CHAPTER 3

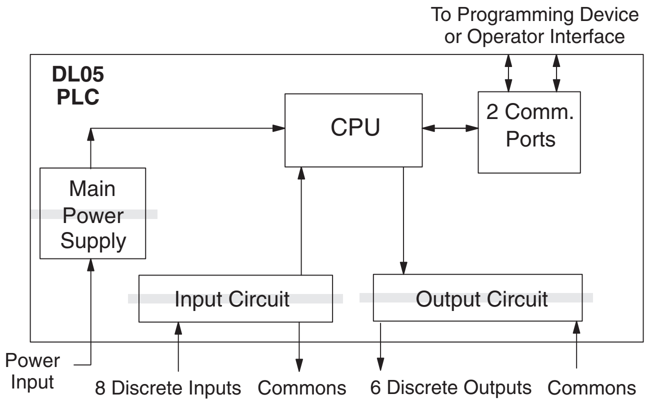
CPU Specifications and Operation

In This Chapter...

Introduction

The Central Processing Unit (CPU) is the heart of the Micro PLC. Almost all PLC operations are controlled by the CPU, so it is important that it is set up correctly. This chapter provides the information needed to understand:

- Steps required to set up the CPU
- Operation of ladder programs
- Organization of Variable Memory



NOTE: *The High-Speed I/O function (HSIO) consists of dedicated but configurable hardware in the DL05. It is not considered part of the CPU, because it does not execute the ladder program. For more on HSIO operation, see Appendix E.*

DL05 CPU Features

The DL05 Micro PLC which has 6K words of memory comprised of 2K of ladder memory and 4K words of V-memory (data registers). Program storage is in the FLASH memory which is a part of the CPU board in the PLC. In addition, there is RAM with the CPU which will store system parameters, V-memory, and other data which is not in the application program. The RAM is backed up by a “super-capacitor”, storing the data for several hours in the event of a power outage. The capacitor automatically charges during powered operation of the PLC.

The DL05 supports fixed I/O which includes eight discrete input points and six output points. If more than the fourteen fixed I/O points are needed, select an I/O module for your application from the DL05/06 Option Modules User Manual. This module will plug into the expansion slot.

Over 120 different instructions are available for program development as well as extensive internal diagnostics that can be monitored from the application program or from an operator interface. Chapters 5, 6, and 7 provide detailed descriptions of the instructions.

The DL05 provides two built-in RS232C communication ports, so you can easily connect a handheld programmer, operator interface, or a personal computer without needing any additional hardware.

CPU Specifications

Specifications	
Feature	**DL05**
Total Program memory (words)	6K
Ladder memory (words)	2048
Total V-memory (words)	4096
User V-memory (words)	3968
Non-volatile V-Memory (words)	128
Contact execution (boolean)	2.0 us
Typical scan (1k boolean)	2.7–3.2 ms
RLL Ladder Style Programming	Yes
RLL and RLL^{PLUS} Programming	Yes
Run Time Edits	Yes
Supports Overrides	Yes
Scan	Variable / fixed
Handheld programmer	Yes
*Direct*SOFT programming for Windows.	Yes
Built-in communication ports (RS232C)	Yes
FLASH Memory	Standard on CPU
Local Discrete I/O points available	14
Local Analog input / output channels maximum	None
High-Speed I/O (quad., pulse out, interrupt, pulse catch, etc.)	Yes, 2
I/O Point Density	8 inputs, 6 outputs
Number of instructions available (see Chapter 5 for details)	129
Control relays	512
Special relays (system defined)	512
Stages in RLL^{PLUS}	256
Timers	128
Counters	128
Immediate I/O	Yes
Interrupt input (external/timed)	Yes
Subroutines	Yes
For/Next Loops	Yes
Math	Integer
Drum Sequencer Instruction	Yes
Time of Day Clock/Calendar	Only with the optional Memory Cartridge
Internal diagnostics	Yes
Password security	Yes
System error log	No
User error log	No
Battery backup	No (built–in super–cap) Yes, with memory cartridge

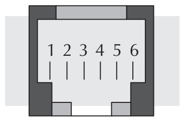
CPU Hardware Setup

Communication Port Pinout Diagrams

Cables are available that allow you to quickly and easily connect a Handheld Programmer or a personal computer to the DL05 PLCs. However, if you need to build your own cables, use the pinout information shown below. The DL05 PLCs require an RJ-12 phone plug to fit the built-in jacks.

The Micro PLC has two built-in RS232C communication ports. Port 1 is generally used for connecting to a D2-HPP, a PC with ***Direct***SOFT, operator interface, Modbus slave, or a ***Direct***NET slave. The baud rate is fixed at 9600 baud. Port 2 can be used to connect to a D2-HPP, ***Direct***SOFT, operator interface, Modbus master/slave, or a ***Direct***NET master/slave. Port 2 has a range of speeds from 300 baud to 38.4K baud.

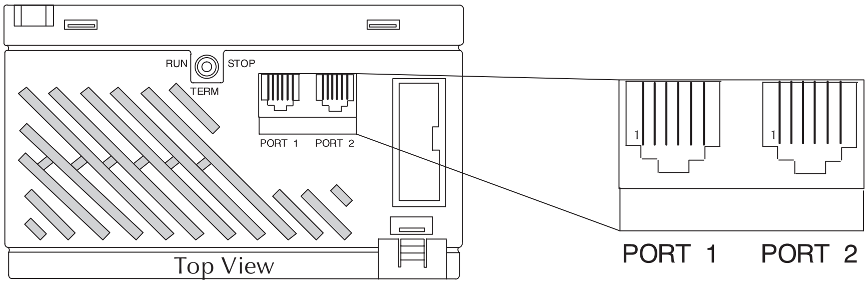
NOTE: *The 5V pins are rated at 220mA maximum, primarily for use with some operator interface units.*



6-pin Female Modular Connector

Port 1 Pin Descriptions		
1	0V	Power (–) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive Data (RS232C)
4	TXD	Transmit Data (RS232C
5	5V	Power (+) conection
6	0V	Power (–) connection (GND)

Port 2 Pin Descriptions		
1	0V	Power (–) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive Data (RS232C)
4	TXD	Transmit Data (RS232C
5	RTS	Request to Send
6	0V	Power (–) connection (GND)

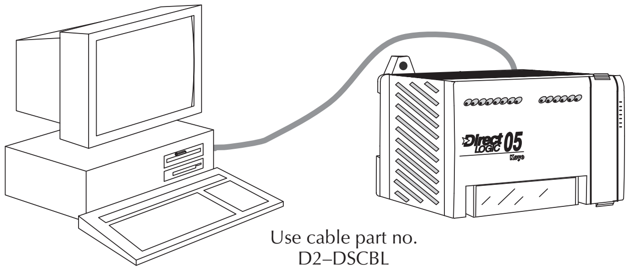


Communication Port 1	
Com 1	Connects to HPP, ***Direct***SOFT, operator interfaces, etc. 6-pin, RS232C 9600 Baud (Fixed) Parity - odd (default) Station address 1 (fixed) 8 data bits 1 start, 1 stop bit Asynchronous, Half-duplex, DTE Protocol: (Auto-Select) K sequence (Slave only) ***Direct***NET (Slave only) Modbus RTU (Slave only)

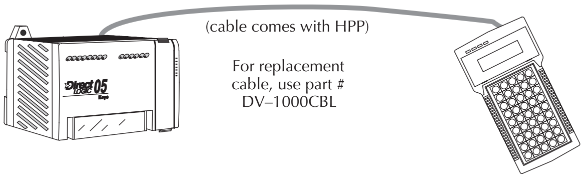
Communication Port 2	
Com 2	Connects to HPP, ***Direct***SOFT, operator interfaces, etc. 6-pin, RS232C Communication speed (baud) 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 Parity - odd (default), even, none Station address 1 (default) 8 data bits 1 start, 1 stop bit Asynchronous, Half-duplex, DTE Protocol: (Auto-Select) K sequence (Slave only) ***Direct***NET (Master/Slave) Modbus RTU (Master/Slave) Non-sequence/Print

Connecting the Programming Devices

If you're using a Personal Computer with the *Direct*SOFT programming package, you can connect the computer to either of the DL05's programming ports. For an engineering office environment (typical during program development), this is the preferred method of programming.



The Handheld programmer is connected to the CPU with a handheld programmer cable. This device can be used for maintaining existing installations or making small program changes whenever a PC is not available. The handheld programmer is shipped with a cable, which is approximately 6.5 feet (200cm) long.

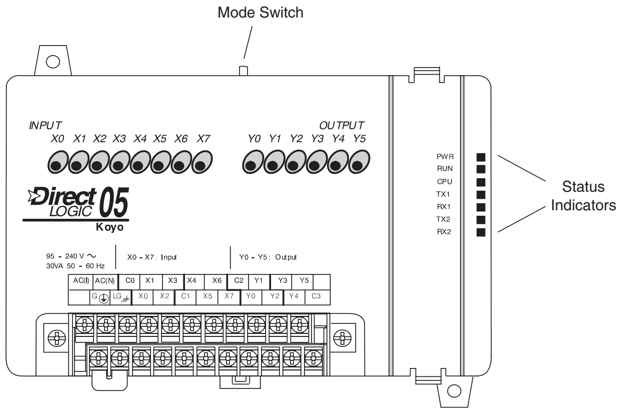


CPU Setup Information

Even if you have years of experience using PLCs, there are a few things you need to do before you can start entering programs. This section includes some basic things, such as changing the CPU mode, but it also includes some things that you may never have to use. Here's a brief list of the items that are discussed. Selecting and Changing the CPU Modes

- Using Auxiliary Functions
- Clearing the program (and other memory areas)
- How to initialize system memory
- Setting retentive memory ranges

The following paragraphs provide the setup information necessary to get the CPU ready for programming. They include setup instructions for either type of programming device you are using. The D2–HPP Handheld Programmer Manual provides the Handheld keystrokes required to perform all of these operations. The *Direct*SOFT Programming Software User Manual provides a description of the menus and keystrokes required to perform the setup procedures.



Status Indicators

The status indicator LEDs on the CPU front panels have specific functions which can help in programming and troubleshooting.

Indicator	Status	Meaning
PWR	ON	Power good
	OFF	Power failure
RUN	ON	CPU is in Run Mode
	OFF	CPU is in Stop or program Mode
	Blinking	CPU is in upgrade Mode
CPU	ON	CPU self diagnostics error
	OFF	CPU self diagnostics good
TX1	ON	Data is being transmitted by the CPU - Port 1
	OFF	No data is being transmitted by the CPU - Port 1
RX1	ON	Data is being received by the CPU - Port 1
	OFF	No data is being received by the CPU - Port 1
TX2	ON	Data is being transmitted by the CPU - Port 2
	OFF	No data is being transmitted by the CPU - Port 2
RX2	ON	Data is being received by the CPU - Port 2
	OFF	No data is being received by the CPU - Port 2

Mode Switch Functions

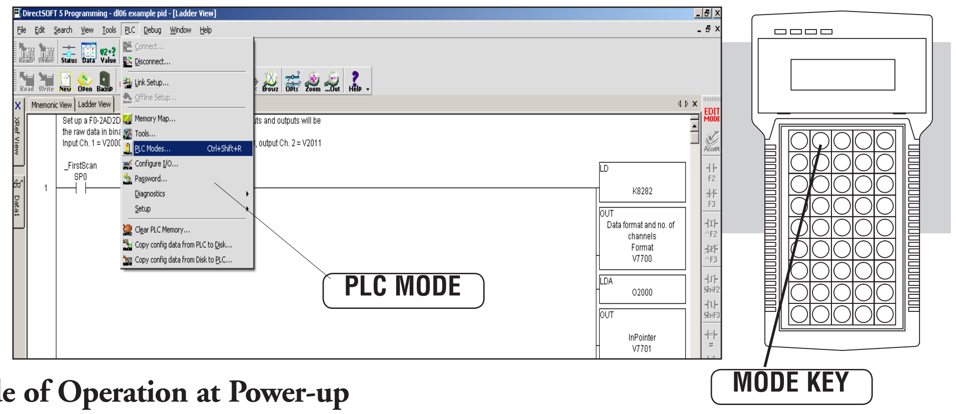
The mode switch on the DL05 PLC provides positions for enabling and disabling program changes in the CPU. Unless the mode switch is in the TERM position, RUN and STOP mode changes will not be allowed by any interface device, (handheld programmer, *Direct*SOFT programing package or operator interface). Programs may be viewed or monitored but no changes may be made. If the switch is in the TERM position and no program password is in effect, all operating modes as well as program access will be allowed through the connected programming or monitoring device.

NOTE: *If the DL05 is switched to the RUN Mode without a program in the PLC, the PLC will produce a FATAL ERROR which can be cleared by cycling power to the PLC.*

Changing Modes in the DL05 PLC

Mode Switch Position	CPU Action
RUN (Run Program)	CPU is forced into the RUN mode if no errors are encountered. No changes are allowed by the attached programming/monitoring device.
TERM (Terminal) RUN,	PROGRAM and the TEST modes are available. Mode and program changes are allowed by the programming/ monitoring device.
STOP	CPU is forced into the STOP mode. No changes are allowed by the programming/monitoring device.

There are two ways to change the CPU mode. You can use the CPU mode switch to select the operating mode, or you can place the mode switch in the TERM position and use a programming device to change operating modes. With the switch in this position, the CPU can be changed between Run and Program modes. You can use either ***Direct***SOFT or the Handheld Programmer to change the CPU mode of operation. With ***Direct***SOFT use the PLC menu option **PLC > Mode** or use the **Mode** button located on the Online toolbar. With the Handheld Programmer, you use the MODE key.


DirectSOFT 5 Programming - dl06 example pid - [Ladder View]
Connect...
Disconnect...
Link Setup...
Offline Setup...
Memory Map...
Tools...
PLC Modes... Ctrl+Shift+R
Configure I/O...
Password...
Diagnostics
Setup
Clear PLC Memory...
Copy config data from PLC to Disk...
Copy config data from Disk to PLC...
PLC MODE
MODE KEY

Mode of Operation at Power-up

The DL05 CPU will normally power-up in the mode that it was in just prior to the power interruption. For example, if the CPU was in Program Mode when the power was disconnected, the CPU will power-up in Program Mode (see warning note below).

WARNING: Once the super capacitor has discharged, the system may not power-up in the mode it was in when this occurred. There is no way to determine which mode will be entered as the startup mode. However, the PLC can power-up in either Run or Program Mode if the mode switch is in the TERM position. Failure to adhere to this warning greatly increases the risk of unexpected equipment startup.

The mode which the CPU will power-up in is also determined by the state of B7633.13. If the bit is set and the Mode Switch is in the TERM position, then the CPU will power-up in the state it was in at power-down.

Auxiliary Functions

Many CPU setup tasks involve the use of Auxiliary (AUX) Functions. The AUX Functions perform many different operations, ranging from clearing ladder memory, displaying the scan time, copying programs to EEPROM in the handheld programmer, etc. They are divided into categories that affect different system parameters. Appendix A provides a description of the AUX functions.

You can access the AUX Functions from ***Direct***SOFT or from the D2–HPP Handheld Programmer. The manuals for those products provide step-by-step procedures for accessing the AUX Functions. Some of these AUX Functions are designed specifically for the Handheld Programmer setup, so they will not be needed (or available) with the ***Direct***SOFT package. The following table shows a list of the Auxiliary functions for the Handheld Programmer.

AUX 2* — RLL Operations		5B	HSIO Configuration
21	Check Program	5D	Scan Control Setup
22	Change Reference	**AUX 6* — Handheld Programmer Configuration**	
23	Clear Ladder Range	61	Show Revision Numbers
24	Clear All Ladders	62	Beeper On / Off
AUX 3* — V-Memory Operations		65	Run Self Diagnostics
31	Clear V-Memory	**AUX 7* — EEPROM Operations**	
AUX 4* — I/O Configuration		71	Copy CPU memory to HPP EEPROM
41	Show I/O Configuration	72	Write HPP EEPROM to CPU
AUX 5* — CPU Configuration		73	Compare CPU to HPP EEPROM
51	Modify Program Name	74	Blank Check (HPP EEPROM)
53	Display Scan Time	75	Erase HPP EEPROM
54	Initialize Scratchpad	76	Show EEPROM Type (CPU and HPP)
55	Set Watchdog Timer	**AUX 8* — Password Operations**	
56	Set Communication Port 2	81	Modify Password
57	Set Retentive Ranges	82	Unlock CPU
58	Test Operations	83	Lock CPU
59	Override Setup		

Clearing an Existing Program

Before you enter a new program, be sure to always clear ladder memory. You can use AUX Function 24 to clear the complete program.

You can also use other AUX functions to clear other memory areas.

- AUX 23 — Clear Ladder Range
- AUX 24 — Clear all Ladders
- AUX 31 — Clear V-Memory

Initializing System Memory

The DL05 Micro PLC maintain system parameters in a memory area often referred to as the "scratchpad". In some cases, you may make changes to the system setup that will be stored in system memory. For example, if you specify a range of Control Relays (CRs) as retentive, these changes are stored in system memory. AUX 54 resets the system memory to the default values.

WARNING: You may never have to use this feature unless you want to clear any setup information that is stored in system memory. Usually, you'll only need to initialize the system memory if you are changing programs and the old program required a special system setup. You can usually load in new programs without ever initializing system memory. Remember, this AUX function will reset all system memory. If you have set special parameters such as retentive ranges, etc. they will be erased when AUX 54 is used. Make sure you that you have considered all ramifications of this operation before you select it.

Setting Retentive Memory Ranges

The DL05 PLCs provide certain ranges of retentive memory by default. The default ranges are suitable for many applications, but you can change them if your application requires additional retentive ranges or no retentive ranges at all. Appendix F has more information pertaining to the different types of memory. The default settings are:

Memory Area	DL05	
	Default Range	Available Range
Control Relays	C400 – C777	C0 – C777
V-memory	V1400 – V7777	V0 – V7777
Timers	None by default	T0 – T177
Counters	CT0 – CT177	CT0 – CT177
Stages	None by default	S0 – S377

You can use AUX 57 to set the retentive ranges. Appendix A contains detailed information about auxiliary functions. You can also set the retentive ranges by using Setup in *Direct*SOFT, PLC > Setup > Retentive Ranges.

WARNING: The DL05 PLCs do not have battery back-up (unless the memory cartridge, D0–01MC, is installed) The super capacitor will retain the values in the event of a power loss, but only for a short period of time, depending on conditions.

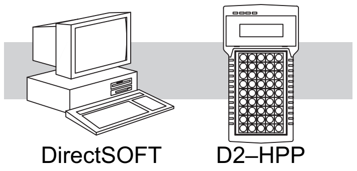
Using a Password

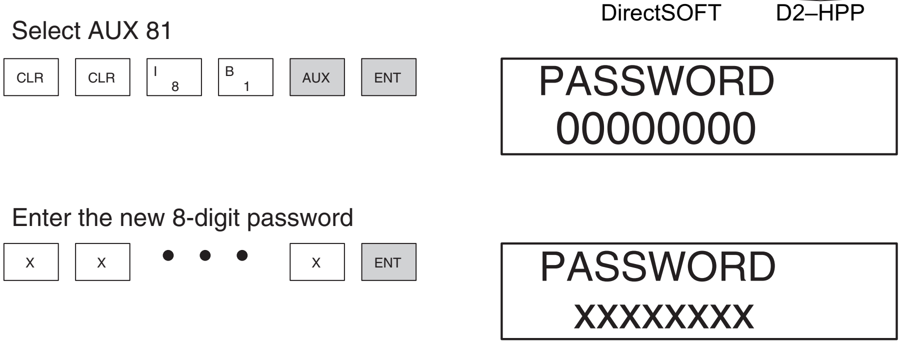
The DL05 PLCs allow you to use a password to help minimize the risk of unauthorized program and/or data changes. Once you enter a password you can “lock” the PLC against access. Once the CPU is locked you must enter the password before you can use a programming device to change any system parameters.

You can select an 8-digit numeric password. The Micro PLCs are shipped from the factory with a password of 00000000. All zeros removes the password protection. If a password has been entered into the CPU you cannot just enter all zeros to remove it. Once you enter the correct password, you can change the password to all zeros to remove the password protection.

WARNING: Make sure you remember your password. If you forget your password you will not be able to access the CPU. The Micro PLC must be returned to the factory to have the password (along with the ladder project) cleared from memory. It is the policy of AutomationDirect to require the memory of the PLC to be cleared along with the password.

You can use the D2–HPP Handheld Programmer or *Direct*SOFT to enter a password. The following diagram shows how you can enter a password with the Handheld Programmer.





Press CLR to clear the display

There are three ways to lock the CPU once the password has been entered.

1. If the CPU power is disconnected, the CPU will be automatically locked against access.
2. If you enter the password with *Direct*SOFT, the CPU will be automatically locked against access when you exit *Direct*SOFT.
3. Use AUX 83 to lock the CPU.

When you use *Direct*SOFT, you will be prompted for a password if the CPU has been locked. If you use the Handheld Programmer, you have to use AUX 82 to unlock the CPU. Once you enter AUX 82, you will be prompted to enter the password.

CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL05 CPUs control all aspects of system operation. There are four main areas to understand before you create your application program:

- CPU Operating System — the CPU manages all aspects of system control. A quick overview of all the steps is provided in the next section.
- CPU Operating Modes — The two primary modes of operation are Program Mode and Run Mode.
- CPU Timing — The two important areas we discuss are the I/O response time and the CPU scan time.
- CPU Memory Map — DL05 CPUs offer a wide variety of resources, such as timers, counters, inputs, etc. The memory map section shows the organization and availability of these data types.

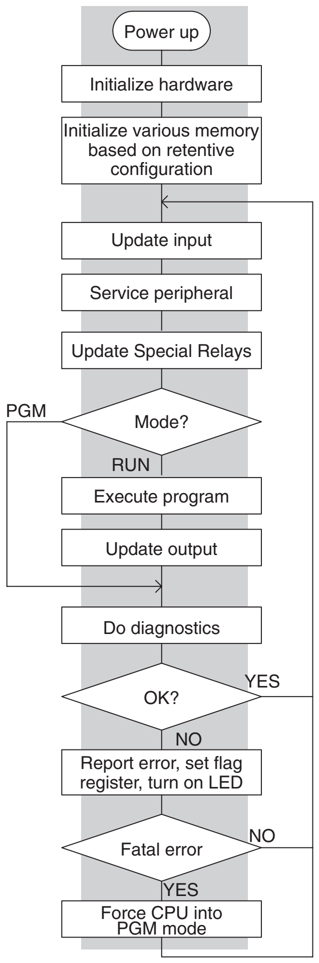
CPU Operating System

At powerup, the CPU initializes the internal electronic hardware. Memory initialization starts with examining the retentive memory settings. In general, the content of retentive memory is preserved, and non-retentive memory is initialized to zero (unless otherwise specified).

After the one-time powerup tasks, the CPU begins the cyclical scan activity. The flowchart to the right shows how the tasks differ, based on the CPU mode and the existence of any errors. The “scan time” is defined as the average time around the task loop. Note that the CPU is always reading the inputs, even during program mode. This allows programming tools to monitor input status at any time.

The outputs are only updated in Run mode. In program mode, they are in the off state.

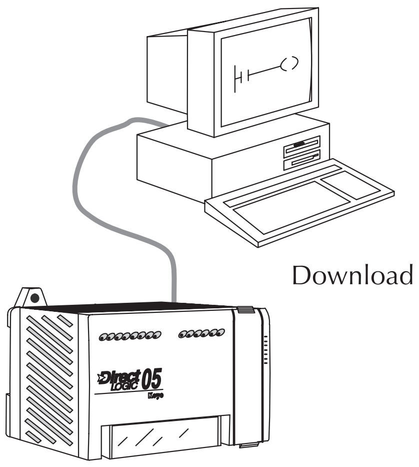
Error detection has two levels. Non-fatal errors are reported, but the CPU remains in its current mode. If a fatal error occurs, the CPU is forced into program mode and the outputs go off.



Program Mode

In Program Mode, the CPU does not execute the application program or update the output points. The primary use for Program Mode is to enter or change an application program. You also use program mode to set up the CPU parameters, such as HSIO features, retentive memory areas, etc.

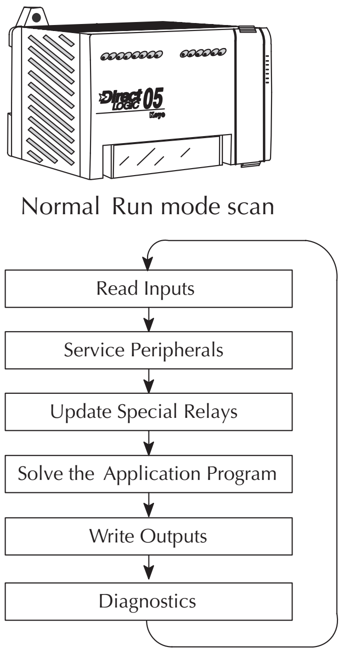
You can use a programming device, such as a PC with *Direct*SOFT Programming Software or the D2–HPP Handheld programmer to place the CPU in Program Mode.



Run Mode

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 Variable memory locations



Run Mode operation can be divided into several key areas. For the vast majority of applications, some of these execution segments are more important than others. For example, you need to understand how the CPU updates the I/O points, handles forcing operations, and solves the application program. The remaining segments are not that important for most applications.

You can use *Direct*SOFT or the D2–HPP Handheld Programmer to place the CPU in Run Mode.

You can also edit the program during Run Mode. The Run Mode Edits are not “bumpless” to the outputs. Instead, the CPU maintains the outputs in their last state while it accepts the new program information. If an error is found in the new program, then the CPU will turn all the outputs off and enter the Program Mode. This feature is discussed in more detail in Chapter 9.

WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Changes during Run Mode become effective immediately. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

Read Inputs

The CPU reads the status of all inputs, then stores it in the image register. Input image register locations are designated with an X followed by a memory location. Image register data is used by the CPU when it solves the application program.

Of course, an input may change after the CPU has just read the inputs. Generally, the CPU scan time is measured in milliseconds. If you have an application that cannot wait until the next I/O update, you can use Immediate Instructions. These do not use the status of the input image register to solve the application program. The Immediate instructions immediately read the input status directly from the I/O modules. However, this lengthens the program scan since the CPU has to read the I/O point status again. A complete list of the Immediate instructions is included in Chapter 5.

Service Peripherals and Force I/O

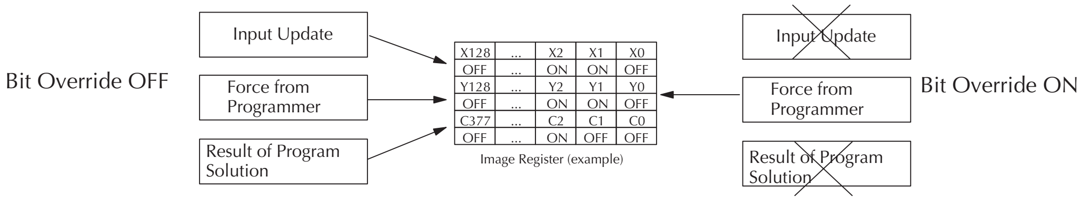
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified. There are two basic types of forcing available with the DL05 CPUs.

- Forcing from a peripheral – not a permanent force, good only for one scan
- Bit Override – holds the I/O point (or other bit) in the current state. Valid bits are X, Y, C, T, CT, and S. (These memory types are discussed in more detail later in this chapter).

Regular Forcing — This type of forcing can temporarily change the status of a discrete bit. 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 next scan. This is primarily useful during testing situations when you need to force a bit on to trigger another event.

Bit Override — Bit override can be enabled on a point-by-point basis by using AUX 59 from the Handheld Programmer or, by using the Data View option within ***Direct***SOFT. Bit override basically disables any changes to the discrete point by the CPU. For example, if you enable bit override for X1, and X1 is off at the time, then the CPU will not change the state of X1. This means that even if X1 comes on, the CPU will not acknowledge the change. So, if you used X1 in the program, it would always be evaluated as "off" in this case. Of course, if X1 was on when the bit override was enabled, then X1 would always be evaluated as "on".

There is an advantage available when you use the bit override feature. The regular forcing is not disabled because the bit override is enabled. For example, if you enabled the Bit Override for Y0 and it was off at the time, then the CPU would not change the state of Y0. However, you can still use a programming device to change the status. Now, if you use the programming device to force Y0 on, it will remain on and the CPU will not change the state of Y0. If you then force Y0 off, the CPU will maintain Y0 as off. The CPU will never update the point with the results from the application program or from the I/O update until the bit override is removed. The following diagram shows a brief overview of the bit override feature. Notice the CPU does not update the Image Register when bit override is enabled.



WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

Update Special Relays and Special Registers

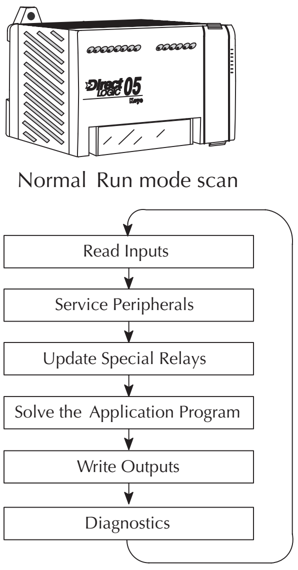
There are dedicated V-memory locations that contain Special Relays and other dedicated register information. This portion of the execution cycle makes sure these locations get updated on every scan. Also, there are several different Special Relays, such as diagnostic relays, etc., that are also updated during this segment.

Solve Application Program

The CPU evaluates each instruction in the application program during this segment of the scan 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. Output image register locations are designated with a Y followed by a memory location. The actual outputs are updated during the write outputs segment of the scan cycle. There are immediate output instructions available that will update the output points immediately instead of waiting until the write output segment. A complete list of the Immediate instructions is provided in Chapter 5.

The internal control relays (C), the stages (S), and the variable memory (V) are also updated in this segment.

You may recall that you can force various types of points in the system. (This was discussed earlier in this chapter.) If any I/O points or memory data have been forced, the output image register also contains this information.



Write Outputs

Once the application program has solved the instruction logic and constructed the output image register, the CPU writes the contents of the output image register to the corresponding output points. Remember, the CPU also made sure that any forcing operation changes were stored in the output image register, so the forced points get updated with the status specified earlier.

Diagnostics

During this part of the scan, the CPU performs all system diagnostics and other tasks such as calculating the scan time and resetting the watchdog timer. There are many different error conditions that are automatically detected and reported by the DL05 PLCs. Appendix B contains a listing of the various error codes.

Probably one of the more important things that occurs during this segment is the scan time calculation and watchdog timer control. The DL05 CPU has a "watchdog" timer that stores the maximum time allowed for the CPU to complete the solve application segment of the scan cycle. If this time is exceeded the CPU will enter the Program Mode and turn off all outputs. The default value set from the factory is 200ms. An error is automatically reported. For example, the Handheld Programmer would display the following message "E003 S/W TIMEOUT" when the scan overrun occurs.

You can use AUX 53 to view the minimum, maximum, and current scan time. Use AUX 55 to increase or decrease the watchdog timer value.

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 that 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.

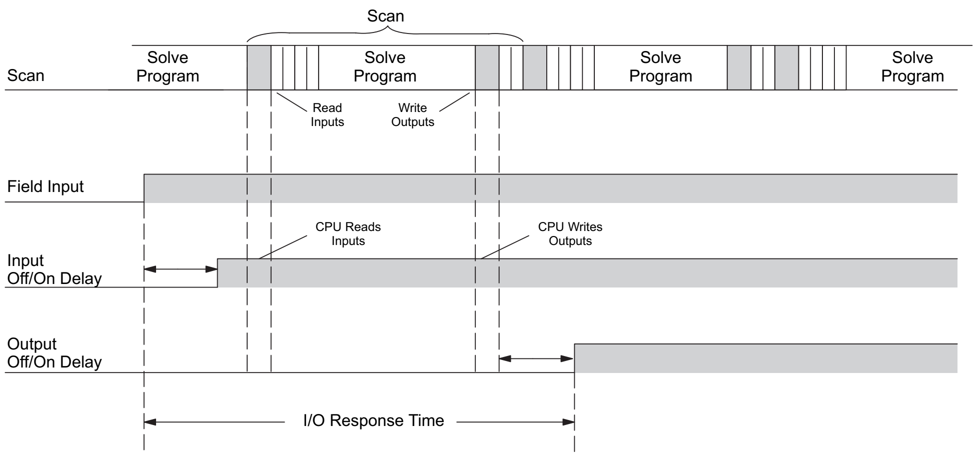
There are four things that can affect the I/O response time.

- The point in the scan cycle when the field input changes states
- Input Off to On delay time
- CPU scan time
- Output 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 input changes just before the Read Inputs portion of the execution cycle. In this case the input status is read, the application program is solved, and the output point gets updated. The following diagram shows an example of the timing for this situation.

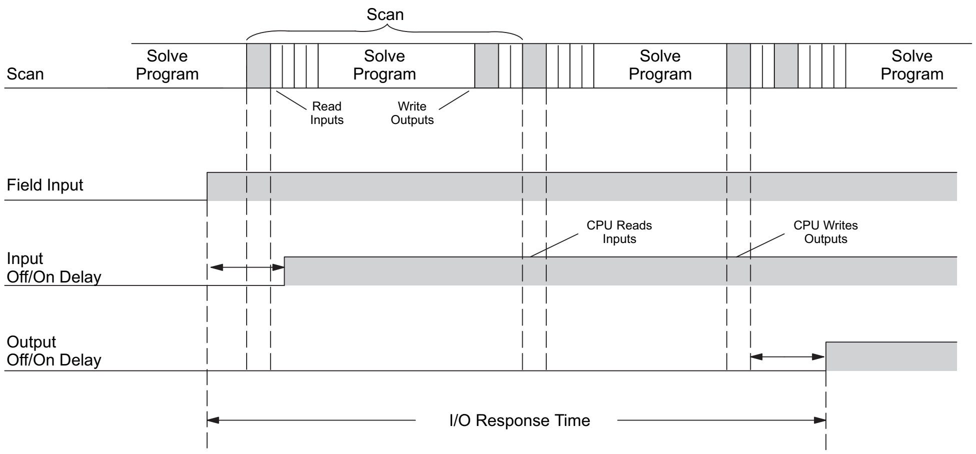


In this case, you can calculate the response time by simply adding the following items:

Input Delay + Scan Time + Output Delay = Response Time

Normal Maximum I/O Response

The I/O response time is longest when the input changes just after the Read Inputs portion of the execution cycle. In this case the new input status is not 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:

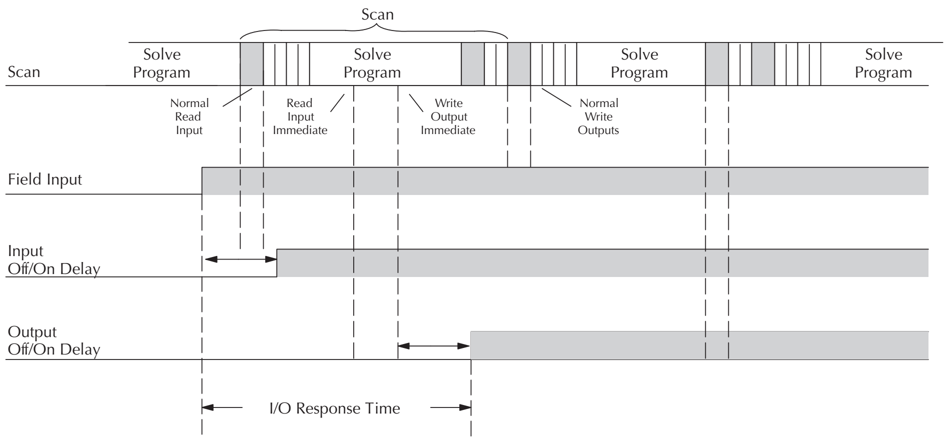
Input Delay +(2 x Scan Time) + Output Delay = Response Time

Improving Response Time

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

- You can choose instructions with faster execution times
- You can use immediate I/O instructions (which update the I/O points during the program execution)
- You can use the HSIO Mode 50 Pulse Catch features designed to operate in high-speed environments. See Appendix E for details on using this feature.
- Change Mode 60 filter to 0ms for X0, X1, X2 and X3.

Of these four things the Immediate I/O instructions are probably the most important and most useful. The following example shows how an immediate input instruction and immediate output instruction would affect the response time.



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

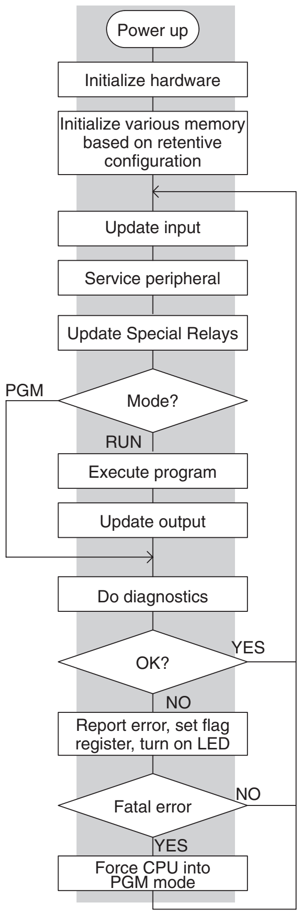
Input Delay + Instruction Execution Time + Output Delay = Response Time

The instruction execution time would be calculated by adding the time for the immediate input instruction, the immediate output instruction, and any other instructions in between the two.

***NOTE**: Even though the immediate instruction reads the most current status from I/O, it only uses the results to solve that one instruction. It does not use the new status to update the image register. Therefore, any regular instructions that follow will still use the image register values. Any immediate instructions that follow will access the I/O again to update the status.*

CPU Scan Time Considerations

The scan time covers all the cyclical tasks that are performed by the operating system. You can use *Direct*SOFT or the Handheld Programmer to display the minimum, maximum, and current scan times that have occurred since the previous Program Mode to Run Mode transition. 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 scan cycle. Each of these segments requires a certain amount of time to complete. Of all the segments, the following are the most important.



- Input Update
- Peripheral Service
- Program Execution
- Output Update
- Timed Interrupt Execution

The only one you really have the most control over is the amount of time it takes to execute the application program. This is because different instructions take different amounts of time to execute. So, if you think you need a faster scan, then you can try to choose faster instructions.

Your choice of I/O type and peripheral devices can also affect the scan time. However, these things are usually dictated by the application.

The following paragraphs provide some general information on how much time some of the segments can require.

Reading Inputs

The time required during each scan to read the input status is 40µs. Don't confuse this with the I/O response time that was discussed earlier.

Writing Outputs

The time required to write the output status is 629µs. Don't confuse this with the I/O response time that was discussed earlier.

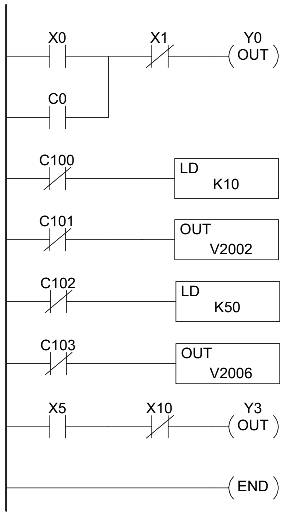
Application Program Execution

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, and the amount of execution overhead.

Just add the execution times for all the instructions in your program to determine to total execution time. Appendix C provides a complete list of the instruction execution times for the DL05 Micro PLC. For example, the execution time for running the program shown below is calculated as follows:

Instruction	Time
STR X0	2.0 μs
OR C0	1.6 μs
ANDN X1	1.6 μs
OUT Y0	6.8 μs
STRN C100	2.3 μs
LD K10	42.7 μs
STRN C101	2.3 μs
OUT V2002	16.6 μs
STRN C102	2.3 μs
LD K50	42.7 μs
STRN C103	2.3 μs
OUT V2006	16.6 μs
STR X5	2.0 μs
ANDN X10	1.6 μs
OUT Y3	6.8 μs
END	24.0 μs
SUBTOTAL	174.2 μs

Overhead	DL05
Minimum	0.66 μs
Maximum	2.5 μs

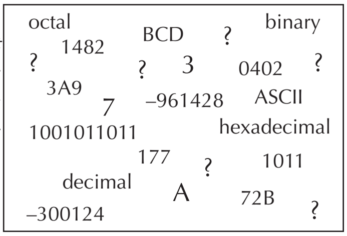


TOTAL TIME = (Program execution time + Overhead) x 1.1

The program above takes only 174.2 μs to execute during each scan. The DL05 spends 0.1 ms, on internal timed interrupt management, for every 1.0 ms of instruction time. The total scan time is calculated by adding the program execution time to the overhead (shown above) and multiplying the result (ms) by 1.1. “Overhead” includes all other housekeeping and diagnostic tasks. The scan time will vary slightly from one scan to the next, because of fluctuation in overhead tasks.

Program Control Instructions — the DL05 PLCs have an interrupt routine feature that changes the way a program executes. Since this instruction interrupts normal program flow, it will have an effect on the program execution time. For example, a timed interrupt routine with a 10.0 ms period interrupts the main program execution (before the END statement) every 10.0 ms, so the CPU can execute the interrupt routine. Chapter 5 provides detailed information on interrupts.

PLC Numbering Systems

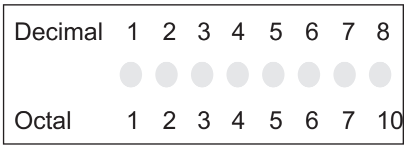


If you are a new PLC user or are using *AutomationDirect* PLCs for the first time, please take a moment to study how our PLCs use numbers. You'll find that each PLC manufacturer has their own conventions on the use of numbers in their PLCs. We want to take just a moment to familiarize you with how numbers are used in *AutomationDirect* PLCs. The information you learn here applies to all of our PLCs.

As any good computer does, PLCs store and manipulate numbers in binary form: just ones and zeros. So why do we have to deal with numbers in so many different forms? Numbers have meaning, and some representations are more convenient than others for particular purposes. Sometimes we use numbers to represent a size or amount of something. Other numbers refer to locations or addresses, or to time. In science we attach engineering units to numbers to give a particular meaning (see Appendix I for numbering system details).

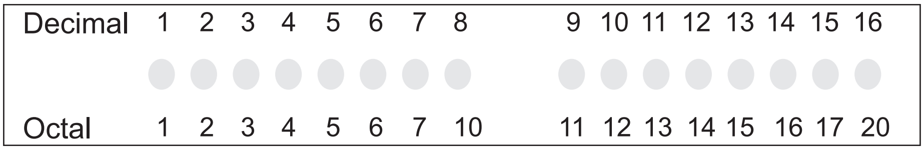
PLC Resources

PLCs offer a fixed amount of resources, depending on the model and configuration. We use the word "resources" to include variable memory (V-memory), I/O points, timers, counters, etc. Most modular PLCs allow you to add I/O points in groups of eight. In fact, all the resources of our PLCs are counted in octal. It's easier for computers to count in groups of eight than ten, because eight is an even power of 2.

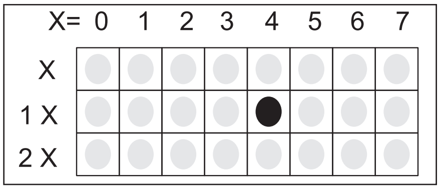


Octal means simply counting in groups of eight things at a time. In the figure to the right, there are eight circles. The quantity in decimal is "8", but in octal it is "10" (8 and 9 are not valid in octal). In octal, "10" means 1 group of 8 plus 0 (no individuals).

In the figure below, we have two groups of eight circles. Counting in octal we have "20" items, meaning 2 groups of eight, plus 0 individuals Don't say "twenty", say "two–zero octal". This makes a clear distinction between number systems.


Decimal 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Octal 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20

After counting PLC resources, it is time to access PLC resources (there is a difference). The CPU instruction set accesses resources of the PLC using octal addresses. Octal addresses are the same as octal quantities, except they start counting at zero. The number zero is significant to a computer, so we don't skip it.



Our circles are in an array of square containers to the right. To access a resource, our PLC instruction will address its location using the octal references shown. If these were counters, "CT14" would access the black circle location.

V–memory

Variable memory (called V-memory) stores data for the ladder program and for configuration settings. V-memory locations and V-memory addresses are the same thing, and are numbered in octal. For example, V2073 is a valid location, while V1983 is not valid (“9” and “8” are not valid octal digits).

V-memory address (octal)	V-memory data (binary) MSB															LSB
V2017	0	1	0	0	1	1	1	0	0	0	1	0	1	0	0	1

Each V-memory location is one data word wide, meaning 16 bits. For configuration registers, our manuals will show each bit of a V-memory word. The least significant bit (LSB) will be on the right, and the most significant bit (MSB) on the left. We use the word “significant”, referring to the relative binary weighting of the bits.

V-memory data is 16-bit binary, but we rarely program the data registers one bit at a time. We use instructions or viewing tools that let us work with decimal, octal, and hexadecimal numbers. All these are converted and stored as binary for us.

A frequently-asked question is “How do I tell if a number is octal, BCD, or hex”? The answer is that we usually cannot tell just by looking at the data... but it does not really matter. What matters is: the source or mechanism which writes data into a V-memory location and the thing which later reads it must both use the same data type (i.e., octal, hex, binary, or whatever). The V-memory location is just a storage box... that’s all. It does not convert or move the data on its own.

Binary-Coded Decimal Numbers

Since humans naturally count in decimal (10 fingers, 10 toes), we prefer to enter and view PLC data in decimal as well. However, computers are more efficient in using pure binary numbers. A compromise solution between the two is Binary-Coded Decimal (BCD) representation. A BCD digit ranges from 0 to 9, and is stored as four binary bits (a nibble). This permits each V-memory location to store four BCD digits, with a range of decimal numbers from 0000 to 9999.

BCD number	4	9	3	6
V-memory storage	0 1 0 0	1 0 0 1	0 0 1 1	0 1 1 0

In a pure binary sense, a 16-bit word can represent numbers from 0 to 65535. In storing BCD numbers, the range is reduced to only 0 to 9999. Many math instructions use Binary-Coded Decimal (BCD) data, and ***Direct***SOFT and the handheld programmer allow us to enter and view data in BCD.

Hexadecimal Numbers

Hexadecimal numbers are similar to BCD numbers, except they utilize all possible binary values in each 4-bit digit. They are base-16 numbers so we need 16 different digits. To extend our decimal digits 0 through 9, we use A through F as shown.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

A 4-digit hexadecimal number can represent all 65536 values in a V-memory word. The range is from 0000 to FFFF (hex). PLC’s often need this full range for sensor data, etc. Hexadecimal is just a convenient way for humans to view full binary data.

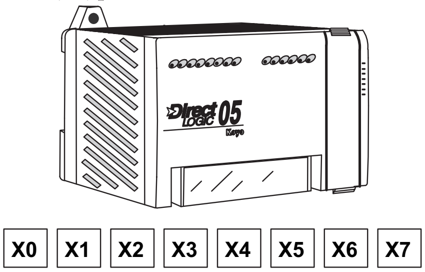
Hexadecimal number	A	7	F	4
V-memory storage	1 0 1 0	0 1 1 1	1 1 1 1	0 1 0 0

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 DL05 Micro PLCs. A memory map overview for the CPU follows the memory descriptions.

Octal Numbering System

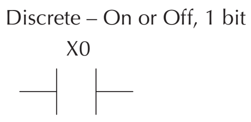
All memory locations and resources are numbered in Octal (base 8). For example, the diagram shows how the octal numbering system works for the discrete input points. Notice the octal system does not contain any numbers with the digits 8 or 9.

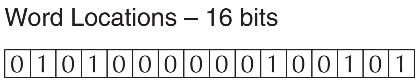


Discrete and Word Locations

As you examine the different memory types, you'll notice two types of memory in the DL05, discrete and word memory. Discrete memory is one bit that can be either a 1 or a 0. Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc.

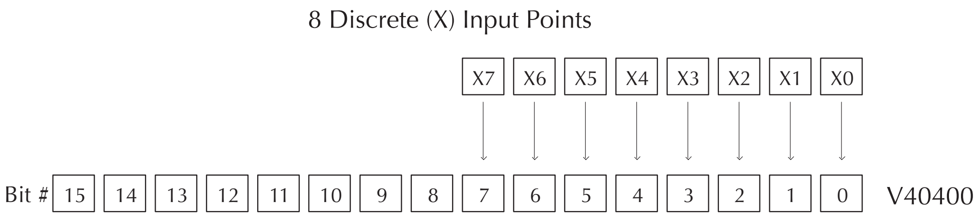
Some information is automatically stored in V-memory. For example, the timer current values are stored in V-memory.





V-memory Locations for Discrete Memory Areas

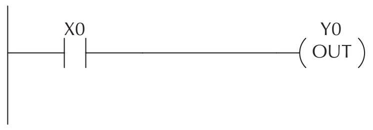
The discrete memory area is for inputs, outputs, control relays, special relays, stages, timer status bits and counter status bits. However, you can also access the bit data types as a V-memory word. Each V-memory location contains 16 consecutive discrete locations. For example, the following diagram shows how the X input points are mapped into V-memory locations.



These discrete memory areas and their corresponding V-memory ranges are listed in the memory area table for DL05 Micro PLCs on the following pages.

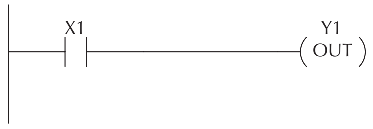
Input Points (X Data Type)

The discrete input points are noted by an X data type. There are 8 discrete input points and 256 discrete input addresses available with DL05 CPUs. In this example, the output point Y0 will be turned on when input X0 energizes.



Output Points (Y Data Type)

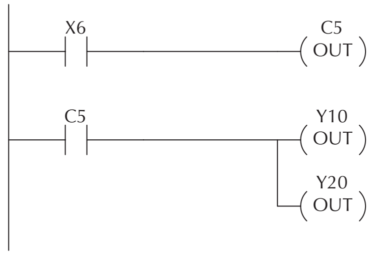
The discrete output points are noted by a Y data type. There are 6 discrete outputs and 256 discrete output addresses available with DL05 CPUs. In this example, output point Y1 will be turned on when input X1 energizes.



Control Relays (C Data Type)

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 16 consecutive discrete locations.

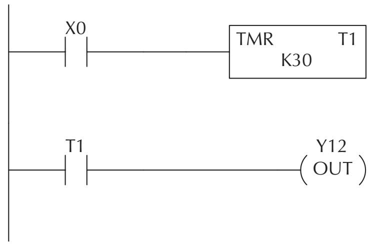
In this example, memory location C5 will energize when input X6 turns on. The second rung shows a simple example of how to use a control relay as an input.



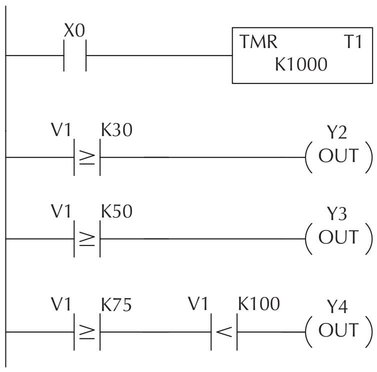
Timers and Timer Status Bits (T Data Type)

Timer status bits reflect the relationship between the current value and the preset value of a specified timer. The timer status bit will be on when the current value is equal or greater than the preset value of a corresponding timer.

When input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K30) timer status contact T1 turns on. When T1 turns on, output Y12 turns on. Turning off X0 resets the timer.



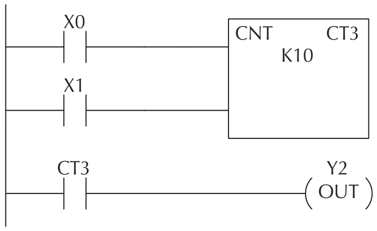
Timer Current Values (V Data Type)



As mentioned earlier, some information is automatically stored in V-memory. This is true for the current values associated with timers. For example, V0 holds the current value for Timer 0, V1 holds the current value for Timer 1, etc.

The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor several time intervals from a single timer.

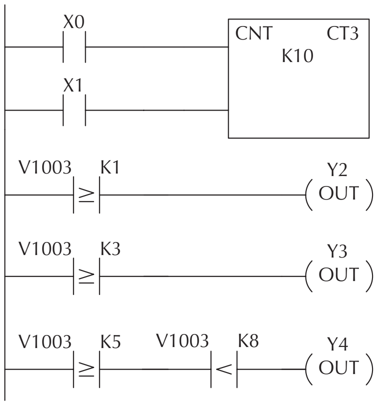
Counters and Counter Status Bits (CT Data type)



Counter status bits that reflect the relationship between the current value and the preset value of a specified counter. The counter status bit will be on when the current value is equal to or greater than the preset value of a corresponding counter.

Each time contact X0 transitions from off to on, the counter increments by one. (If X1 comes on, the counter is reset to zero.) When the counter reaches the preset of 10 counts (K of 10) counter status contact CT3 turns on. When CT3 turns on, output Y2 turns on.

Counter Current Values (V Data Type)

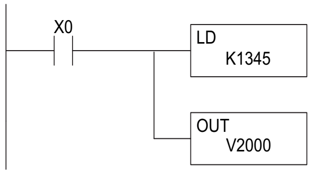


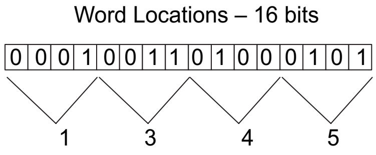
Just like the timers, the counter current values are also automatically stored in V-memory. For example, V1000 holds the current value for Counter CT0, V1001 holds the current value for Counter CT1, etc.

The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor the counter values.

Word Memory (V Data Type)

Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc. Some information is automatically stored in V-memory. For example, the timer current values are stored in V-memory. The example shows how a four-digit BCD constant is loaded into the accumulator and then stored in a V-memory location.

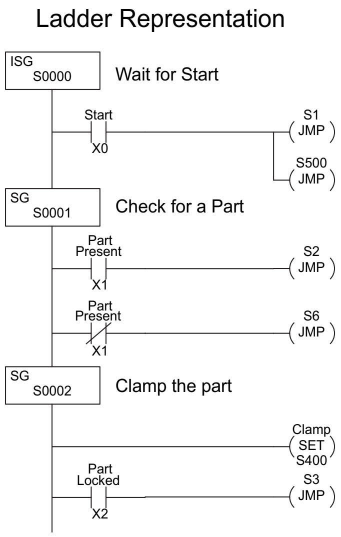




Stages (S Data type)

Stages are used in RLL^{PLUS} programs to create a structured program, similar to a flowchart. Each program Stage denotes a program segment. When the program segment, or Stage, is active, the logic within that segment is executed. If the Stage is off, or inactive, the logic is not executed and the CPU skips to the next active Stage. (See Chapter 7 for a more detailed description of RLL^{PLUS} programming.)

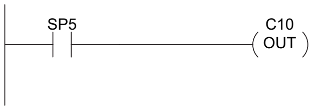
Each Stage also has a discrete status bit that can be used as an input to indicate whether the Stage is active or inactive. If the Stage is active, then the status bit is on. If the Stage is inactive, then the status bit is off. This status bit can also be turned on or off by other instructions, such as the SET or RESET instructions. This allows you to easily control stages throughout the program.



Special Relays (SP Data Type)

Special relays are discrete memory locations with pre-defined functionality. There are many different types of special relays. For example, some aid in program development, others provide system operating status information, etc. Appendix D provides a complete listing of the special relays.

In this example, control relay C10 will energize for 50ms and de-energize for 50ms because SP5 is a pre–defined relay that will be on for 50ms and off for 50 ms.



SP4: 1 second clock

SP5: 100ms clock

SP6: 50ms clock

DL05 System V-memory

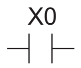
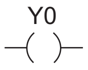
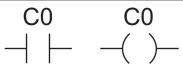
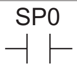
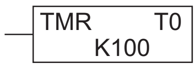
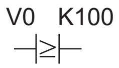
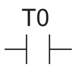
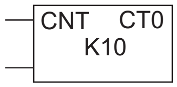
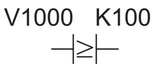
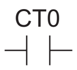
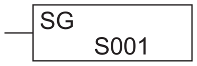
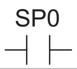
System Parameters and Default Data Locations (V Data Type)

The DL05 PLCs reserve several V-memory locations for storing system parameters or certain types of system data. These memory locations store things like the error codes, High-Speed I/O data, and other types of system setup information.

System V-memory		Description of Contents	Default Values/ Ranges
V2320–V2377		The default location for multiple preset values for the High-Speed Counter	N/A
V7620–V7627		**Locations for DV–1000 operator interface parameters**	
	V7620	Sets the V-memory location that contains the value	V0 – V2377
	V7621	Sets the V-memory location that contains the message	V0 – V2377
	V7622	Sets the total number (1–16) of V-memory locations to be displayed.	1–16
	V7623	Sets the V-memory location containing the numbers to be displayed.	V0–V2377
	V7624	Sets the V-memory location containing the character code to be displayed	V0–V2377
	V7625	Contains the function number that can be assigned to each key.	V-memory location for X, Y, or C points used
	V7626	Power-up operational mode.	0, 1, 2, 12, 3
	V7627	Change preset value.	0000 to 9999
V7630		Starting location for the multi–step presets for channel 1. The default value is 2320, which indicates the first value should be obtained from V2320. Since there are 24 presets available, the default range is V2320–V2377. You can change the starting point if necessary.	Default: V2320 Range: V0–V2320
V7631–V7632		Reserved	N/A
V7633		Sets the desired function code for the high speed counter, interrupt, pulse catch, pulse train, and input filter. Location can also be used to set the power-up in Run Mode option.	Default: 0060 Lower Byte Range: Range: 10 – Counter 20 – Quadrature 30 – Pulse Out 40 – Interrupt 50 – Pulse Catch 60 – Filtered discrete In. Upper Byte Range: Bits 8–12, 14, 15: Unused Bit 13: Power–up in RUN, if Mode Switch is in TERM position.
V7634 - X0		Setup Registers for High-Speed I/O functions	Default: 1006
V7635 - X1			
V7636 - X2			
V7637		Pulse/Direction	Default: 0000
V7640–V7646		Reserved	N/A
V7647		Timed Interrupt	Default: 0000 Range: 0003–03E7h (3–9999ms)
V7650–V7654		Reserved	N/A
V7655		Port 2: Setup for the protocol, time-out, and the response delay time.	Default: 00E0
V7656		Port 2: Setup for the station number, baud rate, STOP bit, and parity.	Default: 8501

System V-memory	Description of Contents	Default Values/ Ranges
V7657	Port 2: Setup completion code used to notify the completion of the parameter setup	Default: 0A00
V7660	Scan control setup: Keeps the scan control mode	Default: 0000
V7661	Setup timer over counter: Counts the times the actual scan time exceeds the user setup time	N/A
V7662–V7717	Reserved	
V7720–V7722	Locations for DV–1000 operator interface parameters	
V7720	Titled Timer preset value pointer	
V7721	Title Counter preset value pointer	
V7722	HiByte-Titled Timer preset block size, LoByte-Titled Counter preset block size	
V7723–V7750	Reserved	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed	
V7752–V7754	Reserved	
V7755	Error code — stores the fatal error code	
V7756	Error code — stores the major error code	
V7757	Error code — stores the minor error code	
V7760–V7762	Reserved	
V7763	Program address where syntax error exists	
V7764	Syntax error code	
V7765	Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition	
V7766	Contains the number of seconds on optional Memory Cartridge clock (00-59)	
V7767	Contains the number of minutes on optional Memory Cartridge clock (00-59)	
V7770	Contains the number of hours on optional Memory Cartridge clock (00-23)	
V7771	Contains the day of week on optional Memory Cartridge (Mon., Tues., Wed., etc.)	
V7772	Contains the numerical day of month on optional Memory Cartridge (01, 02, etc.)	
V7773	Contains the numerical month on optional Memory Cartridge (01 to 12)	
V7774	Contains the year on optional Memory Cartridge (00 to 99)	
V7775	Scan — stores the current scan time (milliseconds)	
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds)	
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds)	

DL05 Memory Map Table

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Decimal	Symbol
Input Points (See note)	X0 – X377	V40400 - V40417	256	
Output Points (See note)	Y0 – Y377	V40500 – V40517	256	
Control Relays	C0 – C777	V40600 - V40637	512	
Special Relays	SP0 – SP777	V41200 – V41237	512	
Timers	T0 – T177	V41100 – V41107	128	
Timer Current Values	None	V0 – V177	128	
Timer Status Bits	T0 – T177	V41100 – V41107	128	
Counters	CT0 – CT177	V41140 – V41147	128	
Counter Current Values	None	V1000 – V1177	128	
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	
Data Words (See Appendix F)	None	V1200 – V7377	3968	None specific, used with many instructions.
Data Words Non-volatile (See Appendix F)	None	V7400 – V7577	128	None specific, used with many instructions. May be non-volatile if MOV inst. is used. Data can be rewritten to EEPROM at least 100,000 times before it fails.
Stages	S0 – S377	V41000 – V41017	256	 
System parameters	None	V7600 – V7777	128	None specific, used for various purposes

NOTE: *The DL05 has 8 discrete inputs and 6 discrete outputs which are standard. The number of inputs and/or outputs can be increased by adding one of the available option modules. Refer to either the DL05/06 Option Modules User Manual (D0-OPTIONS-M), our catalog or our website.*

DL05 Aliases

An alias is an alternate way of referring to certain memory types, such as timer/counter current values, V-memory locations for I/O points, etc., which simplifies understanding the memory address. The use of the alias is optional, but some users may find the alias to be helpful when developing a program. The table below shows how the aliases can be used.

DL05 Aliases		
Address Start	**Alias Start**	**Example**
V0	TA0	V0 is the timer accumulator value for timer 0; therefore, its alias is TA0. TA1 is the alias for V1, etc.
V1000	CTA0	V1000 is the counter accumulator value for counter 0; therefore, its alias is CTA0. CTA1 is the alias for V1001, etc.
V40400	VX0	V40400 is the word memory reference for discrete bits X0 through X17; therefore, its alias is VX0. V40401 is the word memory reference for discrete bits X20 through X37; therefore, its alias is VX20.
V40500	VY0	V40500 is the word memory reference for discrete bits Y0 through Y17; therefore, its alias is VY0. V40501 is the word memory reference for discrete bits Y20 through Y37; therefore, its alias is VY20.
V40600	VC0	V40600 is the word memory reference for discrete bits C0 through C17; therefore, its alias is VC0. V40601 is the word memory reference for discrete bits C20 through C37; therefore, its alias is VC20.
V41000	VS0	V41000 is the word memory reference for discrete bits S0 through S17; therefore, its alias is VS0. V41001 is the word memory reference for discrete bits S20 through S37; therefore, its alias is VS20.
V41100	VT0	V41100 is the word memory reference for discrete bits T0 through T17; therefore, its alias is VT0. V41101 is the word memory reference for discrete bits T20 through T37; therefore, its alias is VT20.
V41140	VCT0	V41140 is the word memory reference for discrete bits CT0 through CT17; therefore, its alias is VCT0. V41141 is the word memory reference for discrete bits CT20 through CT37; therefore, its alias is VCT20.
V41200	VSP0	V41200 is the word memory reference for discrete bits SP0 through SP17; therefore, its alias is VSP0. V41201 is the word memory reference for discrete bits SP20 through SP37; therefore, its alias is VSP20.

X Input Bit Map

This table provides a listing of individual Input points associated with each V-memory address bit for the DL05's eight physical inputs. Actual available references are X0 to X377 (V40400 – V40417).

MSB	DL05 Input (X) Points															Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	–	007	006	005	004	003	002	001	000	V40400

This table provides the listing for the individual option slot Input points available.

MSB	DL05 Option Slot Input (X) Points															Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40404

Y Output Bit Map

This table provides a listing of individual output points associated with each V-memory address bit for the DL05's six physical outputs. Actual available references are Y0 to Y377 (V40500 – V40517).

MSB	DL05 Output (Y) Points														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	–	–	–	005	004	003	002	001	000	V40500

This table provides the listing for the individual option slot Output points available.

MSB	DL05 Option Slot Output (Y) Points														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40504

Control Relay Bit Map

This table provides a listing of the individual control relays associated with each V-memory address bit

MSB	DL05 Control Relays (C)														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V40600
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V40601
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V40602
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V40603
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40604
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V40605
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V40606
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V40607
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V40610
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V40611
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V40612
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V40613
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V40614
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V40615
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V40616
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V40617
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V40620
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V40621
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V40622
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V40623
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V40624
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V40625
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V40626
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V40627
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V40630
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V40631
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V40632
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V40633
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V40634
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V40635
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V40636
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V40637

Stage Control/Status Bit Map

This table provides a listing of individual™ Stage control bits associated with each V-memory address bit.

MSB	DL05 Stage (S) Control Bits														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	**V41000**
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	**V41001**
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	**V41002**
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	**V41003**
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	**V41004**
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	**V41005**
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	**V41006**
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	**V41007**
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	**V41010**
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	**V41011**
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	**V41012**
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	**V41013**
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	**V41014**
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	**V41015**
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	**V41016**
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	**V41017**

Timer Status Bit Map

This table provides a listing of individual timer contacts associated with each V-memory address bit.

MSB	DL05 Timer (T) Contacts														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	**V41100**
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	**V41101**
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	**V41102**
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	**V41103**
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	**V41104**
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	**V41105**
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	**V41106**
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	**V41107**

Counter Status Bit Map

This table provides a listing of individual counter contacts associated with each V-memory address bit.

MSB	DL05 Counter (CT) Contacts														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	**V41140**
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	**V41141**
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	**V41142**
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	**V41143**
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	**V41144**
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	**V41145**
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	**V41146**
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	**V41147**