

14. PLC MEMORY

Topics:

- ControlLogix memory types; program and data
- Data types; output, input, status, bit, timer, counter, integer, floating point, etc.
- Memory addresses; words, bits, data files, expressions, literal values and indirect.

Objectives:

- To know the basic memory types available
- To be able to use addresses for locations in memory

14.1 INTRODUCTION

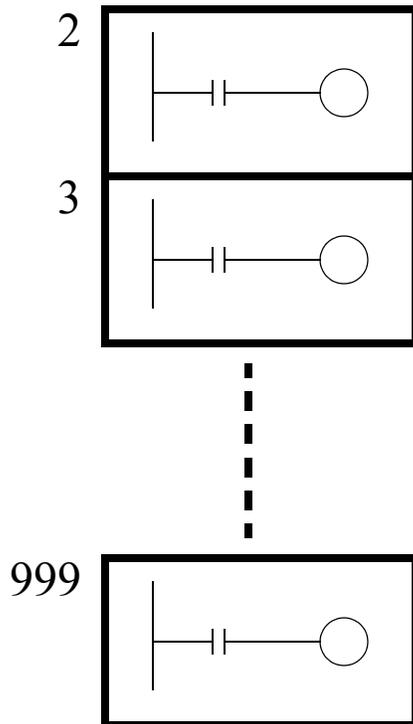
Advanced ladder logic functions such as timers and counters allow controllers to perform calculations, make decisions and do other complex tasks. They are more complex than basic input contacts and output coils and they rely upon data stored in the memory of the PLC. The memory of the PLC is organized to hold different types of programs and data. This chapter will discuss these memory types. Functions that use them will be discussed in following chapters.

14.2 PROGRAM VS VARIABLE MEMORY

The memory in a PLC is divided into program and variable memory. The program memory contains the instructions to be executed and cannot be changed while the PLC is running. (Note: some PLCs allow on-line editing to make minor program changes while a program is running.) The variable memory is changed while the PLC is running. In ControlLogix the memory is defined using variable names (also called tags and aliases).

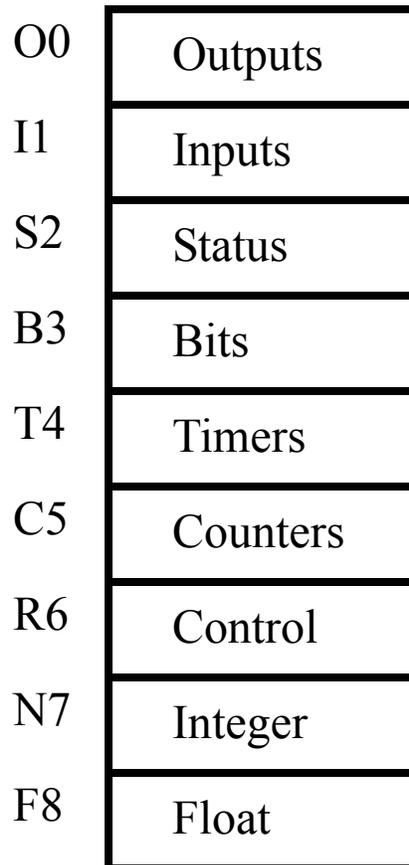
ASIDE: In older Allen Bradley PLCs the memory was often organized as files. There are two fundamental types of memory used in Allen-Bradley PLCs - Program and Data memory. Memory is organized into blocks of up to 1000 elements in an array called a file. The Program file holds programs, such as ladder logic. There are eight Data files defined by default, but additional data files can be added if they are needed.

Program Files



These are a collection of up to 1000 slots to store up to 1000 programs. The main program will be stored in program file 2. SFC programs must be in file 1, and file 0 is used for program and password information. All other program files from 3 to 999 can be used for *subroutines*.

Data Files



This is where the variable data is stored that the PLC programs operate on. This is quite complicated, so a detailed explanation follows.

14.3 PROGRAMS

The PLC has a list of 'Main Tasks' that contain the main program(s) run each scan of the PLC. Additional programs can be created that are called as subroutines. Valid program types include Ladder Logic, Structured Text, Sequential Function Charts, and Function Block Diagrams.

Program files can also be created for 'Power-Up Handling' and 'Controller Faults'. The power-up programs are used to initialize the controller on the first scan. In previous chapters this was done in the main program using the 'S:FS' bit. Fault programs are used to respond to specific failures or issues that may lead to failure of the control system. Normally these programs are used to recover from minor failures, or shut down a system safely.

14.4 VARIABLES (TAGS)

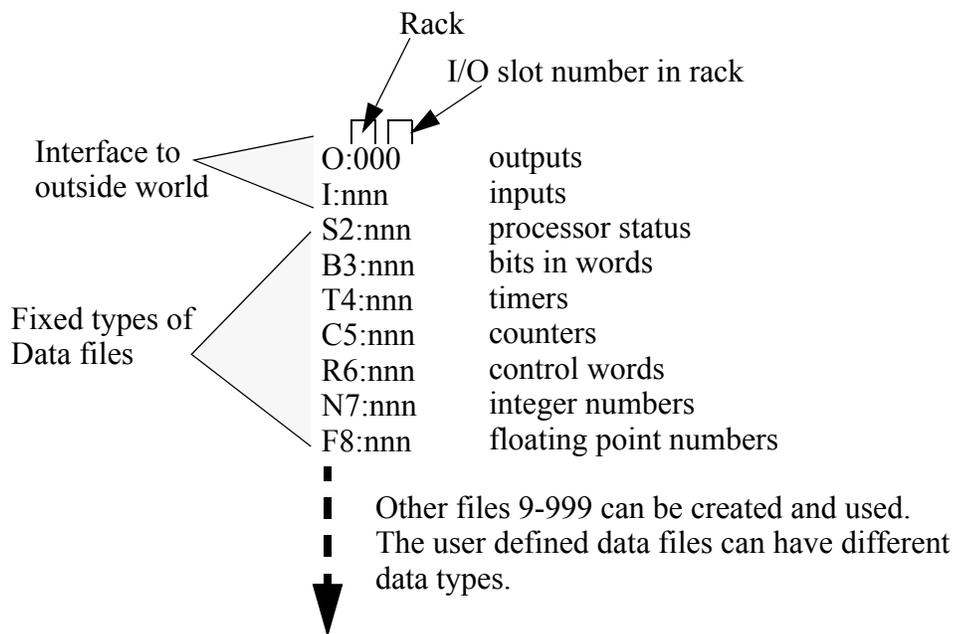
Allen Bradley uses the terminology 'tags' to describe variables, status, and input/output (I/O) values for the controller. 'Controller Tags' include status values and I/O definitions. These are scoped, meaning that they can be global and used by all programs on the PLC. These can also be local, limiting their use to a program that owns it.

Variable tags can be an alias for another tags, or be given a data type. Some of the common tag types are listed below.

Type	Description
BOOL	Holds TRUE or FALSE values
CONTROL	General purpose memory for complex instructions
COUNTER	Counter memory
DINT	32 bit 2s compliment integer -2,147,483,648 to 2,147,483,647
INT	16 bit 2s compliment integer -32,768 to 32,767
MESSAGE	Used for communication with remote devices
PID	Used for PID control functions
REAL	32 bit floating point value +/-1.1754944e-38 to +/-3.4028237e38
SINT	8 bit 2s compliment integer -128 to 127
STRING	An ASCII string
TIMER	Timer memory

Figure 191 Selected ControlLogic Data Types

For older Allen Bradley PLCs data files are used for storing different information types, as shown below. These locations are numbered from 0 to 999. The letter in front of the number indicates the data type. For example, *F8:* is read as *floating point numbers in data file 8*. Numbers are not given for *O:* and *I:*, but they are implied to be *O0:* and *I1:*. The number that follows the *:* is the location number. Each file may contain from 0 to 999 locations that may store values. For the input *I:* and output *O:* files the locations are converted to physical locations on the PLC using rack and slot numbers. The addresses that can be used will depend upon the hardware configuration. The status *S2:* file is more complex and is discussed later. The other memory locations are simply slots to store data in. For example, *F8:35* would indicate the 36th value in the 8th data file which is floating point numbers.



Data values do not always need to be stored in memory, they can be define literally. Figure 192 shows an example of two different data values. The first is an integer, the second is a real number. Hexadecimal numbers can be indicated by following the number with *H*, a leading zero is also needed when the first digit is *A*, *B*, *C*, *D*, *E* or *F*. A binary number is indicated by adding a *B* to the end of the number.

8 - an integer
 8.5 - a floating point number
 08FH - a hexadecimal value *8F*
 01101101B - a binary number *01101101*

Figure 192 Literal Data Values

Data types can be created in variable size 1D, 2D, or 3D arrays.

Sometimes we will want to refer to an array of values, as shown in Figure 193. This data type is indicated by beginning the number with a pound or hash sign '#'. The first example describes an array of floating point numbers starting in file 8 at location 5. The second example is for an array of integers in file 7 starting at location 0. The length of the array is determined elsewhere.

test[1, 4] - returns the value in the 2nd row and 5th column of array test

Figure 193 Arrays

Expressions allow addresses and functions to be typed in and interpreted when the program is run. The example in Figure 194 will get a floating point number from 'test', perform a sine transformation, and then add 1.3. The text string is not interpreted until the PLC is running, and if there is an error, it may not occur until the program is running - so use this function cautiously.

expression - a text string that describes a complex operation.

“sin(test) + 1.3” - a simple calculation

Figure 194 Expressions

These data types and addressing modes will be discussed more as applicable functions are presented later in this chapter and book.

Figure 195 shows a simple example ladder logic with functions. The basic operation is such that while input *A* is true the functions will be performed. The first statement will move (MOV) the literal value of 130 into integer memory *X*. The next move function will copy the value from *X* to *Y*. The third statement will add integers value in *X* and *Y* and store the results in *Z*.

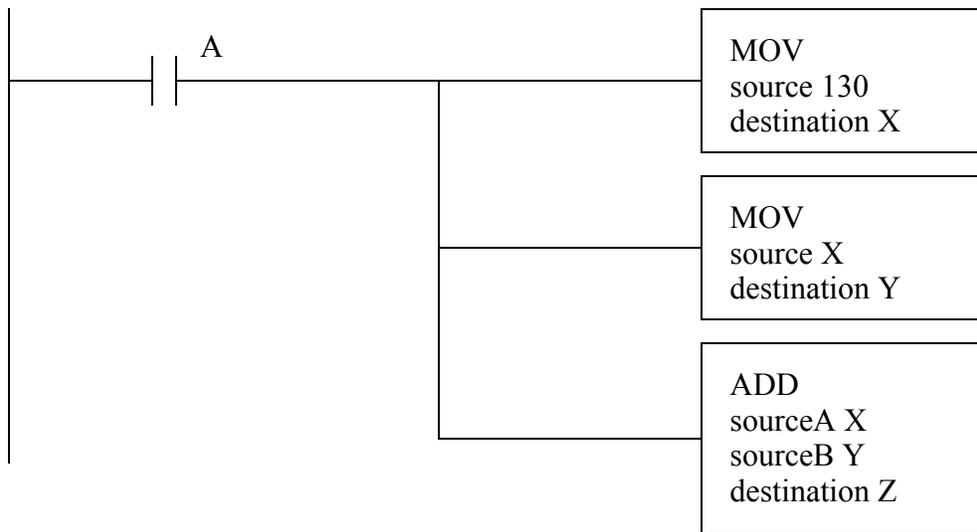


Figure 195 An Example of Ladder Logic Functions

14.4.1 Timer and Counter Memory

Previous chapters have discussed the basic operation of timers and counters. The ability to address their memory directly allows some powerful tools. The bits and words for timers are;

- EN - timer enabled bit
- TT - timer timing bit
- DN - timer done bit
- FS - timer first scan
- LS - timer last scan
- OV - timer value overflowed
- ER - timer error
- PRE - preset word
- ACC - accumulated time word

Counter have the following bits and words.

- CU - count up bit
- CD - count down bit
- DN - counter done bit
- OV - overflow bit
- UN - underflow bit
- PRE - preset word
- ACC - accumulated count word

As discussed before we can access timer and counter bits and words. Examples of these are shown in Figure 196. The bit values can only be read, and should not be changed. The presets and accu-

mulators can be read and overwritten.

Words

timer.PRE - the preset value for timer T4:0
 timer.ACC - the accumulated value for timer T4:0
 counter.PRE - the preset value for counter C5:0
 counter.ACC - the accumulated value for counter C5:0

Bits

timer.EN - indicates when the input to timer T4:0 is true
 timer.TT - indicates when the timer T4:0 is counting
 timer.DN - indicates when timer T4:0 has reached the maximum
 counter.CU - indicates when the count up instruction is true for C5:0
 counter.CD - indicates when the count down instruction is true for C5:0
 counter.DN - indicates when the counter C5:0 has reached the preset
 counter.OV - indicates when the counter C5:0 passes the maximum value (2,147,483,647)
 counter.UN - indicates when the counter C5:0 passes the minimum value (-2,147,483,648)

Figure 196 Examples of Timer and Counter Addresses

Consider the simple ladder logic example in Figure 197. It shows the use of a timer timing *TT* bit to seal on the timer when a door input has gone true. While the timer is counting, the bit will stay true and keep the timer counting. When it reaches the 10 second delay the *TT* bit will turn off. The next line of ladder logic will turn on a light while the timer is counting for the first 10 seconds.

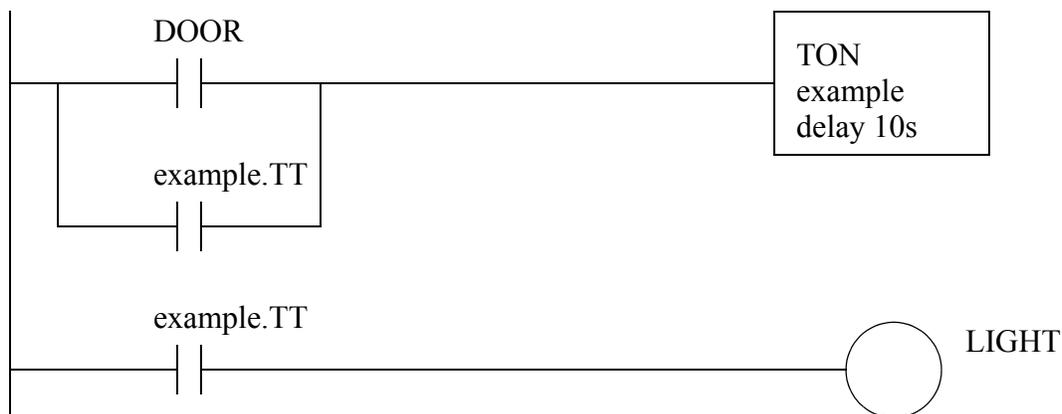


Figure 197 Door Light Example

14.4.2 PLC Status Bits

Status memory allows a program to check the PLC operation, and also make some changes. A selected list of status bits is shown in Figure 198 for Allen-Bradley ControlLogix PLCs. More complete lists are available in the manuals. The first six bits are commonly used and are given simple designations for use with simple ladder logic. More advanced instructions require the use of Get System Value (GSV) and Set System Value (SSV) functions. These functions can get/set different values depending upon the type of data object is being used. In the sample list given one data object is the 'WALLCLOCKTIME'. One of the attributes of the class is the DateTime that contains the current time. It is also possible to use the 'PROGRAM' object instance 'MainProgram' attribute 'LastScanTime' to determine how long the program took to run in the previous scan.

Immediately accessible status values

- S:FS - First Scan Flag
- S:N - The last calculation resulted in a negative value
- S:Z - The last calculation resulted in a zero
- S:V - The last calculation resulted in an overflow
- S:C - The last calculation resulted in a carry
- S:MINOR - A minor (non-critical/recoverable) error has occurred

Examples of SOME values available using the GSV and SSV functions

- CONTROLLERDEVICE - information about the PLC
- PROGRAM - information about the program running
 - LastScanTime
 - MaxScanTime
- TASK
 - EnableTimeout
 - LastScanTime
 - MaxScanTime
 - Priority
 - StartTime
 - Watchdog
- WALLCLOCKTIME - the current time
 - DateTime
 - DINT[0] - year
 - DINT[1] - month 1=january
 - DINT[2] - day 1 to 31
 - DINT[3] - hour 0 to 24
 - DINT[4] - minute 0 to 59
 - DINT[5] - second 0 to 59
 - DINT[6] - microseconds 0 to 999,999

Figure 198 Status Bits and Words for ControlLogix

An example of getting and setting system status values is shown in Figure 199. The first line of ladder logic will get the current time from the class 'WALLCLOCKTIME'. In this case the class does not have an instance so it is blank. The attribute being recalled is the DateTime that will be written to the DINT array time[0..6]. For example 'time[3]' should give the current hour. In the second line the Watchdog time for the MainProgram is set to 200 ms. If the program MainProgram takes longer than 200ms to execute a fault will be generated.

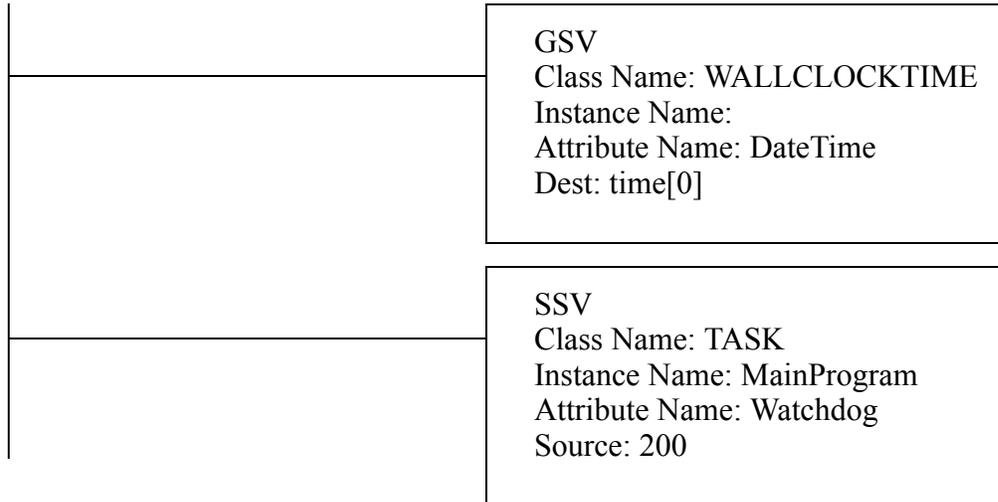


Figure 199 Reading and Setting Status bits with GSV and SSV

As always, additional classes and attributes for the status values can be found in the manuals for the processors and instructions being used.

A selected list of status bits is shown below for Allen-Bradley Micrologic and PLC-5 PLCs. More complete lists are available in the manuals. For example the first four bits *S2:0/x* indicate the results of calculations, including carry, overflow, zero and negative/sign. The *S2:1/15* will be true once when the PLC is turned on - this is the first scan bit. The time for the last scan will be stored in *S2:8*. The date and clock can be stored and read from locations *S2:18* to *S2:23*.

S2:0/0 carry in math operation
S2:0/1 overflow in math operation
S2:0/2 zero in math operation
S2:0/3 sign in math operation
S2:1/15 first scan of program file
S2:8 the scan time (ms)
S2:18 year
S2:19 month
S2:20 day
S2:21 hour
S2:22 minute
S2:23 second
S2:28 watchdog setpoint
S2:29 fault routine file number
S2:30 STI (selectable timed interrupt) setpoint
S2:31 STI file number
S2:46-S2:54,S2:55-S2:56 PII (Programmable Input Interrupt) settings
S2:55 STI last scan time (ms)
S2:77 communication scan time (ms)

14.4.3 User Function Control Memory

Simple ladder logic functions can complete operations in a single scan of ladder logic. Other functions such as timers and counters will require multiple ladder logic scans to finish. While timers and counters have their own memory for control, a generic type of control memory is defined for other function. This memory contains the bits and words in Figure 200. Any given function will only use some of the values. The meaning of particular bits and words will be described later when discussing specific functions.

EN - enable bit
 EU - enable unload
 DN - done bit
 EM - empty bit
 ER - error bit
 UL - unload bit
 IN - inhibit bit
 FD - found bit
 LEN - length word
 POS - position word

Figure 200 Bits and Words for Control Memory

14.5 SUMMARY

- Program are given unique names and can be for power-up, regular scans, and faults.
- Tags and aliases are used for naming variables and I/O.
- Files are like arrays and are indicated with [].
- Expressions allow equations to be typed in.
- Literal values for binary and hexadecimal values are followed by *B* and *H*.

14.6 PRACTICE PROBLEMS

1. How are timer and counter memory similar?
2. What types of memory cannot be changed?
3. Develop Ladder Logic for a car door/seat belt safety system. When the car door is open, or the seatbelt is not done up, a buzzer will sound for 5 seconds if the key has been switched on. A cabin light will be switched on when the door is open and stay on for 10 seconds after it is closed, unless a key has started the ignition power.
4. Write ladder logic for the following problem description. When button *A* is pressed a value of 1001 will be stored in *X*. When button *B* is pressed a value of -345 will be stored in *Y*, when it is not pressed a value of 99 will be stored in *Y*. When button *C* is pressed *X* and *Y* will be added, and the result will be stored in *Z*.
5. Using the status memory locations, write a program that will flash a light for the first 15 seconds after it has been turned on. The light should flash once a second.
6. How many words are required for timer and counter memory?
7. A machine is being designed for a foreign parts supplier. As part of the contractual agreement the logic will run until February 26, 2008. However, after that date the machine will enable a 'contract_expired' value and no longer run. Write the ladder logic.

14.7 ASSIGNMENT PROBLEMS

1. Could timer 'T' and counter 'C' memory types be replaced with control 'R' memory types? Explain your answer.