3 Entering Programs

In This Chapter. . . .

Entering Ladder Programs

Purpose of Section

This section will demonstrate how to use the Handheld programmer for mnemonic programming. The D2–HPP is commonly used for program changes and creating simple RLL programs. Again, for larger more complex PLC applications, we recommend ***Direct***SOFT™ ,our PC based programming software. Basic knowledge of boolean logic and PLC programming is helpful to better understand the examples provided. For more programming examples, you should reference the appropriate DL105 or DL205 User Manuals for details on specific instructions.

Handheld Programmer Key Sequences

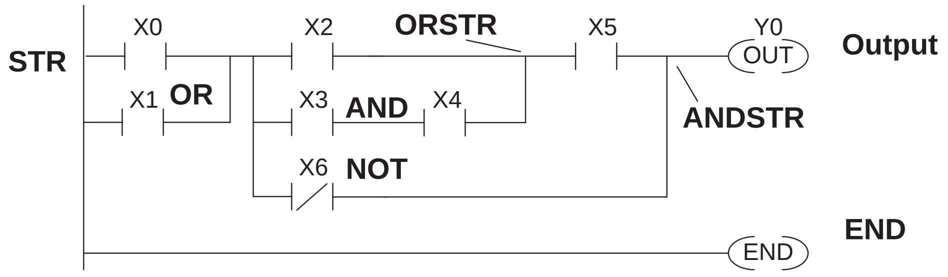
The Handheld programmer will buffers all keystrokes until the **ENT** (enter) key is pressed. The instruction syntax is checked for validity, when the enter key is pressed. If an instruction or data type is invalid an error message will be displayed. For a complete listing of error messages, please refer to Chapter 6.

Instruction Overview

The Handheld programmer only allows mnemonic instruction programming. A brief description of the most common used instructions are given below. The combination in which the mnemonics are entered will determine the Relay Ladder Logic (RLL) network structure and result.

- **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. (not available with DL 105)
- **OR** – Joins one element in parallel with a previous element or group of elements.
- **OR STR** – Joins parallel branches (not available with DL 105)
- **Output** – Each rung must have at least one output (Y, C, or box instruction)
- **NOT** – used with other instructions to utilize normally closed elements.
- **END** – All programs must contain an END statement.

All networks must begin with the STR (store) or STRN (Store Not) instruction and are then combined with other instruction entries. Networks must conclude with at least one output instruction (Y coil, C coil, or Box instruction). Below is a ladder network showing how various mnemonics instructions are combined in a single network.



Navigating the Program

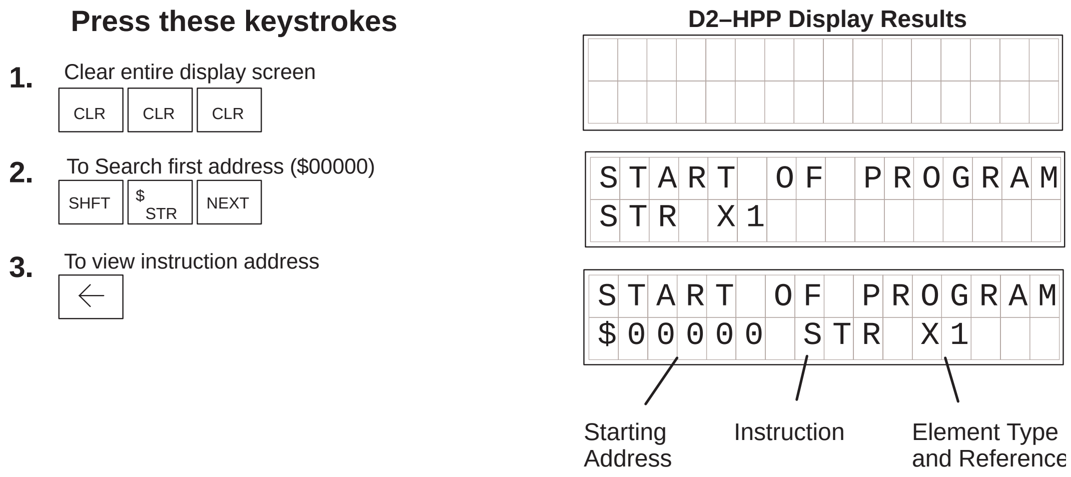
The Handheld programmer display screen, allows program instructions and their associated data to be viewed by the operator. All instructions are stored with a instruction addresses (*not* the same as rung addresses used in ***Direct***SOFT™). Newly entered instructions may be saved by pressing the **ENT** (enter) key.

Previous / Next Keys

Pressing the **NEXT** or **PREV** keys, allow scrolling through the mnemonic instructions in your program. It is not necessary to clear the display, before using these keys.

Starting at Address 0

When creating a new program, you should always begin the first program instruction at address zero ($00000). If you are in the Program mode, the 'START OF PROGRAM' message will appear, when positioned at the beginning address of your program. Use the left arrow **(**←**)** key to display the instruction addresses. To search the first address of your program, follow the example figure below.

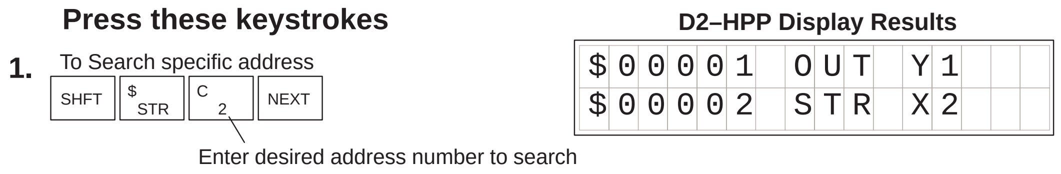

Press these keystrokes
D2–HPP Display Results
1. Clear entire display screen
CLR
CLR
CLR
2. To Search first address ($00000)
SHFT
$ STR
NEXT
3. To view instruction address
←
Starting Address
Instruction
Element Type and Reference

```
START OF PROGRAM
STR X1
```

```
START OF PROGRAM
$00000 STR X1
```

Searching for Addresses

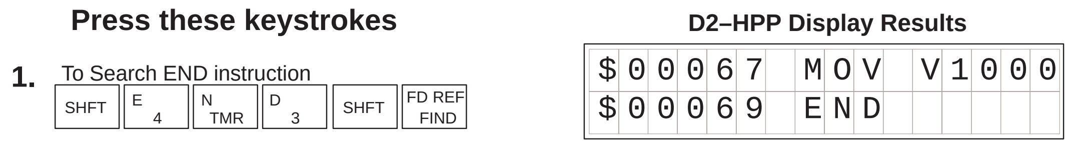
You may search for and display instructions in your program by entering the specific addresses number. The following examples below demonstrate how to search and find different program items. The entire display screen must be cleared before performing the following examples.


Press these keystrokes
D2–HPP Display Results
1. To Search specific address
SHFT
$ STR
C 2
NEXT
Enter desired address number to search

```
$00001 OUT Y1
$00002 STR X2
```

Searching for END Instruction

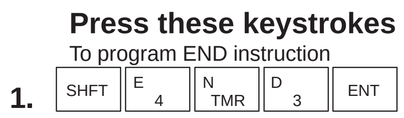
To search for the **END** command, follow the example below.



```
$00067 MOV V1000
$00069 END
```

Entering END Instruction

All programs require a **END** command. To enter the END instruction press the following keys.



Program Mode

The Program Mode is most commonly used to enter program instructions. After entering instructions, the changes are not executed until the CPU is placed in the Run mode. This will prevent unexpected machine operation which may be caused by changes which are performed.

With the Handheld programmer connected to the CPU, press the **MODE** key to select the Mode Change display. You may access the various modes by pressing the **NEXT** and **PREV** keys, while viewing the Mode Change display. To change to program mode follow the example below.

Press these keystrokes

1. To change modes.

MODE | NEXT | PREV | ENT | ENT

D2–HPP Display Results

```
*MODE CHANGE*
GO TO PGM MODE
```

Mode to Select

Entering a Simple Network

All programs begin starting at instruction address $00000. Use the **STR** (store) key to start programming your first network which contains a normally open contact (element) and output coil. The following will create a simple Store network.

Press these keystrokes

1. To enter Input contact.

$ STR | → | A 0 | ENT

2. Enter Ouput coil

GX OUT | → | B 1 | ENT

3. To type END instruction

SHFT | E 4 | N TMR | D 3 | ENT

D2–HPP Display Results

Begin Program entry here

```
START OF PROGRAM
STR X0
```

```
STR X0
NOP
```

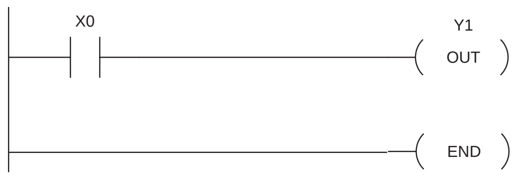
```
STR X0
OUT Y1
```

```
OUT Y1
NOP
```

```
OUT Y1
END
```

```
END
NOP
```

Equivalent Ladder Logic



Selecting Different Element Types

In the example above, you may press the **PREV** / **NEXT** keys, after the right (→) arrow key, to scroll the different element types available. While displaying the desired element type enter the element address, then press **ENT** (enter).

Now that you have completed your first mnemonic instruction network, please continue through each of the following program examples. Append each of the remaining examples to the first network. To continue adding the examples begin each new networks at the last instruction programmed (END command).

NOTE: Always ensure the last instruction of your program is the **END** command. If the END command is missing, the Handheld programmer will not allow you to change modes, or run the program. Error #4 ’No Program’ may be displayed.

Entering Normally Closed Elements

To enter a network which contains a normally closed contact, begin with the **STRN** (Store Not) instruction. The following example demonstrates how to enter a network using the STRN instruction.

Press these keystrokes

1. To enter normally closed input

SP STRN | → | C 2 | ENT

2. Enter Ouput coil

GX OUT | → | C 2 | ENT

3. END instruction

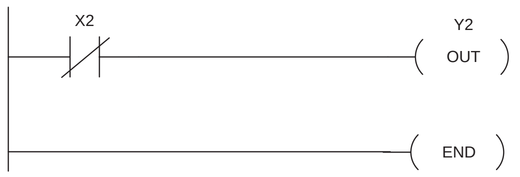
SHFT | E 4 | N TMR | D 3 | ENT

HPP Display Results

Continue program entry here

```
OUT  Y1
END

OUT  Y1
STRN X2

STRN X2
OUT  Y2

OUT  Y2
NOP

END
NOP
```

Equivalent Ladder Logic



Entering Series Elements

Some networks require more than one element on a branch, this is referred to as contacts in series. To program elements in series, you begin the network as before using the store (STR,STRN) instruction. The **AND** instruction is used to join two elements in series. The following example demonstrates how to enter two series contacts and a single output coil.

Press these keystrokes

1. To enter first Input contact

$ STR | → | B 1 | ENT

2. To enter second Input contact

$ STR | → | C 2 | ENT

3. To enter Ouput coil

GX OUT | → | C 2 | ENT

4. END instruction

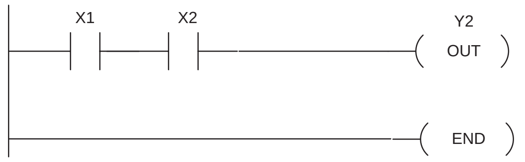
SHFT | E 4 | N TMR | D 3 | ENT

HPP Display Results

Continue program entry here

```
OUT  Y2
END

STR  X1
NOP

STR  X1
STR  X2

STR  X2
NOP

OUT  Y2
NOP

END
NOP
```

Equivalent Ladder Logic



Entering Parallel Elements

To program a network with parallel elements (more than one branch per network), you will use the **OR** instruction. Once again, you begin the network as before using the store (STR,STRN) instruction for first element, then continue the **parallel** branch with the to create and second element data. You join the two parallel rungs using the coil OUT command. Follow the example below to create the most simple form of a parallel branch network.

Press these keystrokes

D2–HPP Display Results

Continue program entry here

```
OUT   Y2
END
```

1. Enter first Input contact

$ STR	→	B 1	ENT

```
STR   X1
NOP
```

2. To start second branch and element

Q OR	→	C 2	ENT

```
STR   X1
OR   X2
```

3. To join parallel branch and enter Ouput coil

GX OUT	→	C 2	ENT

```
OR   X2
OUT   Y2
```

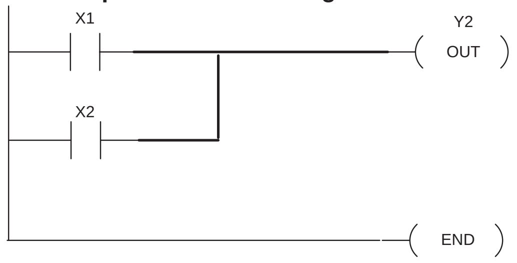
4. END instruction

SHFT	E 4	N TMR	D 3	ENT

```
OUT   Y2
NOP
```

```
END
NOP
```

Equivalent Ladder Logic



Later in this section, various examples using parallel element programming are provided. Branch programming examples require close observation of which order the mnemonic instructions are entered. If the instruction or data are not properly entered, the Handheld programmer display will response with a error message. Please take care and caution that the result of entering parallel logic does not present logical result problems.

Joining Series Elements in Parallel

Often it is necessary to program networks which contain parallel branches and series elements together to accomplish desired control. The **ORST (or store)** key allows you to program parallel branches with serial elements. The following example shows a simple network using the ORSTR instruction.

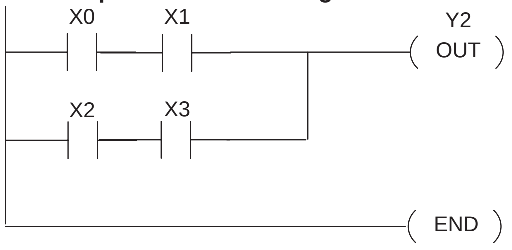
Press these keystrokes

D2–HPP Display Results

Continue program entry here

	Keystrokes	D2–HPP Display Results
		OUT Y2 / END
1. To enter Input conact X0	$ STR, →, A 0, ENT	OUT Y2 / STR X0
2. To enter second series conact	V AND, →, B 1, ENT	STR X0 / AND X1
3. To begin parallel branch and contact X2	MSTR, →, C 2, ENT	AND X1 / STR X2
4. To enter second parallel contact	V AND, →, D 3, ENT	STR X2 / AND X3
5. To OR parallel branches	ORST, →, C 2, ENT	AND X3 / ORSTR
6. Ouput coil	GX OUT, →, C 2, ENT	ORSTR
7. END instruction	SHFT, E 4, N TMR, D 3, ENT	OUT Y2 / END
		END / NOP

Equivalent Ladder Logic



Joining Parallel Branches in Series

The **ANDSTR** instruction joins one or more parallel branches which may be in series. The following example shows a simple network with parallel and series branches.

Press these keystrokes

1. Enter first Input conact

$ STR | → | A 0 | ENT

2. Enter second Input contact

$ STR | → | B 1 | ENT

3. Create branch and parallel contact

Q OR | → | C 2 | ENT

4. To join branch

L ANDST | ENT

5. Enter Ouput coil

GX OUT | → | D 3 | ENT

6. END instruction

SHFT | E 4 | N TMR | D 3 | ENT

HPP Display Results

Continue program entry here

```
OUT  Y2
END
```

```
STR  X0
NOP
```

```
STR  X0
STR  X1
```

```
STR  X1
OR  X2
```

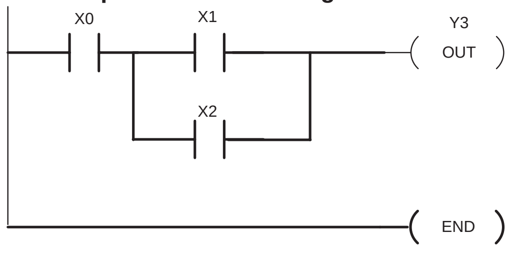
```
OR  X2
ANDSTR
```

```
ANDSTR
OUT  Y3
```

```
OUT  Y3
END
```

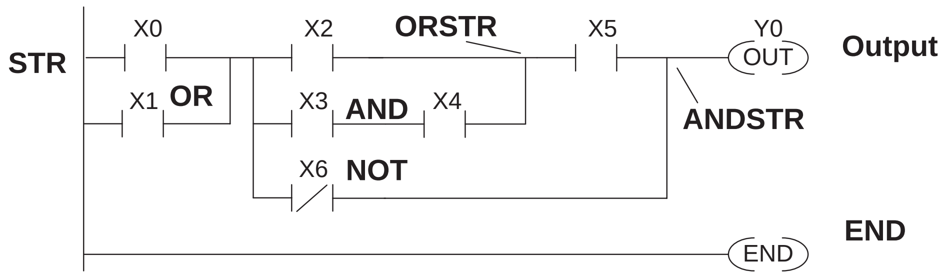
```
END
NOP
```

Equivalent Ladder Logic



Combination Networks

For combination networks, you may combine both the series elements and parallel branches. Combination logic allows you to solve almost any application problem. The following example is a ladder network, which is marked with **MNEMONIC** instructions and lists the order which the instructions may be entered.



Example Mnemonic Listing

ADDRESS	INSTRUCTION	DESCRIPTION
$00000	STR X0	Starts branch 1 with X0
$00001	OR X1	Joins X1 in parallel with X0
$00002	STR X2	Starts branch 2 with X2
$00003	STR X3	Starts branch 3 with X3
$00004	ANDN X4	Joins X4 (NOT) with X3
$00005	ORSTR	Joins branches 2 and 3
$00006	AND X5	Starts branch 4 with X5
$00007	ORN X6	Joins X6 (NOT) in parallel with X5
$00008	ANDSTR	Joins branches 4 and 5 with 1–3
$00009	OUT Y0	Stores the output and finishes the network
$00010	END	Ends the program

There are limits on how many boolean logic instructions can be combined in one network. The ***Direct***LOGIC™ CPU's use an 8 level stack to evaluate the various logic elements. The stack is a temporary storage area used to help evaluate the various logic combinations. Each time you enter a STR instruction, the instruction is placed on the top of the stack. All other instructions on the stack are pushed down one level. The And Store (ANDSTR) and Or Store (ORSTR) instruction combine levels of the stack when processed. Since the stack storage is eight levels, an error will occur if the CPU encounters a network that uses more than eight combined levels per network. For more details on the 8 level stack, please refer to section titled 'Programming Basics' in the DL105 or DL205 User Manuals.

Entering Timers and Counters

To enter a timer or counter, you also must prepare operand and enter preset values. This can be a constant value (K memory), or a V-memory location in the case of the DL240 CPU.

There are two methods of programming timers. You can have the timer with discrete timer control and status bits, or use comparative contacts, which enable at different time intervals during the control and status.

Timer Example Using Discrete Status Bits

The following timer example uses discrete status, with a preset of 3 seconds. If the timer is enabled for 3 seconds the status bit (T2) will turn ON. The timer will reset if X1 turns off, which in turn will resets the status bit (T2) off, and the accumulative value of the timer.

Press these keystrokes

HPP Display Results

Continue program entry here

```
OUT  Y3
END
```

1. To enter the first Input contact.

$ STR | → | B 1 | ENT

```
STR  X1
TMR  T2  K30
```

2. Enter the timer reference and preset value.

N TMR | → | C 2 | → | D 3 | A 0 | ENT

```
TMR  T2  K30
TMR  T2  V2300
```

3. Begin new network and Timer status element.

$ STR | → | SHFT | T MLR | C 2 | ENT

```
STR  T2
OUT  Y0
```

4. To enter Output coil.

GX OUT | → | A 0 | ENT

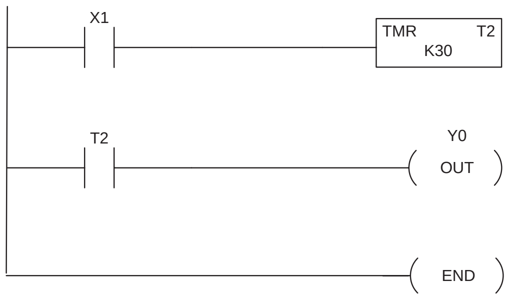
```
OUT  Y0
END
```

5. END instruction

SHFT | E 4 | N TMR | D 3 | ENT

```
END
NOP
```

Equivalent Ladder Logic



Accumulating Timers & Counters

The Accumulating Timer which has additional lines connected to the timer instruction, can allow separate **start** and **reset** elements. All input element contacts are entered before the timer or counter instruction. The timer inputs may be of various types, e.g. timer status (T#), control relays (CR), etc. To scroll through the different operand data types, while programming the example below, press the **NEXT** key after the arrow → key is pressed. Although the Handheld programmer may allow you to select various data types, please refer to the DL105 or DL205 User Manual according to which CPU you are programming. For example, the DL240 will allow V–Memory registers for timer presets, where as the DL130 and DL230 will only allow K–Memory to be loaded as presets.

Entering Accumulating Timers (two Inputs)

This example demonstrates how to program a Accumulating Timer with a preset of 5 seconds. The timer discrete status bit (T0) contact will energize when the timer has timed for 5 seconds. The timer will reset when input X1 turns on, turning the timer discrete status bit off and resetting the timer current (timed) value to zero.

Press these keystrokes

1. To enter timer start Input conact
   [$ STR] [→] [A 0] [ENT]
2. To enter the reset Input conact
   [$ STR] [→] [B 1] [ENT]
3. Select Timer type and reference number
   [N TMR] [SHFT] [A 0] [→] [A 0]
4. Enter Timer preset
   [→] [F 5] [A 0] [ENT]
5. Begin new network with Timer status bit contact
   [$ STR] [→] [SHFT] [T MLR] [A 0] [ENT]
6. Enter Output Coil Y0
   [GX OUT] [→] [A 0] [ENT]
7. Enter END
   [SHFT] [E 4] [N TMR] [D 3] [ENT]

HPP Display Results

Continue program entry here

```
OUT  Y0
END
```

```
oUT  Y0
STR  X0
```

```
STR  X0
STR  X1
```

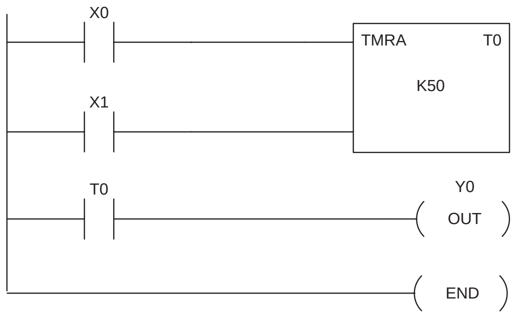
```
STR  X1
TMRA  T0  K50
```

```
STR  T0
OUT  Y0
```

```
OUT  Y0
END
```

```
END
NOP
```

Equivalent Ladder Logic



Entering Relational Contacts

Relational contacts may be used to compare various types of information. For example, you may want to compare the current value of a timer with a constant value (K–Memory) or a V-memory register value. There are several types of compare operations that can be programmed, such as, less than, greater than, etc. See the DL105 or DL205 User Manual for more details on all relational contact instructions. The following example demonstrates how to program a greater than or equal to relational contact.

Press these keystrokes

1. Enter first compare reference
[$ STR] [→] [SHFT] [T MLR] [A 0]

2. Select constant reference to compare
[→] [B 1] [A 0] [A 0] [ENT]

3. Enter Ouput coil Y0
[GX OUT] [→] [A 0] [ENT]

4. Begin second Compare network and reference
[$ STR] [→] [SHFT] [T MLR] [A 0]

5. Enter compare V–Memory reference
[→] [SHFT] [V AND] [C 2] [A 0] [A 0] [A 0]
[ENT]

6. Enter Output Coil Y1
[GX OUT] [→] [B 1] [ENT]

D2–HPP Display Results

Continue entry here

```
OUT  Y0
END
```

```
OUT  Y0
STR  T0  K100
```

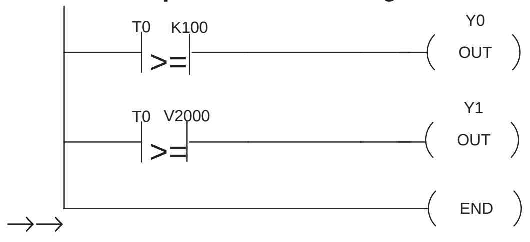
```
STR  T0  K100
OUT  Y0
```

```
OUT  Y0
STR  T0  V2000
```

```
STR  T0  V2000

```

```
STR  T0  K100
OUT  Y0
```

Equivalent Ladder Logic

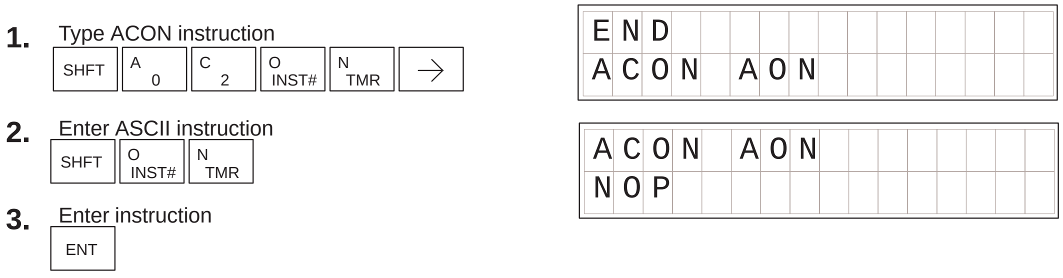


Entering ASCII Characters

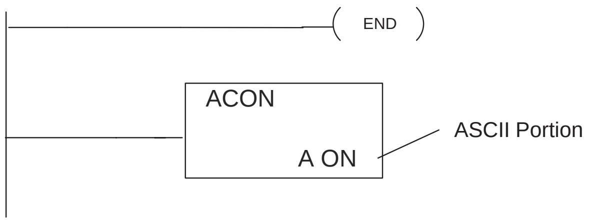
The DL105 and DL205 allow you to enter ASCII characters as part of the ACON instruction used for messages. An overview of the ACON instruction is provided in Chapter 6 of this manual. The example below shows the keystrokes used to enter the ASCII portion of the instruction with the Handheld programmer.

Press these keystrokes

D2–HPP Display Results



Equivalent Ladder Logic



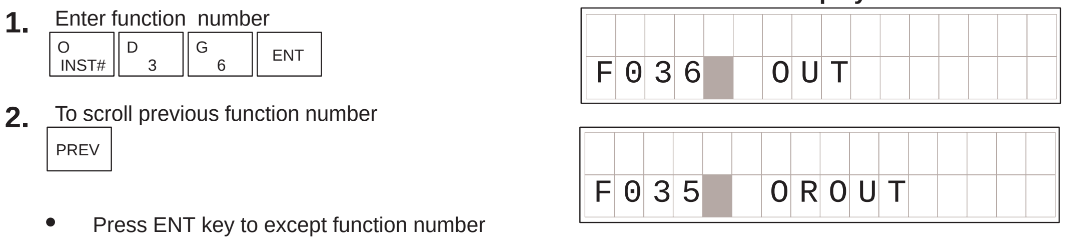
NOTE: More detailed information on the ACON instruction may be referenced in the DL105 and DL205 User Manuals.

Using the INST # key

Some mnemonic instructions may be entered by using a instruction number. The instruction number may also be referred to as *function* number. Use the Handheld programmer **INST#** key to begin the function number entry. If known, you may enter the specific instruction number , or scroll through available function numbers by pressing the **PREV**/**NEXT** keys. The following example demonstrates using the instruction number function.

Press these keystrokes

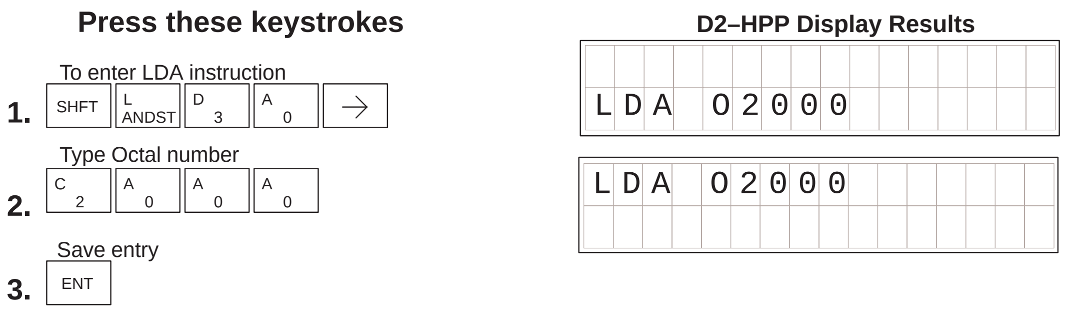
D2–HPP Display Results

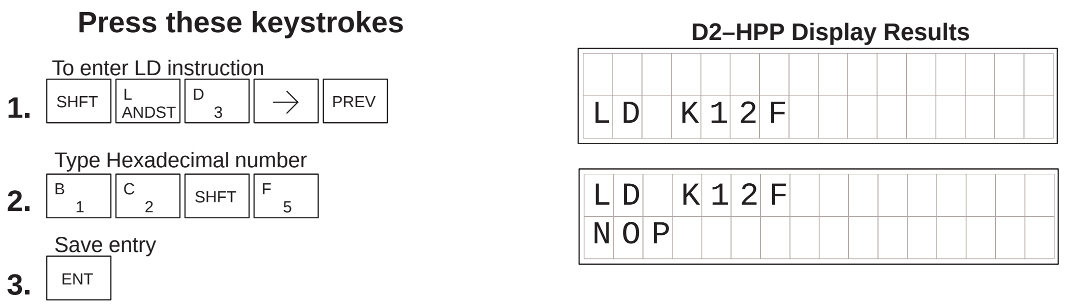


- Press ENT key to except function number
- Press the CLR key to exit

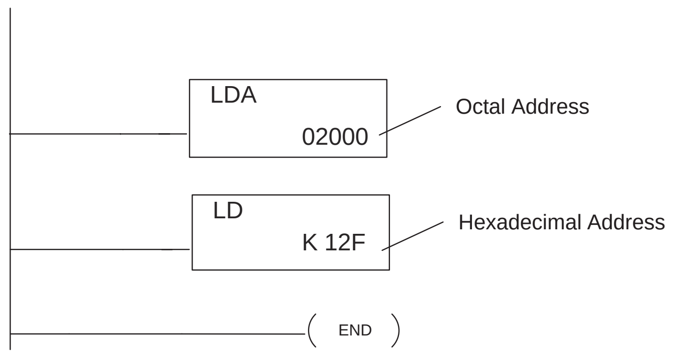
Entering Octal and Hex Numbers

For some instructions entries , special number formats are used for reference data. For example, the LDA (Load Address) instruction requires an octal number for the address reference. Also, you may want to load a hexadecimal value into the accumulator. The following example demonstrates how to enter octal and hexadecimal numbers using the Handheld programmer. For specific instruction information and optional number formats, please refer to the DL 105 and DL205 User Manuals.





Equivalent Ladder Logic

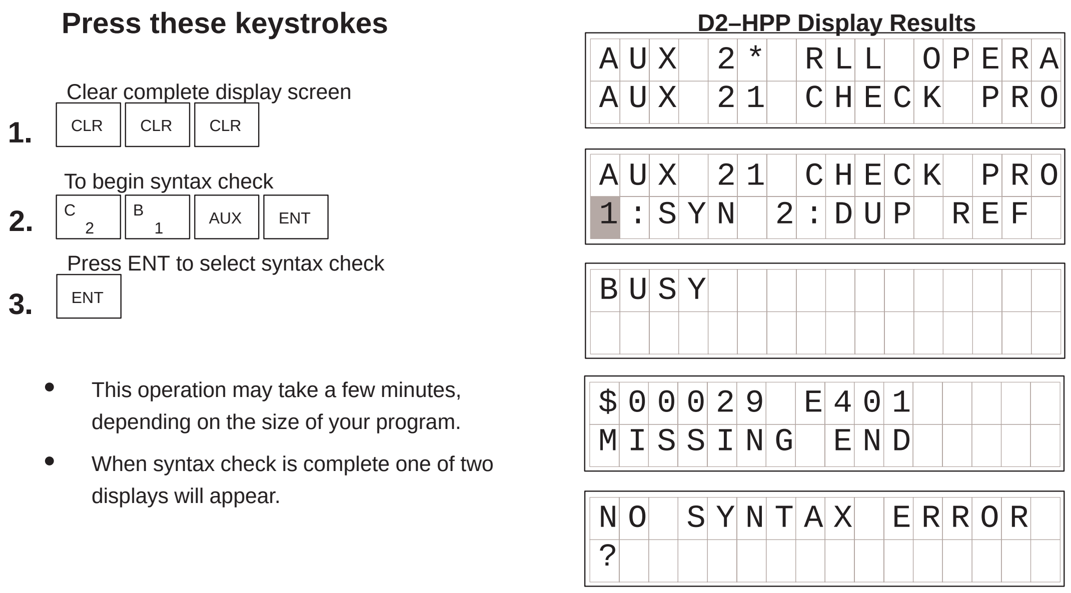


Checking for Program Errors

Error Checking The Handheld programmer may also check your program for errors. You may choose two different types of program error checking.

- Syntax errors check
- Duplicate References check

Syntax Check Use the **AUX 21** function, to select the 'CHECK PROGRAM' operation. Operation 1 performs a syntax check on the entered program logic. The following figure demonstrates how to access the Syntax check operation.



```
AUX 2* RLL OPERA
AUX 21 CHECK PRO
```

```
AUX 21 CHECK PRO
1:SYN 2:DUP REF
```

```
BUSY
```

```
$00029 E401
MISSING END
```

```
NO SYNTAX ERROR
?
```

- This operation may take a few minutes, depending on the size of your program.
- When syntax check is complete one of two displays will appear.

Each error is labeled with an Error Code when displayed. Please refer to Chapter 6 for a complete listing of Error Code numbers. Upon receiving an error message, attempt correcting the problem and continue running the Syntax check until the message 'NO SYNTAX ERROR' appears.

Duplicate Reference Check

You may also use Check Program, Option 2, for multiple uses of the same output coil. The following example below demonstrates how to access AUX 21 and perform a Duplicate Reference check.

Press these keystrokes

D2–HPP Display Results

1. Clear complete display screen

CLR CLR CLR

```
AUX 2*  RLL OPERA
AUX 21 CHECK PRO
```

2. To begin syntax check

C 2 B 1 AUX ENT

```
AUX 21 CHECK PRO
1:SYN 2:DUP REF
```

3. Position cursor on number 2 for DUP REF check

→ ENT

```
BUSY
```

- When Program Check is complete one of these two displays will appear.

Error Display (example)

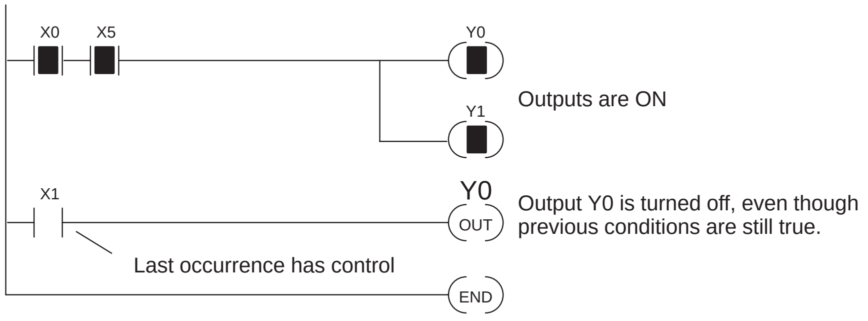
```
$00012 E471
DUP COIL REF
```

No Duplicate Reference display

```
NO DUP REFS
?
```

If a Duplicate Reference error occurs, please refer to Chapter 6 for a complete listing of Error Code numbers. You should correct the problem and continue running the Duplicate Reference check until the message NO DUP REFS appears.

NOTE: You can use the same coil in more than one location. However, the last occurrence of the element will take priority. Consider the following example.


X0
X5
Y0
Y1
Outputs are ON
X1
Y0
OUT
Output Y0 is turned off, even though previous conditions are still true.
Last occurrence has control
END