3

Entering Programs

In This Chapter. . . .

— Entering Simple Ladder Programs
— Checking for Program Errors

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 DL405 instructions. Once you are familiar with the basic keystrokes, you should use the DL405 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 it has been entered correctly. If the instruction was 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 you must become familiar with to enter programs with the Handheld.

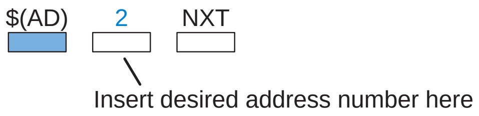
- STR – Stores a normally open element and indicates the beginning of a rung or network.
- AND – Joins one element (such as a contact) in series with another element or group of elements.
- AND STR – Joins a group of elements in series with another group of elements.
- OR – Joins one element in parallel with a previous element or group of elements.
- ORSTR – Joins parallel branches (each branch must begin with a STR instruction)
- Output – 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.

Traversing the Program

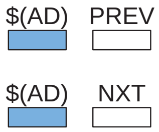
The instructions and associated data are located at program addresses (not the same as rung addresses used in ***Direct***SOFT). You may access an instruction by going directly to the instruction address or you may use the next and previous functions to toggle through the program addresses.

NOTE: Before using the first two below methods shown below ensure the display is clear, otherwise the entry will not be accepted.

Accessing Addresses



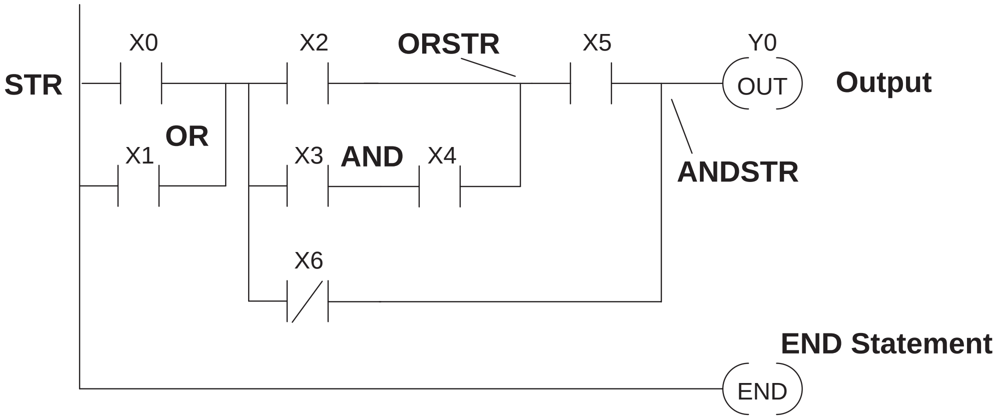
Address Previous / Next Functions



Previous / Next Keys

When using only the previous and next keys to toggle through the program addresses, it is not necessary to clear the display.

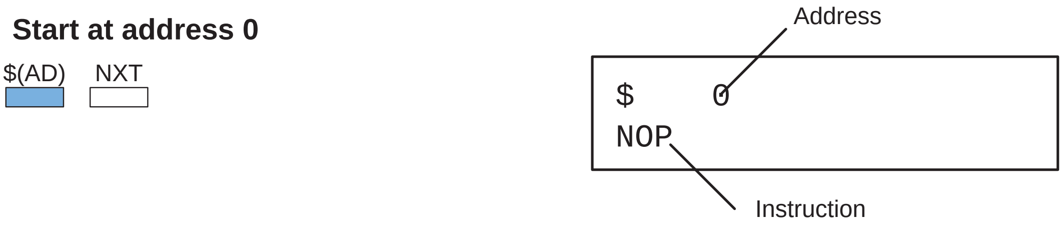
The following diagram shows a typical network and how each of these elements are used.



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. (The remaining examples will not show this display, but the keystrokes are required.)

Start at address 0

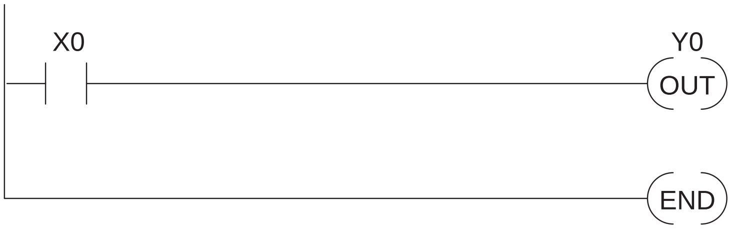


```
$    0
NOP
```

Once you're at address 0, you can start entering a program.

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.



Enter the contact

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the output coil

OUT Y(OUT) 0 ENT

```
$    1
OUT Y0
```

Enter the END statement

END ENT

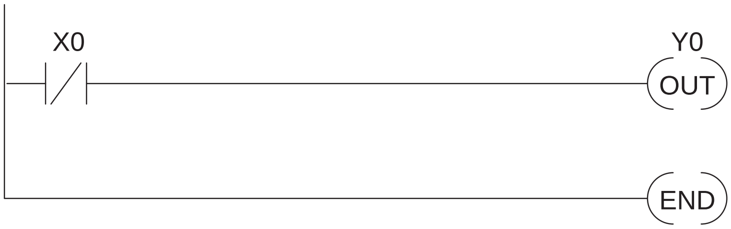
```
$    2
END
```

The example shows an X input contact and a Y output coil. If you examine the Handheld keyboard, you will notice specific keys for the other available data types.

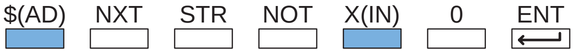
C(CR) — Control Relay SPCL — Special Relay S(SG) — Stage

Entering Normally Closed Elements

Normally closed elements are entered with the STRN (Store Not) instruction. The following example shows a simple rung with a normally closed contact.



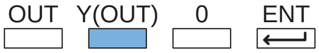
Enter the contact



Starting at Address 0

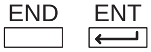
```
$    0
STRN X0
```

Enter the output coil



```
$    1
OUT Y0
```

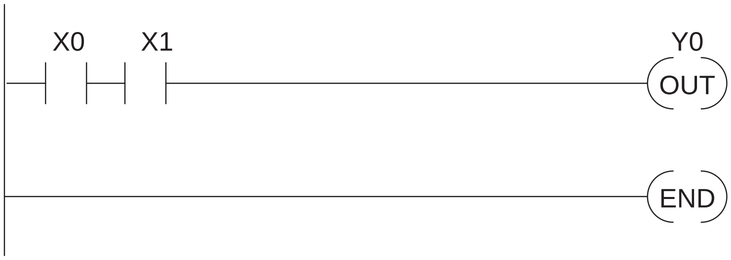
Enter the END statement



```
$    2
END
```

Entering Series Elements

You must start the first rung with a STR instruction, since it contains more than one element and since it is also the beginning of the network. The AND instruction joins the series contacts. The following example shows how to enter two series contacts and a single output coil.



Enter the first contact

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the second contact

AND X(IN) 1 ENT

```
$    1
AND X1
```

Enter the output coil

OUT Y(OUT) 0 ENT

```
$    2
OUT Y0
```

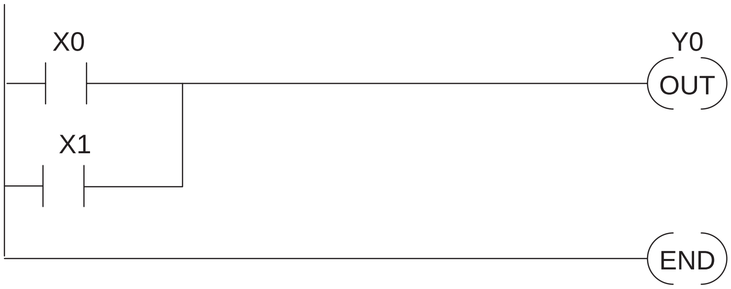
Enter the END statement

END ENT

```
$    3
END
```

Entering Parallel Elements

You must start the first rung with a STR instruction, since it contains more than one element and since it is also the beginning of the network. The OR instruction joins the parallel contacts. The following example shows how to enter two parallel contacts and a single output coil.



Enter the first contact

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the second contact

OR X(IN) 1 ENT

```
$    1
OR X1
```

Enter the output coil

OUT Y(OUT) 0 ENT

```
$    2
OUT Y0
```

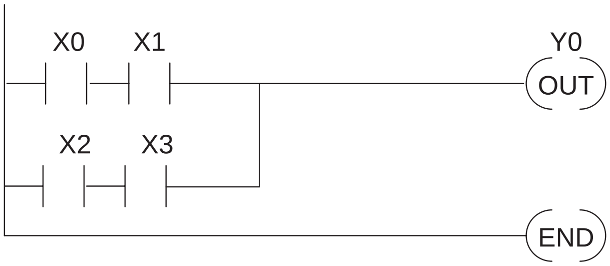
Enter the END statement

END ENT

```
$    3
END
```

Joining Series Branches in Parallel

Quite often it is necessary to joins one or more branches, of serial 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

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the second contact

AND X(IN) 1 ENT

```
$    1
AND X1
```

Start the next rung

STR X(IN) 2 ENT

```
$    2
STR X2
```

Add the next contact

AND X(IN) 3 ENT

```
$    3
AND X3
```

Join the branches

OR STR ENT

```
$    4
ORSTR
```

Enter the output coil

OUT Y(OUT) 0 ENT

```
$    5
OUT Y0
```

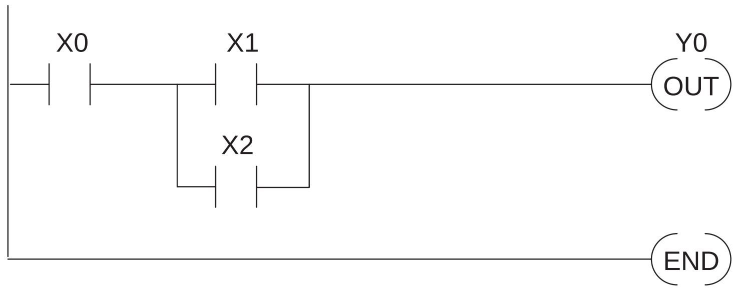
Enter the END statement

END ENT

```
$    6
END
```

Joining Parallel Branches in Series

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



Enter the first contact

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the second contact

STR X(IN) 1 ENT

```
$    1
AND X1
```

Enter the parallel contact

OR X(IN) 2 ENT

```
$    2
OR X2
```

Join the parallel branch

AND STR ENT

```
$    3
ANDSTR
```

Enter the output coil

OUT Y(OUT) 0 ENT

```
$    4
OUT Y0
```

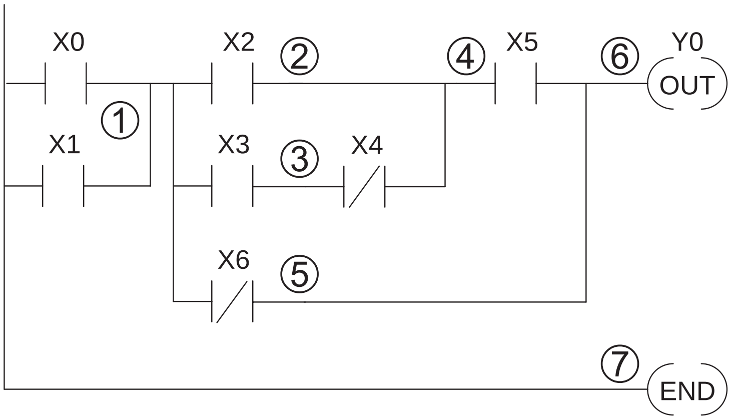
Enter the END statement

END ENT

```
$    5
END
```

Combination Networks

You can combine the various types of series and parallel branches to solve most any application problem. The following example shows a simple combination network.



①. Start the network

Starting at Address 0

$(AD) NXT STR X(IN) 0 ENT

```
$    0
STR X0
```

OR X(IN) 1 ENT

```
$    1
OR X1
```

②. Start branch 2

STR X(IN) 2 ENT

```
$    2
STR X2
```

③. Start branch 3, join with branch 2

STR X(IN) 3 ENT

```
$    3
STR X3
```

AND NOT X(IN) 4 ENT

```
$    4
ANDN X4
```

OR STR ENT

```
$    5
ORSTR
```

④. Add branch 4

AND X(IN) 5 ENT

```
$    6
AND X5
```

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

OR NOT X(IN) 6 ENT

```
$    7
ORN X6
```

AND STR ENT

```
$    8
ANDSTR
```

⑥. Add the output

OUT Y(OUT) 0 ENT

```
$    9
OUT Y0
```

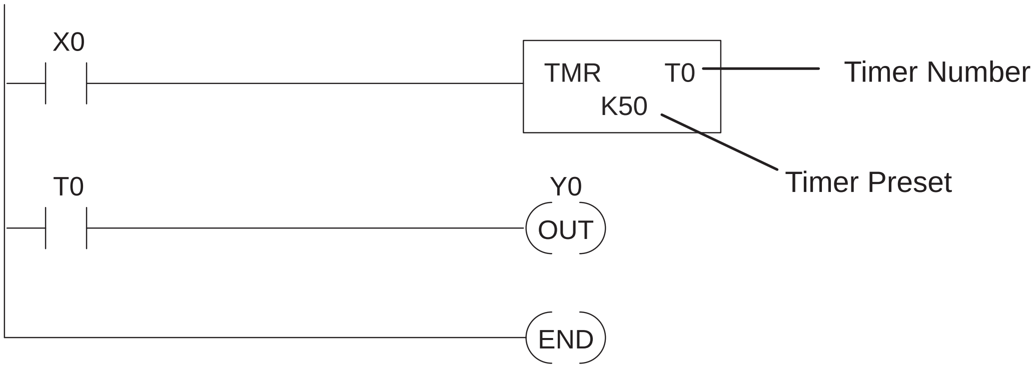
⑦. Enter the END statement

END ENT

```
$    10
END
```

Entering Timers and Counters

To enter a timer or counter, you also have to enter a preset value. This can be a constant, entered with the ***K(CON)*** key, or a V-memory location, entered with the ***V*** instruction reference key. This example shows how to enter these constants.



Enter the first contact

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the timer

with a constant

TMR TMR 0 K(CON) 5 0 ENT

```
$    1
TMR T0 K50
```

or

with a V-memory preset

TMR TMR 0

V 3 5 0 0 ENT

```
$    1
TMR T0 V3500
```

Enter the timer contact

STR TMR 0 ENT

```
$    3
STR T0
```

Enter the output

OUT Y(OUT) 0 ENT

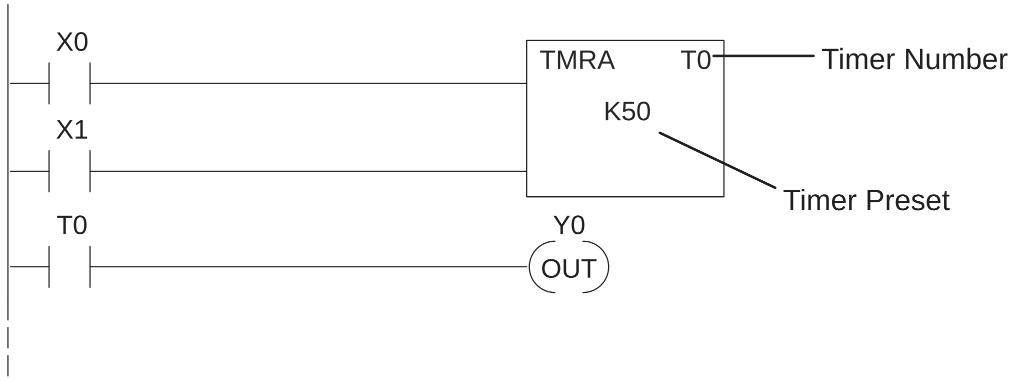
```
$    4
OUT Y0
```

End the program

END ENT

```
$    5
END
```

Some timers are accumulating timers and have reset lines. Also, two there are two types of counters that have multiple input lines. The following example shows how to use the Handheld to enter the additional input lines. Note that all input line contacts are entered before the actual instruction is entered. This is true for both timers and counters.



Enter the timer enable contact

$(AD) NXT STR X(IN) 0 ENT

Starting at Address 0

```
$    0
STR X0
```

Enter the timer reset contact

STR X(IN) 1 ENT

```
$    1
STR X1
```

Enter the timer

TMR SHFT A SHFT TMR 0
KCON 5 0 ENT

```
$    2
TMRA T0 K50
```

Enter the timer contact

STR TMR 0 ENT

```
$    5
STR T0
```

Enter the output

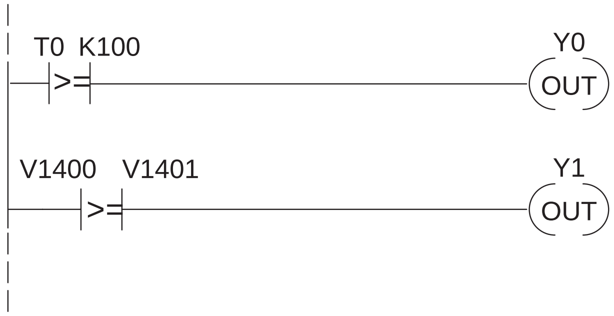
OUT Y(OUT) 0 ENT

```
$    6
OUT Y0
```

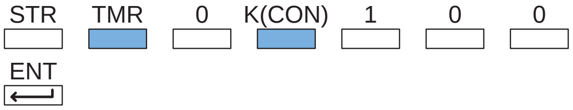
Entering Relational Contacts

Relational contacts allow you to quickly and easily compare various types of information. For example, you may want to compare the current value of a timer with a constant or a value contained in a V-memory location. Or, you could quickly compare two V-memory locations. There are several types of comparisons that can be made, less than, greater than, etc. See the DL405 User Manual for details on relational contacts.

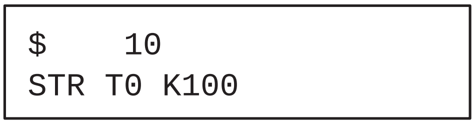
The following example shows how to enter a relational contact.



Timer contact with a constant

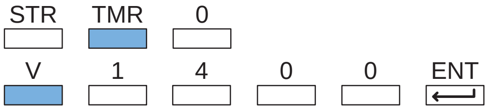


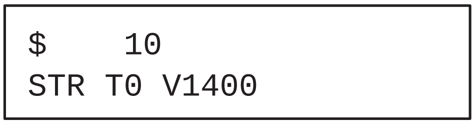
Address 10 shown as example



```
$    10
STR T0 K100
```

Timer contact compared to a V-memory location

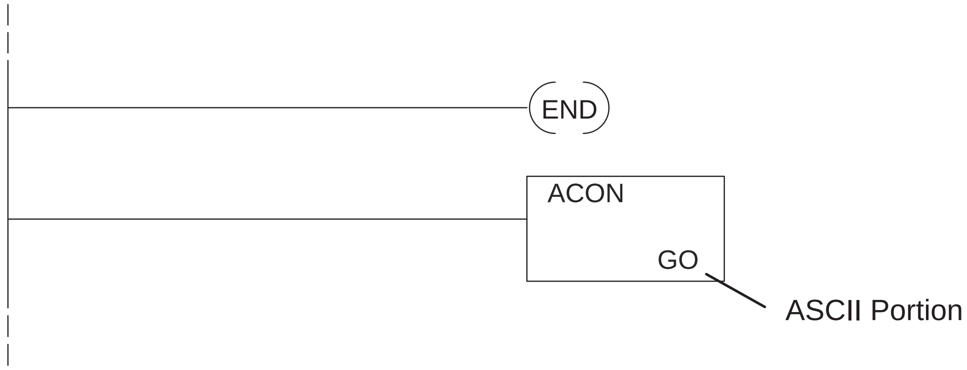




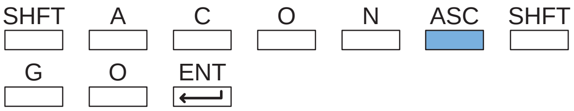
```
$    10
STR T0 V1400
```

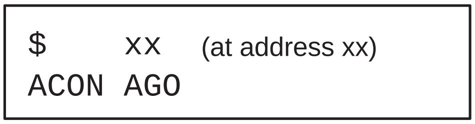
Entering ASCII Characters

Some DL405 instructions, like the ACON instruction in the DL440, allow you to enter ASCII characters as part of the instruction. (An overview of the ACON instruction is provided in Chapter 6. Also, the DL405 User Manual provides detailed information.) Here is a simple example that shows how to enter the ASCII portion of the instruction with the Handheld Programmer.



ACON ASCII example

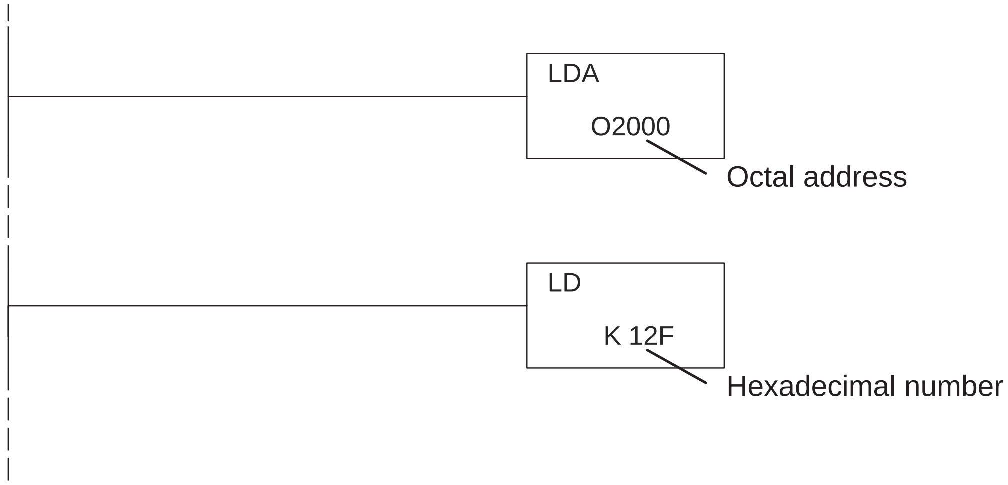




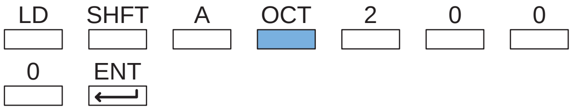
```
$       xx   (at address xx)
ACON AGO
```

Entering Octal and Hex Numbers

Some DL405 instructions require different number formats as part of the instruction. 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 shows you how to enter octal and hex numbers with the Handheld Programmer. (See the DL405 User Manual for details on the actual instructions.)

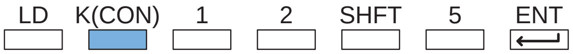


LDA Octal example



```
$      XX    (at address xx)
LDA O2000
```

LD Hexadecimal example



(SHFT 5 is hex F)

```
$      XX    (at address xx)
LD K12F
```

Checking for Program Errors

Error Checking

The Handheld automatically checks for errors during program entry. However, there may be occasions when you want to check a program that is already in the CPU. There are two types of checks available.

- Syntax
- Duplicate References

Syntax Check

You can use AUX 21, CHECK PROGRAM to check the program syntax. This check will find a wide variety of programming errors. The following example shows how to access AUX 21.

Use AUX 21 to perform syntax check

AUX 2 1 ENT ENT

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

Select syntax check

1 ENT (This may take a minute or so.)

```
BUSY
```

One of two displays will appear

Error Display (example)

```
$ 8 E401 MISSING END
TMRA   T 002   K00050
```

(shows location in question)

Syntax OK display

```
NO SYNTAX ERROR
?
```

If you get an error, see the Error Codes Section for a complete listing of programming error codes. Correct the problem and continue running the Syntax check until the NO SYNTAX ERROR message appears.

Duplicate Reference Check

You can use AUX 21, CHECK PROGRAM to check for multiple uses of the same output coil. The following example shows how to access AUX 21.

Use AUX 21 to perform syntax check

AUX 2 1 ENT ENT

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

Select Duplicate Reference check

2 ENT (This may take a minute or so.)

```
BUSY
```

One of two displays will appear

Error Display (example)

```
$ 12 E471 DUP COIL REF
OUT    Y 0000
```

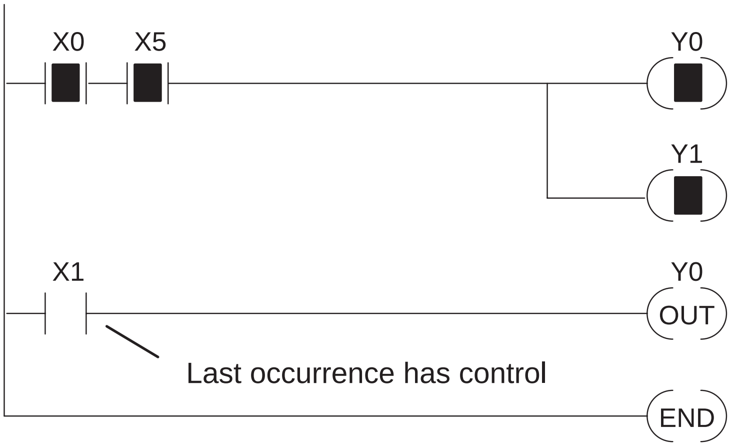
(shows location in question)

Syntax OK display

```
NO DUP REFS
?
```

If you get a Duplicate Reference error, see Error Codes Section for a complete listing of programming error codes. Correct the problem and continue running the Duplicate Reference check until the NO DUP REFS message 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.



Outputs are ON

Y0 is turned off, even though previous conditions are still true.

Last occurrence has control