

# System Operation

---

## In This Chapter. . . .

- Introduction
  - CPU Operating System
  - Initial Mode Setting and Memory Initialization
  - Program Mode Operation
  - Run Mode Operation
  - I/O Response Time
  - CPU Scan Time Considerations
  - Memory Map
  - I/O Point Bit Map
  - Control Relay Bit Map
  - Special Relays
  - Timer / Counter Registers and Contacts
  - Data Registers
  - Stage Control / Status Bit Map
  - Shift Register Bit Map
  - Special Registers
-

## Introduction

Achieving the proper control for your equipment or process requires a good understanding of how the DL305 CPUs control all aspects of the system operation. This includes many things, such as I/O updates, program execution, etc. Take a few minutes to understand how the CPU stores and processes information. For more complex applications, this knowledge will make it easier to program and debug your application program to meet your system performance requirements.

There are four primary things you need to understand before you create your application program

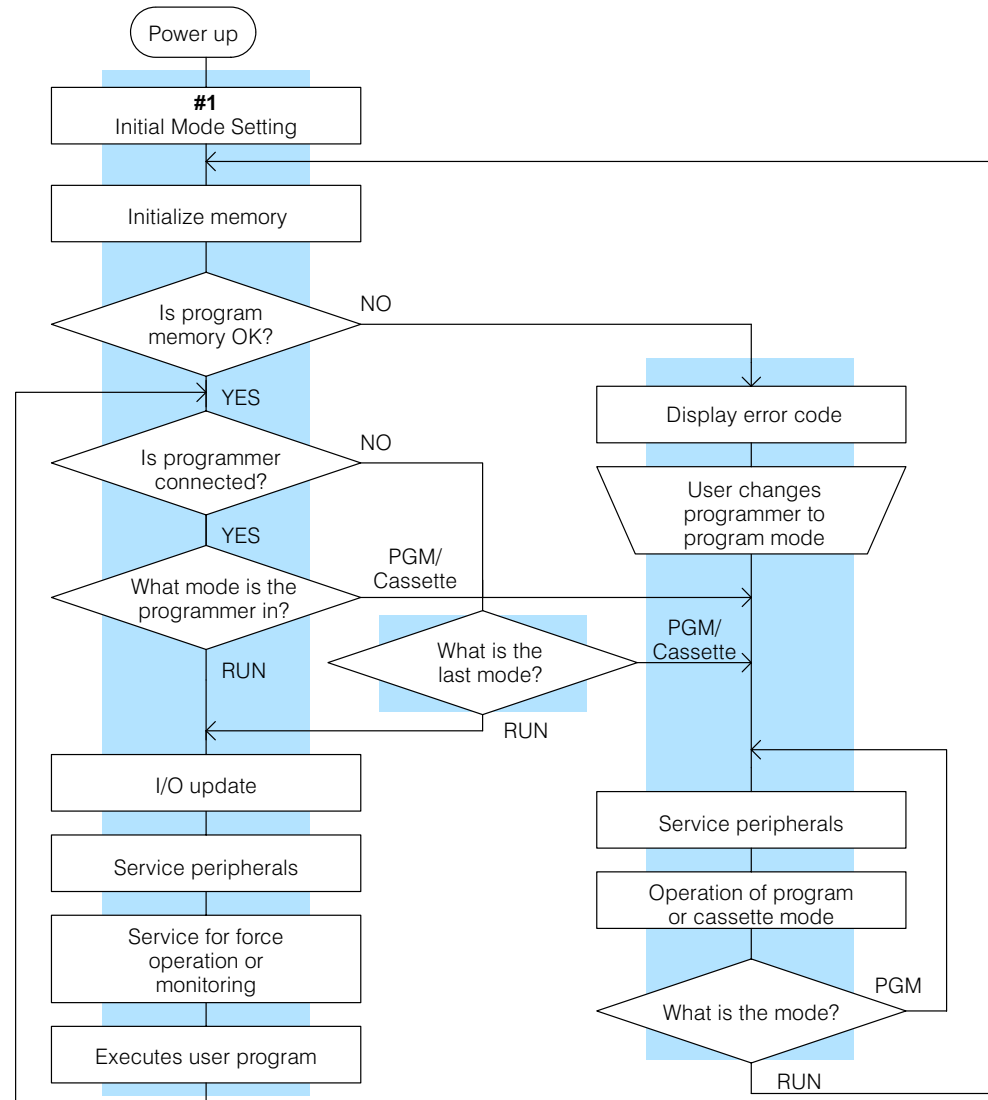
- **CPU Operating System** — the CPU is the heart of the system. It manages all aspects of system control. A quick overview of all the steps is provided in the next section.
- **CPU Operating Modes** — the CPU has different operating modes that allow different types of operations. There are two primary modes of operation, Program Mode and Run Mode.
- **CPU Timing** — it is important you understand how the CPU timing can affect the system operation. The two most important areas are the I/O response time and the CPU scan time.
- **CPU Memory Map** — The DL305 CPUs offer a wide variety of memory types, such as timers, counters, inputs, etc. It is important to understand what memory types are available and how the memory areas are organized.

The remainder of this chapter discusses these items in detail.

## CPU Operating System

After the initial power-up sequence, the DL305 CPUs process data cyclically. There are several tasks the CPU must perform during each cycle, such as updating the I/O status, servicing external communications, executing the application program, etc. There are many different segments of execution, but the overwhelming majority of your concerns will be with the portion of the execution cycle that occurs during Run Mode. These operations will be discussed throughout the remainder of this chapter.

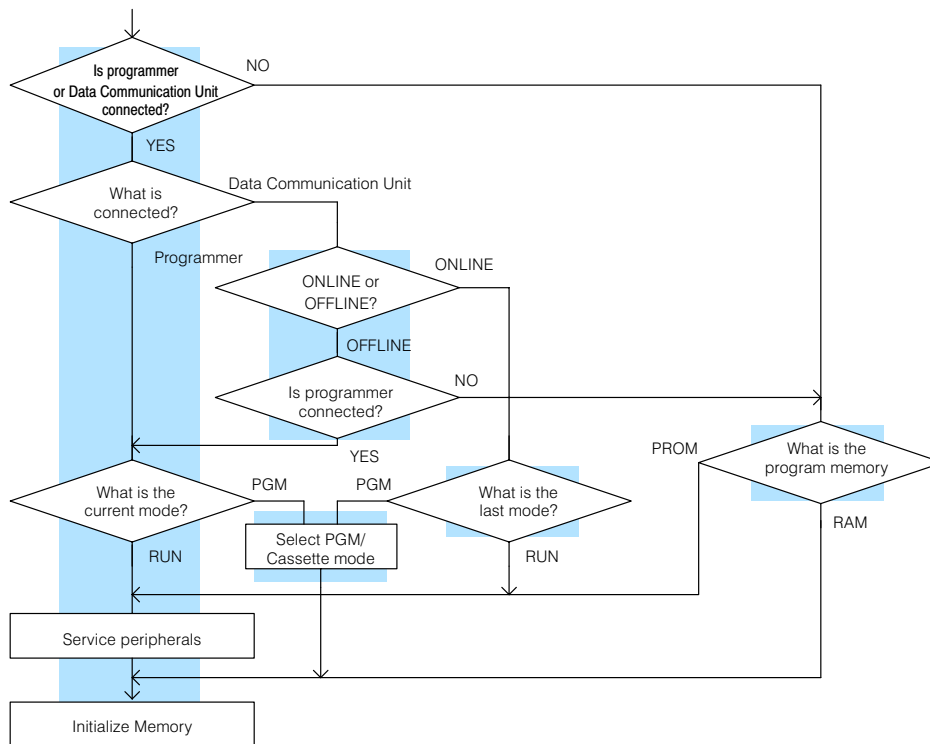
**DL305 CPU  
Operational Flow  
Chart**



## Initial Mode Setting and Memory Initialization

### Flow Chart for Initial Mode Setting (#1)

The previous flowchart contained a step called Initial Mode Setting. Once power has been connected to the system, the CPU executes the following procedure to determine which mode of operation should be entered.



**Memory  
Initialization**

Before the CPU can begin normal operation, all memory areas are initialized. The CPU completes the following operations to initialize the memory areas.

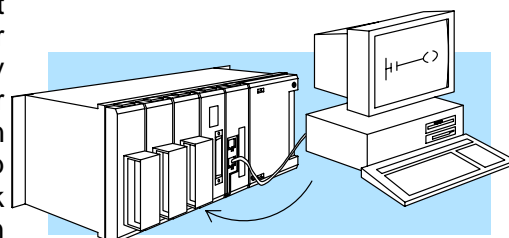
Item	Procedure
I/O Points	Input Points: activated, CPU will monitor status. Output Points: turned off
Control Relays	Non-Retentive: turned off Retentive: retains the condition prior to the system power interruption.
Shift Registers	Retains the condition prior to the system power interruption.
Timer / Counter Current Values	Timers: reset to zero Counters: retains the current count prior to the system power interruption.
Stages	Non-Retentive: turned off Retentive: retains the condition prior to the system power interruption.
Data Registers	Retains the condition prior to the system power interruption.

**NOTE:** Not all memory areas are retentive by default. See Chapter 3 for details on how to set the CPU to have retentive memory. Also, the memory map section provided later in this chapter shows the exact ranges that can be selected as retentive.

## Program Mode Operation

In Program Mode, the CPU does not execute the application program or update the output modules. The primary use for Program Mode is to enter or change an application program. You can also use the program mode to set up CPU parameters, such as the network address for the communication port on the DL340, retentive memory areas, etc.

You can use the Handheld Programmer key switch to select Program Mode operation, or you can use a **DirectSOFT** menu option to change the CPU mode.



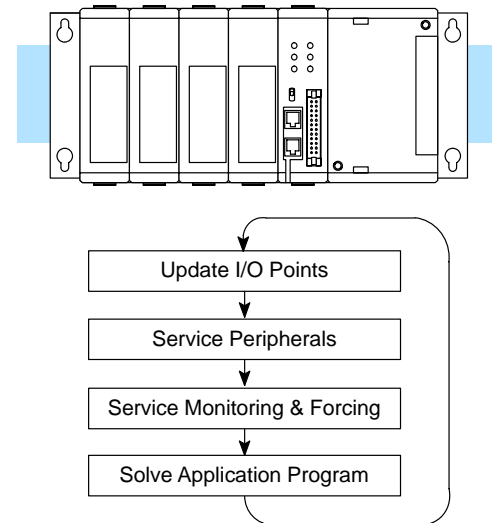
Download Program

## Run Mode Operation

In Run Mode, the CPU executes the application program and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

- Monitor and change I/O point status
- Update timer/counter preset values
- Update Register memory locations

Even though there are many steps in the overall flowchart of operation, the Run Mode operation can be divided into a few key areas. It is very important you understand how each of these areas of execution can affect the results of your application program solutions.

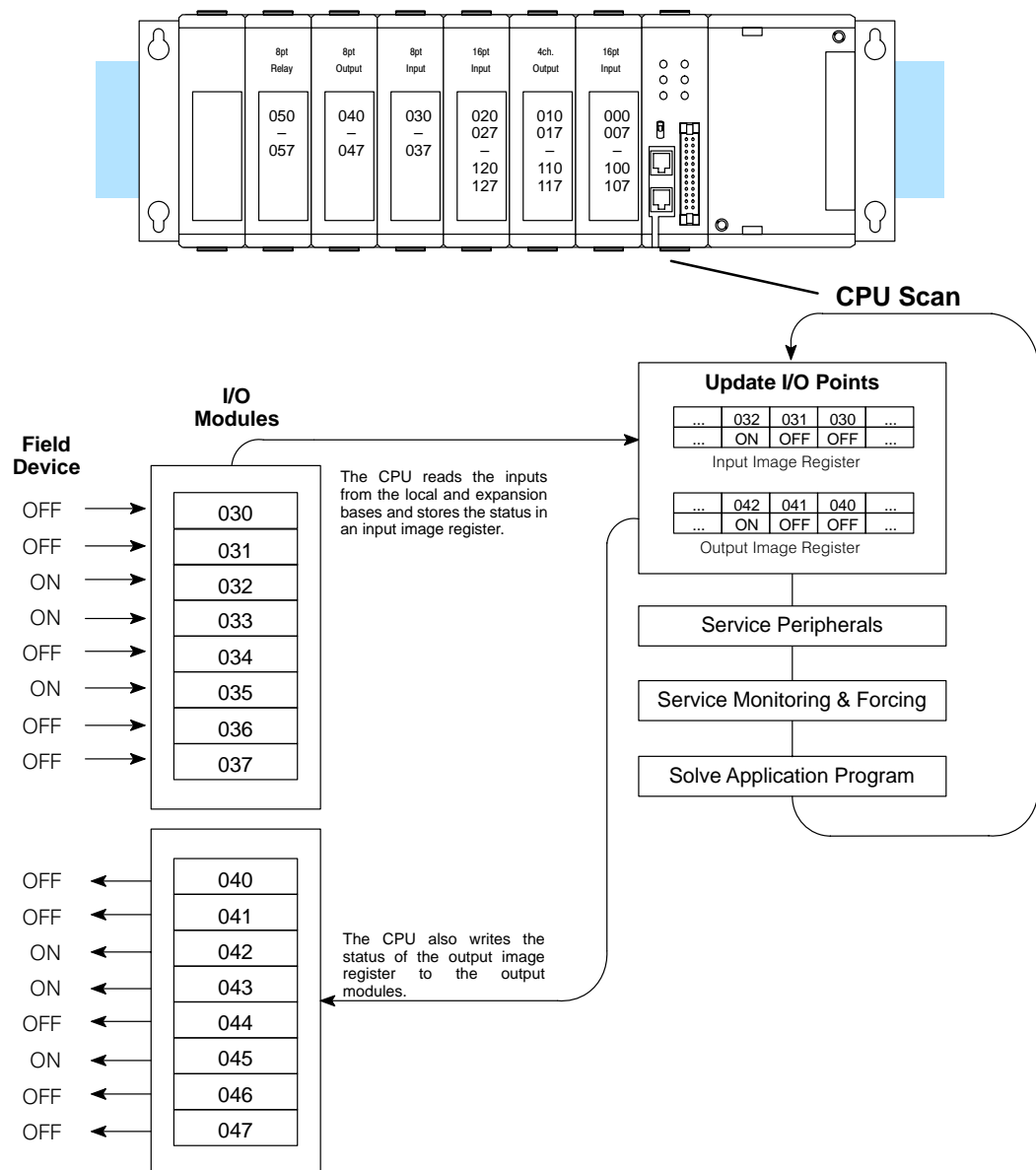


**Update I/O Points**

The CPU reads the status of all input modules present in the local CPU base and local expansion bases. The status of each input is stored in the input image register area. The input image register data is used by the CPU when it solves the application program.

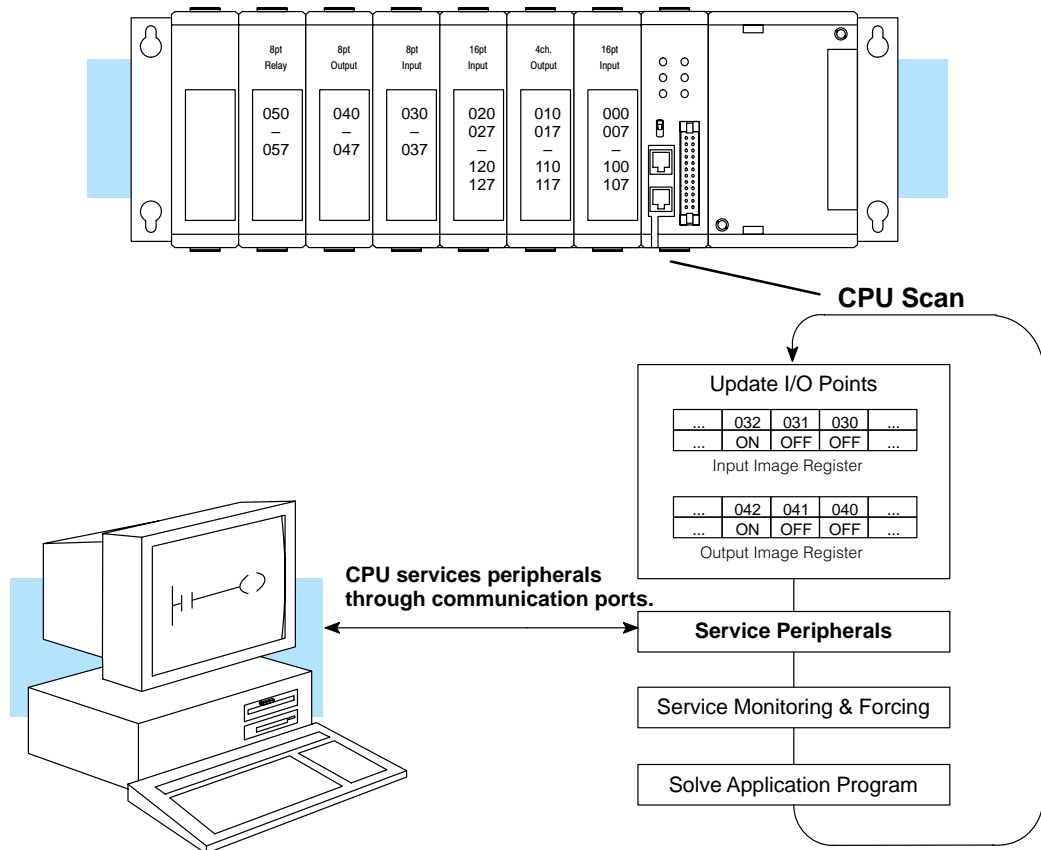
You may be thinking, “What if an input changes *after* the CPU has read the inputs?” Good question! Generally, the CPU program execution time is measured in milliseconds, so in most cases this is not a problem. If you need to know more, I/O response timing is explained in more detail later in this chapter.

The CPU also uses the output image register to update the output modules present in the local CPU base and local expansion bases.





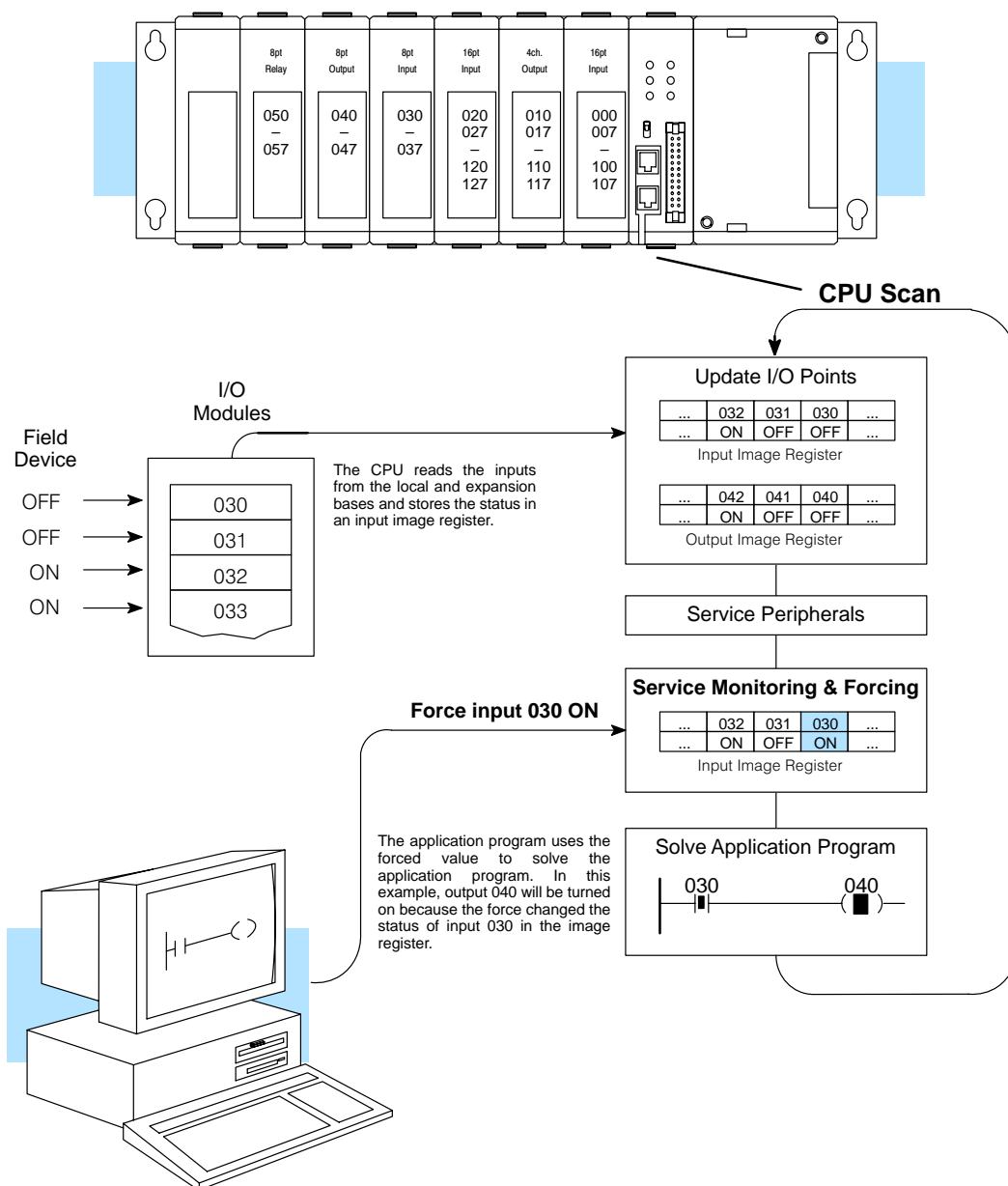
**Service Peripherals** After the CPU updates the I/O points, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, if you were using an operator interface to read or write data, the CPU would service these requests during this portion of the scan.



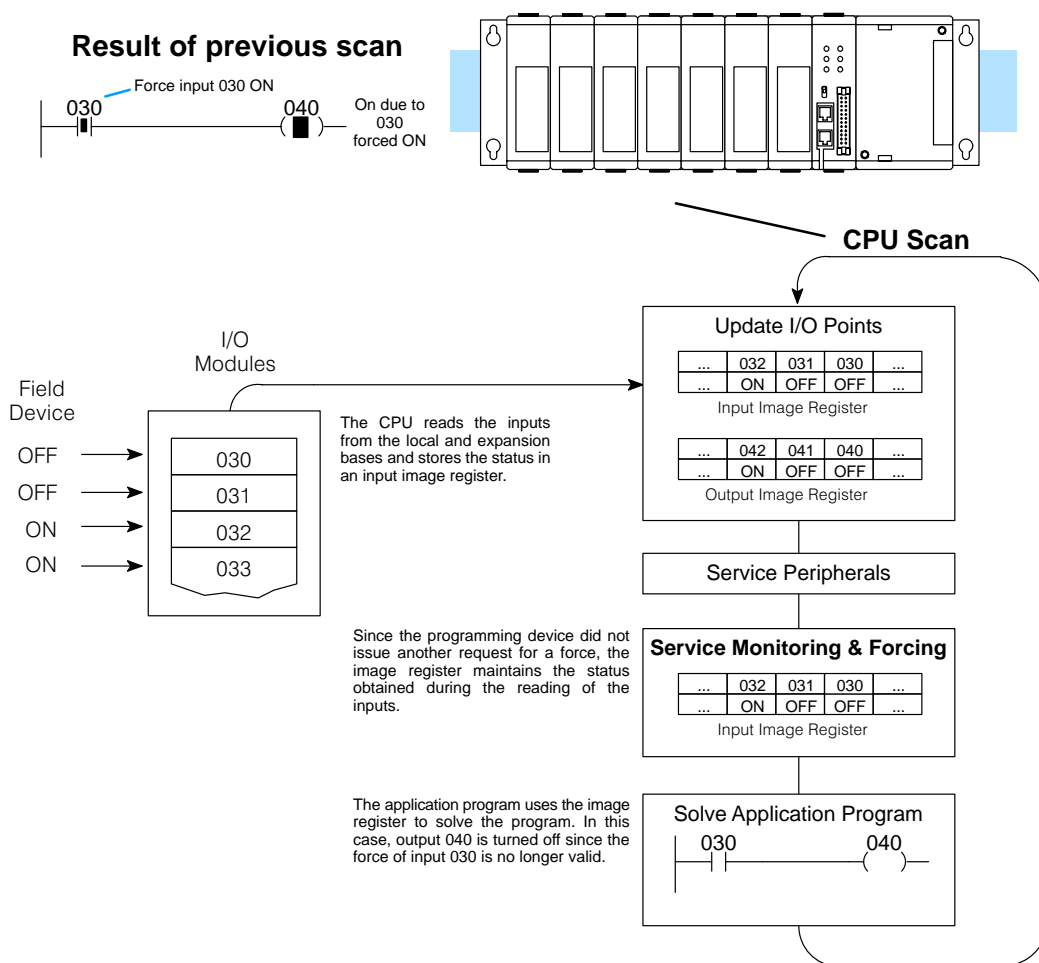
## Service for Monitoring and Forcing Operations

After the CPU updates any communications requests from peripheral devices, it determines if any forcing operations have been requested. The CPU also services any monitoring requests during this portion. For example, if you are using the Handheld Programmer to monitor the current value of a timer or counter, the CPU will provide this information during this portion of the scan.

Here's an example of one of the more popular requests. For example, you may want to force an input on, even though it is really off. This allows you to change the point status that was stored in the image register. This value will be valid until the image register location is written to during the Update I/O Points segment of the next scan.



It is important to note the DL305 CPUs only retain the forced value for one scan if the input point used corresponds to a module that is installed in the base. The following example shows how the forcing actually works *on the next CPU scan*.



As you can see from the example, the input forcing will not be valid when the CPU reads the input status on the next scan.

Output point forcing works in a similar manner. That is, if you force an output on and the application program results dictate the output should be turned on, then the output image register will show the results of the application program instead of the forcing request. This is discussed in more detail in the next section.

**Solve Application Program**

The CPU evaluates each instruction in the application program during this segment of the cycle. The instructions define the relationship between the input conditions and the desired output response.

The CPU uses the output image register area to store the status of the desired action for the outputs. The actual outputs are updated during the Update I/O Points segment of the cycle.

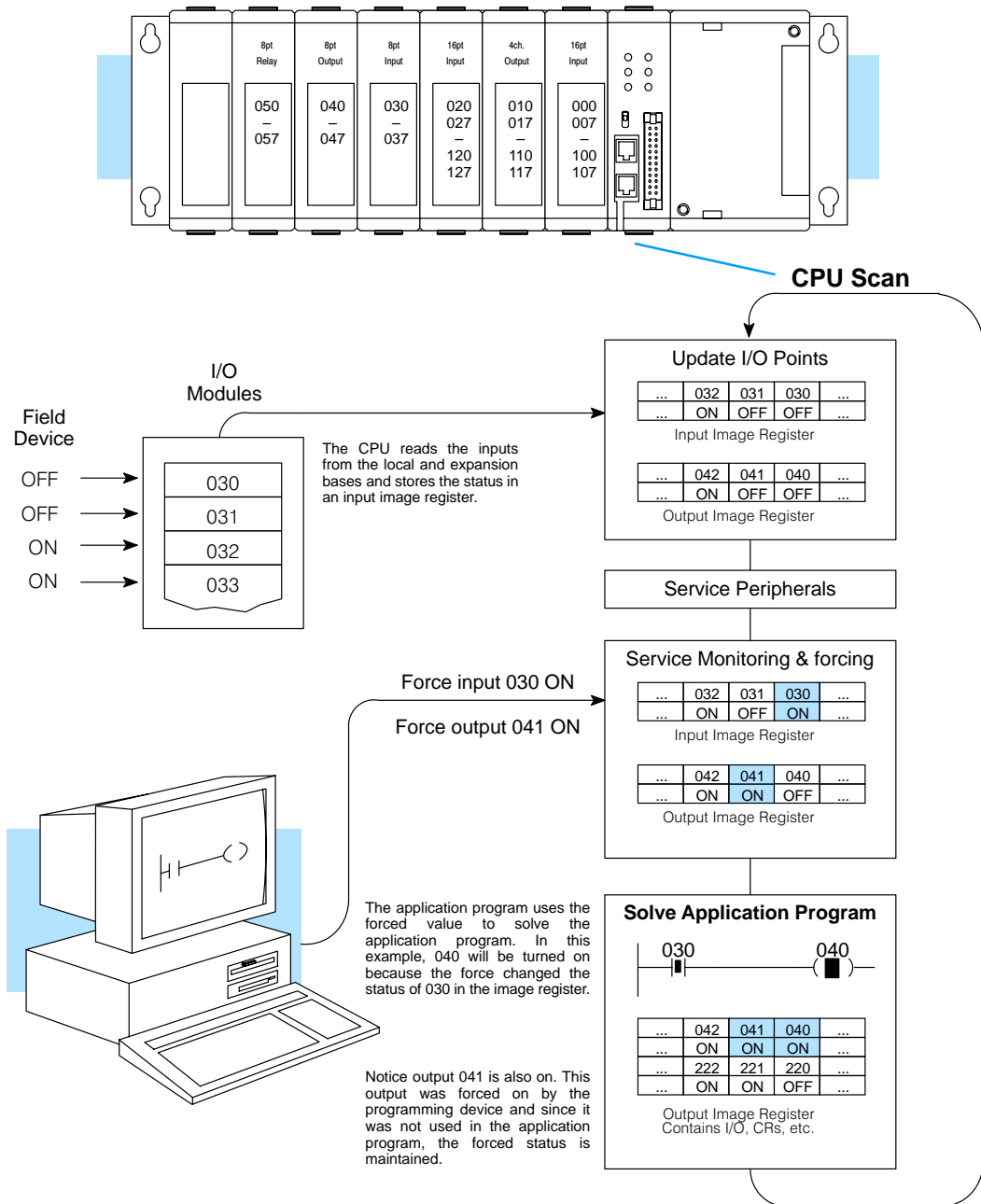
The internal control relays (C), stages (S), and data registers (R) are also updated in this segment.

You may recall the CPU may have obtained and stored forcing information when it serviced any peripheral devices. If any output points or register locations have been forced, the output image register also contains this information.

---

**NOTE:** If an output point was used in the application program, the results of the program solution will overwrite any forcing information that was stored. For example, if output 030 was forced on by the programming device, and a rung containing 030 was evaluated such that 030 should be turned off, then the output image register will show that 030 should be off. Of course, you can force output points that are not used in the application program. In this case, the point remains forced because there is no solution that results from the application program execution.

---



## I/O Response Time

### Is Timing Important for Your Application?

I/O response time is the amount of time required for the control system to sense a change in an input point and update a corresponding output point. In the majority of applications, the CPU performs this task in such a short period of time you may never have to concern yourself with the aspects of system timing. However, some applications do require extremely fast update times. In these cases, you may need to know how to determine the amount of time spent during the various segments of operation.

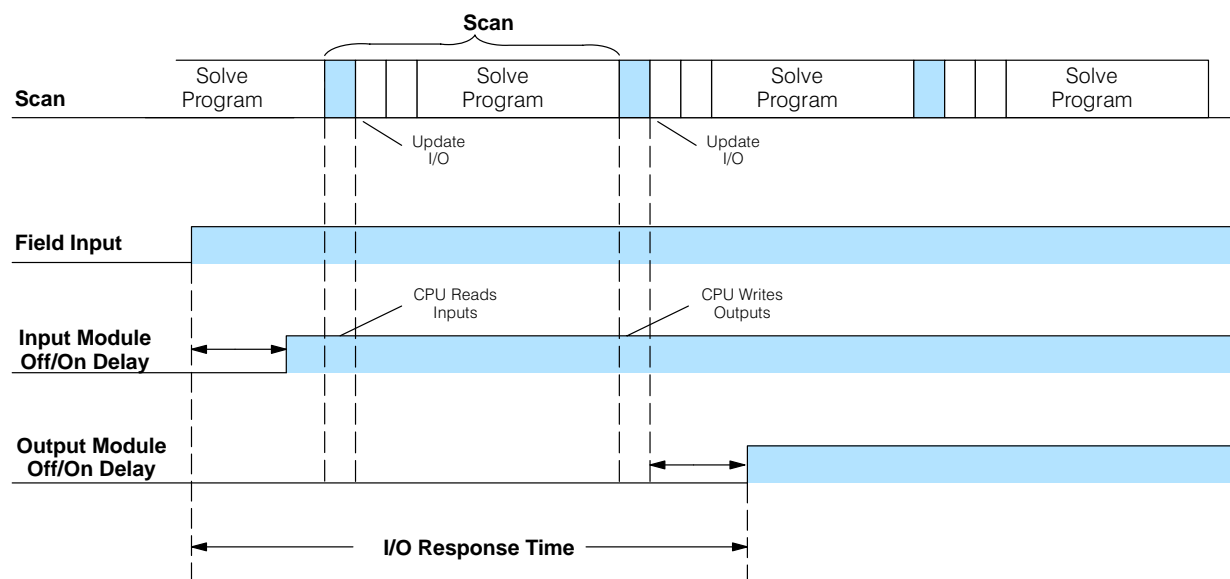
There are four things that can affect the I/O response time.

- The point in the cycle when the field input changes states
- Input module Off to On delay time
- CPU scan time
- Output module Off to On delay time

The next paragraphs show how these items interact to affect the response time.

### Normal Minimum I/O Response

The I/O response time is shortest when the module senses the input change just before the I/O Update portion of the execution cycle. In this case the input status is read, the application program is solved, and the output point gets updated on the following scan. The following diagram shows an example of the timing for this situation.

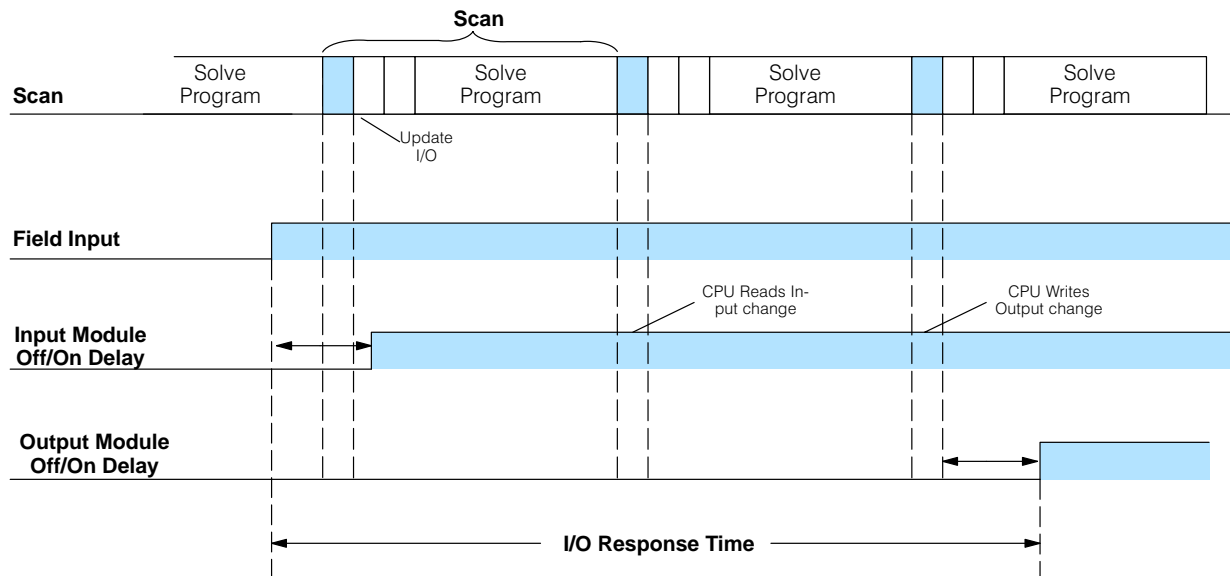


In this case, you can calculate the response time by simply adding the following items.

$$\text{Input Delay} + \text{Scan Time} + \text{Output Delay} = \text{Response Time}$$

### Normal Maximum I/O Response

The I/O response time is longest when the module senses the input change just after the Update I/O portion of the execution cycle. In this case the new input status does not get read until the following scan. The following diagram shows an example of the timing for this situation.



In this case, you can calculate the response time by simply adding the following items.

$$\text{Input Delay} + (2 \times \text{Scan Time}) + \text{Output Delay} = \text{Response Time}$$

### Improving Response Time

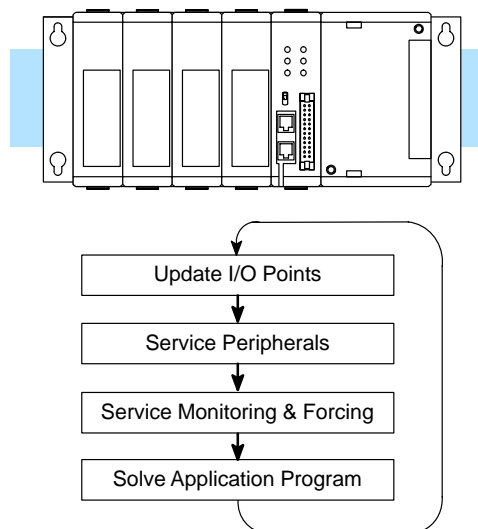
There are a few things you can do to help improve throughput.

- You can try to choose instructions with faster execution times
- You can choose modules that have faster response times

## CPU Scan Time Considerations

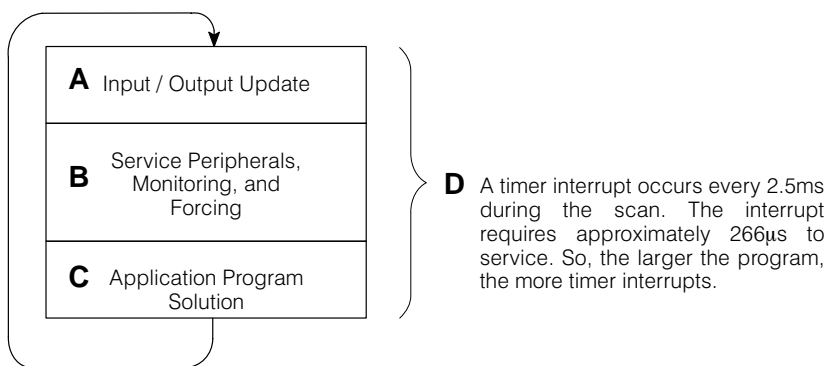
The scan time covers all the cyclical tasks that are performed by the operating system. This information can be very important when evaluating the performance of a system.

As we've shown previously there are several segments that make up the overall execution cycle. Each of these segments requires a certain amount of time to complete. Of all the segments, the only ones you really need to understand are those that occur during Run Mode. Even within this portion, your primary concern should be to understand the instruction execution times.



### DL330 / DL330P Scan Calculation

The following table provides execution timing guidelines for the execution cycle.



**A** = 3.3 ms typical I/O update

**B** = 0 - 5.2 ms maximum to service peripherals, monitoring, and forcing

**C** = Total of instruction execution time

$$D = 266\mu s \times \frac{A + B + C + D}{2.5ms}$$

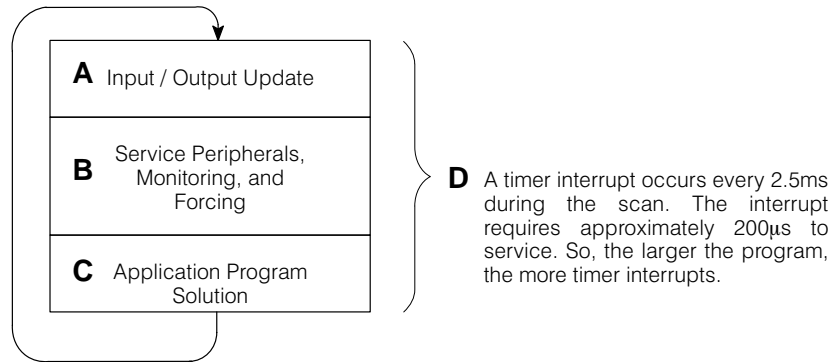
**Actual Scan** = A + B + C + D

**NOTE:** There are other events that occur during the execution cycle, but the areas shown are the most important. This information is provided so you will understand the basic scan calculations. Scan time can vary from scan-to-scan.



**DL340 Scan Calculation**

Typical scan overhead is from 2.5 – 3.5ms. However, the following table provides more precise execution timing guidelines for the execution cycle.



**A** = 2 ms typical I/O update

**B** = 0 – 1.2 ms maximum to service peripherals, monitoring, and forcing

**C** = Total of instruction execution time

$$D = 200 \mu s \times \frac{A + B + C + D}{2.5ms}$$

**Actual Scan** = A + B + C + D

**NOTE:** There are other events that occur during the execution cycle, but the areas shown are the most important. This information is provided so you will understand the basic scan calculations. Scan time can vary from scan-to-scan.

### Application Program Execution

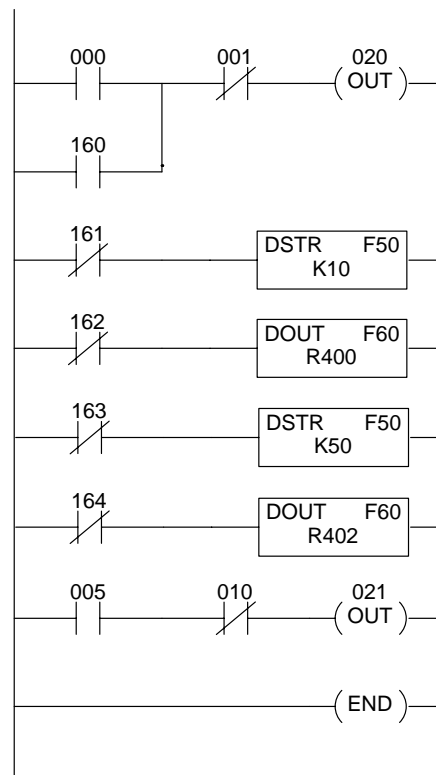
The CPU processes the program from address 0 to the END instruction. The CPU executes the program left to right and top to bottom. As each rung is evaluated the appropriate image register or memory location is updated.

The time required to solve the application program depends on the type and number of instructions used.

You can add the execution times for all the instructions in your program to determine how much time is required to execute the instructions.

For example, the execution time for a DL330 running the program shown would be calculated as follows.

Instruction	Time
STR 000	6.6μs
OR 160	6.6μs
ANDN 001	8.4μs
OUT 020	7.5μs
STRN 161	9.1μs
DSTR K10	14.3μs
STRN 162	9.1μs
DOUT R400	52.6μs
STRN 163	9.1μs
DSTR K50	14.3μs
STRN 164	9.1μs
DOUT R402	52.6μs
STR 005	6.6μs
ANDN 010	8.4μs
OUT 021	7.5μs
END	~0μs
<b>TOTAL</b>	<b>221.8μs</b>



**NOTE:** Appendix C provides the instruction execution times for the DL305 CPUs.

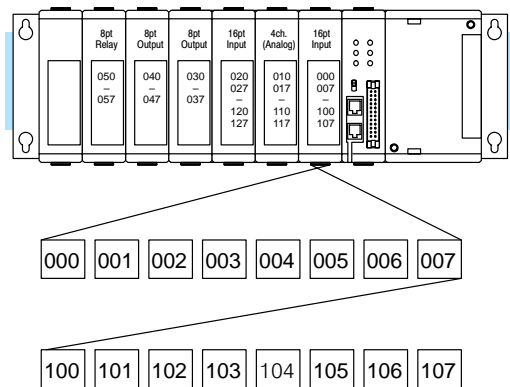
# Memory Map

With any PLC system, you generally have many different types of information to process. This includes input device status, output device status, various timing elements, parts counts, etc. It is important to understand how the system represents and stores the various types of data. For example, you need to know how the system identifies input points, output points, data words, etc. The following paragraphs discuss the various memory types used in the DL305 CPUs.

**NOTE:** The DL305 CPUs do not all have the same memory ranges. Make sure you review the detailed memory maps at the end of this section to determine the available memory types for your particular model of CPU.

## Octal Numbering System

All memory locations or areas are numbered in Octal (base 8). For example, the diagram shows how the octal numbering system works for the discrete input points. Notice the octal system does not contain any numbers with the digits 8 or 9.

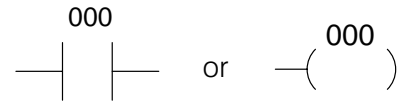


## Two Basic Memory Types: Discrete and Word

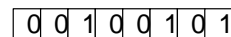
As you examine the different memory types, you'll notice two types of memory in the DL305, discrete and word memory. Discrete memory is one bit that can be either a 1 or a 0. Word memory is referred to as Data Register memory and is an 8-bit location normally used to manipulate data/numbers, store data/numbers, etc.

Some information is automatically stored in Register (R) memory. For example, the timer current values are stored in Registers that correspond to the timer or counter number. So, the current value for timer T600 is automatically stored in R600.

### Discrete – On or Off, 1 bit



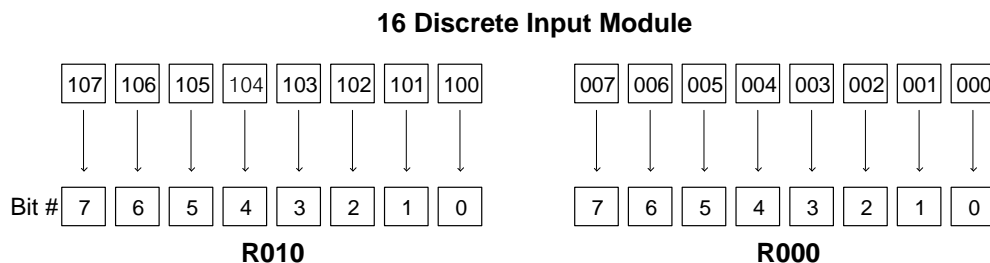
### Word Locations – 8 bits



## R Memory Locations for Discrete Memory Areas

The discrete memory area is for inputs, outputs, control relays, etc. However, you can also access the bit data types as an R-memory word. Each R-memory location contains 8 consecutive discrete locations.

Remember, the DL305 system does not have a separate memory type for input and output points. The type of point assigned to the location depends on the type of module installed in the slot that corresponds to the register location. Also, the number of registers assigned to the module depends on the number of points. For example, a 16-point module would require two registers since each register only contains 8 bits. The following diagram shows how the points for a 16-point module installed in the slot next to the CPU would map into registers.



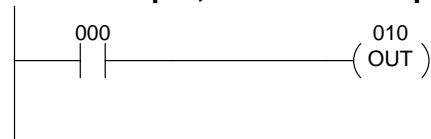
These discrete memory areas and their corresponding R-memory ranges are listed in the memory area tables at the end of this chapter.

## I/O Points

The discrete input and output points do not have separate data types. The type of point assigned to the reference address depends on the type of module installed in the base. Depending on the type of CPU, you can have up to 184 I/O points in a DL305 system.

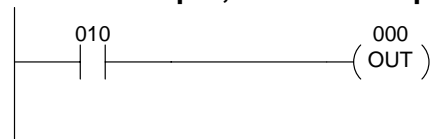
In the first example, output point 010 will be turned on when input 000 energizes. This assumes an 8-point input module is installed in the first slot and an 8-point output module is installed in the second slot.

### 1st Slot – Input, 2nd Slot – Output



The second example shows how the numbers can represent a different type of point. For this example, the module positions were reversed.

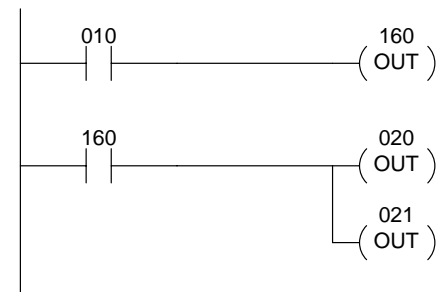
### 1st Slot – Output, 2nd Slot – Input



**NOTE:** Unused I/O references can be used as control relays in the application program.

## Control Relays

Control relays are discrete bits normally used to control the user program. The control relays do not represent a real world device, that is, they cannot be physically tied to switches, output coils, etc. They are internal to the CPU. Because of this, control relays can be programmed as discrete inputs or discrete outputs. These locations are used in programming the discrete memory locations (C) or the corresponding word location which contains 8 consecutive discrete locations.



In this example, memory location 160 will energize when input 010 turns on. The second rung shows a simple example of how to use a control relay as an input.

---

**NOTE:** Some of the references normally assigned as Control Relays can also be used to refer to a 16-point I/O modules in some situations. Make sure you review the memory maps at the end of this chapter if you use 16-point modules.

---