circuit. 97
– 7.3 Start/Stop Circuit 98
– 7.4 Flip/Flop Circuit 100
– 7.5 Alarm Circuit. 102
– 7.6 Start/Stop with Jog 104

Page

Timing and Counting
– 7.7 On Delay 106
– 7.8 Off Delay 108
– 7.9 One Minute Clock. 110
– 7.10 Up/Down Counting 112

Data Instructions
– 7.11 Moving Data. 114
– 7.12 Comparing Data. 117
– 7.13 Math Commands 120

Advanced Instructions
– 7.14 Sequencers. 123
– 7.15 FIFO 125
– 7.16 High-Speed Counter 128
– 7.17 Two Stage Alternator. 129
– 7.18 Three Station Alternator 133

Appendices

Glossary . . . . . (Appendix A) . . . 142
Input and Output Devices . . . . (Appendix B) . . . 153
Instruction Execution Times . . . (Appendix C) . . . 161
Sample Program Worksheets . . . . . (Appendix D) . . . 164

Index 167

PREFACE

Welcome to "*MicroMentor – Understanding and Applying Micro Programmable Controllers.*" In less than a decade, micro PLCs have gone from a blueprint to one of the fastest growing segments of the control products market. Unfortunately, scant literature exists about micro PLCs. In addition, many of the current PLC text books are too cumbersome for today's busy personnel.

With those thoughts in mind, Allen-Bradley produced this book as an introduction to micro PLCs for the design engineer, electrical technician and maintenance person with little or no background in programmable logic controls. Readers will quickly learn about the micro PLC's evolution, capabilities, operation, and advantages over other control options. Non-specific to any manufacturer, the text also covers basic programming, instructions, application examples, and troubleshooting.

Written to be easily understood, the MicroMentor can augment classroom material, and it can serve as a supplement to the operator manuals and technical data supplied by micro PLC manufacturers.

The authors hope that those experienced with micro PLCs will use this book as a training aid, and that MicroMentor prompts all readers to ask, "What is the best control solution for my application?"

Good Luck!

CHAPTER 1

Introduction to PLCs

History of PLCs 1.0
Why Use a PLC? 1.1
Traditional PLC Applications 1.2

1

1.0

History of PLCs

A programmable logic controller (PLC) is an electronic device that controls machines and processes. It uses a programmable memory to store instructions and execute specific functions that include On/Off control, timing, counting, sequencing, arithmetic, and data handling.

PLC development began in 1968 in response to a request from the Hydramatic Division of General Motors. At the time, GM frequently spent days or weeks replacing inflexible relay-based control systems whenever it changed car models or made line modifications. To reduce the high cost of rewiring, GM's control specification called for a solid state system that had the flexibility of a computer, yet could be programmed and maintained by plant engineers and technicians. It also had to withstand the dirty air, vibration, electrical noise, humidity and temperature extremes found in the industrial environment.

The first PLCs were installed in 1969 and quickly became a success. Functioning as relay replacements, even the early PLCs were more reliable than relay-based systems, largely due to the ruggedness of their solid state components compared with the moving parts in electromechanical relays. PLCs provided material, installation, troubleshooting and labor cost savings

1

An early PLC

by reducing wiring and associated wiring errors. They took up less space than the counters, timers and other control components they replaced. And their ability to be reprogrammed dramatically increased flexibility when changing control schemes.

Perhaps the biggest key to industry's acceptance of the PLC was that the initial programming language was based on the ladder diagrams and electrical symbols commonly used by electricians (see Fig. 1-1). Most plant personnel were already

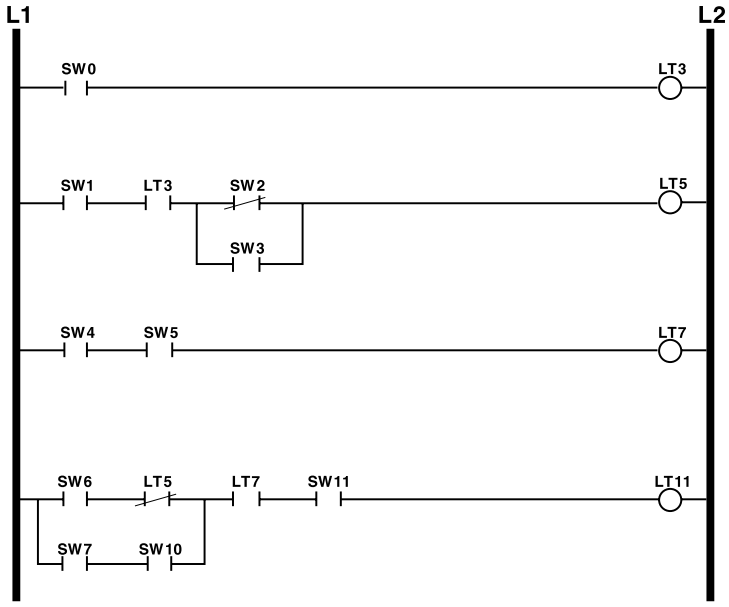


Fig. 1-1 Typical electrical ladder diagram

1

trained in ladder logic, and they easily adopted it for PLCs. In fact, ladder logic still plays an integral role in programming and troubleshooting, even though more "advanced" programming languages have been developed.

1.1

Why Use a PLC?

"Should we be using a programmable logic controller?" During the 1970s and early '80s, many engineers, manufacturing managers and control system designers spent considerable time debating this issue, trying to evaluate cost effectiveness.

Today, one generally accepted rule is that PLCs become economically viable in control systems that require three to four or more relays. Given that micro PLCs cost only a few hundred dollars, coupled with the emphasis manufacturers place on productivity and quality, the cost debate becomes almost immaterial.

In addition to cost savings, PLCs provide many value-added benefits:

- *Reliability*. Once a program has been written and debugged, it can be easily transferred and downloaded to other PLCs. This reduces programming time, minimizes debugging, and increases reliability. With all the logic existing in the PLC's memory, there's no chance of making a logic wiring error. The only wiring required is for power and inputs and outputs.
- *Flexibility*. Program modifications can be made with just a few key strokes. OEMs (original equipment manufacturers) can easily

implement system updates by sending out a new program instead of a service person. End-users can modify the program in the field, or, conversely, OEMs can prevent end-users from tinkering with the program (an important security feature).

- *Advanced functions*. PLCs can perform a wide variety of control tasks, from a single, repetitive action to complex data manipulation. Standardizing on PLCs opens many doors for designers, and simplifies the job for maintenance personnel.
- *Communications*. Communicating with operator interfaces, other PLCs or computers facilitates data collection and information exchange.
- *Speed*. Because some automated machines process thousands of items per minute — and objects spend only a fraction of a second in front of a sensor — many automation applications require the PLC's quick response capability.
- *Diagnostics*. The troubleshooting capability of programming devices and the diagnostics resident in the PLC allow users to easily trace and correct software and hardware problems.

1.2

Traditional PLC Applications

No matter what the application, the use of PLCs helps increase competitiveness. Processes using PLCs include: packaging, bottling and canning, material handling, machining, power generation, HVAC/building control systems, security systems, automated assembly, paint lines, and water treatment. PLCs are applied in a variety of

1

industries, including food and beverage, automotive, chemical, plastics, pulp and paper, pharmaceuticals, and metals. Virtually any application that requires electrical control can use a PLC.

Traditional PLC applications

Wastewater treatment facility

Papermaking operation

CHAPTER 2

The Micro PLC

Development of the Micro PLC 2.0
What Makes a Micro PLC a Micro? 2.1
Capabilities Overview 2.2
Micro PLC Applications 2.3

2

2.0

Development of the Micro PLC

Until the introduction of the micro PLC in the mid 1980s, the potential to increase automation on simple machines or less complex processes remained largely untapped. This was due to the lack of attractive alternatives to hardwired relay control.

Though OEMs had benefitted by using PLCs to control equipment, process lines, or even whole plants, they could not always justify using a PLC on small applications and low-cost machines. And if cost was not an issue, size often was. Sometimes even small PLCs were simply too large to fit in the space allocated for electrical controls.

As such, the driving force behind the development of the micro PLC was the demand by OEMs for a PLC that was small and inexpensive enough to replace relays, dedicated timers and counters, and single board controllers. For a $5,000 machine, a small PLC control system costing $1,000 is not economical. However, at a few hundred dollars, a micro PLC is cost effective and provides all the benefits of traditional PLC logic control.

PLCs have followed a product development curve similar to that of the personal computer; early PLCs were large, cost thousands of

Shown near actual size, a 16 I/O micro PLC. Hundreds of electromagnetic relays would be needed to obtain an equivalent level of control.

dollars, and had relatively few features. But with the evolution of microprocessors and other board-level components, PLCs grew in sophistication while size and cost shrank. In fact, advanced features that were considered strictly in the domain of medium-size PLCs five years ago are now common in micro PLCs.

Typical Micro PLC Features

- Math capabilities
- Data handling instructions

2

- High-speed counting
- BCD to binary conversion routines
- Drum timer and sequencer functionality
- Subroutines and interrupts
- Programmed with a personal computer
- Communication with other electronic devices

2.1

What Makes a Micro PLC a Micro?

Several criteria are used to categorize PLCs as micro, small, medium or large. Criteria include functionality, number of inputs and outputs (see Fig. 2-1), cost, and physical size.

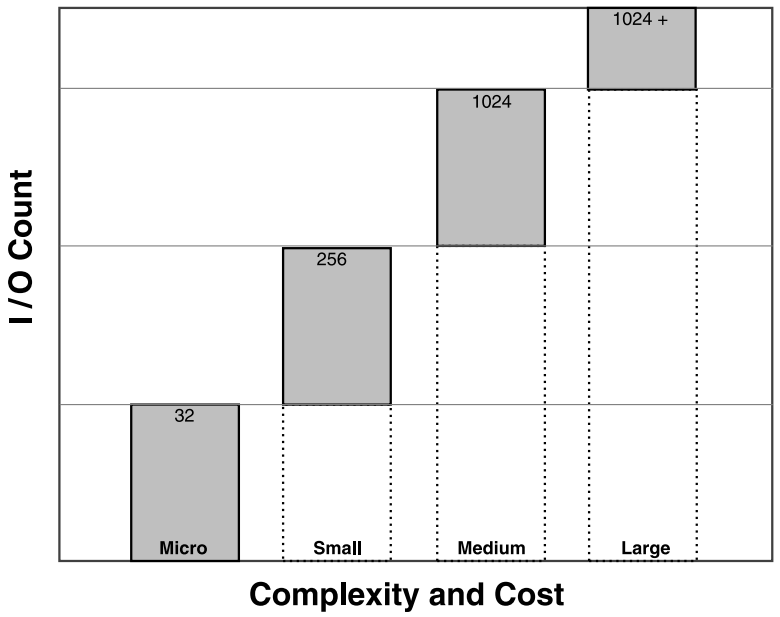


Fig. 2-1 I/O count is the most common method of categorizing PLCs.

Generally speaking, micro PLCs share the following characteristics:

- ≤32 I/O
- Cost <$500
- 1K of memory
- Small size, roughly:
  - 5" (127 mm) long x 3" (76 mm) high x 3" (76 mm) deep (16 I/O)
  - 8" (203 mm) long x 3" (76 mm) high x 3" (76 mm) deep (32 I/O)

Micro PLCs come as self-contained units with the processor, power supply, and I/O all in one package. Because they are self-contained, micro PLCs are also known as packaged controllers. A modular PLC is one that has separate components that interconnect. The advantage of

A 32 I/O packaged micro controller is considerably more compact than a 32 I/O modular controller.

2

- **Relay logic instructions**
  - examine if closed (normally open contacts)
  - examine if open (normally closed contacts)
  - output energize (coils)
  - output latch
  - output unlatch
  - one-shot rising
- **Timers**
  - on-delay timer
  - off-delay timer
  - retentive timer
- **Up and down counters**
- **High-speed counter**
- **Math**
  - add
  - subtract
  - divide
  - multiply
  - clear
  - square root
- **Boolean logic**
  - AND, OR, Exclusive OR, NOT, Negate
- **Comparison**
  - =, ≠, <, ≤, >,≥
  - limit
- **Data handling**
  - move, masked move
  - FIFO and LIFO (First-In First-Out; Last-In First-Out)
  - BCD to binary conversion
  - binary to BCD conversion
- **Application specific instructions**
  - sequencer
  - bit shift
- **Program flow**
  - subroutine
  - MCR (master control reset)
  - immediate input or output with mask
  - selectable timed interrupt
  - jumps

Fig. 2-2 The instruction set of a typical micro PLC.

a packaged controller is that the all-in-one package is smaller, less costly, and convenient to install (see photo-p.11). However, few packaged controllers have expandable I/O capabilities, where all modular controllers can be expanded easily by adding more I/O cards to the rack.

2.2

Capabilities Overview

A PLC's capabilities are determined by the type of commands a user can instruct it to execute. While the instruction set and names of instructions will vary slightly among micro PLC manufacturers, Fig. 2-2 gives an overview of the instructions commonly available.

As has been noted, PLCs were initially designed to function as electronic replacements for hardwired control devices — primarily relay coils and contacts, counters and timers. Today, these functions still comprise the majority of instructions used in micro PLC applications.

By way of example, imagine designing a control system for a conveyor in a food packaging operation. Based on the status of field devices, a PLC can start a conveyor, sense the presence of a box, move the box forward to the desired position, hold it there for a predetermined filling time, and count the number of full boxes coming off the line.

Micro PLCs also open up new control possibilities with advanced functions such as: four-function math, data comparison (i.e., equal to, greater than, etc.), data handling (such as parts sorting or fault tracking), sophisticated subroutines, sequencing (replacing drum sequencers),

and other features that experienced control system designers can appreciate. To demonstrate the value of these features, application examples are provided in Chapter 7.

Possibly the most exciting feature of micro PLCs is their high-speed counting capabilities. Speed, the key to success for many automated applications, can also cause problems if the speed of the PLC cannot keep up with the manufacturing operation. For example, if parts or material are moving at high speed past a proximity sensor, a normal PLC counter could "miss" some parts. This is because the parts are moving faster than the PLC scans the sensor's input.

However, a high-speed counter operates independently of the program scan. This enables it to count at a much faster rate, typically 2,000 to over 6,000 times per second. In addition, some high-speed counters can energize an output immediately (i.e., without having to wait for the normal program scan time), thus substantially improving speed and performance. This enables the counter to affect control operations when split-second accuracy is critical.

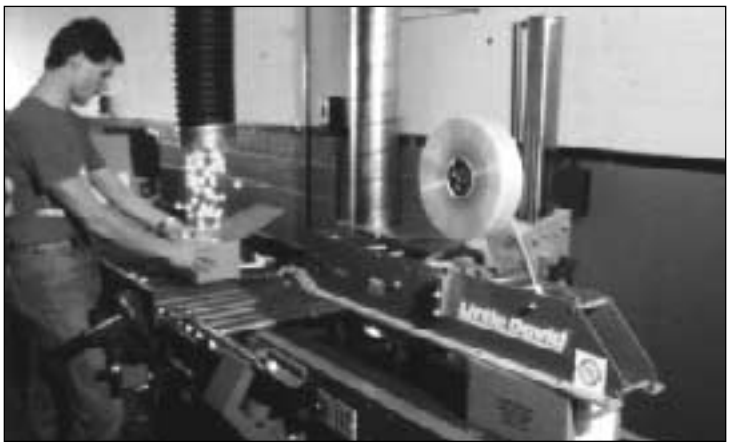
2.3

Micro PLC Applications

Micro PLCs are ideal for controlling stand-alone, discrete machinery or processes. Many applications that are presently controlled by relays and/or custom single board controls are migrating toward micro PLCs. Micro PLC applications are considered in detail in Chapter 5 (How to Apply a Micro PLC) and Chapter 7 (Application Examples).

Canning operation

Typical micro PLC applications



Packaging machine

CHAPTER 3

Micro PLC Operation

Components Overview 3.0
Inputs 3.1
Outputs 3.2
Central Processing Unit – CPU. 3.3
Types of Application Memory. 3.4
Data, Memory and Addressing 3.5
Operating Cycle 3.6
Power Supplies. 3.7
Programming Devices. 3.8
Operator Interfaces 3.9

3

3.0

Components Overview

In order to learn how PLCs operate, a quick overview of PLC components is necessary. All PLCs — from micro to very large — use the same basic components and are structured in a similar fashion. PLC systems consist of:

- Inputs
- Outputs
- Central processing unit (CPU)
- Memory, for program and data storage
- Power supply
- Programming device
- Operator interfaces

3.1

Inputs

The input screw terminals on a PLC form the interface by which field devices are connected to the PLC.

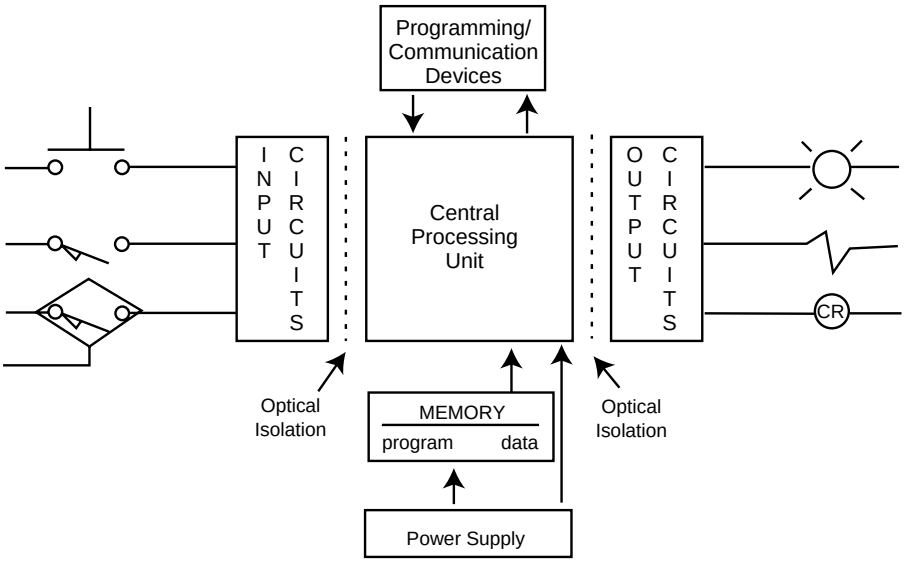


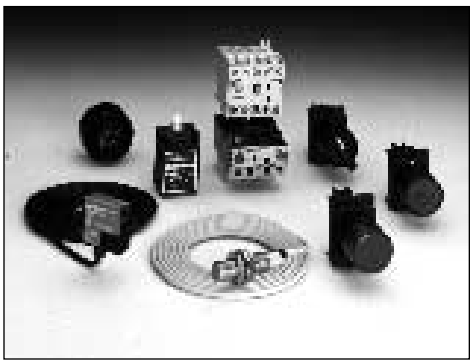
Fig. 3-1 Control of a machine or process entails monitoring the status of devices connected as inputs and, based on a user-written program, controlling devices connected as outputs.

Inputs include items such as pushbuttons, thumbwheel switches, limit switches, selector switches, proximity sensors and photoelectric sensors. These are all discrete devices that provide an On or Off status to the PLC. While larger PLCs can directly accept analog values (variable voltage or current signals) such as from temperature or pressure sensors, micro PLCs do not typically possess this capability.

The electrical signals that field devices send to the PLC are typically unfiltered 120V ac or 24V dc. The input circuitry on the PLC takes this field voltage and “conditions” it to be usable by the PLC. Conditioning is necessary because the internal components of a PLC operate on 5V dc, and this minimizes the possibility of damage by shielding them from voltage spikes. To electrically isolate the internal components from the input terminals, PLCs employ an optical isolator, which uses

light to couple signals from one electrical device to another.

The PLC's input circuitry also "filters" field voltage signals to qualify them as valid, such as a signal from a sensor, or not valid, such as high-frequency electrical "noise" or static. Input filters determine the validity of a signal by its duration; they "wait" to confirm that a signal is a reference from an input device rather than electrical noise. A typical filter time is 8 ms, but some PLCs have adjustable input filter response times. A longer response time provides better filtering of electrical noise. A shorter response time helps in applications that require high-speed operation (e.g., interrupts or counting).



Assorted I/O devices. See Appendix B for a more thorough description of I/O devices.

3.2

Outputs

Connected to the output terminals of the PLC are devices such as solenoids, relays, contactors, motor starters, indicator lights, valves and alarms. Output circuits operate in a manner similar to input circuits: signals from the CPU pass through an isolation barrier before energizing output circuits.

Transistor, relay and triac for PLC output circuitry. Paper clip indicates relative size.

PLCs use a variety of output circuits to energize their output terminals: relays, transistors and triacs.

- Relays are for either ac or dc power. Traditional PLC electromagnetic relays typically handle current up to a few amps. Relays can better withstand voltage spikes, and they have an air gap between their contacts which eliminates the possibility of current leakage. However, they are comparatively slow and subject to wear over time.
- Transistors switch dc power, are silent and have no moving parts to wear out. Transistors are fast and can reduce response time, but only carry loads of 0.5 amp or less. Special types of transistors, such as FETs (Field Effect Transistors) can handle more power, typically up to 1 amp.
- Triacs strictly switch ac power. Like transistors, triac outputs are silent, have no moving parts to wear, are fast, and carry loads of 0.5 amp or less.

Note: solid state outputs (triacs and transistors) can be damaged or destroyed by over-voltage or over-current.

3.3

Central Processing Unit — CPU

The CPU, made up of a microprocessor and a memory system, forms the primary component of the PLC. The CPU reads the inputs, executes logic as dictated by the application program, performs calculations, and controls the outputs accordingly.

PLC users work with two areas of the CPU: program files and data files. Program files store a user's application program, subroutine files,

and the error file. Data files store data associated with the program, such as I/O status, counter/timer preset and accumulated values, and other stored constants or variables. Together, these two areas are called the application memory or user memory.

Also within the CPU is an executive program or system memory that directs and performs "operation" activities such as executing the user program and coordinating input scans and output updates. System memory, which is programmed by the manufacturer, cannot be accessed by the user.

3.4

Types of Application Memory

As the name indicates, programmable logic controllers have programmable memory that allows users to develop and modify control programs. Memory is a physical space inside the CPU where the program files and data files are stored and manipulated.

Memory types fall into two categories: volatile or nonvolatile. Volatile memory can

Even though EEPROM and RAM memory can save application programs if power is lost, they do not necessarily save process data, such as the accumulated value of a timer or counter. If retaining process data is important for an application, look for a micro PLC that offers 100 percent data retention. Upon power loss, this type of PLC automatically saves process data to the nonvolatile EEPROM.

be easily altered or erased, and it can be written to and read from. However, without proper backup, a power loss can cause the loss of programmed contents.

The best known form of volatile memory is Random Access Memory, or RAM. RAM is relatively fast and offers an easy means to create and store users' application programs. If normal power is disrupted, micro PLCs with RAM memory use battery or capacitor backups to prevent program loss. (However, note that capacitors and batteries may fail.)

Nonvolatile memory retains its programmed contents — without a battery or capacitor backup — even if power is lost. The EEPROM — Electrically Erasable Programmable Read Only Memory — is a nonvolatile memory that has the same flexibility as RAM, and is programmed through application software, which runs on a personal computer or through a micro PLC's Hand-Held Programmer.

3.5

Data, Memory and Addressing

Whereas memory is a physical space, data is information stored in that space. The CPU operates just like a computer; it manipulates data using binary digits, or bits. A bit is a discrete location within a silicon chip that either has a voltage present, read as a value of 1 (On), or not present, read as a value of 0 (Off). Thus, data is a pattern of electrical charges that represent a numerical value.

A bit is the smallest unit of memory available. Generally, CPUs process and store data in 16 bit groups, also known as “words.” However, users can still manipulate data on the bit level.

Each word of data has a specific, physical location in the CPU called an “address” or a “register” (note that the terms “word,” “address,” and “register” are often used interchangeably). Every element in the user program is referenced with an address to indicate where data for that element is located. When assigning addresses to I/O in a program, note that the address is related to the terminal where input and output devices are connected (see Fig. 3-2).

Decimal	Hexadecimal	Binary	BCD	Octal	Gray code
0	0	000	0000	0	0000
1	1	001	0001	1	0001
2	2	010	0010	2	0011
3	3	011	0011	3	0010
4	4	100	0100	4	0110
5	5	101	0101	5	0111
6	6	110	0110	6	0101
7	7	111	0111	7	0100
8	8	1000	1000	10	1100
9	9	1001	1001	11	1101
10	A	1010		12	1111
11	B	1011		13	1110
12	C	1100		14	1010
13	D	1101		15	1011
14	E	1110		16	1001
15	F	1111		17	1000

Fig. 3-3

While PLCs operate in binary (1 and 0), they also use binary to convert, accept and manipulate data from other number systems. These systems include binary coded decimal (BCD), hexadecimal, octal, and gray code (see Fig. 3-3).

Beginning PLC users probably do not need to know how to use these different number systems, so they will not be explained further. However, note that they may need to be learned later, as these numbering systems are valuable when working with certain types of inputs. For example, thumbwheel switches usually require four bits per wheel; i.e., they communicate in BCD. Thus, any PLC used with a thumbwheel must be able to accept a BCD input.

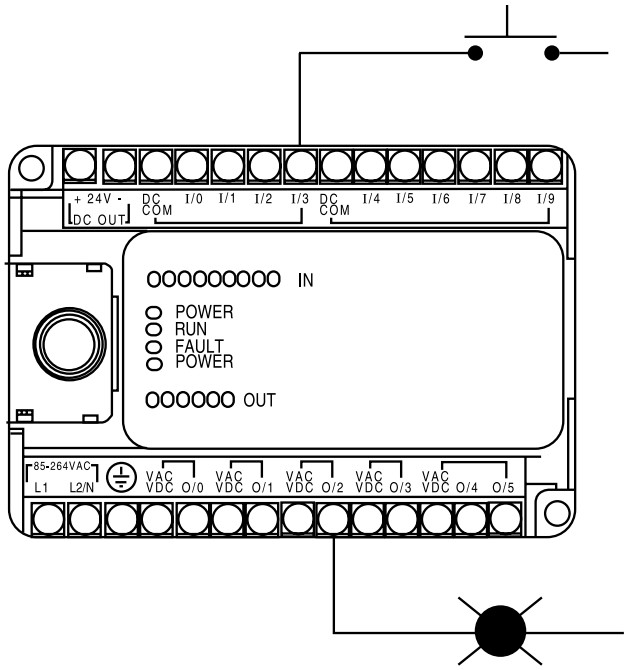
3.6

Operating Cycle

All the components of the PLC system come into play during the operating cycle, which consists of a series of operations performed sequentially and repeatedly.

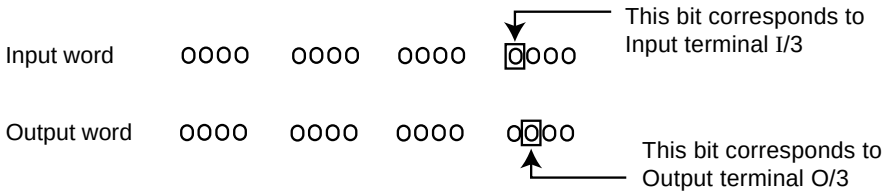
I/O Wiring

Pushbutton wired
to input terminal I/3



Pilot light wired
to output terminal O/2

Memory location of I/O status



Program with addresses

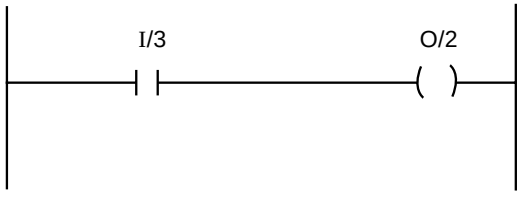


Figure 3-2 This figure shows the relationship between the actual I/O wiring terminal location and the address of the instructions in the program. Note: The I/O address format may differ, depending on the PLC manufacturer.

The major elements of an operating cycle are:

1. *The input scan*. During the input scan the PLC examines the external input devices for a voltage present or absent; i.e., an "On" or "Off" state. The status of the inputs is temporarily stored in an "input image" memory file.
2. *Program scan*. During the program scan, the PLC scans the instructions in the ladder logic program, uses the input status from the input image file, and determines if an output will or will not be energized. The resulting status of the outputs is written to the "output image" memory file.
3. *Output scan*. Based on the data in the output image file, the PLC energizes or de-energizes its output circuits, controlling external devices.

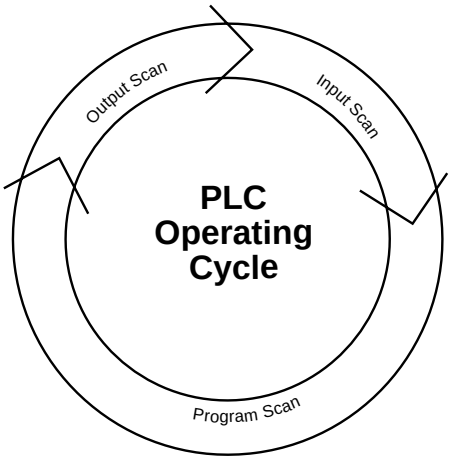


Fig. 3-4 Image of operating cycle.

3.7

Power Supplies

The power supply provides power to the controller's internal electronics, converts the incoming voltage to a usable form and protects the PLC's components from voltage spikes.

Speed

What is the fastest action required in the control process? How much time is needed to control that action? Speed is one of the primary advantages of today's micro controllers. Operating cycles typically take 1 to 25 milliseconds (thousandths of a second). When judging the speed, it is important to look at total throughput time, not just the operating cycle.

Components of throughput time include: time for actuation of the physical input; time for PLC's input circuit to sense the signal; time for input scan, program scan and output scan; time for actuation of the output circuit and corresponding field device; and time for the CPU's "housekeeping" or "overhead" functions. See throughput time worksheet in Appendix D.

For applications that require high-speed operation, advanced micro controllers offer functions such as high-speed counting with direct control of outputs and immediate I/O update instructions. These functions enable the micro controller to detect and react quickly to changing input conditions.

Given that most facilities experience line voltage fluctuations, PLC power supplies are designed to maintain normal operation even if the voltage varies from 10 to 15 percent. Dips or surges in power are caused by natural line losses from the utility, brownouts or the start-up or shutdown of nearby heavy equipment (such as motors or arc welders). For voltage conditions that are especially unstable, consider installing a constant voltage transformer between the PLC and the primary power source.

The PLC's power supply is designed to withstand short power losses without affecting the operation of the system. A PLC can operate for several milliseconds without line power before the power supply signals the processor that it can no longer provide adequate dc power to the system. The power supply then instructs the processor to execute a controlled shut down, which saves the user's program and data in memory.

Another factor affecting the function of the PLC is electromagnetic interference (EMI) or electrical noise. While PLCs are more rugged than most electronic equipment (especially the PCs or single board controllers sometimes

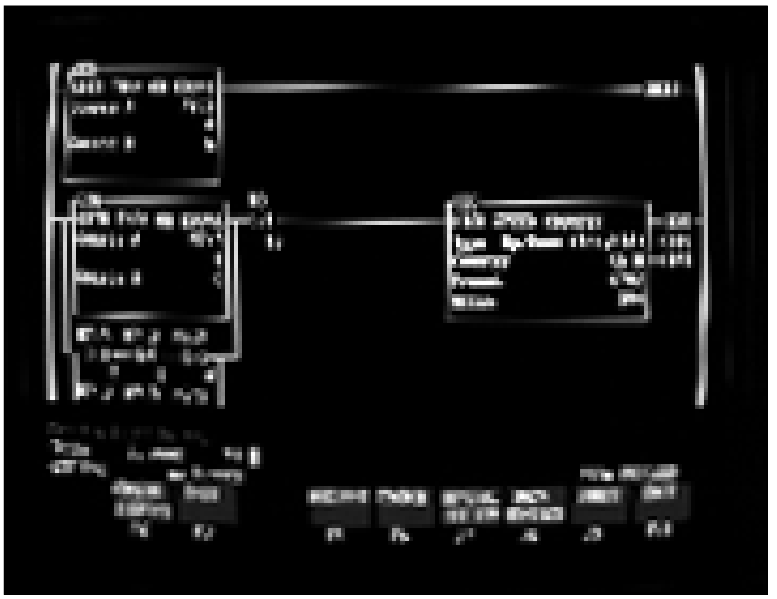
used instead of PLCs), EMI may still be a problem. If so, the PLC should be electrically isolated by installing an isolation transformer.

Until recently, all micro PLCs operated on 24V dc. However, several micro PLC manufacturers now offer products that operate on either 120V ac, 220V ac or 24V dc. This gives the user the option of selecting the voltage that best suits the application. For example, if ac power is used on other parts of the machine (actuators, for example), a micro PLC that can accept ac power may eliminate the need to install a dc power supply.

3.8

Programming Devices

When entering a program into a micro PLC, the two devices most commonly used are a personal computer (PC) and a Hand-Held



Most users create their programs with software run on a PC.

Plant technicians value Hand-Held Programmers (shown actual size) because of their portability, ruggedness and troubleshooting capabilities.

Programmer (HHP).

The PC is used to run PLC programming software. This software allows users to create, edit, document, store and troubleshoot ladder diagrams, and generate printed reports. Software instructions are based on graphical symbols for various functions. Using such software does not require knowledge of higher programming languages, just a general understanding of standard electrical wiring diagrams.

While the HHP can be used to program the PLC, it is more commonly used as a troubleshooting tool. This is because the HHP is compact and has its own memory to store programs. HHPs are invaluable for troubleshooting equipment while on the factory floor, for modifying programs, and transferring programs to multiple machines. The language used by the HHP is a graphical form of instruction list programming based on the PLC's ladder logic instructions.

Operator interface

3.9

Operator Interfaces

In order to convey information about

machine status, the front panel of a micro PLC has a series of indicator lights. These are for such things as power, run, faults or I/O status. To communicate with the PLC — to enter data or monitor and control machine status — traditional operator interfaces include pushbuttons, thumbwheel switches, pilot lights and LED numeric displays.

To improve the interface between the operator and the micro PLC, a new generation of electronic operator interface devices (or peripherals) can be connected. These are not programming devices, but graphic or alphanumeric displays and control panels that consolidate all the functions of traditional operator interface devices into a single panel.

These interfaces can output data and display messages about machine status in descriptive text (“Motor 1 On”), display parts count, and track alarms. They can also be used for data input. By providing better and more easily conveyed information, these interfaces decrease the need for operator training on machine operation and reduce system, component, and installation costs.

These products communicate with the PLC through an RS 232 communications port. This opens up I/O points, which can be used for sensors and output devices and enables a micro PLC to control a more complex machine or process.

CHAPTER 4

Ladder Logic Fundamentals

Programming Languages 4.0
Electrical Ladder Diagrams 4.1
Ladder Logic Programs 4.2
Ladder Logic Instructions 4.3
Combining Instructions 4.4
Program Execution. 4.5

4

4.0

Programming Languages

A *program* is a user-developed series of instructions or commands that direct the PLC to execute actions. A *programming language* provides rules for combining the instructions so that they produce the desired actions.

The most commonly used language for programming PLCs is ladder logic. In fact, more PLC programs are written in ladder logic than any other language. The ladder logic programming language is an adaptation of an electrical relay wiring diagram, also known as a ladder diagram. Because ladder logic is a graphical system of symbols and terms, even those not familiar with electrical relay wiring diagrams can easily learn it.

Other control languages occasionally used to program PLCs include BASIC, C and Boolean. These computer languages facilitate programs that require complex instructions and calculations too cumbersome to implement with a ladder logic program. However, micro PLCs that can be programmed with BASIC and C are not widely available.

The instructions used to program most micro PLCs are based on a combination of Boolean, ladder logic and mnemonic expressions. A

mnemonic expression is a simple and easy to remember term which represents a complex or lengthy instruction. For example, “TON” stands for “timer on.” Different PLCs use slightly different instructions, and these can be found by consulting the user's manual.

4.1

Electrical Ladder Diagrams

Ladder logic programs evolved from electrical ladder diagrams, which represent how electric current flows through devices to complete an electric circuit. These diagrams show the interconnection between electrical devices in an easy-to-read graphical format that guides the electrician when wiring (see Fig. 4-1).

An electrical diagram consists of two vertical bus lines, or power lines,

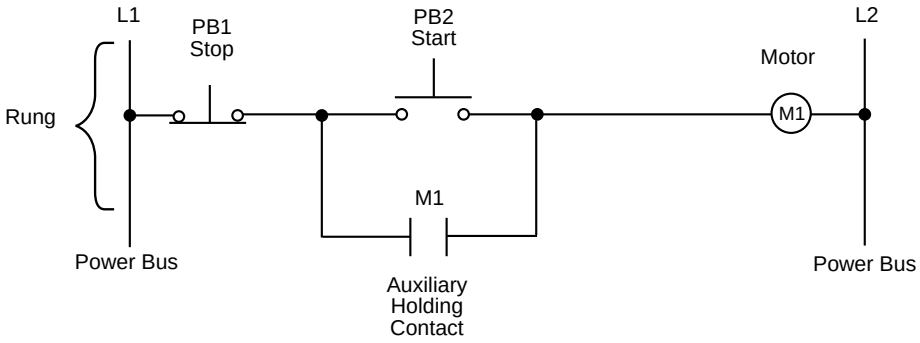


Fig. 4-1 Electrical diagram of a hardwired start/stop circuit.

with current flowing from the left bus to the right bus. Each electrical circuit in the diagram is considered a rung. Every rung has two key components: it contains at least one device that is controlled, and it

contains the condition(s) that control the device, such as power from the bus or a contact from a field device.

A rung is said to have electrical continuity when current flows uninterrupted from left to right across the rung (i.e., all contacts are closed). If continuity exists, then the circuit is complete and the device controlled by the rung turns On (see Fig. 4-2). If continuity does not exist, the device stays Off.

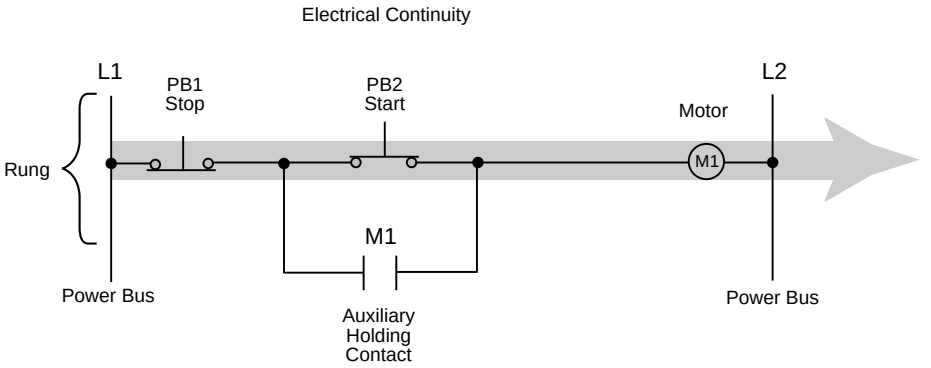


Fig. 4-2 If PB1 is NOT pushed and PB2 is pushed, the circuit will be complete. Under these conditions, the rung has electrical continuity and the motor will turn On.

4.2

Ladder Logic Programs

A PLC ladder logic program closely resembles an electrical ladder diagram (Fig. 4-3). On an electrical diagram, the symbols represent real-world devices and how they are wired. A PLC program uses similar symbols, but they represent ladder logic instructions for the application. A ladder logic program exists only in the PLC's software — it is not the actual power bus or the flow of current through circuits. Another

difference is that in an electrical diagram, devices are described as being open or closed (Off or On). In a ladder logic program, instructions are either True or False (however, the terms are often used interchangeably).

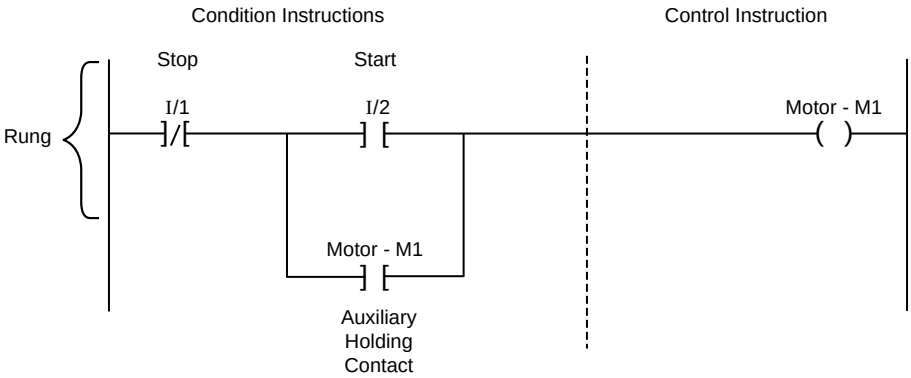


Fig. 4-3 Notice the similarity between the ladder logic program and the hardwired circuit in Fig. 4-1.

Each rung in a ladder logic program *must* contain at least one control instruction (output) and usually contains one or more condition instructions (inputs). Condition instructions are programmed to the left of the control instruction. Examples of condition instructions include signals from connected input devices, contacts associated with outputs, and signals from timers and counters.

Programmed on the right side of the rung, a control instruction is the operation or function that is activated/de-activated by the logic of the rung. Examples of control instructions include output energize (turn On the PLC's output circuitry to activate a field device) and instructions internal to the PLC, such as bit commands, timers, counters and math commands.

4

The control instructions are energized or de-energized based on the status of the condition instructions in the rung. The PLC does this by examining a rung for logical continuity (i.e., all condition instructions are evaluated as True). If logical continuity exists, the PLC energizes the control instruction (see Fig. 4-4). If logical continuity does not exist, then the PLC maintains the control instruction in the Off or de-energized state.

Recall from Chapter 3 that every element in the user program is referenced with an address to indicate where data for that element is located.

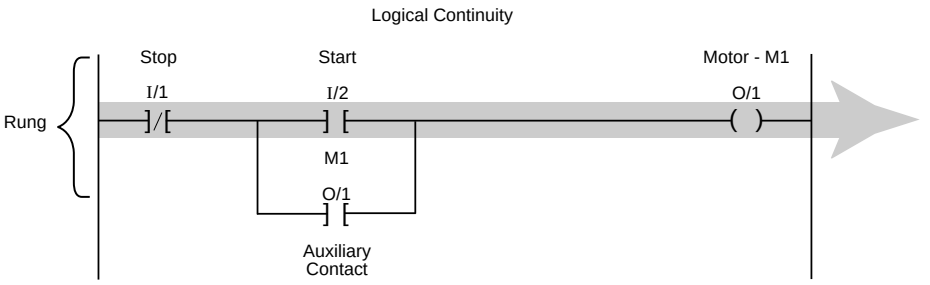


Fig. 4-4 If a signal is NOT present at input terminal I/1 and a signal is present at input terminal I/2, the rung has logical continuity and the PLC will energize output terminal O/1 controlling the motor.

4.3

Ladder Logic Instructions

The most frequently used instructions in a PLC ladder logic program are the normally open (N.O.) instruction, the normally closed (N.C.) instruction, and the output energize instruction (see Fig. 4-5). These instructions are represented as symbols placed on the rungs of the program (which is why PLC users may hear ladder logic described as "contact symbology").

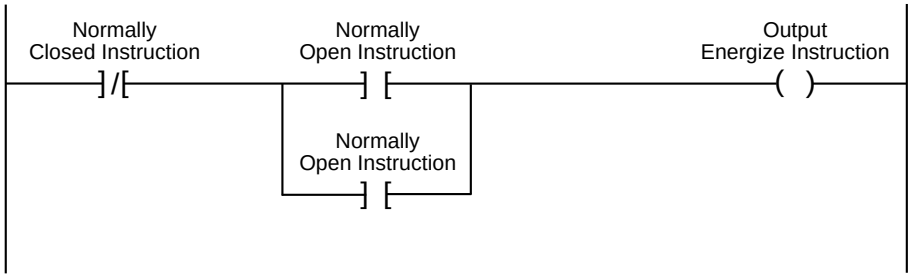


Fig. 4-5 Common ladder symbols.

Normally Open Instruction ⊣ ⊢

A normally open instruction examines a PLC memory location for an On condition (i.e., it checks to see if the bit element at the instruction's address is On (binary 1)). If the PLC detects an On condition, the instruction is True and has logical continuity.

For example, a N.O. pushbutton (PB1) is wired to input terminal I/3 on the PLC. The ladder logic program contains the following rung (Fig. 4-6.1), where I/3 is programmed as a N.O. instruction.

When PB1 is pressed (On), that On status is written to input image

memory location I/3 during the PLC's input scan. When the rung containing the N.O. instruction with address I/3 is scanned, that instruction is seen as True and the PLC energizes output O/4 during its output scan.

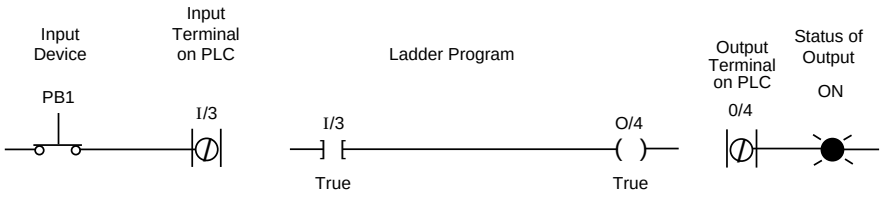


Fig. 4-6.1 Normally open instructions.

When PB1 is released, the Off status is written to address I/3. The N.O. instruction is now False and the rung lacks logical continuity (4-6.2). During the PLC's output scan, output O/4 will be de-energized.

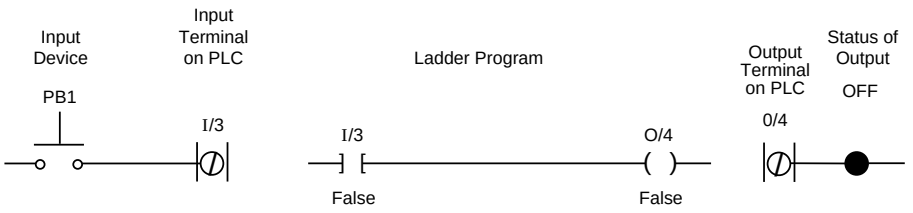


Fig. 4-6.2 Normally open instructions.

Normally Closed Instruction ⊣/⊢

A normally closed instruction examines the PLC memory for an Off condition (i.e., it checks to see if the bit element at the instruction's address is Off, or 0). If the PLC detects an Off condition, the instruction is True and has logical continuity.

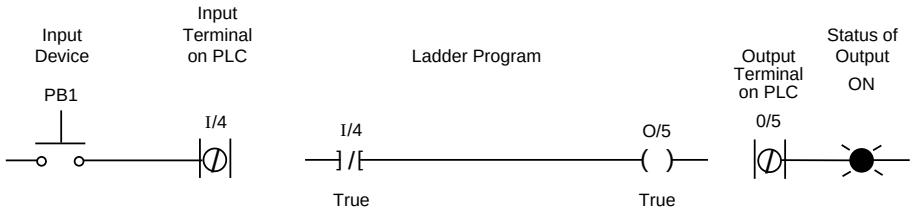


Fig. 4-7.1 Normally closed instructions.

For example, a N.O. pushbutton (PB1) is wired to input terminal I/4 on the PLC. The ladder logic program contains the following rung (Fig. 4-7.1), where I/4 is programmed as a N.C. instruction.

When PB1 is not pressed (Off), that Off status is written to input image memory location I/0 during the PLC's input scan. When the rung containing the N.C. instruction with address I/0 is scanned, that instruction is seen as True (NOT On) and the PLC energizes output O/5 during the output scan.

When PB1 is pressed, the On status is written to address I/4. The N.C. instruction is now False and the rung lacks logical continuity (Fig. 4-7.2). During the PLC's output scan, output O/5 will be de-energized.

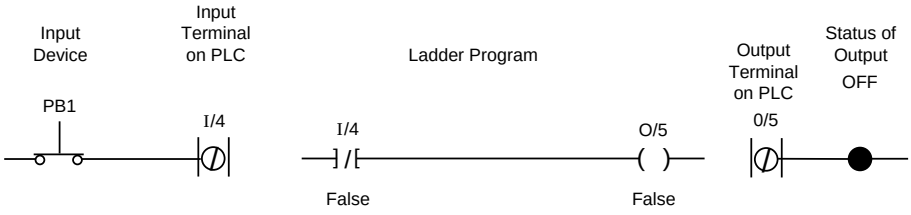


Fig. 4-7.2 Normally closed instructions.

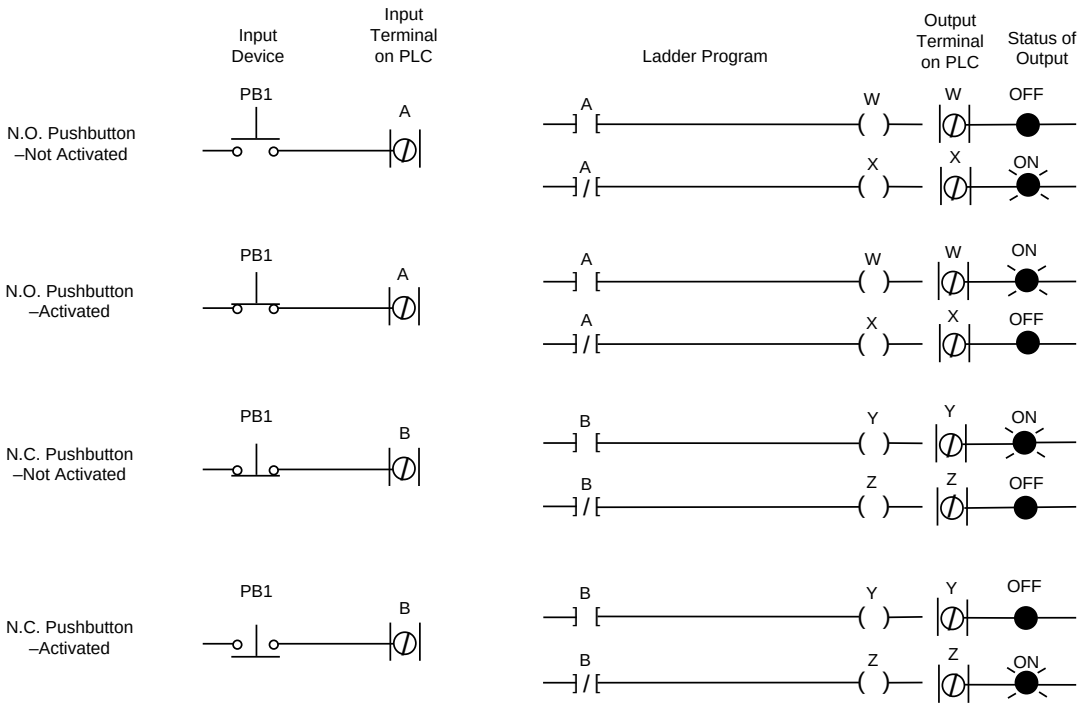


Fig. 4-8 Condition instructions and their results.

Output Energize Instruction —( )—

Controlled by the condition instructions that precede it on a rung, the output energize instruction (OTE) turns On a bit element in the output image file when rung conditions are True. Output energize is the ladder logic equivalent of a relay coil on an electrical diagram.

When logical continuity exists on a rung, the On condition (binary 1) is written to the location in memory associated with the output energize instruction. If the address is that of an external output device, the PLC energizes the output during the output scan. When the rung is False, the PLC de-energizes the output. The output energize instruction

Hardwired to Programmed

Remember to make a distinction between the physical input device and its ladder logic representation, and note that an instruction in a ladder program is programmed independently of how the input device is wired. Therefore, the status of a N.O. pushbutton can be tested with a N.C. instruction, and vice versa. Fig. 4-8 demonstrates all the possible combinations and their results. Also remember that when PLC instructions change state (e.g., make a False-to-True transition), a normally open instruction does not change to a normally closed instruction. Where electromechanical relay contacts open and close, PLC instructions test a memory location for a 1 or 0.

controls real world devices (solenoid valves, motors, lights, etc.) or internal bit elements.

Higher Level Instructions

While relay logic is suitable for simple On/Off sensing and control, many applications require more powerful instructions. To allow this, enhanced ladder language commands have been developed. These instructions deal with numerical data beyond simple 1s or 0s by manipulating data in bytes or words. Examples of higher level instructions include counters, timers, sequencers, math, comparison and other operations that N.O., N.C. and OTE instructions cannot perform.

To keep the implementation of these operations simple, higher level instructions are usually represented in ladder logic programming as function blocks. As shown

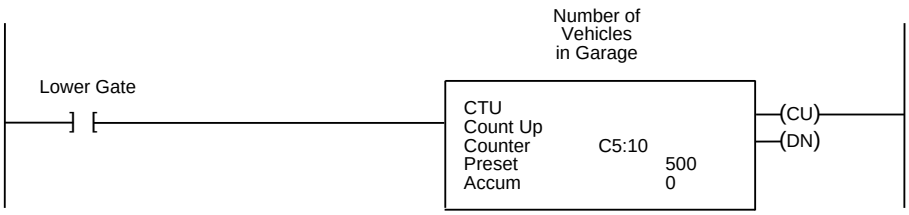


Fig. 4-9 Higher level instructions – such as this counter – are represented with "function blocks" in the ladder program.

in Fig. 4-9, function blocks are literally programmed as blocks on the rung of a ladder program. Depending on their operation, higher level instructions can be either condition instructions (e.g., comparison instructions) or control instructions (e.g., timer or counter instructions).

4.4

Combining Instructions

Two fundamental logic operations — AND and OR — provide the rules for governing how instructions are combined.

AND Logic

Condition instructions programmed in series are the ladder diagram equivalent of AND logic (Fig. 4-10). For example, picture a metal stamping operation where the machine activates only if the operator simultaneously pushes both a left-hand start button (X) AND a right-hand start button (Y).

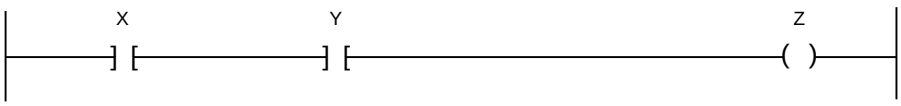


Fig. 4-10 With instructions programmed in series, output Z will be True (On) only if both input X AND input Y are True (On).

The output of an AND equation will be True only if *all* conditions in series are True. If any condition is False, then the rung does not have logical continuity and the output will be Off.

OR Logic

Condition instructions programmed in parallel are the ladder diagram equivalent of the OR operation (see Fig. 4-11). For example, imagine a conveyor that has two run switches, one located at each end. The conveyor could be configured to start if an operator pressed a start button at one end (X) OR the other (Y).

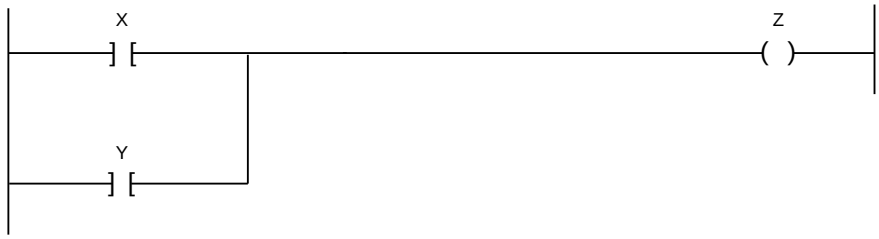
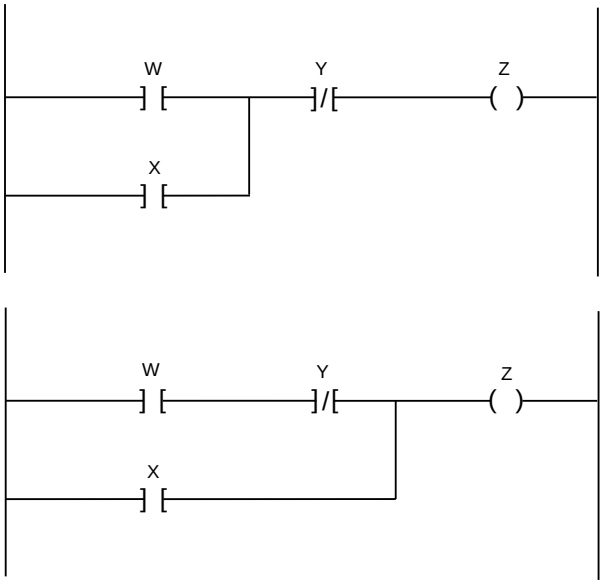


Fig. 4-11 With instructions programmed in parallel, output Z will be True (On) if either X OR Y are True (On).

The output of an OR equation will be True if any condition in parallel is True. If all conditions are False, then the rung does not have logical continuity and the output will be False.



Notice that AND and OR logic (series and parallel circuits) can be combined on a single rung, as shown in Fig. 4-12.

Fig. 4-12 Combining series and parallel logic.

Branch Operations

The function of a branch is to allow both condition and control instructions to be programmed in parallel in a single rung (Fig. 4-13).

- *Condition* instructions programmed in parallel are the equivalent of an OR operation.
- *Control* instructions programmed in parallel are the equivalent of an AND operation.

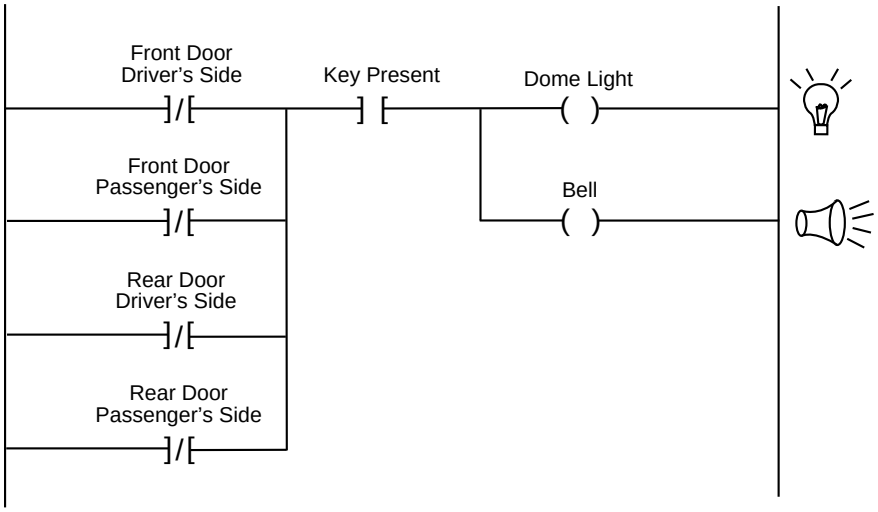


Fig. 4-13 In this example, branch instructions are applied to a program controlling the dome light and "door ajar" bell of a 4-door sedan. The light and bell (multiple outputs) will turn On if ANY of the doors (multiple inputs) are opened while a key is present in the ignition.

Branch operations also provide the relay wiring equivalent of an auxiliary holding contact or memory function (refer back to Fig. 4-3). Auxiliary contacts keep their output energized after a momentary start signal is no longer present.

As shown in Fig. 4-14, an auxiliary holding contact is always programmed with the same address as its referenced output (remember, one of the advantages of a PLC is that an address can be used more than once). Momentarily pressing start button I/2 energizes control instruction O/3 (which turns On the motor), and it also energizes condition instruction O/3. Energizing O/3 in the branch operation maintains the On status of the output until stop button I/1 is pressed.

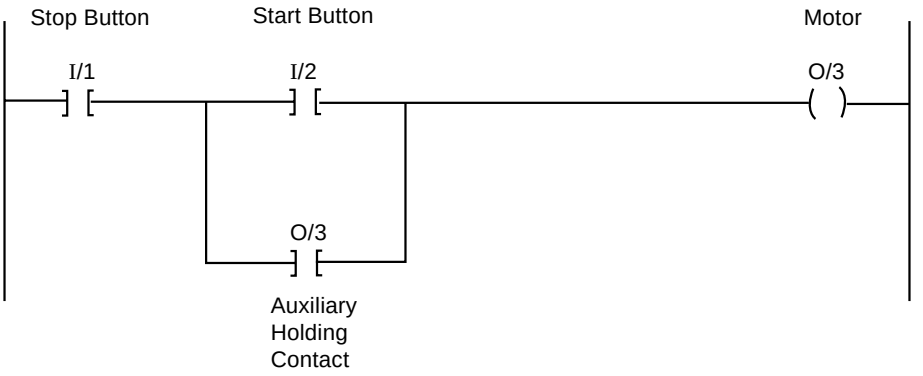


Fig. 4-14 An auxiliary holding contact keeps its referenced output energized even after a momentary start signal has been removed.

4.5

Program Execution

Before reading how the PLC executes a ladder logic program, re-reading Chapter 3.6, “Operating Cycle” may be helpful.

The PLC solves each rung sequentially, from top to bottom of the program. Even if the output of the current rung (e.g., rung 5) affects a previous rung (e.g., rung 2), the PLC does not go back to solve the earlier rung until the next program scan. For the output of one rung to affect an instruction in another rung in the same scan, it must have a lower rung number than the rung it is to affect. That is, the controlling rung must be programmed before the controlled rung.

While rungs are often ordered to show a sequence of events — the top-most rung is the first event and so on — this is done purely for organizational convenience. In both electrical diagrams and ladder logic programs, rung order does not necessarily dictate the sequence of operation. Remember, the *status* of the condition instructions of each rung dictates the sequence in which outputs are controlled.

CHAPTER 5

How to Apply a Micro PLC

What is a Potential Control Application? . 5.0
What are the Application's Requirements?. 5.1
Selecting a Control Method 5.2
What are the PLC Specifications? 5.3
Program Development Procedures. 5.4
Installation Requirements. 5.5

5

5.0

What is a Potential Control Application?

Any situation where coordinated operation of electrical or electronic devices is required is a potential control application. Typical devices controlled include contactors, solenoid valves, relays, lights and motors.

Machines or processes that operate based on any of the following characteristics could be considered potential control applications:

- Repetitive operations
- Time-driven operations
- Event-driven operations
- High-speed control
- Requirements for data acquisition/manipulation

Examples include conveyors, form and fill operations, packaging operations, strapping machines, palletizing and wrapping machines, traffic light sequencing, gate control, cut-to-length lines, semi-automatic welding and painting, storage and retrieval systems, pump alternators, car washes, printing presses, vending machines, and many more.

These applications may be able to be controlled by relays, PLCs, or single board controllers (SBCs) — all of which possess logic capabilities. However, before selecting a control system, the application's requirements must be determined, as they help guide the selection process.

5

Personal computers (PCs) are also sometimes used for control applications, but always for more complex control requirements than the applications controlled by relays, micro PLCs or SBCs.

5.1

What are the Application's Requirements?

No matter what type of control system is ultimately selected, the first step in approaching a control situation is to specify the application's requirements. This includes determining:

- Input and output device requirements.
- The need for special operations in addition to discrete (On/Off) logic, including:
  - Timing
  - Counting
  - High-speed counting
  - Sequencing
  - Data acquisition
  - Data calculations
- The electrical requirements for inputs, outputs, and system power.
- How fast the control system must operate (speed of operation).
- If the application requires sharing data outside the process, i.e., communication.
- If the system needs operator control or interaction.
- The physical environment in which the control system will be located.

To determine application requirements, designers need to begin by identifying all operations the control system needs to perform and the

conditions that affect the system. [Note: If an operation specification exists for the process/machine, consult it before beginning. If no specification exists, it needs to be created at this point.]

As an example, imagine designing a control system for a parking garage with a 500 car capacity. The first step is to define and describe the car parking process. Note that while descriptive text is used here (most people simply write out a description with pen and paper), sequence of operation charts or process sheets are used, too.

What is the desired operation for the parking garage?

- The car approaches an automated ticket machine at a gate.
- The driver pushes a button on the ticket machine to receive a ticket. If there is space left in the garage, the driver will receive a ticket. The machine should not provide a ticket if the garage is full or if the gate is already up.
- Removing the ticket raises the gate and turns on a green "enter" light.
- After the car clears the gate, the gate lowers and the green light shuts off.
- The number of vehicles in the garage needs to be known at any time.
- If maximum capacity is reached, a "Garage Full" sign is illuminated, the ticket machine will not provide a ticket, and the gate will not raise.
- An alarm must sound when the gate is obstructed.

Input and Output Requirements

After defining the operation of the system, the next step is to

determine what input and output devices the system requires. List the function required and identify a specific type of device. Also, group devices by whether they sense an event has occurred or is occurring (inputs) or whether they control something (outputs).

From the description of the parking garage control system, the following I/O requirements can be listed:

Function (inputs)	**Device**
Ticket request	Pushbutton
Ticket taken	Limit switch
Car cleared gate	Photoelectric sensor
Car departed garage	Photoelectric sensor
Gate obstructed	Motor overload contact
Gate in up position	Proximity sensor
Gate in down position	Proximity sensor
Function (outputs)	**Device**
Provide ticket	Solenoid
Garage Full sign	Light
Green light	Light
Alarm	Horn
Raise gate	Gear motor forward
Lower gate	Gear motor reverse

From the list of field devices, the parking garage control system requires seven inputs and six outputs.

Advanced Function Requirements

Applications often require operations beside simple discrete (On/Off) logic. These advanced functions include timing, counting, sequencing, communications, math, comparison, and many other operations involving data manipulation and calculation.

List the advanced functions required and note how they will be used. From the description of the parking garage control system, the following advanced function requirements can be listed:

Function	**Use**
Up counter	Count cars entering garage
Down counter	Count cars leaving garage

Electrical Requirements

When determining the electrical requirements of a system, consider three items: incoming power (power for the control system), input device voltage, and output voltage and current. Because the voltage used with each device may be different, making a distinction is important.

To decide what voltage to use, consider the following:

- What type of power is available (e.g., 24V dc, 120 or 240V ac)?
- How will the machine or process controlled be used?
- Will people come in contact with the machine?
- What power do the field devices use?
- What electrical codes apply?

In the parking garage example, safety is a primary consideration because people physically contact the ticket machine. By using 24V dc power for the input and output devices, hazards to the user dramatically decrease. However, the gate controller selected for the parking garage requires devices capable of switching 120V ac, such as dry contact relays. (Since people do not touch the gate controller in the normal course of operation, it poses a minimal hazard to users.)

While it may be more convenient to use one voltage, application requirements often dictate the need for different voltages. If this is the case, as with the parking garage, isolate the different voltages from each other on separate commons.

Summarizing the electrical requirements for a control system in a chart facilitates organization. For the parking garage example, it looks like this:

Function (inputs)	**Device**	**Voltage**
Ticket request	Pushbutton	24V dc
Ticket taken	Limit switch	24V dc
Car cleared gate	Photoelectric sensor	24V dc
Car departed garage	Photoelectric sensor	24V dc
Gate obstructed	Motor overload contact	24V dc
Gate in up position	Proximity sensor	24V dc
Gate in down position	Proximity sensor	24V dc

Function (outputs)	**Device**	**Voltage**
Ticket provided	Solenoid	24V dc
Full sign	Light	24V dc
Green light	Light	24V dc
Alarm	Horn	24V dc
Gate up	Gate controller	120V ac
Gate down	Gate controller	120V ac

Advanced functions	**Device**	**Voltage**
Up counter	To be determined	TBD
Down counter	To be determined	TBD

Control system	**Voltage**
To be determined	24V dc or 120V ac

Speed of Operation

When determining speed of operation, consider these points:

- How fast does the process occur or machine operate?
- Are there “time critical” operations or events that must be detected?
- In what time frame must the fastest action occur (input device detection to output device activation)?
- Does the control system need to count pulses from an encoder or flow-meter and respond quickly?

The control system selected needs to meet the speed demands of the process or machine, so knowing these criteria is important.

Clearly, the parking garage control system does not require a fast response. Considering that PLC- or SBC-based control systems respond in milliseconds, the relative speed of operation for many applications, such as the parking garage, is very slow compared to the processing speed of a PLC or SBC.

Operator Interfaces and Communication

In order to convey information about machine or process status, or to allow an operator to input data, many applications require operator interfaces. Traditional operator interfaces include pushbuttons, thumb-wheel switches, pilot lights, and LED numeric displays. Electronic operator interface devices display messages about machine status in descriptive text (“Motor 1 On”), display parts count and track alarms. They can also be used for data input (see section 3.9 for details).

Communication involves sharing application data or status with another electronic device, such as a computer or a monitor in an

operator's station. Communication can take place locally through a twisted-pair wire, or remotely via telephone or radio modem. PLC-based control systems are designed to support communication and electronic operator interfaces, where relay-based systems are not. SBC-based systems typically support communications, and some operator interfaces.

As it has been defined, the parking garage control system does not require operator interfaces beyond the ticket request pushbutton, the green enter light and the alarm horn. However, advanced communication capabilities could provide benefits. For example, if a portion of the garage was being repaired and 50 parking spaces were eliminated, it would be advantageous for the garage operator to change the control system parameters so that only 450 vehicles could be admitted. In addition, the control system could also let drivers know an area had been temporarily closed.

Environment

Consider the environment where the control system will be located. Will it be subjected to temperature extremes? Water? Humidity? Salt? Shock? Dust? Vibration? In harsh environments, house the control system in an appropriate NEMA- or IP-rated enclosure. Also, remember to consider accessibility for maintenance, troubleshooting or reprogramming.

If the control system for the parking garage is located in the ticket machine, it needs to be housed in an enclosure to protect it against moisture and dirt. Considering that outdoor temperature extremes may exceed the control system operating temperature, the enclosure may also need temperature and condensation controls. See the section on

5

"Installation Requirements" later in this chapter for further environmental considerations.

5.2

Selecting a Control Method

Once application requirements have been defined, the next step is determining which type of control method can accomplish the task.

As noted at the start of this chapter, system designers can select from three types of control systems: relays, PLCs or SBCs. To help determine which control method is best suited for the task, develop a chart which integrates application requirements with control methods. The following chart (Fig. 5-1) has been filled out for the parking garage example.

PLC Advantages

While relay-based control systems can perform some "advanced" functions (typically timing and counting, with limited sequencing), a wide range of higher level instructions can only be performed by PLCs or SBCs.

The data acquisition and communication capabilities of PLCs also deserve special mention, as they far exceed the capabilities of traditional relays. PLCs can gather information from the machine for production and status reports, out-of-spec or faulty parts count, total parts count, production rates, and machine run time (which is valuable for periodic maintenance operations). Further, PLCs can communicate this data to other control equipment or to operators in remote locations.

Application Characteristic	Required?	Quantity	*Can the control method accomplish task?* Relay	PLC	SBC
Inputs	Yes	7	Yes	Yes	Yes
Outputs	Yes	6	Yes	Yes	Yes
Timers	No	0	Yes	Yes	Yes
Counters	Yes	1 up/down	Yes	Yes	Yes
High speed required?	No	0	No	Yes	Yes
Data calculations?	No	0	No	Yes	Yes
Data acquisition	No	0	No	Yes	Yes
Communications	No	0	No	Yes	Yes
Operator interfaces	No	0	No	Yes	No (typically)

Fig. 5-1 Comparison of application requirements and control options.

As Fig. 5-1 shows, all three control methods can accomplish the task, so selecting a control method cannot be based on application requirements alone. However, this does not mean that all three methods provide the optimum solution. To differentiate between control methods, evaluate the relative cost impact of each method using the following criteria:

Criteria	Relays	Micro PLCs	SBCs
System design and development	Not applicable	Not applicable	****
Control system hardware	**/***	*/**	*
Panel assembly	***	*	*
Panel space	***	*	*
Implementing logic	***	**	***
Duplicating application	****	*	*
Documenting logic	****	*	**
Modifying logic	****	*	**
Maintenance	***	*	**

Fig. 5-2 Relative cost comparison of control methods.

* = *Low*
** = *Moderate*
*** = *High*
**** = *Very high*

Space and Cost

System designers usually consider physical space and cost for components the two most important issues — by far. Many applications, especially machinery, have a small, finite amount of space allocated for

controls. If an *assembled* control system occupies more space than allotted, it often cannot be used because too many changes to the machinery would need to be made to accommodate it.

Once mounted on a panel, a relay-based control system typically occupies much more space than the equivalent control implemented with a micro PLC or SBC. With micro PLCs available in the size of a brick and smaller, only the simplest relay-based system takes up less space. With the control system for the parking garage requiring 13 I/O and a counter, a micro PLC or SBC are the most “space efficient” control solutions.

Several cost factors influence the selection of a control method, including control system design and development, costs for components, assembly, space, and logic implementation.

- **Control system design and development costs** are incurred in the design of the system.
  - For a relay system, these costs are not applicable as the components have already been designed and produced.
  - For a micro PLC, these costs are not applicable because the PLC has already been designed and produced.
  - For an SBC, costs involve securing the services of an electronic engineer to design the board and test its viability (unlike relays and PLCs, SBCs are not typically available “off-the-shelf”).

  Note: Many installations require the control system to meet global industrial standards, such as UL, CE or CSA. PLCs usually have been certified to meet those standards, where relay- and SBC-based systems typically are not.

- **Component costs** are for the control-related hardware. Costs also include receiving, inventory, and the quality control of the components.
  - For a relay system, this includes relays, mechanical timers, and counters.
  - For a micro PLC, all necessary hardware is packaged in the PLC.
  - For an SBC, this includes the board, its components, and circuitry.
- **Assembly costs** cover putting the components together so they are usable.
  - For a relay system, this includes mounting components on a panel and wiring the logic power.
  - For a micro PLC, the only assembly costs are for mounting the unit to a panel with screws or on a DIN rail.
  - For an SBC, this involves securing a manufacturing facility to produce it. For this reason, SBCs become economically viable only in high volume or very unique applications.
- **Panel space costs** include the size of the panel and the enclosure needed to house the control system. The larger the enclosure, the greater the material costs for it.
  - For a relay system with many components, size could be prohibitive.
  - For a micro PLC, size is minimal.
  - For an SBC, size is usually minimal.
- **Logic implementation costs** relate to the “installation” of the logic into the control system (assuming costs for developing the logic are similar for all three control methods).

- For a relay system, implementing logic involves wiring the components together. Each subsequent application requires the same amount of labor to assemble, debug, and adjust timer and counter presets.
- For a micro PLC, costs include purchase of programming software or a Hand-Held Programmer. Programming a subsequent application only requires downloading the program; there are no *program* debugging costs for duplicate applications. However, users still need to commission each control system (see Chapter 6).
- For an SBC, costs involve retaining an electrical engineer to program a microprocessor. Programming each subsequent application typically requires copying a memory chip; there are no program debugging costs for duplicate applications. Commissioning is also required.

Future Costs

Total costs for a control system don't end after implementation. After system start up, it may be necessary to modify the control logic, document system changes, and troubleshoot the system.

With a relay-based system, re-wiring costs associated with logic changes can be extraordinarily high — it was just this type of situation that prompted General Motors to call for PLC development in the first place. The labor involved with relays can be intensive and costly, especially if more than one machine needs rewiring. Further, documenting relay wiring logic changes requires drafting a new wiring diagram. Because this task is so tedious (and adds cost), system changes can go undocumented. In fact, short of tracing every wire, there

is no way to ensure that the latest wiring diagram actually reflects the logic being executed by the system.

With an SBC-based control system, users typically cannot communicate with the microprocessor, nor is there programming software available. Logic changes are not easy to implement, automated documenting capabilities do not usually exist, and users typically cannot upload or download programs. SBC-based systems are difficult to troubleshoot because they rarely have troubleshooting features built into their software. Users of these systems must go to the manufacturer for support because no one else understands the SBC operation.

PLCs offer considerably more flexibility. Programming software facilitates relatively quick logic changes, and permits the new program to be easily downloaded to multiple machines. The program is always up-to-date, and documentation is accomplished with the push of a button. Troubleshooting help and diagnostic functions are a standard part of the software, and can be conducted with the Hand-Held Programmer as well (see Chapter 6).

PLCs are the easiest control system to support. Assistance for programming and troubleshooting is available at reasonable costs from many sources. And, if a PLC fails, a replacement PLC can be purchased off-the-shelf from the nearest industrial electrical supplier — there is no need to wait for a shipment from the factory. Furthermore, the ruggedness of PLCs compared to SBCs gives them a definite advantage in harsh environments or when durability is a primary consideration.

5

Selecting the Micro PLC

For all criteria by which control systems are evaluated — cost, size, flexibility, and supportability — micro PLCs provide the user with distinct advantages over other control options for many control applications. Thus, a micro PLC has been selected to provide the logic control for the parking garage.

5.3

What are the PLC Specifications?

After determining application requirements and selecting a method for providing system control, the next step is to determine specifications for the control system. When determining PLC specifications, identifying application requirements in certain categories can be helpful. Categories that typically need to be considered are:

- Total number of I/O
- Electrical requirements
- Output circuits
- Memory requirements
- Speed of operation
- Communication
- Operator interfaces

I/O Total

To determine a PLC's I/O requirements, examine the application requirements to determine how many input and output devices the PLC needs to monitor and control.

Reviewing the I/O requirements for the parking garage, a PLC for this application requires seven inputs and six outputs.

Note: When determining I/O total, many people add an extra 10% for unanticipated I/O needs, as well as future changes to the control system.

Electrical Requirements

To determine a PLC's electrical requirements, consider the voltage and current requirements for the PLC (incoming power), each output, and the inputs.

Until recently, micro PLCs operated on 24V dc — only. This limitation often necessitated installing a dc power source, especially when the other control system components operated on 120V ac. Newer micro PLCs, however, offer users standard voltage options: 24V dc, 120V ac, or 240V ac. For the parking garage, a PLC using 24V dc may be the best choice for the stated safety reasons. However, if the PLC is not located inside the ticket machine, using 120V ac may be acceptable.

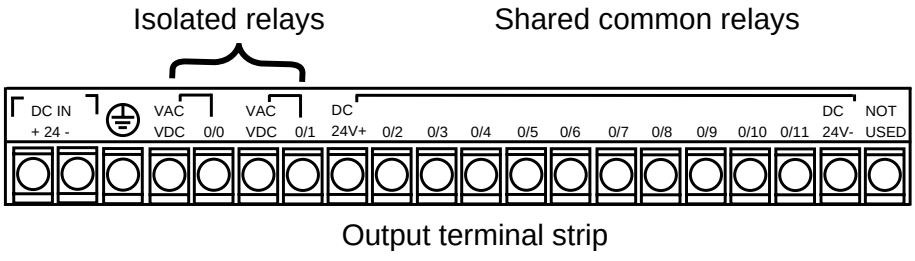


Fig. 5-3 Diagram of a micro PLC's output terminal. Note that output terminals O/0 and O/1 are isolated relays.

For applications requiring control of different output voltages, the PLC selected needs to have “isolated output terminals” to keep the voltages separated. [Note: Power from different sources or of different voltages must be isolated from each other.] In the parking garage example, the 120V ac gate controller signal must be isolated from the other output signals, which are 24V dc. Some micro PLCs now offer individually isolated outputs, with other outputs on different commons (Fig. 5-3).

A micro PLC accepts signals for all its inputs at the same voltage level, usually 120V ac or 24V dc. The application requirements and the power available dictate which voltage is selected. Recall that for the parking garage, the inputs operate on 24V dc for safety reasons. The chart below summarizes the electrical requirements for the parking garage:

Incoming power	Output voltages	Input voltage
24V dc	120V ac (2 devices) 24V dc (4 devices)	24V dc (7 devices)

Output Circuits

Recall from section 3.2 that micro PLCs are available with different types of outputs to suit different situations. For the parking garage, relay outputs will work best. Relays can switch both dc and ac current, have adequate response times, and wear is not a significant issue. In addition, micro PLCs with relay outputs usually cost less than those with solid state outputs.

For applications requiring fast response or having a high cycle rate (such as a high-speed cut-to-length line), a micro PLC with solid state output circuits (transistor, FET or triac) might be the optimum choice.

These circuits respond faster and do not wear out because there are no moving parts.

Memory Requirements

To quickly estimate the memory an application requires, a general rule is to add the number of I/O and then multiply by 10, where 10 is the words of memory needed per I/O. The parking garage control system has 13 I/O, plus one "extra" for expansion, yielding a total of 14. 14 x 10 words = 140 estimated words of memory required.

Today, nearly all micro PLCs have at least 1/2K of memory available for application programs (1/2K equals 512 words). For the parking garage control system, as well as most low I/O count applications, micro PLCs usually have more than sufficient memory. Typically, applications will exceed a micro controller I/O capacity before its memory capacity.

Once the logic required for an application has been developed, PLC users can calculate how much memory a program will consume by referring to the PLC operator's manual, which typically lists memory use for all of the instructions. See the worksheet in Appendix D for an example.

Speed of Operation

If application requirements indicate the need for a PLC with high-speed operation, look for a PLC with the following features:

- Adjustable input filters (see section 3.1).
- Transistor, FET or triac outputs (not relays — see section 3.2).
- High-speed counter, high-speed interrupts, and immediate outputs.

5

High-speed counters, high-speed interrupts and the ability to immediately update outputs allows PLCs to meet the demands of most high-speed applications. On user-specified conditions, high-speed interrupts and immediate output instructions direct the PLC to immediately process the logic and update the I/O — independently of the normal program scan. This can substantially improve speed and performance.

Simplifying the program also increases performance, because program length directly impacts scan time. Every instruction in a program takes time to execute, and reducing or simplifying the program reduces time. PLC users can calculate program execution time by referring to the PLC operator's manual, which should list execution times for all of the instructions. See the worksheet in Appendix D for an example, as well as a listing of typical instruction execution times in Appendix C.

5.4

Program Development Procedures

Even the simplest programs rarely go directly from the programmer's head to the PLC. In fact, attempting this "time saving" step often prolongs the process. Instead, begin by writing out the operation sequence — both sentences and flow charts work well. There are three steps to developing a sequence of operation:

- Define the rules of operation for each control point.
- Identify and label inputs and outputs.
- Convert the rules of operation to ladder logic.

Define Rules of Operation

What conditions permit or prevent responses from the control system? Defining these conditions is known as developing the rules of operation. To begin, carefully describe the control system at its most basic level. Recall from section 5.1 that the parking garage control system was described like this:

- The driver approaches an automated ticket machine at a gate.
- The driver pushes a button on the ticket machine to receive a ticket. The machine should not provide a ticket if the lot is full or the gate is up.
- Removing the ticket raises the gate and turns on a green light.
- After the car clears the gate, the gate lowers and the green light shuts off.
- The vehicle population is known at any time.
- If maximum capacity is reached (500 cars), a “Full” sign is illuminated, the ticket machine will not provide a ticket and the gate will not raise.
- An alarm sounds when the gate is obstructed.

Outputs	**Inputs**
Provide ticket	Ticket request pushbutton
Raise gate	Ticket taken limit switch
Lower gate	Vehicle cleared gate photo sensor
“Garage Full” sign	Car departed garage photo sensor
Green (enter) light	Gate obstructed (motor overload contact)
Alarm	Gate up proximity sensor
	Gate down proximity sensor

5

To control any machine or process, first identify each action, or control point. Ask, “What action is the system controlling?” Then, create a simple description of the conditions that control each action. Start with the control point and work back to define the conditions (inputs) that produce the desired action. Notice that each control point corresponds to an output on a rung of the ladder program.

When carefully written, the rules of operation convert easily to a ladder logic program, as the parking garage example shows:

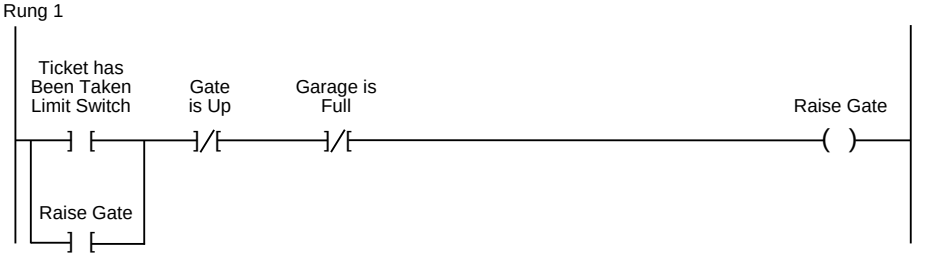
Rules of Operation

Control point:
- The ticket machine will provide a ticket

Conditions:
- If the driver presses the ticket request pushbutton
- AND the “Full” sign is NOT on
- AND the gate is lowered

Control point:
- Raise the gate until fully up

Conditions:
- After the driver takes the ticket
- AND the gate is NOT up
- AND the “Full” sign is NOT on



Control point: • Vehicle present latch

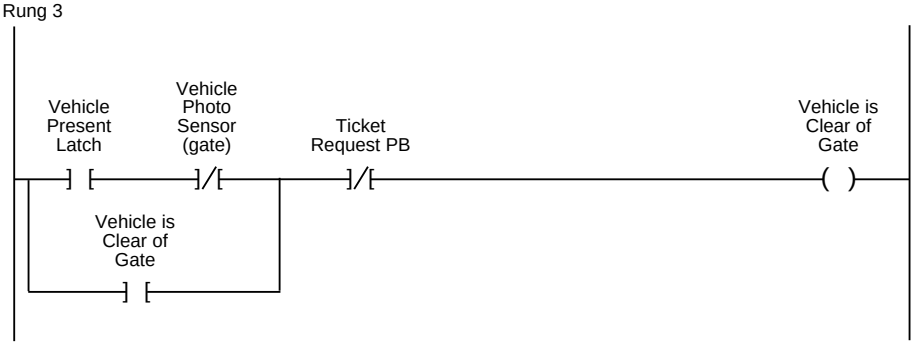
Conditions: • Vehicle has been detected
• AND the vehicle has NOT cleared the gate

Rung 2

Control point: • Vehicle clear of gate

Conditions: • Vehicle present latch is on
• AND a vehicle is NOT detected
• AND the ticket request pushbutton is NOT pressed

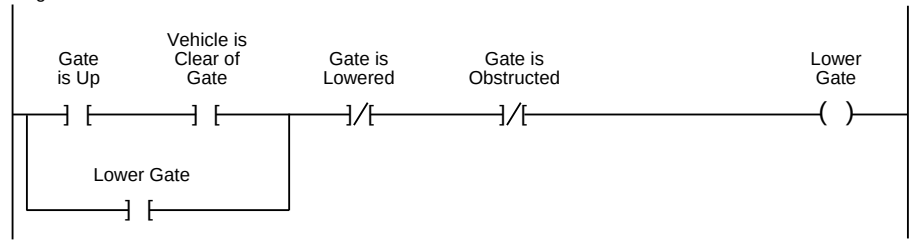
Rung 3



Control point: • Lower the gate until fully down

Conditions:
- If the gate is up
- AND the car has cleared the gate
- AND the gate is NOT down
- AND the gate is not obstructed

Rung 4



Control point: • Turn on the green light

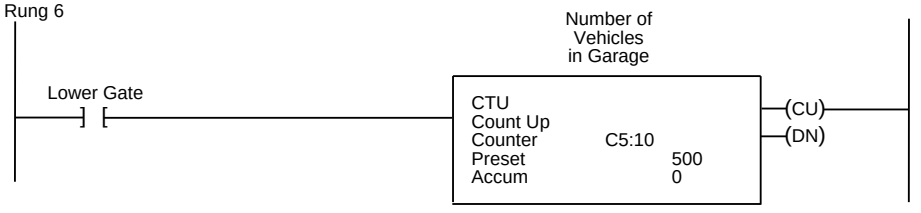
Condition: • If the gate is up

Rung 5

Control point: • Count cars entering/turn on full sign at 500th car

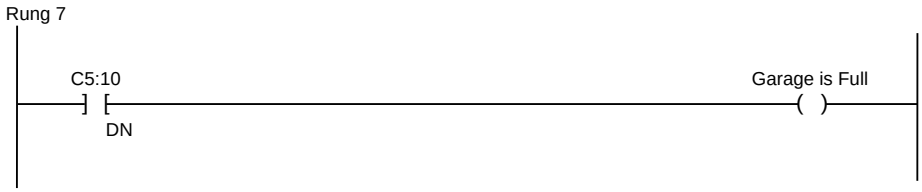
Conditions:
- If the gate has been lowered
- If accumulated counter value ≥ preset value of 500

Rung 6



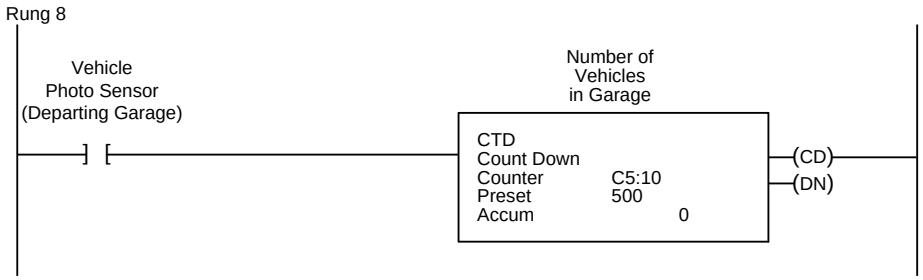
Control point: • Turn on the “Full” sign

Condition: • If accumulated counter value ≥ preset value of 500



Control point: • Decrement the counter (count departing vehicles)

Condition: • If a vehicle departs the garage



Control point: • Sound alarm

Condition: • If the gate is obstructed

Programming Tips

- When programming condition instructions, refer back to Fig. 4-8 to determine if a normally open or a normally closed instruction produces the desired action.

- When defining the rules of operation, the text should use language that helps convert the operating characteristics to ladder logic. Recall from Chapter 4 that AND logic connects instructions in series on a ladder diagram rung, while the OR logic connects instructions in parallel.
- If an output needs to remain on after the condition that originally energized it is no longer present, use an auxiliary holding contact or a latched output.
- A condition instruction can be used more than once in a program because it exists in the software (a benefit over hardwired relays). Also, remember that the status of an output can be used as a condition instruction.
- Only program a specific output instruction once. If an output instruction with the same address is programmed more than once, the last occurrence of the instruction in the user program will determine the actual output state.
- When each I/O (field device) is wired to a terminal on the PLC, it then has a unique address which corresponds to that terminal.
- Follow the instruction manual! Each PLC manufacturer uses slightly different terms and techniques. These should be noted and followed carefully.

Addressing

All elements of a ladder diagram are labeled with a letter/numerical designation. Because every PLC manufacturer has a variation of this designation, be sure to follow the addressing conventions outlined in the operator's manual.

The parking garage example uses “I” to indicate inputs, “O” for outputs. All input and output terminals in this example are numbered starting with zero (0). The program for the parking garage has its inputs and outputs addressed as such:

Input address

I/0 Ticket request pushbutton
I/1 Ticket taken limit switch
I/2 Car cleared gate photoelectric sensor
I/3 Car departed garage photoelectric sensor
I/4 Gate obstructed (motor overload contact)
I/5 Gate up proximity sensor
I/6 Gate lowered proximity sensor

Output address

O/0 Ticket provided solenoid
O/1 Gate up motor controller
O/2 Gate down motor controller
O/3 Garage Full sign
O/4 Green light
O/5 Alarm horn

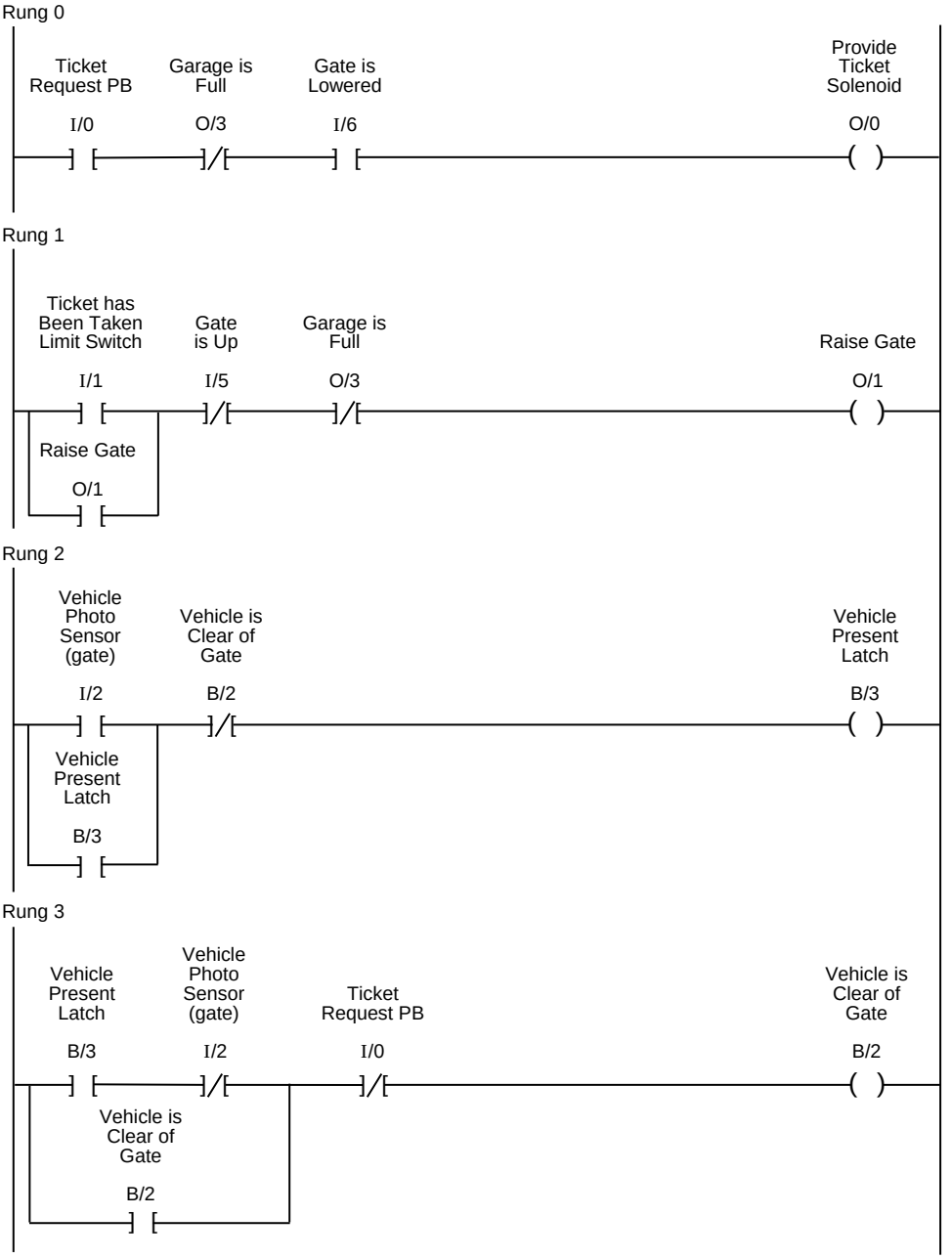
Counter address

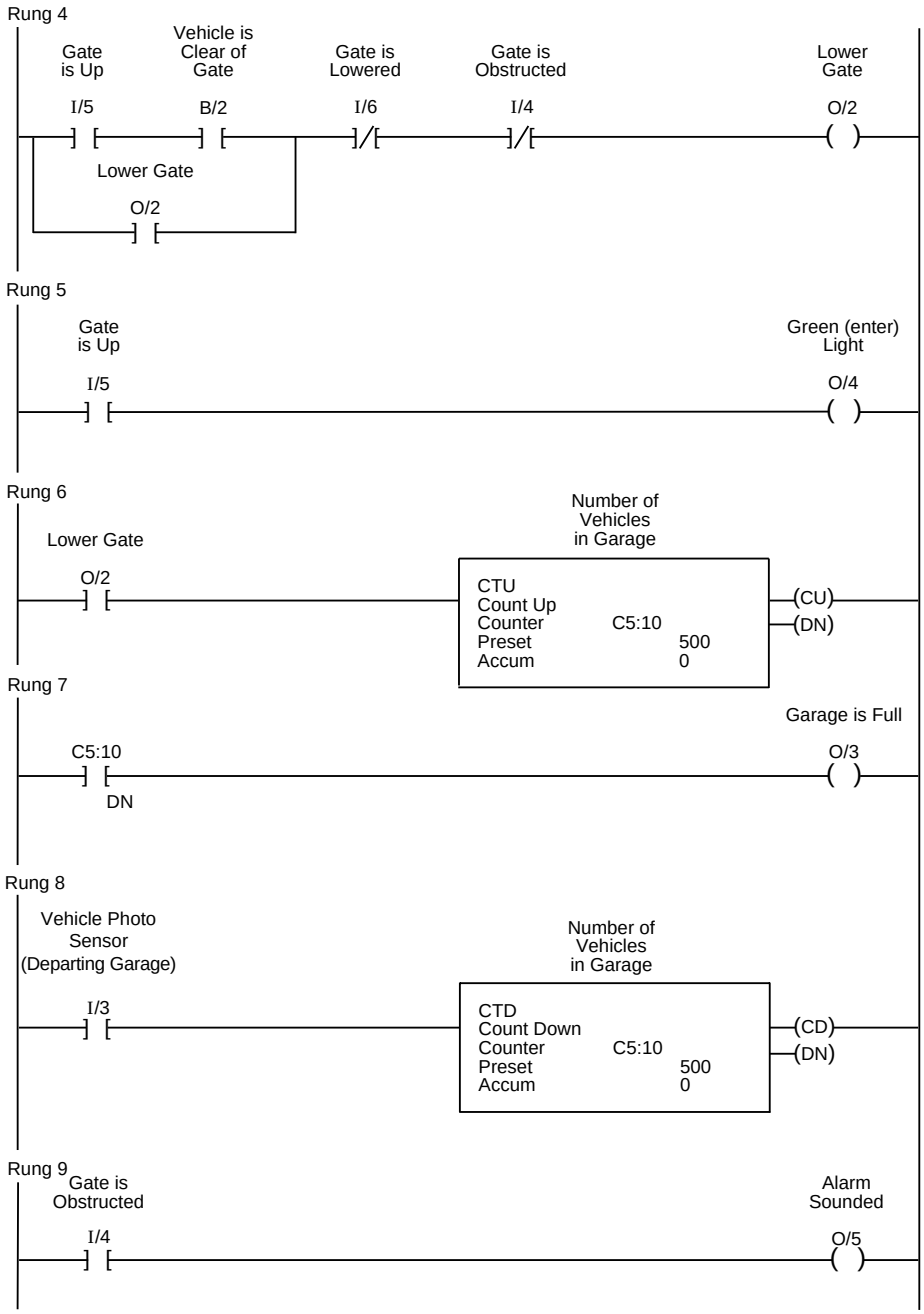
The program for the parking garage also needs two counters (notice that the counter uses an internal address):

C5:10 Count Up (CTU), for cars entering
C5:10 Count Down (CTD), for cars departing

With the addresses inserted, the program for the parking garage is complete and looks like this (see next page):

5


Rung 0
Ticket Request PB
I/0
Garage is Full
O/3
Gate is Lowered
I/6
Provide Ticket Solenoid
O/0
Rung 1
Ticket has Been Taken Limit Switch
I/1
Gate is Up
I/5
Garage is Full
O/3
Raise Gate
O/1
Raise Gate
O/1
Rung 2
Vehicle Photo Sensor (gate)
I/2
Vehicle is Clear of Gate
B/2
Vehicle Present Latch
B/3
Vehicle Present Latch
B/3
Rung 3
Vehicle Present Latch
B/3
Vehicle Photo Sensor (gate)
I/2
Ticket Request PB
I/0
Vehicle is Clear of Gate
B/2
Vehicle is Clear of Gate
B/2


Rung 4
Gate is Up
I/5
Vehicle is Clear of Gate
B/2
Gate is Lowered
I/6
Gate is Obstructed
I/4
Lower Gate
O/2
Lower Gate
O/2
Rung 5
Gate is Up
I/5
Green (enter) Light
O/4
Rung 6
Lower Gate
O/2
Number of Vehicles in Garage
CTU
Count Up
Counter C5:10
Preset 500
Accum 0
(CU)
(DN)
Rung 7
C5:10
DN
Garage is Full
O/3
Rung 8
Vehicle Photo Sensor (Departing Garage)
I/3
Number of Vehicles in Garage
CTD
Count Down
Counter C5:10
Preset 500
Accum 0
(CD)
(DN)
Rung 9
Gate is Obstructed
I/4
Alarm Sounded
O/5

5

5.5

Installation Requirements

A PLC user's manual contains detailed installation instructions pertinent to that particular model, and they should be followed carefully. As with any product being installed, proper planning assures smooth start-up. When installing micro PLCs, consider the physical and electrical environments and requirements for power, mounting and wiring. The following are some suggestions for installing PLCs.

Physical Environment

Whether the micro PLC is mounted within a machine or in a separate enclosure, it requires protection against temperature extremes, humidity, dust, shock, vibration, or corrosive environments.

- Be careful about locating the PLC in an enclosure with other heat-generating sources; 55°C is the maximum ambient operating temperature for most micro PLCs. Ensure sufficient ventilation and space between components. Install a fan to help circulate the air if necessary.
- Installing the PLC in a NEMA Type 12 (IP 60) enclosure provides protection against dust, falling dirt, and dripping noncorrosive liquids. A NEMA 12 enclosure is rated for both indoor and outdoor installation.
- Installing the PLC in a NEMA Type 4 (IP 65) enclosure provides protection against windblown dust and rain, splashing and hose-directed water, and external icing. A NEMA 4 enclosure is also rated for both indoor and outdoor installation.

- Enclosures do not protect against the internal condensation that can occur with temperature fluctuations. To protect against condensation, as well as extreme cold (below 0°C), consider installing some type of heating element in the enclosure.

Electrical Environment

- Do not mount the PLC near high voltage equipment, such as motors and arc welders, as electrical interference could cause errors. A properly grounded steel enclosure helps reduce electrical interference.
- If possible, do not locate the PLC on the same power feed as high frequency equipment, such as inverters (ac drives). Power "filtering" may be required for "dirty" or "noisy" electrical environments.
- Using a shielded, twisted-pair cable (with the shield connected to ground at one end) between field devices and the input terminals reduces the effects of high frequency disturbances.

Power

- Follow the manufacturer's recommended procedures for wiring the PLC.
- Place the main power disconnect switch where operators and maintenance personnel have quick and easy access to it. If the built-in disconnect switch is mounted inside an enclosure, make sure to install an externally panel-mounted switch.

Mounting

- Mount the micro PLC using the manufacturer's recommendations.

Generally, mount the PLC to the back panel or sides of an enclosure — not the top or bottom — using either a DIN rail or mounting screws. Be sure to provide proper ventilation.

- Do not exceed the shock and vibration specifications published by the PLC manufacturer. Avoid sources of high vibration. Use cushioned mounting if necessary.
- Allow enough clearance between the door and the components. Consider using documentation pockets, which often are affixed to the inside of the door.

Wiring

- Allow at least 2 in. (50 mm) between I/O wiring ducts or terminal strips and the PLC for ease of access during installation and maintenance.
- Do not run signal or communication wiring and power wiring in the same conduit. Wires with different signal characteristics should be routed along separate paths.
- Follow manufacturer's grounding instructions carefully.
- Inductive output devices such as motor starters and solenoids may require surge suppression to protect the PLC output contacts. Locate the suppression device (e.g., a varistor for an ac load, a diode for dc) as close as possible to the output device.

CHAPTER 6

Commissioning and Troubleshooting

Commissioning 6.0
Troubleshooting Overview 6.1
Finding the Problem. 6.2
Troubleshooting the PLC 6.3
Troubleshooting I/O 6.4
Program Troubleshooting. 6.5
Faults 6.6
Safety 6.7
Troubleshooting Model. 6.8

6

6.0

Commissioning

Preparing a control system for start-up, also called commissioning, involves executing a series of tests to ensure that the PLC, the ladder logic program, the I/O devices and associated wiring operate according to specifications.

Before commissioning any control system, the technician must have a clear understanding of how the control system operates and how the various components interact (e.g., sequence of operation, timing, and speed-related issues). For a PLC-based system, understanding the application can be accomplished by studying a printout of the current program. If properly documented, the printout should note addresses for I/O devices and contain comments describing the operation of each program rung.

Assuming installation is complete and the application program has been loaded into the PLC, the following checklist provides a good guide for commissioning a PLC:

1. Be aware of the hazards posed by inadvertently energized outputs. Before applying power to the PLC or the input devices, disconnect or otherwise isolate any output device that could potentially cause

damage or injury (typically an output that causes movement like starting a motor, opening a valve, etc.).

2. Apply power to the PLC and the input devices. To verify that there is proper power, check the PLC and input devices with a voltmeter. If there is a power problem, tighten connections and check for broken wiring or faulty input devices.
3. Examine the PLC's LED status indicators. If power is properly applied to the PLC, the “power” indicator should be On, and there should be no “fault” indication (Fig. 6-1). If the PLC is not powering up properly, the PLC may be faulty. However, remember that PLCs rarely fail. But if they do fail, it usually happens immediately upon powering up. A PLC almost always functions either as designed or not at all (they are designed not to run on a fault).
4. After making sure that the PLC has power, verify communication with the PLC. To do this, use a Hand-Held Programmer (HHP) or a PC running the PLC programming software. If communication is possible, the technician can assume the PLC is functional.
5. Place the PLC in a mode that prevents it from energizing its output circuits. Depending on the make of the PLC, this mode may be called the “disable,” “test scan” or “stop” mode. This mode permits the PLC to monitor input devices, execute the program, and update the output image file while keeping the output circuits de-energized.
6. One at a time, manually activate each input device. Verify that the PLC's input status LEDs turn On and Off as expected. Using the HHP or PC, monitor the associated condition instruction to verify that the input device corresponds to the correct program address,

and that the instruction turns On and Off as expected. If they do not operate as expected, see the “troubleshooting” section of this chapter.

7. Manually test each output. Many technicians do this by applying power to the terminal where the output device is wired. This checks the field device and its associated wiring.
8. After verifying all inputs, outputs and program addresses, verify all preset values for counters, timers, etc.
9. Place the PLC in the run mode and verify that the “run” LED is On. Reconnect any output devices that were disconnected in step 1. Test all emergency stop buttons. Test total system operation.

6.1

Troubleshooting Overview

When a control system error occurs, many new PLC users first suspect the PLC is at fault. Usually, this assumption is unjustified, as devices other than the PLC, such as sensors, solenoids and wiring, cause the vast majority of faults. It is worth repeating that PLCs are among the most rugged, durable and reliable control equipment available today. However, faults are inevitable in any control system, including PLCs. Fortunately, PLCs have been specifically designed to incorporate troubleshooting aides that enable users to get the

application up and running quickly. This is an advantage over relays, SBCs and other control solutions.

Troubleshooting consists of three activities: understanding how the application (control system) operates, finding the problem and correcting it. Before troubleshooting any control system, the technician must understand how the system works and how the various components interact. As with commissioning, a hard copy of the program is required.

6.2

Finding the Problem

If a control system has been operating, the technician should be confident of the accuracy of the program logic. In this case, malfunctioning field devices or loose wiring associated with the field devices cause most errors. For a control system that has never worked (e.g., just being commissioned), programming errors should also be considered.

Before spending hours troubleshooting a system and searching for a “complicated” problem, first rule out any obvious problem (e.g., a broken belt or jammed machinery). Then, cycle power to the PLC. Remember that power surges or other momentary problems may have caused the PLC to stop, and it may only need to be re-started.

6

6.3

Troubleshooting the PLC

If the PLC is running properly, its power and run LEDs should be On, and there should be no fault indication (refer to Fig. 6-1). If the fault LED is On, use a Hand-Held Programmer (HHP) or a PC running the PLC programming software to determine the cause of the fault. Then, consult the user manual to determine possible causes and corrective actions (refer to section 6.6 and Fig. 6-2 for more details).

If all LEDs are Off, verify that the PLC has proper power with a voltmeter. If a power problem exists, verify that all wiring connections are good and that there are no broken wires. Check for power from the circuit breaker or fuse block.

After verifying PLC power, check communication with the controller. Do this by using a Hand-Held Programmer (HHP) or a PC running the PLC programming software. If communication is possible, assume that the PLC is functioning properly, and investigate field devices, field wiring and field power.

6.4

Troubleshooting I/O

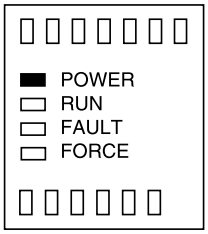
If attempts to re-start the PLC fail to solve the problem (and the PLC has power), most technicians start troubleshooting at the outputs and work backwards. This is usually the quickest and most efficient procedure. Typically, operators or technicians first notice a problem when an action (output) fails to occur.

Begin troubleshooting by examining the output LEDs. [Note: Using

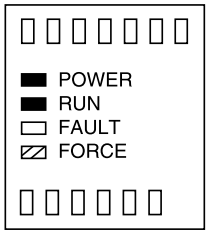
Understanding LED Status

Most micro PLCs have LEDs to indicate I/O status (On/Off) and to indicate if the controller: has power, is running, has faulted, and if a force exists. Between the time power is applied to the controller and the time it has to establish communication with a connected programming device, the only form of communication between the technician and the controller is through the LEDs.

When powered up:



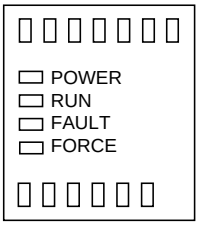
When placed in RUN:



Refer to the following key to determine the status of the LED indicators:

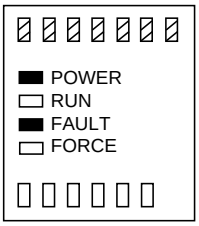
- Indicates the LED is OFF.
- Indicates the LED is ON.
- Indicates the LED is FLASHING.
- Status of LED does not matter.

If the LEDs indicate:



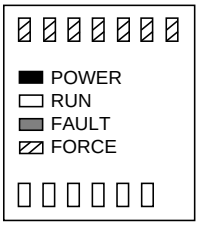
The Following Error Exists	Probable Cause
No input power or power supply error	No Line Power
	Power Supply Overloaded

If the LEDs indicate:



The Following Error Exists	Probable Cause
Hardware faulted	Processor Memory Error
	Loose Wiring

If the LEDs indicate:



The Following Error Exists	Probable Cause
Application fault	Hardware / Software Major Fault Detected

Fig. 6-1

the HHP greatly simplifies and speeds troubleshooting.]

- If the output LED is On and the output device is not On, test for power at the suspected output terminal.
  - If there is power at the output terminal, the PLC is functioning.
  - If power is not present on the PLC output terminal, the PLC has failed and must be replaced.
- Next, test for power at the non-functioning output device.
  - If there is power, then the device is faulty and should be fixed or replaced.
  - If there is no power at the device, then there is a blown fuse in the field wiring or another wiring fault between the PLC and the device.

If the PLC and output devices are functional, examine the program (a printout will be helpful, or use an HHP) and look at the rung(s) with the non-functioning output(s). Determine what condition instructions (inputs) need to be True to enable activation of the output(s) and start tracing them to find out which conditions are not satisfied.

- If the input device is supposedly On, but the corresponding input LED is not On, use the HHP or a voltmeter to check for a signal at the input terminal.
  - If there is no signal, examine wire connections between the terminal and the field device and tighten or repair wiring as necessary.
  - Check the devices for proper power, and see if the field device is broken.

6.5

Program Troubleshooting

The Hand-Held Programmer

For the technician troubleshooting PLCs in the field or on the plant floor, the value of a Hand-Held Programmer cannot be overstated. Features include an ability to:

- *Identify the status (On/Off) of any I/O or bit element.*
- *Display the data located in a higher level instruction, such as the accumulated value of a timer or counter.*
- *Trace or search for faulty instructions.*
- *Force instructions On or Off.*
- *Identify and clear faults.*
- *Download and upload programs.*

Hand-Held Programmers are also more rugged and portable than most PCs.

For a system that was working but has stopped, suspect the program only after checking the PLC and verifying the integrity of the field devices and associated wiring. However, the same procedure is used to debug new and existing programs.

Start program troubleshooting by identifying which outputs operate properly and which outputs do not. Then, using the HHP or programming software, trace* back from the output on the non-functioning rung and examine the logic to determine what may be preventing the output from energizing.

Typical logic errors include:

- Programming a normally open instruction instead of a normally closed instruction (or vice versa).
- Using an incorrect address in the program.

* Most programming software packages and HHPs have a feature called the "trace," "search" or "find" function. Simply enter the address of the instruction to be found, and the HHP searches for the first occurrence of that address. If the address is found, the search feature can also search for other instances of the same address. This lets you quickly find all occurrences of an address and verify that the logic associated with it is both correct and operating as expected (no I/O faults, etc.).

6

6.6

Faults

Fault messages are displayed on the HHP or programming software for easier problem identification. Error messages, coupled with information from the PLC user manual, help locate the fault, determine its cause, and suggest corrective actions. This “self-diagnostic” capability (which is not available with most other control systems) greatly facilitates troubleshooting. Some of the more common causes of faults include memory errors, data corruption errors, watchdog timer errors and momentary power problems.

6.7

Safety

After identifying the problem and determining the appropriate corrective measure, consider the following safety measures when repairing the system:

- Disconnect the power to the whole system while making repairs, and make sure there is no chance of someone inadvertently reconnecting the power.
- Make sure that no system elements can be harmed if and when the system is restored to working order.

- Some applications require all system components (field devices) to be in a “start” position (this is often due to mechanical considerations). Before bringing a control system back on-line, know the system requirements.
- After making repairs, ensure that the system works properly to the extent that operators and bystanders are not jeopardized by system operation. This may include partial or full testing of the system.

6.8

Troubleshooting Model

In addition to becoming familiar with all of the troubleshooting tools and techniques available, it’s important to develop a troubleshooting routine. The following error recovery model (Fig. 6-2) demonstrates a common routine for troubleshooting hardware and software problems. After expending all reasonable efforts to restore the PLC to proper operation, call your distributor or manufacturer. Good distributors and manufacturers employ skilled technicians and engineers who can provide assistance, often over the phone.

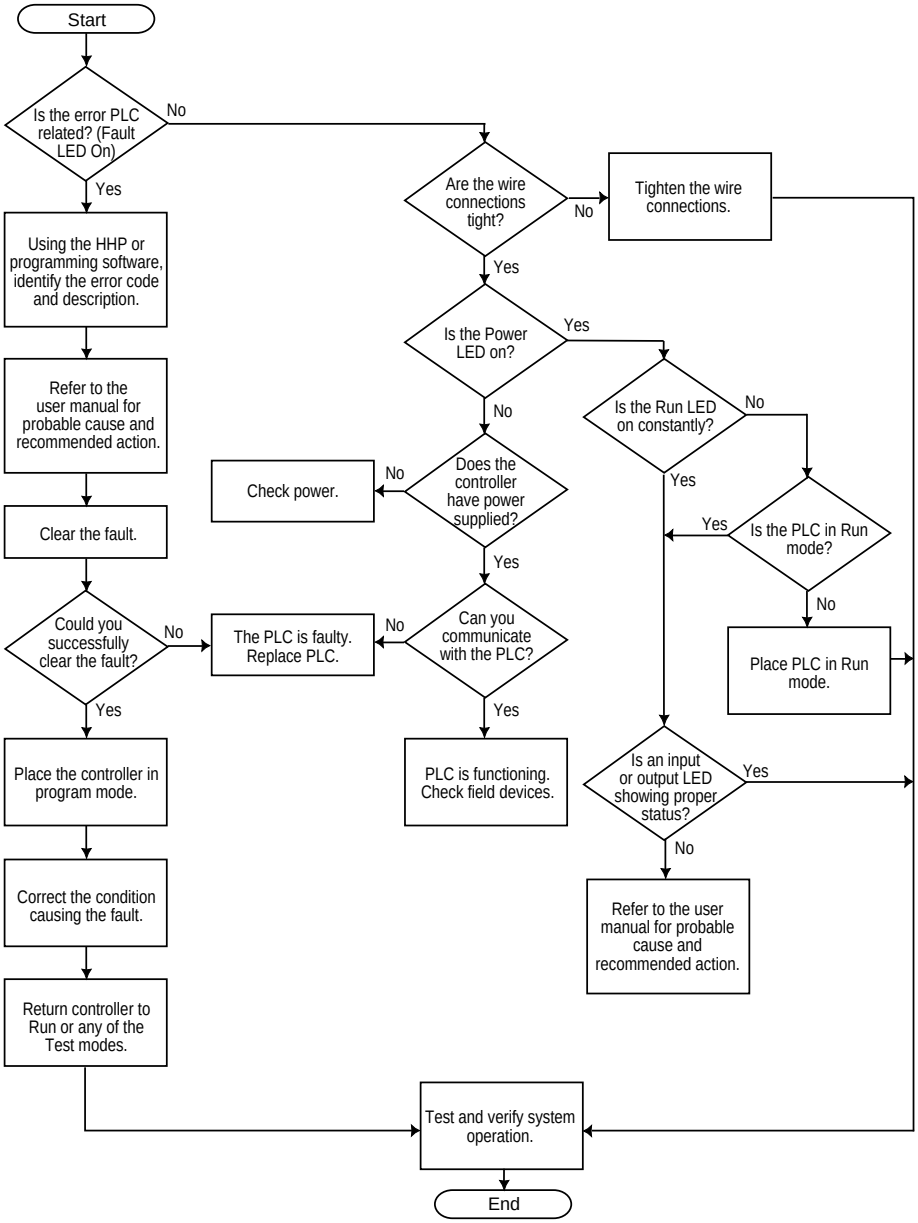


Fig. 6-2 Troubleshooting Model

CHAPTER 7

Application Examples

Introduction. 7.0

Basic Logic

- OR circuit 7.1
- AND circuit.....7.2
- Start/stop circuit 7.3
- Flip/flop circuit.....7.4
- Alarm circuit.....7.5
- Start/stop with jog.....7.6

Timing and Counting

- On delay.....7.7
- Off delay 7.8
- One minute clock.....7.9
- Up/down counting.....7.10

Data Instructions

- Moving data 7.11
- Comparing data.....7.12
- Math commands.....7.13

Advanced Instructions

- Sequencers 7.14
- FIFO.....7.15
- High-speed counter 7.16
- Two stage alternator 7.17
- Three station alternator 7.18

7

7.0

Introduction

As the parking garage example in Chapter 5 demonstrates, developing a ladder logic program for a PLC consists of identifying the logic required and building the program one rung at a time. While different programs can achieve the same outcome, every program uses the same building blocks: the micro PLC's instruction set.

This chapter takes some of the most commonly used instructions and demonstrates their use in control applications. In addition to explaining how PLC users can apply these powerful tools, the examples highlight typical micro PLC applications, and how to build complex programs from the simple steps shown.

7.1

OR Circuit

Uses

This type of logic is used to turn On an output device/control instruction when any input device/condition instruction in the rung provides logical continuity.

Operation

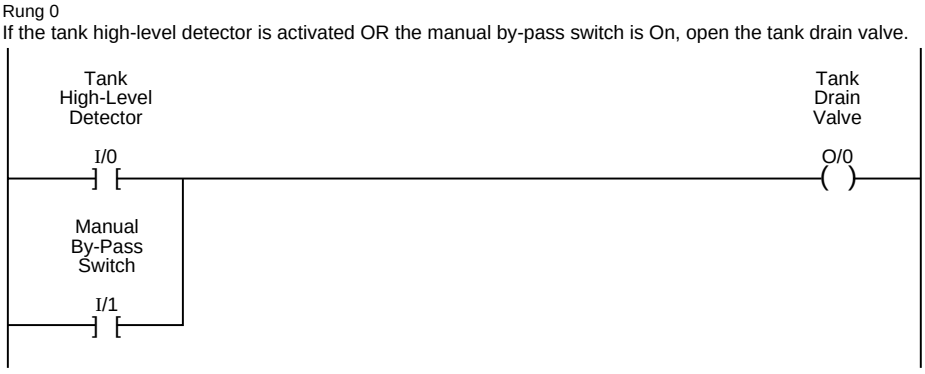
Turn On an output with more than one input device/condition instruction.

Ladder Logic

The logic used in this example consists of one rung with two condition instructions programmed in parallel.

RUNG 0

- This rung shows that whenever input device I/0 OR input device I/1 is On, output device O/0 will be energized.



7.2

AND Circuit

Uses

This type of logic is used to turn On an output device/control instruction when all input devices/condition instructions in the rung provide logical continuity.

Operation

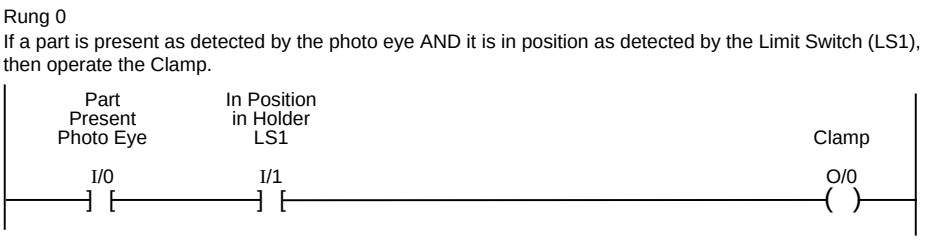
Turn On an output only when all input devices/condition instructions have logical continuity.

Ladder Logic

The logic used to perform this consists of one rung with at least two condition instructions programmed in series.

RUNG 0

- This rung shows that whenever input devices I/0 AND I/1 are On, output device O/0 will be energized.



7.3

Start/Stop Circuit

Uses

This is used to start a device with a momentary input and stop it with a second momentary input. Typically the start and stop input devices are momentary pushbuttons or a similar type of device. Once the start pushbutton is pressed, the output energize instruction will stay On until the momentary stop pushbutton is pressed.

Operation

Turn On an output with a momentary input, and keep it On until instructed to turn it Off.

Ladder Logic

The logic used to perform this consists of one rung. Note that in this example the stop pushbutton is a normally closed switch, but is programmed as a normally open instruction.

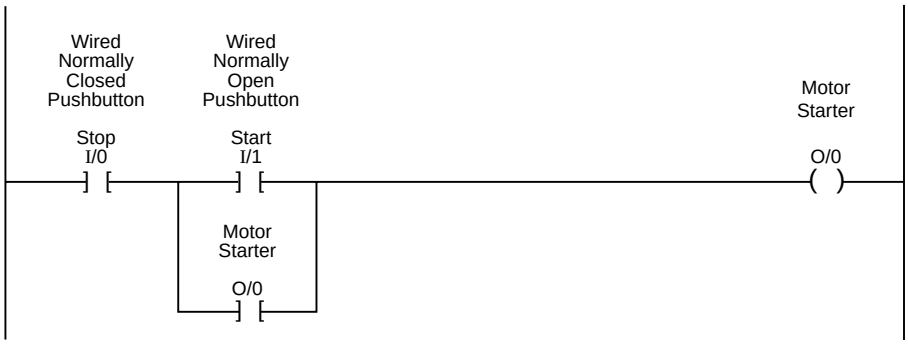
RUNG 0

- Before any inputs are activated, N.O. instruction I/0 is True (since a N.C. pushbutton is wired to input terminal I/0, and that pushbutton has not been pressed), and N.O. instruction I/1 is False.

  When the start pushbutton is pressed, N.O. instruction I/1 becomes True, energizing output O/0. The True status of *control* instruction O/0 is reflected in N.O. *condition* instruction O/0, which is programmed in parallel with the start instruction. This keeps the output On even when I/1 is no longer true.

  When the stop pushbutton is pressed, N.O. instruction I/0 becomes False, and the output is de-energized.

Rung 0
Start the motor running by pressing the Start pushbutton. Keep the motor running until the Stop pushbutton is pressed.



7

7.4

Flip/Flop Circuit (Push-On/Push-Off)

Uses

This circuit is used to provide a single change of state each time a new condition is detected. The mechanical equivalent of this function would be a push-On/push-Off pushbutton. This type of logic can be handy for a wide range of miscellaneous uses, such as alternators or memory circuits.

Operation

Turn On and maintain an output with momentary pushbutton; turn the output Off the next time the same pushbutton is pressed.

Ladder Logic

The logic used to perform this consists of three rungs that make use of special instructions. The logic also takes advantage of how the PLC scans the user program.

RUNG 0

- A momentary pushbutton wired to input I/5 is in series with a one-shot rising [OSR] instruction, B3/2, that controls output B3/0. An OSR is a specialized instruction that is only energized for one processor scan. This causes control instruction B3/0 to be energized for one processor scan. Another way to think of this is as a “leading edge” triggered device.

RUNG 1

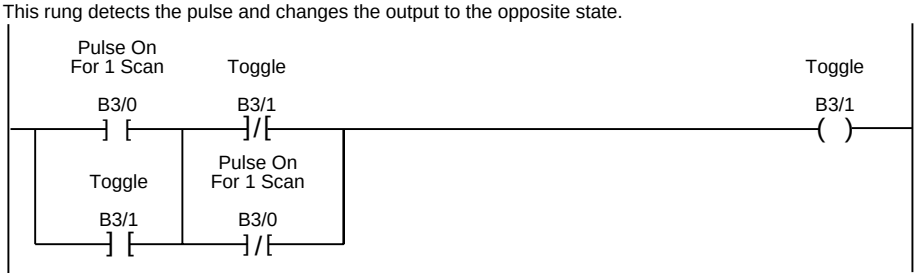
- The second rung detects the pulse each time the condition instruction I/5 is energized and changes the output to the opposite state each time the pushbutton in rung 0 goes True.

RUNG 2

- This rung directly controls the load device wired to terminal O/0.

Rung 0
This rung generates a pulse each time the pushbutton is pressed.

Rung 1
This rung detects the pulse and changes the output to the opposite state.



Rung 2
This rung uses the toggle bit to turn on the load device.

7

7.5

Alarm Circuit with Flash and Acknowledge

Uses

This type of logic is used to detect, hold, and reset alarm events.

Operation

- Detect the alarm condition and maintain the event.
- Flash an indicator to represent an alarm is present.
- Maintain the indication after the alarm has been acknowledged, but is still present.
- Reset (clear) the alarm.

Ladder Logic

The logic used to perform this operation uses three rungs. Note the use of the internal timer, S4/4, used here as the flasher.

RUNG 0

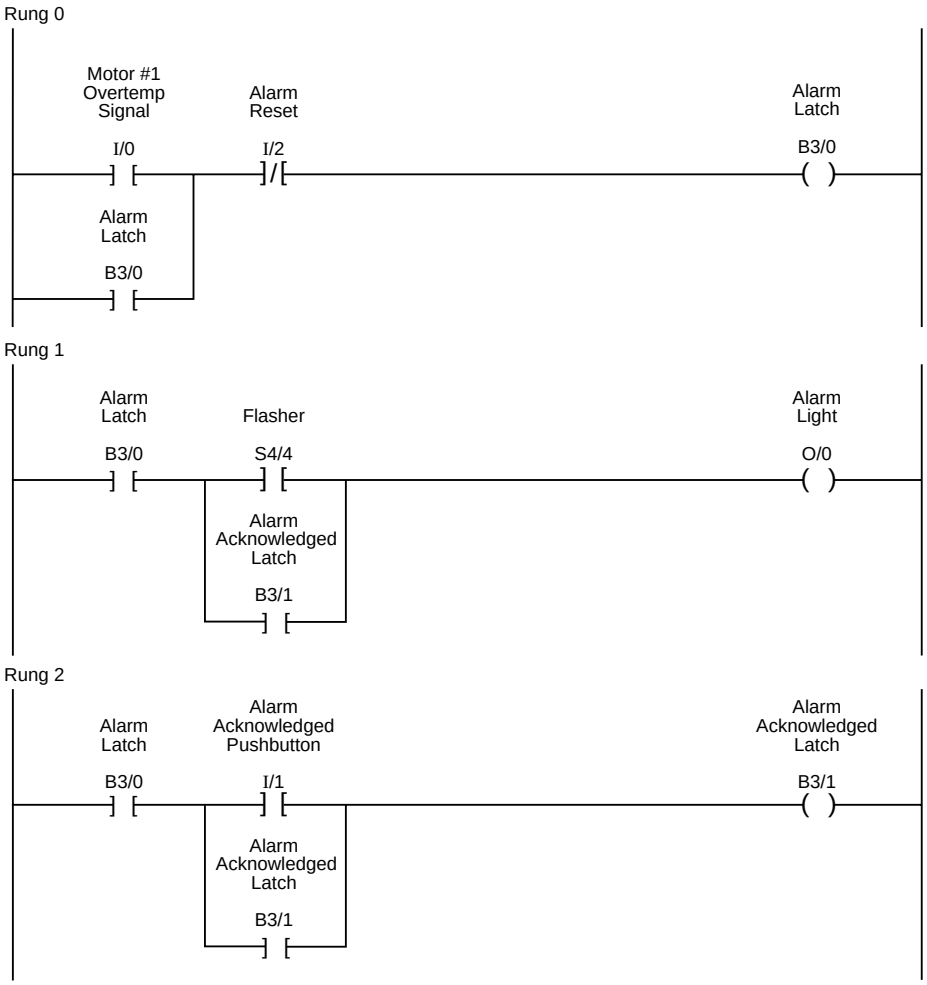
- This rung latches the alarm state. When motor #1 (input I/0) is over its temperature limit and the alarm reset button (input I/2) has not been pressed, the *control* instruction B3/0 is turned On.

RUNG 1

- This rung flashes alarm light O/0 when an alarm is present as indicated by the *condition* instruction B3/0 being On. In this example, S4/4 is an internal PLC address that cycles On and Off at .32-second intervals.

RUNG 2

- When the alarm condition is acknowledged by pressing the pushbutton I/1, control instruction B3/1 is energized. This address is also turned On as a condition instruction in rung 1, bypassing the flasher at address S4/4 and changing the state of alarm light O/0 from flashing to steady.



If at any time the alarm condition is corrected (the motor cools down), I/0 goes Off. The alarm condition will be maintained until an operator acknowledges the alarm. The alarm acknowledged pushbutton I/1 must be pressed to unlatch B3/0. This in turn de-energizes alarm light O/0.

7.6

Start/Stop with Jog Program

Uses

Use this logic to start a device with a momentary input, or to jog the device with a separate input.

Operation

Turn On an output with a momentary input and keep it On until instructed to turn it Off. Or, turn On an output whenever the jog pushbutton is pressed. If the jog pushbutton is released, the output must turn Off.

Ladder Logic

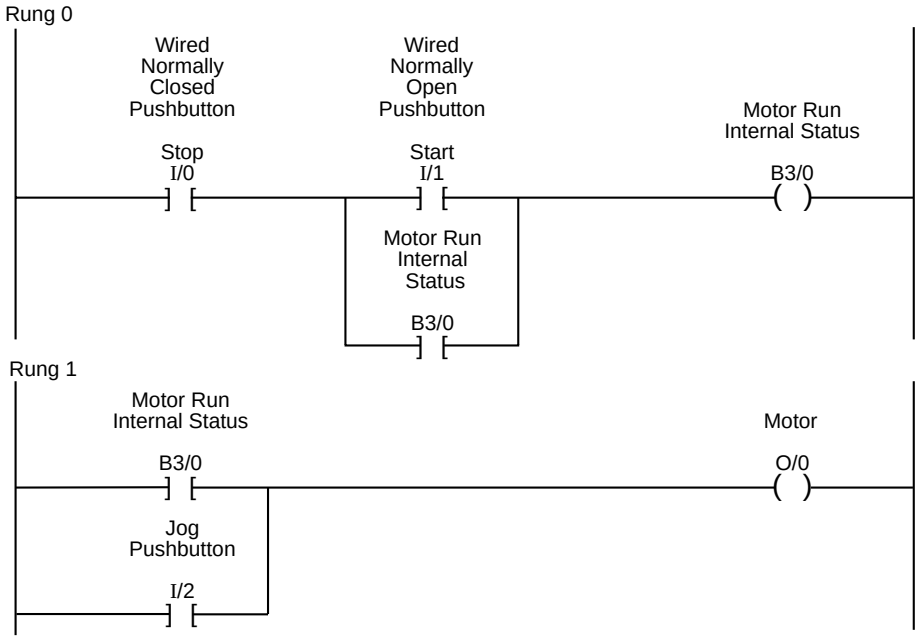
The logic used to perform this consists of two rungs with three conditional devices programmed in series and parallel:

RUNG 0

- This is the start/stop rung. It operates in the same manner as in the start/stop example in section 7.3, except that instead of energizing an external output address, internal bit B3/0 is energized when start pushbutton I/1 is pressed.

RUNG 1

- This is the rung that controls the actual output address O/0. If bit B3/0 has been energized in rung 0, the output is energized. If B3/0 has not been energized, the output can be jogged by pushing the jog pushbutton I/2. Every time I/2 is pressed, motor O/0 turns On.



7.7

On Delay

Uses

This logic turns On a device after a programmed time delay.

Operation

The On delay can be programmed to delay activation of a control instruction/output device for a preset period of time.

Ladder Logic

The logic used in this application consists of three rungs:

RUNG 0

- This is the start/stop rung. It operates in the same manner as the start/stop example in section 7.3, however, instead of energizing an external output address, internal bit B3/0 is energized when start pushbutton I/0 is pressed.

RUNG 1

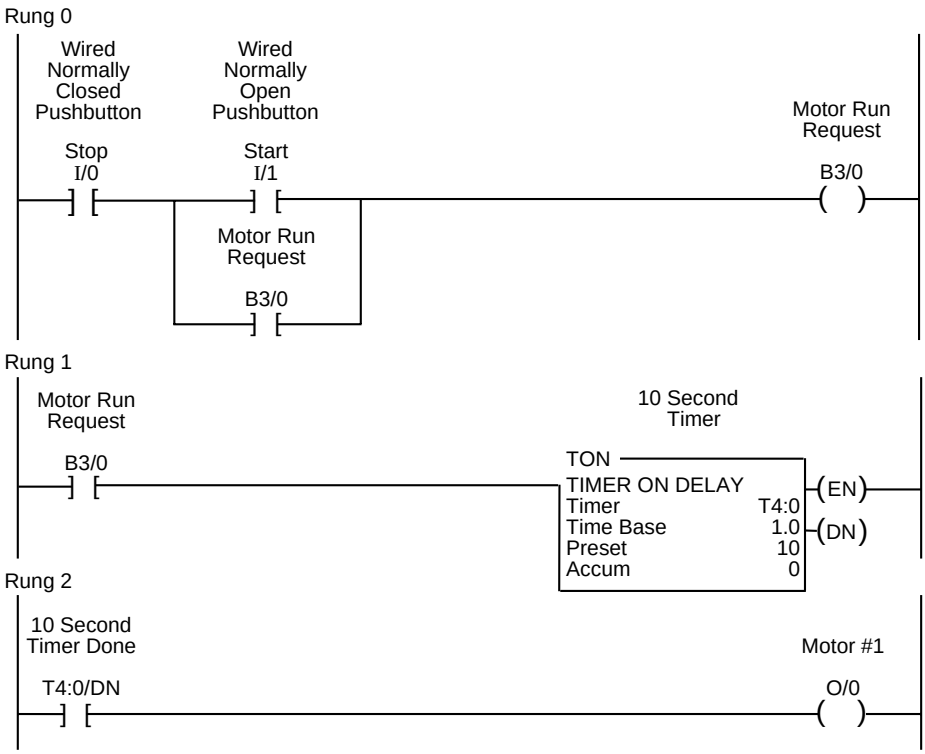
- This rung contains an On delay timer with an address of T4:0. When condition instruction B3/0 has been energized by the control instruction B3/0 in rung 0, the timer begins timing. Notice that the time base in the timer function block reads one second. This means that the timer will time in one second increments. Also notice that the preset value reads 10. This means that the timer will be done timing after a time delay of 10 one second increments, for a total of ten seconds. The timer done bit T4:0/DN in rung 2 will be energized at this point. If at any time rung 1 lacks logical continuity (B3/0 is Off), the timer will reset to zero.

The length of the time delay can be adjusted by changing the preset value. In addition, most PLCs allow the option of changing the time base, or resolution of the timer. The smaller the time base selected, the better the accuracy of the timer. Typical time bases are 0.01, 0.1, and 1.0 second.

The accumulated value of the timer (shown as ACCUM in the function block) is the number of increments the timer has accumulated since it began timing.

RUNG 2

- This is the rung that controls the actual output address O/0. If the timer has timed for 10 seconds (the timer done bit T4:0/DN is energized), the output O/0 is energized.



7.8

Off Delay

Uses

This logic turns Off a device after a programmed time delay.

Operation

The Off delay program allows a control instruction/output device to be turned Off after a preset amount of time.

Ladder Logic

The key item in these rungs is the normally closed condition instruction programmed in series with the control instruction on the first rung.

RUNG 0

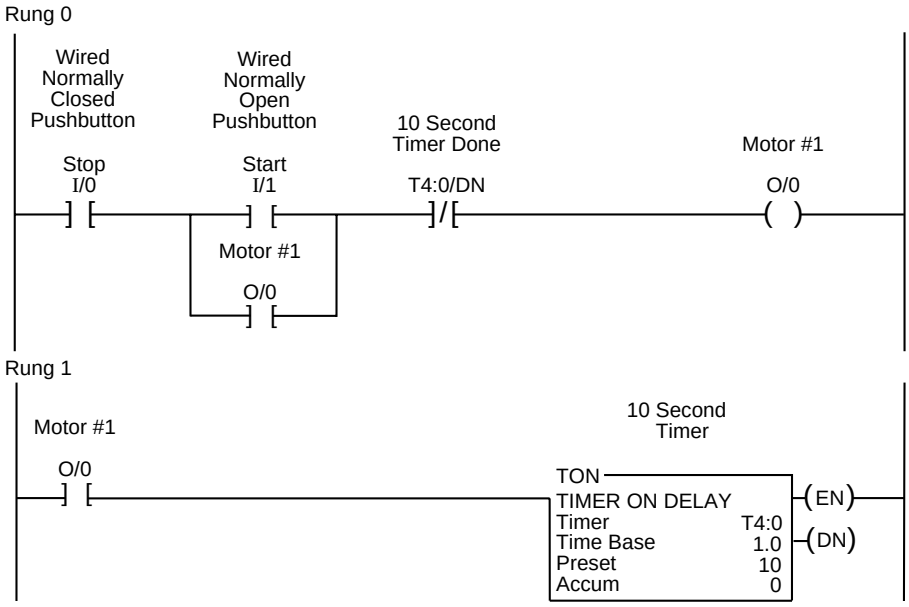
- This is the rung that controls the actual output address O/0. It operates in the same manner as the start/stop example in section 7.3. Notice that a normally closed instruction has been added in series with the output. This condition instruction has the address of the timer done bit, T4:0/DN from the timer in rung 1. It is the addition of this instruction that creates the Off delay operation of the rung.

RUNG 1

- This rung contains an On delay timer with an address of T4:0. When the output O/0 from rung 0 has been energized the timer begins timing. Notice that the time base in the timer function block reads one second. This means that the timer will time in one second increments. Also notice that the preset value reads 10. This means

that the timer will be done timing after 10 one second increments have passed–for a total delay of ten seconds. The timer done bit T4:0/DN will be energized at this point. This will de-energize the normally closed instruction T4:0/DN in rung 0, turning Off the output.

See the example in section 7.7 for a more thorough description of timer operation.



7

7.9

One Minute Clock

Uses

This is an example of a repetitive or free running clock.

Operation

In this example, the clock interval is set for 1 minute, but any interval could be selected. If a different time interval is required, simply change the value in the preset location in the timer function block. As discussed in the sections on On and Off Delays, the “resolution” of the clock will be determined by its time base. In this example, the timer is programmed with a 1-second time base, so the timer will only be capable of timing accuracies greater than or equal to 1 second. If a more accurate time is required, then use a timer with a time base less than 1 second.

Ladder Logic

The logic used in this application consists of 2 rungs:

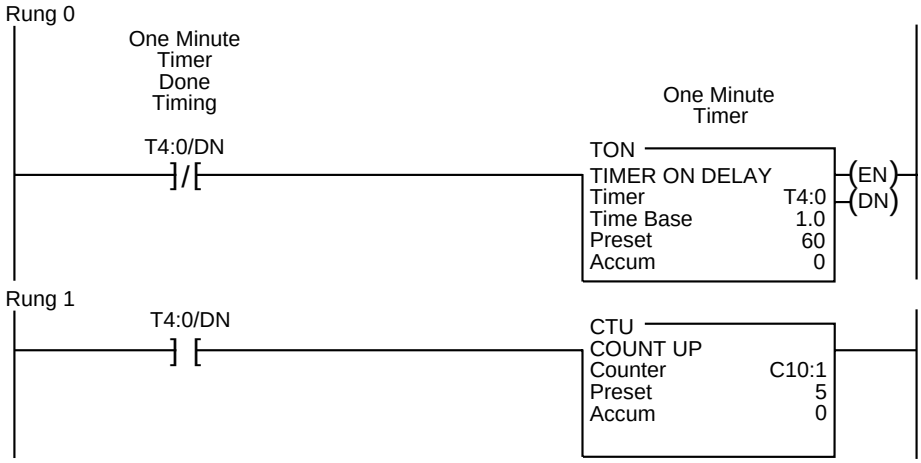
RUNG 0

- This is the timing rung. Notice that the condition instruction that controls the timer is the done bit of the timer, address T4:0/DN. Since this condition instruction is normally closed, it will have logical continuity when the timer is not done, that is, when the accumulated value is less than the preset value. Therefore, prior to the preset time being reached, the timer times.

Once the preset value is reached, the normally closed instruction becomes False and the timer resets to zero on the next scan of the program. The normally closed instruction is now True, and the timer begins timing from zero.

RUNG 1

- This rung contains a counter instruction. The condition instruction that controls this counter is the done bit from the timer in the previous rung. In this case it is a normally open instruction. As soon as the accumulated time of the timer in rung 0 reaches 60 seconds (the preset value of 60, using a time base of 1 second), the done bit energizes and increments the counter.



Retentive Timers

Timers are available that retain their time when the conditions preceding the timer instruction are False (open). Retentive timers are very useful for keeping track of the amount of time a device has been On. This can be very helpful for tracking device maintenance or other run-time type requirements. Retentive timers are reset using a separate instruction that is used to clear a timer. The instruction is called reset (RES), and is programmed as a control instruction.

7.10

Up/Down Counting

Uses

Up/Down counters are often used to monitor and track materials in conveying/packaging systems. An example is a bottle labeling application where the bottle making machine produces bottles at a greater rate than the labeling machine can apply labels. One method for compensating for the difference in production rates is to add a buffer area where the bottles can stack up to await labeling.

Operation

A counter is used to track how many bottles are in the buffer. The counter increments its count when a bottle enters the holding area from the bottle making machine, and decrements each time a bottle exits the holding area.

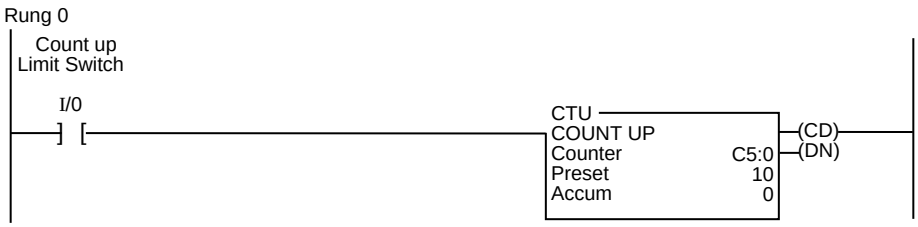
When the holding area is full, a signal can be sent to the bottle making machine to stop producing bottles.

Ladder Logic

The logic used in this application consists of 4 rungs:

RUNG 0

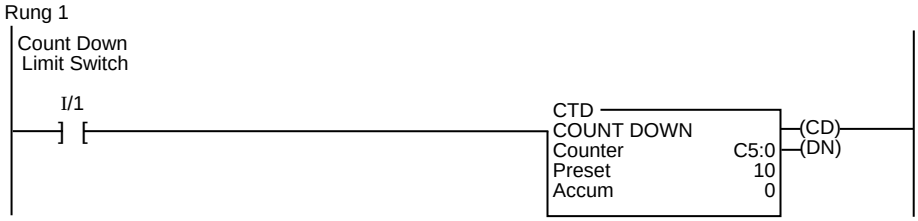
- This rung contains a count up instruction with an address of C5:0. Each time the limit switch wired to the input terminal I/0 is activated, condition instruction I/0 changes from False to True, and the counter increments by one count.



RUNG 1

- This rung contains the count down instruction. Notice that it has the same address as the count up instruction in rung 0, C5:0. Each time the limit switch wired to terminal I/1 is activated, condition instruction I/1 is made True, and the counter decrements by one count.

  It is important to note that any number of condition instructions can be on the rung that controls a counter instruction. Anytime the status of the rung goes from False to True, an up counter instruction will increment, and a down counter will decrement by one count.



RUNG 2

- This is the rung that controls the output O/0. When the number of counts accumulated in the counter equals or exceeds the counter's preset value, the done bit C5:0/DN is energized, turning On output O/0.

RUNG 3

- This is the reset rung. When the condition instruction I/2 comes On, the accumulated value of counter C5:0 is reset to zero.

7.11

Moving Data

Uses

One of the most useful and versatile features a PLC has is its ability to move and manipulate data. This ability turns the PLC into a powerful processing platform, capable of changing data values in integer files, timers, counters, stacks and many other areas. Moving data is done for control purposes, or to simply better organize information.

Operation

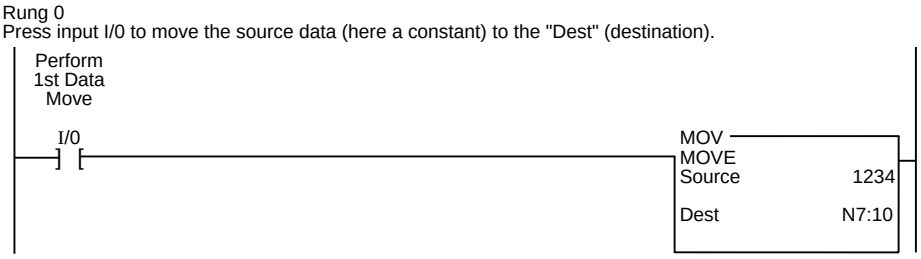
To move data in a PLC is a simple command: Move data from point A to point B. The structure is easy to understand and troubleshoot.

Ladder Logic

The logic used in this application consists of 4 rungs; the first three rungs illustrate actual move commands, while the third is used to clear one of the destination registers.

RUNG 0

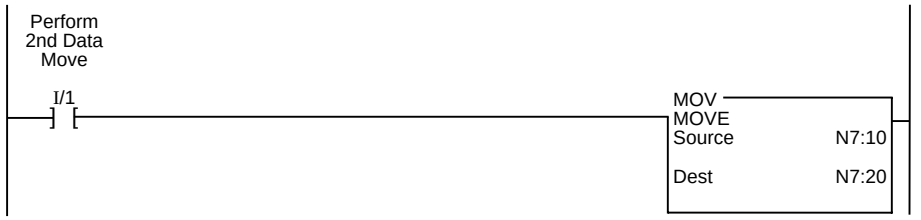
- This rung demonstrates moving a constant to an integer location. Whenever condition instruction I/0 is energized, the PLC will move the data (1234) in the “Source” location to the “Dest” (destination) location (Integer location N7:10). An integer location is a specific word where the data is stored. The data in the source location may be either a constant or an address internal to the PLC.



RUNG 1

- This rung demonstrates the moving of data from one integer location to another. Whenever condition instruction I/1 is energized, the data at N7:10 (Source) will be moved to N7:20 (Destination).

Rung 1
Move the source data (here a variable) into the destination.



RUNG 2

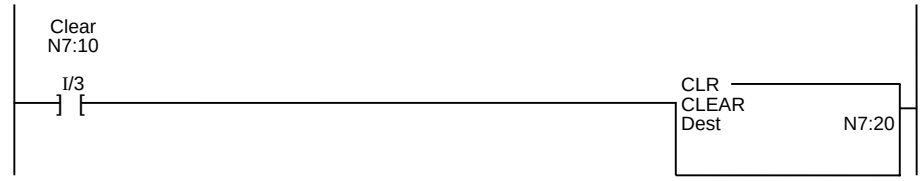
- This rung demonstrates the moving of data from an integer location to the preset value of a counter. Whenever input instruction I/2 is energized, the data at N7:20 (Source) will be moved to the counter preset C5:10.PRE (Destination).

Rung 2
Move the source data (here a variable) into the preset value of a counter instruction.

RUNG 3

- This rung is simply used to clear the data from the working register. Whenever condition instruction I/3 is energized, data is cleared from Destination N7:20.

Rung 3
Clear all data at the Destination (dest) address.



7.12

Comparing Data

Uses

PLCs can monitor and take action based on numerical values.

Operation

In many instances, devices may need to be controlled when they are less than, equal to or greater than other data values or set points used in the application, like timer and counter values. Comparison instructions are always programmed as condition instructions.

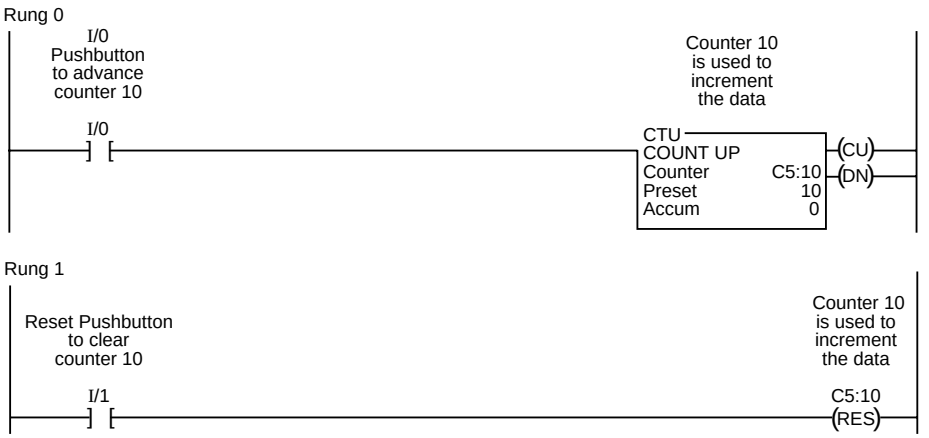
7

Ladder Logic

The logic used in this application consists of 6 sample rungs:

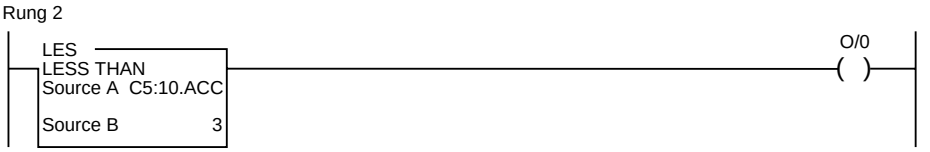
RUNG 0 and RUNG 1

- Rung 0 uses pushbutton I/0 to increment a counter (C5:10). Rung 1 uses pushbutton I/1 to reset the counter. These rungs simply setup some data values to use in the following rungs.



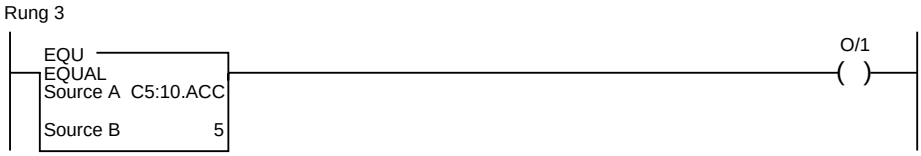
RUNG 2

- This rung contains a Less Than instruction. The “LES” will turn On the control instruction O/0 whenever the data in source A (the accumulated value of counter C5:10) is less than the data in source B, a constant, 3.



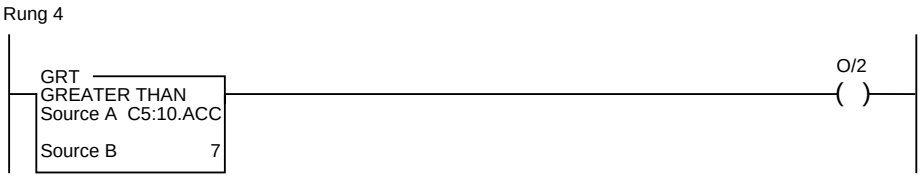
RUNG 3

- This rung contains an Equal instruction. The “EQU” will turn On the control instruction O/1 whenever the data in source A (the accumulated value of counter C5:10) is exactly the same as data in source B, a constant, 5.



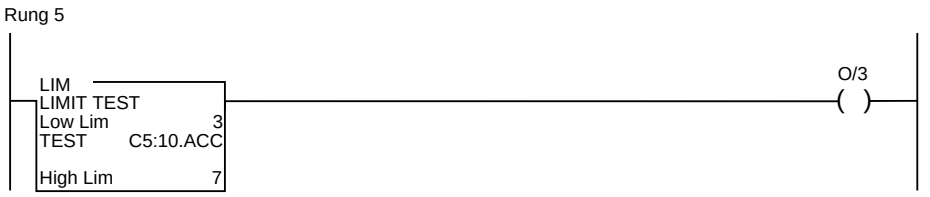
RUNG 4

- This rung contains a Greater Than instruction. The “GRT” will turn On the control instruction O/2 whenever the data in source A (the accumulated value of counter C5:10) is greater than the data in source B, a constant, 7.



RUNG 5

- This rung contains a Limit instruction. The “LIM” will turn On the control instruction O/3 whenever data in the “Test” position (the accumulated value of counter C5:10) is greater than the data in “Low Limit,” the constant, 3, and is less than the data in “High Limit,” the constant, 7.



7.13

Math Commands

Uses

Most PLCs on the market today offer a range of math capabilities. Some examples of the use of math include: combining parts counts, subtracting detected defects, calculating run rates, and logging or counting product.

Operation

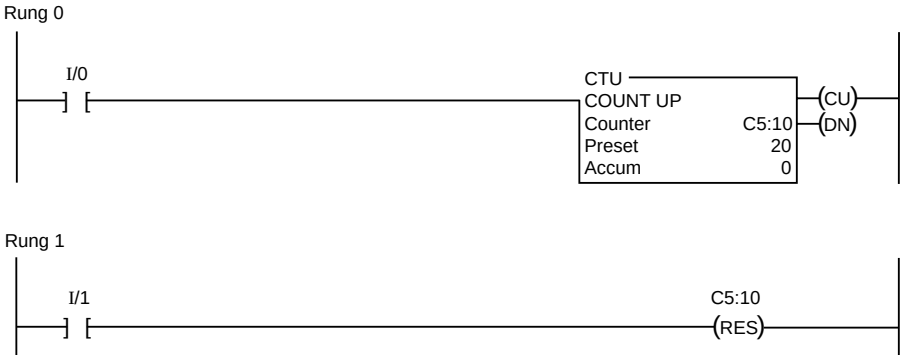
Math operations are performed as control instructions in the rung. Here, we have illustrated the program from a PLC that supports function block math commands. This type of math instruction is much easier to use than one that uses an accumulator for math operations.

Ladder Logic

The four basic math instructions are illustrated below:

RUNG 0 and RUNG 1

- These first two rungs make use of a counter to provide an easy method of changing a data value to be used in the math instructions to follow. Condition instruction I/0 will increment counter C5:10 each time it is energized. Condition instruction I/1 will reset the accumulated value of counter C5:10 when it is energized.

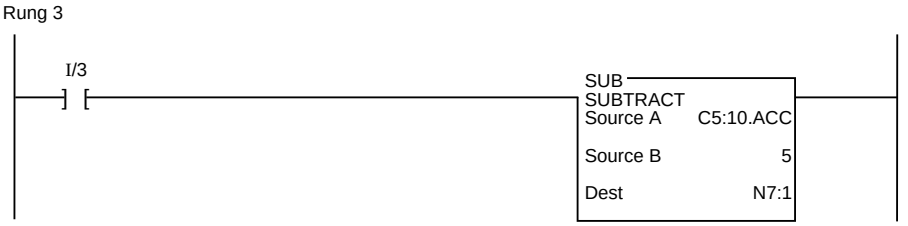


RUNG 2

- When condition instruction I/2 is on, the PLC will enable the add (ADD) instruction. In this example, the data in source A (in this case the constant, 5) will be added with the data in source B (the accumulated value of counter C5:10), with the result being placed in the Dest (destination), N7:0.

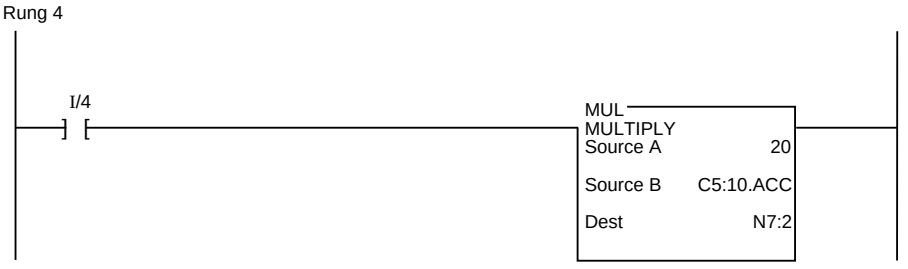
RUNG 3

- When condition instruction I/3 is on, the PLC will enable the subtraction (SUB) instruction. In this example, the data in source B (the constant, 5) will be subtracted from the data in source A (the accumulated value of counter C5:10), with the result being placed in the Dest (destination), N7:1.



RUNG 4

- When condition instruction I/4 is on, the PLC will enable the multiply (MUL) instruction. In this example, the data in source A (the constant, 20) will be multiplied by the data in source B (the accumulated value of counter C5:10), with the result being placed in the Dest (destination), N7:2.



RUNG 5

- When condition instruction I/5 is on, the PLC will enable the divide (DIV) instruction. In this example, the data in source A (the accumulated value of counter C5:10) will be divided by the data in source B (the constant, 2), with the result being placed in the Dest (destination), N7:3.

7.14

Sequencers

Uses

Many of the micro PLCs on the market today offer a command that replaces electromechanical devices called drum sequencers or drum switches. These electromechanical devices were designed for simple control systems that required specific “On” or “Off” patterns of outputs that are continuously repeated. A sequencer instruction can perform the same function as a drum switch, but with more flexibility. It is typically used for sequencing the operation of valves, solenoids or lights for many varieties of machines or processes.

7

Operation

Typically, these instructions take the form of a single high level instruction. A memory location is designated within the PLC that forms the “pattern” of the outputs during the sequence. The table below illustrates this architecture. (Fig. 7-14)

The bit data file (B3:0 through B3:3) contains the data for each step of the sequence controlled by the sequencer instruction. The bit patterns that are stored in each of these locations form the output pattern that will be seen for each of the sequencer steps.

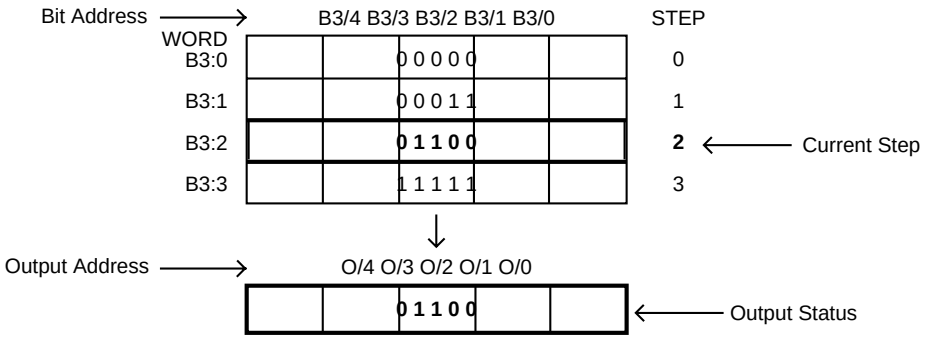


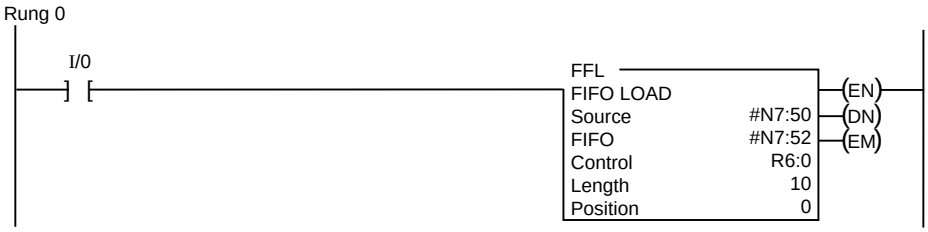
Fig. 7-14

Ladder Logic

RUNG 0

- The sequencer instruction typically looks like this: A single instruction that identifies where the output pattern data is stored (B3:0), the destination or address of that output data, and the length or number of steps of the sequence. This instruction also manages or tracks what the current sequencer position is. Each time the

conditional logic preceding the instruction changes from False to True, the sequencer will increment to the next step.



7.15

FIFO (First-In First-Out)

Uses

FIFOs are part of a special set of commands that deal with storing numeric data. These commands are primarily used in tracking products and materials during processes. An example would be an overhead conveyor system that feeds parts into a paint booth. Each part requires a different color, and the color ID is tracked while the part is moved through the manufacturing process. If the conveyor is running and a problem occurs in the paint booth, the parts need to be stored until the paint booth is back on-line.

One method is to have a holding area into which the PLC can redirect the parts. As each part is sent into the area, the color ID is loaded into a FIFO stack. When the paint booth returns to operation, the PLC will draw a part out of the holding area and track the identifier with it. This assures that the part will get the correct color of paint.

Operation

FIFO commands typically take the form of two high level instructions, FIFO Load (FFL) and FIFO Unload (FFU). These instructions are used in pairs. The FFL instruction loads words into a user-created group of registers called a FIFO stack. The FFU instruction unloads words from the FIFO stack in the same order as they were entered (Fig. 7-15).

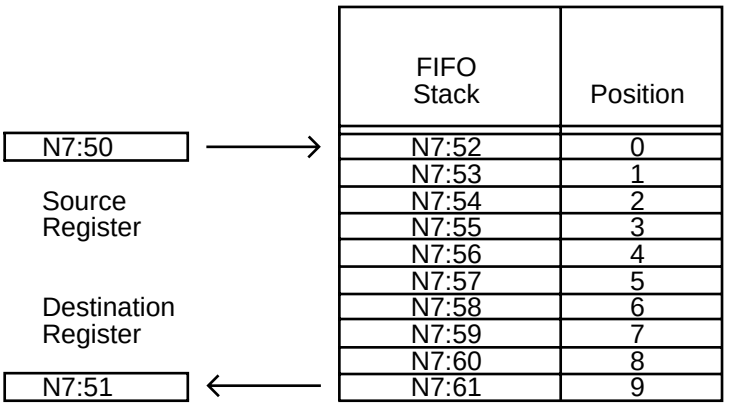


Fig. 7-15

The unique feature of the FIFO stack is its ability to manage where the data is. This is done by tracking where data is entered into the stack. The FIFO instructions manage all aspects of entering and removing data from the stack.

Ladder Logic

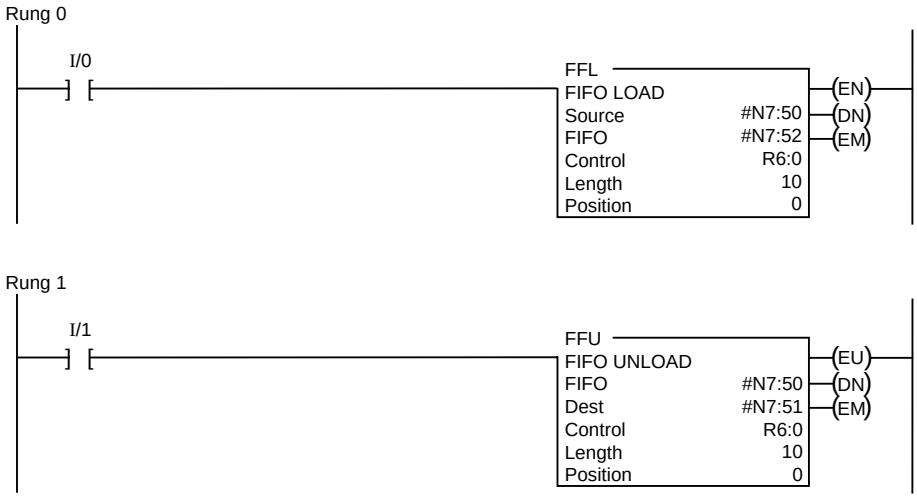
RUNG 0

- This rung controls the transfer of data to the FIFO stack. When the logic preceding the FIFO Load instruction changes from False to True, the data located in the source register N7:50 is stored in the next

available location in the stack. This location is designated by the current value of the position parameter of the instruction. As soon as the data is transferred, this position value will point to the next position in the stack. The size of the stack corresponds to the value programmed as the length parameter. In this example, the FIFO stack is 10 words long.

RUNG 1

- When the logic preceding the FIFO Unload instruction changes from False to True, data is retrieved from the stack. In other words, the “oldest” data (the first in) will be transferred to the destination register N7:51.



7

7.16

High-Speed Counter (HSC)

Uses

Many micro PLCs on the market today have the ability to detect and control high-speed operations. One of the most versatile features is a full function high-speed counter. This feature gives a micro PLC the ability to count a high-speed input signal and control the corresponding outputs based on the accumulated count, independent of the processor's scan. This capability allows micro PLCs to be used in applications that previously required much larger PLCs. Some of the micro PLCs available today even have the ability to modify what outputs will be controlled during the HSC's operation. This functionality allows the micro PLC to replace cut-to-length controllers, rotary cam switches, programmable limit switches and other mechanical devices.

Operation

Many of the micro PLCs on the market that are capable of HSC operation have a number of modes for specific types of operation (Up, Down, Up/Down, Quadrature, Quadrature With External Hold & Reset, etc.). In addition, many also support special commands that are intended to be used with the HSC, such as high-speed compares, resets, updates, etc.

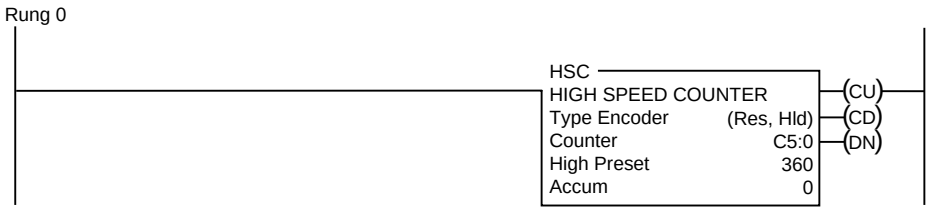
Ladder Logic

RUNG 0

- The High-Speed Counter instruction (HSC) in this rung contains all of the parameters required to define its mode of operation. These

parameters are selected when the instruction is programmed. A separate High-Speed Counter Load instruction (not shown here) is required if outputs/control instructions are to be controlled directly by this high-speed function.

The operation of a high-speed counter instruction is very specific to the micro PLC used. Consult the controller's user manual before using the instruction.



7.17

Two Stage Alternator

Uses

This type of logic is used to alternate devices (typically pumps) in applications like the emptying of wells, reservoirs, and vessels (tanks) where the rate of flow into the tank is not constant.

Operation

In an application like this, two smaller pumps are frequently used instead of one large one. Alternating pump operation (Pump 1 as the primary, then pump 2 as the primary) reduces the maintenance required on the individual devices and provides more reliable operation.

In addition, the secondary or “standby” pump is available if the rate of water entering the vessel is more than the first pump can handle. If this situation occurs, the second pump will also turn On and assist the primary pump. The triggers for these events could be analog signals, or simple discrete inputs (float switches, etc.). This illustration shows a typical application with float switches in a tank (Fig 7-17).

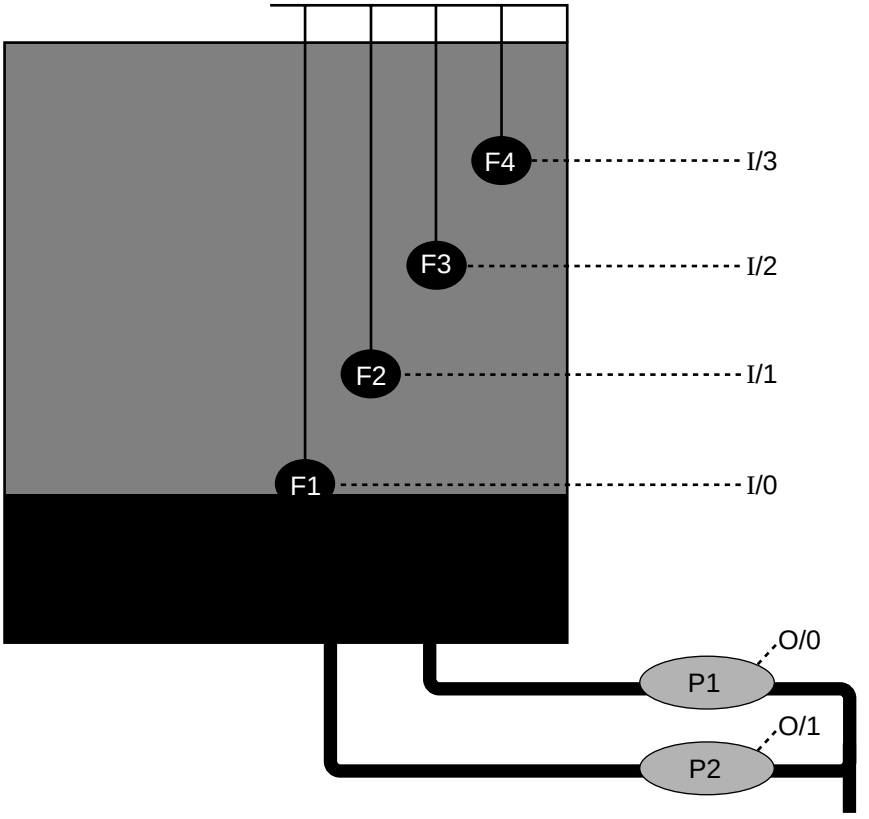


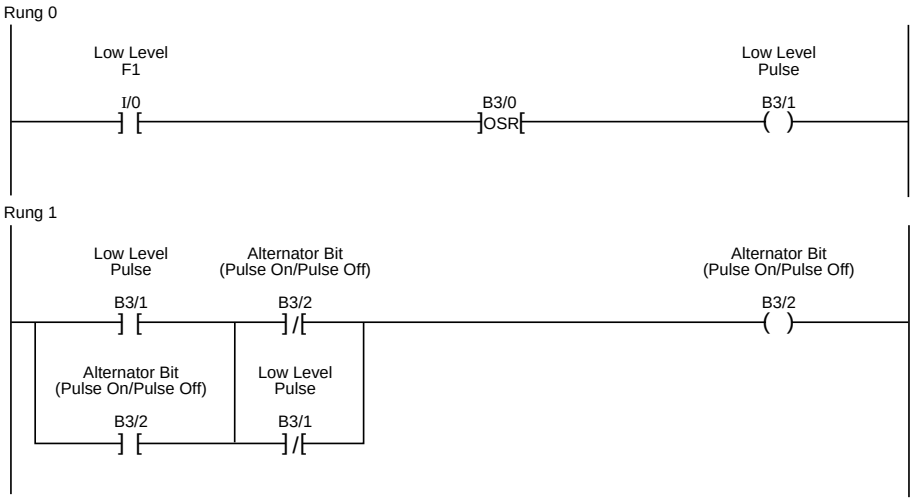
Fig. 7-17

Ladder Logic

The logic used in this application consists of 4 rungs:

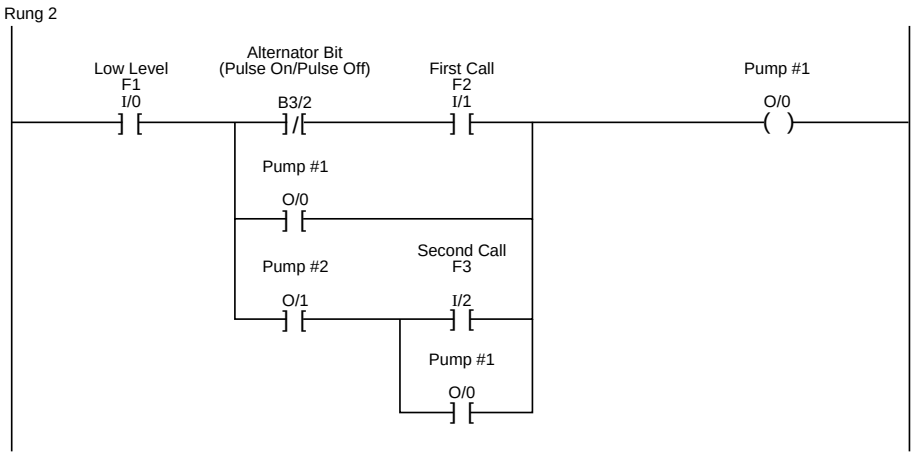
RUNG 0 AND RUNG 1

- These two rungs form a flip/flop circuit as described in the example in section 7.4. Each time the fluid in the tank reaches the low level float switch F1 (I/0), the alternator bit in rung 1, B3/2 changes state. The status of this bit determines which pump will be the first to turn On.



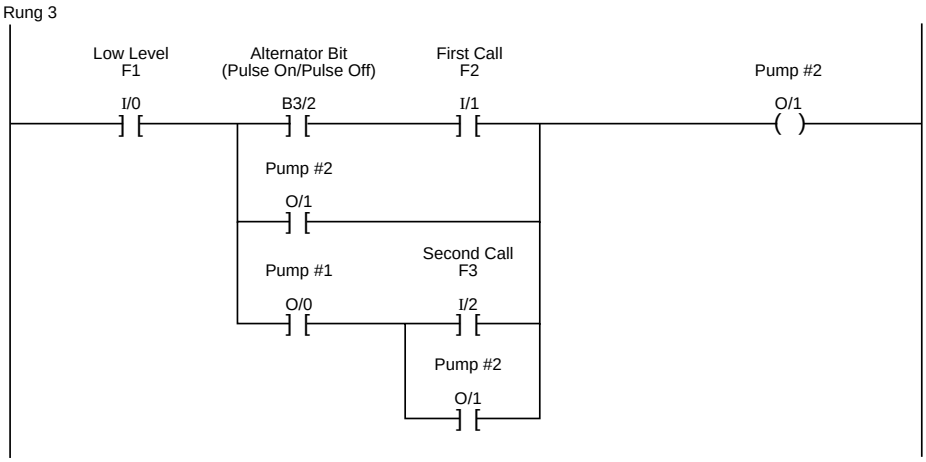
RUNG 2

- This rung controls the operation of pump 1, O/0. If the low float I/0 is On and the alternator bit B3/2 is Off, and the level in the tank has reached the First Call float F2 (I/1), this pump will be the first one energized. If B3/2 is On, pump 1 will be the second pump energized.



RUNG 3

- This rung controls the operation of pump 2, O/1. If the low float I/0 is On and the alternator bit B3/2 is On, and the level in the tank has reached the First Call float F2 (I/1), this pump will be the first one energized. If B3/2 is Off, pump 1 will be the second pump energized.



One of the powerful features of PLCs is the ability to monitor and alert operators to alarm conditions. You may have noticed that float switch #4 (F4) is not being used in the program. This float switch is an alarm condition. It can be used in the program to make sure the pumps are running if this float is tripped. This "check" operation would help minimize damage if the level 1 or level 2 floats malfunctioned. It can also sound an alarm that indicates the tank is about to overflow.

7.18

Three Station Alternator

Uses

This example is similar in function to the example in section 7.17, except that we are adding an additional device to alternate – 3 rather than 2. For ease of description, we will discuss three pumps that empty a tank. The control system needs to be able to rotate the pump that

turns On first each time a request is made, and also to bring other pumps on-line as demand increases.

Operation

A series of five float switches are used to monitor the level of fluid in the tank (Fig. 7-18). The control system monitors these float switches, and determines which pump is the primary pump, lag pump 1 and lag pump 2.

Input Device Status	Pump Requirements
Float Switch 1 Off	All pumps off
Float Switch 1 On	None
Float Switches 1, 2 On	Primary pump On
Float Switches 1, 2 & 3 On	Primary and Lag #1 pumps On
Float Switches 1, 2, 3 & 4 On	Primary, Lag #1 and Lag #2 pumps On
Float Switches 1, 2, 3, 4 & 5 On	ALARM condition

Whenever the primary pump is needed (called), the control system will then rotate the assignment of the primary pump. This ensures even wear between all three pumps and verifies that each pump is operational. As each pump is designated as the primary, the remaining lag pumps will also be rotated.

A breakdown of priorities for each pump at any given time is included here. The sequence for the running of each pump is called a stage. There are three pumps, and therefore three stages that operate as follows:

Stage	Pump 1	Pump 2	Pump 3
1	Primary	Lag 1	Lag 2
2	Lag 2	Primary	Lag 1
3	Lag 1	Lag 2	Primary

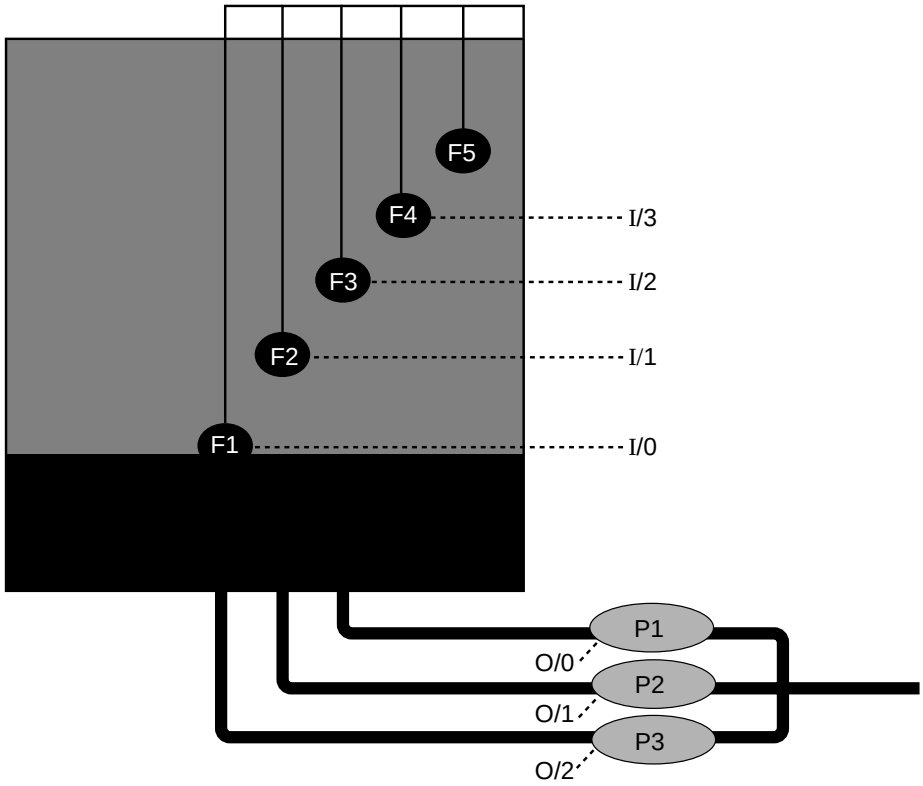


Fig. 7-18

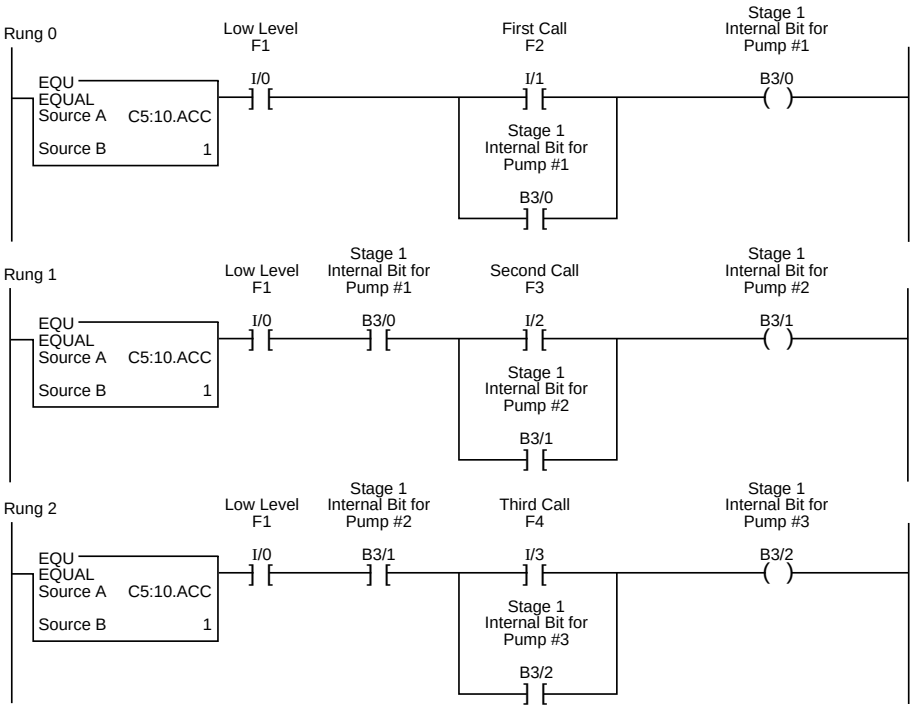
7

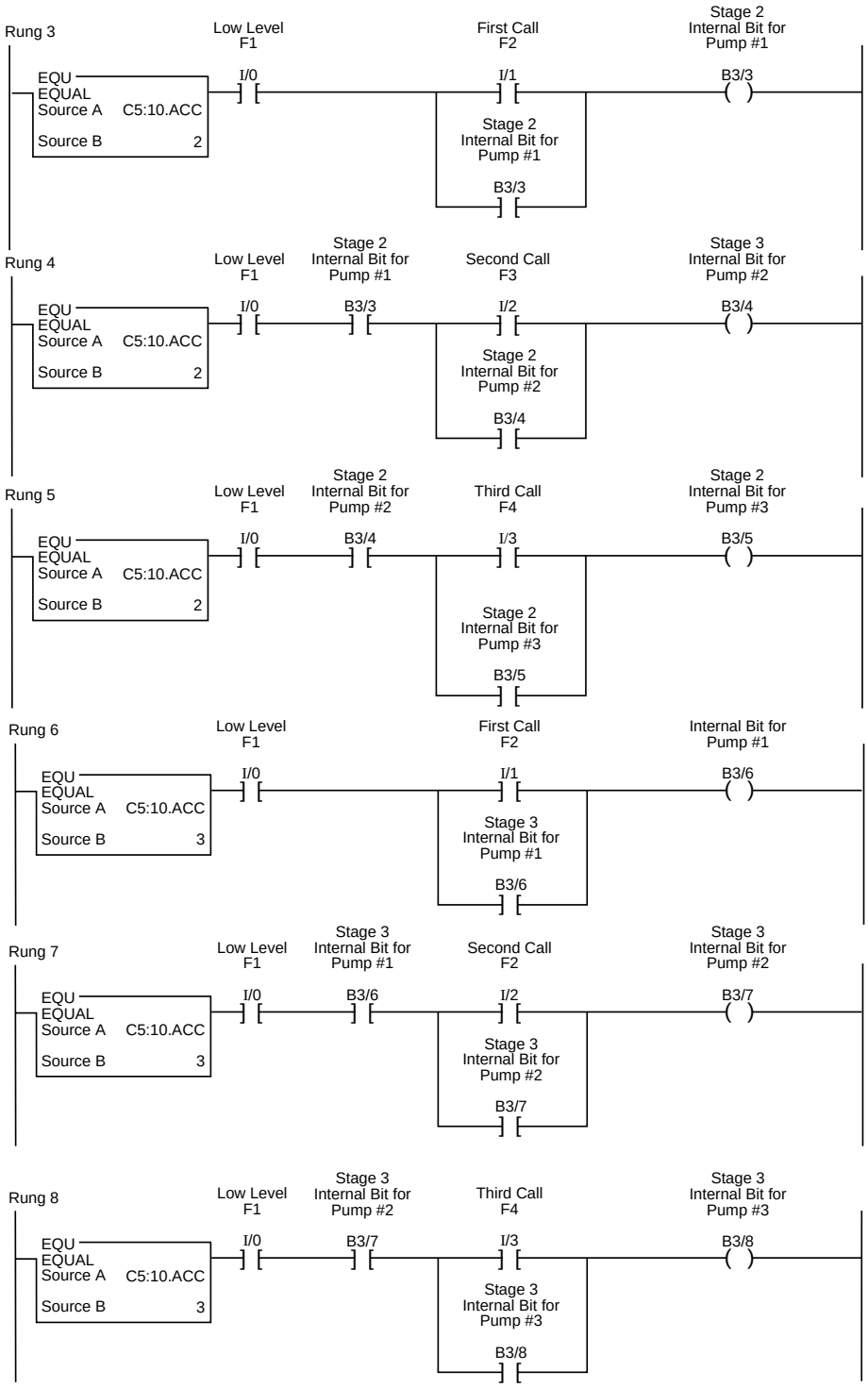
Ladder Logic

The logic used in this application consists of 15 rungs. The EQUAL TO comparison instruction at the start of the first ten rungs compares the accumulated value of the counter in rung 13 to a constant. The value of the constant designates which stage is to be run (i.e., the operating sequence of the pumps).

RUNGS 0 through 8

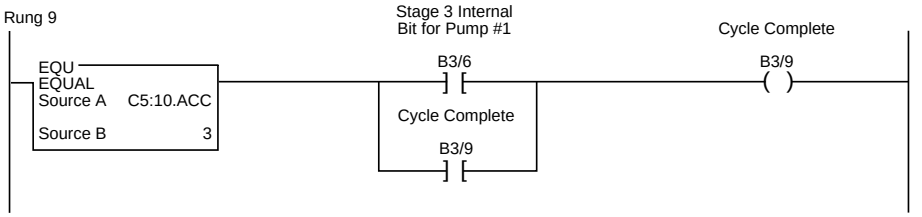
- The first 9 rungs in the program set the priority assignment for the primary, Lag #1 and Lag #2 pumps.




Rung 3
Low Level
F1
First Call
F2
Stage 2
Internal Bit for
Pump #1
EQU
EQUAL
Source A C5:10.ACC
Source B 2
I/0
I/1
B3/3
Stage 2
Internal Bit for
Pump #1
B3/3
Rung 4
Low Level
F1
Stage 2
Internal Bit for
Pump #1
Second Call
F3
Stage 3
Internal Bit for
Pump #2
EQU
EQUAL
Source A C5:10.ACC
Source B 2
I/0
B3/3
I/2
B3/4
Stage 2
Internal Bit for
Pump #2
B3/4
Rung 5
Low Level
F1
Stage 2
Internal Bit for
Pump #2
Third Call
F4
Stage 2
Internal Bit for
Pump #3
EQU
EQUAL
Source A C5:10.ACC
Source B 2
I/0
B3/4
I/3
B3/5
Stage 2
Internal Bit for
Pump #3
B3/5
Rung 6
Low Level
F1
First Call
F2
Internal Bit for
Pump #1
EQU
EQUAL
Source A C5:10.ACC
Source B 3
I/0
I/1
B3/6
Stage 3
Internal Bit for
Pump #1
B3/6
Rung 7
Low Level
F1
Stage 3
Internal Bit for
Pump #1
Second Call
F2
Stage 3
Internal Bit for
Pump #2
EQU
EQUAL
Source A C5:10.ACC
Source B 3
I/0
B3/6
I/2
B3/7
Stage 3
Internal Bit for
Pump #2
B3/7
Rung 8
Low Level
F1
Stage 3
Internal Bit for
Pump #2
Third Call
F4
Stage 3
Internal Bit for
Pump #3
EQU
EQUAL
Source A C5:10.ACC
Source B 3
I/0
B3/7
I/3
B3/8
Stage 3
Internal Bit for
Pump #3
B3/8

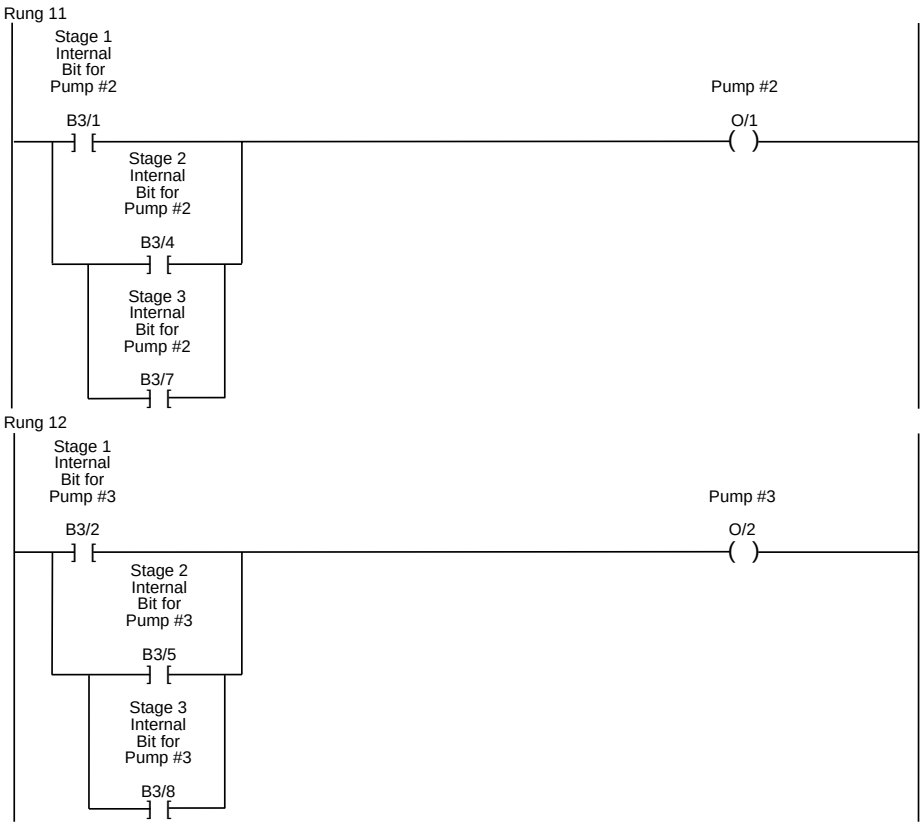
RUNG 9

- This rung sets internal bit B3/9 when the final stage has been completed.



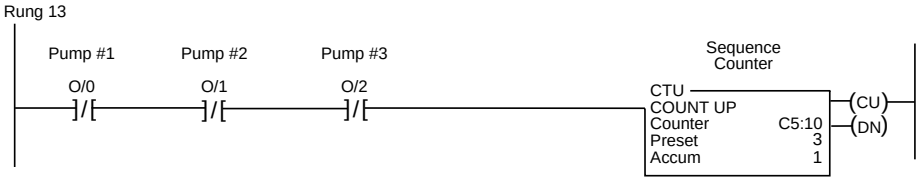
RUNGS 10, 11, 12

- These three rungs link the preceding rungs to actual output terminals on the PLC. Making use of internal bits for logic purposes provides an easy method of controlling an output from multiple sources within a program.



RUNG 13

- This rung is the counter rung and controls which stage will be run next. The counter increments each time all pumps are off.



RUNG 14

- This rung resets the counter after the last stage is run and starts the entire sequence over again.

Notice that float switch F5 is not used in this program. It can be used in the program as an alarm condition that ensures all pumps are running if this float is tripped. This would help prevent damage if any of the other floats malfunctioned. Or, it can sound an alarm to identify a problem with the tank, for instance, that it is about to overflow.

Appendices

Glossary . . . . . Appendix A
Input and Output Devices . . . . . Appendix B
Instruction Execution Times . . . . . Appendix C
Sample Program Worksheets . . . . . Appendix D

A

Appendix A–Glossary

address: A unique memory location, identified by an alphanumeric character. For example, I/2 is the memory address for data located in bit 2 of the input file.

alphanumeric: Character strings composed of any combination of letters or numbers.

analog: A numeric value that represents measurable quantities, such as temperature, weight, pressure, etc. Compare with **digital**.

AND: A Boolean operation that produces a True output only when all conditions are True, and a False output if any condition is False.

application: A machine or process that requires a control system for operation.

application memory: The portion of the total system memory dedicated to storage of the application program and associated data.

BASIC: Beginner's All-Purpose Symbolic Instruction Code–a versatile, easy to learn computer language, commonly used for simple programming tasks.

battery backup: A battery or set of batteries that provide power to maintain the contents of processor memory in case of a system power outage. Note: Processors utilizing EEPROM memory typically do not require battery backup.

BCD: Binary Coded Decimal–A binary system in which each decimal digit from 0 to 9 is represented by four binary digits (bits). A thumbwheel switch is usually a BCD device, and when connected to a programmable controller, each decade, or decimal place, requires four wires.

A

binary: A numbering system using only the digits 0 and 1. Also called base 2.

bit: The smallest storage location in memory. A bit contains either a 1 (On/True) or a 0 (Off/False).

Boolean operators: Logical operators such as AND, and OR, that can be used singly or in combination to form logical statements or circuits. These statements must have an output response which is either True (1) or False (0).

branch: A parallel logic path within a ladder logic rung.

bus: 1) A group of lines used for data transmission or control. 2) Power distribution conductors.

byte: A group of adjacent bits usually operated upon as one unit, such as when moving to and from memory. There are eight bits in one byte. A byte is capable of storing and displaying a numeric equivalent between 0 and 255.

C: A computer system programming language initially developed for the UNIX operating system.

communication scan: A part of the PLC's operating cycle that manages communication with other devices, such as a hand-held programmer. See also **input scan, output scan, and program scan.**

condition instruction: Instruction pertaining to the input portion of a rung on a ladder diagram. It is the condition or status of these instructions that determine how the control instruction is to be controlled. See also **control instruction.**

contact: 1) One of the conducting parts of a connector, switch, or relay that are engaged or disengaged to open or close an electrical pathway. 2) With reference to PLC ladder logic programs: a condition that provides a logical pathway (continuity) when True.

contact symbology: A set of symbols used to express logic (the control program) using conventional relay symbols. For instance, -] [- indicates a normally open contact, -]/[- indicates a normally closed contact, and -( )- indicates a relay coil or output.

A

continuity: Having the capability of passing a voltage, logic state, or any other signal unimpeded.

control instruction: Instruction pertaining to the output portion of a rung on a ladder diagram. These commands detail exchanges of data with external output devices or internal devices such as timers, counters, math functions or other high-level instructions. See also **condition instruction**.

controller: A device capable of controlling other devices. For example, a programmable controller is used to monitor input devices, implement logic, and control output devices.

counter: A device or software instruction that counts the occurrence of some event. It may be pulses resulting from operations such as switch closures, or other discrete events.

CPU: Central Processing Unit–The decision-making section of a programmable controller that executes the instructions contained in the user program.

CSA: Canadian Standards Association–An agency which regulates the specifications and testing required of electrical devices used in Canada.

cycle: A single sequence of operation. In the PLC, one full operating scan from start to finish.

data: Within the PLC, a general term for any type of information stored in memory.

data table: The part of the PLC memory that contains I/O values and files where data is monitored, manipulated, and changed for control purposes.

debug: The process of locating the source of control system malfunctions and correcting the problems.

diagnostics: The detection and indication of errors or malfunctions.

digital: Information presented as a discrete value; 1 or 0. Compare with **analog**.

A

drum timer: A mechanical device, which controls a sequential operation by means of a drum with pegs, where the presence of a peg represents a logical “1,” and the absence of a peg represents a logical “0.” Its operation is similar to that of a music box mechanism.

EEPROM: Electrically Erasable Programmable Read-Only Memory–A type of PROM that is programmed and erased by electrical pulses. Data stored to a EEPROM will not be erased just by interrupting power to the chip.

EIA: Electronic Industries Association–An agency which sets electrical/electronic standards. See also **RS-232**.

EMI: ElectroMagnetic Interference–Magnetic fields generated by electrical devices.

execution time: The time required to perform one specific instruction, a series of instructions, or a complete program. The execution time for a given instruction may vary depending on the status of the instruction (True or False) and other parameters.

False: The status of an instruction that does not provide logical continuity on a ladder rung.

fault: Any malfunction that interferes with the normal operation of an application.

FET: Field Effect Transistor–A high-performance, solid state device capable of switching higher current dc loads than transistors.

FIFO (First-In First-Out): The order in which data is entered into and retrieved from a file. See also **LIFO (Last-In First-Out)**.

force: Software function that allows the programmer to energize or de-energize an input or output independent of the program logic. It is used primarily for troubleshooting.

hardware: Includes all the physical components of the control system, including the programmable controller, peripherals and interconnecting wiring. Compare with **software.**

A

IEC: International Electrotechnical Commission–An international association with members representing electrical manufacturers. The IEC establishes standards for the construction and operation of electrical devices.

I/O (Inputs and Outputs): Consists of devices that provide data to (input), and receive data from (output) the PLC.

input device: A device, such as a pushbutton, sensor, or a switch of some sort, that supplies signals to the PLC.

input scan: Part of the controller's operating cycle. During the input scan, the controller examines all input devices for an On or Off state. This status is temporarily written to the "input image" memory file for use during the program scan. See also **communication scan, program scan, and output scan.**

instruction: A command defining an operation to be performed by the controller. A rung in a program consists of a set of condition (input) instructions and control (output) instructions. See also **condition instruction** and **control instruction.**

IP: Ingress Protection–A designation code defined by IEC publication number 529 specifying the level of resistance an enclosure exhibits towards penetration by objects, dust or water.

ladder logic: A PLC program written in a format resembling an electrical ladder diagram. The program is used by a programmable controller to sense inputs and control output devices.

latch: A ladder program output instruction that retains its state even though the conditions that caused it to latch On may go Off. A latched output must be unlatched to turn Off. A latched output will retain its last state (On or Off) if power is removed.

LED: Light Emitting Diode–A semiconductor diode, the junction of which emits light when passing a current. LEDs are used as diagnostic indicators on various PLC hardware components.

LIFO (Last-In First-Out): The order in which data is entered into and retrieved from a file. See also **FIFO (First-In First-Out).**

limit switch: An electrical switching device that is actuated by some part and/or motion of a machine or equipment.

logic: A process of solving complex problems through the repeated use of simple functions that can be either True or False. It is a general term for digital circuits and programmed instructions designed to perform decision-making and computational functions.

Master Control Relay (MCR): A hard-wired relay that can be de-energized by one of any number of series-connected emergency stop switches. Whenever the master control relay is de-energized, its contacts open to de-energize all application I/O devices.

memory: The part of the controller where programs and data are stored.

mnemonic: An easy to remember term that is used to represent a complex or lengthy set of information.

modular controller: Programmable controller in which the power supply, processor, and I/O interfaces reside in separate units, or modules. Compare with **packaged controller.**

NEMA Standards: Standards for the performance and construction of electrical equipment that have been agreed upon and approved by the members of the National Electrical Manufacturer's Association (NEMA).

normally closed contact: A switch or relay contact pair that is closed when the switch or the coil of the relay is not activated, and open when the switch mechanism or coil is activated. Compare with **normally open contact.**

normally closed instruction: A ladder program symbol that will allow logical continuity (flow) if the referenced address is Off. Compare with **normally open instruction**.

A

normally open contact: A switch or relay contact pair that is open when the switch or the coil of the relay is not activated, and closed when the switch mechanism or coil is activated. Compare with **normally closed contact.**

normally open instruction: A ladder program symbol that will allow logical continuity (flow) if the referenced address is On. Compare with **normally closed instruction.**

one-shot: A programming instruction that turns On a bit for a single program scan.

operating voltage: For inputs, the voltage range needed for the input to be in the On state. For outputs, the allowable range for user-supplied voltage. The PLC or other control system itself will have a specified range of allowable voltage for system operation.

OR: A logical operation that produces a True output when one of any number of conditions is True, and a False output if all conditions are False.

output device: A device, such as a pilot light or a motor starter coil, that is controlled by the PLC.

output scan: A part of the controller's operating cycle. Using information obtained during the program scan about the status of the output devices, the controller energizes or de-energizes its output circuits to control output devices. See also **communication scan, input scan, and program scan.**

packaged controller: Programmable controller with the processor, power supply, inputs and outputs all in one package. Compare with **modular controller.**

peripheral: External devices that are connected via a communications port to the programmable controller, usually for programming, data exchange or operator interface.

power supply: Electrical circuit that filters, conditions and supplies appropriate voltages for system components and circuitry.

processor: A central processing unit. See also **CPU**.

program: A set of instructions stored in memory that are executed in a predetermined order by the central processing unit.

program scan: A part of the controller's operating cycle. During the program scan the ladder logic program is executed and the output data file is updated based on the logic of the program and the status of the input data file. See also **communication scan, input scan, and output scan.**

RAM: Random Access Memory–A fast, volatile (when power is interrupted, data is lost) form of memory. Each bit in RAM can be stored or retrieved in the same amount of time, at any time. Commonly referred to as read/write memory because it can be written to as well as read from. This type of memory typically uses a battery or capacitor for back up power.

read: To acquire data from a memory location. For example, the controller reads information from the input data file to solve the program.

register: A temporary storage space for various types of information and data, such as timer or counter values. In PLCs, a register is normally 16 bits wide (1 word).

relay: An electrically operated mechanical device, the contacts of which open and close based on the presence of an electrical signal.

relay logic: A program written with relay symbols (contacts and coils). Relay logic is commonly referred to as contact symbology.

retentive data: Information (data) stored in memory that is not lost when power is interrupted.

RS-232: An EIA standard that specifies electrical and mechanical characteristics for serial binary communications. It is a single-ended serial communication interface.

rung: Ladder logic is comprised of a set of rungs. A rung contains condition (input) and control (output) instructions.

A

SBC: Single Board Controller–A custom control solution using a proprietary electronic circuit board designed to control one specific application.

scan time: The time required to read all inputs, execute the control program, and update all outputs.

sequencing: Using a software device to initiate or terminate events in a desired sequence.

solenoid: A device that transforms electrical current into linear (mechanical) motion; it consists of one or more electromagnets that move a metal plunger. The plunger is sometimes returned to its original position after excursion with a spring or permanent magnet.

solid state: Circuitry designed using only integrated circuits, transistors, diodes, etc.; no relays or other electromechanical devices are used.

software: 1) The ladder logic program stored in the PLC. 2) Executable programming package used to develop ladder logic programs. Compare with **hardware**.

system: A set of one or more PLCs that, together with I/O devices, computers, associated software, peripherals, terminals and communications networks, provide a means of performing information processing for the control of machines or processes.

system memory: The total memory space within the controller, including the user program, data and the operating system.

terminal: A point on a PLC where external I/O devices, such as a pushbutton or pilot light, are wired.

throughput: The amount of time it takes to sense an input and energize the corresponding output.

thumbwheel switch: A rotary switch used to input numerical information into a controller.

time base: The unit of time used by a timer to register events. A one second time base is accurate to the nearest second. Many controllers are capable of operating with .01 or .001 second time bases.

transistor: A solid state, electronic device that functions as an electrically controlled switch commonly used to control dc loads. A component of dc output circuits.

triac: A solid state, electronic device that functions as an electrically controlled switch for ac loads. A component of ac output circuits.

True: The status of an instruction that provides logical continuity on a ladder rung.

UL: Underwriters' Laboratories–An agency that recommends minimum specifications for the construction and operation of electrical equipment used in the United States. UL also tests equipment to determine adherence to those specifications.

watchdog timer: A timer that monitors the logical operations within the circuitry of the processor. If the timer ever times out, it indicates that there is a problem with the normal operation of the processor, and operation is terminated.

word: A unit of memory composed of 16 individual bits. Words or portions of words are used when programming instructions, or performing math operations.

write: To move or “copy” data to a memory location. For example, the controller writes the information to the output data file based on the logic of the ladder program.

B

Appendix B–
Sample Input and Output Devices

Input Devices

Input devices are field devices that act as information gatherers for the PLC. Think of them as the eyes and ears of the PLC. Most micro PLCs need to recognize a discrete (On or Off) signal. Input devices typically communicate with the PLC by switching current On or Off by either electromechanical or solid state contacts. Solid state input devices like transistors, FETs and triacs are sensitive to input wiring conditions, polarity and leakage current issues. Electromechanical input devices such as switches and relays close sets of contacts to allow current to pass, and as such are less sensitive to those situations. Check the specifications for the sensors and the PLC before making connections. It is likely that the manufacturer of the sensor or switch you are using has a version of the device that is appropriate for use with your particular micro PLC.

B

Operator-Manipulated Switches

The *pushbutton switch* is one of the simplest and most commonly used forms of input control. Pushbuttons are used to start and stop equipment, and to initiate processes.

Selector switches incorporate an operator, or switch mechanism, that has several positions. Selector switches use a rotary motion of the knob or other operator to accomplish switching.

Foot switches are used where the operator's hands need to be used to manipulate other things while operating the equipment, or where repetitive hand operations of a switch might cause the operator discomfort.

Thumbwheel switches are a common way of entering numerical data into a control circuit. Each digit, or decade, has a physical marking that represents a number from 0 to 9. Each decade requires four inputs to connect it to the PLC. By changing the sequence of Ons and Offs (BCD code) the switch "tells" the controller what number has been entered.

Limit Switches

Limit switches are used to sense the position of objects or materials. Conveyors, doors, swingarms, valves and many other devices use limit switches to provide control system information on the physical position of equipment. The limit switch uses an actuating mechanism to make or break switch contacts. Many types of actuating mechanisms are available, but the most common are the roller lever, the push roller, the fork lever, and the wobble stick.

Float Switches

Float switches are the easiest means to monitor liquid level in a container. They are typically used in wet wells, tanks, sumps, reservoirs, etc. As the liquid level in the container changes, the actuating mechanism moves. Control of the level of liquid in the container is achieved by setting the limit switch to activate at a desired liquid level.

Flow Switches

A flow switch is inserted into a pipe or duct to sense the movement of a fluid. The fluid might be air, water, oil, or some other gas or liquid. The sensing element is a valve or vane that extends into the fluid stream. The vane will move and actuate electrical contacts whenever the flow is sufficient to exceed a preset spring tension on the vane.

B

Pressure Switches

Pressure switches are used to detect a pressure level and provide digital feedback to the PLC if the level exceeds a specified amount. They are typically used to notify the control system or operator that an excessive pressure condition exists. Pressure switches use a spring-loaded bellows mechanism to close contacts. Pressure of the fluid being sensed is directed into the bellows by tubing or other means. When the pressure in the bellows exceeds the preset spring tension, the switch is actuated.

Temperature Switches

Temperature switches are typically used to detect overtemperature conditions. When the temperature of the object or process being monitored approaches a preset threshold, the device switches. Bimetallic and bulb/capillary type temperature switches typically use switching contacts, while thermocouple switches typically use solid state outputs.

Encoders

An encoder is a form of sensor that changes rotary motion into high-speed pulses. Encoders are either incremental, which track speed and direction of motion of a shaft, or absolute, which track shaft position at all times. The number of pulses generated corresponds to distance or degree of shaft rotation.

Proximity Sensors

Proximity sensors are used to detect the presence or absence of an object without making contact with it. Capacitive sensors sense the change in dielectric field strength as an object moves closer to and further from the sensor. Inductive sensors depend upon the changes in inductance within a coil when a metallic object comes within range of the sensor. Which sensor is appropriate for a given application depends on the material to be sensed.

Photoelectric Sensors

Photoelectric sensors use a light beam to detect objects. There are three basic types of photoelectric sensors:

- In *transmitted beam* sensors, the object being sensed moves between a light source and a receiver module that contains the photodetector.
- In *retroreflective* type sensors, the object to be sensed moves between the sensor, (which contains both the light source and the photodetector) and a reflector.
- In *diffuse* sensors, the natural reflectivity of the object being sensed causes the return signal that triggers the photodetector.

Ultrasonic Proximity Sensors

Ultrasonic proximity sensors use the comparative strength of the return signal from a projected ultrasonic signal to sense how far an object is from the source of the sound, in much the same way that a bat navigates during flight. They are typically used to detect the level of materials.

Output Devices

Output devices are field devices used to carry out the control instructions for the PLC. Think of them as the hands and feet of the PLC. The micro PLC is capable of activating a large variety of output devices. Output voltage and current characteristics of the PLC are the only limiting factors for output device application. The following is a listing of the most popular output devices:

Lamps

Lamps are used to indicate status of an operation or to warn of undesirable or dangerous conditions. Lamp color can be used to differentiate functions or parameters. Lamps come in several different types: incandescent, flourescent, neon and LEDs. Supply voltage and current, lamp life and cost are the three variables which determine which lamp is best for a given application.

Audible Alarms

Audible alarms are available in the form of horns, buzzers, bells, chimes, capacitive alerters, and even synthesized voice modules. All may be used in the control process to alert the machine operator to a condition or event.

Relays

Electromechanical relays use a low amperage control signal to electromagnetically engage a set of contacts. This set of contacts is used to switch a current that can be much higher than the original control signal. In a similar fashion, semiconductor devices like transistors, FETs, triacs or other devices can use a lower amperage output from a PLC to switch a higher current load. These devices are sometimes referred to as solid state relays.

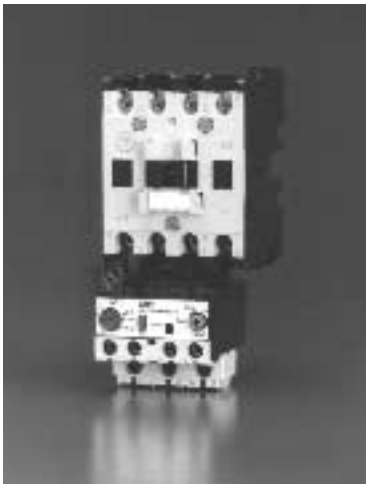
Contactors

Contactors are relays that are able to switch high current loads (>10A). The coil voltage of a large contactor in many cases must be switched by a relay contact, because the coil operating current is higher than the output current of the PLC. Contactors are used for switching motors, heaters, etc.

B

Motor Starters

Motor starters are contactors which have the protection of an overload circuit. An overload circuit protects the motor from damage if operating current is less than the starting inrush current (the current rating of the fuses), but greater than the current normally observed during operation.



Solenoids

Solenoids convert electrical signals to mechanical motion. An electromagnetic coil attracts a plunger or other mechanism to an alternate position when energized. Spring tension or gravity is used to return the plunger to the original position when the electromagnet is de-energized. Solenoids are most commonly used as part of other machines or components.

Valves

Solenoid operated valves are a very common type of output device. A linear solenoid operates the valve mechanism to control the flow of materials in a process. The addition of the valve allows the PLC to control pneumatic and hydraulic operations in addition to electrical and electromechanical operations.

C

Appendix C–Instruction Execution Times

Typical Instruction Execution Times and Memory Usage

The table below lists the execution times and memory usage for controller instructions typically shown in a PLC.

Instruction Type	Name	Mnemonic	Time needed to execute the instruction when it is not True (0) (approx. µsec.)	Time needed to execute as a True (1) statement (approx. µsec.)	Memory usage (user words)
Application Specific	Bit Shift Left	BSL	19.80	53.71 + 5.24 x position value	2.00
Application Specific	Bit Shift Right	BSR	19.80	53.34 + 3.98 x position value	2.00
Application Specific	Interrupt Subroutine	INT	0.99	1.45	0.50
Application Specific	Selectable Timer Interrupt Disable	STD	3.16	6.69	0.50
Application Specific	Selectable Timer Interrupt Enable	STE	3.16	10.13	0.50
Application Specific	Selectable Timer Interrupt Start	STS	6.78	24.59	1.25
Application Specific	Sequencer Compare	SQC	27.40	60.52	2.00
Application Specific	Sequencer Load	SQL	28.12	53.41	2.00
Application Specific	Sequencer Output	SQO	27.40	60.52	2.00
Basic	Count Down	CTD	27.22	32.19	1.00
Basic	Count Up	CTU	26.67	29.84	1.00
Basic	Examine if Closed	XIC	1.72	1.54	0.75
Basic	Examine if Open	XIO	1.72	1.54	0.75
Basic	One-Shot Rising	OSR	11.48	13.02	1.00
Basic	Output Energize	OTE	4.43	4.43	0.75
Basic	Output Latch	OTL	3.16	4.97	0.75
Basic	Output Unlatch	OTU	3.16	4.97	0.75
Basic	Reset	RES (timer/ counter)	4.25	15.19	1.00
Basic	Retentive Timer	RTO	27.49	38.34	1.00
Basic	Timer Off–Delay	TOF	31.65	39.42	1.00
Basic	Timer On–Delay	TON	30.38	38.34	1.00
Comparison	Equal	EQU	6.60	21.52	1.50
Comparison	Greater Than	GRT	6.60	23.60	1.50
Comparison	Greater Than or Equal	GEQ	6.60	23.60	1.50
Comparison	Less Than	LES	6.60	23.60	1.50
Comparison	Less Than or Equal	LEQ	6.60	23.60	1.50
Comparison	Limit Test	LIM	7.69	36.93	1.50
Comparison	Masked Comparison for Equal	MEQ	7.69	28.39	1.50

C

Instruction Type	Name	Mnemonic	Time needed to execute the instruction when it is not True (0) (approx. µsec.)	Time needed to execute as a True (1) statement (approx. µsec.)	Memory usage (user words)
Comparison	Not Equal	NEQ	6.60	21.52	1.50
Data Handling	And	AND	6.78	34.00	1.50
Data Handling	Convert from BCD	FRD	5.52	56.88	1.00
Data Handling	Convert to BCD	TOD	6.78	49.64	1.00
Data Handling	Decode 4 to 1 of 16	DCD	6.78	27.67	1.50
Data Handling	Encode 1 of 16 to 4	ENC	6.78	54.80	1.50
Data Handling	Exclusive Or	XOR	6.92	33.64	1.50
Data Handling	FIFO Load	FFL	33.67	61.13	1.50
Data Handling	FIFO Unload	FFU	34.90	73.78 + 4.34 x position value	1.50
Data Handling	File Copy	COP	6.60	27.31 + 5.06/ word	1.50
Data Handling	Fill File	FLL	6.60	26.86 + 3.62/ word	1.50
Data Handling	LIFO Load	LFL	33.67	61.13	1.50
Data Handling	LIFO Unload	LFU	35.08	64.20	1.50
Data Handling	Masked Move	MVM	6.78	33.28	1.50
Data Handling	Move	MOV	6.78	25.05	1.50
Data Handling	Negate	NEG	6.78	29.48	1.50
Data Handling	Not	NOT	6.78	28.21	1.00
Data Handling	Or	OR	6.78	33.68	1.50
High-Speed Counter	High-Speed Counter	HSC	21.00	21.00	1.00
High-Speed Counter	High-Speed Counter Interrupt Disable	HSD	7.00	8.00	1.25
High-Speed Counter	High-Speed Counter Interrupt Enable	HSE	7.00	10.00	1.25
High-Speed Counter	High-Speed Counter Load	HSL	7.00	66.00	1.50
High-Speed Counter	High-Speed Counter Reset	RES (high-speed counter)	6.00	51.00	1.00
High-Speed Counter	High-Speed Counter Reset Accumulator	RAC	6.00	56.00	1.00
High-Speed Counter	Update High-Speed Counter Image Accumulator	OTE (high-speed counter)	7.00	12.00	0.75
Math	Add	ADD	6.78	33.09	1.50
Math	Clear	CLR	4.25	20.80	1.00
Math	Divide	DIV	6.78	147.87	1.50
Math	Double Divide	DDV	6.78	6.00	1.00
Math	Multiply	MUL	6.78	57.96	1.50
Math	Scale Data	SCL	6.78	169.18	1.75

Instruction Type	Name	Mnemonic	Time needed to execute the instruction when it is not True (0) (approx. µsec.)	Time needed to execute as a True (1) statement (approx. µsec.)	Memory usage (user words)
Math	Square Root	SQR	6.78	71.25	1.25
Math	Subtract	SUB	6.78	33.52	1.50
Program Flow Control	Immediate Input with Mask	IIM	6.78	35.72	1.50
Program Flow Control	Immediate Output with Mask	IOM	6.78	41.59	1.50
Program Flow Control	Jump to Label	JMP	6.78	9.04	1.00
Program Flow Control	Jump to Subroutine	JSR	4.25	22.24	1.00
Program Flow Control	Label	LBL	0.99	1.45	0.50
Program Flow Control	Master Control Reset	MCR	4.07	3.98	0.50
Program Flow Control	Return from Subroutine	RET	3.16	31.11	0.50
Program Flow Control	Subroutine	SBR	0.99	1.45	0.50
Program Flow Control	Suspend	SUS	7.87	10.85	1.50
Program Flow Control	Temporary End	TND	3.16	7.78	0.50

D

Appendix D–Sample Program Worksheets

Throughput Time Worksheet

Throughput is the amount of time it takes for the PLC to sense an input and energize the corresponding output. Components of throughput time include: time for the PLC's input circuit to sense the signal; time for the input, output and program scans; time for actuation of the PLC's output circuits; and time for the CPU's "housekeeping" functions.

Once your program is written, use the following worksheet to estimate PLC throughput time. To assist you, typical times have been provided where needed. To determine actual throughput time, consult your PLC users manual. This is very important, as execution times differ between PLC manufacturers.

Procedure	**Maximum Scan Time**
1. Input scan time	8 µs (typically)
2. Output scan time	8 µs (typically)
3. Housekeeping time	180µs (typically)
4. To estimate program scan time, take your program and add instruction execution times when all instructions are True	µs*
5. To estimate program throughput time:	
A. Without communications**, add sections 1-4	µs
B. With communications, add sections 1-4 and multiply by 1.05	µs
6. PLC input circuit filter time	µs
7. PLC output circuit turn-on time	µs
8. To estimate total throughput time for the PLC, add sections 5-7	µs

(Note – This will result in the "worst case," or longest possible throughput time)

* An example set of instruction execution times is provided in Appendix C.

** Communication with devices, such as a Hand-Held Programmer, a personal computer, or an electronic operator interface.

D

Estimating Memory Usage for The Control System

Once your program is written, use the following worksheet to estimate memory usage. To assist you, typical words of memory have been provided where needed. To determine actual memory usage, consult your PLC user manual. This is very important, as the amount of memory consumed by various instructions differ between PLC manufacturers.

1. Determine the total number of instruction words used by the instructions in your program and enter the result	*	
2. Multiply the total number of rungs by 0.75 and enter the result – do not count Start of File or End of File rungs		
3. Words allocated by controller	280	(typically)
4. Add steps 1-3 for total estimated memory usage		
5. Subtract the total from 1024 to determine memory remaining		

Important: The calculated memory usage is only an estimate. Actual memory usage can vary by 10 to 15%.

* An example set of instruction memory usage is provided in Appendix C.

Index

address 24-25, 38, 76-79,84, 86, 91-92, 142
AND logic 12, 44, 46, 97, 142
application memory 22, 142
auxiliary holding contact 47-48, 76
BASIC programming language 34, 142
basic instructions 161
battery backup 23, 142
BCD 10, 12, 24, 142
Boolean 12, 34, 143
branching instructions 46-48
bus 35, 143
C programming language 34, 143
communication with programming device 31
communication with operator interface 5, 31
comparison instructions 44, 117, 161
condition instructions 42, 44-46, 75-76, 143
contact symbology 39, 143, 149
control instructions 37-38, 44, 46, 144
counter instructions 111, 113, 128-129

CSA 62, 144
EEPROM 22-23, 145
EMI 27-28, 145
FET 21, 68-69, 145
floating point decimal 24
function blocks 22, 43-44, 106-108, 110, 120
GM 2, 64
Gray code 24
hexadecimal 24
HHP, fault codes 91-92
HHP, use in programming 28-30
HHP, use in troubleshooting 30, 85, 88, 91
high-speed counter 14, 69-70, 128-129
IEC 146
input device 18-19, 43, 84-85, 90, 146, 153-158
input scan 22, 26, 40-41, 146
installation 62-63, 80-82
instruction set 12-14, 96, 161-163
integer 24, 114-116
IP 59, 80, 146
ladder logic 4, 36-44, 49, 70, 72, 146
languages, programming 3-4, 30, 34, 142-143
latched output 76, 146
limit switch 19, 55, 57, 147, 155
logical continuity 38-45, 96-96, 145, 147, 148, 151
math instructions 9, 12, 120-122, 162-163
micro PLC, capabilities 13-14
micro PLC, characteristics 9-10
micro PLC, introduction 8-9
micro PLC, typical applications 14
micro PLC, typical features 8-9
mnemonic 34-35, 147

modular controller 13, 147
NEMA 4 80, 147
NEMA 12 80, 147
normally closed instruction 39-40, 75, 91, 108-111, 147
normally open instruction 39-40, 43, 75, 91, 99, 111, 147
octal 24
operating cycle 24-27, 49, 143, 146, 148, 149
operator interfaces 5, 18, 30-31, 58-59, 148
optical isolation 19-20
OR logic 12, 44-46, 76, 96, 143, 148, 162
OSR (one-shot rising) 100, 161
output device 20, 24, 42, 53-54, 84, 86, 90, 148, 158-160
output image file 26
output scan 26-27, 40-42, 148
packaged controller 11, 13, 148
photoelectric sensors 19, 55, 57, 157
PLC, advantages over relays 2, 61
PLC, economic benefits 2-5, 61
PLC, history of 2-3
PLC, typical applications 5-6
power supplies 11, 18, 26-28, 148
program file 21-22
program printout 84, 90
program scan 14, 26-27, 49, 70,149
programming examples 57-61, 68, 72-79, 96-140
proximity sensors 14, 19, 157-158
pushbuttons 19, 31, 39, 41, 43, 58, 98, 154
RAM 22, 23, 149
register 24, 149
retentive data 22, 149
retentive timers 12, 111
RS-232 31,149

RES (reset) 112, 161
rung 35-46, 49, 72, 149
SBC, description 52, 150
SBC, typical application 60
SBC, use of 62-65
solenoid 20, 43, 52, 82, 86, 123, 150, 160
system memory 22, 142, 150
temperature limitations 59, 80-81
throughput time 27, 150, 164
thumbwheel switches 19, 24, 31, 58, 142, 150, 154
time base 106-111, 151
timer instructions 12, 37, 43-44, 112, 161
transistor 21, 68, 69, 145, 150, 151, 153, 159
triac 21, 68, 69, 151, 153, 159
twisted-pair cable 59, 81
UL 62, 151
warning lamps 20, 158

Micro Programmable Logic Controllers

Small enough to fit in one hand, the compact micro PLC provides a powerful solution to today's electronic control applications – from the simple to the complex – at an affordable price.

The *MicroMentor* reference book serves as an introduction to micro PLCs for anyone, from design engineers or electrical technicians to maintenance personnel and students. While some knowledge of basic electronic controls is helpful, it is not required, nor is previous experience with programmable logic controllers.

Through easy-to-understand text and numerous illustrations, the reader will gain a practical understanding of micro PLCs. *MicroMentor* demonstrates the advantages of micro PLCs over electromechanical controls, and it helps readers evaluate the best control system for their application.

Topics covered include:

- History of PLCs
- Micro PLC capabilities and operation
- Programming with ladder logic
- How to apply a micro PLC
- Commissioning and troubleshooting
- Application examples

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 world's 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

1761-MMB – September 1995