3

Entering RLL*PLUS* Programs

In This Chapter. . . .

- RLL*PLUS* Programming Basics
- Entering an Initial Stage
- Entering Jump Instructions
- Entering Stage Instructions
- Entering Timers
- Entering Counters
- Entering Shift Registers

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.

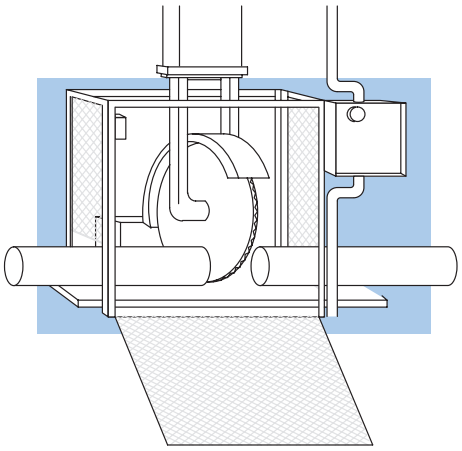
The primary benefit from this programming method is the number of logic interlocks is *significantly* reduced. This is because the individual program segments, called "stages" are completely self contained. When a program segment is active, the inputs and outputs in that stage will be examined and updated 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.

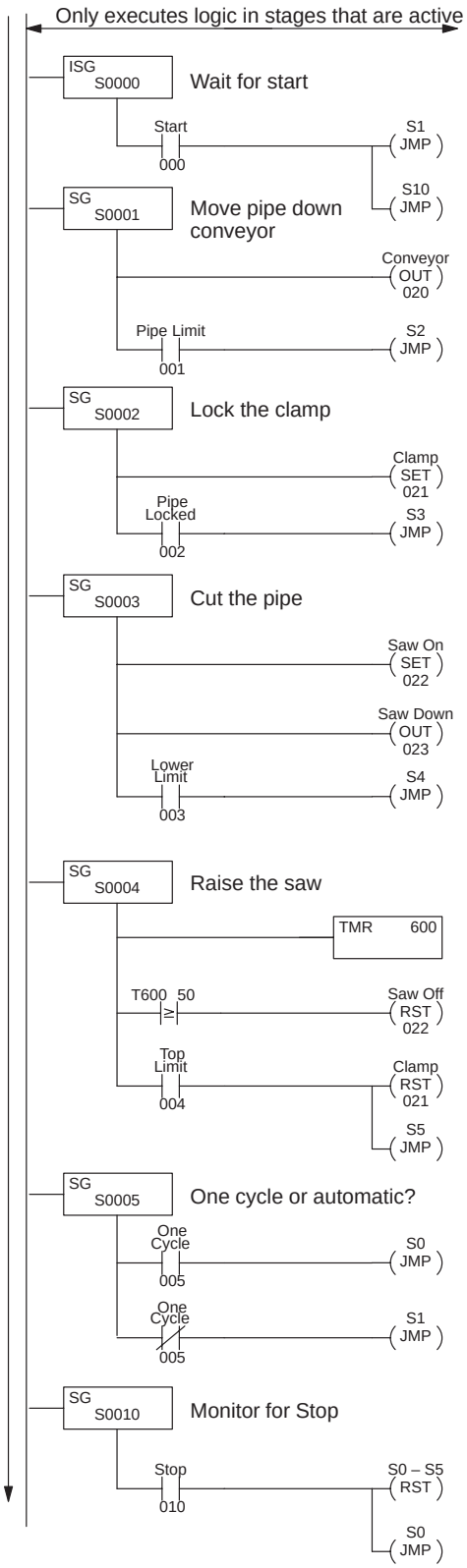
Cutoff Saw



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)

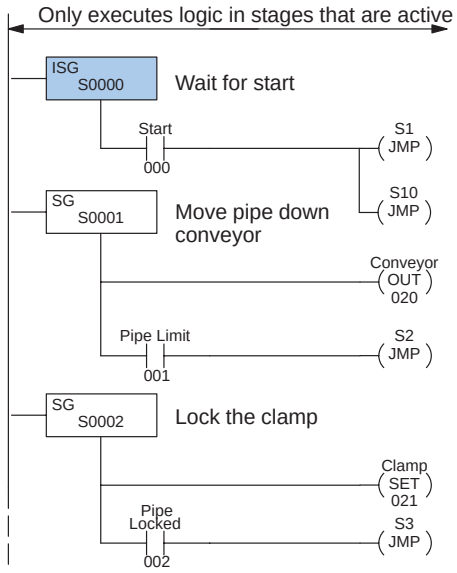
There are also a few other instructions that are used differently (and entered differently) in RLL*PLUS* programs.

- Timers
- Counters
- Shift Registers

The following pages show you how to enter these instructions.

Entering an Initial Stage

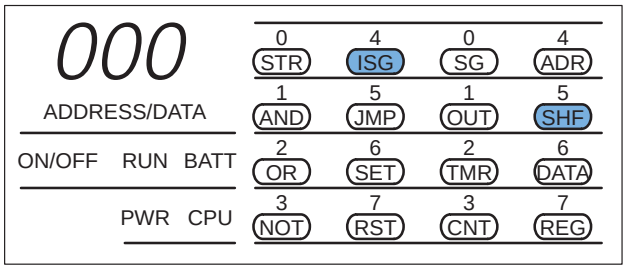
The Initial Stage identifies a starting point in the program. When the CPU enters Run mode, this is where the program execution will begin. The following keystrokes are used to enter an initial stage. (Remember, the RLL*PLUS* Handheld programmer keypad layout and display is different from the RLL Handheld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



Enter the initial stage

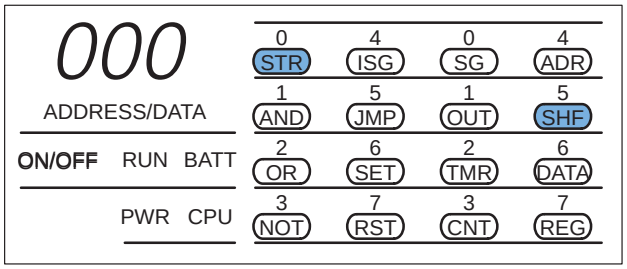
ISG SHF 0 ENT

Data display *before* ENT is pressed



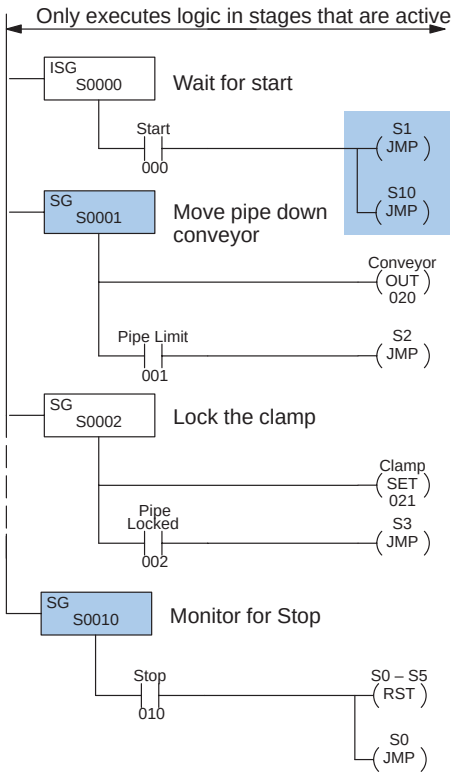
Enter the contact

STR SHF 0 ENT



Entering Jump Instructions

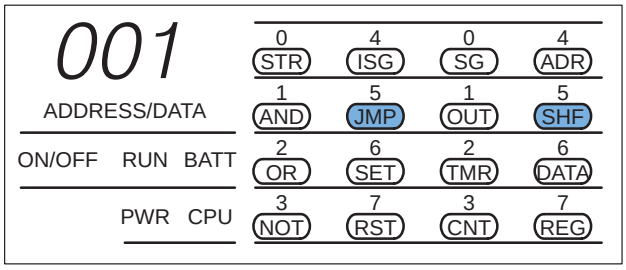
The Jump (JMP) instruction provides a way to transition to multiple points in the program. If you look at the example program you'll notice the program branches to two locations after the operator presses the start switch. The Jump instruction provides this transition. Since the program is jumping to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



Enter the first jump

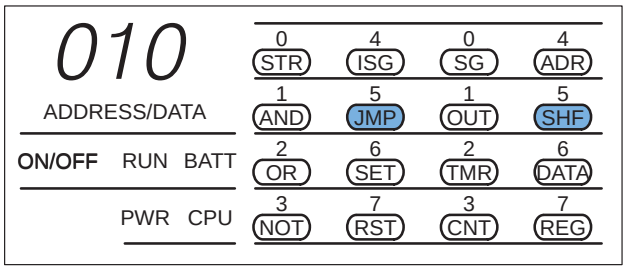
JMP SHF 1 ENT

Data display *before* ENT is pressed



Enter the second jump

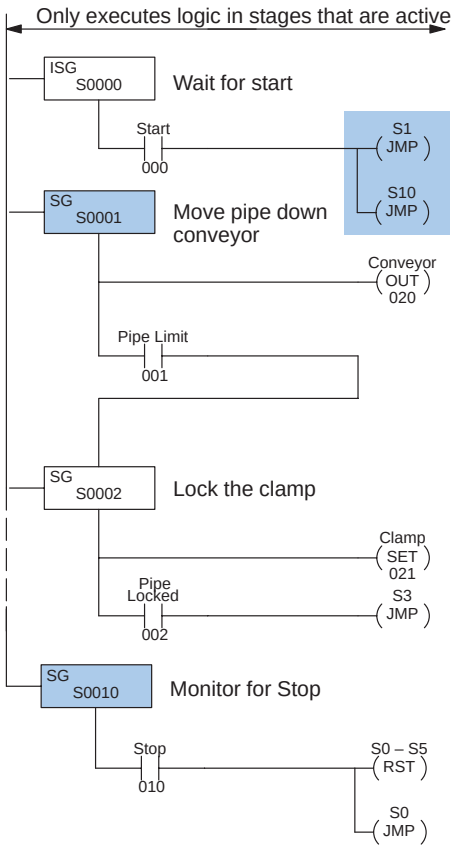
JMP SHF 1 0 ENT



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.

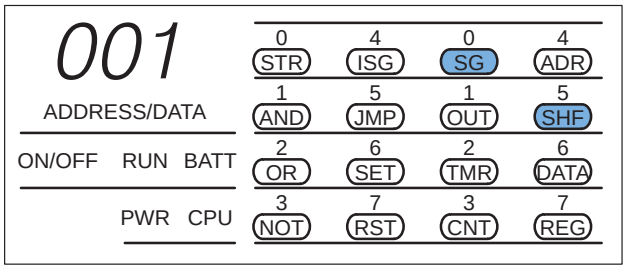


The keystrokes on the following page show how to enter stages 1 and 2. Notice we have changed Stage 2 slightly. This is an example of how a power flow transition looks. A JMP instruction is not required in this example to move from Stage 1 to Stage 2. How do you know when to use a JMP instruction? Simple,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

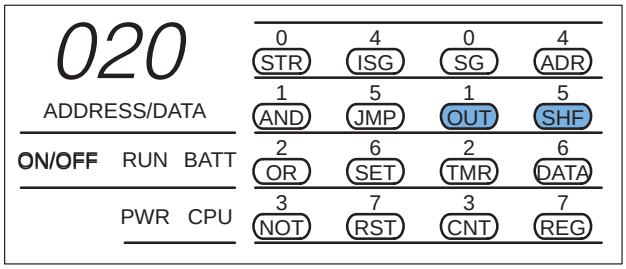
SG SHF 1 ENT

Data display *before* ENT is pressed



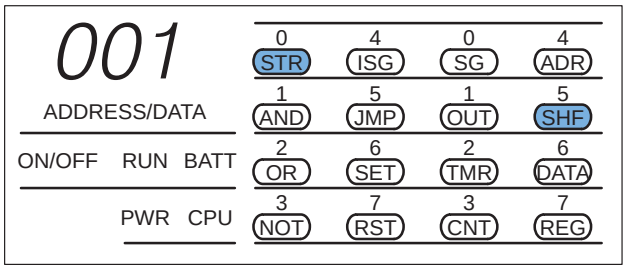
Enter the output for the conveyor

OUT SHF 2 0 ENT



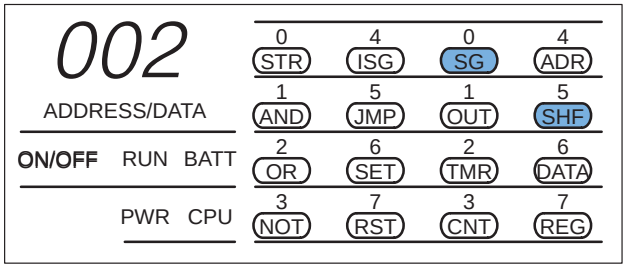
Enter the Pipe Limit contact

STR SHF 1 ENT



Enter Stage 2

SG SHF 2 ENT

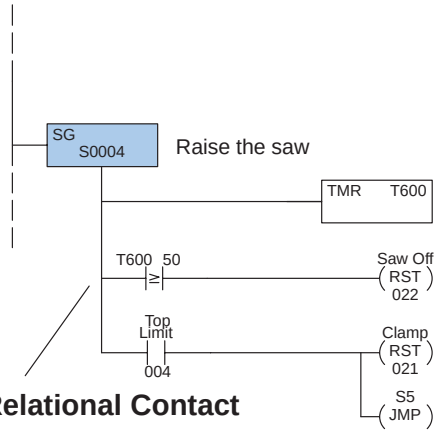


Entering Timers

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.



Relational Contact

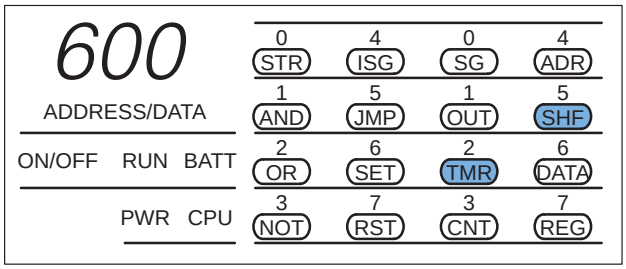
The following example shows how a time delay was added when the saw was being raised. We wanted to keep the saw motor running for 5 seconds so the saw could clear the pipe before being turned off. (Note, the keystrokes only show how to enter the timer and the relational contact, not the whole stage.)

Enter the Timer

TMR SHF 6 0 0

ENT

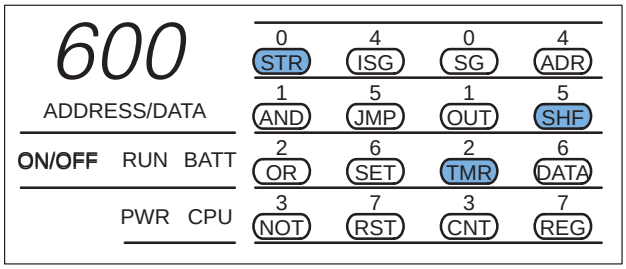
Data display *before* ENT is pressed



Enter the comparative timer contact

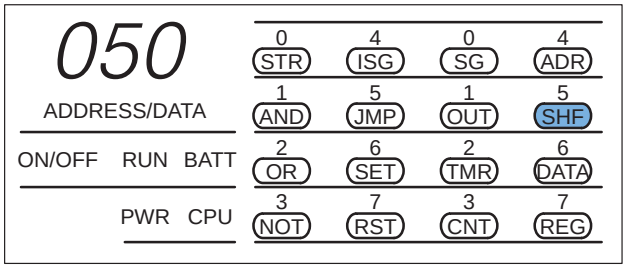
STR TMR SHF 6 0

0 ENT



Enter the compare value

SHF 5 0 ENT

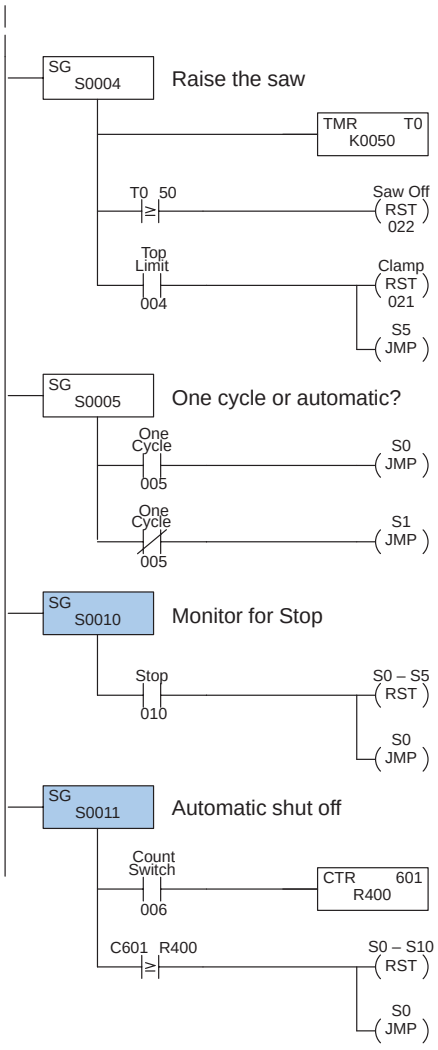


Entering Counters

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.



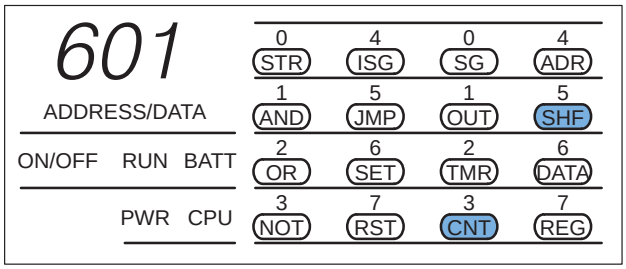
Let's say we wanted to use an operator interface to tell the machine how many pipes to cut. (We'll assume the number is loaded into a register, R400.) Once the correct number of pipes have been cut, the saw should automatically stop. The following example shows how you would add an automatic shutoff 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

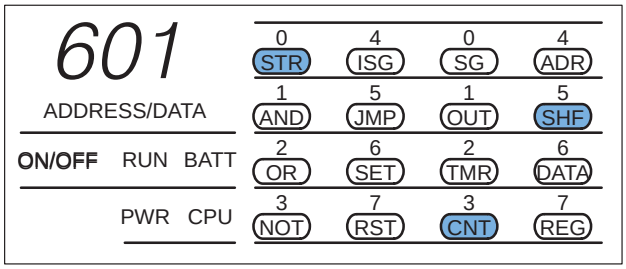
Data display *before* ENT is pressed



Enter the relational contact

STR CNT SHF 6 0

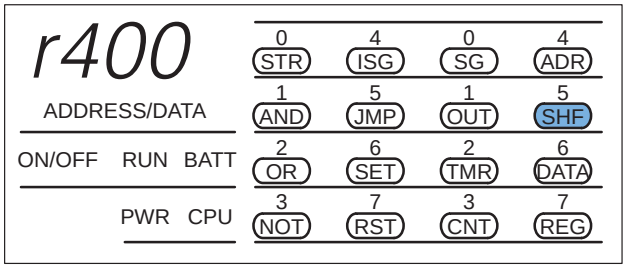
1 ENT



Enter the relational contact comparison register

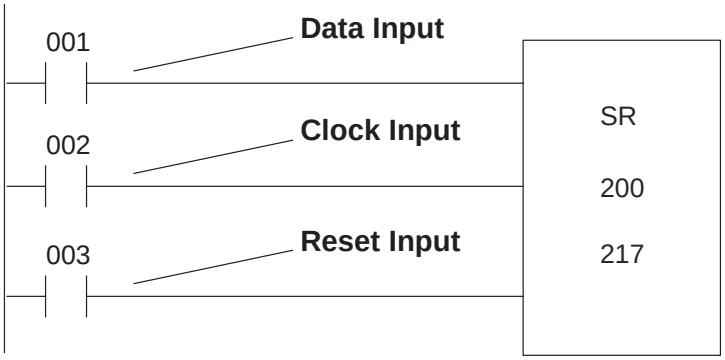
R 4 0 0 ENT

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



Entering Shift Registers

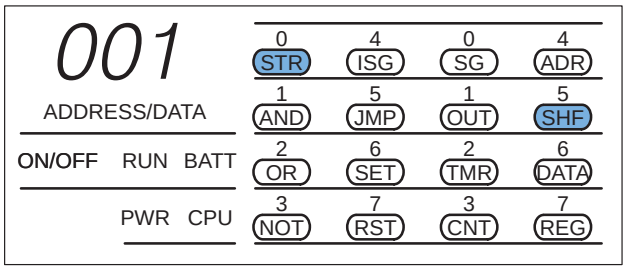
Shift Registers operate the same in RLL*PLUS* programs as they do in RLL programs. However, the keystrokes required to enter a Shift Register are different because the SR key is not on the RLL*PLUS* Handheld Programmer. Also, you do not have a separate range of bits available for use as shift register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

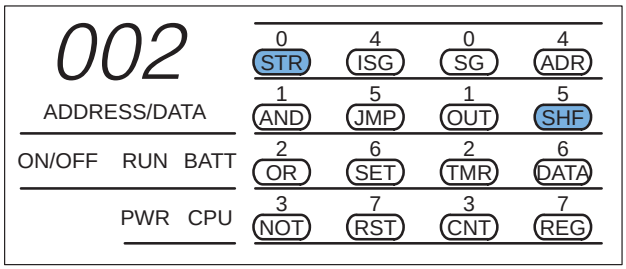
STR SHF 1 ENT

Data Display *before* ENT is pressed



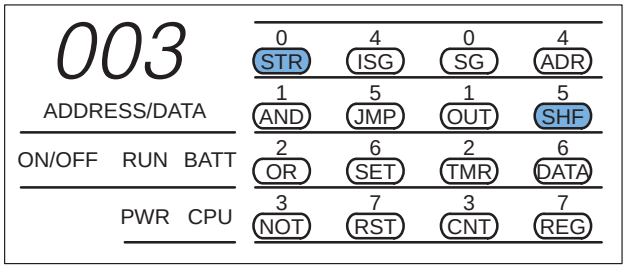
Enter the Clock input

STR SHF 2 ENT



Enter the Reset input

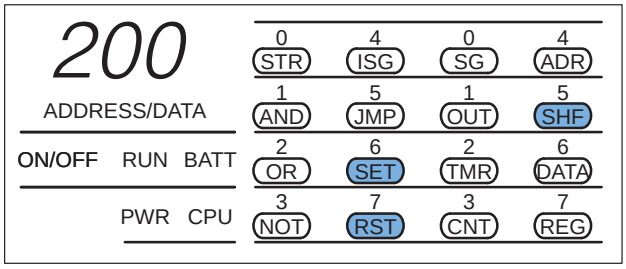
STR SHF 3 ENT



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