1 ARCHITECTURE AND OPERATION

To understand the Allen-Bradley MicroLogix 1000, you need to start with the basics. This first module explains the basic operation of programmable controllers and details the specific features of the MicroLogix 1000 PLC.

This first module is broken up into four sections:

1. MicroLogix 1000 basic principles of operation
2. MicroLogix specifications
3. I/O structure and memory system
4. Addressing notation

Key Points

After finishing this module, you will:

- understand the basic operating principles of the MicroLogix—how it works, what the components are, and what the components do
- know the three basic specifications for the MicroLogix 1000—how the program is written, how data is represented in the system, and what configurations are available
- grasp the MicroLogix's intricate memory system—how the I/O is set up, what makes up the memory system, and how the memory system is organized
- understand the unique MicroLogix addressing notation

1-1 MicroLogix 1000 Basic Principles of Operation

The MicroLogix 1000 programmable logic controller may appear to be like any other PLC, but it has special features, specifications, and capabilities that make it a unique tool for implementing process or machine control. The MicroLogix 1000 follows many of the same basic principles of operation that all PLCs follow. At the end of this section, you will know:

- what PLCs do
- why PLCs are invaluable to industrial facilities
- what makes up a PLC
- how a PLC operates

PLC Fundamentals

A MicroLogix 1000 is a **programmable logic controller**—an industrial computer that controls a machine or process. A PLC interfaces with the field input and output devices that are part of a control application. Then, through the control program stored in its memory, the PLC uses the data supplied by the input devices to manipulate or control the output devices. The overall PLC process, which is shown in Figure 1-1, is very simple. A PLC measures or senses signals coming from a machine or process. Then, through its internal program, the PLC provides control back to the machine or process.

Programmable logic controllers provide many benefits over traditional electromechanical control systems. One of the best benefits is that PLCs make it easier and less costly to change a control system. They eliminate the need to rewire the input and output devices if the control requirements change. If the control requirements for a PLC application change, all you need to do is

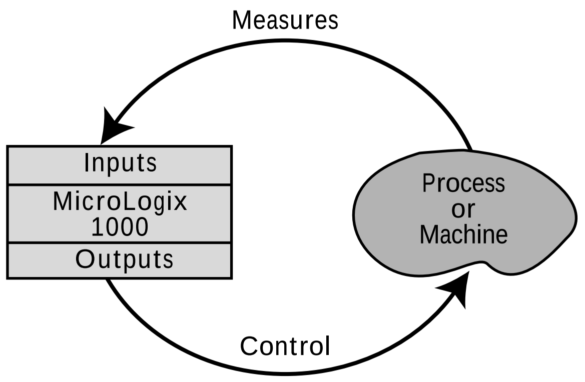


Figure 1-1. PLC operation.

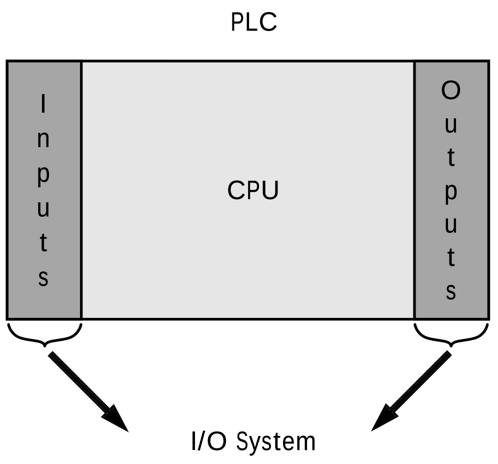


Figure 1-2. A PLC and its components: the central processing unit (CPU) and the input/output (I/O) system.

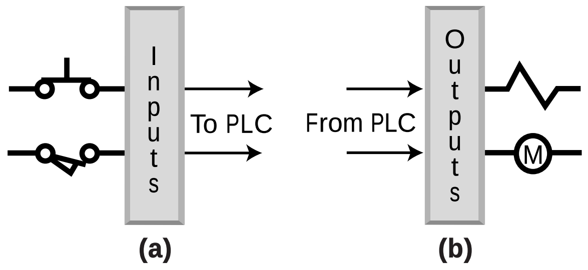


Figure 1-3. (a) A PLC's input interface interprets the data from the input devices and then sends it to the CPU. **(b)** A PLC's output interface interprets the data from the CPU and sends it to the output devices.

change the control program. Another benefit of PLCs is that they are more powerful and more accurate than electromechanical systems.

PLC Components

A PLC is made up of two basic components (see Figure 1-2):

- the input/output (I/O) system
- the central processing unit (CPU)

The **input/output system** is the part of the PLC that physically connects to devices in the outside world. The **central processing unit**, on the other hand, is where the PLC stores all of its data and does all of its computer processing. Each of the components of a PLC has specific functions.

Input/Output System. The input/output system is made up of two components, the input interface and the output interface (see Figure 1-3).

An **input interface** is a bank of terminals that physically connects input devices, like push buttons and limit switches, to a PLC. These input devices provide data to the PLC. The role of an input interface is to translate data from the inputs into a form that the PLC's central processing unit can understand.

An **output interface** is a bank of terminals that physically connects output devices, such as solenoids and motor starters, to a PLC. These output devices receive control data from a PLC. The role of an output interface is to translate data from the PLC's CPU into a form that the output devices can understand.

To put it simply, the I/O system communicates information from the input devices to the CPU. It also communicates data from the CPU to the output devices.

CPU. The CPU is made up of three parts (see Figure 1-4):

- the memory system
- the processor
- the power supply

The **memory system** stores the PLC's control program, as well as the data received from and sent to the I/O system. It also keeps track of which I/O devices are connected to which I/O interfaces. The **processor** is the computerized part of the CPU that performs the control program. It manipulates the data stored in the memory system and determines what control output should occur based on the given input conditions. The **power supply** provides power to both the memory system and processor so that they have power and so that they work properly.

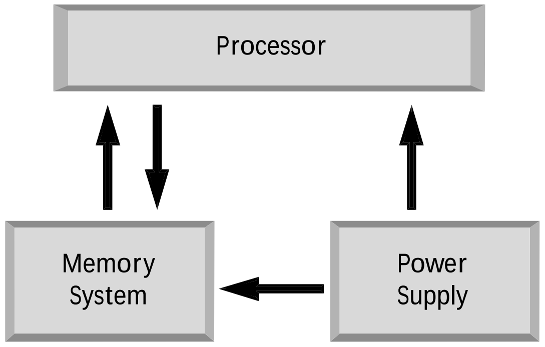


Figure 1-4. A CPU with its three components—the processor, the memory system, and the power supply.

PLC Operation

All PLCs, including the MicroLogix 1000, perform a three-step operation called a **scan** (see Figure 1-5). The scan consists of:

1. reading the input data that the PLC receives from the input devices
2. executing the control program stored in memory
3. updating, or writing, the status of the output devices based on the outcome of the control program execution

A PLC performs the scan over and over again, constantly updating the outputs based on how new input conditions affect the control program.

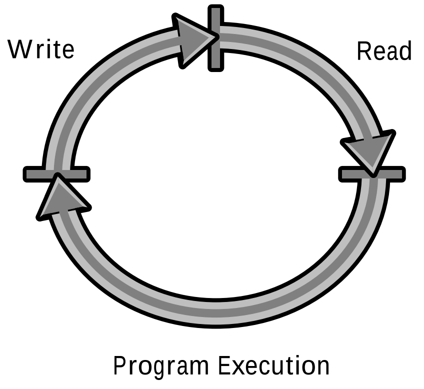


Figure 1-5. A PLC's scan consists of reading the inputs, executing the control program, and updating the outputs.

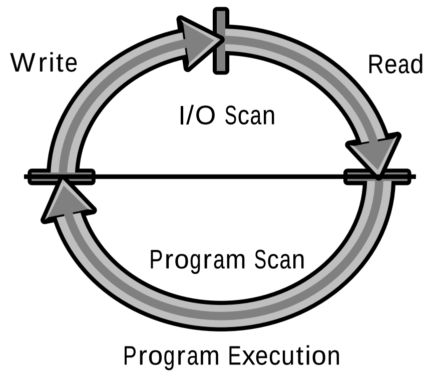


Figure 1-6. A PLC's total scan consists of two different scans: the I/O scan and the program scan.

The scan can be broken up into two different parts, the **I/O scan** and the **program scan** (see Figure 1-6). During the I/O scan, the PLC reads inputs and updates the outputs. During the program scan, the PLC executes the control program.

The **scan time** is the specific amount of time required for a PLC to perform both the I/O scan and the program scan. Each PLC's scan time is different. A MicroLogix 1000 can perform a scan in milliseconds. However, when it is on-line with a programming device, the MicroLogix experiences two delays during its scan (see Figure 1-7). These are:

- the service communications delay
- the overhead delay

The **service communications delay** is the time required for the MicroLogix 1000 to send data to the programming or monitoring device, which may be a personal computer or a handheld programmer. The **overhead delay** is the time required for housekeeping operations, like memory management and updating timer information. Although both of these delays add to the MicroLogix 1000's scan time, it still performs its scan very quickly.

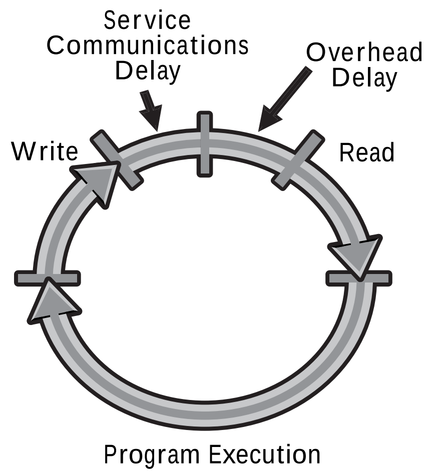


Figure 1-7. The MicroLogix experiences two scan delays, a service communications delay and an overhead delay, when it is on-line with a programming device.

1-2 MicroLogix Specifications

The MicroLogix 1000 PLC is a powerful micro–programmable controller capable of implementing all kinds of control functions. Although the MicroLogix 1000 follows the basic PLC principles of operation, it does have many unique functions and specifications. This section will discuss some of these specifications. At the end of this section, you will know:

- what the control program is and how it is represented in the MicroLogix 1000
- what number systems are used by the MicroLogix 1000 to represent data
- how the MicroLogix 1000 is configured

Control Program Notation

A control application can be implemented using either the traditional hardwired method or the PLC softwired method. In the traditional hardwired method, the input and output devices are wired directly to each other. The sequence of operation, which is the logic behind the system, is determined by the way the devices are physically connected (see Figure 1-8).

In the PLC softwired method, the input and output devices are wired to the PLC's input and output interface terminals—not to each other (see Figure 1-9). The control program, which resides in the PLC's memory, provides the connections between the devices. So instead of being hardwired, the devices are "softwired" to each other. The MicroLogix 1000's softwired control program is represented through **ladder diagram notation**.

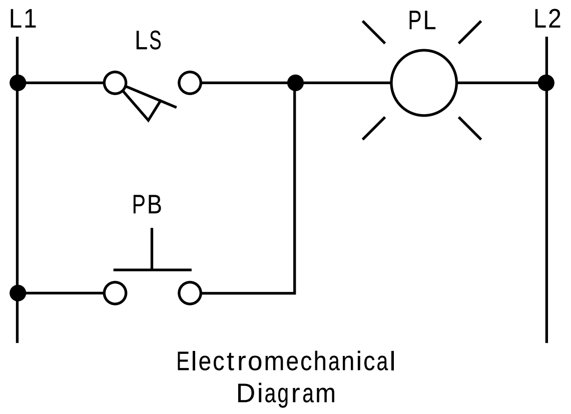


Figure 1-8. A hardwired circuit where either a limit switch or a push button can turn on a pilot light.

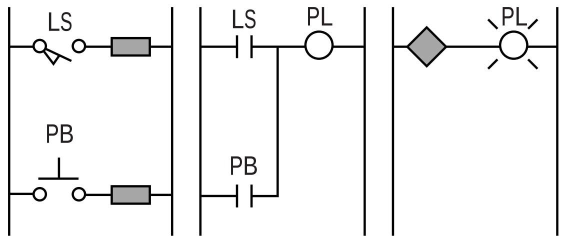


Figure 1-9. The circuit in Figure 1-8 implemented in a PLC via input/output connections.

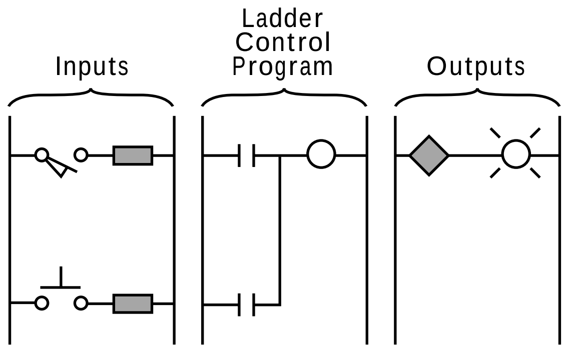


Figure 1-10. An example of ladder diagram notation.

Ladder diagram notation has a particular format, as shown in Figure 1-10. The left side of a ladder circuit shows the input devices wired to the input terminals, which are represented by rectangles. The right side shows the output devices wired to the output terminals, which are represented by diamonds. The middle part is the ladder diagram logic that connects the inputs and the outputs together.

The logic performed within a ladder program works just like its equivalent electromechanical circuit would. However, a ladder program represents its inputs by a contact rather than by a device symbol. Likewise, it represents its outputs by a coil instead of by a device symbol. The PLC uses an addressing scheme in the ladder program to keep track of all its inputs and outputs, including which contacts and coils reference them. The last section of this module explains this addressing scheme.

The use of a MicroLogix PLC has many benefits over a traditional electromechanical application. The first is flexibility. In a hardwired, or traditional, system, the devices must be physically rewired if the control requirements change. This takes time and money. However, in a PLC system, no rewiring is necessary. All changes are made to the PLC's ladder program instead. This process is much quicker and less costly than rewiring. The second benefit is reliability. The MicroLogix 1000 PLC is solid-state and has no moving parts, which makes it very dependable.

Number Systems

Number systems are used to represent data in a PLC. The MicroLogix 1000 PLC uses several different types of number systems to represent program data, address data, and internal data. They are:

- binary
- decimal
- hexadecimal
- octal
- binary codes

Binary. The MicroLogix 1000 uses the **binary number system** to represent program data. The binary number system uses only two numbers, 0 and 1, to represent data. PLCs, including the MicroLogix 1000, use the binary system to represent I/O data because PLCs are discrete devices capable of recognizing only two states, ON and OFF.

Using the binary system, a PLC indicates that a device is ON, or activated, by placing a value of 1 in the appropriate bit in memory (see Figure 1-11). Conversely, a PLC indicates that a device is OFF, or not activated, by placing a value of 0 in the appropriate bit in memory.

Decimal. The MicroLogix 1000 uses the **decimal number system** to represent the address data of inputs and outputs, as well as contacts, coils, timers, counters, and other internal instructions. The decimal number system uses ten numbers, 0 through 9, to represent data.

The addresses represented by decimal numbers identify which contacts and coils refer to which input and output devices. An address tells the PLC specifically which I/O device is wired to

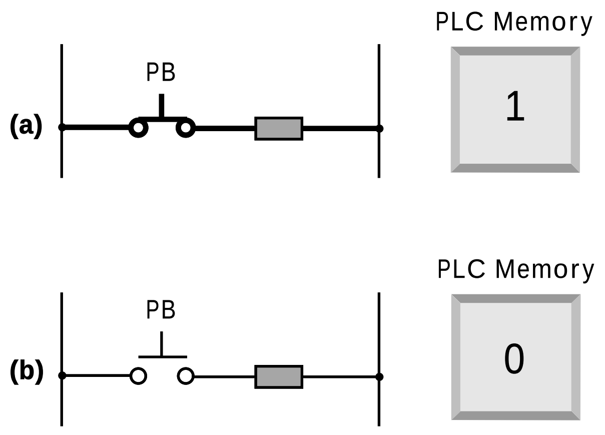


Figure 1-11. **(a)** If a device is ON, a PLC will store a 1 in memory. **(b)** If a device is OFF, a PLC will store a 0 in memory.

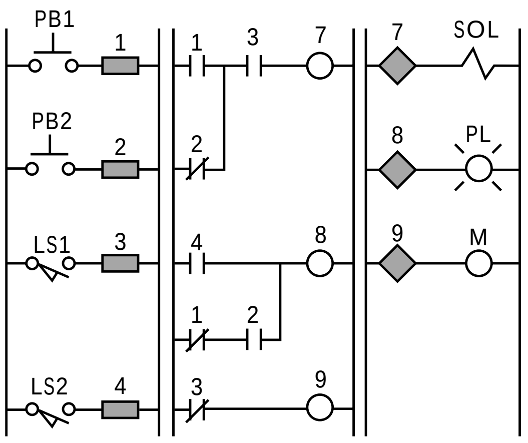


Figure 1-12. The numbers above the contacts, terminals, and coils are addresses expressed by decimal numbers.

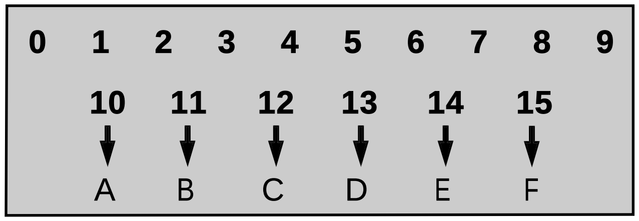


Figure 1-13. In hexadecimal, the numbers 0 through 9 are represented by the digits 0–9 and the numbers 10 through 15 by the letters *A–F*.

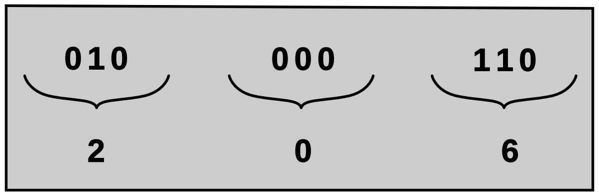


Figure 1-14. The octal number system groups binary numbers into groups of three and then represents each group with a number.

which terminal. Figure 1-12 shows an example of address notation. An address instructs a contact or coil to examine a terminal to see whether the device connected to it is ON or OFF.

Hexadecimal. The **hexadecimal number system** is different from other number systems because it uses both numbers and letters to represent data. The MicroLogix 1000 uses this number system to represent constants and other internal values. The hexadecimal system uses 16 numbers. It represents the numbers 0 through 9 by the digits 0–9. It represents the numbers 10 through 15 by the letters *A–F* (see Figure 1-13).

Octal. The MicroLogix 1000 uses the **octal number system** as a shorthand way to express binary data. The octal number system uses eight numbers, 0 through 7, to represent data. It groups binary numbers into groups of three and then uses one of the numbers 0 through 7 to represent the group of numbers (see Figure 1-14).

Binary Code. A **binary code** is a code that lets a PLC communicate with the outside world. Since PLCs are discrete devices and the rest of the world is not, PLCs must have a way to interpret and communicate nonbinary information from devices like thumbwheel switches and seven-segment indicators. The MicroLogix 1000 uses binary codes to do just that. A binary code translates nonbinary data, like letters, into a binary coded format that the PLC can understand. It also communicates binary information from the PLC to nonbinary outside devices. The MicroLogix 1000 uses two binary codes, ASCII and BCD, to perform these functions.

Configurations

A MicroLogix 1000 PLC comes in many configurations. These configurations differ by:

- the number of inputs and outputs
- the type of power supply
- the type of I/O interfaces

Inputs and Outputs. The number of inputs and outputs determines the size of a MicroLogix PLC. The MicroLogix 1000 comes in two sizes: 16 I/O and 32 I/O. A 16 I/O MicroLogix can connect with up to 10 input devices and 6 output devices (see Figure 1-15). A 32 I/O model can connect with up to 20 input devices and 12 output devices (see Figure 1-16). The size of a MicroLogix 1000 should be chosen based on the amount of I/O required for its application.

Power Supply. The MicroLogix 1000 also has two types of power supplies. These are 24 VDC (volts DC) and 120/240 VAC (volts AC). The power supply should be chosen based on the power requirements and the power availability for the application.

I/O Interfaces. A MicroLogix 1000 PLC has many options available for both its input and output interfaces. A MicroLogix 1000 can have one of two types of input interfaces, either 24 VDC or 120 VAC. These input interfaces allow the MicroLogix 1000 to connect with either 24 VDC or 120 VAC input devices, respectively.

Just as a MicroLogix 1000 has a choice of inputs, it has a choice of outputs as well. The MicroLogix 1000 uses three types of outputs:

- relay
- transistor
- triac

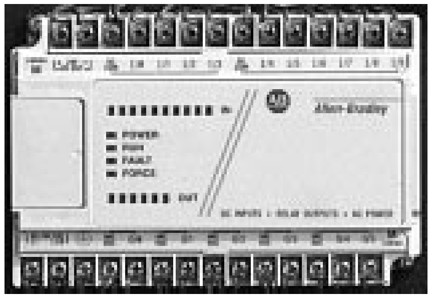


Figure 1-15. A 16 I/O MicroLogix.

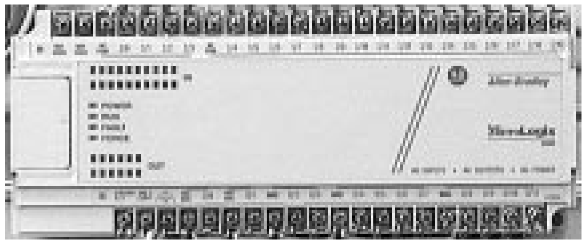


Figure 1-16. A 32 I/O MicroLogix.

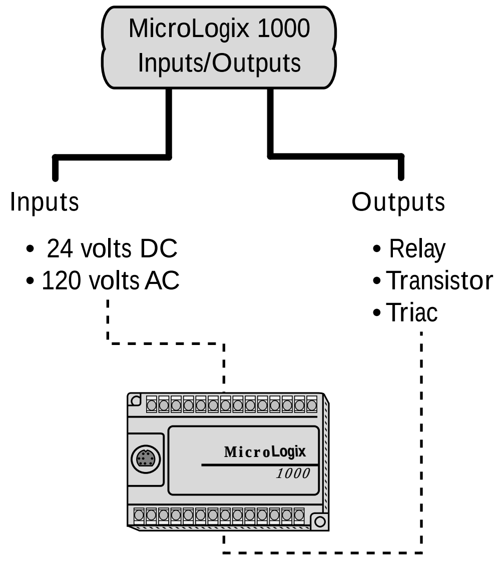


Figure 1-17. The inputs and outputs available in a MicroLogix 1000.

A **relay** output allows the MicroLogix to interface with output devices that must receive a signal ranging between either 5 and 264 VAC or 5 and 125 VDC. A **transistor** output is specifically designed for outputs requiring a 24 VDC output signal. A **triac** specifically supplies a 120/240 VAC signal to its output devices. Figure 1-17 shows the types of input and output interfaces available in the MicroLogix 1000**.**

1-3 I/O Structure and Memory System

This section covers the I/O structure and the memory system of the MicroLogix 1000. Both are very important aspects of the PLC. At the end of this section, you will know:

- how the I/O system is structured
- what makes up the MicroLogix 1000 memory system
- how the memory system is organized

I/O Structure

A MicroLogix's I/O structure is directly related to the way its memory system is organized. Each input is connected to a separate input terminal that has a unique address in the PLC. Also, each output is connected to a separate output terminal that has a unique terminal and memory address. Figure 1-18 shows an example of addressed inputs and outputs connected to the terminals of a MicroLogix PLC.

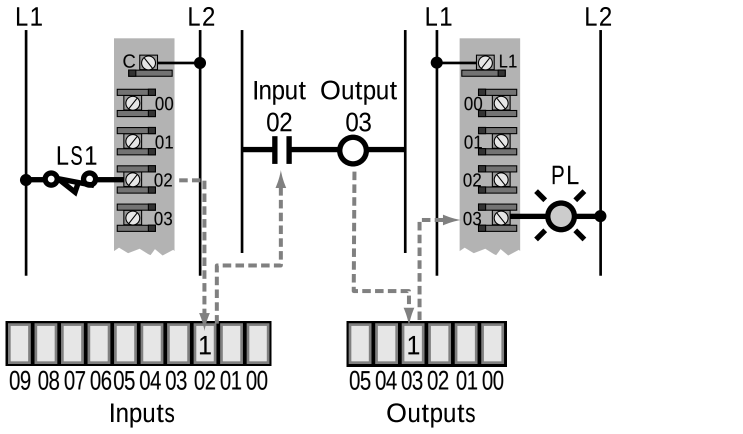


Figure 1-18. The limit switch is connected to the input terminal with address 2. Thus, its status is stored in input address 2 in memory. Likewise, the pilot light is connected to the output terminal with address 3 and has this same address in memory.

Memory System

The memory system of a MicroLogix 1000 consists of four units (see Figure 1-19):

- file sections
- files
- words
- bits

File sections are the largest unit of memory. They specify where major categories of data are stored. The MicroLogix's memory contains two file sections, the program file section and the data

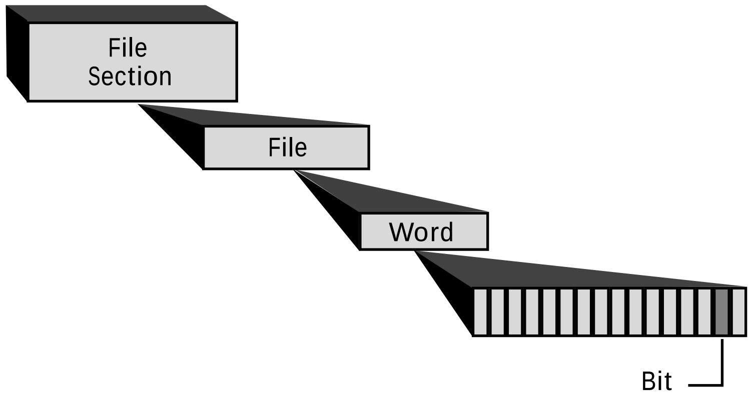


Figure 1-19. The four units used in a MicroLogix's memory system.

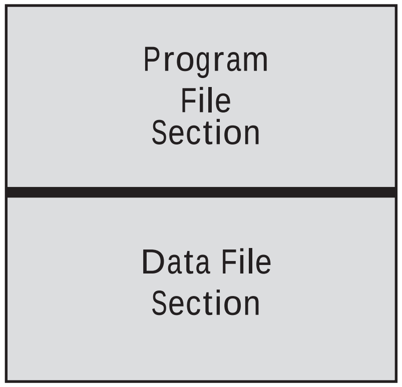


Figure 1-20. The two file sections of a MicroLogix 1000's memory system.

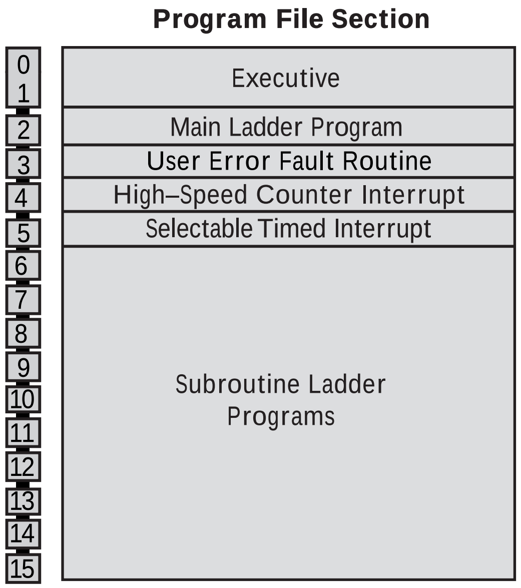


Figure 1-21. The program file section of the MicroLogix 1000.

file section. Each file section is made up of a particular number of **files**. Files are areas in the MicroLogix's memory where a specific type of data, like input data, is stored. Each file consists of a certain number of **words**. Words are groups of memory locations that store pieces of data. Each word can hold up to 16 pieces of data, and each piece of data is called a **bit**. A bit is a binary digit that comprises the smallest unit of memory. A bit holds only one piece of information, either a 1 or a 0.

The MicroLogix's memory system is organized into file sections, files, words, and bits in order to store all of the information that the PLC needs to operate. This information includes the control program, input and output status data, internal data, and routine functioning data. The MicroLogix's memory system has a lot of information to keep track of, so it needs a well-structured organization in order to do that.

As mentioned previously, the MicroLogix's memory contains two file sections—the **program file section** and the **data file section** (see Figure 1-20). Each of these file sections stores a different kind of information.

Program File Section. The program file section stores all the data a MicroLogix needs to operate. This includes data about the processor, the main control program, and any subroutines. Figure 1-21 shows a map of the MicroLogix's program section.

The program section consists of 16 files numbered 0 through 15. They store information as follows:

- *Files 0* and *1* contain the executive software of the MicroLogix 1000. This software is responsible for controlling all of the functions of the PLC and keeping track of what is happening while the PLC is operating. These files also contain data about the processor, including type, configuration, and passwords access.

- *File 2* holds the main ladder program that is entered into the PLC's memory. This ladder program controls the machine or process.
- *File 3* stores an error fault routine that is executed when a recoverable, or fixable, fault occurs in the PLC's control program. When this routine is executed, the MicroLogix corrects the problem to get the system up and running again.
- *File 4* stores the high-speed counter interrupt program that is executed when a high-speed counter instruction causes an interruption in the control program.
- *File 5* contains the selectable timed interrupt program, which is used to interrupt the normal program scan so that a subroutine can be executed immediately.
- *Files 6–15* store the subroutine ladder programs that are called by the main ladder control program. This area can store up to ten subroutines. Files 4 and 5 can be used to store additional subroutines, if necessary.

Data File Section. The **data file section** stores all of the program and I/O data used by the MicroLogix 1000. This section is divided into eight files numbered 0 through 7, as shown in Figure 1-22. Each file stores a different type of information. Following is an outline of the data files:

- *File 0* is the output file. It stores data about the status of each output device connected to the MicroLogix's output terminals. Each of the PLC's outputs is mapped to an address bit in this file. File 0 is also known as the **output image table**. The output file contains one word.
- *File 1* is the input file. It stores data about the status of each input device. The input file is also known as the **input image table**. As with the output file, each of

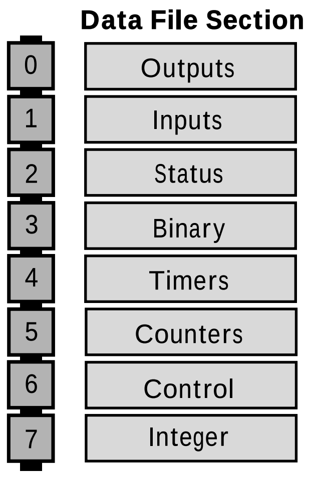


Figure 1-22. The data file section of the MicroLogix 1000.

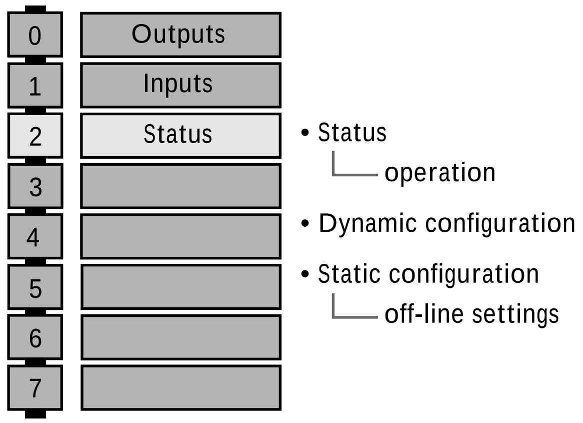


Figure 1-23. The contents of a MicroLogix 1000's status file.

the MicroLogix's inputs is mapped to a specific bit in the input file. The input file is different from the output file because it contains two words to account for all of the possible input devices that can be connected to a 32 I/O MicroLogix.

- *File 2* is the status file. It stores information about how the PLC is operating and how it is set up. This file contains 33 words that hold three types of data: basic status data, dynamic configuration data, and static configuration data (see Figure 1-23).
- *File 3* is the binary, or bit, file. It stores data about the status of internal coils and contacts. The binary file contains 32 words. Because the binary file stores data about internal instructions, its bits do not map real field devices, as the bits in the input and output files do.
- *File 4* is the timer file. It contains data about the timers used in the control program. It includes data about each timer's status, preset value, and accumulated value. The MicroLogix can use up to 40 timers in its control program, and it dedicates one word to each of the three pieces of information it stores about each timer. So, in essence, the timer file has 120 words, three for each of its 40 timers.
- *File 5* is the counter file. It stores data about the 32 counters available in the MicroLogix 1000. It stores three pieces of data about each counter: the counter's status, preset value, and accumulated value. Each piece of counter data is stored in its own word. So, in essence, this file has 96 words, three words for each of the 32 counters.

- *File 6* is the control file. It stores information used by specialized PLC instructions, like shift and sequencer instructions. This file can hold data for 16 instructions. It uses three words for each instruction. Therefore, this file can contain up to 48 words.
- *File 7* is the integer file. This file stores miscellaneous numerical data, such as constant and variable data, binary codes, and mask patterns. The integer file uses 105 words to store all of this information.

Job Aid 1-1 at the end of this module provides an overview of the program and data file sections of the MicroLogix 1000's memory system.

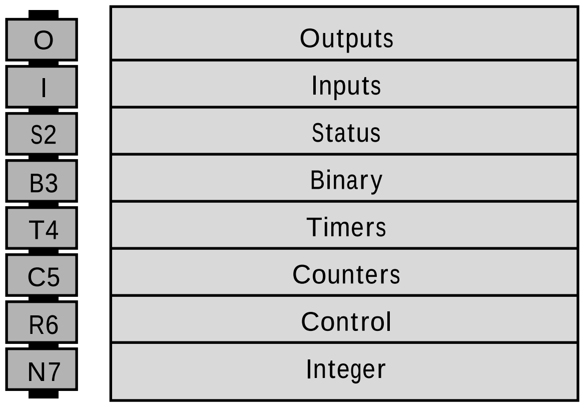


Figure 1-24. The letter codes used for each file in the data file section.

1-4 Addressing Notation

The concept of addressing is very important in a PLC. This is how the PLC keeps track of all of its data. At the end of this section, you will understand:

- the mnemonics of the MicroLogix 1000 addressing notation
- the unique addressing of timers, counters, control files, and input files
- the addressing concepts used for specific situations

Addressing Mnemonics

Every bit in every file of the MicroLogix's data section has a unique **address**. This address allows the MicroLogix to keep track of all its data. The MicroLogix's addressing code has its own language, or mnemonics, which is used to express an address. This addressing code gives the PLC all of the information it needs to find any piece of data stored anywhere in the PLC's memory. Each address has three parts:

- the file label
- the word label
- the bit label

The first part of a MicroLogix's address is the **file label**. This label tells the PLC which file the data is stored in. A letter or a letter/number combination is used to denote each file. Figure 1-24 lists the letter codes used for each file in the data section.

The **word label** is the next part of the address. It lets the PLC know which word in the file the data is located in. The last part of the address is the **bit label**. This label tells the PLC which bit of the word the data is in.

Additionally, the MicroLogix 1000 uses **delimiters** to separate the different parts of an address. A colon (:) is used to separate the file label and the word label. A slash (/) is used to separate the word label and the bit label. Figure 1-25 shows an example of the MicroLogix 1000's addressing notation.

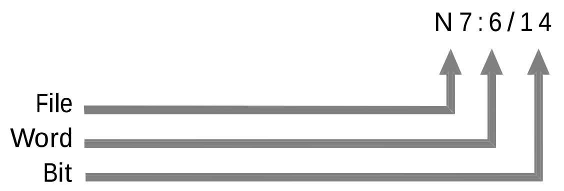


Figure 1-25. This address indicates that the data is stored in the integer file in word 6, bit 14.

Special Addressing Situations

The timer, counter, and control files also use the addressing system just explained. However, they add two extra characters to the word label—a period and a number.

The first number in this special word label identifies the timer, counter, or special instruction number. The period acts as a delimiter. The last number in the word label refers to one of the three words associated with the timer, counter, or special instruction. Figure 1-26 shows an example of the addresses for three timers.

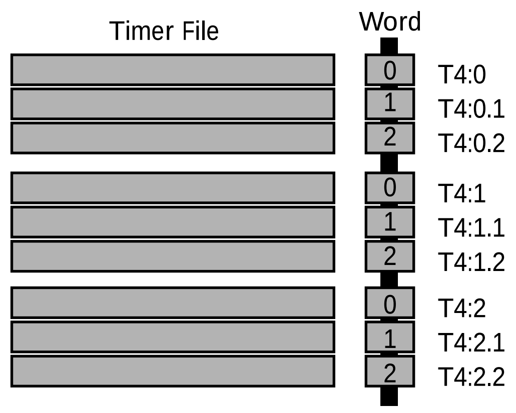


Figure 1-26. Three timers, each with three word addresses. Note that the period/word number extension is dropped from each timer's first word address.

Like the timer, counter, and control files, the input file also has a unique address code. It also adds two extra characters, a period and a number, to the word label. This is done to account for the additional input word needed for a 32 I/O MicroLogix (see Figure 1-27). The first input word is labeled I:0.0. It holds the data for the first 16 inputs. The second word is labeled I:0.1. It holds the data for the remaining 4 inputs.

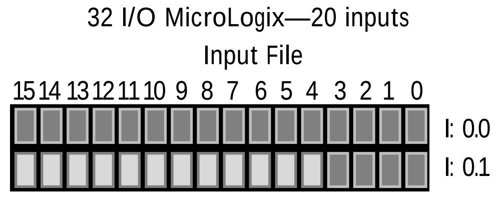


Figure 1-27. The MicroLogix 1000's input file with the two words contained in it.

Addressing Examples

Following are some examples of the addressing scheme used in a MicroLogix 1000 programmable controller:

Example 1—Output Device. Figure 1-28 shows a pilot light connected to output terminal 4 of a 16 I/O MicroLogix 1000. Since the pilot light is an output, its address will start with the letter *O* followed by a colon **(O:)**. The output file has only one word, so the pilot light's status data will be stored in word 0

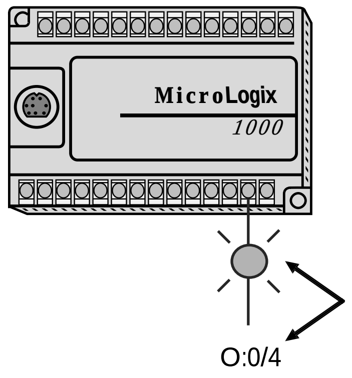


Figure 1-28. Output device address.

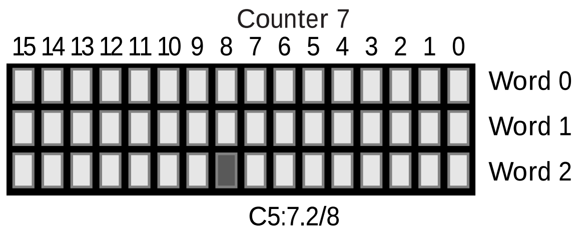


Figure 1-29. Counter file address.

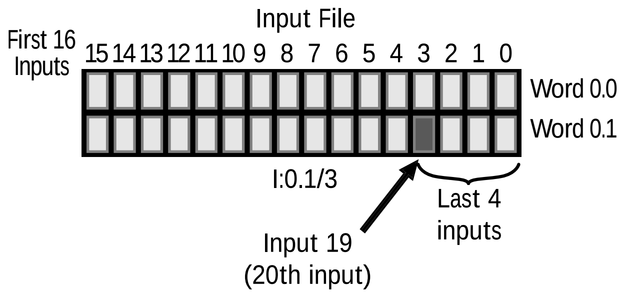


Figure 1-30. Input file address.

(O:0). Finally, the pilot light is connected to terminal 4, so it will be mapped to bit 4 of output word 0 **(O:0/4)**. Therefore, the pilot light will have the address **O:0/4**.

Example 2—Counter. Figure 1-29 shows the address used to check the data value stored in bit 8 of counter 7's accumulated value. The MicroLogix stores the accumulated value for a counter in the last word of the three words associated with the counter. The first part of the address **(C5:)** indicates that the data is stored in the counter file, since it is counter data. The second part of the address indicates that the accumulated value for counter 7 is stored in word 2 **(C5:7.2)**. The data value specified is located in bit 8, so this data has the address **C5:7.2/8**.

Example 3—Input Device. Figure 1-30 shows the memory map for a push button connected to the last input terminal, input 19, of a 32 I/O MicroLogix. A push button is an input, so this device's address will start with an *I* and a colon **(I:)**. The push button is connected to the last input terminal, terminal 19. Since a word has only 16 bits, this input's address must be located in the second word **(I:0.1)**. Specifically, this input device's status is stored in bit 3 of the second word of the 32 I/O MicroLogix; therefore, its address is **I:0.1/3**.

Entering Address Data

When working with a MicroLogix 1000, address and other program data can be entered in one of two ways—with a handheld programming device or with a personal computer equipped with the RSLogix software. Each of these addressing methods uses its own addressing notation. The addressing notation shown in the video and in this book is the basic one used by the RSLogix software. Job Aid 1-2 at the end of this module shows the differences between the RSLogix and handheld programmer addressing notations. Job Aid 1-3 shows some special addressing notations you may encounter when using the RSLogix software.

1-5 Review

- PLCs make it cheaper and easier to make changes to a control system.
- The MicroLogix 1000, like other PLCs, is an industrial computer that controls a machine or process.
- PLCs consist of two basic parts: the CPU and the I/O system.
- All PLCs perform a three-step operation called a scan, which involves reading the inputs, executing the control program stored in memory, and updating the status of the output devices.
- The MicroLogix's control program, which is represented by ladder diagrams, implements the softwired logic connections between the PLC's input and output devices.
- The MicroLogix 1000 uses several different number systems—binary, decimal, hexadecimal, and octal, as well as binary codes—to represent data.
- The MicroLogix comes in two sizes—16 I/O and 32 I/O—and has two types of power supplies—24 VDC and 120/240 VAC.
- The MicroLogix is available with two types of input interfaces—24 VDC or 120 VAC—and three types of outputs—relay, transistor, and triac.
- The I/O structure of the MicroLogix is directly related to the way the memory system is organized.
- The MicroLogix's memory system, which stores all of the information the PLC needs to operate, is divided into four units: file sections, files, words, and bits.
- The MicroLogix 1000 has two file sections: the program file section and the data file section.
- The program file section contains 16 files, while the data file section contains 8 files.
- PLCs use addresses to keep track of their data and to specify which contacts and coils reference which input and output devices.
- A MicroLogix's address has three parts: a file label, a word label, and a bit label.
- Most of the files in the MicroLogix's memory system use the same addressing notation; however, some files use a special word labeling mnemonic.

1-6 Job Aids

Job Aid 1-1: MicroLogix 1000 Memory Structure

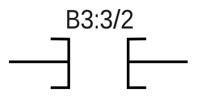
	FILE #	FILE NAME	FUNCTION
Program Section	1	Executive	Controls the function of the PLC and contains processor data: type, configuration, and passwords.
	2	Main Ladder Program	Controls the machine or process.
	3	Error Fault Routine	Executed when a recoverable, or fixable, fault error occurs.
	4	High-Speed Counter Interrupt File	Executed when a high-speed counter instruction causes an interruption in the control program.
	5	Selectable Timed Interrupt Program	Used to interrupt the normal program scan of the MicroLogix so a subroutine can be executed.
	6 ↓ 15	Subroutine Ladder Programs	Called by the main ladder control program. Can store up to ten subroutines.

	FILE #	FILE NAME	FILE LABEL	FUNCTION	NUMBER OF WORD	AVAILABLE ADDRESSES
Data Section	0	Outputs	O	Contains data about the status of each output device connected to the output terminals.	1	0:0
	1	Inputs	I	Contains data about the status of each input device connected to the input terminals.	2	I:0.0 I:0.1
	2	Status	S2	Stores information about how the PLC is operating and how it is set up. Holds three types of data: basic status data, dynamic configuration data and static configurations data.	33	S2:0 ↓ S2:32
	3	Binary	B3	Stores data about the status of internal coils and contacts.	32	B3:0 ↓ B3:31
	4	Timers	T4	Contains data about the timers used in the control program. Keeps data about each timer's status, preset value, and accumulated value in 3 separate words.	40 40 > 120 40	T4:0, T4:0.1, T4:0.2 ↓ T4:39, T4:39.1, T4:39.2
	5	Counters	C5	Contains data about the counters used in the control program. Keeps data about each counter's status, preset value, and accumulated value. Can store up to 32 counters and uses 3 words for each counter.	32 32 > 96 32	C5:0, C5:0.1, C5:0.2 ↓ C5:31, C5:31.1, C5:31.2
	6	Control	R6	Stores information used by specialized instruction. Can hold data for 16 instructions, using 3 words for each instruction.	16 16 > 48 16	R6:0, R6:0.1, R6:0.2 ↓ R6:15, R6:15.1, R6:15.2
	7	Integer	N7	Stores miscellaneous numerical data: constant and variable data used by arithmetic instructions, binary codes, and mask patterns.	105	N7:0 ↓ N7:104

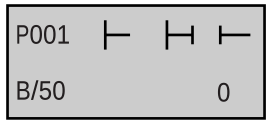
Job Aid 1-2: Differences Between the RSLogix And Handheld Programmer Addressing Notations

The RSLogix software program and a handheld programmer use slightly different notations for displaying an address. The following example illustrates two of these differences.

The address B3:3/2 is shown as follows on an RSLogix screen, depending on the properties settings:



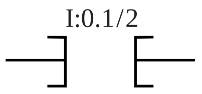
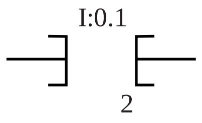
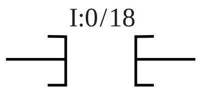
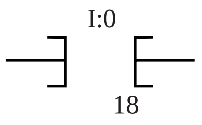
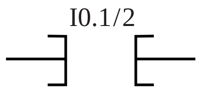
This same address is shown as follows on a handheld programming screen:



Note that the handheld programmer drops the number from the word label (B3). It also expresses the addressed bit according to which bit it is in the total file (50), rather than which bit it is in which word (3/2).

Job Aid 1-3: Addressing Notations Used with the RSLogix Software

An address, such as the address I:0.1/2 can be shown several different ways on an RSLogix programming screen, depending on the way the software's properties are set up (View/Properties/Ladder):

Bit Address Format: [✓ Single Line] Split Line Binary Bit Display Mode: [✓ /Bit] Word/Bit I/O Bit Display Mode: Slot/Bit [✓ Slot.Word/Bit] Short Address: ❑ Display ❑ Entry 	Bit Address Format: Single Line [✓ Split Line] Binary Bit Display Mode: [✓ /Bit] Word/Bit I/O Bit Display Mode: Slot/Bit [✓ Slot.Word/Bit] Short Address: ❑ Display ❑ Entry 
Bit Address Format: [✓ Single Line] Split Line Binary Bit Display Mode: [✓ /Bit] Word/Bit I/O Bit Display Mode: [✓ Slot/Bit] Slot.Word/Bit Short Address: ❑ Display ❑ Entry 	Bit Address Format: Single Line [✓ Split Line] Binary Bit Display Mode: [✓ /Bit] Word/Bit I/O Bit Display Mode: [✓ Slot/Bit] Slot.Word/Bit Short Address: ❑ Display ❑ Entry 
Bit Address Format: [✓ Single Line] Split Line Binary Bit Display Mode: /Bit Word/Bit I/O Bit Display Mode: Slot/Bit Slot.Word/Bit Short Address: ☒ Display ❑ Entry 	Bit Address Format: Single Line [✓ Split Line] Binary Bit Display Mode: /Bit Word/Bit I/O Bit Display Mode: Slot/Bit Slot.Word/Bit Short Address: ☒ Display ❑ Entry