DL305

Handheld Programmer

Manual Number D3-HP-M

WARNING

Thank you for purchasing automation equipment from PLC***Direct***™. We want your new ***Direct***LOGIC™ automation equipment to operate safely. Anyone who installs or uses this equipment should read this publication (and any other relevant publications) before installing or operating the equipment.

To minimize the risk of potential safety problems, you should follow all applicable local and national codes that regulate the installation and operation of your equipment. These codes vary from area to area and usually change with time. It is your responsibility to determine which codes should be followed, and to verify that the equipment, installation, and operation is in compliance with the latest revision of these codes.

At a minimum, you should follow all applicable sections of the National Fire Code, National Electrical Code, and the codes of the National Electrical Manufacturer's Association (NEMA). There may be local regulatory or government offices that can also help determine which codes and standards are necessary for safe installation and operation.

Equipment damage or serious injury to personnel can result from the failure to follow all applicable codes and standards. We do not guarantee the products described in this publication are suitable for your particular application, nor do we assume any responsibility for your product design, installation, or operation.

If you have any questions concerning the installation or operation of this equipment, or if you need additional information, please call us at 1-800-633-0405.

This publication is based on information that was available at the time it was printed. At PLC***Direct***™ we constantly strive to improve our products and services, so we reserve the right to make changes to the products and/or publications at any time without notice and without any obligation. This publication may also discuss features that may not be available in certain revisions of the product.

Trademarks

Manual History

If you contact us in reference to this manual, remember to include the revision number.

Title: DL305 Handheld Programmers, D3-HP & D3-HPP
Manual Number: D3-HP-M

Issue	Date	Effective Pages	Description of Changes
Original	1/94	Cover/Copyright Contents Manual Revisions 1-1 – 1-15 2-1 – 2-26 3-1 – 3-13 4-1 – 4-9 5-1 – 5-11 6-1 – 6-13 A-1 – A-14	Original Issue
REV A	3/96		Minor changes throughout
Rev. B	5/98	Entire Manual 1-15, 2-5 and 2-15	Downsized to spiral version. Minor changes

Table of Contents

Chapter 1: Getting Started

Chapter 2: Entering RLL Programs

Chapter 3: Entering RLL*PLUS* Programs

Chapter 4: Changing Programs

Chapter 5: Protecting and Storing Programs

Chapter 6: System Monitoring and Troubleshooting

Appendix A: DL305 Memory Map

Getting Started

In This Chapter. . . .

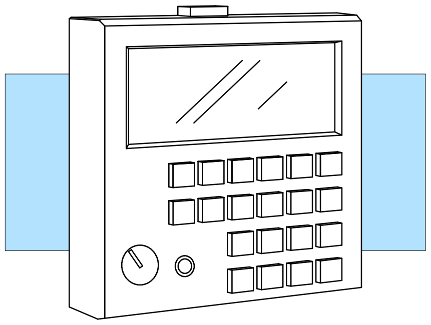
Introduction

DL305 Handheld Programmer

The DL305 Handheld Programmer is a general purpose programming tool for use with the DL305 family of automation products.

The Handheld is well suited for entering small programs or for troubleshooting machine operations. It is not the ideal choice for entering larger, more complex programs. For these types of programs, you should consider using ***Direct***SOFT, our PC-based programming software.

DL305 Handheld



There are two versions of Handheld Programmers available.

- D3-HP — RLL version for all RLL CPUs
- D3-HPP — RLL*PLUS* version for all RLL*PLUS* CPUs

RLL*PLUS* is just like normal RLL, but a few instructions have been added that make it much easier to use and understand. Programs are usually much shorter and considerably easier to troubleshoot. The best thing to do right now is to make sure you have the correct version of Handheld for use with your CPU. (Trust me, it’s easier this way.)

Purpose of this manual

This manual will teach you the basic keystrokes used with the Handheld. It does not provide an example of every instruction. Once you understand the basic keystroke techniques, you should use the DL305 User Manual to determine the instruction operation details and keystroke requirements for the individual instructions.

Since we constantly try to improve our product line, we occasionally issue addenda that document new features and changes to the products. If there are addenda included with this manual, please read through them to see which areas of the manual or product have changed.

Who should read this manual

If you understand the DL305 instruction set and system setup requirements, this manual will provide all the information you need to get a basic understanding of the Handheld. This manual *is not* intended to be a tutorial on the DL305 instruction set or system operation, but rather a user reference manual for the Handheld Programmer.

How this manual is organized

Ch 1: Getting Started – this chapter provides an overview of the Handheld Programmer, general specifications, and the basic things you need to start entering programs.

Ch 2: Entering RLL Programs – discusses all the operations used to enter a program.

Ch 3: Entering RLL^{PLUS} Programs – provides the keystrokes needed to enter RLL^{PLUS} programs.

Ch 4: Changing Programs – shows you how to quickly edit an existing program.

Ch 5: Protecting and Storing Programs – shows you how to store programs on cassette tapes.

Ch 6: System Monitoring and Troubleshooting – provides an overview of the various features used to monitor and troubleshoot the system.

Appendix A: DL305 Memory Map – provides a detailed listing of the DL305 memory map for I/O, timers, counters, etc.

Supplemental Manuals

There is another manual that may occasionally be referenced by this manual. This manual is not absolutely necessary to use the Handheld, but it does provide additional details on several related subjects.

- DL305 User Manual (D3–USER–M)

Now, you know what material is necessary to quickly understand the DL305 Handheld Programmer. So, let's get started!

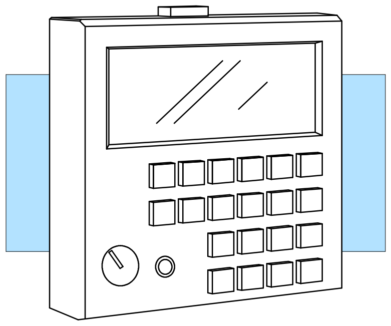
How can I use the Handheld?

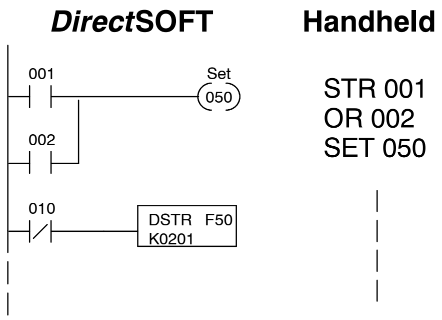
As a Programming Tool

The DL305 Handheld Programmer is ideally suited for entering or changing small programs with instruction mnemonics. You can enter programs up to the limits of the CPU you are using, but larger programs are much easier to design and enter with ***Direct***SOFT Programming Software.

In addition to entering programs, the Handheld is ideal for making on-site program or system changes.

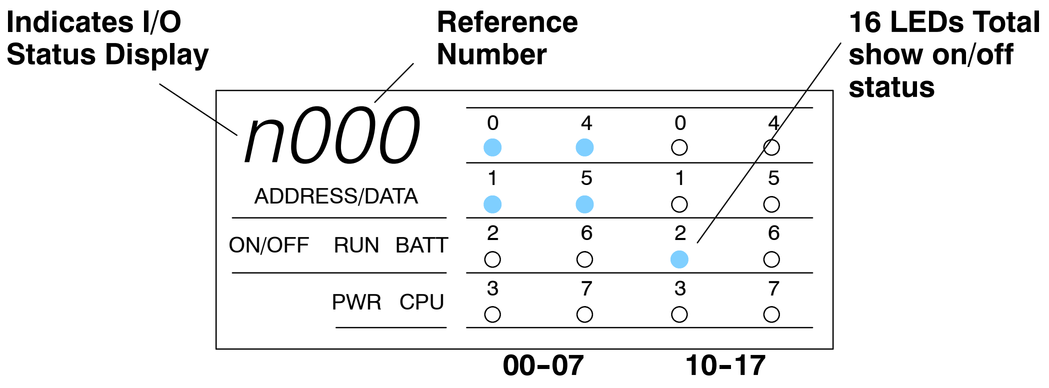
Since the Handheld has a built-in cassette tape interface, you can also use it to store and load programs from cassette tapes.





To Monitor Machine Operations

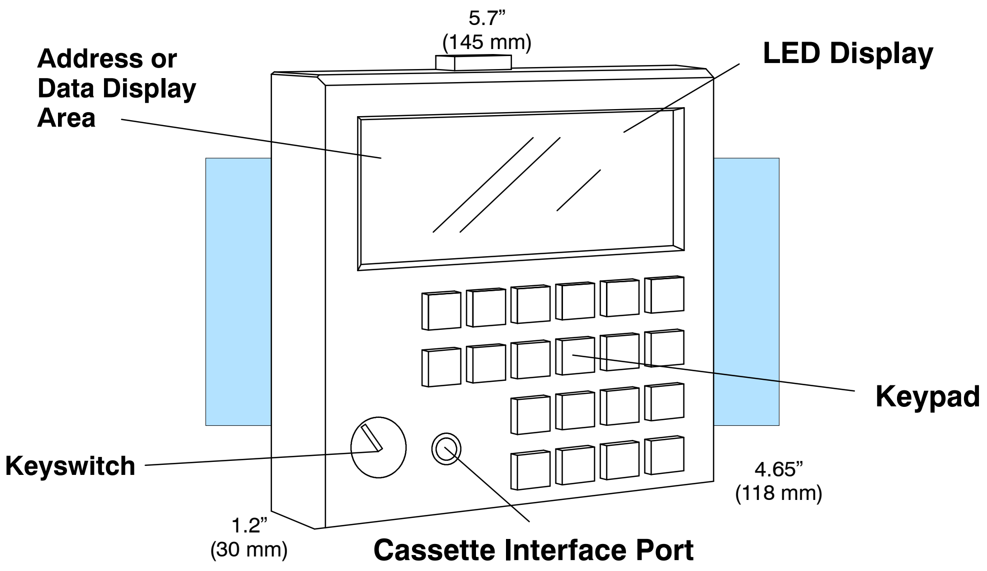
The Handheld is especially useful if you need to quickly look at the status of an I/O point, timer/counter value, or register location. You can monitor up to 16 I/O points at one time. For example, the following diagram shows how the Handheld display area indicates I/O status.



Physical Characteristics and Specifications

Handheld Layout

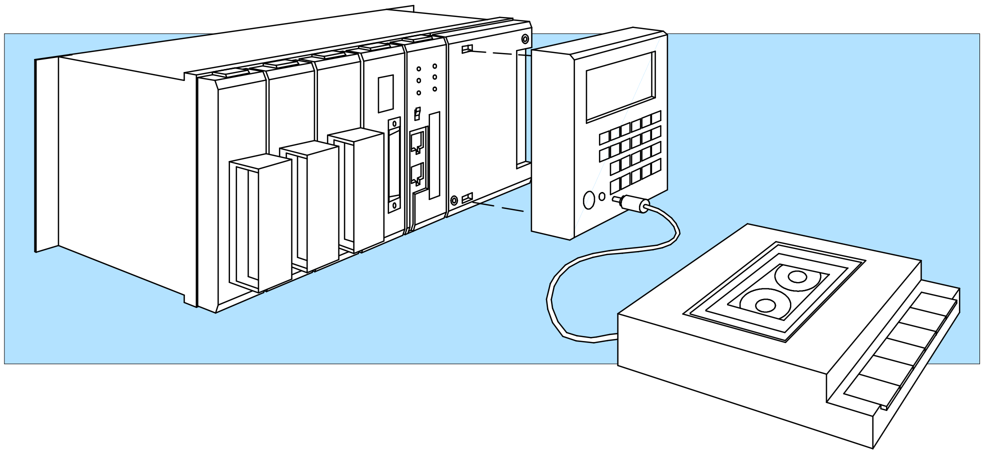
The Handheld was designed to be more than a program entry tool and includes many different status LEDs that make it easy to understand the machine operations.



Connection Options

You can mount the Handheld directly to the CPU, or you can use a cable. The cable, part number D3-HPCBL-1, is approximately 4.6 feet (1.5m) in length and provides much more flexibility.

A cassette interface cable, supplied with the Handheld Programmer, is required to connect a cassette recorder.



WARNING: The CPU will automatically change modes when you connect the Handheld Programmer if the keyswitch is set for a different mode of operation. For example, if the CPU is in Run mode and the Handheld Programmer keyswitch is set to the PRG (Program) position, the CPU will automatically enter Program mode when the Handheld is connected.

Specifications The following table provides specifications for the DL305 Handheld Programmer.

Environmental	
Operating Temperature	32° to 140 F° (0° to 60 C°)
Storage Temperature	-4° to 176 F° (-20° to 80 C°)
Humidity	5 to 95% (non-condensing)
Environmental Air	No corrosive gases
Vibration	MIL STD 810C 514.2
Shock Resistance	MIL STD 810C 516.2
Noise Immunity	NEMA ICS3-304, impulse 1KV, 1□s
Power	Obtained through PLC port, 60 mA @ 5 VDC 60 mA @ 9 VDC
Dimensions	4.3” L x 4.7” H x 0.9” D 110mm W x 118mm H x 24mm D
Weight	7.5 oz. (210 g)

CPUs Supported	**Programming Operations**
DL330, DL330P, DL340	Read, Write, or erase programs
Simatic® TI315™, TI325™, TI330™, TI335™, plus stage versions*	Insert or delete an instruction
Texas Instruments® TI315™, TI325™, TI330™, TI335™, plus stage versions*	Search for a specific instruction
* Stage versions require the D3-HPP	Locate a specific address
	Read or write to cassette tapes

Cables	**Machine Monitoring Operations**
D3-HPCBL-1, 1.5m programmer cable	I/O status (up to 16 simultaneously)
	On / Off status for contacts, coils, control relays, and register locations
	Timer and counter current values
	Debugging Operations
	Forcing (one scan only)
	Run and Program Mode display
	Program syntax check
	Predefined error codes

Handheld Basics

Status LEDs and Key Groups

When you enter a program, you need to be able to select the instruction, enter any parameters for that instruction, and move to the next task. The Handheld keypad is organized into LED display areas and key areas that make this task easier.

As you examine the keys, you'll notice some of the keys have more than one label. The top label describes the key when the Shift (SHF) key is pressed. (These keys work just like the number keys on a computer keyboard.)

The keys and LEDs areas are as follows.

- Instruction identifier and numeric keys — used to select the type of instruction. Also used to enter numeric values for instruction references and constants (by pressing SHF first).
- Editing keys — used during program entry and editing to scroll through addresses, insert and delete instructions, etc. These same keys also have Shift functions that are primarily used during cassette or machine monitoring operations.
- Address / Data Display — this 4-character, seven-segment display shows the address, reference number (such as the I/O point being used with an instruction), or data value (such as the current value for a timer.)
- Instruction LEDs — show the type of instruction used at the address being displayed.
- CPU Status LEDs — show the status for Power, CPU mode, etc.

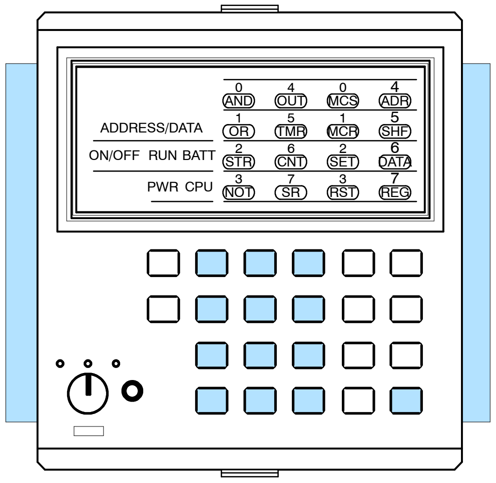
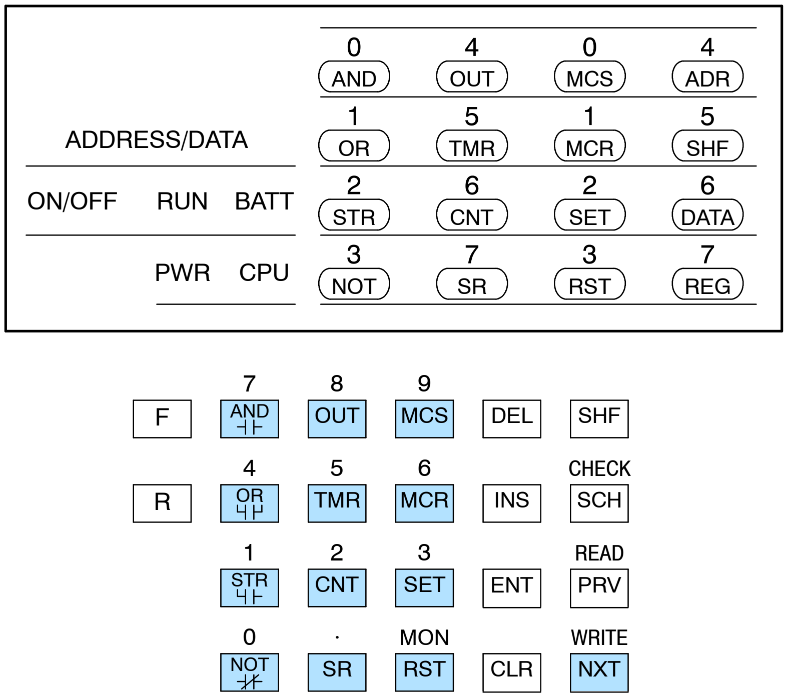
RLL*PLUS* vs. RLL Units

As mentioned earlier there are differences between the two models of DL305 Handheld Programmers. This difference is clearly visible if you examine the keypad and display layout shown on the following page. This manual uses the DL3-HP (RLL version) for most of the examples. This is because, for most instructions, the only difference between the two versions is the key location or the location of the display LED. The key titles are the same, they're just in different locations.

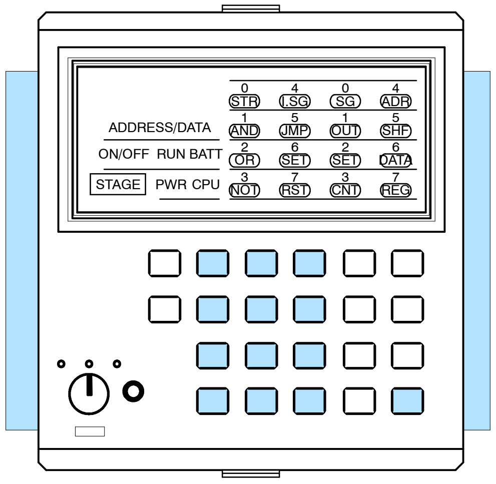
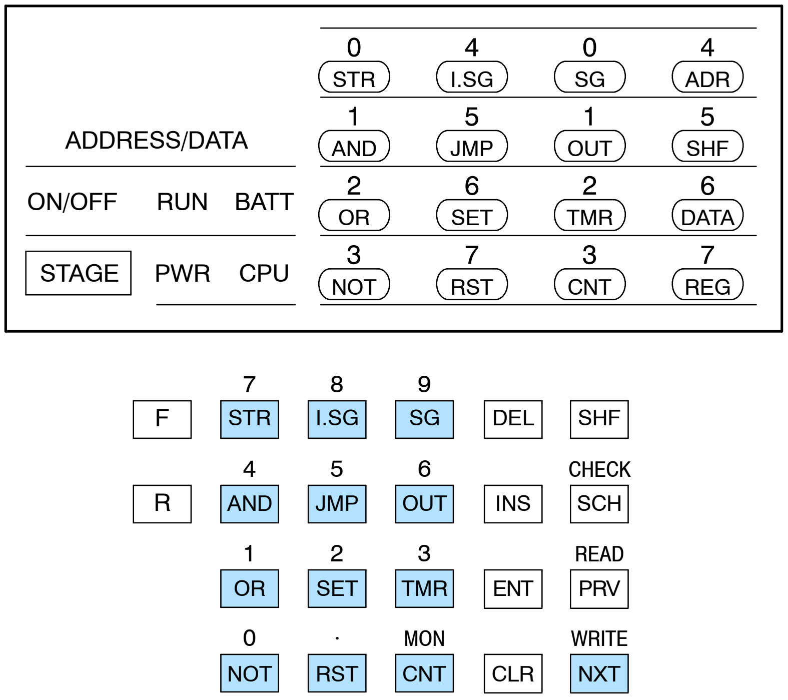
You may have noticed we said the two versions are the same for *most* instructions. The RLL*PLUS* version does have keys for the Master Control Relay Set (MCS), Master Control Relay Reset (MCR), and Shift Register (SR) instructions, but instead has keys for the extra instructions required for RLL*PLUS* programs.

The following diagram shows the key areas, LED areas, and differences between the two types of units.

RLL Version

RLL*PLUS* Version

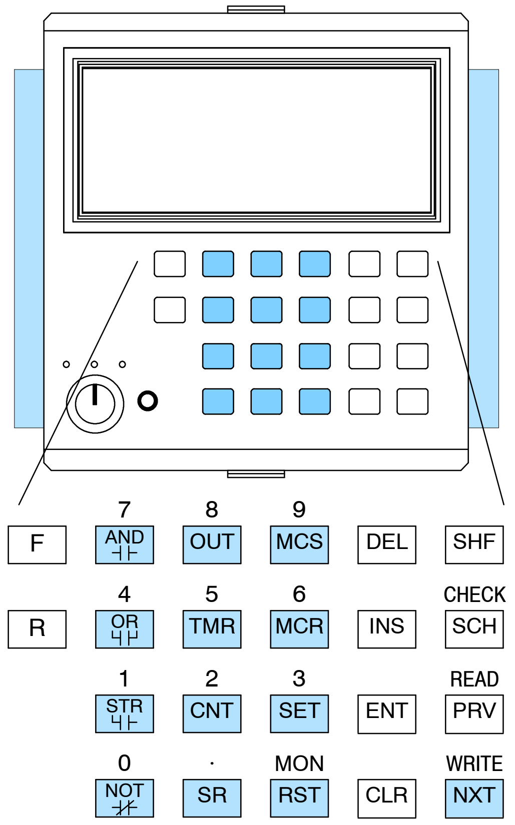



Instruction Identifier and Numeric Keys

The identifier keys are used to specify the exact instruction type and the instruction reference. For example, if you want to store a contact, you have to specify the STR instruction and which contact you want to use.

For example, to enter Store I/O point 001, you would press STR, SHF, 1 and ENT.

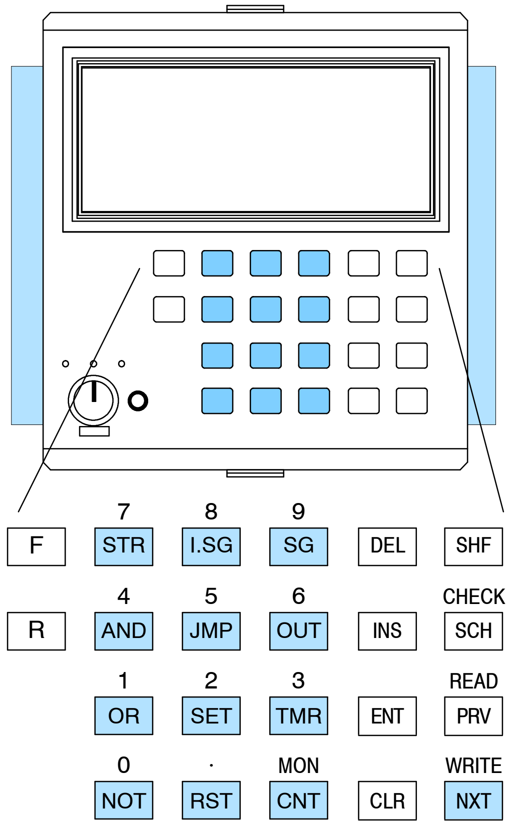
A timer instruction would work the same way. In this case, you would press TMR, SHF, 6, 0, 0, ENT to load the timer. To enter the constant, you would then press SHF, 2, 0, ENT. (This would load a preset of 20.)



Editing Keys

These keys are used to perform various operations during program entry and editing. For example, you can use these keys to insert (INS), delete (DEL), or search (SCH) for a specific instruction.

These keys also have shift functions that are primarily used during cassette tape operations. However, there is one key, Monitor (MON), that is used when you want to monitor the status of an I/O point, timer/counter value, or register location.

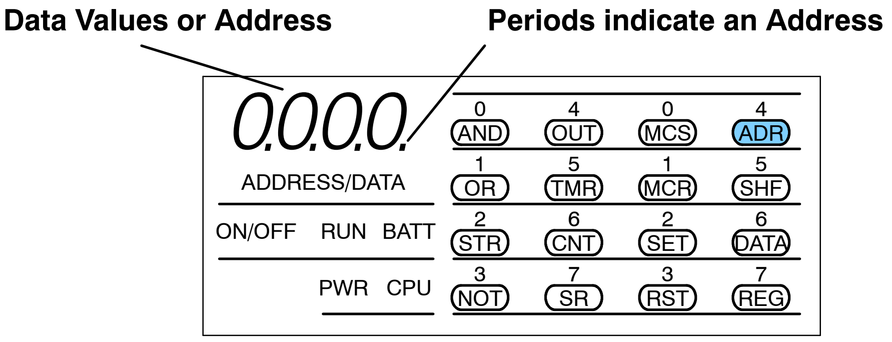


Address / Data Display Area

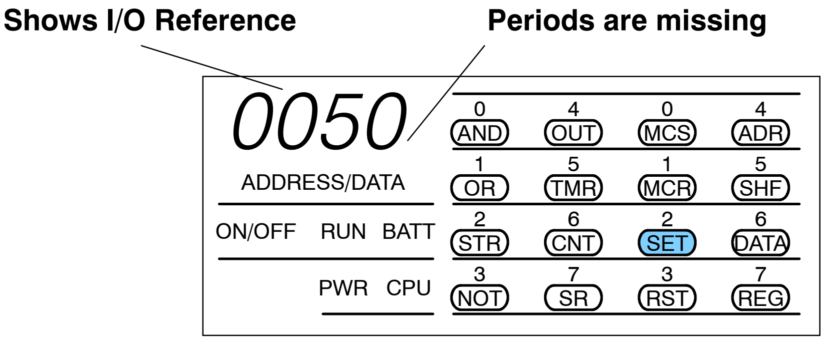
The Address / Data primarily shows two things.

- For programs, it shows address locations or instruction reference numbers.
- For monitoring operations, it shows the current value of timers, counters, and registers.

How do you know which one you're seeing on the display? Simple, whenever an address is shown there are periods that follow each digit. If the periods are missing, you're seeing a data value, a constant, an I/O reference, etc.

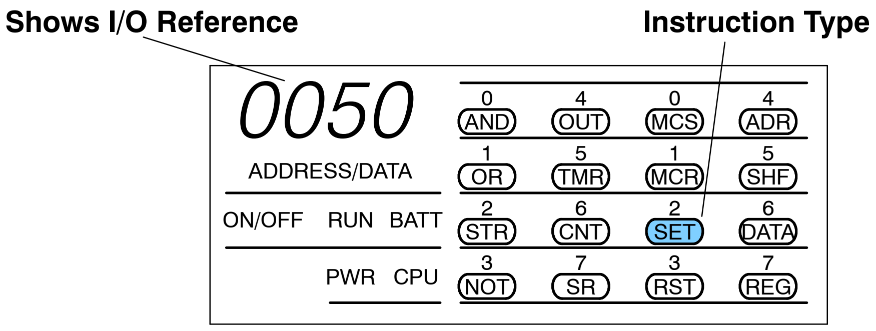


Since the display area can show two types of information, you can easily switch between the two types by pressing CLR and NXT. The following display shows an example of the display if a SET instruction was loaded at address 0000 and you pressed the CLR key. (You can then press NXT to return to the address display.)



Instruction LEDs

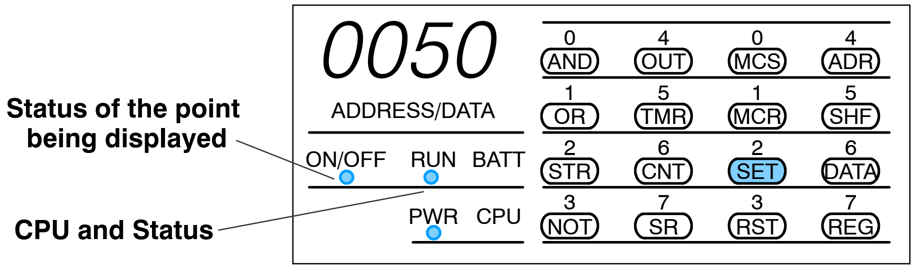
The instruction LEDs show you which instruction is used at the address being displayed. For example, if a SET instruction is located at address 0000, then the SET LED would be on.



These LEDs are also used during monitoring operations to show the On/Off status for up to 16 points. (We'll discuss this in more detail in Chapter 6.)

CPU Status LEDs

The CPU LEDs show you the mode of operation, battery status, power indication, and CPU error condition (if any exists). The ON/OFF LED shows the status for the individual instructions as you step through the program during Run mode. For example, if the instruction was SET 050 and the CPU was in Run mode, then the display would appear as follows.



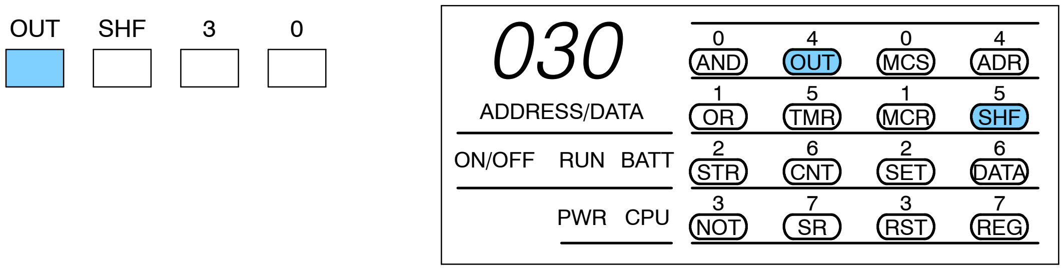
(In Run mode, you can also toggle between the address display and the status display by pressing CLR and NXT. Remember, the address display would have the periods as shown earlier.)

Clearing the Display Area

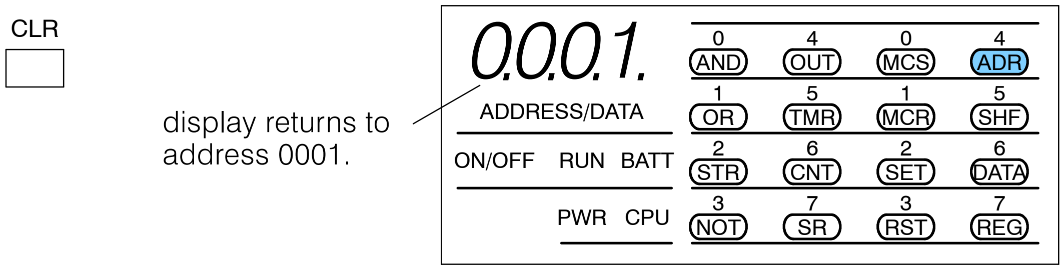
Sometimes we all make mistakes, so it's important to know how to clear the display and start from the beginning. Since the Handheld Programmer buffers the keystrokes until you press ENT, you can clear the display at any time up until the ENT button is pressed. When you press CLR, the Handheld clears the keystrokes you've entered and remains at the current address. At this point you can now enter the correct instruction.

Consider the following example that starts at address 0001.

Keystroke Error (should have used SET instead of OUT)



Press CLR



CPU Setup

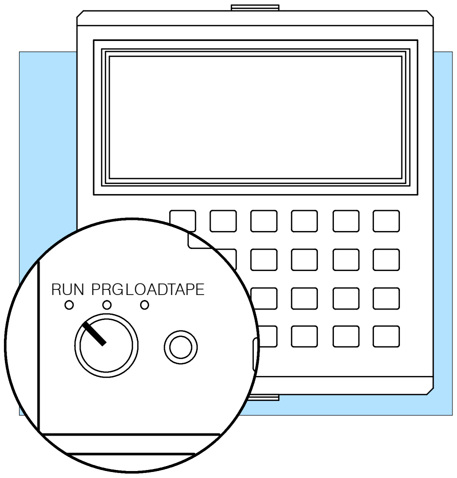
Even if you have years of experience using PLCs with handheld programmers, there are a few things you may need to know before you start entering programs. This section includes some basic things, such as changing the CPU mode and clearing the CPU memory.

Changing the CPU Modes

There are two modes available with the DL305 CPUs.

- RUN — executes program and updates I/O modules
- PGM — allows program entry, does not execute program or update I/O modules

You can only change the CPU mode by using the keyswitch on the front of the handheld programmer.



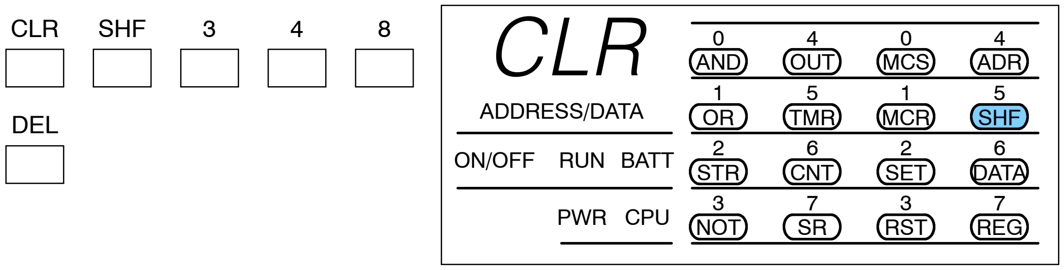
WARNING: The CPU will automatically change modes when you connect the Handheld Programmer if the keyswitch is set for a different mode of operation. For example, if the CPU is in Run mode and the Handheld Programmer keyswitch is set to the PRG (Program) position, the CPU will automatically enter Program mode when the Handheld is connected.

The keyswitch also has a third position, called LOAD. If the keyswitch is in this position you can upload a program from CPU memory to a cassette tape, or download a program from cassette tape to CPU memory.

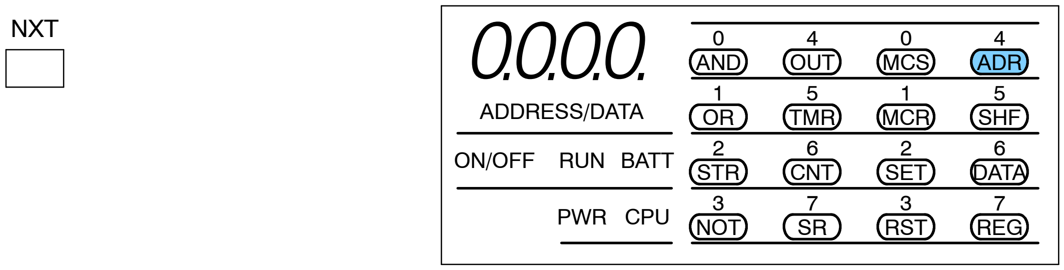
Clearing an Existing Program

Before you enter a new program, you should always clear the CPU memory. Only a few keystrokes are required.

Use these keystrokes



Press NXT to clear memory
or
Press CLR to abort the operation



You've got the Basics!

Now that you understand how to use the display and how to perform some basic CPU operations, you're ready to enter a program. Chapter 2 provides the keystrokes for entering many of the instructions you'll use in any type of program. If you plan on using RLL*PLUS* programs, you'll also want to look at Chapter 3 for information on the the extra instructions for RLL*PLUS*. Also, with RLL*PLUS*, some of the basic instructions, like timers and counters, are used differently.

2

Entering RLL Programs

In This Chapter. . . .

Entering Simple Ladder Programs

Purpose of the Examples

This section includes many examples that are intended to help you become familiar with the keystrokes required to enter the most basic DL305 instructions. Once you are familiar with the basic keystrokes, you should use the DL305 User Manual as a reference for the remaining instructions.

Handheld Key Sequences

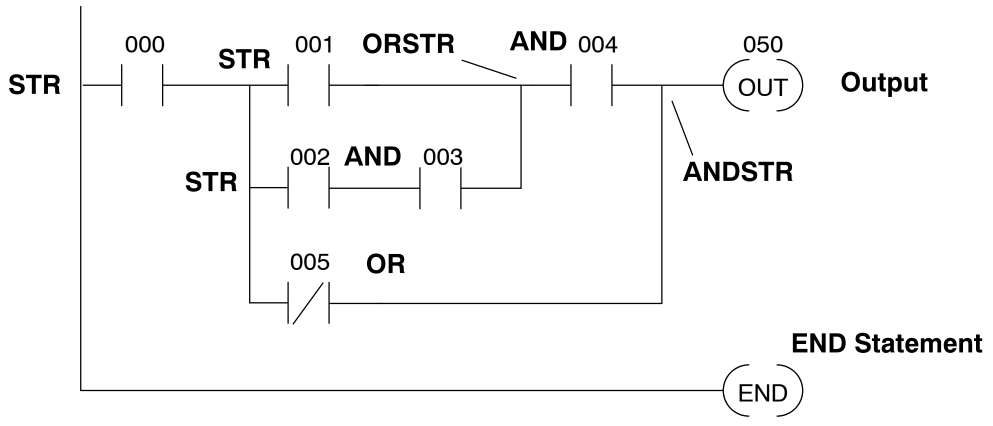
The Handheld buffers all keystrokes until you press the ***ENT*** key. Then, it automatically checks the instruction to make sure that is has been entered correctly. If the instruction is entered incorrectly an error message will be displayed. See Chapter 6 for a complete listing of error messages.

The Basics

There are a few basic instructions that you must become familiar with to enter programs with the Handheld.

- STR – Stores a normally open element and indicates the beginning of a rung or network.
- AND – Joins one element (such as a contact) in series with another element or group of elements.
- AND STR – Joins a group of elements in series with another group of elements.
- OR – Joins a one element in parallel with a previous element or group of elements.
- ORSTR – Joins parallel branches (each branch must begin with a STR instruction)
- OUT – Each rung must have at least one output (Y, C, or box instruction)
- NOT – used with other instructions to utilize normally closed elements.
- All programs must contain an END statement (automatically provided).

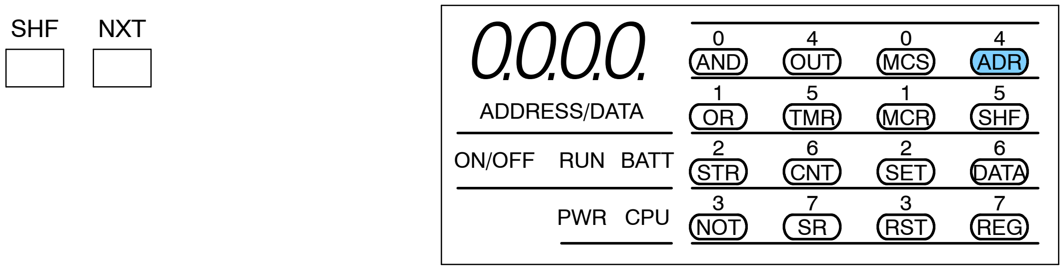
The following diagram shows a typical network and how each of these elements are used.



Starting at Address 0

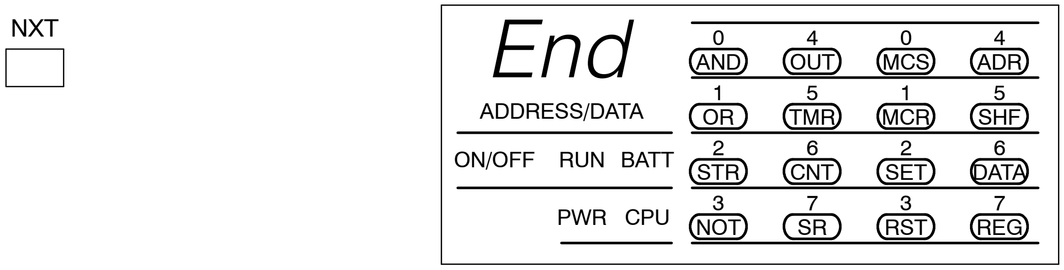
If you're entering a complete program, you should always start at Address 0. The following example shows the keystrokes required to start at address 0000. (The remaining examples will not show this step, but it is required.)

Start at address 0 **Address Display**



Once you're at address 0, you can start entering a program. After you start entering the program, the Handheld automatically increments to the next address after you enter an instruction. You can toggle from the address display to the data display by pressing the NXT key. You can toggle from the data display to the address display by pressing the CLR key. For example, if you start at address 0000 and press NXT, the display changes and shows the instruction type located at address 0000. The following example shows what the display would look like.

Start at address 0, change to data display instead of address display



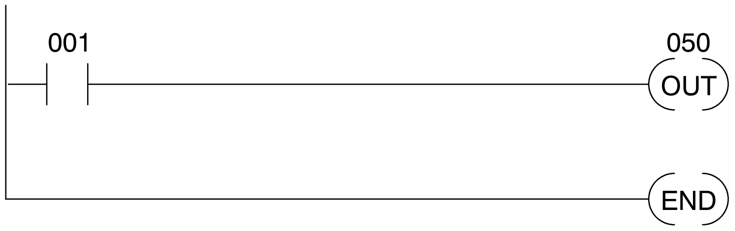
The example keystrokes shown throughout this chapter will indicate which display method is being used. This will make the examples easier to follow. If you prefer a different display you now have the means to change it.

Entering an END Statement

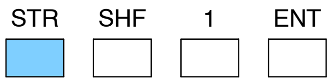
All DL305 programs must have an END statement as the last statement in the program. Whenever you clear the CPU memory, the CPU assumes that *all* memory locations contain an END statement. This means that you do not have to enter an END statement. Just enter your program starting at address 0. You should be aware that if an END statement precedes your ladder logic, the program will not be executed.

Entering Simple Rungs

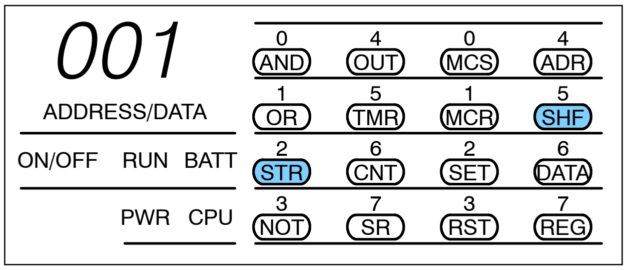
You use the STR instruction to start rungs that contain both contacts and coils. The following example shows how to enter a single contact and a single output coil. Remember, with the DL305 CPUs, you do not have to enter an END statement with the Handheld Programmer. In the following example, notice that when you enter the output and move to the next address, the END statement is already there.



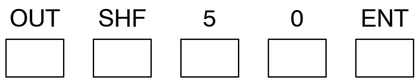
Enter the contact

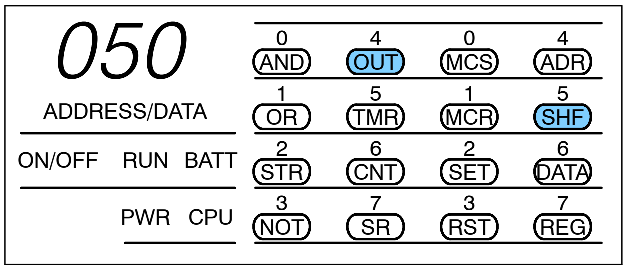


Data display *before* ENT is pressed

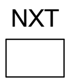


Enter the output

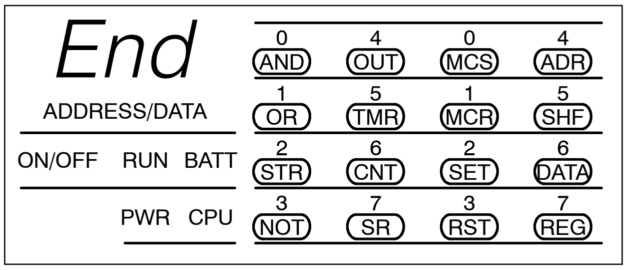




Check the next address



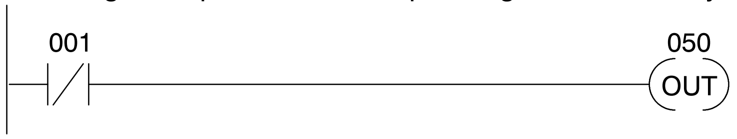
END Statement

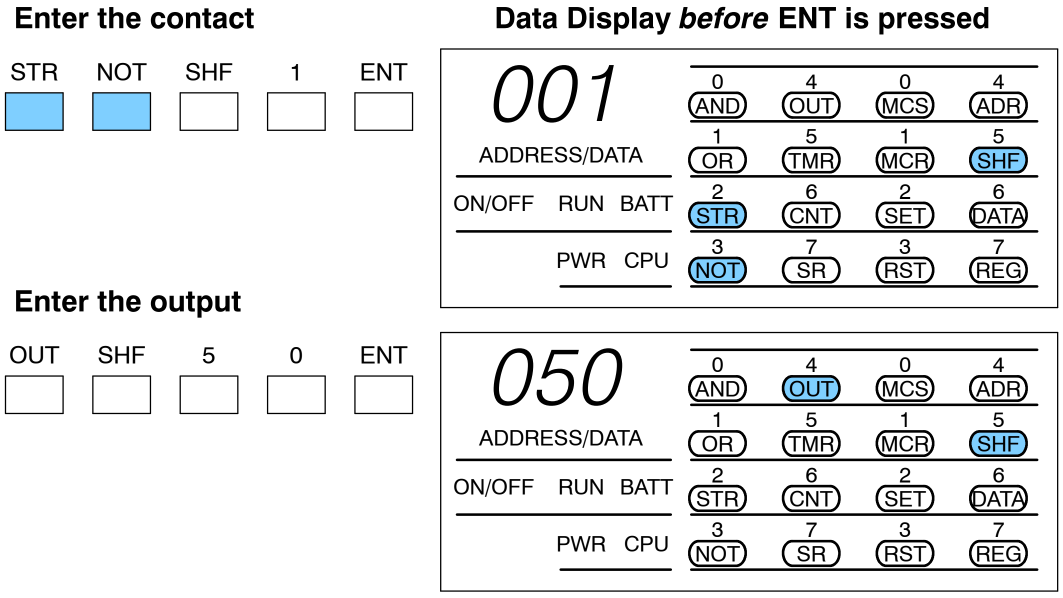


The example shows an input contact and an output coil. Control relays are entered exactly the same as the I/O points.

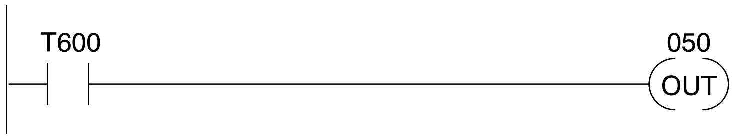
Entering Normally Closed Elements

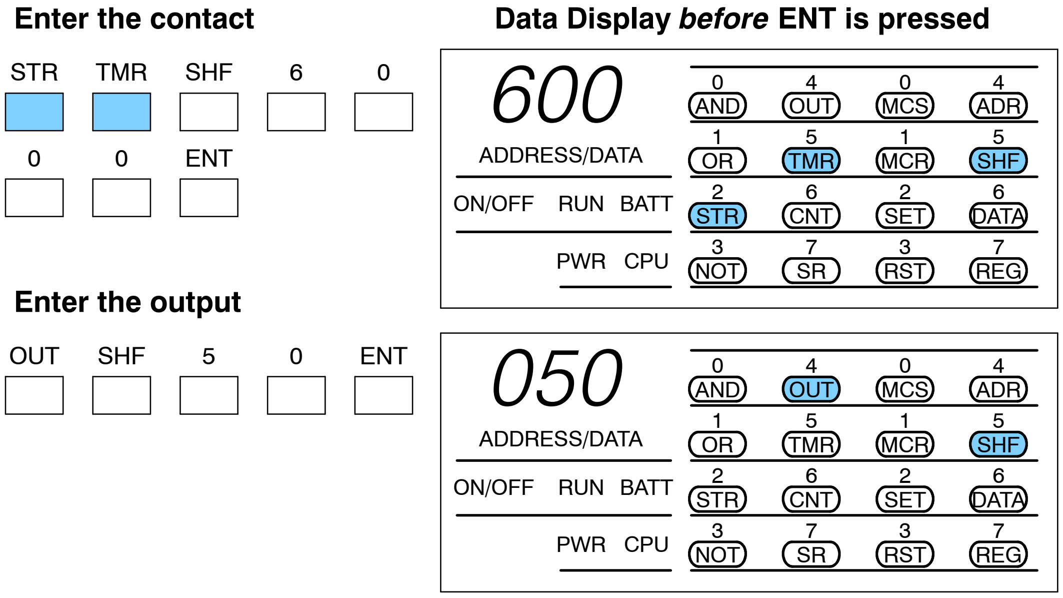
Normally closed elements are entered by using the STR and NOT instructions. The following example shows a simple rung with a normally closed contact.



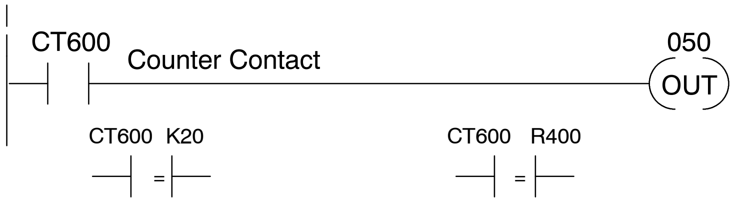


Entering Timer/Counter Contacts



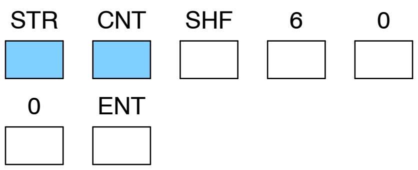


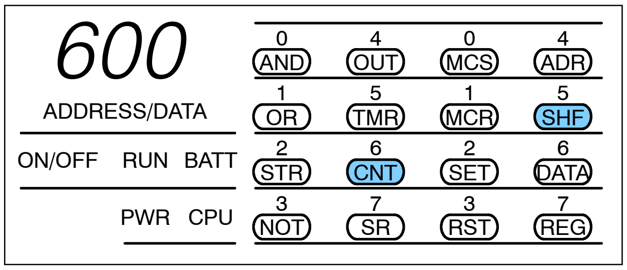
You may also need to enter timer and counter contacts or relational contacts based on counter values. These types of contacts are entered slightly differently than normal input contacts. The following example shows the keystrokes.



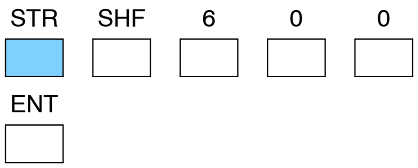
Compared to a constant Compared to register value

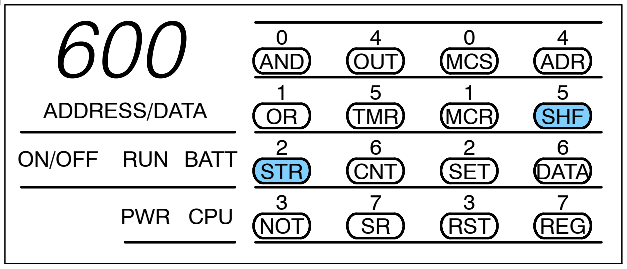
To enter a Timer/Counter contact **Data Display *before* ENT is pressed**



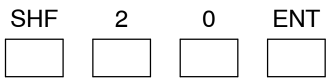


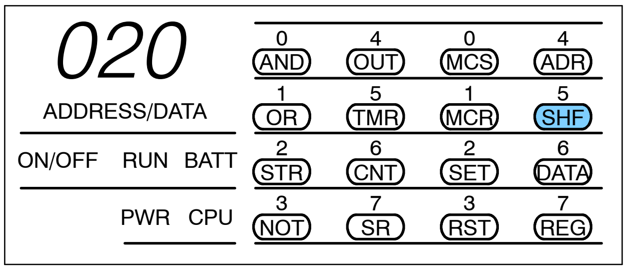
To enter a Timer/Counter comparative contact with ...





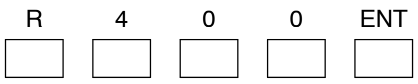
a constant value

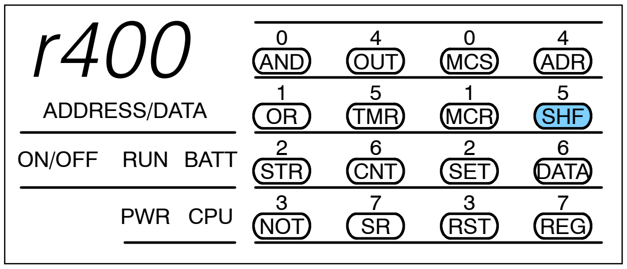




or

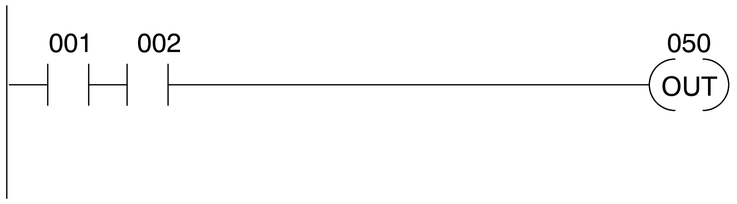
a data register



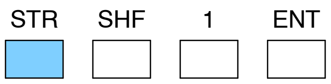


Entering Series Elements

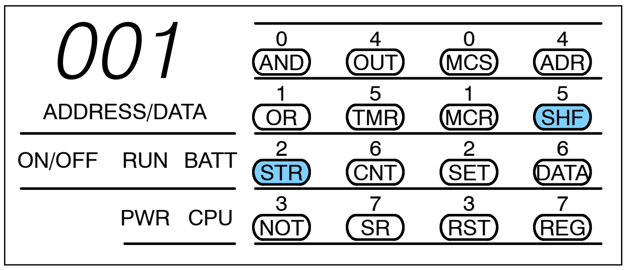
You must program the first element in a rung with a STR instruction, since it is the beginning of the network. The rung can contain more than one element joined together in series by AND instruction(s). The following example shows how to enter two series contacts and a single output coil.



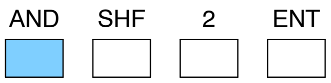
Enter the first contact

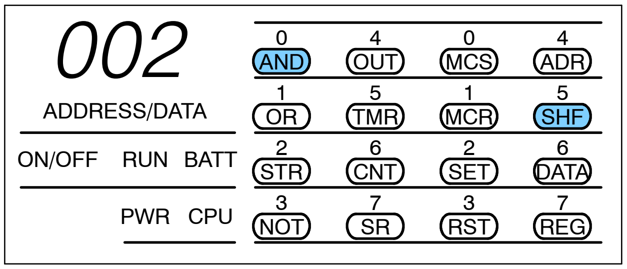


Data Display *before* ENT is pressed

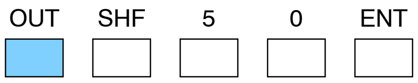


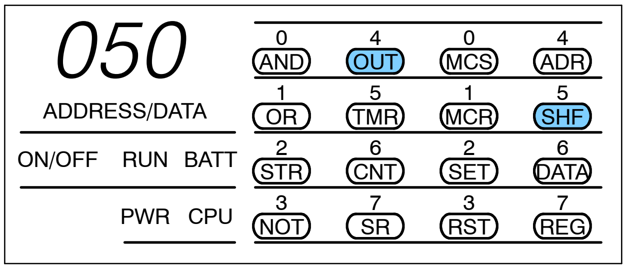
Enter the second contact





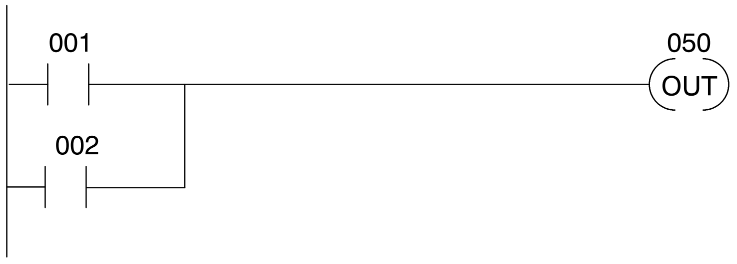
Enter the output





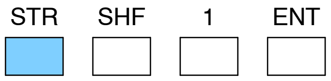
Entering Parallel Elements

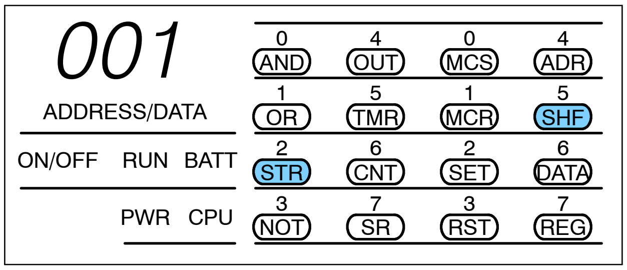
You must program each element with a STR instruction. The elements are joined in parallel by OR instruction(s). The following example shows how to enter two parallel contacts and a single output coil.



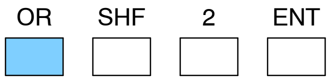
Enter the first contact

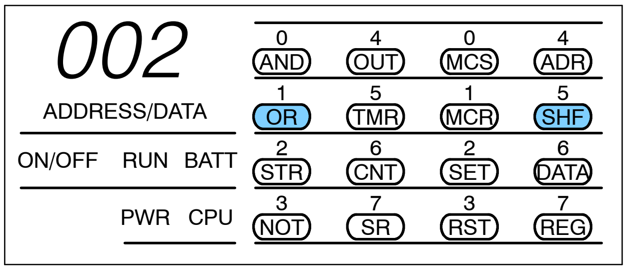
Data Display *before* ENT is pressed



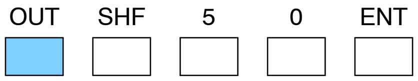


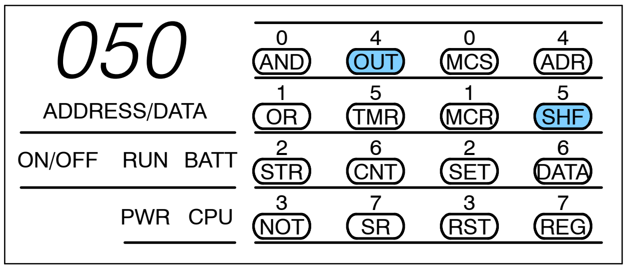
Enter the second contact





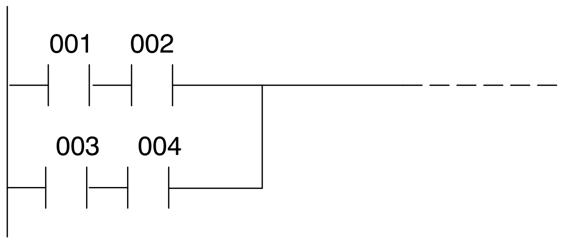
Enter the output





Joining Series Branches in Parallel

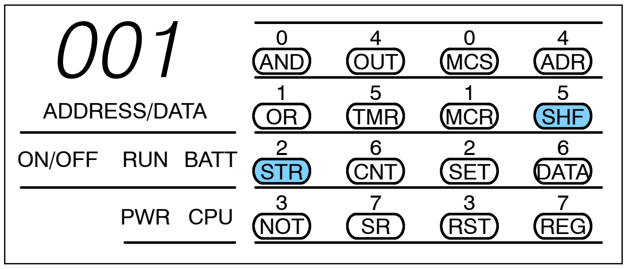
Quite often it is necessary to join several groups of series elements in parallel. The OR STR instruction allows you to do this quite easily. The following example shows a simple network consisting of series elements joined in parallel.



Enter the first contact

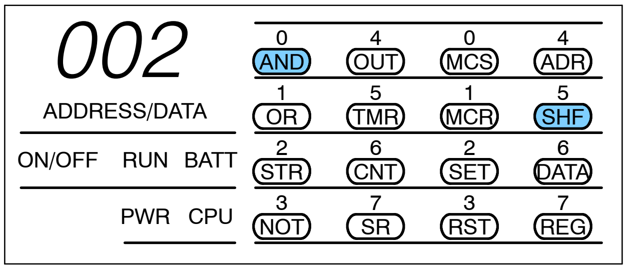
STR SHF 1 ENT

Data Display *before* ENT is pressed



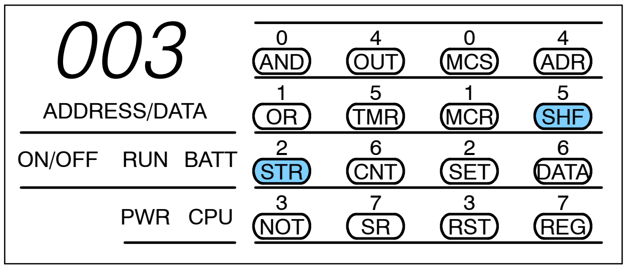
Enter the second contact

AND SHF 2 ENT



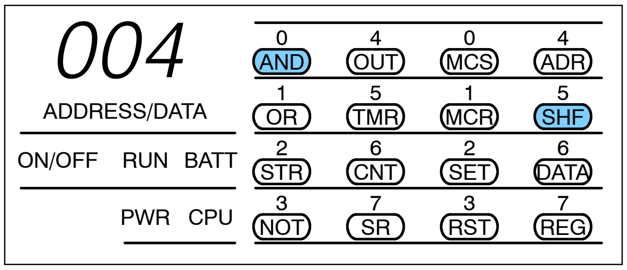
Enter the third contact

STR SHF 3 ENT



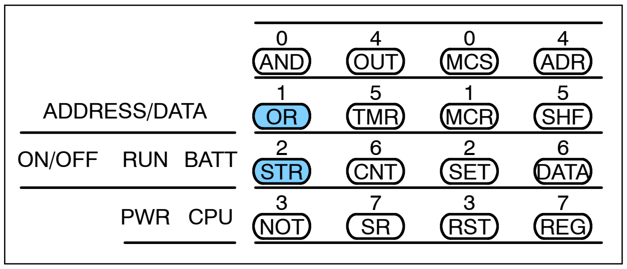
Enter the fourth contact

AND SHF 4 ENT



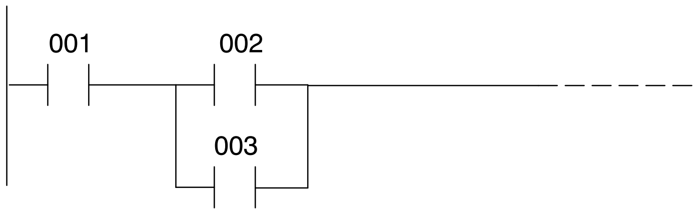
Join the branches

OR STR ENT

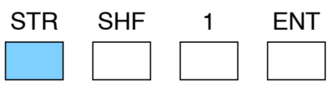


Joining Parallel Branches in Series

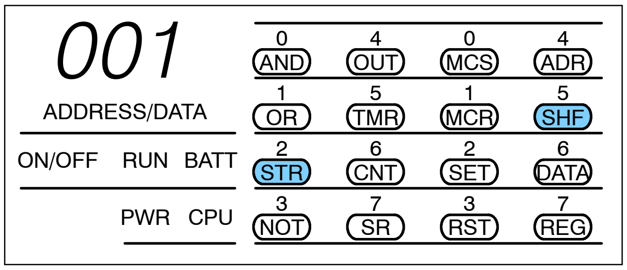
The AND STR instruction joins one or more branches in series. The following example shows a simple network with parallel and series branches.



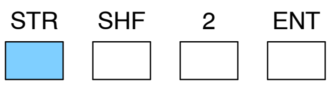
Enter the first contact

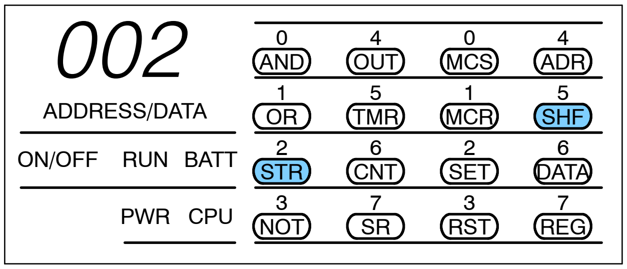


Data Display *before* ENT is pressed

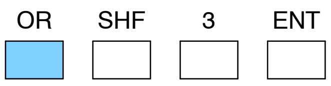


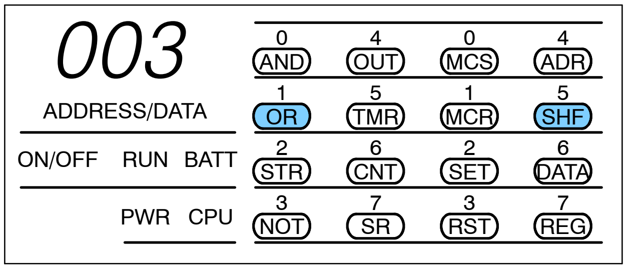
Enter the second contact



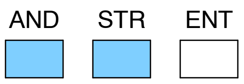


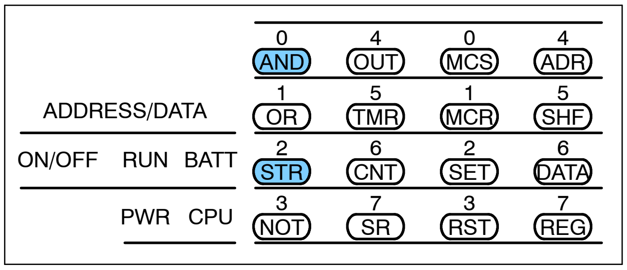
Enter the third contact





Join the elements

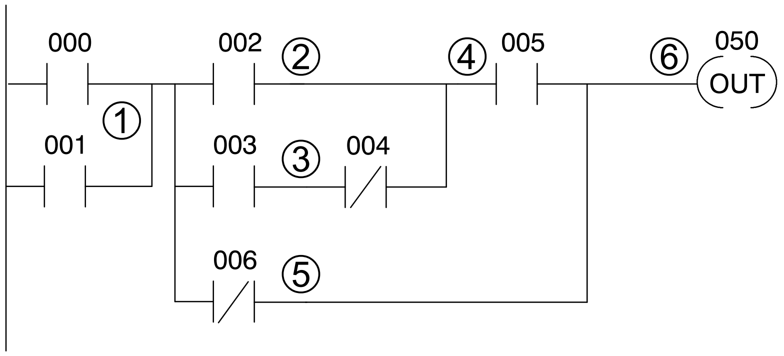




Combination Networks

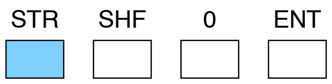
You can combine the various types of series and parallel branches to solve most any application problem. It is doubtful that you will ever exceed them, but there are limits when you build complex networks. This is because the DL305 CPUs use a “stack” to evaluate the boolean elements. If the stack exceeds eight levels, an error code E03 will be displayed on the Handheld when the CPU is switched to Run mode. (See the DL305 User Manual for additional information.)

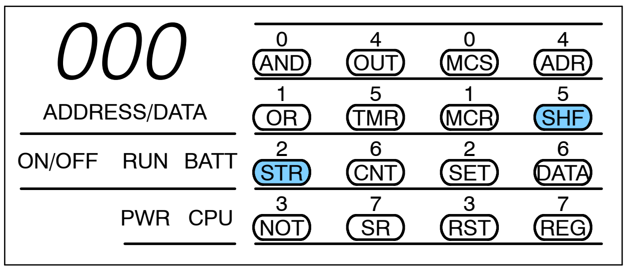
The following example shows a simple combination network.

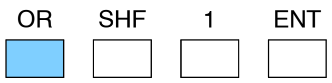


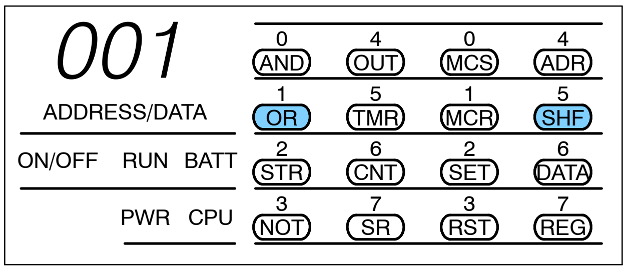
①. Start the network

Data Display *before* ENT is pressed

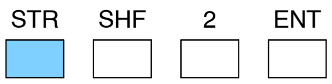


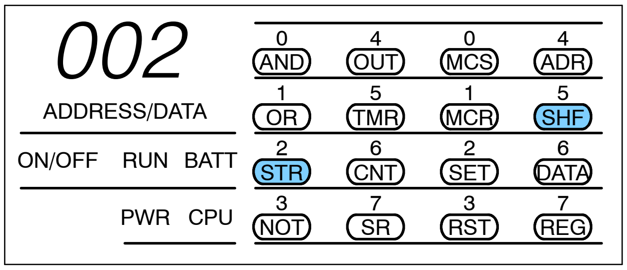


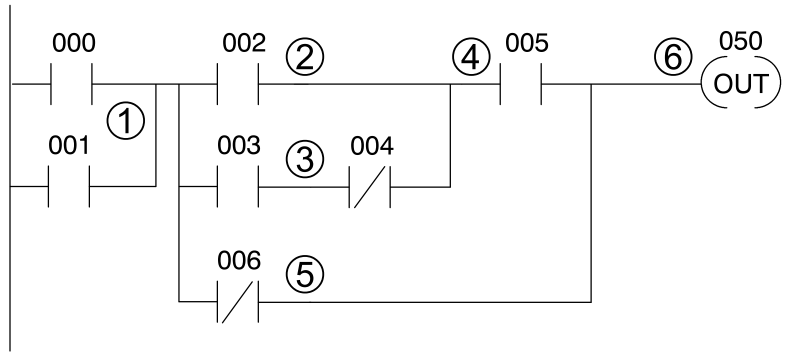




②. Enter branch 2.

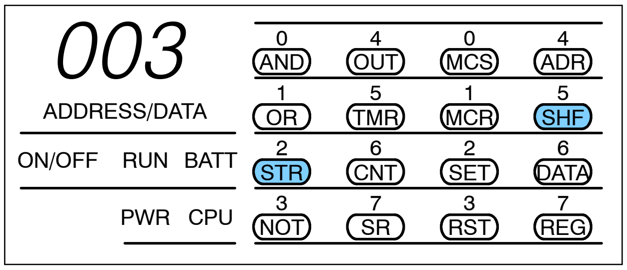




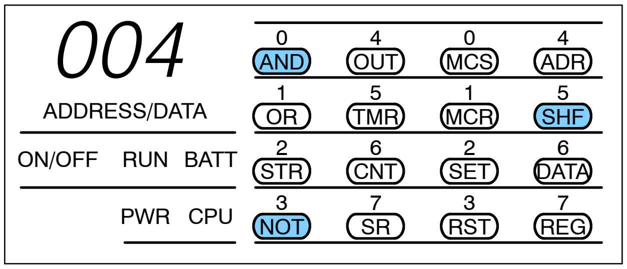


③. Start branch 3, add join with branch 2

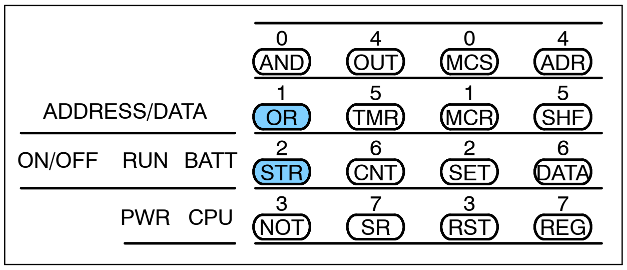
STR SHF 3 ENT



AND NOT SHF 4 ENT

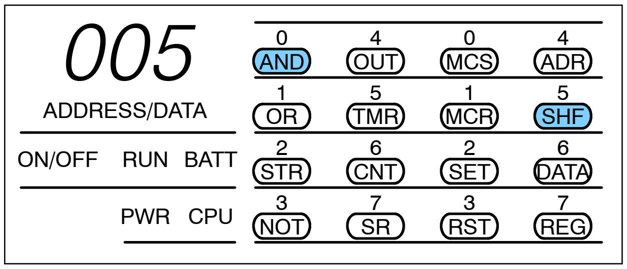


OR STR ENT



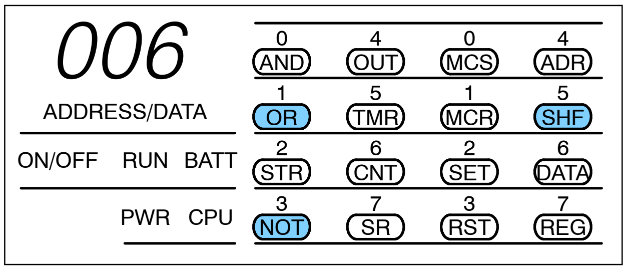
④. Add branch 4

AND SHF 5 ENT

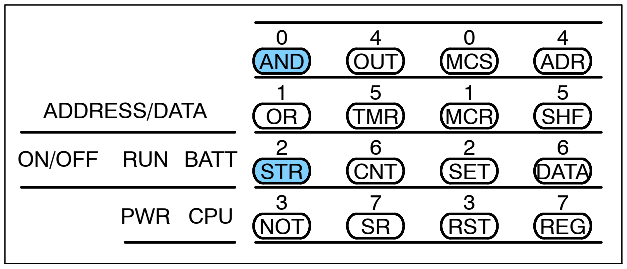


⑤. Add branch 5, join with branches 1–4

OR NOT SHF 6 ENT

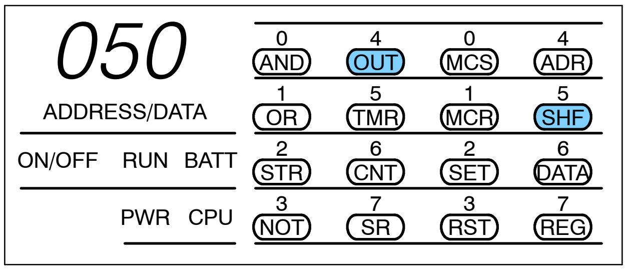


AND STR ENT



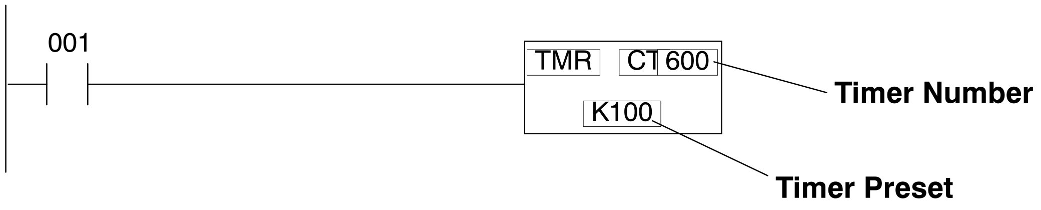
⑥. Add branch 6

OUT SHF 5 0 ENT

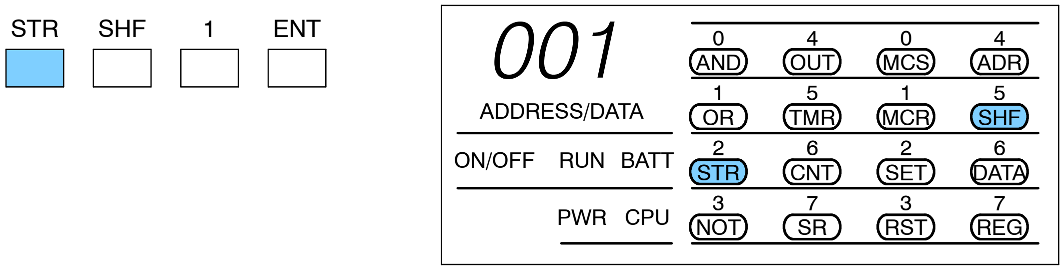


Entering Timers and Counters

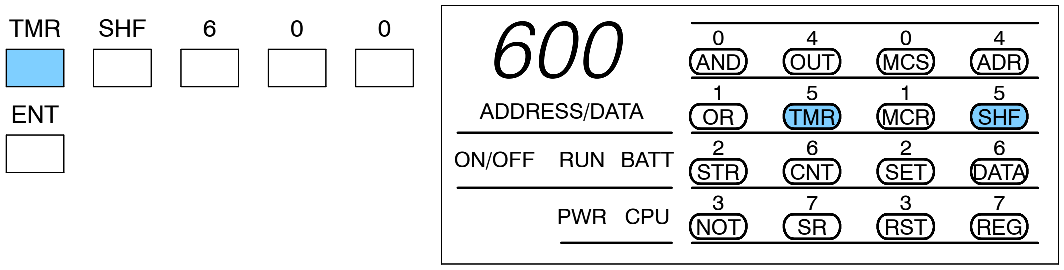
To enter a timer, you also have to enter a preset value. One important thing to remember is that with the DL305, the timers and counters share the same memory area. The range is from 600 – 677, but if you use 600 as a timer, you cannot use it as a counter.



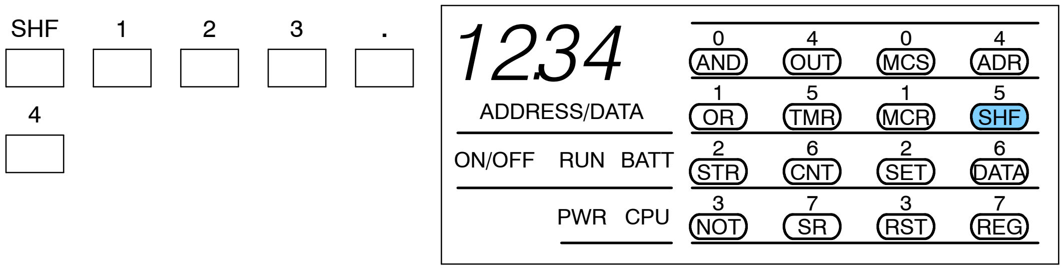
Enter the contact



Enter the Timer



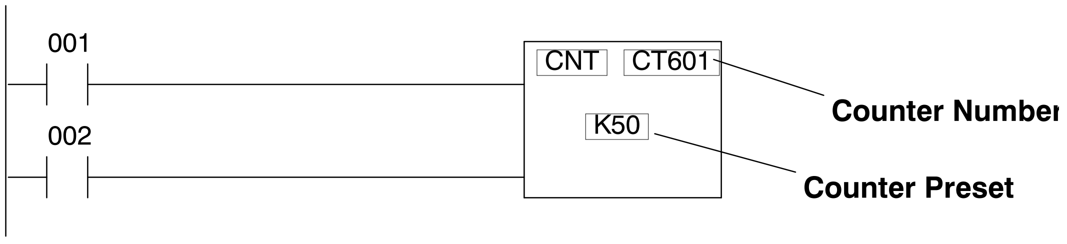
Enter the Timer preset



NOTE: With timer preset values you must use a decimal point to enter a fourth digit.

When the timer (or counter) reaches the preset value, a coil is turned on. You can use this coil as an contact in other parts of the program. See Page 2-5 for the keystrokes required to enter a timer or counter contact.

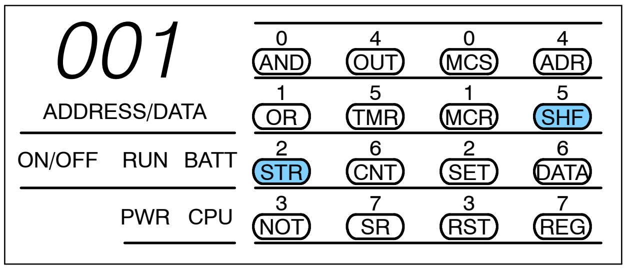
Counters are very similar to timers, but they have enable / reset legs that allow you to reset the counter. The following example shows how to enter the additional input line. Notice that you enter both contacts *before* you complete the counter.



Enter the first contact

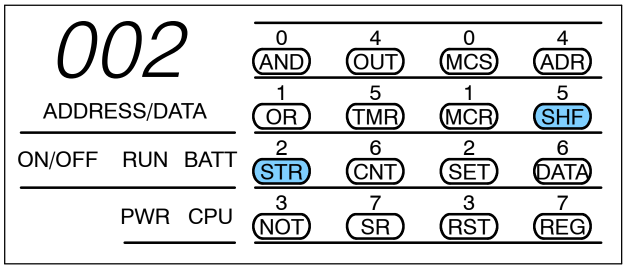
STR	SHF	1	ENT
🟦			

Data Display *before* ENT is pressed



Enter the enable / reset leg

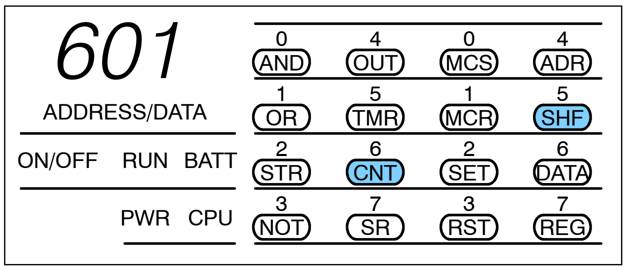
STR	SHF	2	ENT
🟦			



Enter the counter

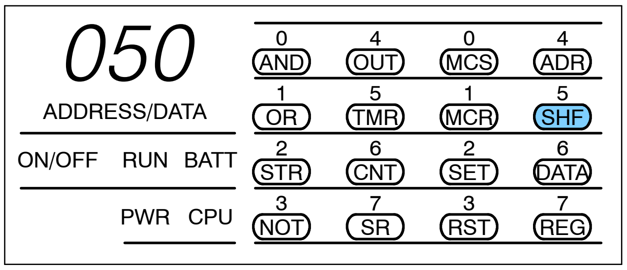
CNT	SHF	6	0	1
🟦				

ENT



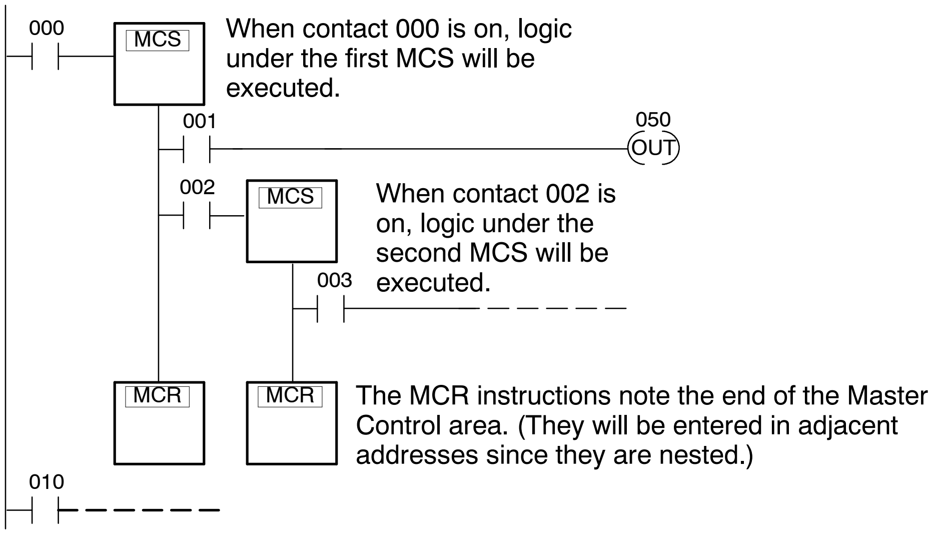
Enter the preset

SHF	5	0	ENT



Entering Master Control Relays

The Master Control Set (MCS) and Master Control Reset (MCR) instructions allow you to quickly enable (or disable) sections of the RLL program. This provides program control flexibility. (See the DL305 User Manual for more details.) The following example shows how the MCS and MCR instructions operate.

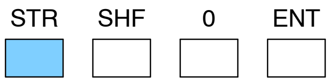


When contact 000 is on, logic under the first MCS will be executed.

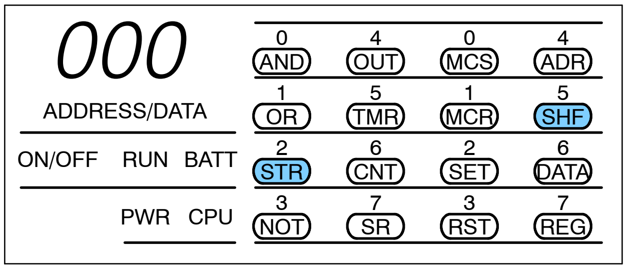
When contact 002 is on, logic under the second MCS will be executed.

The MCR instructions note the end of the Master Control area. (They will be entered in adjacent addresses since they are nested.)

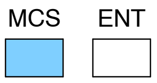
Enter the contact

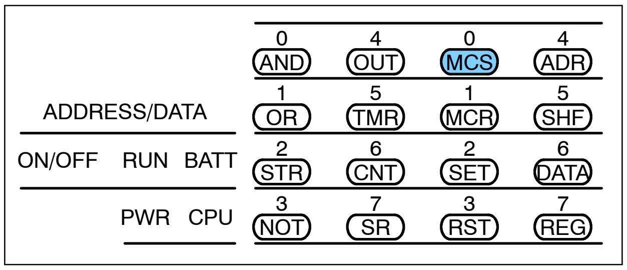


Data Display *before* ENT is pressed

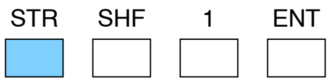


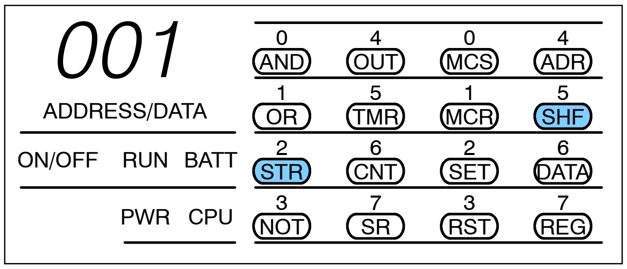
Enter the first MCS instruction



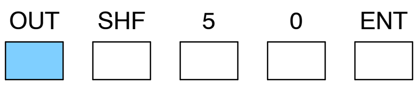


Enter the next contact

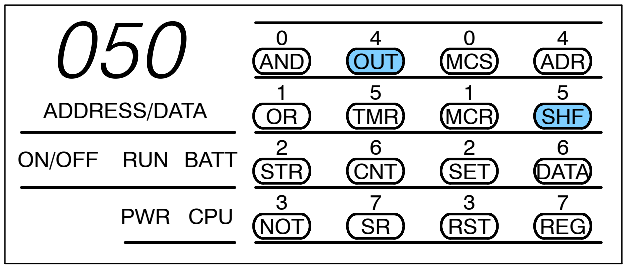




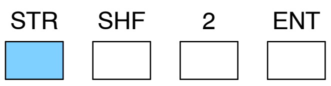
Enter the output

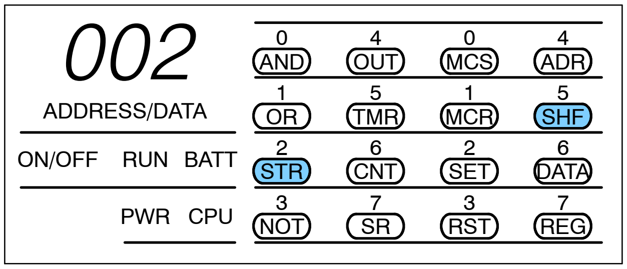


Data Display *before* ENT is pressed

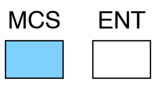


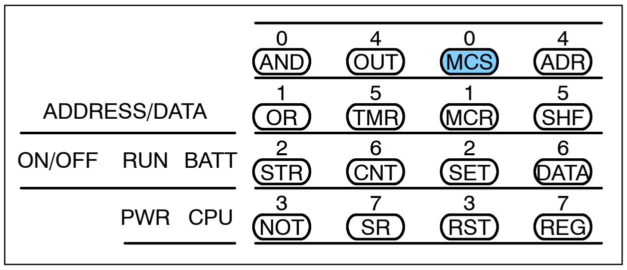
Enter the next contact



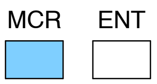


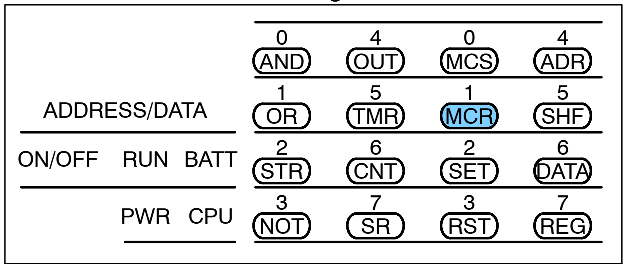
Enter the second MCS instruction



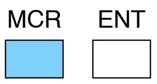


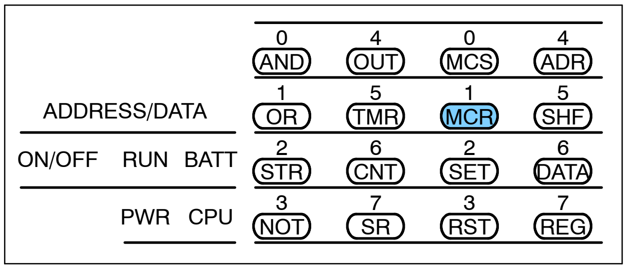
Enter the first MCR instruction





Enter the second MCR instruction





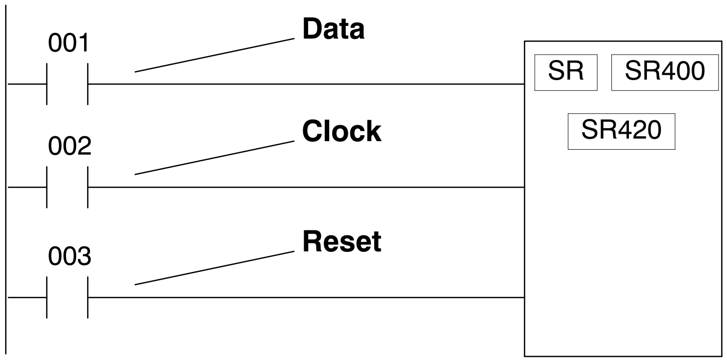
Entering Shift Registers

The DL305 CPUs allow you to use Control Relays with a Shift Register. You can have any number of Shift Registers, but there are only 128 Control Relays that can be used (400-577). Also, you cannot use these bits as Control Relays *and* Shift Register bits. (See the DL305 User Manual for more details.)

The Shift Register has three input contacts.

- Data — used to determine a value (1 or 0) that will be shifted through the register
- Clock — on each low to high transition, the value that is on the data line will be shifted into the shift register.
- Reset — if the reset contact comes on, then the Shift Register is reset and all Control Relays are reset.

The following diagram shows a brief example of how the Shift Register works.

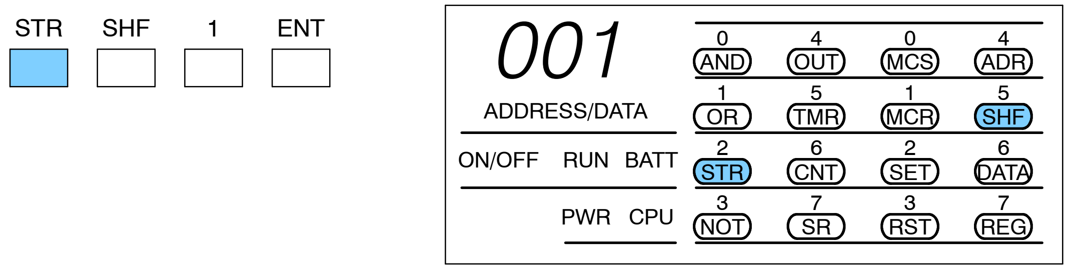


Inputs on Successive Scans			Shift Register Bits															
Data	Clock	Reset	400															417
1	1	0	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
0	1	0	⬜	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
0	1	0	⬜	⬜	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
1	1	0	🟦	⬜	⬜	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
0	1	0	⬜	🟦	⬜	⬜	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
0	0	1	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜

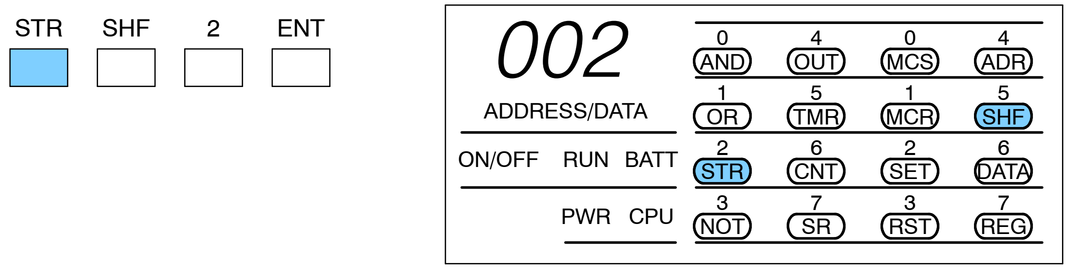
🟦 - indicates ON ⬜ - indicates OFF

You can use the following keystrokes to enter a Shift Register.

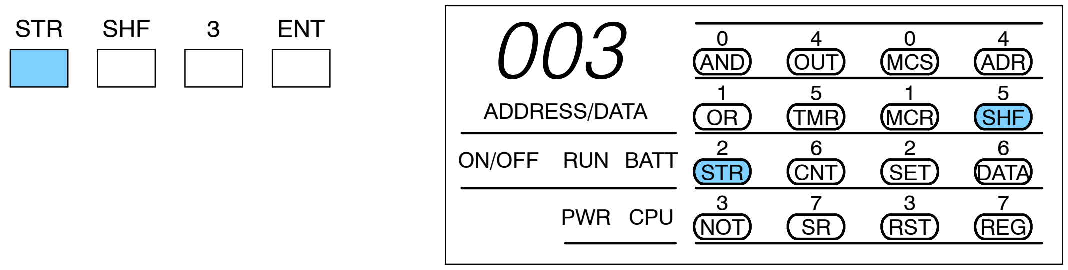
Enter the Data input **Data Display *before* ENT is pressed**



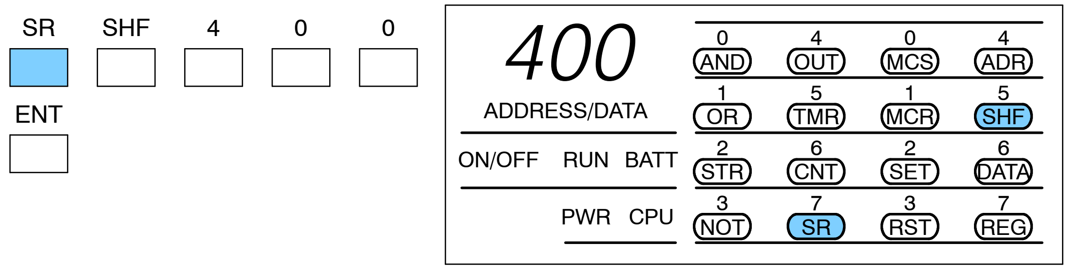
Enter the Clock input



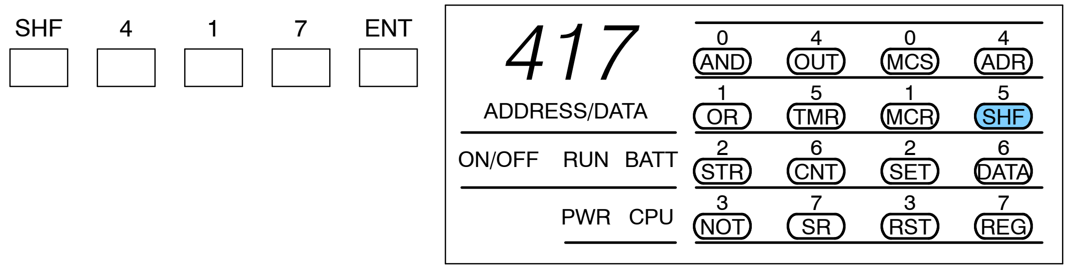
Enter the Reset input



Enter the Shift Register and starting location



Enter the end of the Shift Register



Entering Data Instructions

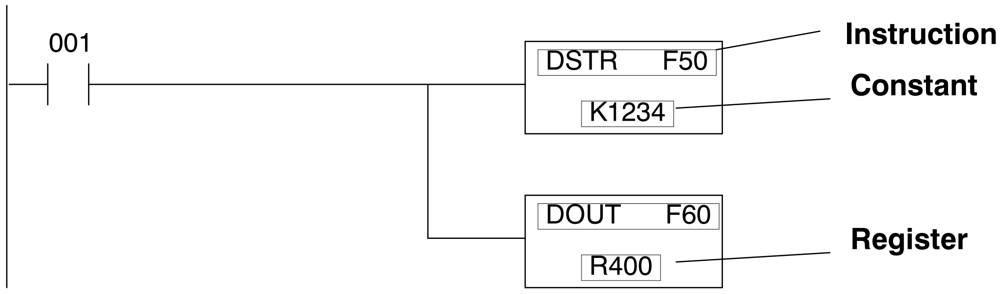
In addition to the simple RLL instructions, the DL305 CPUs also provide Data Instructions that allow you to manipulate words of data. For example, you may want to perform simple math operations on register values. The following table provides a listing of the various Data Operation Instructions. You should see the DL305 User Manual for complete details on how the instructions operate.

Category	Mnemonic	Function #	Description
Data Load Instructions	DSTR	F50	Load a 4-digit constant or 2-bytes of register data into the accumulator
	DSTR 1	F51	Load 1-byte of register data into the accumulator
	DSTR 2	F52	Load the upper 4 bits of a register into the lower 4 bits of the accumulator
	DSTR 3	F53	Load the lower 4 bits of a register into the upper 4 bits of the accumulator
	DSTR 5	F55	Load the digital value of a 16 point module (2 bytes) into the accumulator
Data Out Instructions	DOUT	F60	Write the accumulator to 2 sequential registers
	DOUT 1	F61	Write the lower byte of the accumulator to a register
	DOUT 2	F62	Write the lower 4 bits of the accumulator to the upper 4 bits of a register
	DOUT 3	F63	Write the lower 4 bits of the accumulator to the lower 4 bits of a register
	DOUT 5	F65	Write the contents of the accumulator to a 16-point output module (2 bytes)
Math Instructions	CMP	F70	Compare a 2-byte BCD reference or a 4-digit BCD constant to the accumulator
	ADD	F71	Add a 2-byte BCD reference or a 4-digit BCD constant to the accumulator
	SUBTRACT	F72	Subtract a 2-byte BCD reference or a 4-digit BCD constant from the accumulator
	MULTIPLY	F73	Multiply a 2-byte BCD reference or a 4-digit BCD constant by the value in the accumulator
	DIVIDE	F74	Divide the accumulator by a 2-byte BCD reference or a 4-digit BCD constant

Category	Mnemonic	Function #	Description
Bit Manipulation Instructions	DAND	F75	Performs a bit “AND” on a 2-byte reference or a 4-digit BCD constant and the bits in the accumulator
	DOR	F76	Performs a bit “OR” on a 2-byte reference or a 4-digit BCD constant and the bits in the accumulator
	SHIFT RIGHT	F80	Shifts the contents of the accumulator to the right a specified number of times. 1 – 15 bits can be shifted.
	SHIFT LEFT	F81	Shifts the contents of the accumulator to the left a specified number of times. 1 – 15 bits can be shifted.
Data Conversion Instructions	DECODE	F82	Decodes the first 4 bits of the accumulator into a decimal number.
	ENCODE	F83	Encodes an accumulator bit into a 4-bit code that represents the decimal number (0–15).
	INV	F84	Logically inverts the bit pattern contained in the accumulator (1 to 0, 0 to 1).
	BCD–BIN	F85	Converts the accumulator value from BCD to Binary
	BIN–BCD	F86	Converts the accumulator value from Binary to BCD
Fault Detection Instructions	FAULT	F20	Sends a 4-digit BCD number, from a 2-byte reference or a constant, to the programmer display

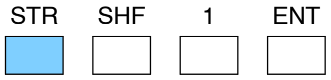
The following pages show you how to enter these functions with the Handheld Programmer.

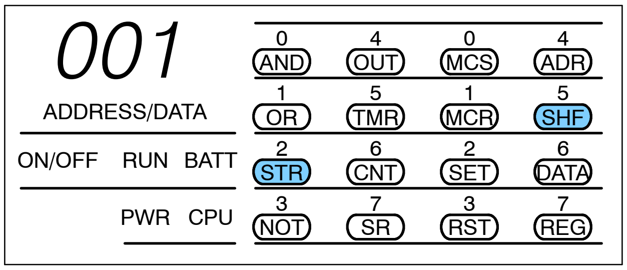
If you examine the Handheld Programmer keypad, you'll notice two keys labeled F and R. These keys are used to access the various Data Operation Instructions. We'll use the following example to show how the keys are used.



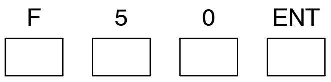
The example uses the DSTR and DOUT instructions. As you enter the instructions you will use the Function Numbers that were shown in the previous instruction table.

Enter the contact

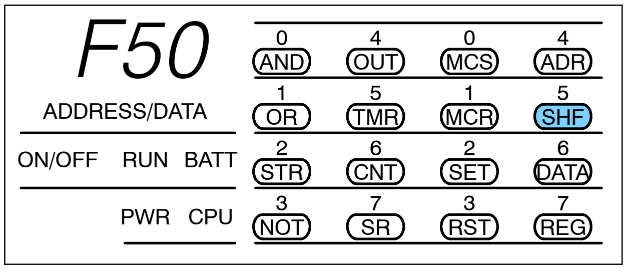




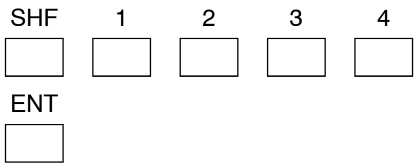
Enter the DSTR instruction

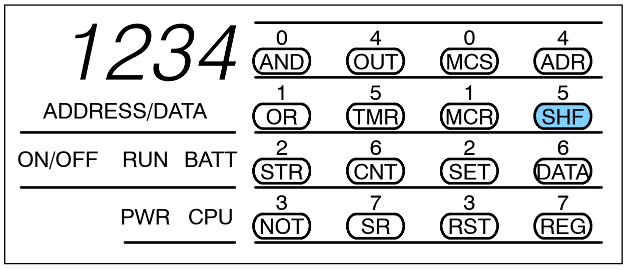


(Notice that you did not have to press the SHF key before entering the numbers.)

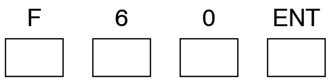


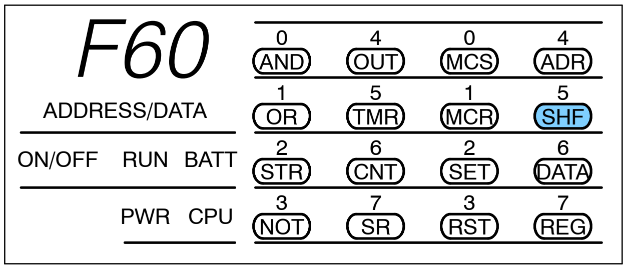
Enter the constant



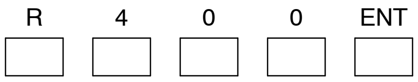


Enter the DOUT instruction

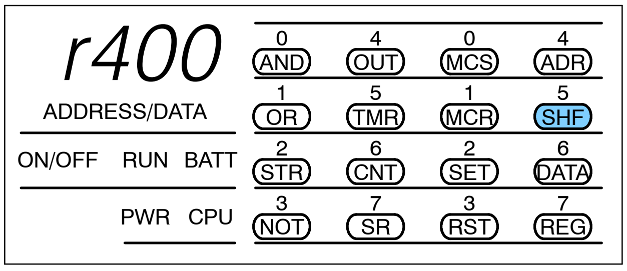




Enter the Register location



(Notice that the R key is used for entering registers and that you do not have to press the SHF key before entering the numbers.)



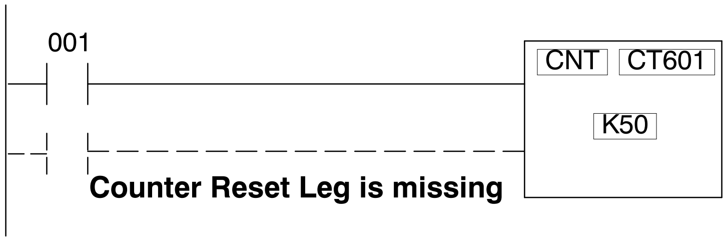
Checking for Program Errors

Automatic Error Checking

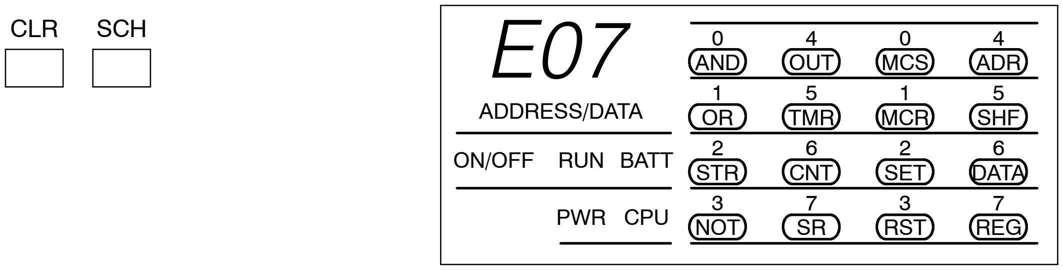
The Handheld automatically checks for some errors during program entry. Chapter 6 provides a complete listing of the error codes.

Syntax Check

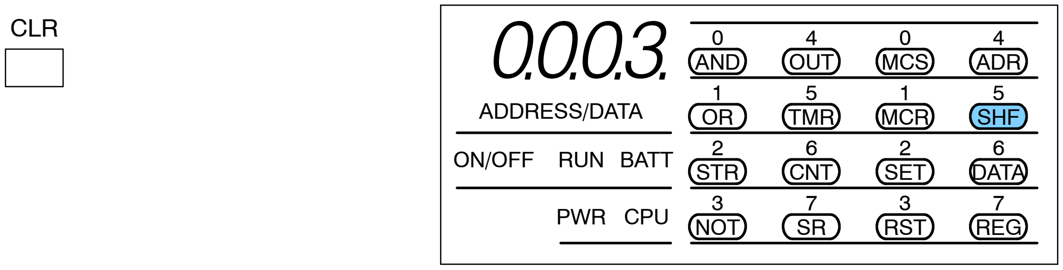
You can also execute a program syntax check which will identify programming errors. This check can be performed in either Program mode or Run mode. The following example shows how the syntax check works.



Execute the syntax check



Press CLR to display the address where the error occurred



Correct the problem and continue running the Syntax check until the E07 message no longer appears.

3

Entering RLL*PLUS* Programs

In This Chapter. . . .

RLL*PLUS* Programming Basics

RLL*PLUS* is a simplified programming method that makes program design and troubleshooting much easier. This programming method is similar to Sequential Function Chart programming and only uses a few new instructions. Before you continue, make sure you have a Handheld Programmer that supports the extra instructions needed with this style of programming (part number DL3–HPP). This chapter does not provide a complete discussion of these instructions. Instead, it just provides a quick overview of how to enter the instructions with the DL305 Handheld Programmer.

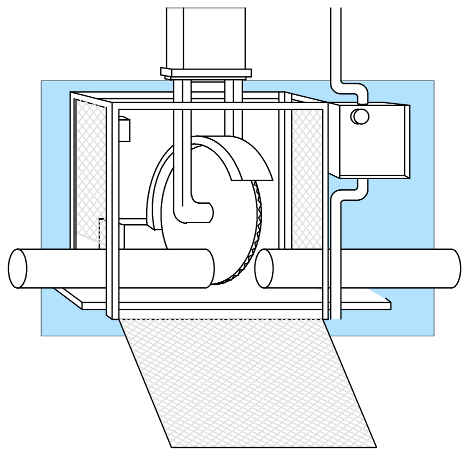
The primary benefit from this programming method is the number of logic interlocks is *significantly* reduced. This is because the individual program segments, called "stages" are completely self contained. When a program segment is active, the inputs and outputs in that stage will be examined and updated as appropriate. If the stage is not active, that portion of the program is not even scanned. How does this remove the interlocking burden? Simple, most interlocks are used because you not only have to *make* something happen, but you also have to *prevent* unwanted actions from happening. See the DL305 User Manual for details on RLL*PLUS* programming.

We'll use the following cutoff saw example to show the different instructions that are used. This example is not intended to show you how to design RLL*PLUS* programs, but is merely used to point out the instructions and the Handheld Programmer keystrokes needed to enter the instructions. If you want to know more about this style of programming, see the DL305 User Manual for complete details.

This simple cutoff saw operates in the following manner.

1. Once the operator presses a start switch, the pipe conveyor is started. (The operator can also press a Stop switch to stop the operation at any time.)
2. The pipe travels along a conveyor until it reaches a physical stop. The stop contains a limit switch that signals the system to stop the conveyor and begin the cutting operation.
3. The pipe is clamped in place.
4. The saw motor is started and the saw cuts the pipe.
5. The saw is retracted, the motor is turned off and the clamp is released.
6. If the saw is in one-cycle mode, the operator must press the start switch again. If it is not in one-cycle mode, the saw continues the operations sequence.

Cutoff Saw



The operator can stop the process at any time just by pressing a stop switch.

The following diagram shows a very simple RLL*PLUS* program that would control this operation.

This representation is what you could expect to see if you were using our ***Direct***SOFT programming software. You may notice the diagram doesn't appear very different from a normal RLL program. However, there are *significant* advantages that aren't obvious at first glance. If you want to know more, take a few extra minutes to read about this time-saving approach in the DL305 User Manual.

WARNING: The example program shown is not suitable for actual applications. There are many safety aspects that have not been considered, which could result in a risk of personal injury or damage to equipment.

The instructions used to create this program are very similar to the ones used in normal RLL programs. For example, you still enter contacts and output coils the same way. There are a couple of new instructions and these instructions have keys on the Handheld Programmer. You'll notice the Initial Stage (ISG) instruction begins the program. If you examine the Handheld you'll notice a key for the ISG instruction.

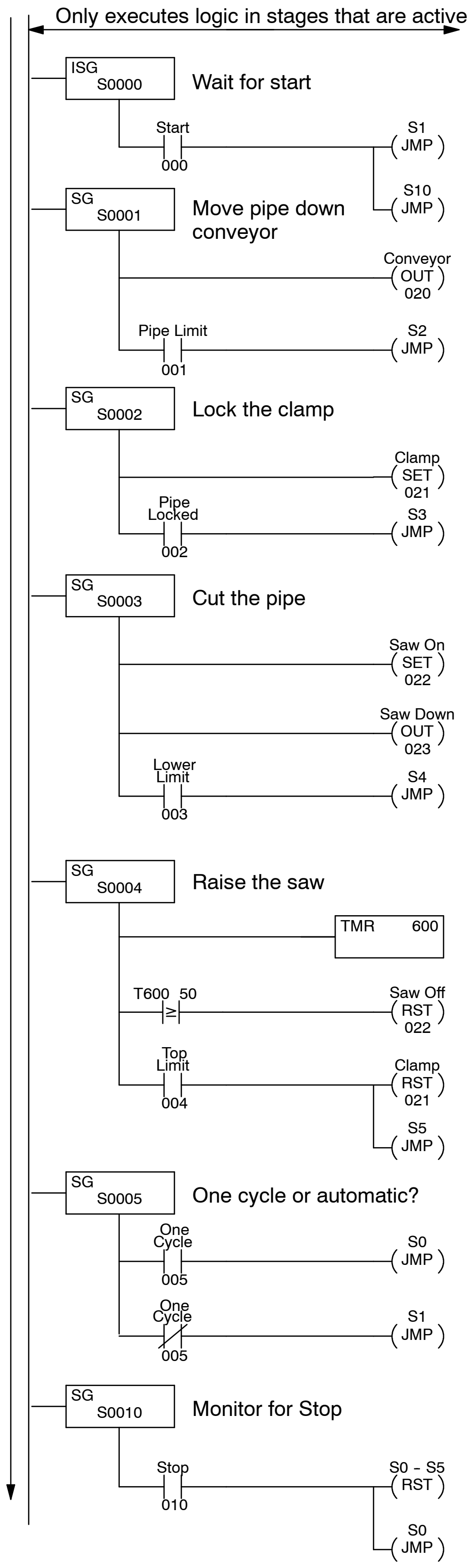
Here's a list of the primary instructions you'll use with RLL*PLUS* programs.

- Initial Stage (ISG)
- Stage (SG)
- Jump to Stage (JMP)

There are also a few other instructions that are used differently (and entered differently) in RLL*PLUS* programs.

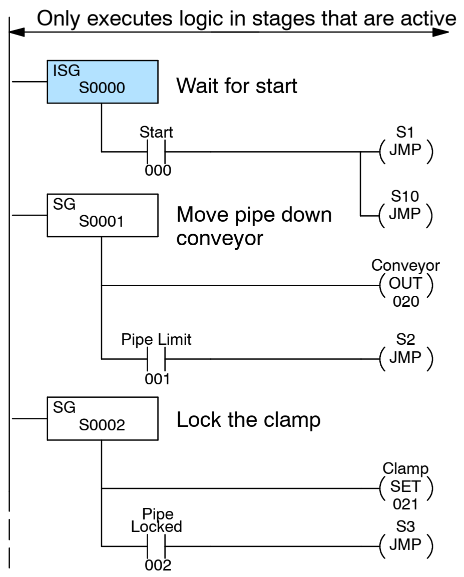
- Timers
- Counters
- Shift Registers

The following pages show you how to enter these instructions.

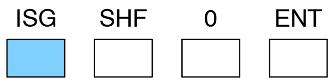


Entering an Initial Stage

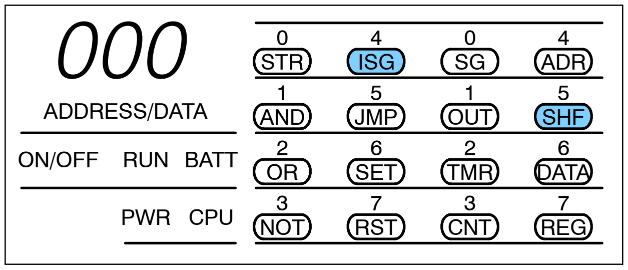
The Initial Stage identifies a starting point in the program. When the CPU enters Run mode, this is where the program execution will begin. The following keystrokes are used to enter an initial stage. (Remember, the RLL*PLUS* Handheld programmer keypad layout and display is different from the RLL Handheld programmer.)



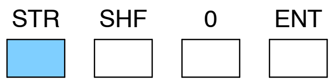
Enter the initial stage

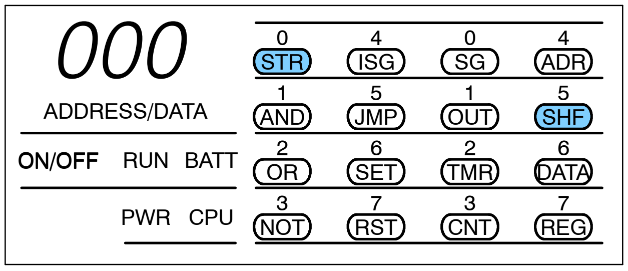


Data display *before* ENT is pressed



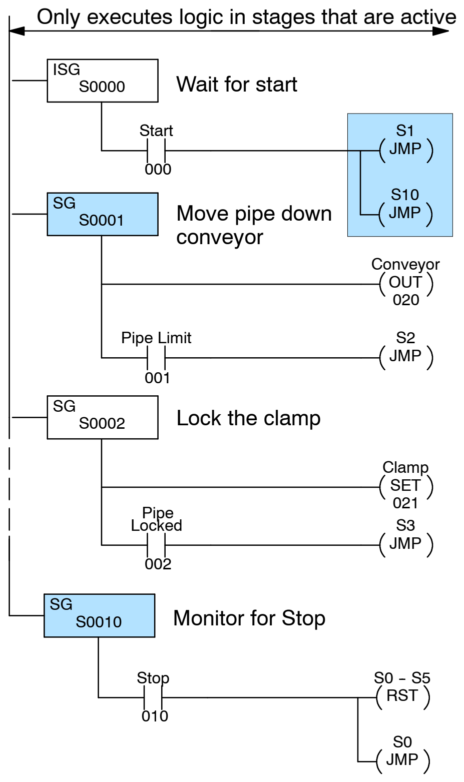
Enter the contact



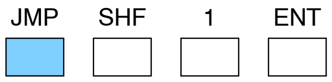


Entering Jump Instructions

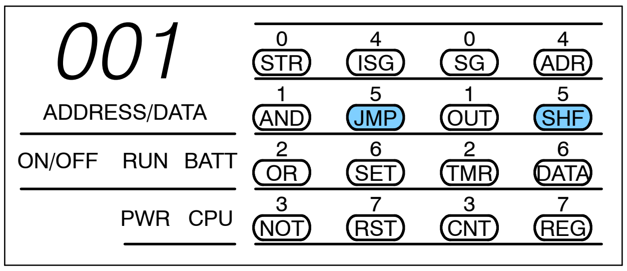
The Jump (JMP) instruction provides a way to transition to multiple points in the program. If you look at the example program you'll notice the program branches to two locations after the operator presses the start switch. The Jump instruction provides this transition. Since the program is jumping to two locations, you'll use two Jump instructions.



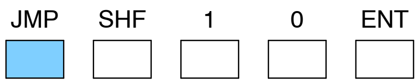
Enter the first jump

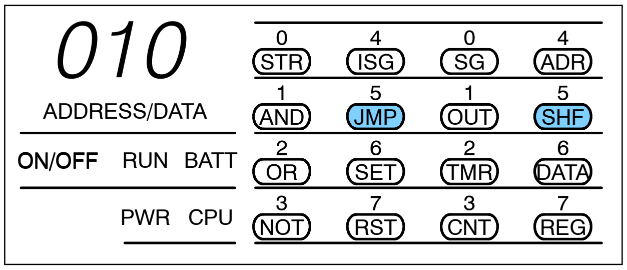


Data display *before* ENT is pressed



Enter the second jump

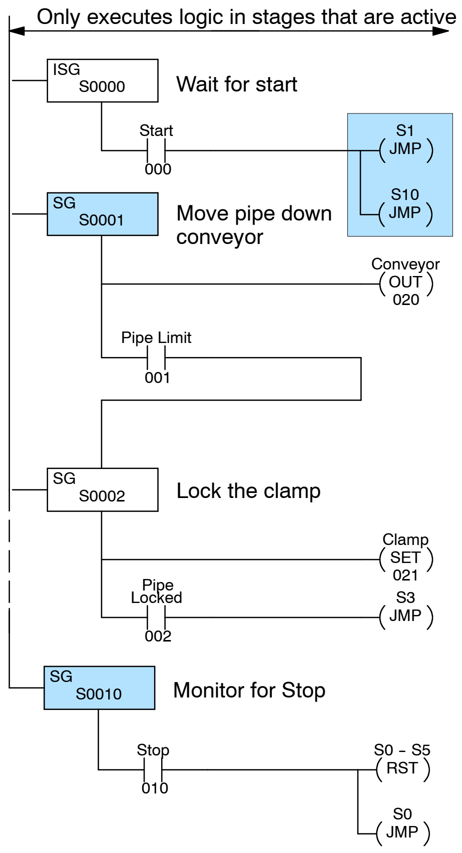




Entering Stage Instructions

The Stage (SG) instruction identifies the starting point of a program segment. Unlike an initial stage, a regular stage does not automatically activate when the CPU enters Run mode. There are three ways to activate a stage.

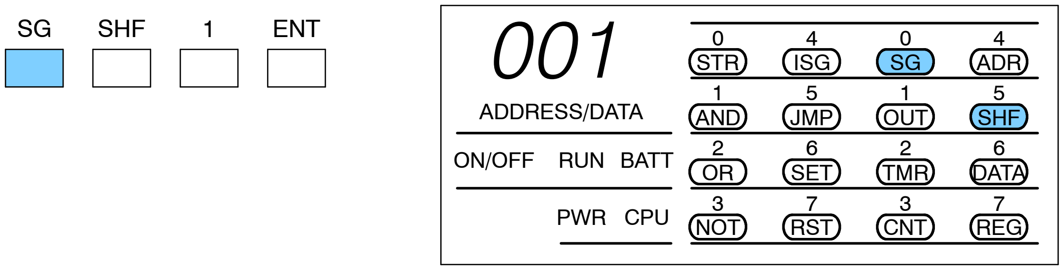
- Jump Transition — if you jump from a stage to another stage, then the destination stage is automatically activated. (Remember the jump example.)
- Power Flow Transition — power can flow through the stages which will activate the next stage.
- SET — just as you can use a SET instruction to turn on an output, you can use a SET to turn on a stage. With this method, the stage will stay on until it is reset (RST) or, until the logic within that stage causes a jump or power flow transition.



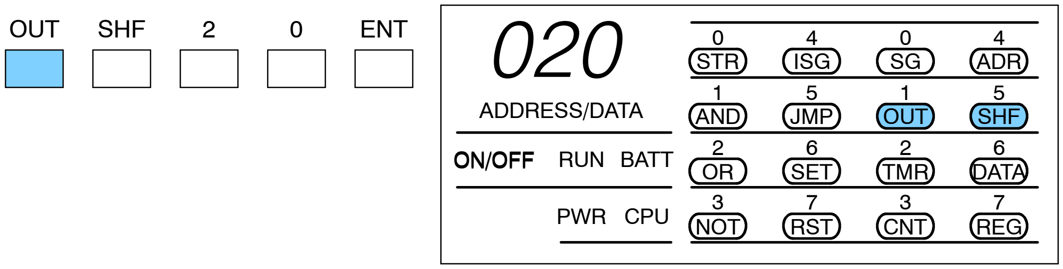
The keystrokes on the following page show how to enter stages 1 and 2. Notice we have changed Stage 2 slightly. This is an example of how a power flow transition looks. A JMP instruction is not required in this example to move from Stage 1 to Stage 2. How do you know when to use a JMP instruction? Simple, if you're moving from one stage to a single stage, you may use a power flow transition. If you're moving from one stage to multiple stages, you must use the JMP instruction.

Enter Stage 1

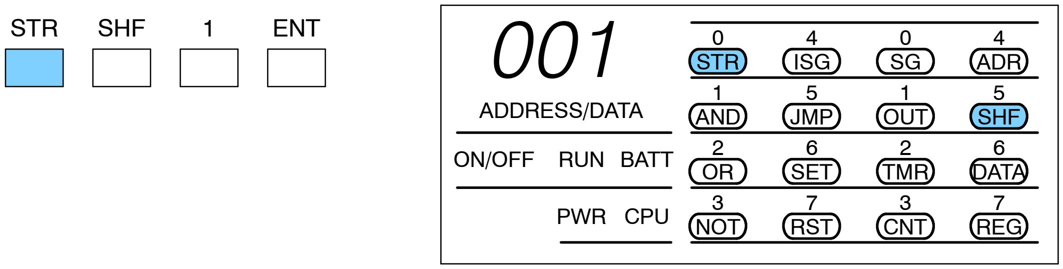
Data display *before* ENT is pressed



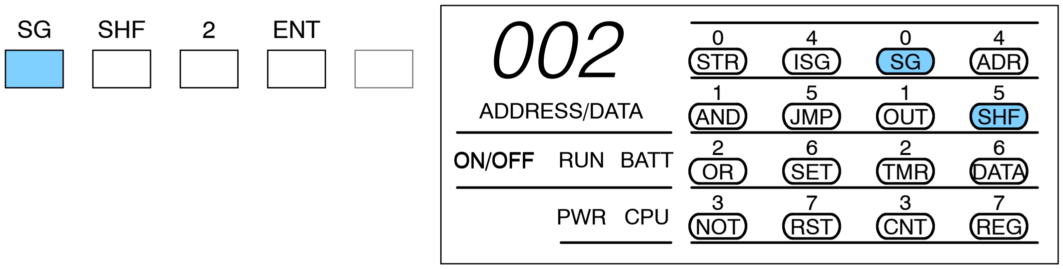
Enter the output for the conveyor



Enter the Pipe Limit contact



Enter Stage 2

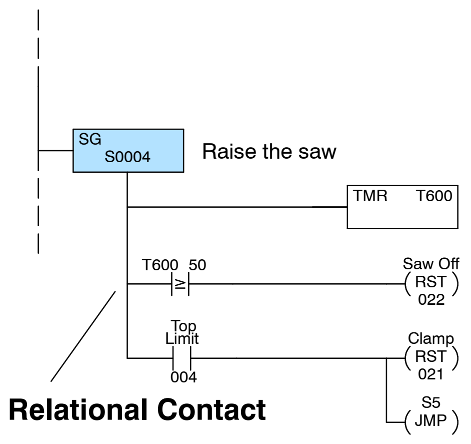


Entering Timers

Timers work differently in RLL*PLUS* programs because they do not require the entry of a preset value. Once the timer input contact has started the timer, the timer continues until the input contact is turned off.

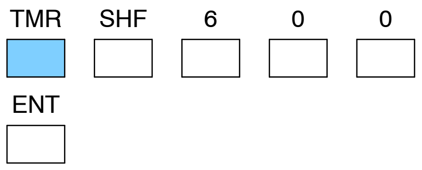
You may recall RLL timers have a timer contact associated with them. When the timer reaches the preset, it turns on the contact, which can then be used as an input contact for other parts of the program.

As well as not having a preset value, RLL*PLUS* timers also do not have a timer contact. Instead of using the timer contact, relational contacts are used to examine the timer value.

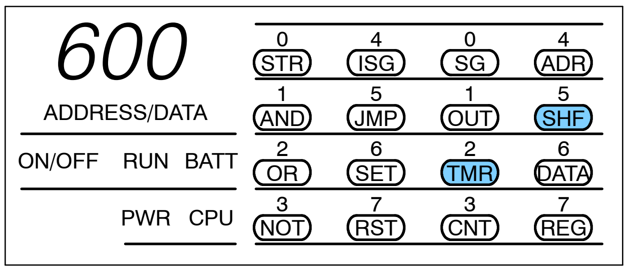


The following example shows how a time delay was added when the saw was being raised. We wanted to keep the saw motor running for 5 seconds so the saw could clear the pipe before being turned off. (Note, the keystrokes only show how to enter the timer and the relational contact, not the whole stage.)

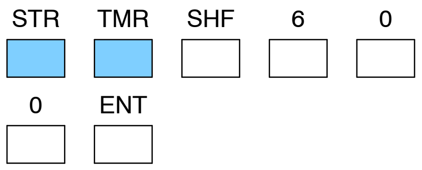
Enter the Timer

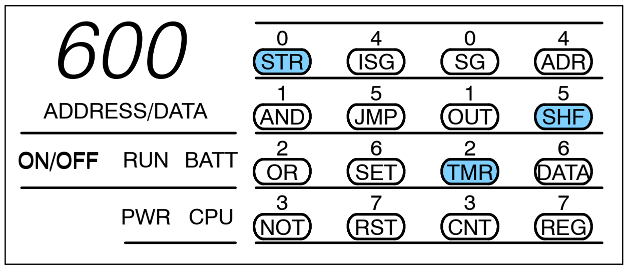


Data display *before* ENT is pressed

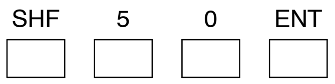


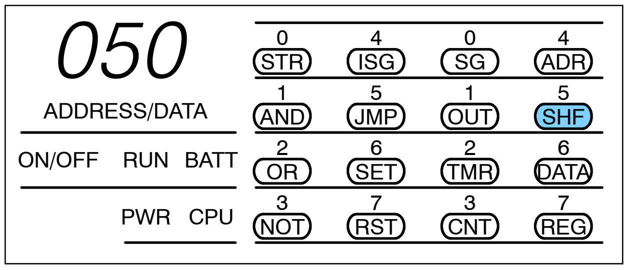
Enter the comparative timer contact





Enter the compare value



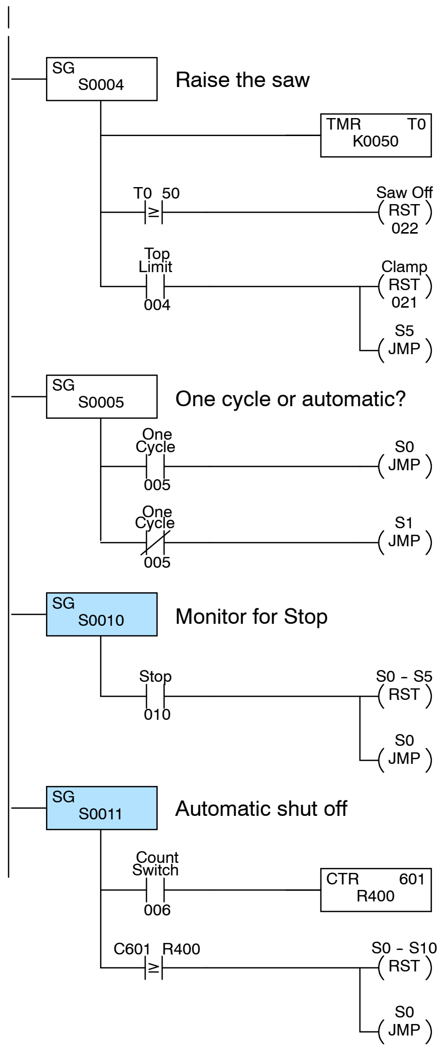


Entering Counters

Counters also work differently in RLL*PLUS* programs because they do not require either a reset input or a preset value. Once the stage is active, the counter input contact controls the value of the counter. Each time the counter input contact has an off to on transition, the counter will increment one count. When the stage becomes inactive, the counter is disabled and reset to 0. (Remember, the CPU does not even scan the logic contained in an inactive stage.)

You may recall RLL counters have a counter contact associated with them. When the counter reaches the preset, it turns on the contact, which can then be used as an input contact for other parts of the program.

As well as not having the reset input and preset value, the RLL*PLUS* counters also do not have a counter contact. Instead of using the counter contact, relational contacts are used to examine the counter value.

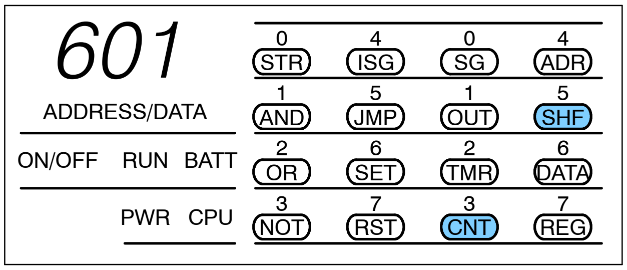


Let's say we wanted to use an operator interface to tell the machine how many pipes to cut. (We'll assume the number is loaded into a register, R400.) Once the correct number of pipes have been cut, the saw should automatically stop. The following example shows how you would add an automatic shutoff stage to the cutoff saw example by using a counter and a relational contact. (Note, the keystrokes only show how to enter the counter and the relational contact, not the whole program.)

Enter the Counter

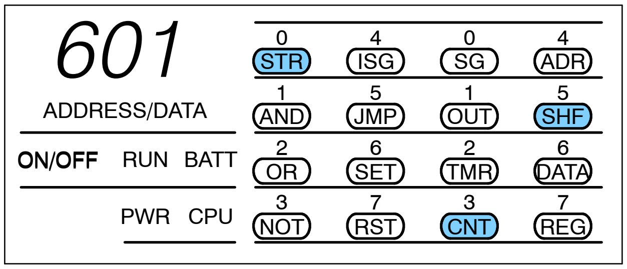
CNT SHF 6 0 1
ENT

Data display *before* ENT is pressed



Enter the relational contact

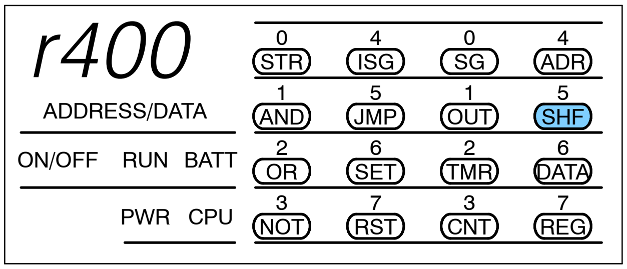
STR CNT SHF 6 0
1 ENT



Enter the relational contact comparison register

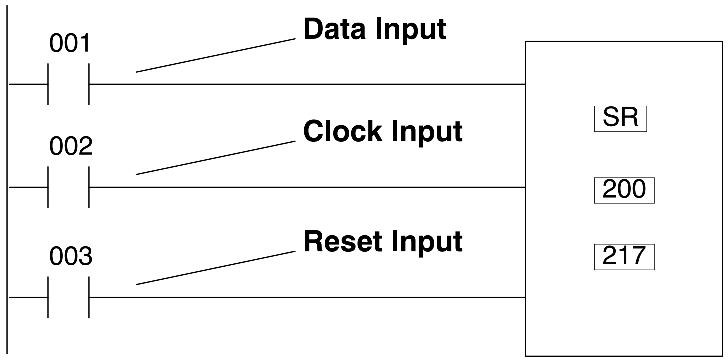
R 4 0 0 ENT

(Notice you did not have to press the SHF key before entering the numbers.)



Entering Shift Registers

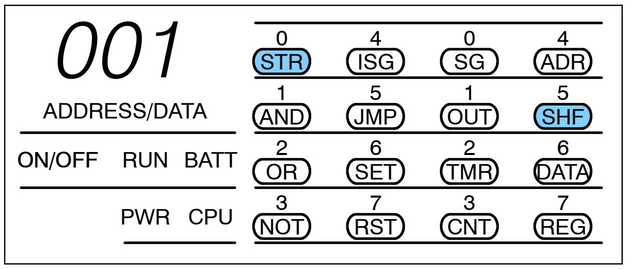
Shift Registers operate the same in RLL*PLUS* programs as they do in RLL programs. However, the keystrokes required to enter a Shift Register are different because the SR key is not on the RLL*PLUS* Handheld Programmer. Also, you do not have a separate range of bits available for use as shift register bits. Instead you have to use the control relays. The following page shows the keystrokes used with this type of Handheld.



Enter the Data input

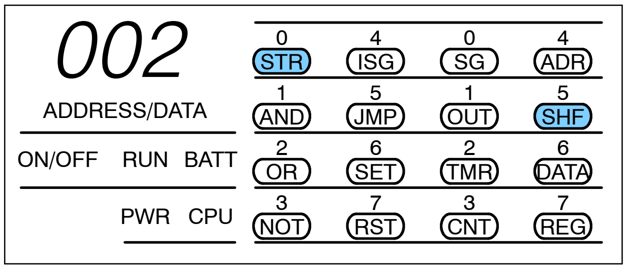
STR SHF 1 ENT

Data Display *before* ENT is pressed



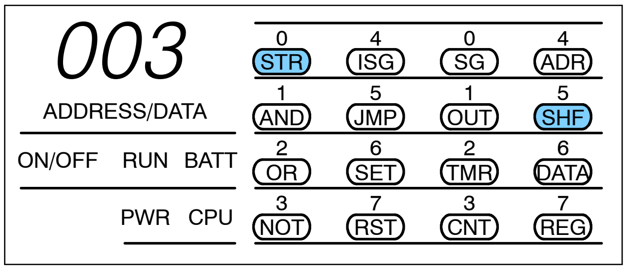
Enter the Clock input

STR SHF 2 ENT



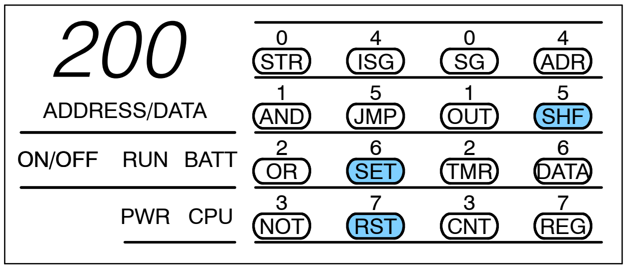
Enter the Reset input

STR SHF 3 ENT



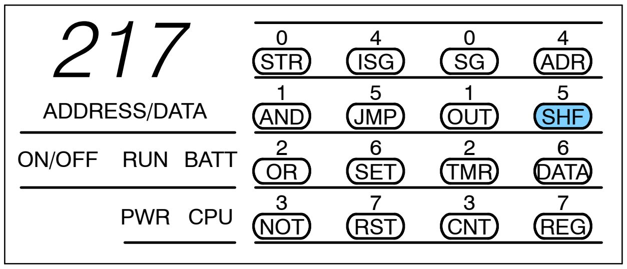
Enter the Shift Register and starting location

SET RST SHF 2 0 0 ENT



Enter the end of the Shift Register

SHF 2 1 7 ENT

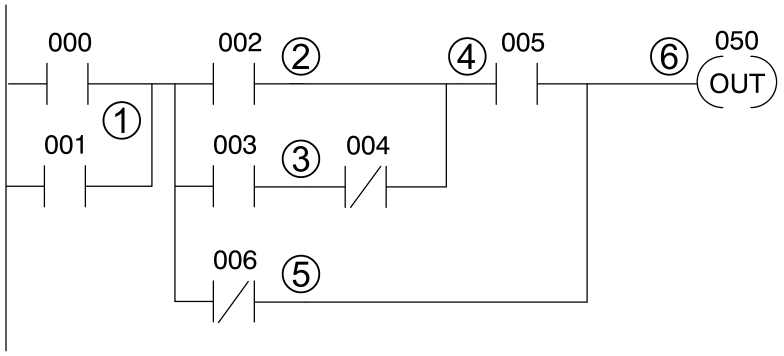


Changing Programs

In This Chapter. . . .

Displaying a Program

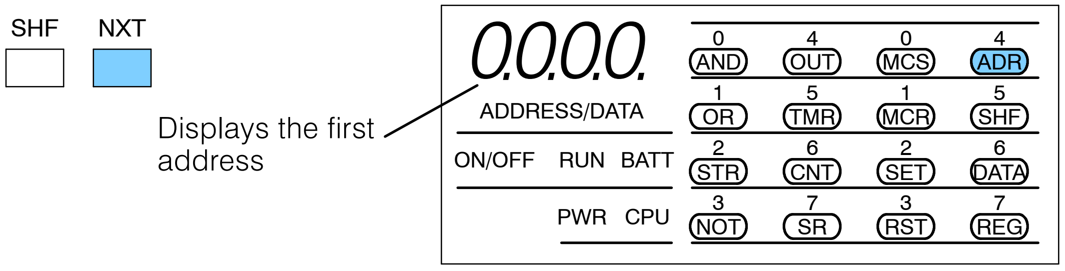
Since the Handheld displays the mnemonic instructions, you can step through the individual program instructions. If the CPU is in the RUN mode, the status of the instruction is also displayed in the status display area.



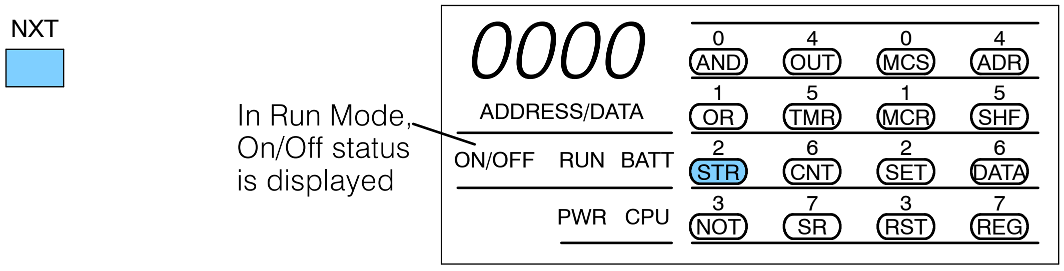
Mnemonic Listing and Addresses

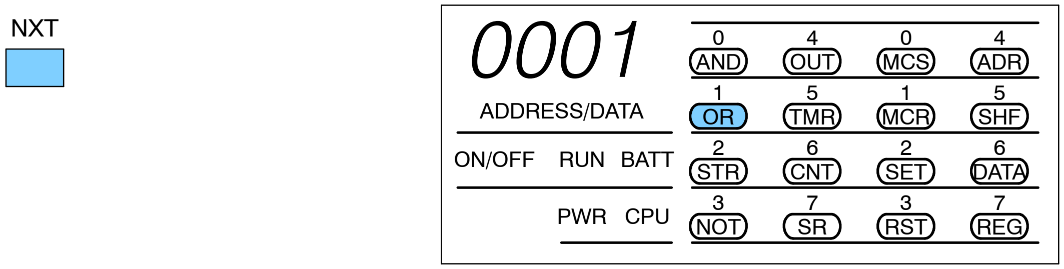
ADDRESS	INSTRUCTION	DESCRIPTION
0000	STR 000	Starts branch 1 with 000
0001	OR 001	Joins 001 in parallel with 000
0002	STR 002	Starts branch 2 with 002
0003	STR 003	Starts branch 3 with 003
0004	ANDN 004	Joins 004 (NOT) with 003
0005	ORSTR	Joins branches 2 and 3
0006	AND 005	Starts branch 4 with 005
0007	ORN 006	Joins 006 (NOT) in parallel with 005
0008	ANDSTR	Joins branches 4 and 5 with 1–3
0009	OUT 050	Stores the output and finishes the network
0010	END	Ends the program

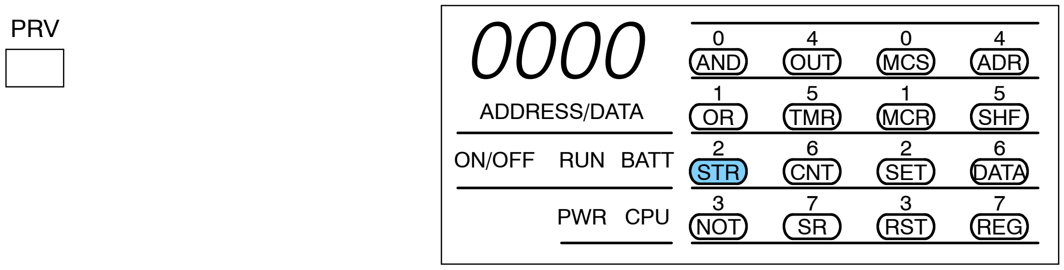
Press SHF and NXT to display the beginning of the program



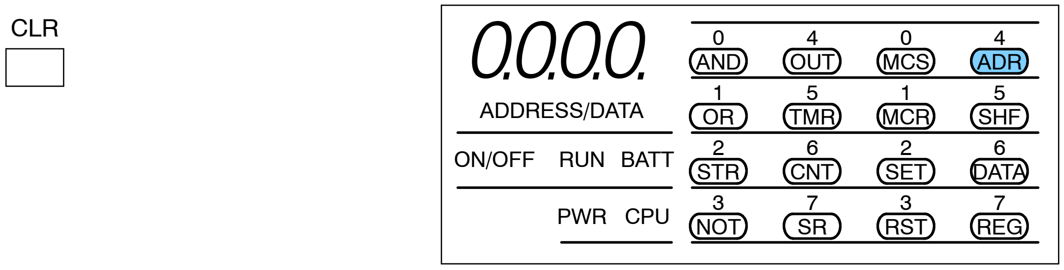
Use PRV or NXT to scroll through the program





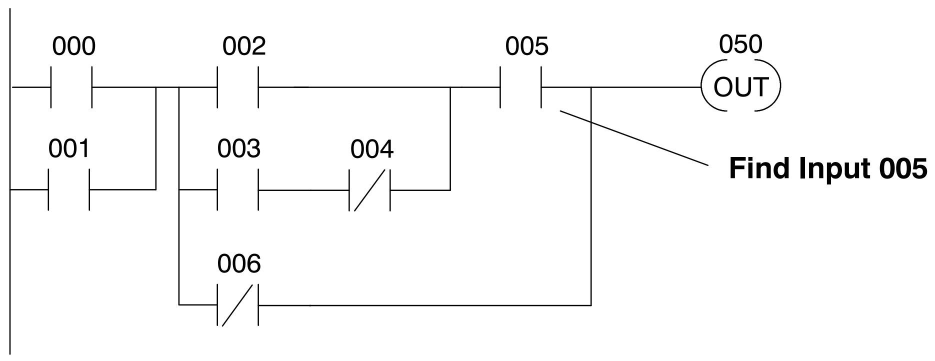


Press CLR or NXT to toggle between the address and instruction display



Finding a Specific Instruction

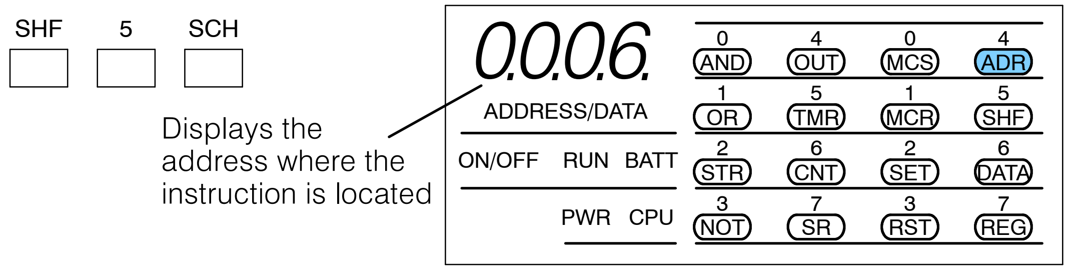
If you do not want to scroll through the program, you can use the Search feature to automatically search for an instruction. The following example shows the instructions, addresses, and corresponding Handheld displays for a small program.



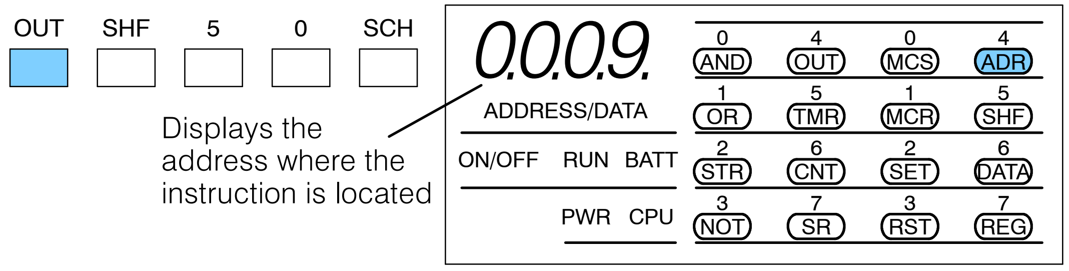
Mnemonic Listing and Addresses

ADDRESS	INSTRUCTION	DESCRIPTION
0000	STR 000	Starts branch 1 with 000
0001	OR 001	Joins 001 in parallel with 000
—	—	—
0006	**AND 005**	**Starts branch 4 with 005**
—	—	—
0010	END	Ends the program

Search for the instruction reference

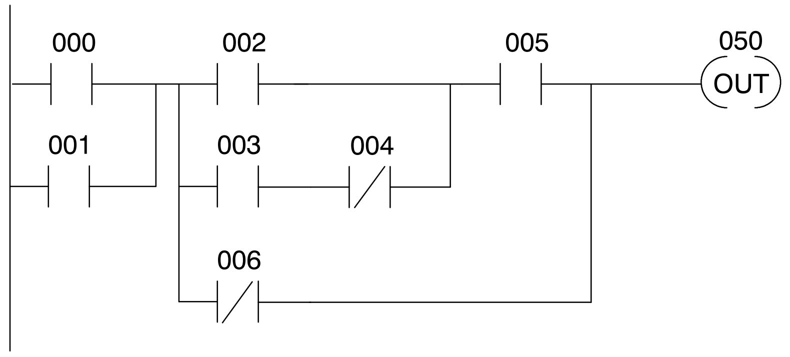


You can also specify how the reference is used
(All outputs require the additional key to indicate how the point is used.)



Finding a Specific Address

You can also search for a specific address. The following example shows the instructions, addresses, and corresponding Handheld displays for a small program.

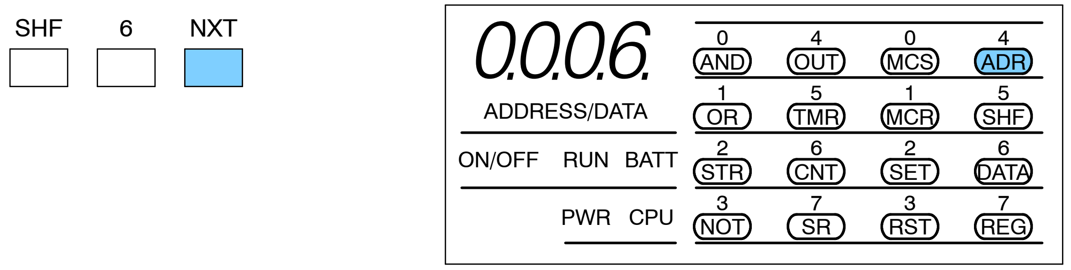


Mnemonic Listing and Addresses

ADDRESS	INSTRUCTION	DESCRIPTION
0000	STR 000	Starts branch 1 with 000
0001	OR 001	Joins 001 in parallel with 000
—	—	—
—	—	—
0006	**AND 005**	**Starts branch 4 with 005**
—	—	—
—	—	—
0010	END	Ends the program

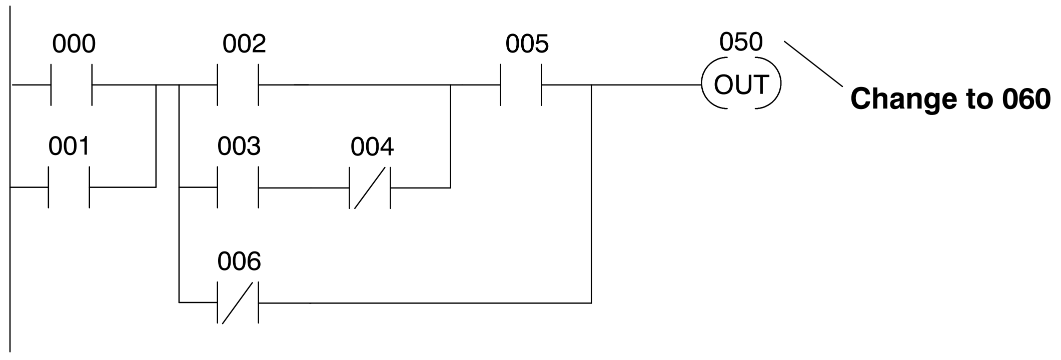
Find

Search for the address



Changing an Instruction

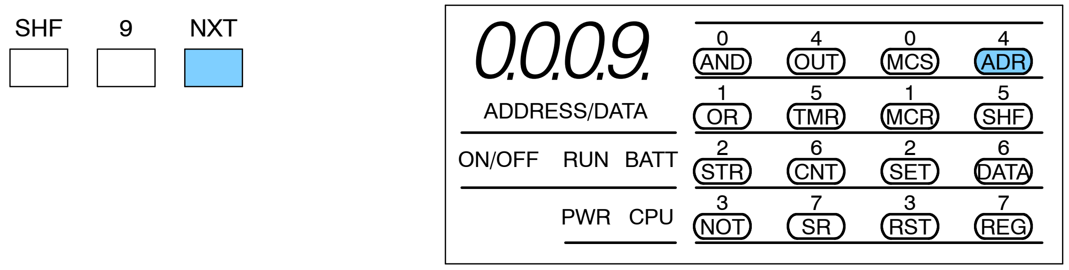
Sometimes you need to change an instruction. For example, you may want to use a different input or output point than the one originally entered into the program. The following example shows the instructions, addresses, and corresponding Handheld displays for a small program.



Mnemonic Listing and Addresses

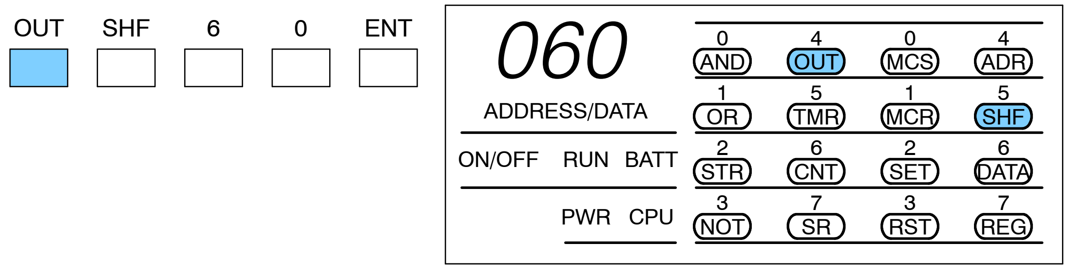
ADDRESS	INSTRUCTION	DESCRIPTION
0000	STR 000	Starts branch 1 with 000
0001	OR 001	Joins 001 in parallel with 000
—	—	—
0006	AND 005	Starts branch 4 with 005
—	—	—
0009	**OUT 060**	**Stores the output and finishes the network**
0010	END	Ends the program

Search for the address



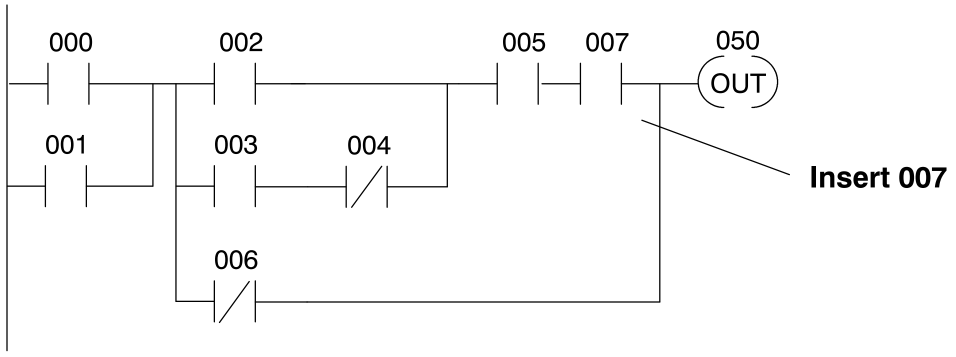
Change the instruction

(Display before ENT is pressed)



Inserting an Instruction

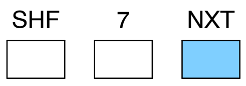
Use the INSERT feature to add an instruction to the program. Insert adds an instruction *before* the instruction being displayed, so make sure you are at the correct program address. Once you've inserted the new instruction, the remaining addresses increment. The following example shows the instructions, addresses, and corresponding Handheld displays for a small program.

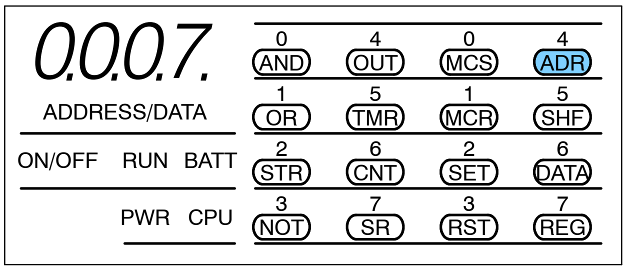


Mnemonic Listing and Addresses

ADDRESS	INSTRUCTION	DESCRIPTION
0000	STR 000	Starts branch 1 with 000
—	—	—
0006	AND 005	Starts branch 4 with 005
	Insert AND 007	**Adds 007 in series with 005**
0007	ORN 006	Joins 006 (NOT) in parallel with 005
—	—	—
0010	END	Ends the program

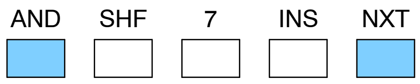
Search for the address

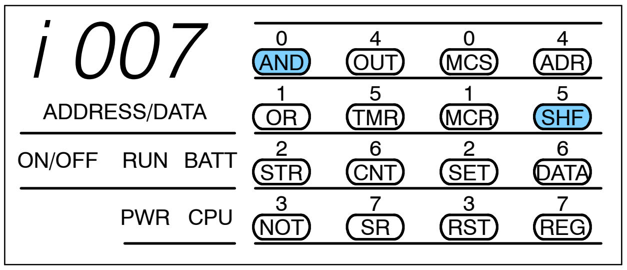




Insert the new instruction

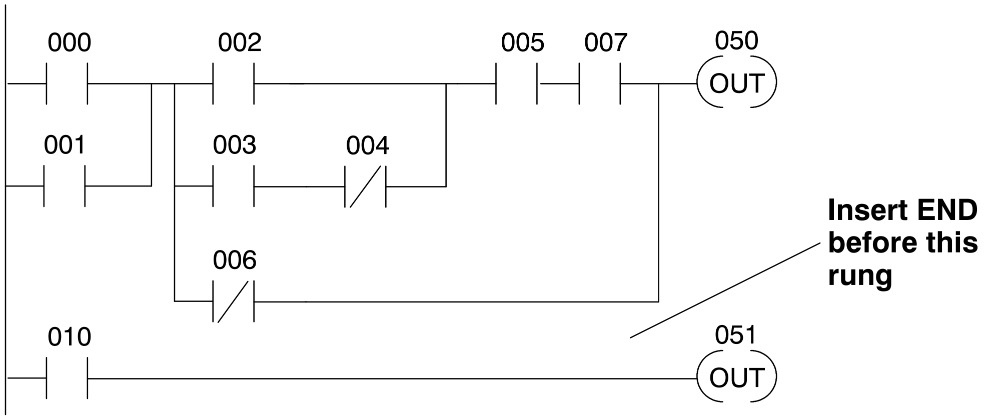
(Display before NXT is pressed)





Inserting an END Statement

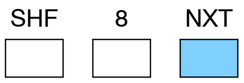
There may be times when you need to insert an END statement (*before* an address) in the program. This is commonly done when you only want to check a portion of the program during machine startup or troubleshooting. You use the INSERT feature, but since the Handheld does not have an END key, special keystrokes are required.

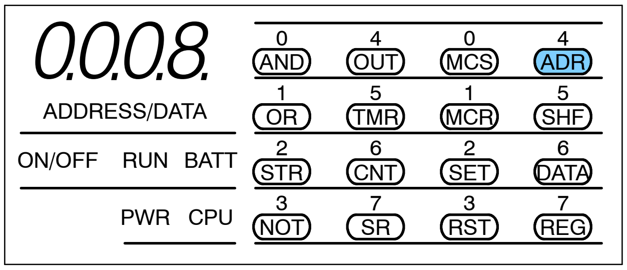


Mnemonic Listing and Addresses

ADDRESS	INSTRUCTION	DESCRIPTION
0000	STR 000	Starts branch 1 with 000
—	—	—
0006	AND 005	Starts branch 4 with 005
	Insert END	**Ends the program**
0008	ORN 006	Joins 006 (NOT) in parallel with 005
—	—	—
0013	END	Ends the program

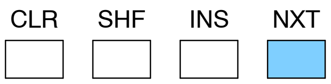
Search for the address

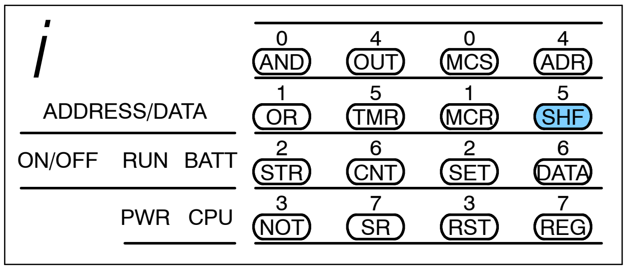




Insert the END statement

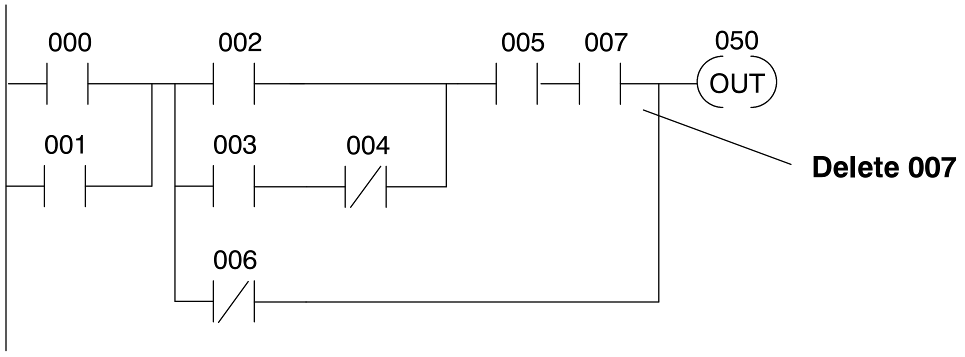
(Display before NXT is pressed)





Deleting an Instruction

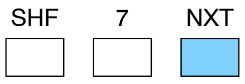
Use the DELETE feature to remove an instruction from the program. Delete removes the instruction being displayed, so make sure you are at the correct program address. Once you've deleted the instruction, the remaining addresses decrement. The following example shows the instructions, addresses, and corresponding Handheld displays for a small program.

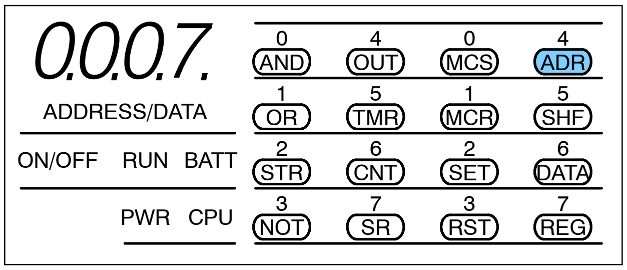


Mnemonic Listing and Addresses

	ADDRESS	INSTRUCTION	DESCRIPTION
	0000	STR 000	Starts branch 1 with 000
	—	—	—
	0006	AND 005	Starts branch 4 with 005
Delete	**0007**	**AND 007**	**Adds X7 in series with X5**
	0008	ORN 006	Joins 006 (NOT) in parallel with 005
	—	—	—
	0011	END	Ends the program

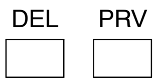
Search for the address

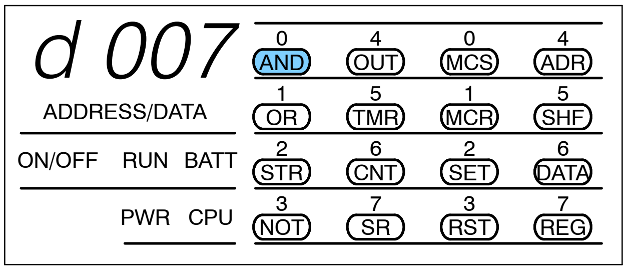




Delete the instruction

(Display before PRV is pressed)





Protecting and Storing Programs

In This Chapter. . . .

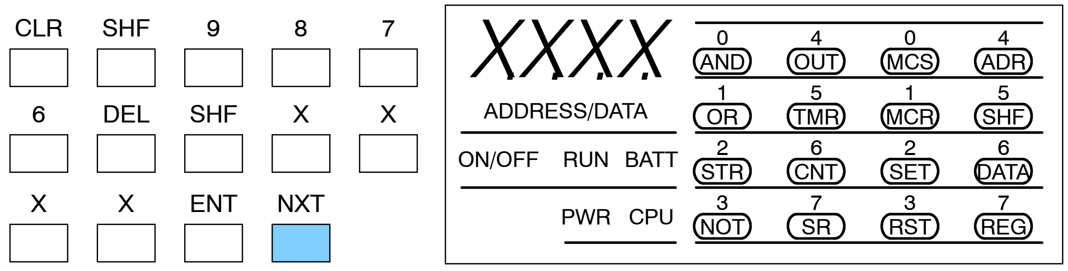
Password Protection

The DL305 CPUs provide an extra measure of protection by allowing you to enter a password that prevents unauthorized machine operations. The password must be a four-character numeric (0–9) code. Once you've entered a password, you can remove it by entering all zeros (0000). This is the default from the factory.

The password is stored in the CPU, not in the Handheld Programmer. If the battery backup is lost and the power is cycled, you could lose the password (and the program too.)

You can enter a password in either the Run or Program mode. Use the following keystrokes to use the password features.

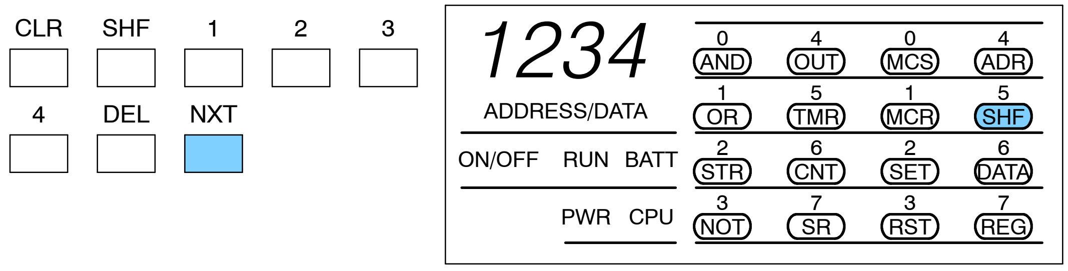
Use these keystrokes to enter a password



(X represents the password. The display depends on the password entered.)

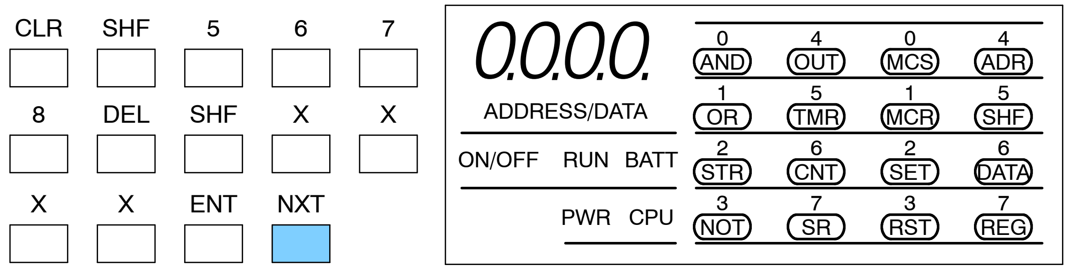
Once you've entered the password, you can enable it by using the following keystrokes.

Use these keystrokes to enable the password



Once password protection has been enabled, you must enter the password before you can make any changes to the program. You can still view the program and use the Handheld to monitor machine operations, but you cannot make changes.

Use these keystrokes to access a password protected CPU



(X represents the password. The display depends on the password entered.)

Storing Programs on Cassette Tapes

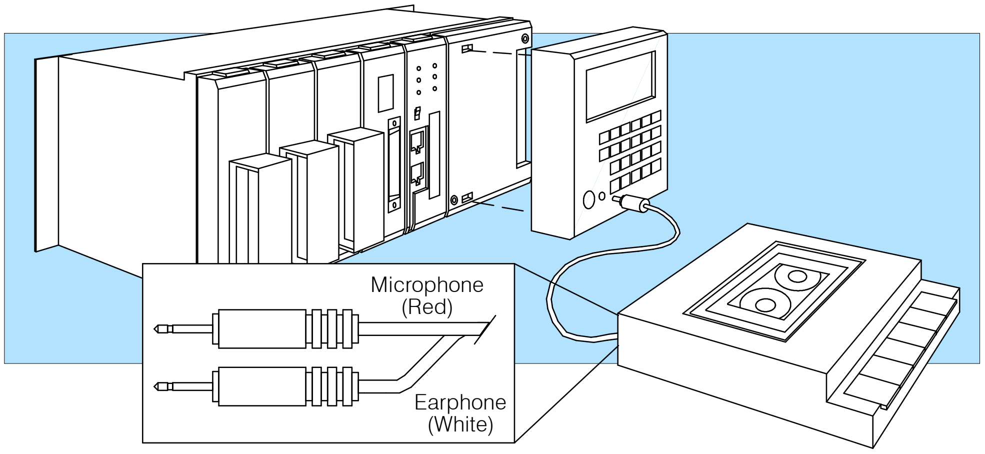
Cassette Characteristics

Although the best way to store programs is to use ***Direct***SOFT and computer diskettes, you can also copy the programs from the CPU to cassette tapes. Both forms of media allow you store several programs, but the diskette method is much easier to use and it's more reliable due to the differences in various cassette recorders.

When you select a recorder, choose one designed for use with Personal Computers (PCs). These types of recorders are much more suitable than those used for normal audio recordings. However, most audio recorders will record or play the digital information accurately if they have both volume and tone controls.

Connecting the Cassette Recorder

The cassette recorder cable, included with the Handheld Programmer, connects to the small jack on the front of the Handheld Programmer.



NOTE: Connect the cable to the tape recorder microphone jack when you are writing a program to the cassette. Connect the cable to the earphone jack when you are reading a program from the cassette.

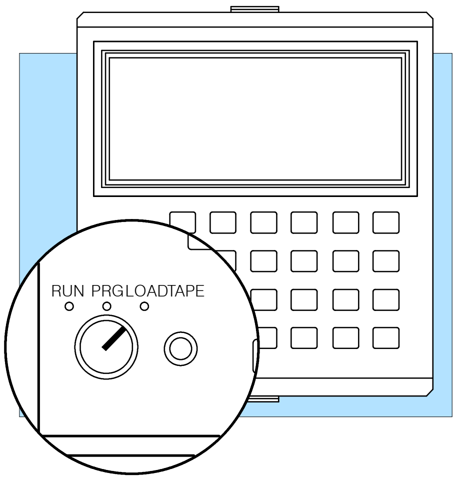
Program Names on Cassettes

Since it is very easy to store multiple programs on a single cassette it is very important idea to name each program. You can use a four-digit number to name the various programs. You do not *have* to use a name, but it's a good idea.

Writing a Program to the Cassette

If you examine the front of the Handheld Programmer you will notice one of the keyswitch positions is labeled “LOAD”. You must set the keyswitch to this position before you can save a program to a cassette.

It generally takes about 75 seconds to copy a program from the CPU to a cassette tape.



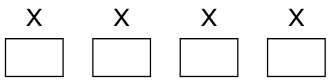
NOTE: Remember tape programs are stored sequentially. It is very easy to overwrite existing programs if you do not position the tape correctly before beginning this procedure. Use the tape counter on the recorder to keep track of program locations.

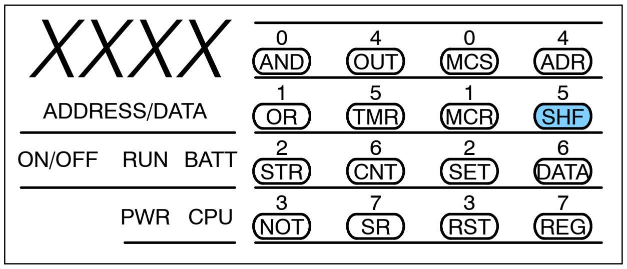
If you want to make your life easier, it’s a good idea to make a few notes on the cassette case.

- File number (if you use one)
- Tape counter reading (helps you position the tape later)
- Volume and tone control settings (helps you read the tape later)

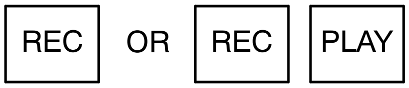
When you have the cable connected to the recorder microphone jack, place the Handheld Programmer keyswitch in the LOAD position. Once the keyswitch has been placed in the LOAD position, the display will go blank and the SHF LED will be turned on. Then, use the following procedure to copy the program.

Enter a four-digit program name (or 0 if you do not want a name)

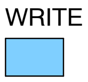




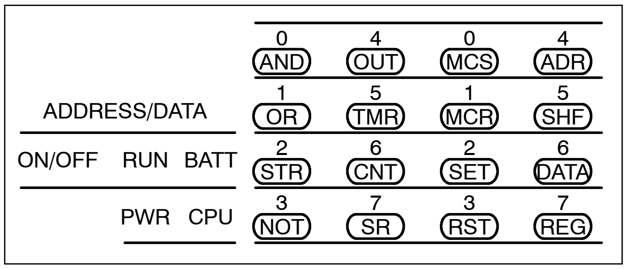
Start the cassette recorder



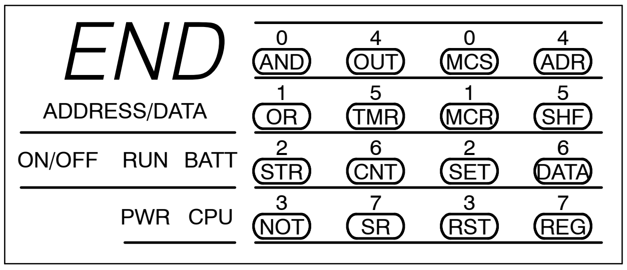
Press Write



(Program takes about 75 seconds to transfer.)



Display when transfer is complete

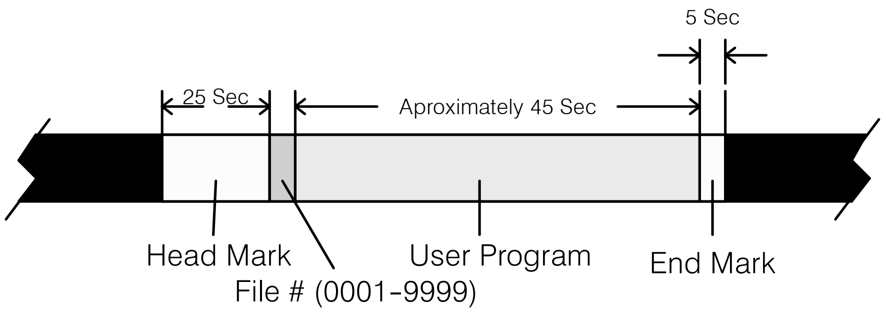


Stop the cassette recorder

Verifying the Tape Contents

It's usually a good idea to verify the tape contents match what is stored in the CPU. The last thing you want to do is to reload the program at a later date only to find the program isn't what you expected.

Before you begin, it's important to understand how the information is placed on the tape. You'll need to know this when in order to position the tape correctly.



☐ The Head and End marks are written with 2kHz signals

▨ One signal is at 2kHz and 0 signal is at 1kHz

▢ This can be omitted if program identification by number is not needed

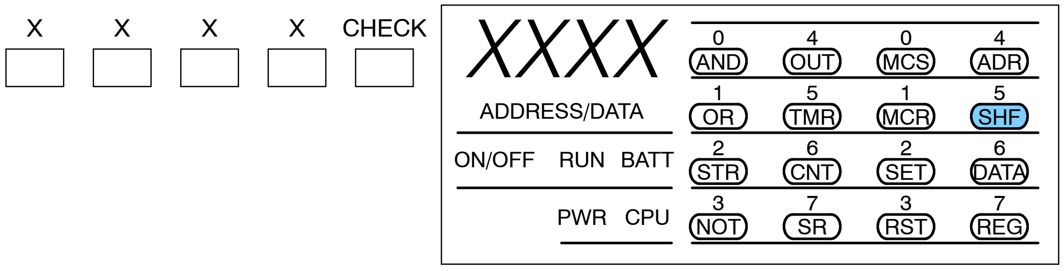
Part of the problem with cassette storage is that tape positioning can mean the difference between a successful transfer and a serious headache. Before you begin the procedure make sure you have positioned the tape just before the first head mark, or, just before the location of the program you want to read. If you position the tape in the head mark, an error will occur.

Set the tape recorder TONE and VOLUME controls to the settings you used when you recorded the program. If you don't remember the settings, set the controls to the midway position. (You may have to adjust the volume later.)

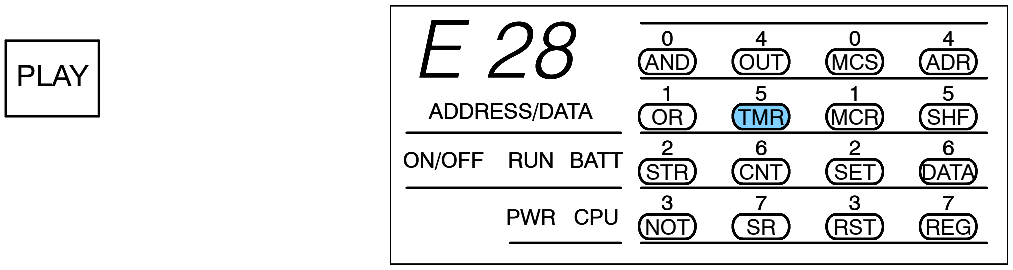
Also, remember the Handheld Programmer keyswitch has to be in the LOAD position to perform tape operations.

When you have the cable connected to the earphone jack, place the Handheld Programmer keyswitch in the LOAD position. The display will go blank and the SHF LED will be turned on. Then, use the following procedure to copy the program.

Enter the four-digit program name (if you used one)



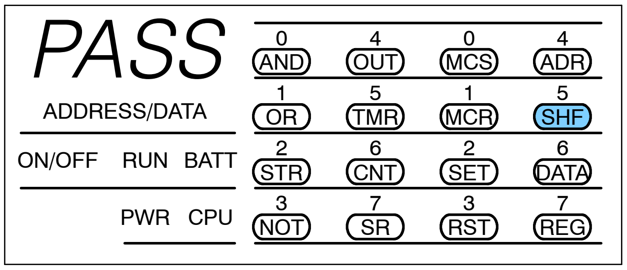
Start the cassette recorder



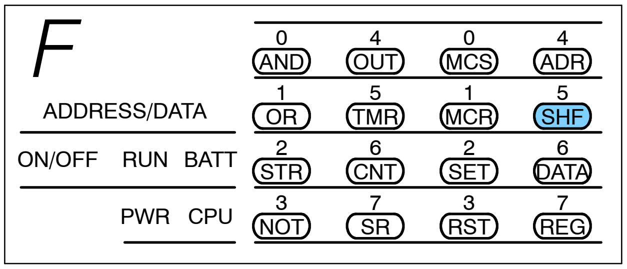
LED 5 flashes and the message E28 appears when the head mark is found. The message disappears and LED 7 comes on if the procedure is working correctly. If the E28 message does not disappear, adjust the volume level until it does. You have 12 seconds to complete the volume adjustment.

If you entered a file number, the system will check the tape to make sure the file numbers match. If the file does not match, the current file will be “passed” and the process will continue until the correct file is located. The following displays are used.

Display when the file number does not match

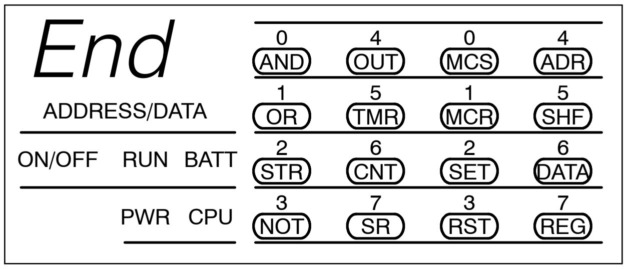


Display when the file number is found

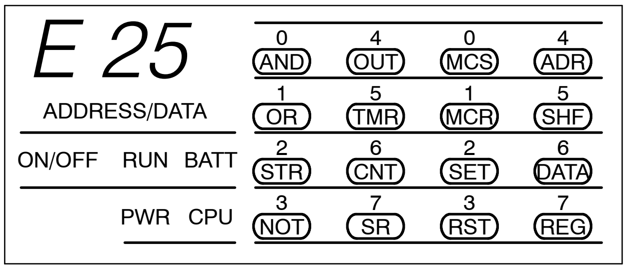


When the check is complete, one of two displays will appear.

Display when the programs match



Display when an error occurs



You can press CLR to remove the error message and start over. If you get an error there are several things you can check.

- Check the tape positioning. It is very important to position the tape correctly. This is the cause of most errors.
- Adjust the volume level and/or tone controls.
- Check the cable connections.
- Clean the tape recorder head. Use the documentation that came with your recorder to determine the correct cleaning procedures.

Reading a Program from a Cassette

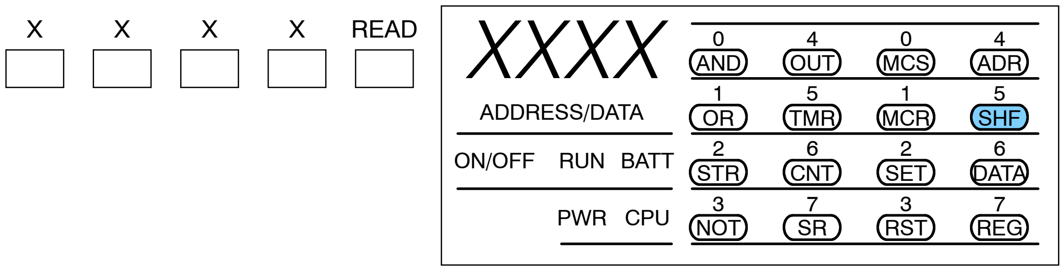
The procedure to read a program is almost exactly the same as the one for verifying the tape contents. Before you begin the procedure make sure you have positioned the tape just before the first head mark, or just before the location of the program you want to read. If you position the tape in the head mark, an error will occur.

Set the tape recorder TONE and VOLUME controls to the settings you used when you recorded the program. If you don't remember the settings, set the controls to the midway position. (You may have to adjust the volume later).

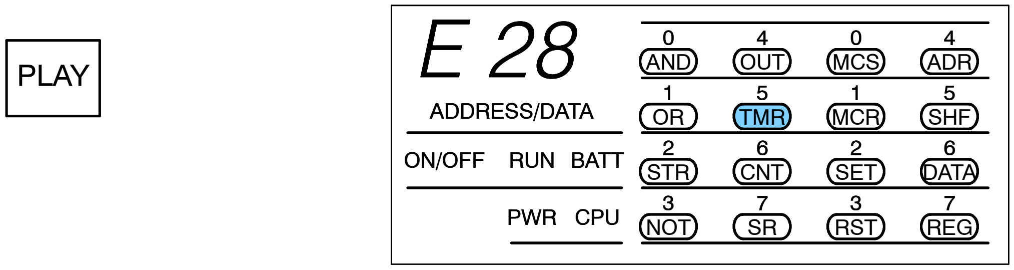
Also, remember the Handheld Programmer keyswitch has to be in the LOAD position to perform tape operations.

When you have the cable connected to the earphone jack, place the Handheld Programmer keyswitch in the LOAD position. The display will go blank and the SHF LED will be turned on. Then, use the following procedure to copy the program.

Enter the four-digit program name (if you used one)



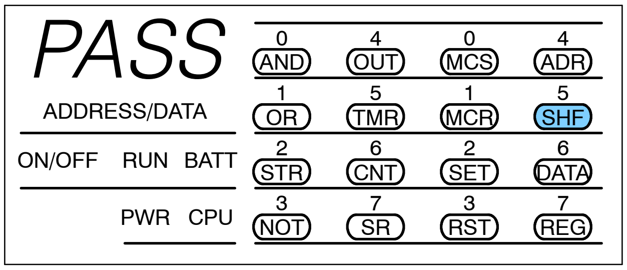
Start the cassette recorder



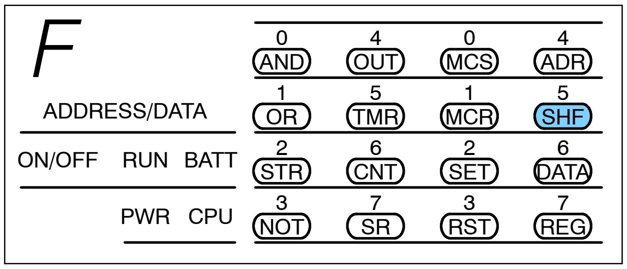
LED 5 flashes and the message E28 appears when the headmark is found. The message disappears and LED 7 comes on if the procedure is working correctly. If the E28 message does not disappear, adjust the volume level until it does. You have 12 seconds to complete the volume adjustment.

If you entered a file number, the system will check the tape to make sure the file numbers match. If the file does not match, the current file will be “passed” and the process will continue until the correct file is located. The following displays are used.

Display when the file number does not match

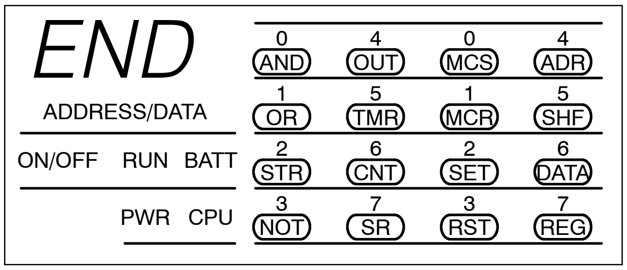


Display when the file number is found



When the program has been loaded, the following display will appear.

Display when the program has been loaded



You can press CLR to remove the error message and start over. If you get an error there are several things you can check.

- Check the tape positioning. It is very important to position the tape correctly. This is the cause of most errors.
- Adjust the volume level and/or tone controls.
- Check the cable connections.
- Clean the tape recorder head. Use the documentation that came with your recorder to determine the correct cleaning procedures.

As you've seen, entering and storing programs with the Handheld is a pretty simple task. Once you've got the program entered and the machine is up and running, you can use the Handheld to monitor and change machine operations almost as easily. The next chapter shows the details.

6

System Monitoring and Troubleshooting

In This Chapter. . . .

Troubleshooting Suggestions

The Handheld is very useful in troubleshooting your machine. As with any problem, you have to find it before you can fix it. There are several operations and features that help you quickly find the exact cause of system problems.

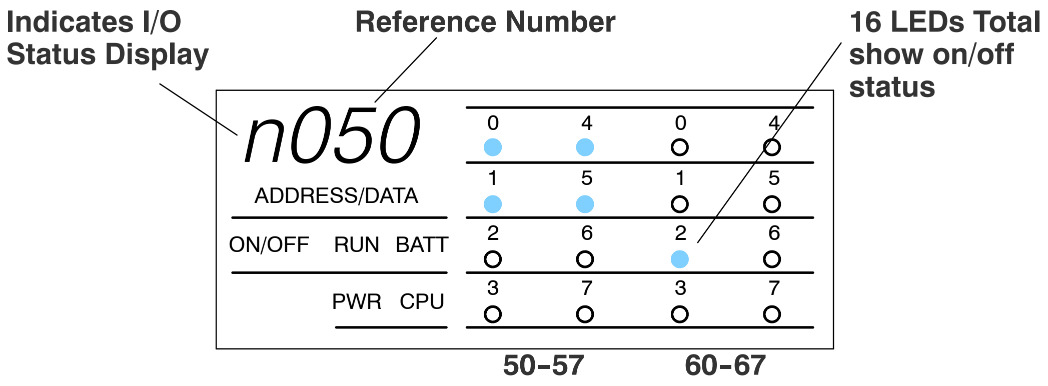
- Monitor Discrete I/O Points — to examine I/O power flow for individual I/O points.
- Force Discrete I/O Points — to examine machine sequences or inconsistencies.
- Monitor Register Locations — to examine word locations to determine if correct values are being used.
- Change Register Values — to force word locations with different values.
- Monitor Timer/Counter Values — to adjust machine timing elements.
- Understand Error Codes — to know where to look first.

Monitoring Discrete Points

You can monitor up to 16 discrete points at one time. The points can be from the following data types.

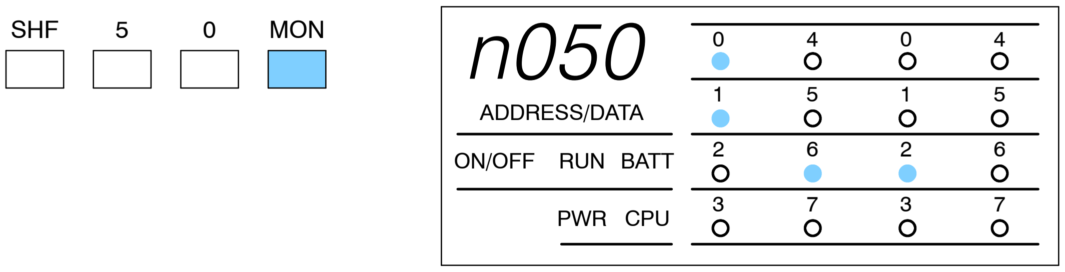
- Inputs
- Output
- Control relays

The status display area is also used to show the status for the points being monitored. Here's how the display is organized.

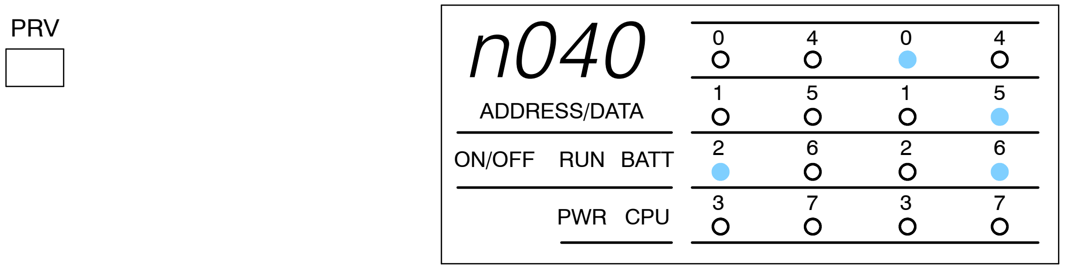


Use the following keystrokes to monitor discrete points. To select a different data type, use the corresponding reference number instead of the one shown. The DL305 memory map is included in Appendix A. This will help you determine the appropriate ranges for the various data types.

Select the reference number to monitor



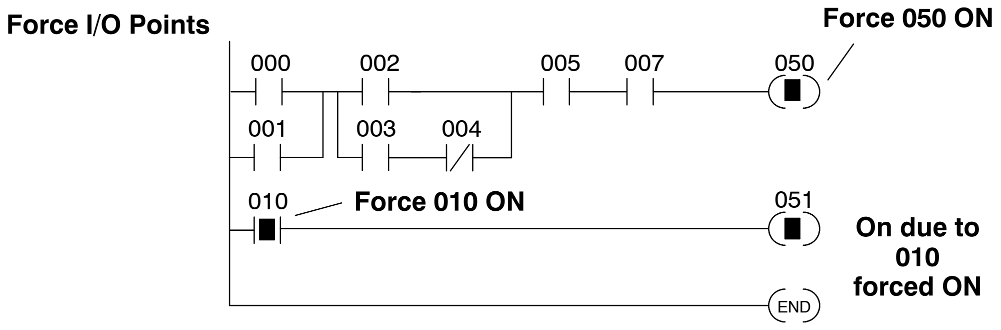
Use PRV or NXT to scroll through the references (8 point increments)



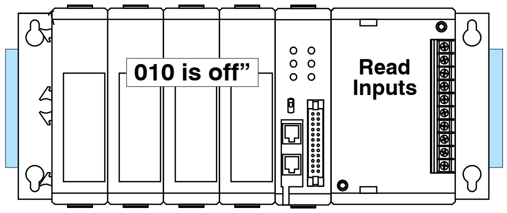
Forcing Discrete Points

You can also force discrete points with the Handheld Programmer.

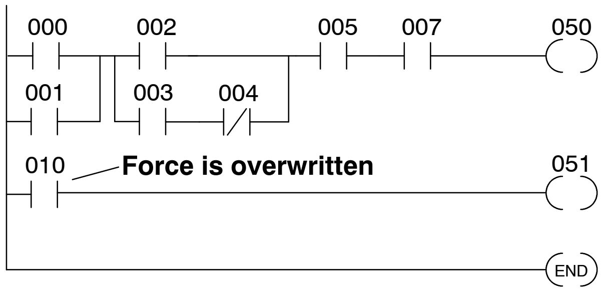
It is important to note that the DL305 CPUs only retain the forced value for one scan if the output point is used in the logic program or if the input point used corresponds to module that is installed in the base. The following example shows how the forcing actually works.



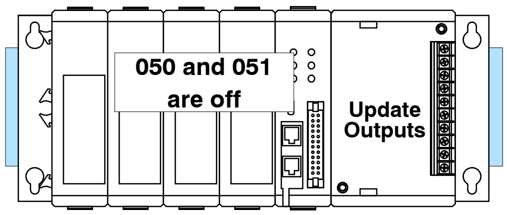
Next Scan



CPU reads the I/O status from the modules. Sees that point 010 is off, overwrites force to turn off 010.



Logic is solved. Point 010, even though previously forced on, is turned off. Points 050 and 051 are turned off since conditions are not met.



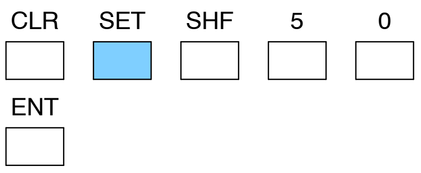
CPU updates the output status with the results obtained from the logic execution. Y0 and Y1 were turned off.

NOTE: If you use a CR as an input, you will not have the "one scan" problem.

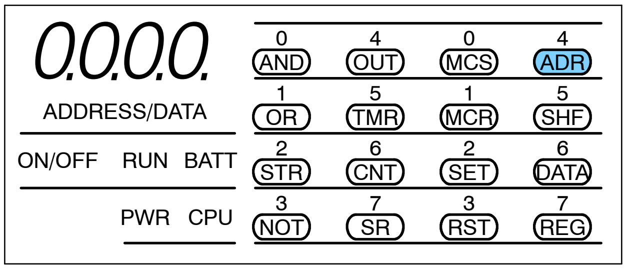
The following example shows the keystrokes required to force an I/O point.

WARNING: Depending on your application, forcing I/O points may cause unpredictable machine operation that can result in a risk of personal injury or equipment damage.

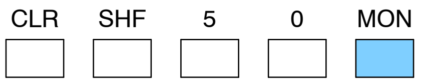
To turn a point on

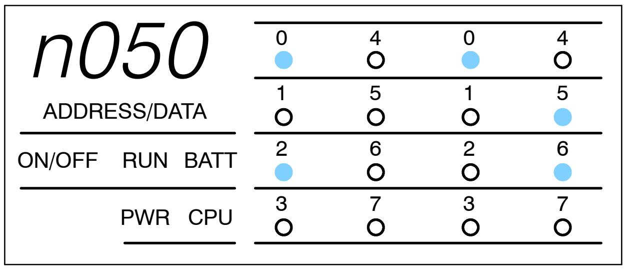


Display returns to address display

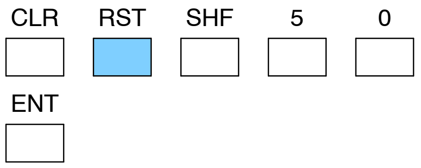


Monitor the point to verify the force (optional)

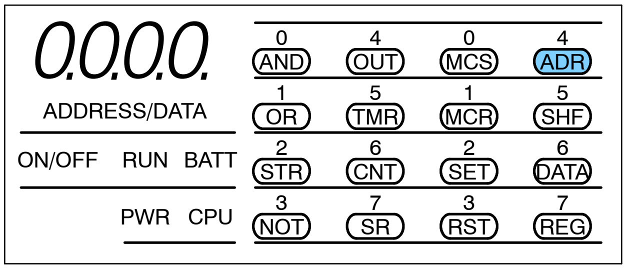




To turn a point off



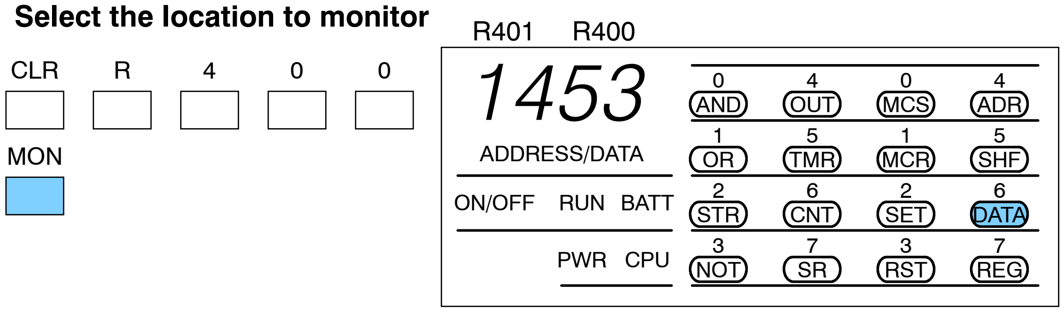
Display returns to address display



Monitoring Register Locations

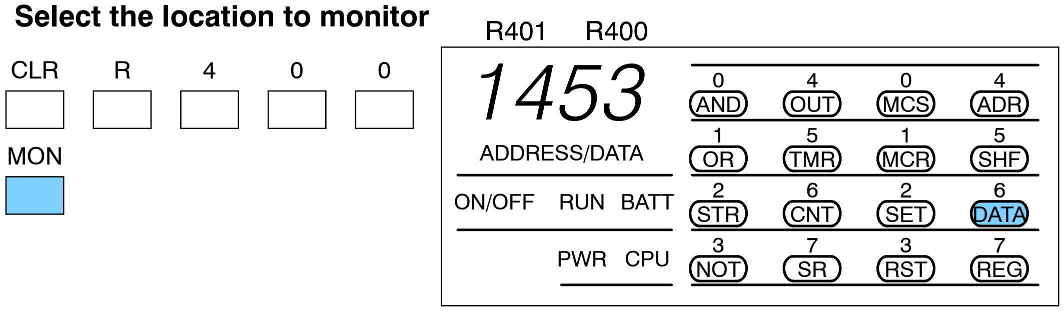
You can use the Handheld to monitor and change register locations. When a register is monitored, the handheld programmer will display two register locations (eight bits each) this means that 4 digits of data will be shown. Since data register locations are 8-bit locations, two consecutive data registers will be displayed. When changing values in data register locations, you can also write two consecutive data register locations.

Select the location to monitor

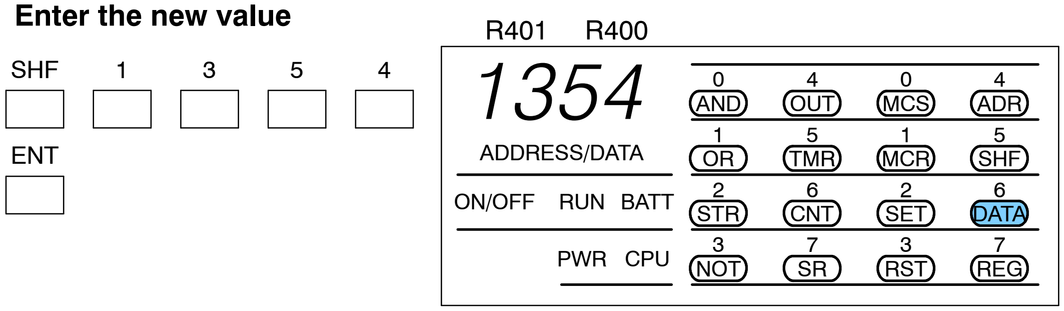


Changing Register Values

Select the location to monitor



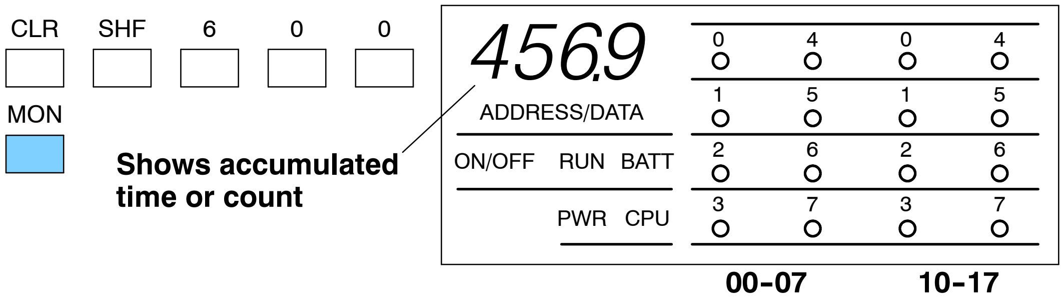
Enter the new value



Monitoring Timer/Counter Current Values

You can also use the Handheld to monitor and/or change timer and counter current values. There are two ways to do this. You can use the register monitoring procedure discussed previously which will display the current value with the decimal point implied for timers. The second method uses slightly different keystrokes and will show timer current values with the decimal point. Using this method also uses the instructions LEDs to indicate the last two digits of the timer/counter memory reference.

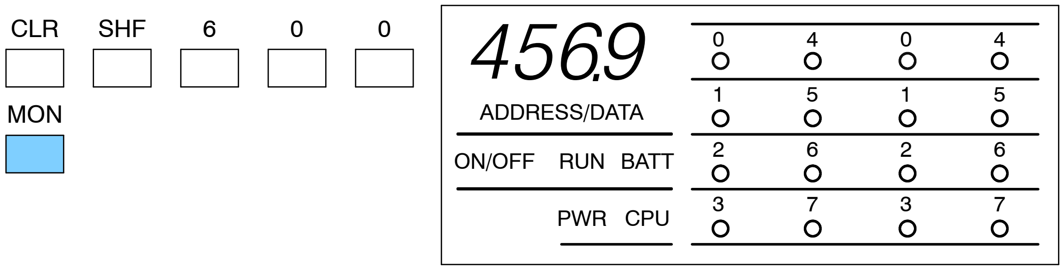
Select the location to monitor



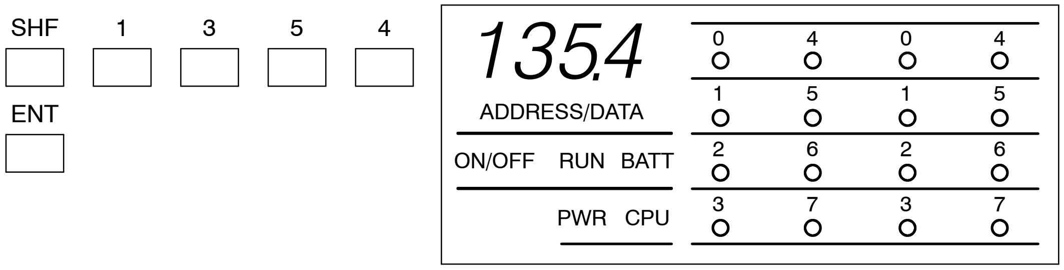
LEDs show last two digits of the register reference number

Changing Timer/Counter Current Values

Select the timer location to monitor



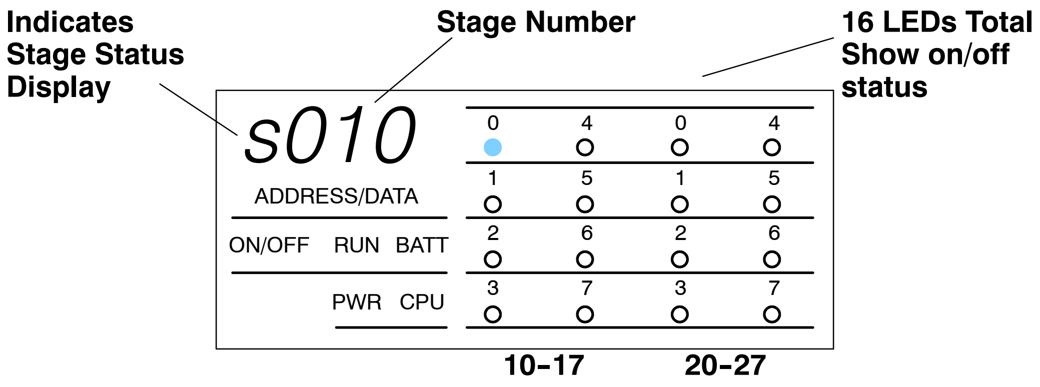
Enter the new value



Monitoring Program Stages

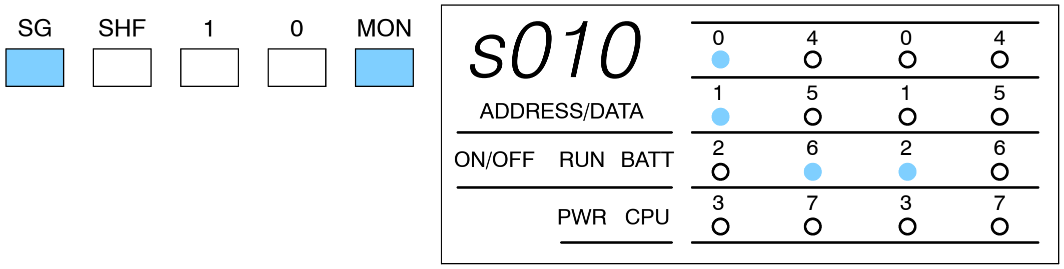
The RLL^{PLUS} CPUs also have stages that can be monitored. This is especially useful since it can quickly show you exactly which parts of the program are being executed.

The procedure is very similar to the one used for monitoring discrete I/O points. Here's how the display is organized.

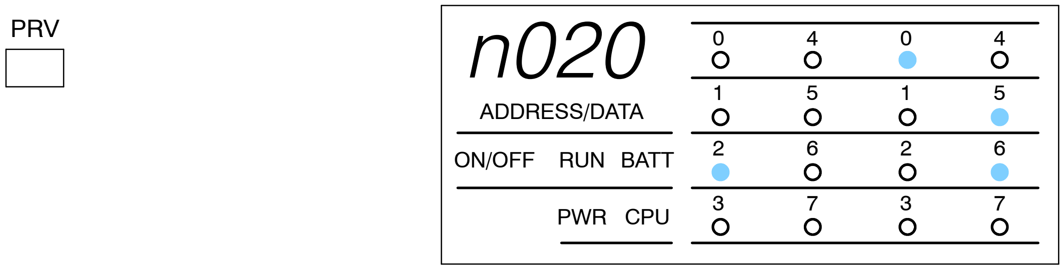


Use the following keystrokes to monitor the stages.

Select the stage number to monitor



Use PRV or NXT to scroll through the references (8 stage increments)

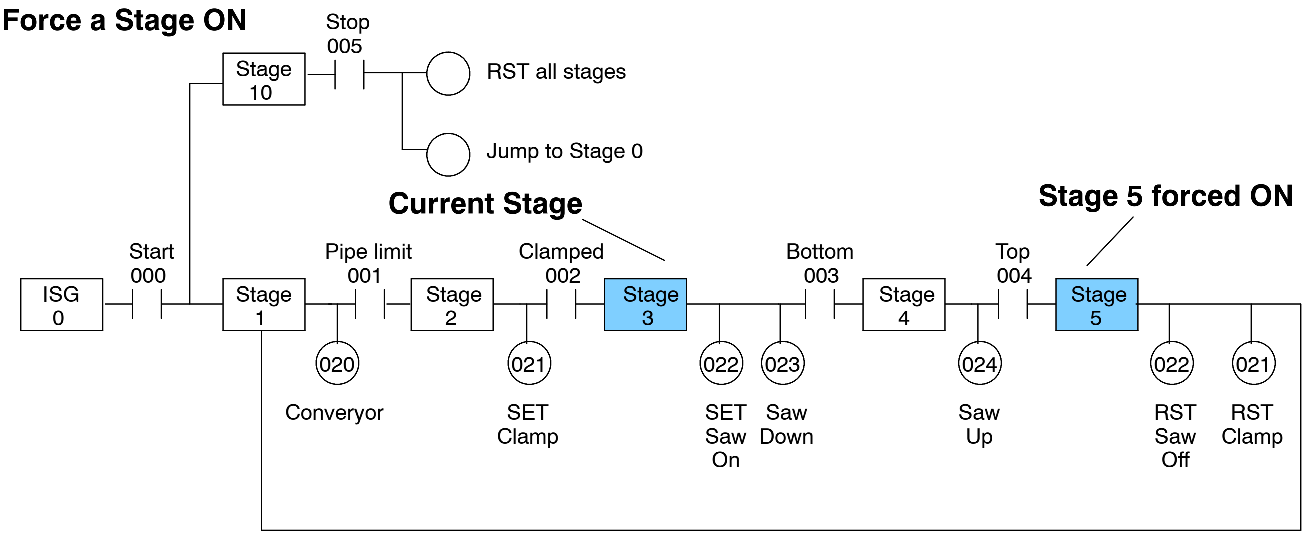


Forcing Program Stages

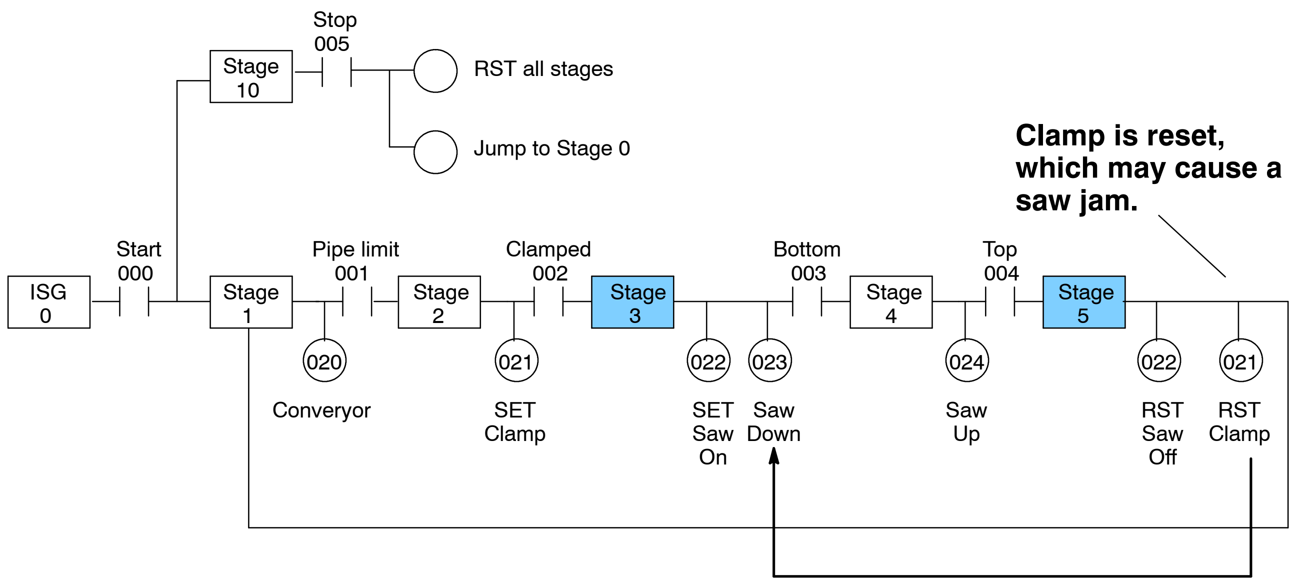
You can also force Program Stages with the Handheld Programmer. However, due to the self-contained nature of this style of programming, you can really cause some problems by forcing stages on and off.

It is important to note that the DL305 CPUs only retain the forced value for one scan if the logic within the stage (or another stage) causes the status to be discarded. The following example shows how the forcing actually works. Assume that the saw takes approximately 10 seconds to reach the bottom limit. (Which is many, many, CPU scans.)

Force a Stage ON



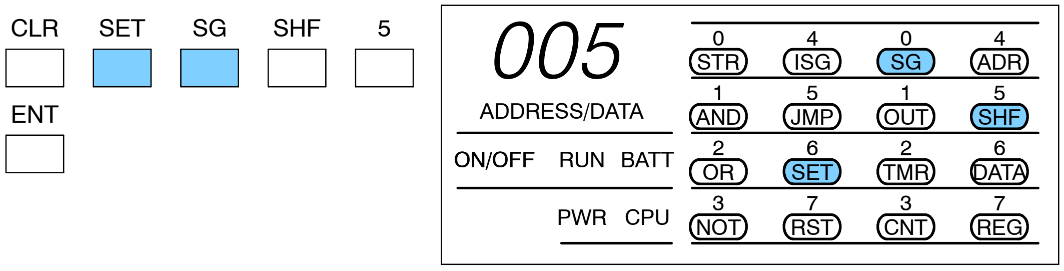
Next Scan



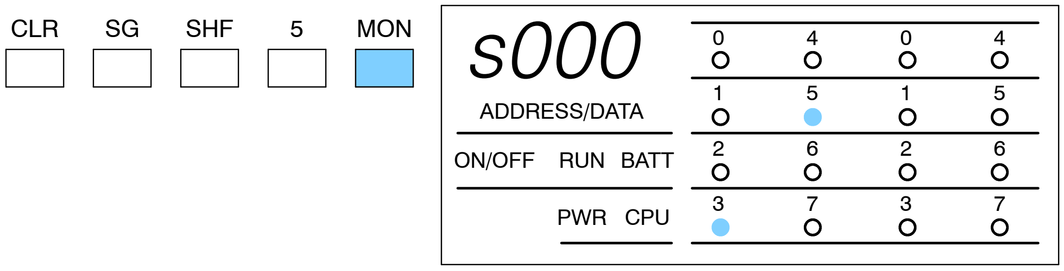
Obviously, there are times when it's perfectly OK to force a program stage on or off. The following example shows the keystrokes required to force an stage.

WARNING: As shown in the example, forcing stages may cause unpredictable machine operation that can result in a risk of personal injury or equipment damage.

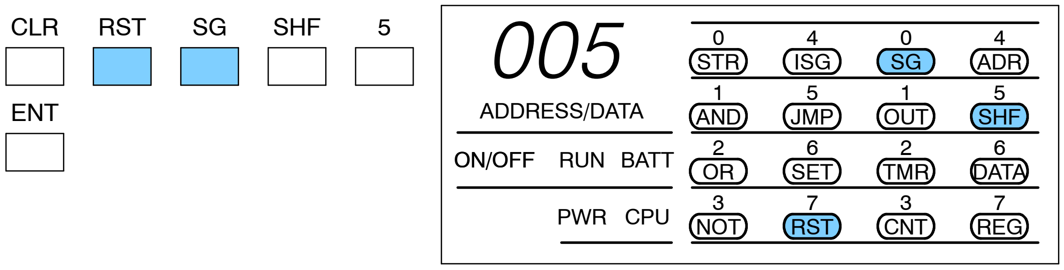
To turn a stage on



Monitor the stage to verify the force (optional)



To turn a stage off



Error Codes

The following table lists the error codes that may appear on the Handheld.

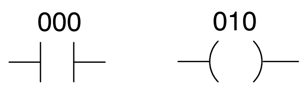
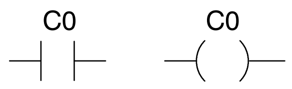
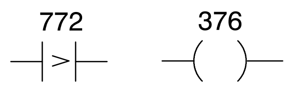
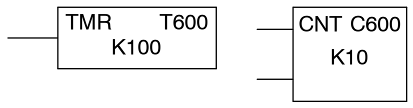
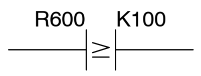
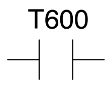
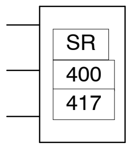
DL305 Error Code	Description
E01 Invalid Keystrokes	Invalid keystroke or series of keystrokes entered into the handheld programmer. Refer to the DL305 Handheld Programmer manual for assistance in the operation you are trying to perform.
E02 Input Point Programmed as Output	An I/O point dedicated to an input module has been used as an output in the application program. Change the I/O reference number in the program which is causing the error.
E03 Stack Overflow	The maximum number of instructions utilizing the internal stack has exceeded eight. These instructions can be a combination of AND STRs, OR STRs and MCS/MCR groups. Reduce the number of these instructions which are pushed onto the stack at one time.
E05 (NON Stage) Duplicate Coil Reference	Two or more output coils have the same data type and number. Change the duplicate coil to correct the error. Duplicate coil references are valid with the SET instruction.
E05 (Stage) Duplicate Stage Reference	Two or more Stages have the same reference number. Change the duplicate Stage number to correct the error.
E06 MCR/MCS Mismatch	The number of MCR instructions do not match the number of MCS instructions. Each MCR must have an accompanying MCS.
E07 Missing CNT or SR Contact	A required input contact is missing from a CNT (example, RESET input) or a SR instruction. Refer to the DL305 User Manual for details on these instructions.
E08 Invalid Data Values	The required data values for a TMR, CNT or SR are missing or incorrect. Refer to the DL305 User Manual for details on these instructions.
E09 Incomplete Program Rung	The rung does not terminate with an output as required. Program an output to terminate the rung properly.
E11 Program Full	There is no available program addresses in memory. Reduce the size of the program.

DL305 Error Code	Description
E21 Program Memory Parity Error	A parity error has occurred in the program memory of the CPU. Clear the memory and reload the program. If the error reoccurs replace the CPU. Electrical noise will cause this problem.
E22 Password Error	The password stored in the CPU is invalid. Press the “CLR” key twice on the handheld programmer and the password will be reset to 0000. Re-enter the password if required.
E25 Tape/Program Mismatch	A mismatch was found when a compare was performed on the program in CPU memory and the program stored on tape.
E28 Volume Incorrect On Tape Device.	The volume is incorrect on the tape player being used to load the program to the CPU. Adjust the volume and retry the operation. Refer to the DL305 Handheld Programmer manual for details on tape operation.
E31 RAM Limit Exceeded	The application program required more RAM for execution than is available. Reduce the length of the program.
E99 Instruction Not Found	A search was performed and the specified instruction was not found in the application program.

DL305 Memory Map

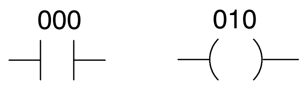
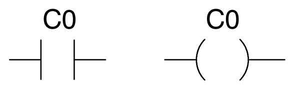
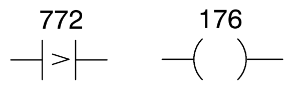
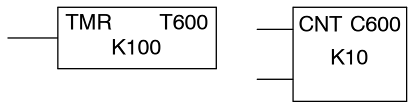
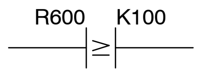
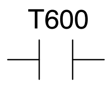
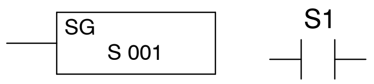
In This Chapter. . . .

DL330 Memory Map

Memory Type	Discrete Memory Reference (octal)	Register Memory Reference (octal)	Qty. Decimal	Symbol
Input / Output Points	000 – 157 700 – 767	R000 – R015 R070 – R076	168 Total	
Control Relays	160 – 373	R016 – R037	140	
Special Relays	374 – 377 770 – 777	R037 R077	12	
Timers / Counters	600 – 673 674 – 677*	None	64	
Timer / Counter Current Values	None	R600 – R673 R674 – R677*	64	
Timer / Counter Status Bits	T600 – T673 T674 – T677*	None	64	
Data Words	None	R400 – R563	116	None specific, used with many instructions
Shift Registers	400 – 577	None	128	
Special Registers	None	R574 – R577	4	R574 – R575 used with FAULT R576 – R577 Auxiliary Accumulator

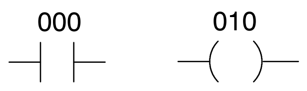
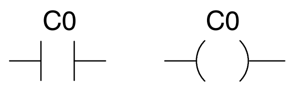
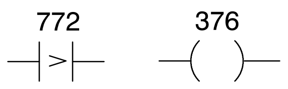
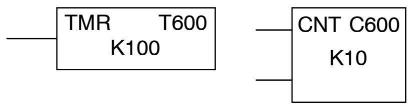
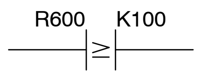
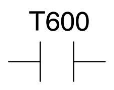
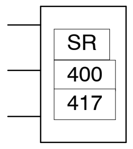
* T/ C Setpoint Unit Only. Can be used as data registers if the Timer/Counter Setpoint Unit or Thumbwheel Interface Module is not used. R564 – R573 contain the preset value used with the Timer / Counter Setpoint Unit. R674 – R677 contain the current values for these timers or counters.

DL330P Memory Map

Memory Type	Discrete Memory Reference (octal)	Register Memory Reference (octal)	Qty. Decimal	Symbol
Input / Output Points	000 – 157 700 – 767	R000 – R015 R070 – R076	168 Total	
Control Relays	160 – 174 200 – 277	R016 – R017 R020 – R027	77	
Special Relays	175 – 177 770 – 777	R017 R077	11	
Timers / Counters	600 – 673 674 – 677*	None	64	
Timer / Counter Current Values	None	R600 – R673 R674 – R677*	64	
Timer / Counter Status Bits	T600 – T673 T674 – T677*	None	64	
Data Words	None	R400 – R563	116	None specific, used with many instructions
Stages	S0 – S177	R100 – R117	128	
Special Registers	None	R574 – R577	4	R574 – R575 used with FAULT R576 – R577 Auxiliary Accumulator

* T/ C Setpoint Unit Only. Can be used as data registers if the Timer/Counter Setpoint Unit or Thumbwheel Interface Module is not used, which provides a total of 128 data registers.
R564 – R573 contain the preset value used with the Timer / Counter Setpoint Unit. R674 – R677 contain the current values for these timers or counters.

DL340 Memory Map

Memory Type	Discrete Memory Reference (octal)	Register Memory Reference (octal)	Qty. Decimal	Symbol
Input / Output Points	000 – 157 700 – 767	R000 – R015 R070 – R076	168 Total	
Control Relays	160 – 373 1000 – 1067	R016 – R037 R100 – R106	180	
Special Relays	374 – 377 770 – 777 1070 – 1077	R037 R077 R107	20	
Timers / Counters	600 – 673 674 – 677*	None	64	
Timer / Counter Current Values	None	R600 – R673 R674 – R677*	64	
Timer / Counter Status Bits	T600 – T673 T674 – T677*	None	64	
Data Words	None	R400 – R563 R700 – R767	172	None specific, used with many instructions
Shift Registers	400 – 577	None	128	
Special Registers	None	R574 – R577 R770 – R777	12	R574–R575 used with FAULT R576–R577 Auxiliary Accumulator R770–R777 Communications Setup

* T/ C Setpoint Unit Only. Can be used as data registers if the Timer/Counter Setpoint Unit or Thumbwheel Interface Module is not used. R564 – R573 contain the preset value used with the Timer / Counter Setpoint Unit. R674 – R677 contain the current values for these timers or counters.

I/O Point Bit Map

These tables provide a listing of the individual Input points associated with each register location for the DL330, DL330P, and DL340 CPUs.

MSB	I/O References						LSB	Register Number
007	006	005	004	003	002	001	000	R0
017	016	015	014	013	012	011	010	R1
027	026	025	024	023	022	021	020	R2
037	036	035	034	033	032	031	030	R3
047	046	045	044	043	042	041	040	R4
057	056	055	054	053	052	051	050	R5
067	066	065	064	063	062	061	060	R6
077	076	075	074	073	072	071	070	R7
107	106	105	104	103	102	101	100	R10
117	116	115	114	113	112	111	110	R11
127	126	125	124	123	122	121	120	R12
137	136	135	134	133	132	131	130	R13
147	146	145	144	143	142	141	140	R14
157	156	155	154	153	152	151	150	R15
167	166	165	164	163	162	161	160	n/a
177	176	175	174	173	172	171	170	n/a
707	706	705	704	703	702	701	700	R70
717	716	715	714	713	712	711	710	R71
727	726	725	724	723	722	721	720	R72
737	736	735	734	733	732	731	730	R73
747	746	745	744	743	742	741	740	R74
757	756	755	754	753	752	751	750	R75
767	766	765	764	763	762	761	760	R76

NOTE: 160 – 167 can be used as I/O in a DL330 or DL330P CPU under certain conditions. 160 – 177 can be used as I/O in a DL340 CPU under certain conditions. You should consult the DL305 User Manual to determine which configurations allow the use of these points.

These points are normally used as control relays. You cannot use them as both control relays and as I/O points. Also, if you use these points as I/O, you cannot access these I/O points as a Data Register reference.

Control Relay Bit Map

The following tables provide a listing of the individual control relays associated with each register location for the DL305 CPUs.

NOTE: 160 – 167 can be used as I/O in a DL330 or DL330P CPU under certain conditions. 160 – 177 can be used as I/O in a DL340 CPU under certain conditions. You should consult the DL305 User Manual to determine which configurations allow the use of these points.

You cannot use them as both control relays and as I/O points. Also, if you use these points as I/O, you cannot access these I/O points as a Data Register reference.

MSB			DL330 Control Relay References				LSB	Register Number
167	166	165	164	163	162	161	160	R16
177	176	175	174	173	172	171	170	R17
207	206	205	204	203	202	201	200	R20
217	216	215	214	213	212	211	210	R21
227	226	225	224	223	222	221	220	R22
237	236	235	234	233	232	231	230	R23
247	246	245	244	243	242	241	240	R24
257	256	255	254	253	252	251	250	R25
267	266	265	264	263	262	261	260	R26
277	276	275	274	273	272	271	270	R27
307	306	305	304	303	302	301	300	R30
317	316	315	314	313	312	311	310	R31
327	326	325	324	323	322	321	320	R32
337	336	335	334	333	332	331	330	R33
347	346	345	344	343	342	341	340	R34
357	356	355	354	353	352	351	350	R35
367	366	365	364	363	362	361	360	R36
				373	372	371	370	R37

* Control relays 340 – 373 can be made retentive by setting a CPU dipswitch. See the DL305 User Manual for details on setting CPU dipswitches.

MSB	DL330P Control Relay References						LSB	Register Number
167	166	165	164	163	162	161	160	R16
			174	173	172	171	170	R17
207	206	205	204	203	202	201	200*	R20
217	216	215	214	213	212	211	210	R21
227	226	225	224	223	222	221	220	R22
237	236	235	234	233	232	231	230	R23
247	246	245	244	243	242	241	240	R24
257	256	255	254	253	252	251	250	R25
267	266	265	264	263	262	261	260	R26
277*	276	275	274	273	272	271	270	R27

* Control relays 200 – 277 can be made retentive by setting a CPU dipswitch. See the DL305 User Manual for details on setting CPU dipswitches.

MSB	DL340 Control Relay References						LSB	Register Number
167	166	165	164	163	162	161	160	R16
177	176	175	174	173	172	171	170	R17
207	206	205	204	203	202	201	200	R20
217	216	215	214	213	212	211	210	R21
227	226	225	224	223	222	221	220	R22
237	236	235	234	233	232	231	230	R23
247	246	245	244	243	242	241	240	R24
257	256	255	254	253	252	251	250	R25
267	266	265	264	263	262	261	260	R26
277	276	275	274	273	272	271	270	R27
307	306	305	304	303	302	301	300	R30
317	316	315	314	313	312	311	310	R31
327	326	325	324	323	322	321	320	R32
337	336	335	334	333	332	331	330	R33
347	346	345	344	343	342	341	340*	R34
357	356	355	354	353	352	351	350	R35
367	366	365	364	363	362	361	360	R36
				373*	372	371	370	R37
1007	1006	1005	1004	1003	1002	1001	1000	R100
1017	1016	1015	1014	1013	1012	1011	1010	R101
1027	1026	1025	1024	1023	1022	1021	1020	R102
1037	1036	1035	1034	1033	1032	1031	1030	R103
1047	1046	1045	1044	1043	1042	1041	1040	R104
1057	1056	1055	1054	1053	1052	1051	1050	R105
1067	1066	1065	1064	1063	1062	1061	1060	R106

* Control relays 340 – 373 can be made retentive by setting a CPU dipswitch. See the DL305 User Manual for details on setting CPU dipswitches.

Special Relays

The following table shows the Special Relays used with the DL305 CPUs.

CPUs	Special Relay	Description of Contents
DL330P	175	100 ms clock, on for 50 ms and off for 50 ms.
	176	Disables all outputs except for those entered with the SET OUT instruction.
	177	Battery voltage is low.
DL330 DL340	374	On for the first scan cycle after the CPU is switched to Run Mode.
	375	100 ms clock, on for 50 ms and off for 50 ms.
	376	Disables all outputs except for those entered with the SET OUT instruction.
	377	Battery voltage is low.
DL330 DL330P DL340	770	Changes timers to 0.01 second intervals. Timers are normally 0.1 second time intervals.
	771	The external diagnostics FAULT instruction (F20) is in use.
	772	The data in the accumulator is greater than the comparison value.
	773	The data in the accumulator is equal to the comparison value.
	774	The data in the accumulator is less than the comparison value.
	775	An accumulator carry or borrow condition has occurred.
	776	The accumulator value is zero.
	777	The accumulator has an overflow condition.
DL340	1074	The RX or WX instruction is active.
	1075	An error occurred during communications with the RX or WX instructions.
	1076	Port 2 communications mode: on = ASCII mode, off = HEX mode
	1077	Port 1 communications mode: on = ASCII mode, off = HEX mode

Timer / Counter Registers and Contacts

The following table shows the locations used for programming timer or counters. Since timers and counters share the same data area, you cannot have timers and counters with duplicate numbers. For example, if you have Timer 600, you cannot have a Counter 600.

Each register contains the current value for the timer or counter. Each timer or counter also has a timer or counter contact with the same reference number.

NOTE: Counter current values are retentive and retain their state after a power cycle.

Timer/Counter References/Registers							
607	606	605	604	603	602	601	600
617	616	615	614	613	612	611	610
627	626	625	624	623	622	621	620
637	636	635	634	633	632	631	630
647	646	645	644	643	642	641	640
657	656	655	654	653	652	651	650
667	666	665	664	663	662	661	660
677*	676*	675*	674*	673	672	671	670

* Used with Timer / Counter Setpoint Unit and /or Thumbwheel Interface Module.

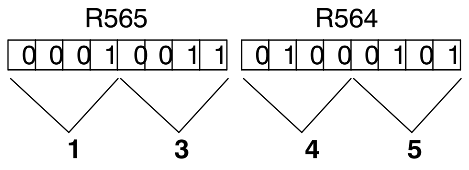
External Timer/Counter Setpoint Unit

Registers 674-677 are used in programming for use with the Timer/Counter Setpoint Unit and the Thumbwheel Interface Module that are available in some compatible product families. The registers contain the current time or count. There is also a status bit for each register with the same reference number. For example, the current value for Timer 674 is stored in R674 and the status contact is T674.

The presets for these modules are stored in R564 - R573 as follows.

- R564 - R565 — 1st T/C preset
- R566 - R567 — 2nd T/C preset
- R570 - R571 — 3rd T/C preset
- R572 - R573 — 4th T/C preset

The example shows how a 4-digit number would be represented in these registers.



Data Registers

The following 8-bit data registers are primarily used with data instructions to store various types of application data. For example, you could use a register to hold a timer or counter preset value.

Some data instructions call for two bytes, which will correspond to two consecutive 8-bit data registers such as R401 and R400. The LSB (Least Significant Bit) will be in register R400 as bit0 and the MSB (Most Significant Bit) will be in register R401 as bit17.

NOTE: Data Registers are retentive.

DL330 / DL330P 8-Bit Data Registers							
407	406	405	404	403	402	401	400
417	416	415	414	413	412	411	410
427	426	425	424	423	422	421	420
437	436	435	434	433	432	431	430
447	446	445	444	443	442	441	440
457	456	455	454	453	452	451	450
467	466	465	464	463	462	461	460
477	476	475	474	473	472	471	470
507	506	505	504	503	502	501	500
517	516	515	514	513	512	511	510
527	526	525	524	523	522	521	520
537	536	535	534	533	532	531	530
547	546	545	544	543	542	541	540
557	556	555	554	553	552	551	550
				563	562	561	560

DL340 8-Bit Data Registers							
407	406	405	404	403	402	401	400
417	416	415	414	413	412	411	410
427	426	425	424	423	422	421	420
437	436	435	434	433	432	431	430
447	446	445	444	443	442	441	440
457	456	455	454	453	452	451	450
467	466	465	464	463	462	461	460
477	476	475	474	473	472	471	470
507	506	505	504	503	502	501	500
517	516	515	514	513	512	511	510
527	526	525	524	523	522	521	520
537	536	535	534	533	532	531	530
547	546	545	544	543	542	541	540
557	556	555	554	553	552	551	550
				563	562	561	560
707	706	705	704	703	702	701	700
717	716	715	714	713	712	711	710
727	726	725	724	723	722	721	720
737	736	735	734	733	732	731	730
747	746	745	744	743	742	741	740
757	756	755	754	753	752	751	750
767	766	765	764	763	762	761	760

Stage Control / Status Bit Map

This table provides a listing of the individual stages and stage control bits. These are only available with the DL330P CPU.

MSB			Stage References				LSB	Register Number
007	006	005	004	003	002	001	000	R100
017	016	015	014	013	012	011	010	R101
027	026	025	024	023	022	021	020	R102
037	036	035	034	033	032	031	030	R103
047	046	045	044	043	042	041	040	R104
057	056	055	054	053	052	051	050	R105
067	066	065	064	063	062	061	060	R106
077	076	075	074	073	072	071	070	R107
107	106	105	104	103	102	101	100	R110
117	116	115	114	113	112	111	110	R111
127	126	125	124	123	122	121	120	R112
137	136	135	134	133	132	131	130	R113
147	146	145	144	143	142	141	140	R114
157	156	155	154	153	152	151	150	R115
167	166	165	164	163	162	161	160	R116
177	176	175	174	173	172	171	170	R117

Shift Register Bit Map

The shift register bits listed below are used in the shift register instruction. These outputs are discrete bits and are not the same locations as the 8 Bit Data Registers. These bits are retentive meaning they retain their state after a power cycle.

NOTE: The DL330P does not have Shift Register bits. Shift Register instructions in the DL330P use Control Relays memory references.

MSB			DL330 / DL340 Shift Register References				LSB	Register Number
407	406	405	404	403	402	401	400	R40
417	416	415	414	413	412	411	410	R41
427	426	425	424	423	422	421	420	R42
437	436	435	434	433	432	431	430	R43
447	446	445	444	443	442	441	440	R44
457	456	455	454	453	452	451	450	R45
467	466	465	464	463	462	461	460	R46
477	476	475	474	473	472	471	470	R47
507	506	505	504	503	502	501	500	R50
517	516	515	514	513	512	511	510	R51
527	526	525	524	523	522	521	520	R52
537	536	535	534	533	532	531	530	R53
547	546	545	544	543	542	541	540	R54
557	556	555	554	553	552	551	550	R55
567	566	565	564	563	562	561	560	R56
577	576	575	574	573	572	571	570	R57

With the DL340 CPU, these bits can also be used as control relays if they are not used with a Shift Register instruction.

Special Registers

This table provides a listing of the special registers used with the DL305 CPUs.

CPUs	Special Register	Description of Contents
DL330 DL330P DL340	R574 – 575	Contains the error code used with the FAULT instruction.
	R576 – 577	Auxiliary accumulator used with the MUL and DIV instructions.
DL340 Only	R771	Sets the upper byte of the station address assigned to the bottom communication port. Therefore, this will contain the 1st and 2nd digits of the address.
	R772	Sets the lower byte of the station address assigned to the bottom communication port. This only contains one digit, which is the 3rd digit of the address.
	R773	Sets the baud rate for the bottom communication port.
	R774	Sets the leading communications delay time for the bottom communication port.
	R775	Sets the trailing communications delay time for the bottom communication port.
	R776	Sets the leading communications delay time for the top communication port.
	R777	Sets the trailing communications delay time for the top communication port.