4

Message Display Mode

In This Chapter. . . .

Overview

Feature List

The Message Display Mode of the DV-1000 provides a *monitoring* function. It supports System Messages from the CPU (no setup parameters required).

- Error Messages
- FAULT instruction messages from ladder logic

The DL240, DL250, DL350, DL440 and DL450 CPUs record a message log of 16 error messages and 16 FAULT instruction messages, each with a time/date stamp. The DV-1000 provides access to these message logs from its keypad.

With Message Display Mode you can create your own displays (User Messages) using setup parameters and ladder programming.

- Create text or numeric messages
- Mix text and numeric messages
- Implement blinking (flashing) characters, for alarm conditions, etc.
- Create multiple screens that let you switch from one to another
- Display signed values (+/– sign follows actual value)
- Embed a decimal point in a number
- Embed the time and date in a message
- Use the extended ASCII character set for effects such as analog bar graphs, or other symbolic information
- Generate long messages that continuously scroll through the display

Message Priority

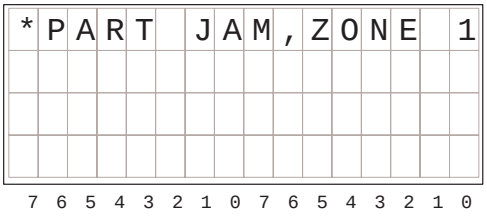
In Message Display Mode, display output can come from three sources: (System) Fault Messages and Error Messages, and (DV-1000) User Messages. To the right are examples of each.

In normal conditions, user messages are the default display. However, system messages have display priority. For example, if the battery is removed the error message “E042 NO CPU BATT” temporarily replaces the user message.

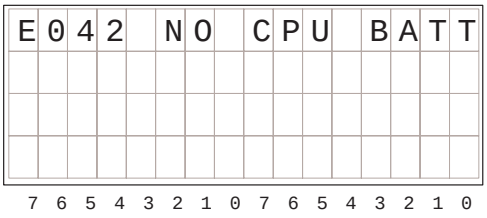
When a Fault Message Box is executed, its display output has priority over both system error messages and user messages. When the cause of a higher priority message ceases, the next lower priority message reappears automatically. The **prioritized order** is:

1. System Fault Message (highest)
2. System Error Message
3. User Messages (lowest)

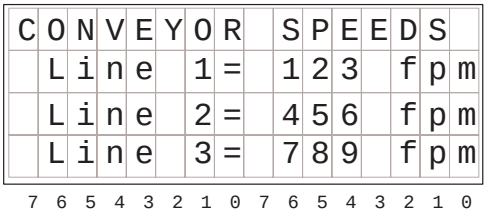
System Fault Messsage



System Error Message



User Message

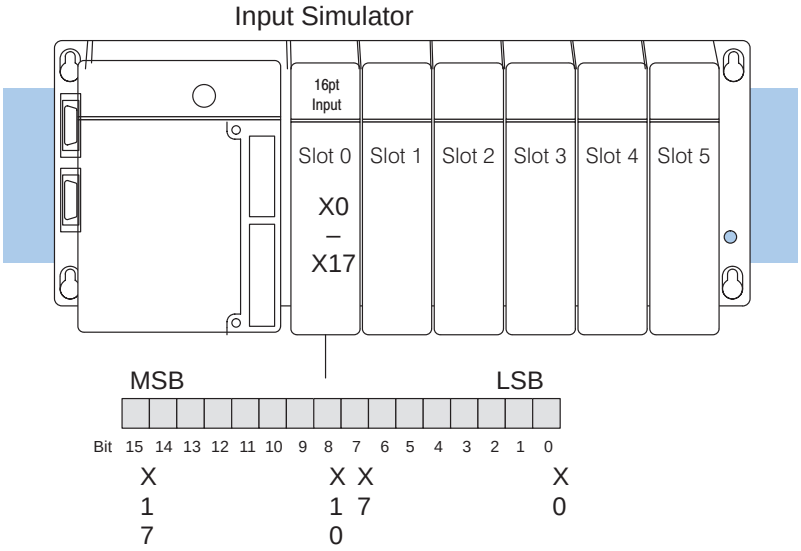


System Requirements for Example Programs

A quick way to learn how the DV-1000 works is to load an example program related to your application, and begin modifying it to do what you want it to do. Example ladder programs in this chapter occasionally require an external input to cause the display to change in some particular way. The programs require up to two inputs: X0 and X1. So, this is a good time to prepare your system to run the example programs as you arrive at each one.

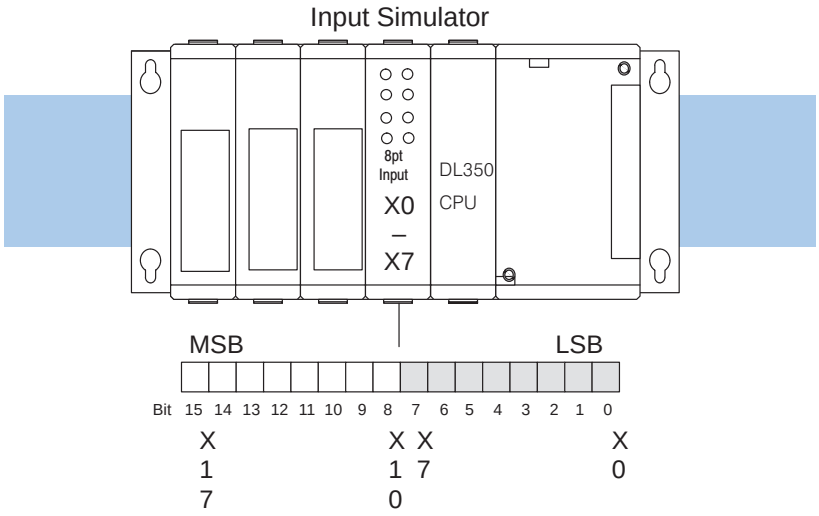
DL405 Requirements

If you are using a DL405 system, we recommend using the input simulator module. If you locate it in the base as the closest input module to the CPU, it will log in as X0 through X17 as shown in the following diagram:



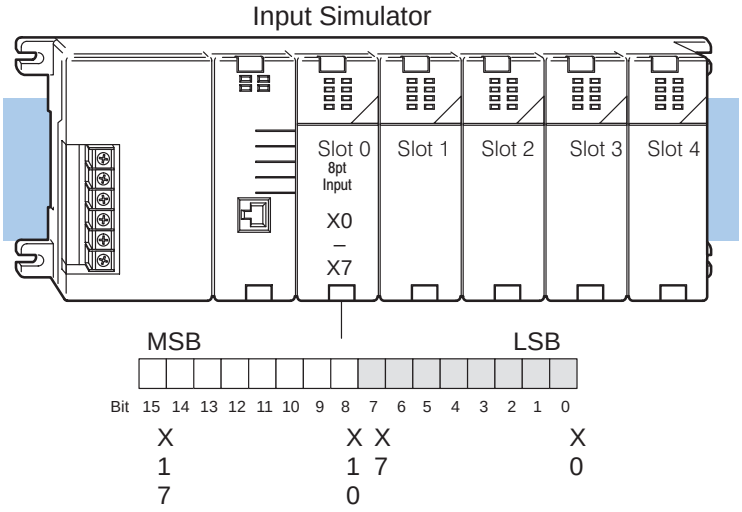
DL350 CPU Requirements

If you are using a DL305 sytem with a DL350 CPU, we recommend using the input simulator. If you locate it in the base as the closest input module to the CPU, it will log in as X0 through X7 as shown in the following diagram:



DL205 Requirements

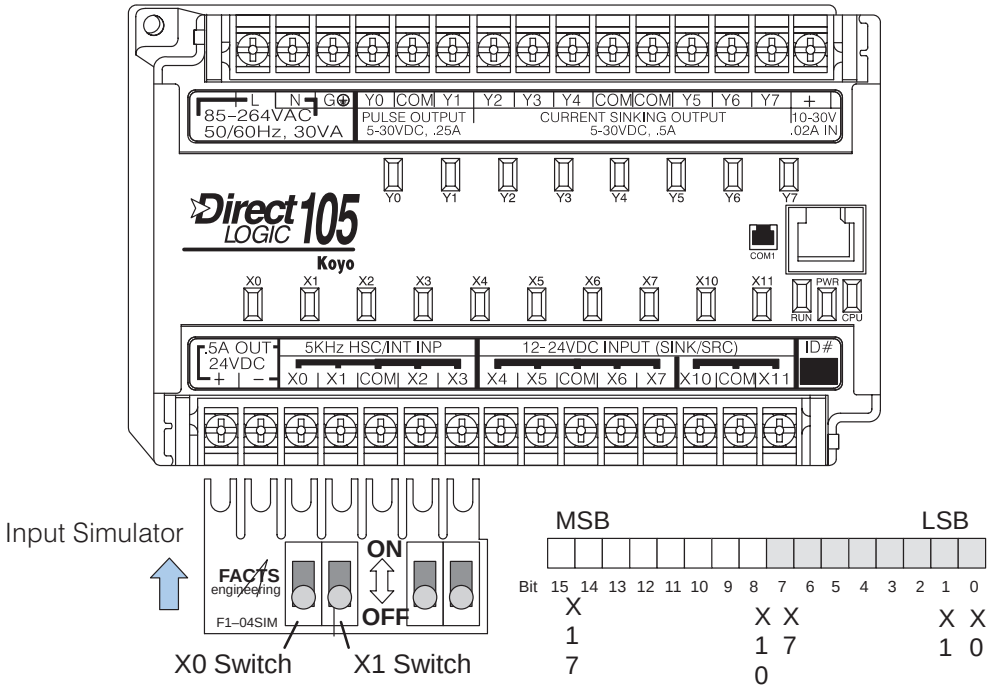
If you are using a DL205 system, we recommend using the input simulator. If you locate it in the base as the closest input module to the CPU, it will log in as X0 through X7 as shown in the following diagram:



DL105 Requirements

If you are using a DL105 system with DC inputs, we recommend using the input simulator, shown below, which provides four input switches for inputs X0 through X3. DC-powered versions need two wires from the power input to the two left-most terminals on the simulator. The input simulator will not work in DL105s with AC type inputs. See the DL105 User Manual (D2-USER-M) for more information on the input simulator.

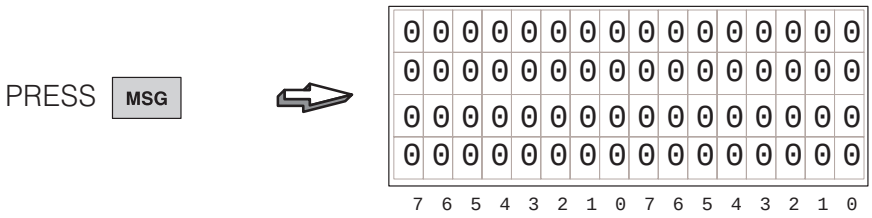
NOTE: When starting ***Direct***SOFT, be sure to initialize the CPU scratchpad or some example programs will not work (inputs X0 and X1 will be inactive). From **PLC** menu, choose **Setup**, then **Initialize Scratchpad.**



System Messages

Selecting Message Mode

You may select Message Display Mode at any time from the keypad by pressing the **Message** Key (unless the keypad is in Bit Control Mode). User messages require setup parameters to tell the DV-1000 where to find its message data in V-memory. If the parameters are not set and V-memory contains all zeros, the display below will appear.

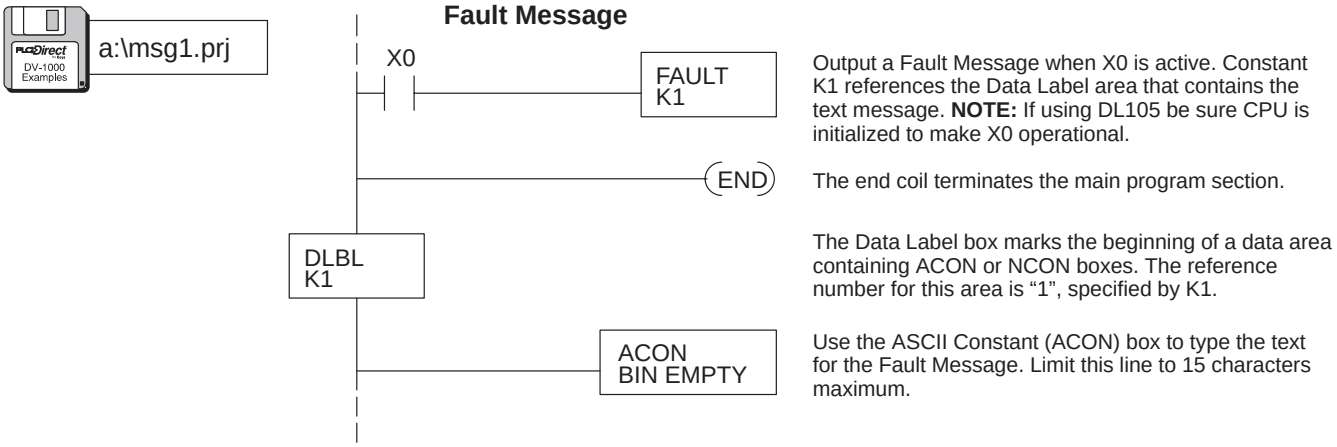


The next section shows how to program the setup parameters and generate your own message. However, we first discuss the higher priority system messages.

System Errors and the Fault Message instruction box in ladder logic generates messages for the outside world. They can be viewed using ***Direct***SOFT, a Hand-held Programmer, or the DV-1000. The DL240, DL250, DL350, DL440 and DL450 CPUs retain a log of up to 16 system error messages and a log of up to 16 fault messages which may also be viewed.

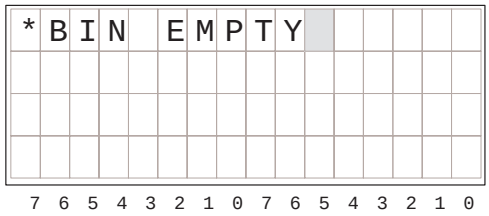
Fault Instruction Message

The following ladder program example will send a Fault Message to the display as long as contact X0 is on. When X0 goes off, the display returns to the user-generated message. Although the FAULT instruction can display 23 characters on a Hand-Held Programmer, **limit the number of characters to 15 for the DV-1000 (top line).**



Load the program above and test it by turning on X0. While X0 is on, the fault message will be displayed. When it is turned off, the fault message is removed and the original message display returns.

Fault Instruction Messsage

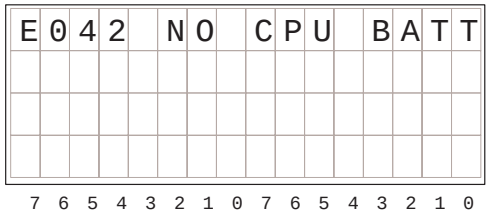


NOTE: The ladder program above generates fault messages which immediately appear on the display. However, note that the fault instruction makes an entry into the fault message log on each scan the fault box is active. Therefore, this approach fills up the message log in 16 CPU scans with duplicate entries. To make only one entry in the log per fault event, use the next ladder example with the PD coil.

System Error Display

System error messages create their own text when the error event occurs. When the DV-1000 is in Message Mode, the error code and message are displayed as long as the error persists. Or, you may press the **Clear** Key to acknowledge the message, and it will be removed from the log and from the display.

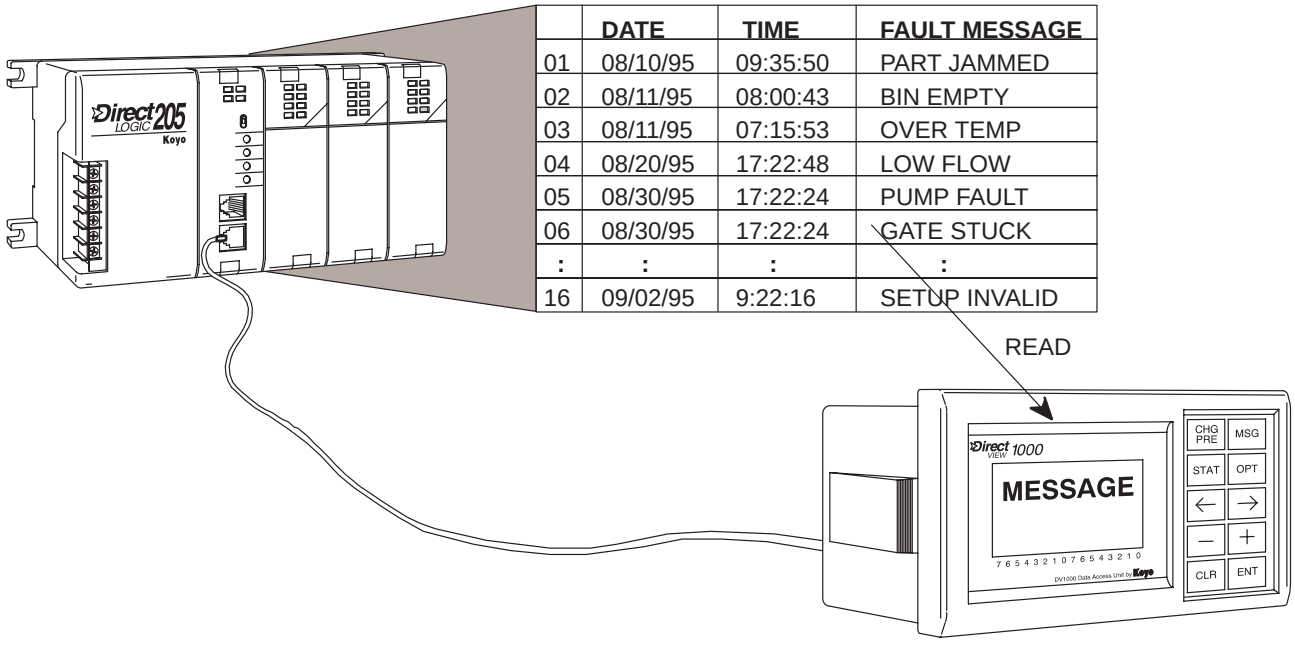
System Error Message



Viewing Message Logs (DL240, DL250, DL350, DL440 and DL450 Only)

The DL240, DL250, DL350, DL440 and DL450 CPUs record up to 16 System Error Messages and 16 Fault Messages in separate message logs. The PLC attaches a time/date stamp to each error or fault message when they occur. These may be viewed one at a time with the DV-1000, as depicted in the following figure.

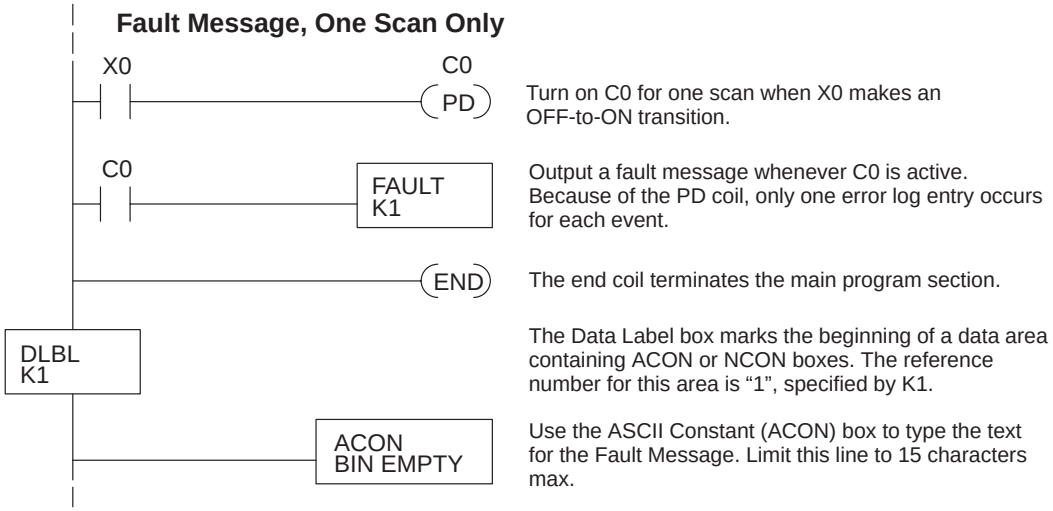
Example Message Log



	DATE	TIME	FAULT MESSAGE
01	08/10/95	09:35:50	PART JAMMED
02	08/11/95	08:00:43	BIN EMPTY
03	08/11/95	07:15:53	OVER TEMP
04	08/20/95	17:22:48	LOW FLOW
05	08/30/95	17:22:24	PUMP FAULT
06	08/30/95	17:22:24	GATE STUCK
:	:	:	:
16	09/02/95	9:22:16	SETUP INVALID

Fault Message Log (DL240, DL250, DL350, DL440 and DL450 Only)

The fault message log records a new entry on each scan a fault message box is active. This means that if we use the previous ladder example with a simple switch activating the fault message, the fault message log will fill up with duplicate messages in just 16 scans. After that, more fault messages just overwrite existing log entries. Therefore, we add a PD coil (positive differential) which activates for one scan when the off-to-on transition of X0 occurs.



Enter the program above or load it from the example disk into your CPU. After going to RUN Mode, turn X0 on and off once to cause one fault message entry in the message log. The message logs are accessible using the Options Menu. Press the **Options** Key to reach the menu shown below. Then use the **Plus (+)** Key to move the cursor over the third menu selection, SHOW ERR/MSG.

PRESS **OPT**

```
    OPTION
1 BIT CONTROL
2 PASSWORD
3 SHOW ERR/MSG
```

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

PRESS **+** **+** **ENT**

```
SHOW ERR/MSG
1 ERROR LIST
2 MESSAGE LIST
```

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Finally, select the message list using the **Plus (+)** Key, and press **Enter**. Here is the last entry in the message log, and it should show the results of your running the program above. Use the **Minus (–)** Key to move upward through the log when multiple entries exist, and the **Plus (+)** Key to move downward.

PRESS **+** **ENT**

```
Bin Empty

 07/25/95
 15:57:24:86
```

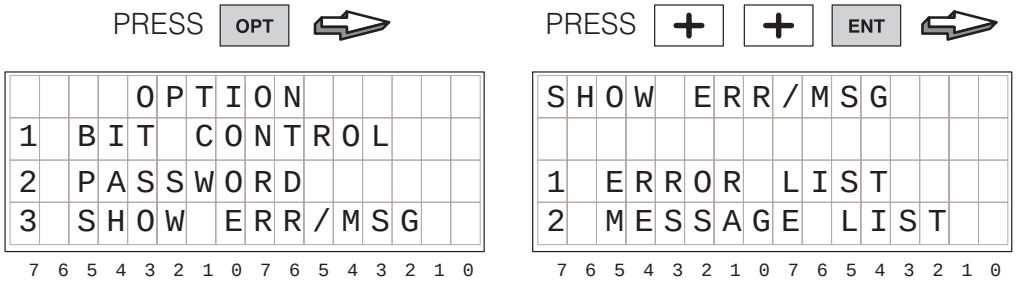
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

NOTE: Fault Message time stamps include hundredths of a second.

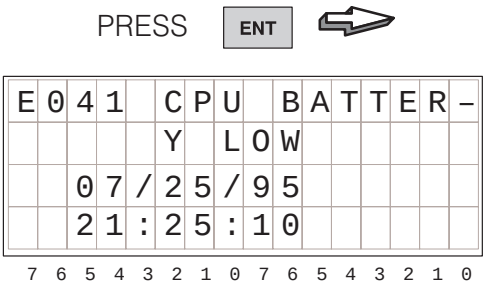
System Error Log (DL240, DL250, DL350, DL440 and DL450 Only)

The System Error Log feature in the CPU records a new entry whenever each error first occurs. No ladder logic is required to create the log. Log entries consist of an error code, text description, and a time/date stamp. The log records up to 16 error events. Each error event may last a few milliseconds, or many hours, but only one log entry is made per event. Subsequent system errors create new entries in the log.

To view the System Error Log, use the same Options Menu as the Fault Message Log, by pressing the **Options** Key. Then use the **Plus (+)** Key to move the cursor over the third menu selection, SHOW ERR/MSG.

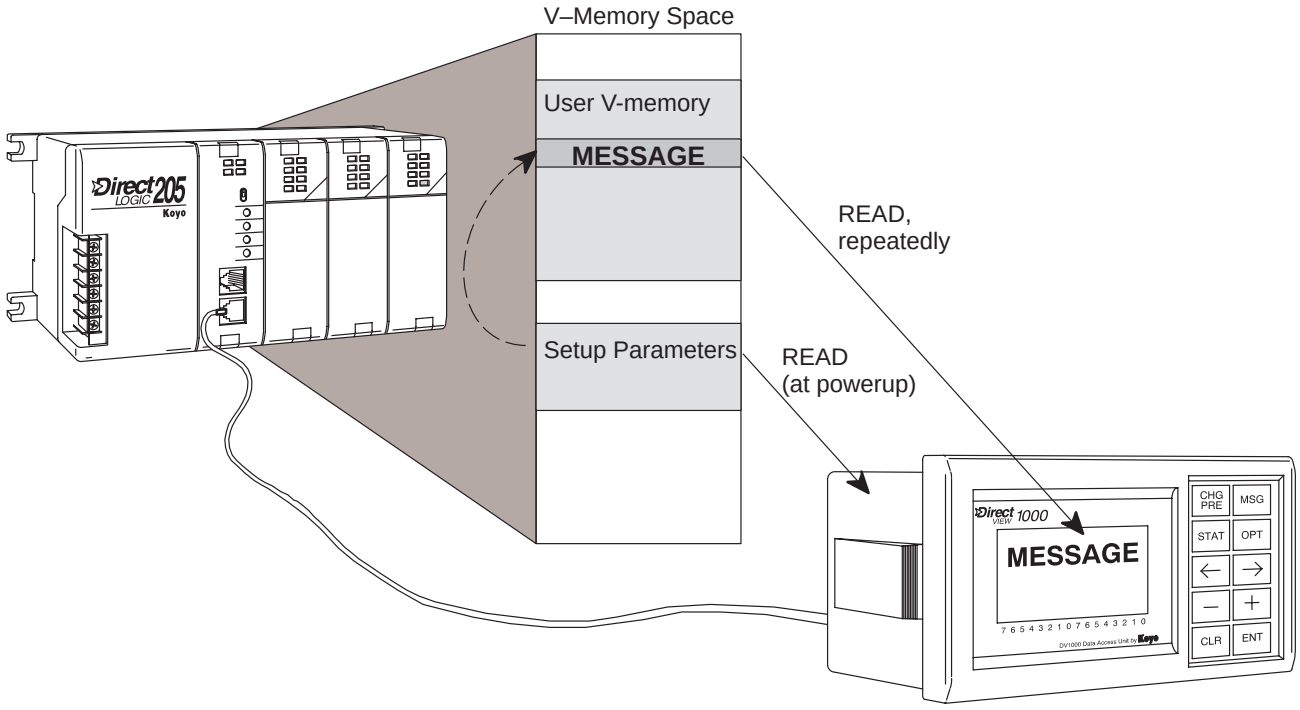


Finally, select the error list using the **Enter** Key. The display to the right shows a low battery voltage event as the last entry in the message log. Use the **Minus (–)** Key to review upward through the log whenever multiple entries exist, and the **Plus (+)** Key to move downward.

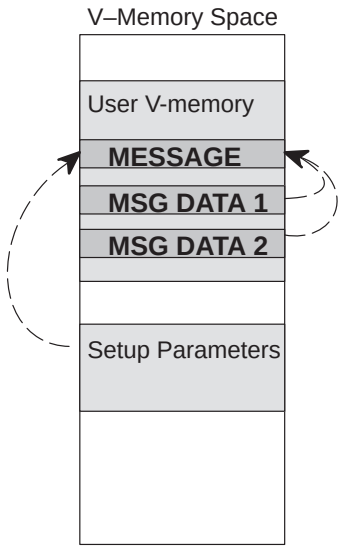


User Messages

User Messages rely on data stored in V-memory to actually create the message. The process is shown in the following figure. After powerup, the DV-1000 reads the setup parameters. Certain parameters can be programmed to point to other blocks of information in V-memory. The DV-1000 reads the text and numeric data pointed to by the setup parameters. Then it combines text and numeric data following simple rules to create the display output. Then it reads the V-memory message data again, and repeats the process over and over.



The DV-1000 currently reads the setup parameters at powerup, and when a key on the keypad is pressed. This means that the ladder program cannot simply change the pointer value for message data. Instead, it must move new data into the original message data block. This chapter covers the use of ACON and MOVMC instructions, which make this task easier.



Setup Parameters for User Messages

The DV-1000 is capable of displaying numbers or alphabetical characters, or a combination of both (alphanumeric). All of the programming to do this must be contained in the CPU's ladder logic program. This means that a portion of the ladder program is dedicated to the DV-1000 operation, while the remainder of the program handles machine or process operation. The Message Display Mode does require setup parameters (see Chapter 3 for an introduction to setup parameters).

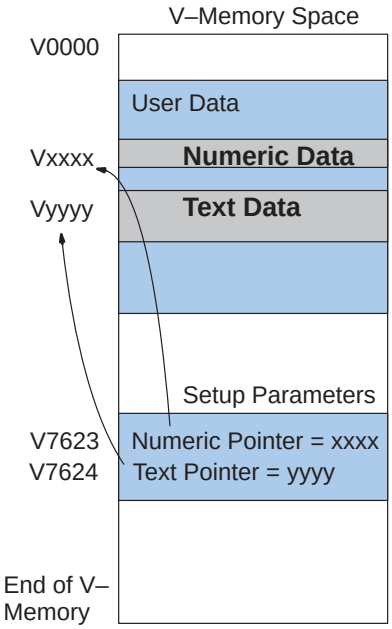
Message Data Parameters

There are two pointers in the setup parameters which pertain to message displays:

- Numeric Message Pointer
- Text Message Pointer

These two setup parameters are located at V7623 and V7624 respectively. The diagram to the right shows how the values xxxx and yyyy point to the address locations of blocks of data in the user data space of V-memory.

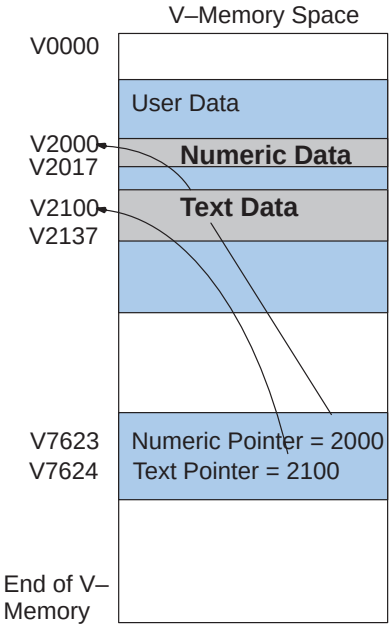
The relative sizes of the shaded blocks indicate that the text data block (32 words) is twice the size of the numeric data (16 words). We explain why this is true a bit later in this chapter. These block sizes are fixed (there is no block size setup parameter for these).



Choosing Data Block Locations

Now we decide where to place our numeric and text data blocks for use in the example programs in this chapter. As described in Chapter 2, we will use an available memory area common to all CPU types, starting at V2000. Therefore, numeric data occupies 16 word locations, from V2000 to V2017. We'll arbitrarily place the text table starting at V2100. It occupies 32 word locations, extending down to V2137. Note that either text or numeric data may be placed first in V-memory. However, the blocks must not overlap each other, but one may immediately follow the other.

The locations we have chosen here for numeric and text data are examples only. The location you choose for your application may be different.

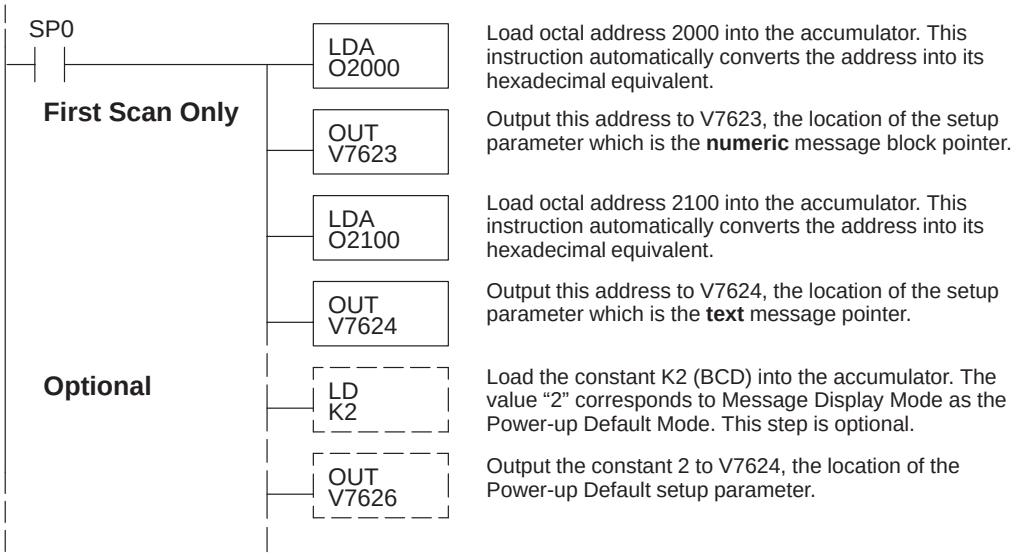


Ladder Example for Message Setup Parameters

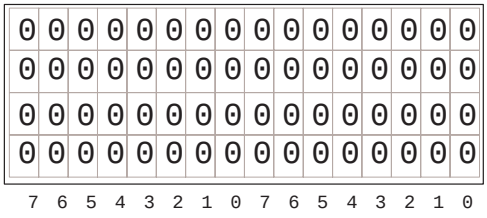
Now that we have chosen locations for numeric and text data, we can create the setup portion of the ladder logic program. The following ladder program locates the data blocks according to the memory map on the previous page. Also, we include instructions which make Message Mode the DV-1000 power-up default mode. For the ladder examples in this chapter, the DV-1000 will automatically power up in Message Display Mode. All ladder examples in this chapter will use the following program as the first rung to program the setup parameters.

Parameter Setup Program #1

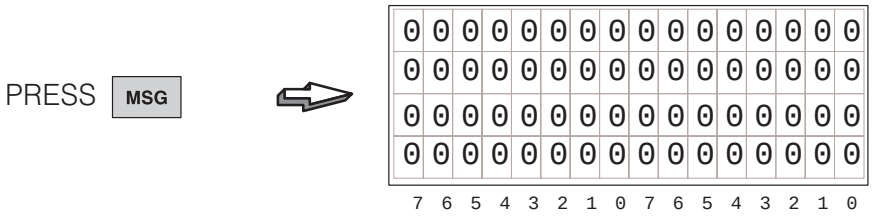
Parameter Setup: Numeric and Text Data



After entering the setup program above, you can test it by powering up the PLC and DV-1000. Note that the V-memory data areas must be cleared (all zeros). The display to the right will appear without your having to use the keypad.



As you program and use the DV-1000, it will eventually be in other modes when you need to use the Message Display Mode. Just press the **Message** Key on the keypad. If the setup parameters exist and are valid, the message display will appear. If they are not, the error message will be displayed: “Setup Parameter Error”. If this occurs, examine the parameter setup rung in the ladder program, using the program above as an example. Verify the numeric and text data blocks are located in available user data space of V-memory (refer to Chapter 3 for CPU-specific memory maps).

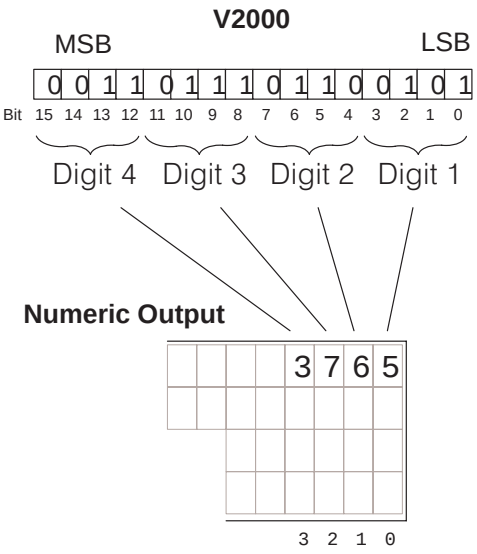


NOTE: Remember to load the example programs while the CPU is in program mode. After loading the program, take the CPU from program mode to run mode. Most of the examples in this chapter have a first scan rung (SP0). This requires a program-to-run mode transition or power cycle, or the example may not work properly!

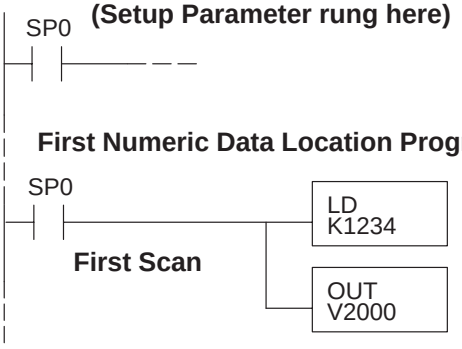
Displaying Numbers

Values in the numeric data block are organized as 16-bit numbers. These may be BCD or hex numbers. In either case, they contain four digits. The example to the right shows memory location V-2000, which contains the number “3765”.

The DV-1000 interprets the numeric data block directly, and displays them as numbers. The value “3765” automatically appears on the display in the corresponding numeric display position when the same number exists in V-memory, and is referenced by the setup parameters.



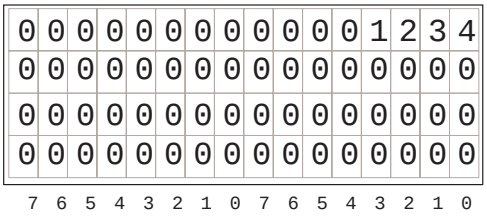
The following example shows how easy it is to display a number from your ladder logic program. All you need is the setup program in the previous section, followed by a rung that places the number “1234” as the first numeric data block entry. The DV-1000 then displays the number at the corresponding first numeric position on the display.



Load the arbitrary constant “1234” into the accumulator. This is the number we will display. We only need to do this once, on the first scan.

Output the constant “1234” to location V2000, the first numeric data location. This value will appear in the first numeric position in the display, shown below.

After loading the program above, you can test it by taking the PLC from Program to Run Mode. The BCD constant “1234” appears in the first numeric position of the display, in the upper *right* corner. We discuss the entire display output positions a bit further in this chapter.



Displaying Changing Values

In real applications, the numbers you want to display will often be changing. Process variables such as temperature, flow rate, or conveyor speed can be displayed in real time. While the setup parameters are written only once (first CPU scan), process variables must be constantly written. In our example, it means writing a changing number to location V-2000. The DV-1000 takes care of the rest of the work. Because our setup parameters specify the location of the numeric (and text) data, the DV-1000 constantly scans these so that any change can be observed in the display.

To simulate a changing value, we use the following program rungs to just alternate between two values. It uses a special relay contact SP4 as a 1 second timer (1/2 sec. on, 1/2 sec. off). Keeping the Parameter Setup rung, add the following rungs as the main program.

SP0 **(Setup Parameter rung here)**

Demonstration of Changing Values

SP4 **1 Sec Timer** — LD K1234

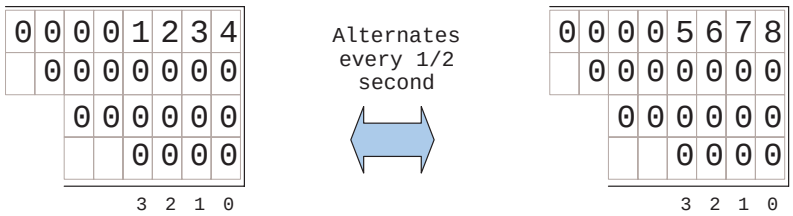
For the first half of each second, load the arbitrary constant 1234 into the accumulator. Special contact SP4 is on 1/2 second, and off 1/2 second.

SP4 — LD K5678

For the second half of each second, load the arbitrary constant 5678 into the accumulator. Special contact SP4 is on 1/2 second, and off 1/2 second.

SP1 **Always On** — OUT V2000

Output whatever value is in the accumulator to location V2000. Half of the time it will be “1234” and the other half of the time it will be “5678”.



The point is: Your ladder program needs to write the Setup Parameters only on the first scan, but it must update the numeric and text data in V-memory as often as their content changes. The DV-1000 reads the numeric and text data repeatedly during operation. If the text or numeric data changes, the display then automatically follows.

Displaying Text

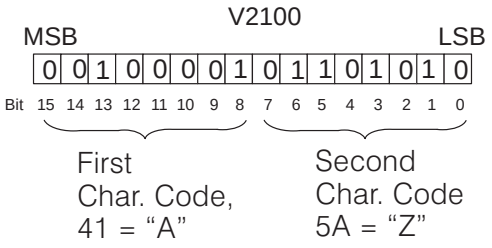
ASCII Codes

Next we discuss how to display text in message displays. The text data block is located at V2100 in these examples (only because that is where our setup parameter has defined it to be). However, we cannot place alphabetical characters directly in V-memory. Instead, we use a numeric code which *represents* text characters, called ASCII codes. Below is a portion of the ASCII table which lists capital letters (see Appendix B for the complete table of characters and symbols with their ASCII codes).

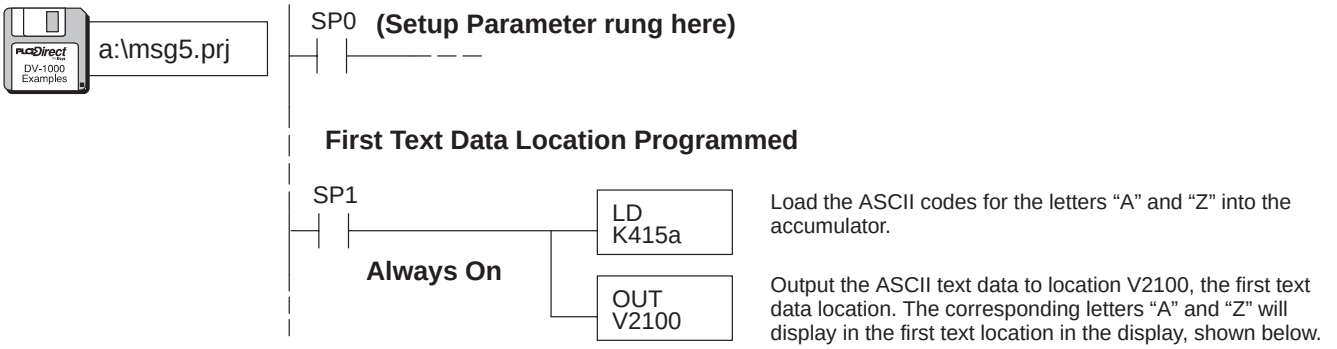
ASCII Code	Char.	ASCII Code	Char.	ASCII Code	Char.	ASCII Code	Char.
41	A	48	H	4F	O	56	V
42	B	49	I	50	P	57	W
43	C	4A	J	51	Q	58	X
44	D	4B	K	52	R	59	Y
45	E	4C	L	53	S	5A	Z
46	F	4D	M	54	T	–	–
47	G	4E	N	55	U	–	–

An ASCII code is an 8-bit binary number (byte) representing a single text character or symbol. Therefore, each V-memory location (16-bit data word) in our text table can contain two ASCII codes, representing two characters.

The example to the right uses the capital letters “A” and “Z”. The ASCII codes for these are 41 and 5A (in hexadecimal) respectively. Memory location V2100 is the first text data location, shown containing the codes for the letters “A” and “Z”.



Use the following ladder rung as the main program to load the ASCII codes for “A” and “Z” into the first text data location (keep the Setup Parameter rung in the program).



After loading the program above, you can test it by taking the PLC from Program to Run Mode. The letters “AZ” appear in the first text position of the display, in the upper *left* corner. Unless you have cleared V2000, the number “1234” or “5678” from the previous example will be displayed in the upper right corner.

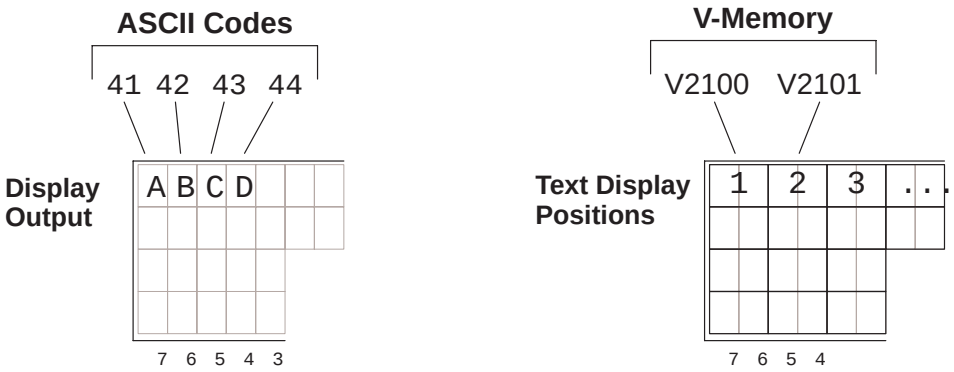
A	Z	0	0	0	0	0	0	0	0	0	1	2	3	4	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Using LDD and OUTD Instructions

The regular LD and OUT instruction manipulates a 16–bit word, updating two ASCII characters on the display. However, we can use the Load Double (LDD) and Out Double (OUTD) to manipulate 32–bit double words, updating four characters at a time (*two* text positions)!

NOTE: If you have less than 10 characters to place, using the LDD instruction(s) is a good choice. **However, for placing more than a dozen characters adjacently, the ACON Instruction is much easier and more efficient.** See the section on using the ACON instruction on page 4–25.

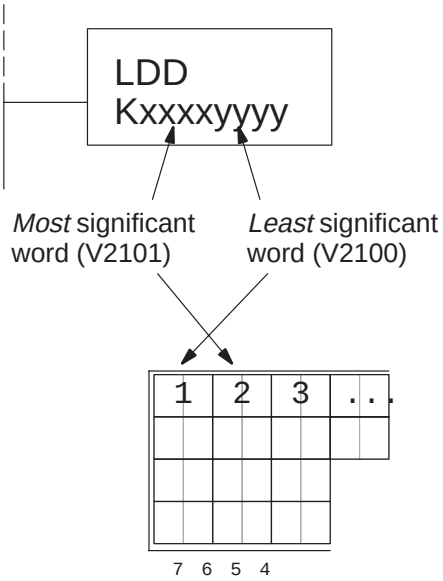
Suppose we place the character string “ABCD” in the first two text positions. The ASCII data for “ABCD” will be stored in V2100, and V2101, the first two data words in the text data block.



NOTE: You **must** swap positions with the first and second pair of ASCII codes in a LDD instruction. Read the following discussion to learn why this is necessary!

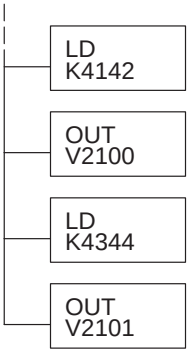
The Load Double Instruction accepts an 8-digit constant K, representing two 16-bit words. The most significant word xxxx is on the left in the LDD box. However, the DV-1000 displays the text of the most significant word on the right. This means that you must swap order of the two pairs of ASCII codes in the LDD box relative to how it is displayed.

The reason for this swap is that we read from left to right, and the ASCII codes are stored from lower memory address to higher memory addresses. On the other hand, computer data numbering goes from right (least significant) to left (most significant).

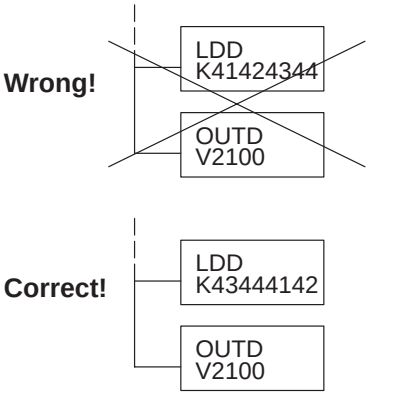


Now let's use the LDD instruction to place "ABCD" in the first two text positions. In the following figure, the method using the regular LD and OUT instructions is on the left. On the right are examples using the LDD and OUTD instructions, *without* the swap (wrong) and *with* the swap (correct).

Using LD and OUT Instructions



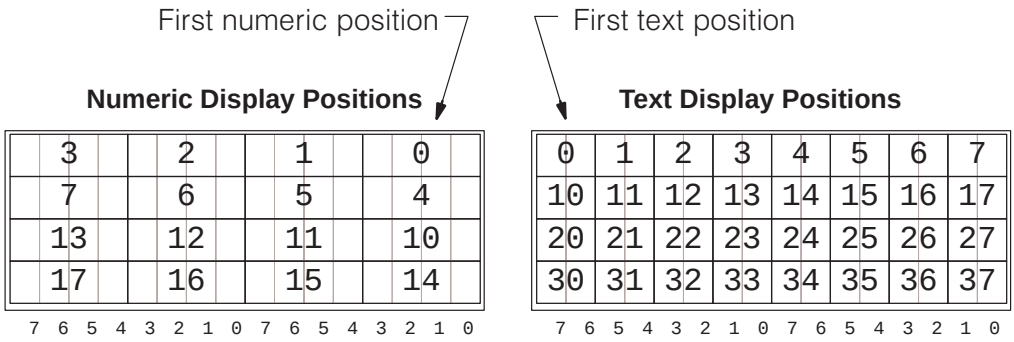
Using LDD and OUTD Instructions



Turning Data into Messages

Numbering of Display Positions

Now we have displayed a number and some text using single entries (one V-memory location) in the numeric and text tables. You have probably noticed that the first locations are in opposite corners of the display. Refer to the figure below. Numeric positions increment from *right to left*, like LSB to MSB orientation. Text positions increment from *left to right*, just as we read from a book. We number the positions in octal, and start with zero.



(Octal numbering of positions**)**

The display contains a total of 16 numeric and 32 text positions. Each position corresponds to one V-memory location (16-bit word). Each numeric position occupies 4 character spaces, while each text position occupies 2 character spaces. The example programs in this manual begins numeric data block at V2000 and text block data at V2100. The figure below shows how numeric and text data locations map to physical locations on the DV-1000 display. Because of the octal numbering, **the display position numbers are equal to the address offsets!**

Display Maps

Numeric Display Map

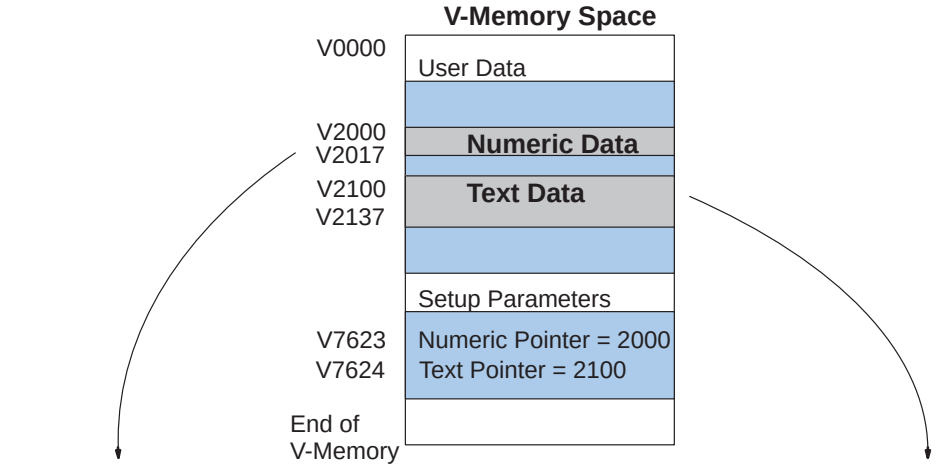
3 V2003	**2** V2002	**1** V2001	**0** V2000
7 V2007	**6** V2006	**5** V2005	**4** V2004
13 V2013	**12** V2012	**11** V2011	**10** V2010
17 V2017	**16** V2016	**15** V2015	**14** V2014

Text Display Map

0 V2100	**1** V2101	**2** V2102	**3** V2103	**4** V2104	**5** V2105	**6** V2106	**7** V2107
10 V2110	**11** V2111	**12** V2112	**13** V2113	**14** V2114	**15** V2115	**16** V2116	**17** V2117
20 V2120	**21** V2121	**22** V2122	**23** V2123	**24** V2124	**25** V2125	**26** V2126	**27** V2127
30 V2130	**31** V2131	**32** V2132	**33** V2133	**34** V2134	**35** V2135	**36** V2136	**37** V2137

How Message Data Gets to the Display

The figure below shows the numeric and text data in V-memory. From there, follow the arrows to detailed numeric and text tables which show the V-memory contents for the examples thus far in this chapter. Notice how the offset from the base address (V2000 and V2100) for each data block corresponds to actual display position numbering. Follow the arrows to the bottom of the page, where the DV-1000 combines numeric and text information into one display.



Numeric Data

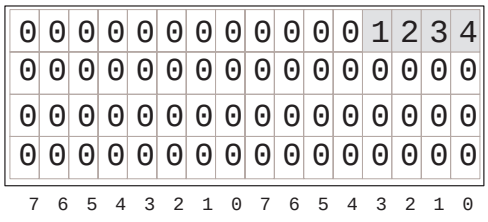
Position	V-Memory	Number
0	2000	1234
1	2001	0000
2	2002	0000
3	2003	0000
4	2004	0000
5	2005	0000
6	2006	0000
7	2007	0000
10	2010	0000
11	2011	0000
12	2012	0000
13	2013	0000
14	2014	0000
15	2015	0000
16	2016	0000
17	2017	0000

Text Data

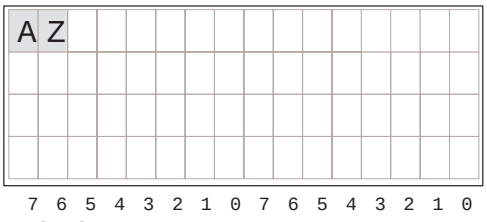
Position	V-Memory	Data	Text
0	2100	415A	A Z
1	2101	0000	
2	2102	0000	
3	2103	0000	
4	2104	0000	
5	2105	0000	
6	2106	0000	
7	2107	0000	
10	2110	0000	
11	2111	0000	
12	2112	0000	
13	2113	0000	
14	2114	0000	
15	2115	0000	
16	2116	0000	
17	2117	0000	

Position	V-Memory	Data	Text
20	2120	0000	
21	2121	0000	
22	2122	0000	
23	2123	0000	
24	2124	0000	
25	2125	0000	
26	2126	0000	
27	2127	0000	
30	2130	0000	
31	2131	0000	
32	2132	0000	
33	2133	0000	
34	2134	0000	
35	2135	0000	
36	2136	0000	
37	2137	0000	

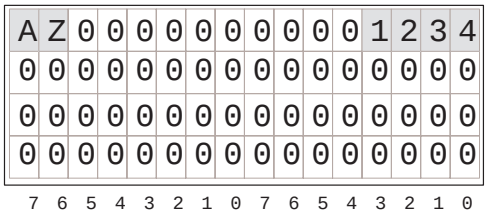
Numeric Display Output



Text Display Output



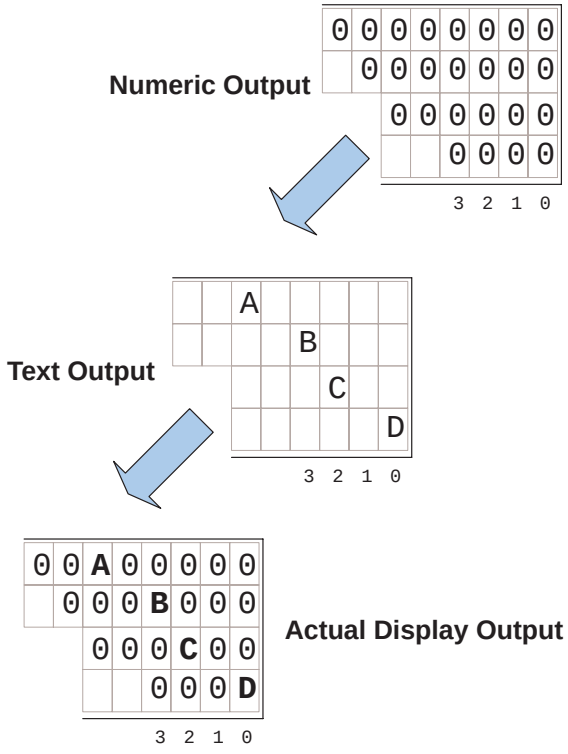
DV-1000 Display



Combined Numeric and Text Displays

Obviously, text and numbers cannot occupy the same position on the actual DV-1000 display at the same time. The two V-memory data blocks certainly coexist, but the DV-1000 gives *display priority to the text data.* Each text character's data masks the corresponding numerical digit if the text's ASCII code is 20 (hex) or greater.

It is convenient to think of the text output as a *control mask* for numeric output. Refer to the figure to the right. If the numeric data block in V-memory is all zeros, the matching numeric output is also zeros, as shown. Now suppose the text data is also all zeros (null characters), except for four random locations as shown. Text data ASCII codes 41 through 44 hex produces the letters A, B, C, and D. Now we come to the actual display output. It is a combination of the numeric and text output characters. The DV-1000 uses a simple principle to determine what to display in each character location:



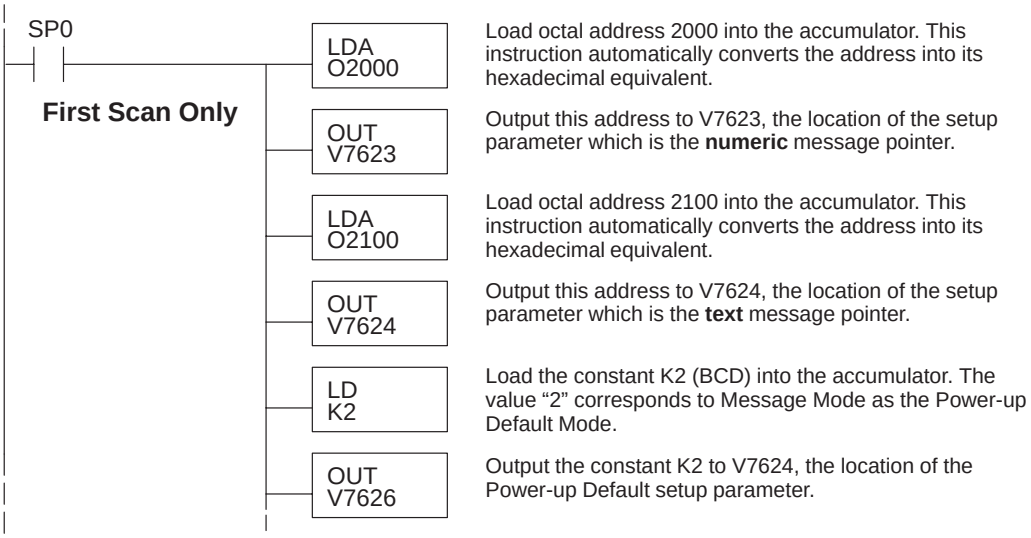
Any text character which has an ASCII code equal to 20 (hex) or greater masks (blocks) the numeric digit in the same display position. That's it. This works on a character-by-character basis.

One of the first things you may want to do in your setup program is fully mask all numerical output, so we begin with a blank display (unless you use ACON boxes to fill the entire display). This is analogous to "starting with a clean slate". Then all the main ladder program has to do is write text where desired, and *unmask* numerical positions where numbers are desired. Perhaps the best character to use is the space character, ASCII code 20 (hex). The display has 64 character positions, requiring 32 words of 2020 hex. For the DL405 CPUs, use the FILL instruction. For the DL105, DL205 and DL350 CPUs, you can use a single LDD instruction followed by sixteen OUTD instructions.

NOTE: You may recall seeing the display filled with zero (0) characters when you first powered up the DV-1000. Since the text data unmasked all the numeric locations, the numeric output (all zeros) was displayed.

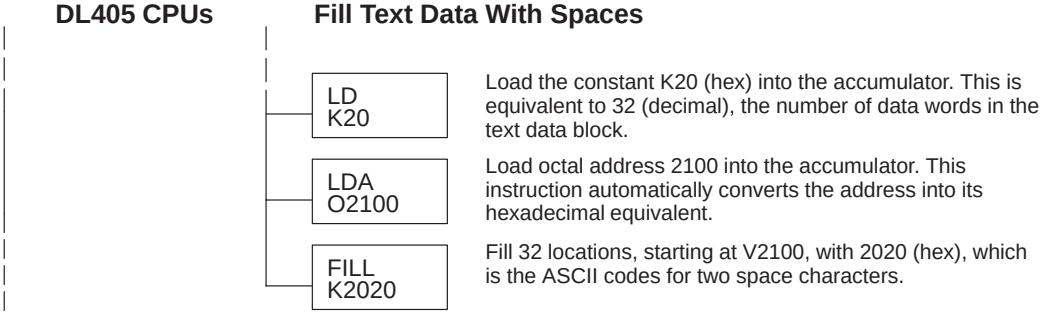
Parameter Setup Program #2

Parameter Setup: Numeric and Text Data



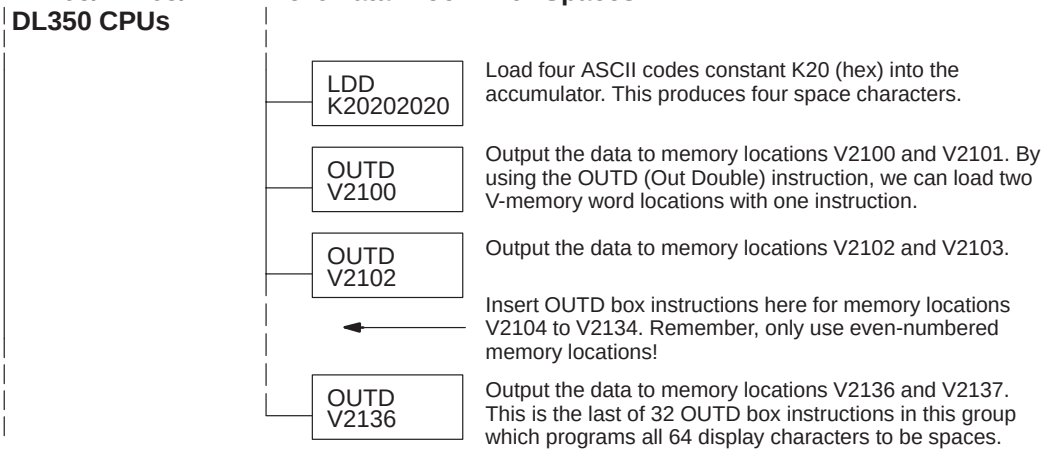
If you are using a DL405 type CPU, add the following ladder section to the parameter setup program above. The FILL instruction is ideal for this situation.

DL405 CPUs **Fill Text Data With Spaces**



If you are using a DL105, DL205 or DL350 type CPU, add the following ladder section to the parameter setup program at the top of this page. The FILL instruction box is not available for these CPUs, so the program uses one LDD box and 16 OUTD boxes. Remember that this section executes only on the first scan (overall scan time is not affected).

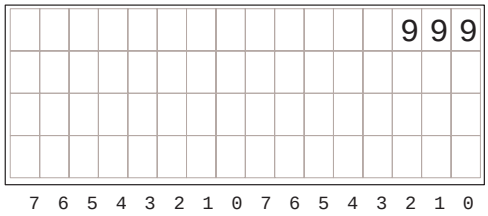
DL105/DL205/ DL350 CPUs **Fill Text Data Block With Spaces**



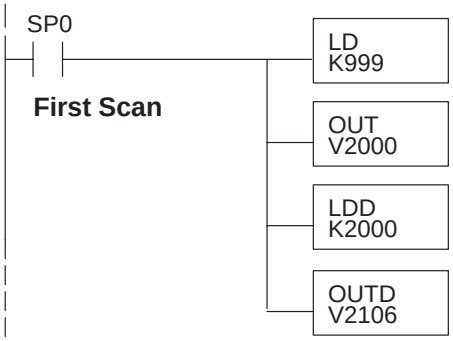
Unmasking Numeric Positions

The setup program on the previous page masks all numeric positions as a starting point. Now we add the main ladder program. It loads the number, and then unmasks the corresponding specific numeric display positions.

As an example, let's place a number in the first numeric position that varies from 0 to 999. However, the V-memory contents will vary from 0000 to 0999. We can leave the leading 0 masked, and only unmask the three digits we want to display, as shown to the right.



Now we're ready to write the main ladder program, which follows. It places "0999" in the first numeric data location, and then unmasks the three display positions of interest. We use the null character, ASCII code "00", to unmask the numeric output.



Load the value "999" in the accumulator.

Output the data to memory location V2000. The actual contents of V2000 are "0999". However, the next instruction will leave the leading "0" masked, and unmask the "999".

Use a Load Double (LDD) to place the ASCII codes for a space " " character followed by three null characters into the accumulator (remember the swap effect for LDD box).

Output the data to memory locations V2106 and V2107. The Out Double instruction updates two text positions at a time (two data words).

Numeric Display Positions

3	2	1	0
7	6	5	4
13	12	11	10
17	16	15	14

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Numeric Data

Position	V-Memory	Number
0	2000	0999
1	2001	0000
2	2002	0000
3	2003	0000
4	2004	0000
5	2005	0000
6	2006	0000
7	2007	0000
10	2010	0000
11	2011	0000
12	2012	0000
13	2013	0000
14	2014	0000
15	2015	0000
16	2016	0000
17	2017	0000

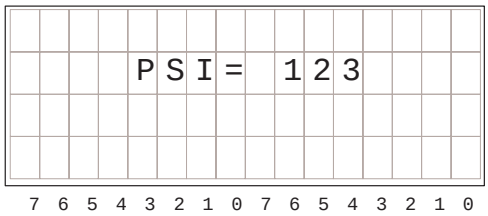
Text Data

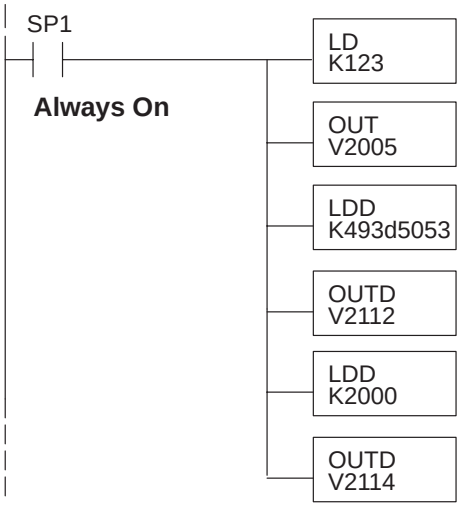
Position	V-Memory	Data	Text
0	2100	2020	
1	2101	2020	
2	2102	2020	
3	2103	2020	
4	2104	2020	
5	2105	2020	
6	2106	2000	
7	2107	0000	
10	2110	2020	
11	2111	2020	
12	2112	2020	
13	2113	2020	
14	2114	2020	
15	2115	2020	
16	2116	0000	
17	2117	0000	

Position	V-Memory	Data	Text
20	2120	2020	
21	2121	2020	
22	2122	2020	
23	2123	2020	
24	2124	2020	
25	2125	2020	
26	2126	2020	
27	2127	2020	
30	2130	2020	
31	2131	2020	
32	2132	2020	
33	2133	2020	
34	2134	2020	
35	2135	2020	
36	2136	2020	
37	2137	2020	

Placing Numbers and Text Together

Still using a blank display as a starting point, let's create a display featuring both text and numbers. Since there is more flexibility in text placement, we plan the display layout around using numeric position #5 for the content "123".The following program writes text and numeric mask characters in the same way.





Load the value "123" in the accumulator.

Output the data to memory location V2005. The actual contents of V2005 are "0123". However, the last instruction will leave the leading "0" masked, and unmask the "999".

Use a Load Double (LDD) to place the ASCII codes for the characters "PSI=" into the accumulator. Remember the LDD instruction swaps the text output positions.

Output the data to memory locations V2112 and V2113. The Out Double instruction updates two text positions at a time (two data words).

Use a Load Double (LDD) to place the ASCII codes for one space (masks leading 0) and three null characters (unmasks "123").

Output the data to memory locations V2114 and V2115. The Out Double instruction updates two text positions at a time (two data words).

Numeric Display Positions

3	2	1	0
7	6	5	4
13	12	11	10
17	16	15	14

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Numeric Data

Position	V-Memory	Number
0	2000	0000
1	2001	0000
2	2002	0000
3	2003	0000
4	2004	0000
5	2005	0123
6	2006	0000
7	2007	0000
10	2010	0000
11	2011	0000
12	2012	0000
13	2013	0000
14	2014	0000
15	2015	0000
16	2016	0000
17	2017	0000

Text Data

Position	V-Memory	Data	Text
0	2100	2020	
1	2101	2020	
2	2102	2020	
3	2103	2020	
4	2104	2020	
5	2105	2020	
6	2106	2000	
7	2107	0000	
10	2110	2020	
11	2111	2020	
12	2112	5053	
13	2113	493d	
14	2114	2000	
15	2115	0000	
16	2116	2020	
17	2117	2020	

Position	V-Memory	Data	Text
20	2120	2020	
21	2121	2020	
22	2122	2020	
23	2123	2020	
24	2124	2020	
25	2125	2020	
26	2126	2020	
27	2127	2020	
30	2130	2020	
31	2131	2020	
32	2132	2020	
33	2133	2020	
34	2134	2020	
35	2135	2020	
36	2136	2020	
37	2137	2020	

Using the ASCII Constant (ACON) Instruction

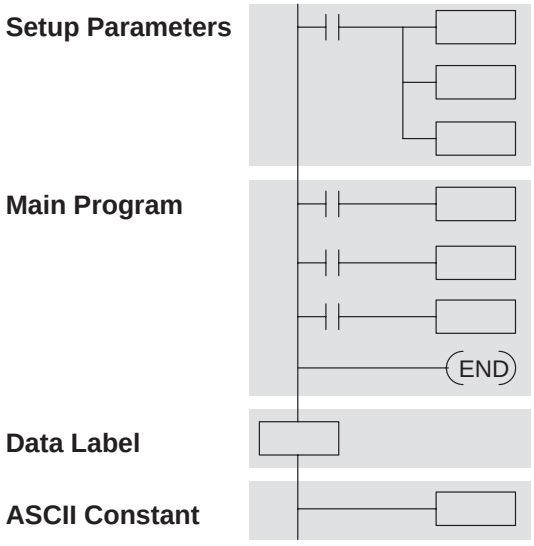
The examples so far in this chapter have covered DV-1000 display basics. The Load Double (LDD) instruction loads ASCII values into memory for only four characters to be displayed. This method requires looking up ASCII values from a table (or memorizing the table), and can potentially use a lot of LDD and OUTD instructions.

Fortunately, the ACON instruction (ASCII Constant) provides an easier method of text entry. The ACON box is available on DL105, DL205 and DL350 CPUs and on DL440 and DL450 CPUs (not available on DL430 CPUs). Using ***Direct***SOFT, you can type the characters you want directly in the instruction box! After learning the LDD method and the ACON method, you can use either one or both, based on individual preferences and your application.

NOTE: In the CPU, the basic ACON function box only accepts two characters. If you are using a Hand-held Programmer in creating ladder logic to run the DV-1000 display, note that its ACON instruction is limited to *two* characters. However, ***Direct***SOFT permits you to enter up to *forty* characters. ***Direct***SOFT creates an extended ACON box for character entry, then subdivides it into multiple 2-character ACON boxes when downloading the program to the CPU. This section assumes you are using ***Direct***SOFT.

Using the ACON method of text entry adds new ladder elements onto our existing program. The figure to the right shows an outline of the program. The setup parameter rung is first, as described in Chapter 2. The main program follows, just as we have been creating main program examples in this chapter.

After the end of the main program (and the End Coil), we add a Data Label box. It announces the data box which follows, and allows the main program to reference the data box by the label. The ACON (ASCII Constant) box follows as the last program element.

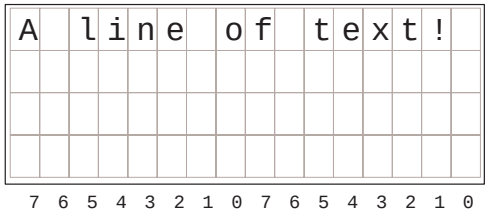


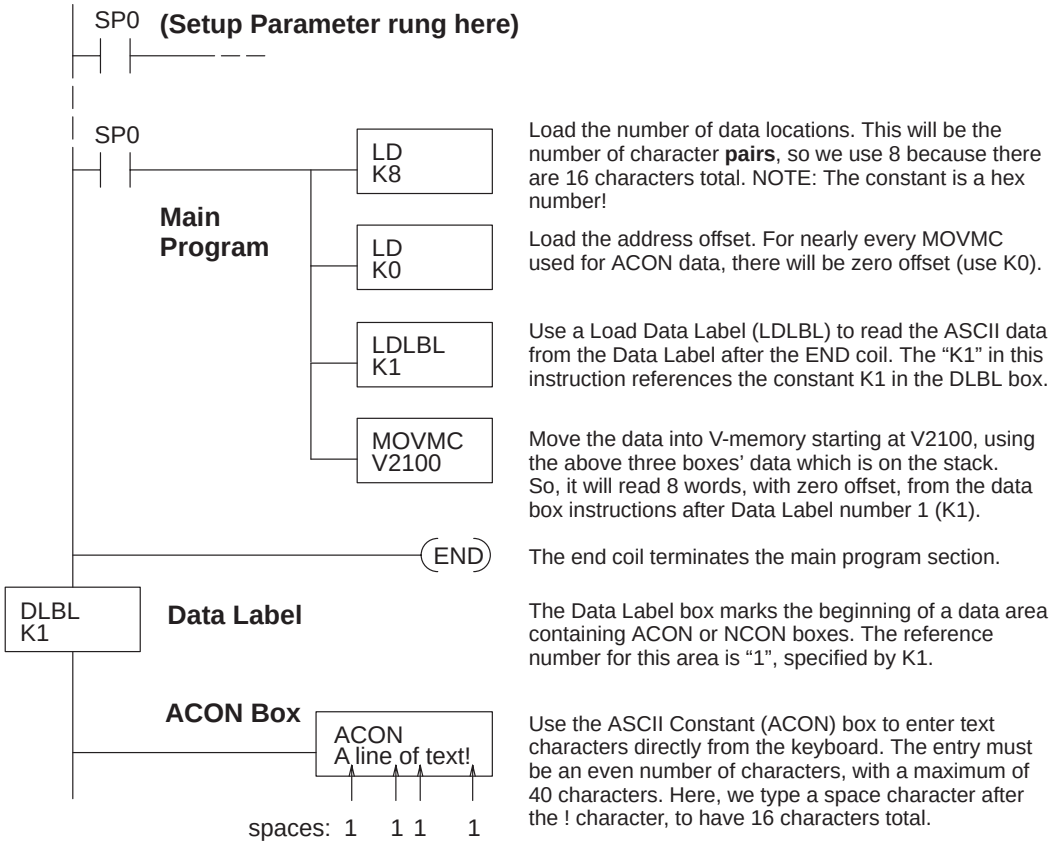
The main program section receives a new instruction, too. The MOVMC (Move Memory Cartridge) instruction reads the ACON data (the text data) into V-memory space so the main ladder program can use it. Note that the MOVMC requires supporting data to be loaded onto the stack before its use. The following example shows how this process works.

ACON Example Program #1

Now we create a program example to use the ACON instruction. The example display to the right shows “A line of text!” as the message on the first line. Even though this is 15 characters, the ACON box must contain an even number of characters (we’ll use 16). And, this example still uses a blank display as a starting point (setup program #2).

Desired Display





Load the number of data locations. This will be the number of character **pairs**, so we use 8 because there are 16 characters total. NOTE: The constant is a hex number!

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Use a Load Data Label (LDLBL) to read the ASCII data from the Data Label after the END coil. The “K1” in this instruction references the constant K1 in the DLBL box.

Move the data into V-memory starting at V2100, using the above three boxes’ data which is on the stack. So, it will read 8 words, with zero offset, from the data box instructions after Data Label number 1 (K1).

The end coil terminates the main program section.

The Data Label box marks the beginning of a data area containing ACON or NCON boxes. The reference number for this area is “1”, specified by K1.

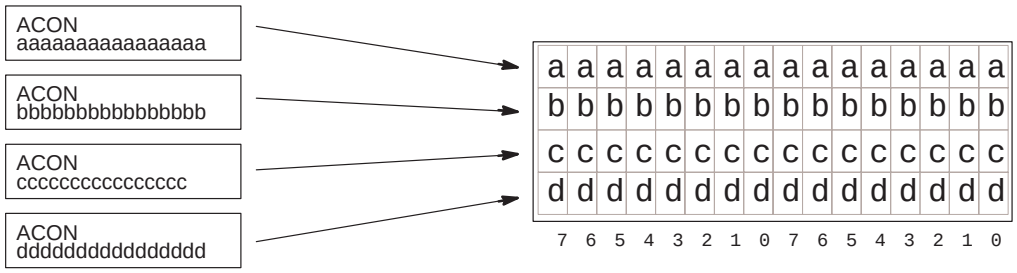
Use the ASCII Constant (ACON) box to enter text characters directly from the keyboard. The entry must be an even number of characters, with a maximum of 40 characters. Here, we type a space character after the ! character, to have 16 characters total.

We recommend using the ACON method whenever you need to put several characters on the display.

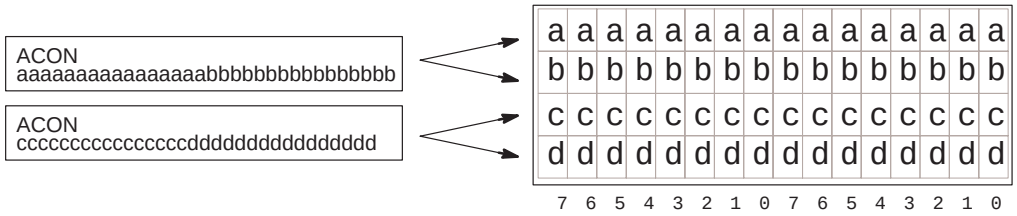
NOTE: Appendix C contains some worksheets to use in creating your own messages. It includes a worksheet for loading ASCII codes individually and another worksheet for using ACON boxes.

Choosing ACON Box Text Length

After using the ACON instruction to place a line of text on the first display row, we're ready to create a full-display message covering all four rows. The previous example used one 16-character ACON box. If we extend this method to generate a message for the entire display, we'll need four ACON boxes in separate data label areas. The organization of ACON boxes to display text is as follows:



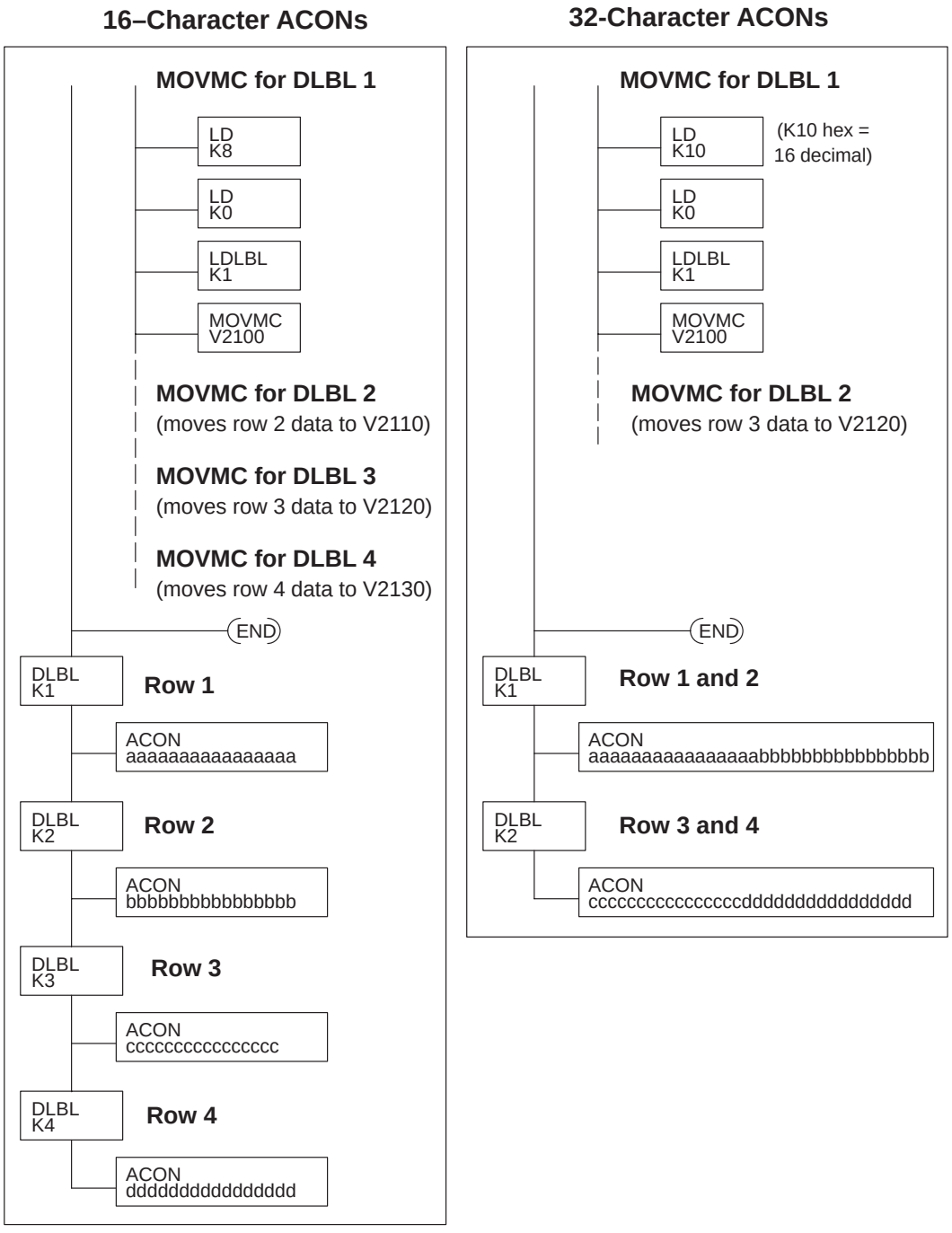
You may recall that the ACON instruction box (using ***Direct***SOFT) can accept up to 40 characters. Although there are many possible combinations, at least two ACON boxes are required to fill the DV-1000's 64-character display. To take maximum advantage of the ACON box capacity, let's create a display using just two ACON instructions. Using equally-sized ACON boxes, each one contains 32 characters which will fill two rows, as shown below:



The text in the 32-character ACON Boxes wraps around from one row to a second row on the display. This makes very efficient use of the ACON instruction. However, you have to carefully count characters and spaces so the display output does what you want it to do.

When writing your own programs, use the length of ACON text that best fits your application and is the easiest for you to use. In the interest of conserving space, most of the examples in this chapter which use ACONs use the 32-character version. However, the programs could have used twice as many 16-character ACON boxes instead.

The following ladder program outlines show equivalent 16-character and 32-character ACON box methods to generate the same display output. It assumes the text data block begins at V2100, but your memory map may differ.



ACON Example Program #2

This example shows the true power of the ACON method of text entry in creating display to the right. It shows the machine number and status, followed by a part count and production rate in parts per hour. Remember, the numeric locations of "1234" and "57" must be unmasked in the text data.

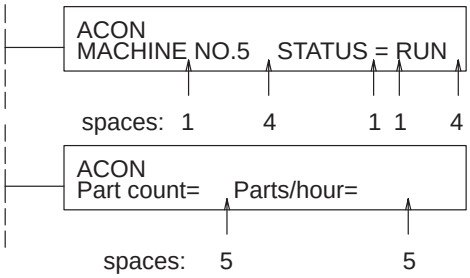
Desired Display

M	A	C	H	I	N	E		N	O	.	5				
S	T	A	T	U	S		=		R	U	N				
P	a	r	t		c	o	u	n	t	=		1	2	3	4
P	a	r	t	s	/	h	o	u	r	=				5	7
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Also, because the ACON text covers the entire display, there is no need to write spaces to all character positions to create a blank starting display. Therefore, we can use the simpler Setup Program #1 that just sets up the numeric and text data blocks.

ACON boxes can contain a maximum of 40 characters. The DV-1000 display contains 64 characters, so we arbitrarily use two equally-sized ACON boxes, at 32 characters each. Insert spaces for blank display locations, counting them precisely.

- Data for the first ACON box contains the *top two lines* of display text, and begins at V2100.
- Data for the second ACON box contains the *bottom two lines* of display text, and begins at V2120.



Numeric Display Positions

3	2	1	0
7	6	5	4
13	12	11	10
17	16	15	14

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Numeric Data

Position	V-Memory	Number
0	2000	0000
1	2001	0000
2	2002	0000
3	2003	0000
4	2004	0000
5	2005	0000
6	2006	0000
7	2007	0000
10	2010	1234
11	2011	0000
12	2012	0000
13	2013	0000
14	2014	0057
15	2015	0000
16	2016	0000
17	2017	0000

Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

ACON Box 2 data **ACON Box 1 data**

Text Data

Position	V-Memory	Data	Text
0	2100	4D41	M A
1	2101	4348	C H
2	2102	494E	I N
3	2103	4520	E
4	2104	4E4F	N O
5	2105	2E35	. 5
6	2106	2020	
7	2107	2020	
10	2110	5354	S T
11	2111	4154	A T
12	2112	5553	U S
13	2113	203D	=
14	2114	2052	R
15	2115	554E	U N
16	2116	2020	
17	2117	2020	

Position	V-Memory	Data	Text
20	2120	5061	P a
21	2121	7274	r t
22	2122	2063	c
23	2123	6F75	o u
24	2124	6E74	n t
25	2125	3D20	=
26	2126	0000	
27	2127	0000	
30	2130	5061	P a
31	2131	7274	r t
32	2132	732F	s /
33	2133	686F	h o
34	2134	7572	u r
35	2135	3D20	=
36	2136	2020	
37	2137	0000	

The following program places numeric and text data into the locations shown above.

SP0 **(Setup Parameter rung here)**

Load Values Into Numeric Data Block

SP1 **Always On**

LD K1234	Load the number “1234” into the accumulator. Your program would read in data from an analog input module for this step, etc.
OUT V2010	Place “1234” in the numeric data block location corresponding to numeric display position 10.
LD K57	Load the number “57” into the accumulator.
OUT V2014	Place “57” in the numeric data block location corresponding to numeric display position 14.

Move ACON Data Into Text Data Block

SP0 **First Scan**

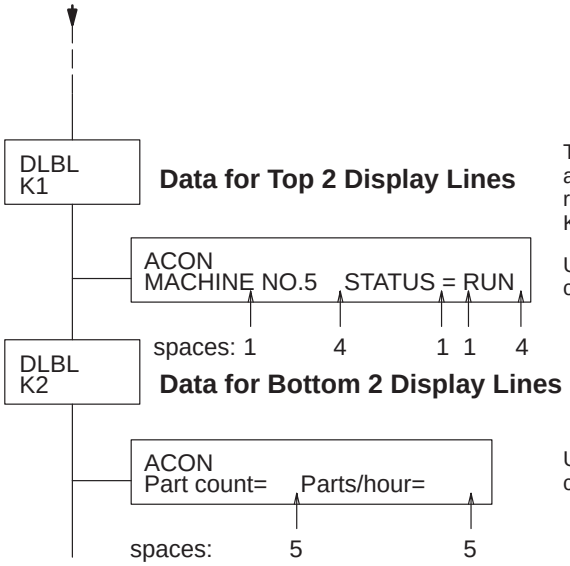
LD K10	Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0	Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K1	Read from the ACON following Data Label K1.
MOVMC V2100	Move data into V-memory starting at V2100, corresponding to the top two lines on the display.
LD K10	Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0	Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K2	Read from the ACON following Data Label K2.
MOVMC V2120	Move the data into V-memory starting at V2120, corresponding to the bottom two lines on the display.

Unmask Numeric Locations

SP0 **First Scan**

LDD K0	Load the constant zero (ASCII null character) into the accumulator.
OUTD V2126	Place zeros in text output position 26 and 27. This unmasks numeric position 10.
LDD K2020	Use a Load Double to update 4 character spaces. The K2020 actually implies K00002020. The zeros unmask the numeric “57”, and the spaces (2020) hide its leading zeros.
OUTD V2136	Place zeros in text output position 36, and spaces in output position 37.
END	The end coil terminates the main program section.

(continued)



The Data Label box marks the beginning of a data area containing ACON or NCON boxes. The reference number for this area is “1”, specified by K1.

Use ASCII Constant (ACON) boxes to enter text characters directly from the keyboard.

Use ASCII Constant (ACON) boxes to enter text characters directly from the keyboard.

Message Display Applications and Techniques

The material in this chapter up to this point has demonstrated basic procedures in getting information to the display. The following examples build on this foundation, showing solutions to typical applications. These programs are the “bells and whistles” of the display function. It’s a good idea to get the basic display screen working first, and then carefully add portions of these techniques to fine-tune your application program. The special display applications we will cover are:

- Multiple message displays
- Blinking text
- Dynamic text
- Embedding the time and date in a message
- Polarity sign for numbers
- Bar Graph Displays
- Automatic Scrolling Displays

Multiple Message Displays

In Message Mode, the DV-1000 continuously scans the numeric and text data in V-memory to combine their information and create a single display screen. To change the display message, we must update the numeric and/or text data blocks in V-memory accordingly. It’s possible to alternate between two or more display screens, based on some arbitrary event. For example, suppose we want either of the following display screens to appear, based on an operator input:

Display Screen 1

C	O	N	V	E	Y	O	R		S	P	E	E	D	S	
	L	i	n	e		1	=		1	2	3		f	p	m
	L	i	n	e		2	=		4	5	6		f	p	m
	L	i	n	e		3	=		7	8	9		f	p	m
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Display Screen 2

T	O	T	A	L		P	R	O	D	U	C	T	I	O	N
	L	i	n	e		1	=	1	1	1	1	1	1	1	1
	L	i	n	e		2	=	2	2	2	2	2	2	2	2
	L	i	n	e		3	=	3	3	3	3	3	3	3	3
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

In order to choose a programming strategy, we must do the following:

- Choose number of display screens
- Create the message (numeric and text) data
- Decide what event triggers display screen changes

In this example, we have 2 display screens, the contents are as shown, and the display will change based on the state of X0 (OFF = Display 1, ON = Display 2). An input simulator or simple switch is useful for creating switch X0. Since most of the display characters in the two display screens are different, we will use ACONs to move data in and out of the text and numeric tables. The ladder program will need to update the text characters once after each display screen change, because they remain fixed thereafter. The one-shot (positive differential) function box will help us write new display data just once. However, the numeric data must be constantly updated, because it changes in real time with the user machine or process. In this case, we will repeat the update each PLC scan.

NOTE: If using DL105, be sure to initialize the CPU or input X0 will not activate. From **PLC** menu, choose **Setup**, then **Initialize Scratchpad.**

a:\msg10.prj

(Setup Parameter rung here)

SP0

Note:

Load Values Into Numeric Data Block

T0

T0

TMRAF T0
K99999999

Use a timer (fast accumulating type) to generate a changing number. Your application will have its own sources for real-time process data, so this rung will be different. Note that this is a 32-bit timer.

X0 Off = Screen 1

X0

LD
V0

Load the timer value (which is mapped to location V0 and V1) into the accumulator. Here we only use 16 bits of the timer value.

OUT
V2005

Place the number in the numeric data block location corresponding to numeric display position 5.

OUT
V2011

Place the number in the numeric data block location corresponding to numeric display position 11.

OUT
V2015

Place the number in the numeric data block location corresponding to numeric display position 15.

X0 On = Screen 2

X0

LDD
V0

Load the timer value (which is mapped to location V0 and V1) into the accumulator. Here we use all 32 bits of the timer value.

OUTD
V2004

Place the number in the numeric data block location corresponding to numeric display position 4.

OUTD
V2010

Place the number in the numeric data block location corresponding to numeric display position 10.

OUTD
V2014

Place the number in the numeric data block location corresponding to numeric display position 14.

X0

C0
PD

Activate control relay C0 for 1 scan when X0 goes from On to Off.

X0

C1
PD

Activate control relay C1 for 1 scan when X0 goes from Off to On.

Load Screen 1 Data into Text Data Block

C0

SP0 X0

LD
K10

Load the number of data locations, which is 16 words (10 hex), or 32 characters.

LD
K0

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

LDLBL
K1

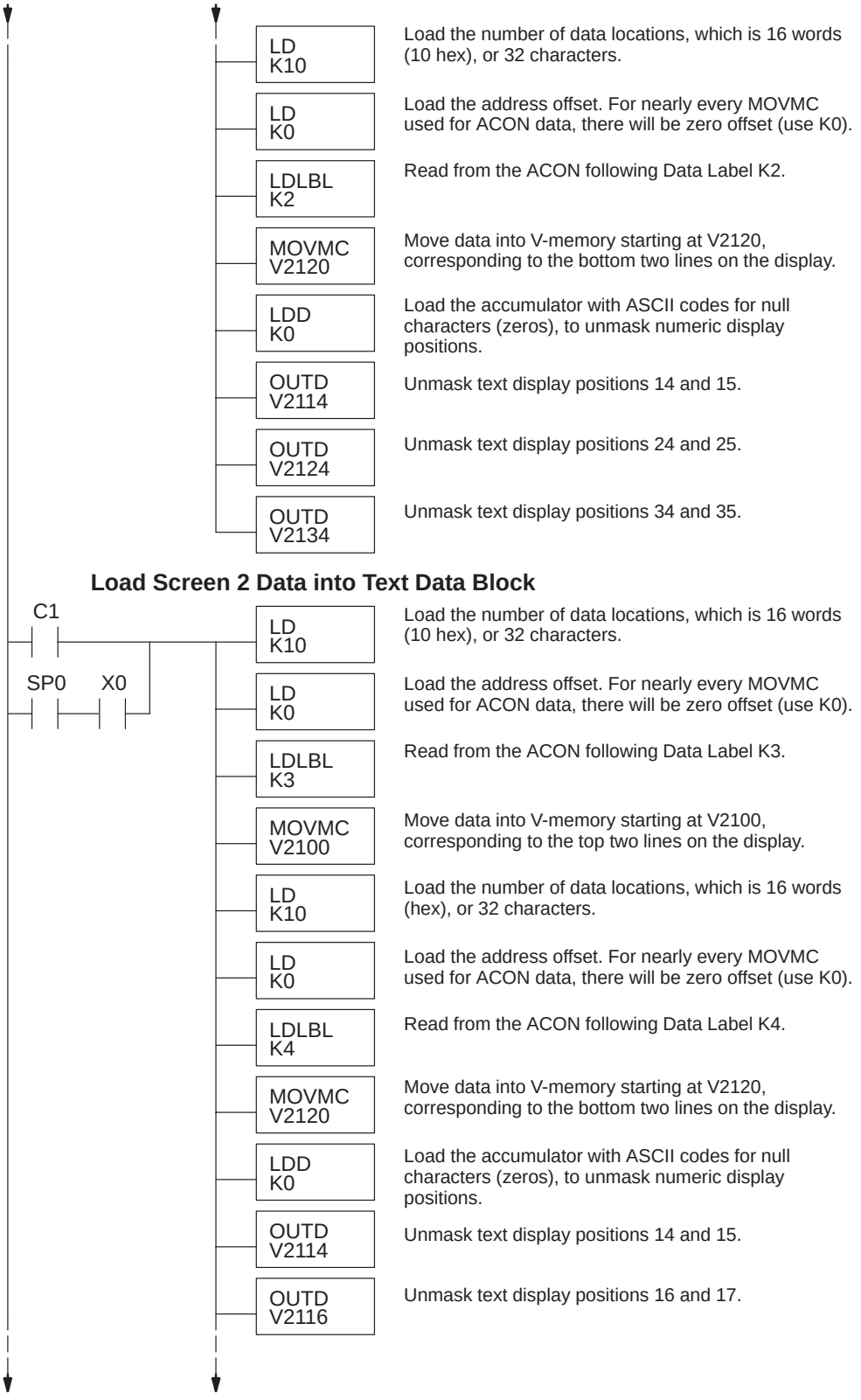
Read from the ACON following Data Label K1.

MOVMC
V2100

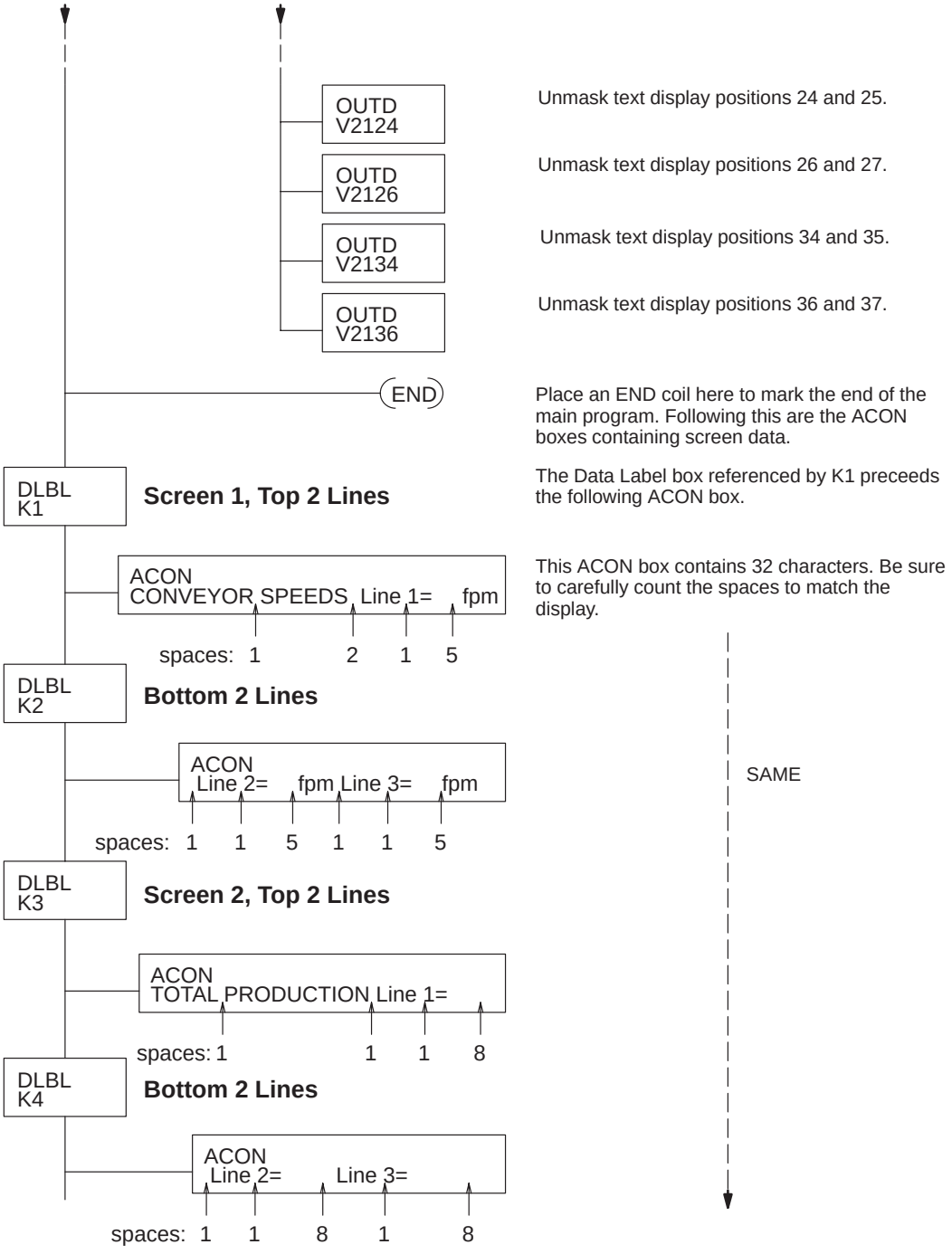
Move data into V-memory starting at V2100, corresponding to the top two lines on the display.

(continued)


a:\msg10.prj
DV-1000
Examples


LD K10
Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0
Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K2
Read from the ACON following Data Label K2.
MOVMC V2120
Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.
LDD K0
Load the accumulator with ASCII codes for null characters (zeros), to unmask numeric display positions.
OUTD V2114
Unmask text display positions 14 and 15.
OUTD V2124
Unmask text display positions 24 and 25.
OUTD V2134
Unmask text display positions 34 and 35.
Load Screen 2 Data into Text Data Block
C1
SP0
X0
LD K10
Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0
Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K3
Read from the ACON following Data Label K3.
MOVMC V2100
Move data into V-memory starting at V2100, corresponding to the top two lines on the display.
LD K10
Load the number of data locations, which is 16 words (hex), or 32 characters.
LD K0
Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K4
Read from the ACON following Data Label K4.
MOVMC V2120
Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.
LDD K0
Load the accumulator with ASCII codes for null characters (zeros), to unmask numeric display positions.
OUTD V2114
Unmask text display positions 14 and 15.
OUTD V2116
Unmask text display positions 16 and 17.

(continued)



Unmask text display positions 24 and 25.

Unmask text display positions 26 and 27.

Unmask text display positions 34 and 35.

Unmask text display positions 36 and 37.

Place an END coil here to mark the end of the main program. Following this are the ACON boxes containing screen data.

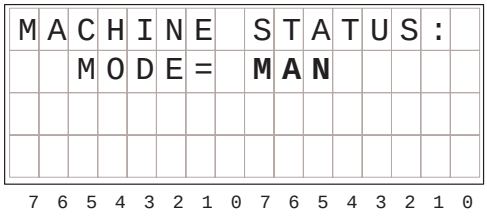
The Data Label box referenced by K1 preceeds the following ACON box.

This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

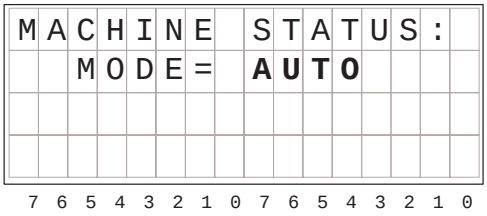
Dynamic Text

In some display screens it's desirable to change only a key word or two upon certain events, leaving most of the text intact. We might call this a "forms" type screen. The unchanging text is the form, and the dynamic text occupies blanks within the form. In this example, the display screen reports the status of a machine. The operational Manual and Automatic modes appear as "MAN", and "AUTO", as in these displays:

X0 = Off



X0 = On



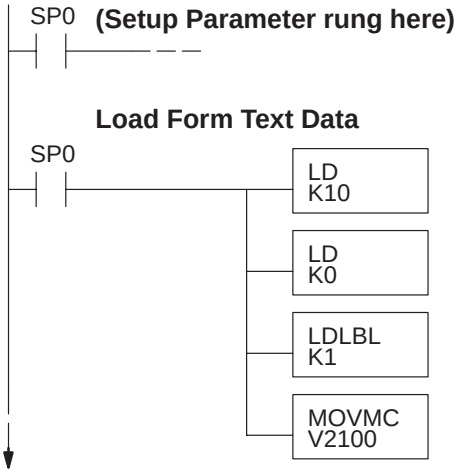
Dynamic Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Now we can decide how to write the dynamic text ladder program. We'll use the ACON text entry method for the basic form of the screen, which will include "MACHINE STATUS MODE= ". The dynamic text will occupy text positions 14 and 15.

The dynamic text will follow the state of the input switch contact X0. When X0 = Off, "MAN " will be displayed. When X0 =On, "AUTO" will be displayed. The following ladder logic outputs these screens.



Note: If using DL105 be sure CPU is initialized to make inputs X0 and X1 operational.

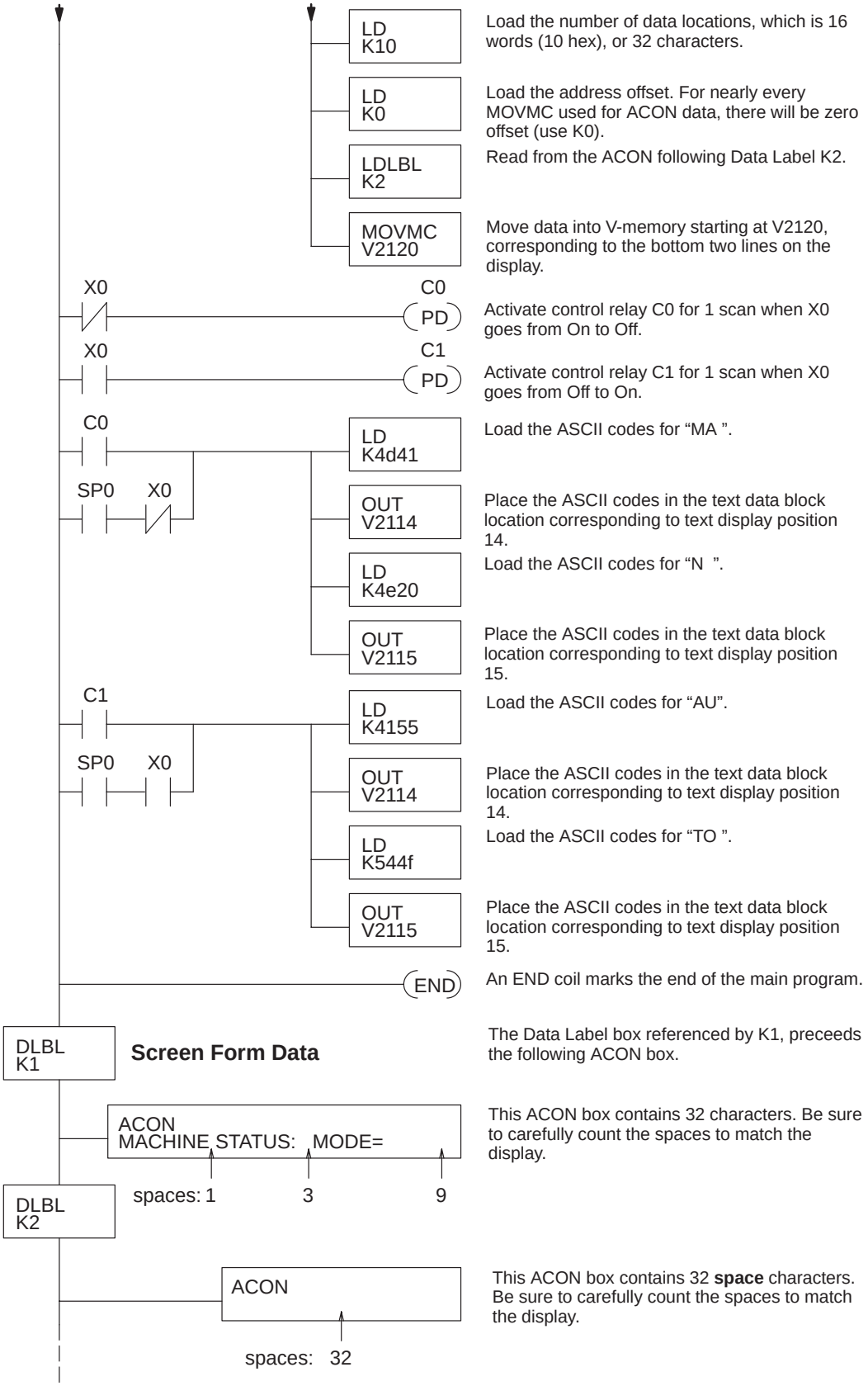
Load the number of data locations, which is 16 words (10 hex), or 32 characters.

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K1.

Move data into V-memory starting at V2100, corresponding to the top two lines on the display.

(continued)



Load the number of data locations, which is 16 words (10 hex), or 32 characters.

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K2.

Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.

Activate control relay C0 for 1 scan when X0 goes from On to Off.

Activate control relay C1 for 1 scan when X0 goes from Off to On.

Load the ASCII codes for “MA ”.

Place the ASCII codes in the text data block location corresponding to text display position 14.

Load the ASCII codes for “N ”.

Place the ASCII codes in the text data block location corresponding to text display position 15.

Load the ASCII codes for “AU”.

Place the ASCII codes in the text data block location corresponding to text display position 14.

Load the ASCII codes for “TO ”.

Place the ASCII codes in the text data block location corresponding to text display position 15.

An END coil marks the end of the main program.

The Data Label box referenced by K1, preceeds the following ACON box.

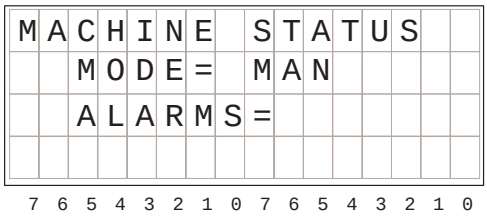
This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

This ACON box contains 32 **space** characters. Be sure to carefully count the spaces to match the display.

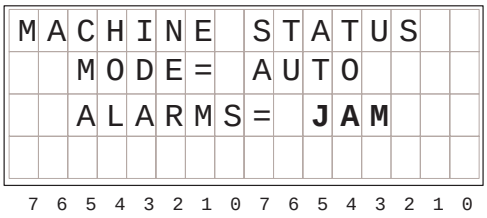
Blinking Text

Under certain conditions such as alarm or fault reporting you may want to implement blinking (or flashing) text. Building on the previous example, we add an ALARMS category. If there are no alarms, the text field after the word “ALARMS” is blank. If there is an alarm condition such as a part jam, the word ”JAM” will appear in the text field and blink at a 1 Hz rate.

X0 = Off, X1 = Off



X0 = On, X1 = On



Now we can decide how to write the blinking text ladder program. We’ll use the ACON text entry method for the basic form of the screen, which will include “MACHINE STATUS, MODE=”, and “ALARMS=”. The blinking text will occupy text positions 25 and 26.

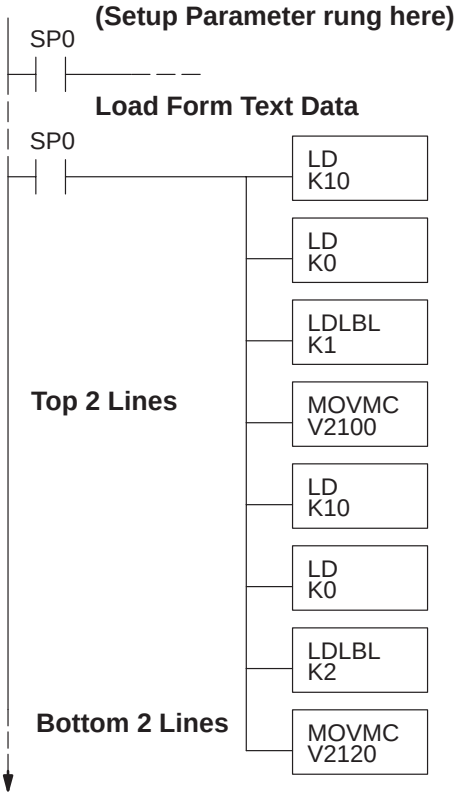
Blinking Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

The following program builds on the dynamic text example. It uses special relay SP4, which alternates between On and Off at a 1 Hz rate to create the blinking effect.

a:\msg12.prj



Note: If using DL105 be sure CPU is initialized to make inputs X0 and X1 operational.

Load the number of data locations, which is 16 words (10 hex), or 32 characters.

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K1.

Move data into V-memory starting at V2100, corresponding to the top two lines on the display.

Load the number of data locations, which is 16 words (10 hex), or 32 characters.

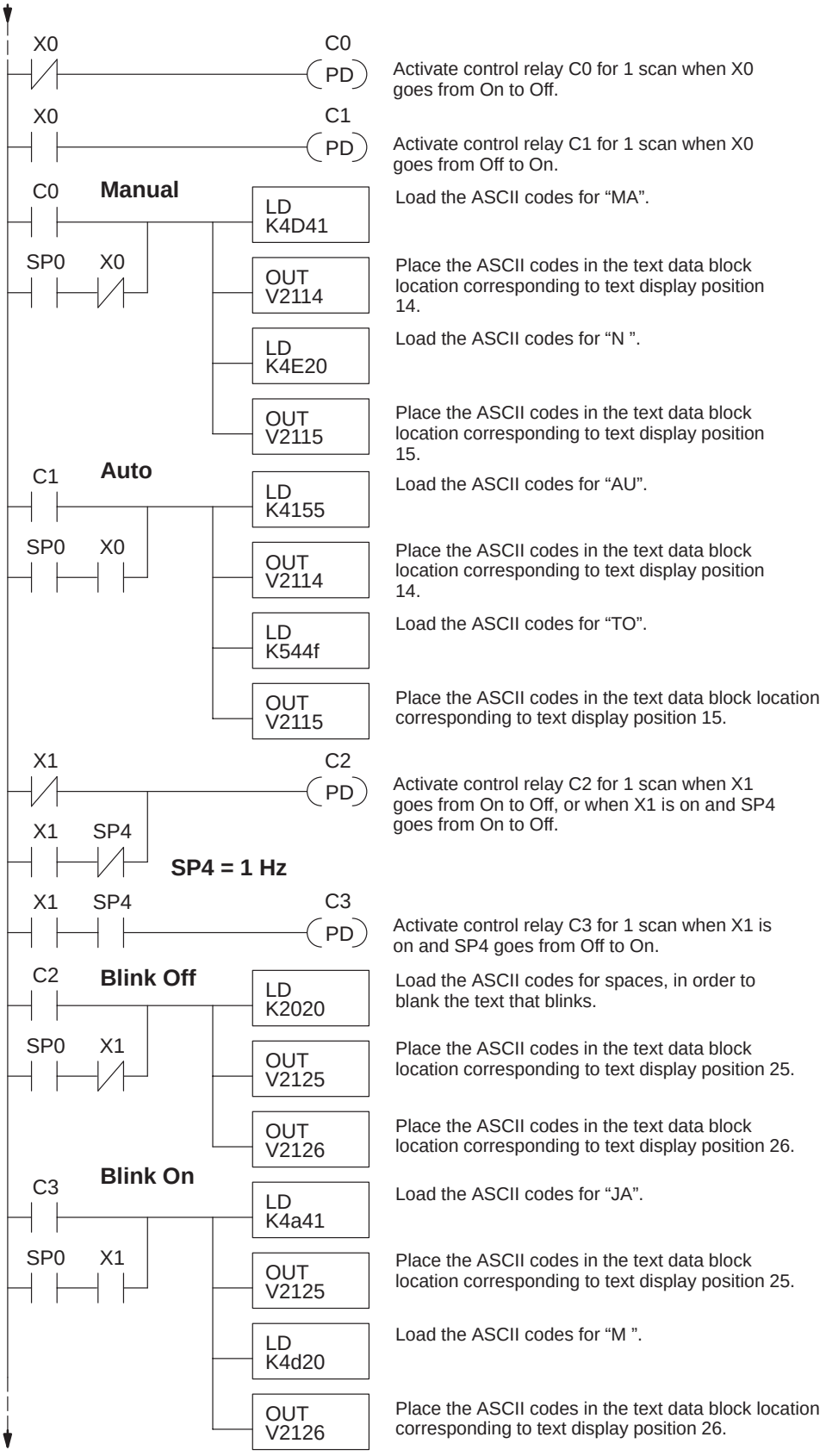
Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K2.

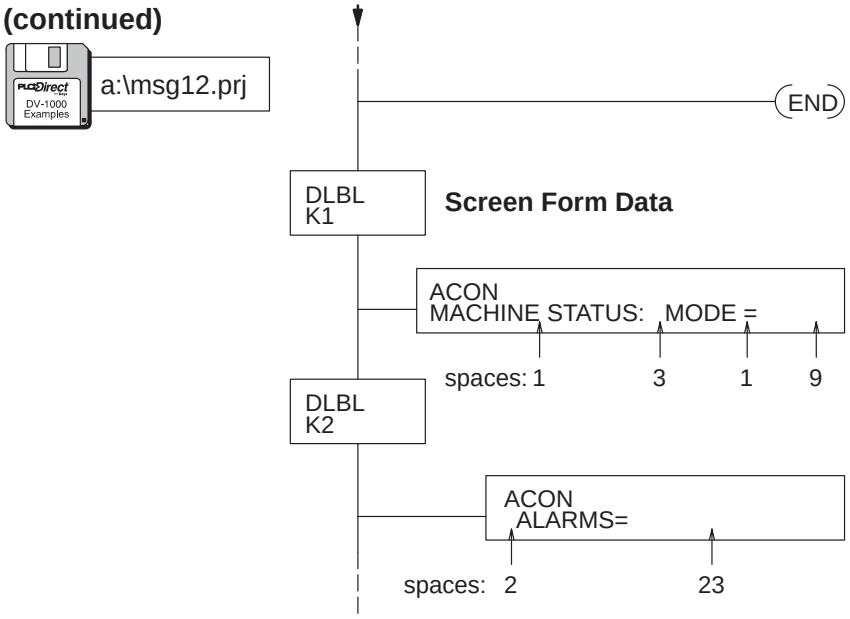
Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.

(continued)

a:\msg12.prj



(continued)



Place an END coil, marking the end of the main program.

The Data Label box referenced by K1, preceeds the following ACON box.

This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

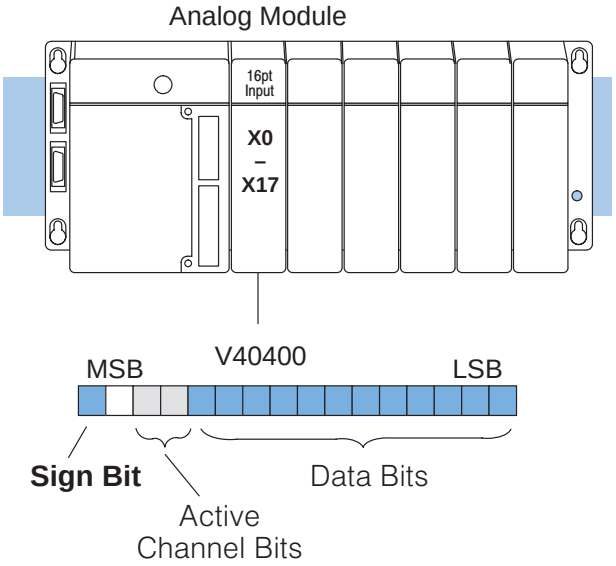
This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

Polarity Sign for Numeric Output

With the DV-1000 you can use plus (+) or minus (–) characters to indicate the algebraic sign of numeric output. To coordinate numeric and text output, the ladder program must examine the sign bit for the value and then select the appropriate polarity sign for display purposes. Sign bits often occur with bipolar inputs or outputs with analog modules, as in the following example.

To the right is a typical analog input module installation. Its input word maps to V40400. However, the input word is shared among four channels of the analog module. Only one channel is active on each scan. Ladder logic decodes the active channel bits, and stores each channel's data separately.

In this fashion, one channel's data is updated each scan . We attach to this process the task of re–writing that channel's polarity sign in the text output.



In this example, numeric display positions 0, 4, 10, and 14 show the value for each of the four channels. The remainder of the display area is text output, including the (+) and (–) characters.

A	n	a	l	o	g		C	h	1		+	3	2	9	3
I	n	p	u	t	s		C	h	2		–	2	4	1	8
							C	h	3		+	1	6	4	4
							C	h	4		–	5	7	6	5
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

The text display is created using two ACON box instructions. Then, ladder logic unmasks the eight text locations corresponding to the four numeric positions by writing null characters. Then active channel bits X14 and X15 are examined and the appropriate polarity character is written to text positions 5, 15, 25, and 35. respectively.

Numeric Display Positions

3		2		1		0	
7		6		5		4	
13		12		11		10	
17		16		15		14	
7	6	5	4	3	2	1	0

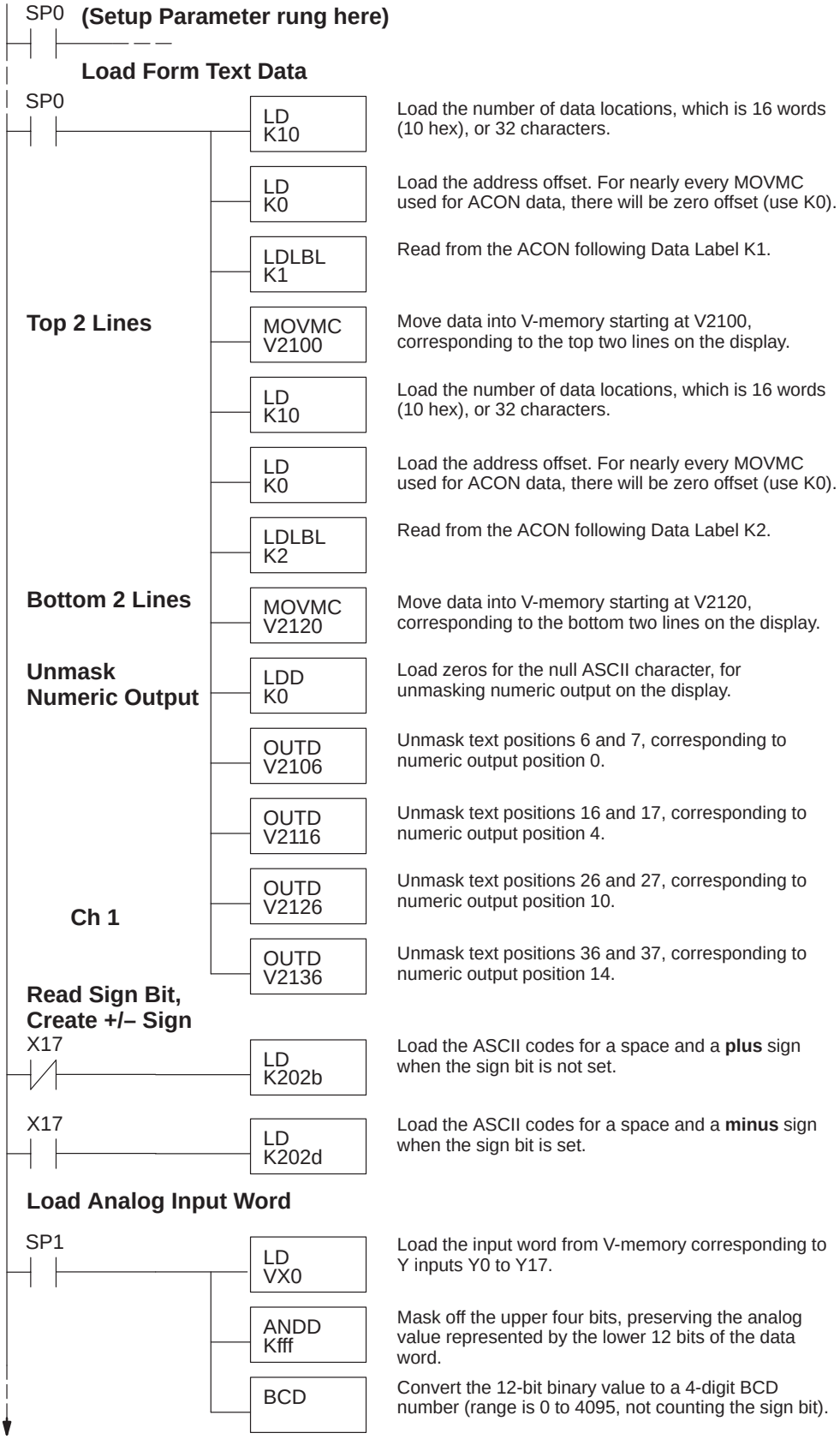
Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
7	6	5	4	3	2	1	0

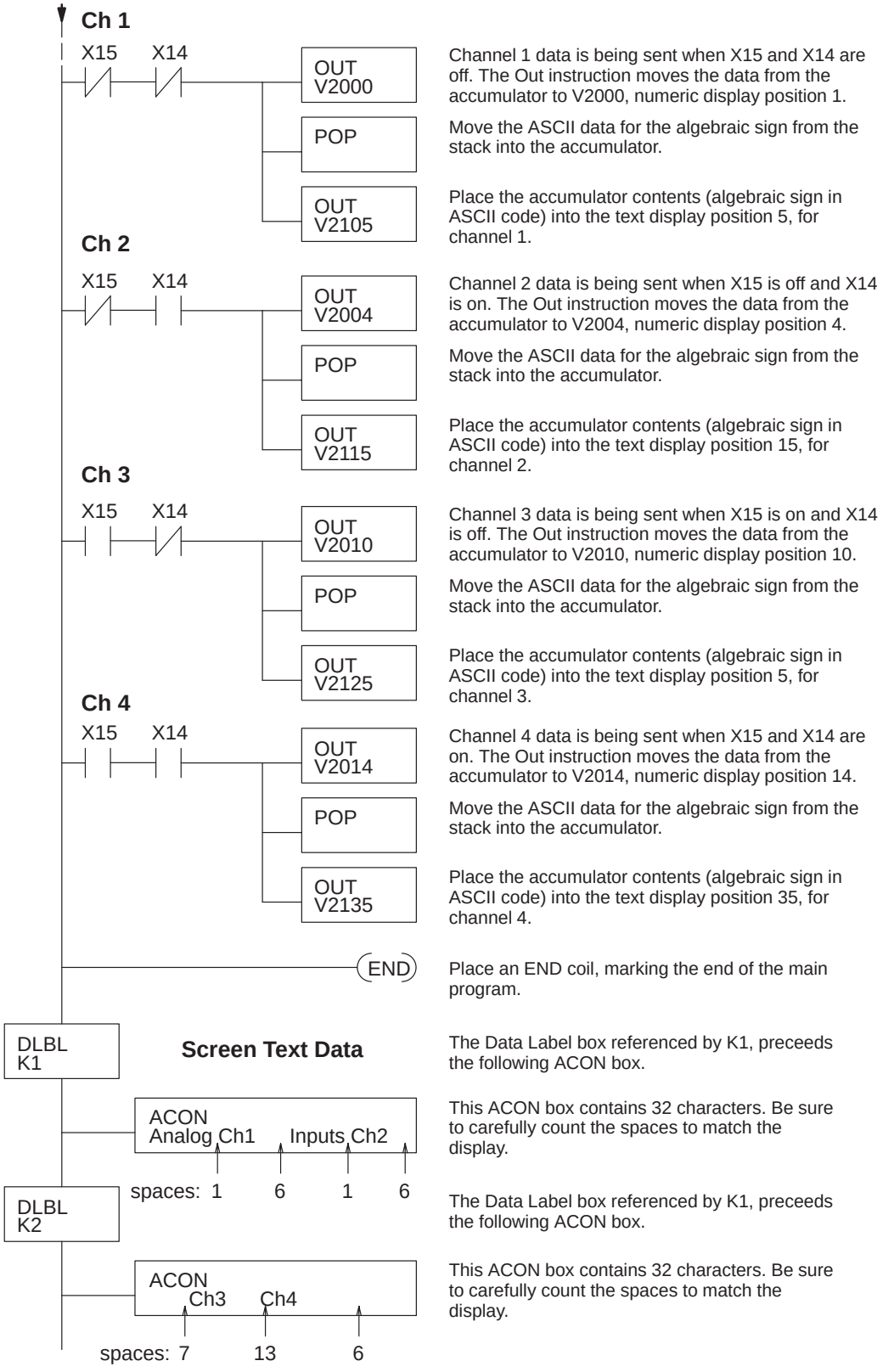
NOTE: This program example is written for the system with the analog module in slot 0. If you want to use a 16-point input simulator instead, it will also work.

The following ladder program example creates the display of signed numbers. It is written for the PLC system shown above.


a:\msg13.prj
DV-1000
Examples


SP0
(Setup Parameter rung here)
Load Form Text Data
SP0
LD K10
Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0
Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K1
Read from the ACON following Data Label K1.
Top 2 Lines
MOVMC V2100
Move data into V-memory starting at V2100, corresponding to the top two lines on the display.
LD K10
Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0
Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K2
Read from the ACON following Data Label K2.
Bottom 2 Lines
MOVMC V2120
Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.
Unmask Numeric Output
LDD K0
Load zeros for the null ASCII character, for unmasking numeric output on the display.
OUTD V2106
Unmask text positions 6 and 7, corresponding to numeric output position 0.
OUTD V2116
Unmask text positions 16 and 17, corresponding to numeric output position 4.
OUTD V2126
Unmask text positions 26 and 27, corresponding to numeric output position 10.
Ch 1
OUTD V2136
Unmask text positions 36 and 37, corresponding to numeric output position 14.
Read Sign Bit, Create +/– Sign
X17
LD K202b
Load the ASCII codes for a space and a plus sign when the sign bit is not set.
X17
LD K202d
Load the ASCII codes for a space and a minus sign when the sign bit is set.
Load Analog Input Word
SP1
LD VX0
Load the input word from V-memory corresponding to Y inputs Y0 to Y17.
ANDD Kfff
Mask off the upper four bits, preserving the analog value represented by the lower 12 bits of the data word.
BCD
Convert the 12-bit binary value to a 4-digit BCD number (range is 0 to 4095, not counting the sign bit).

(continued)



Channel 1 data is being sent when X15 and X14 are off. The Out instruction moves the data from the accumulator to V2000, numeric display position 1.

Move the ASCII data for the algebraic sign from the stack into the accumulator.

Place the accumulator contents (algebraic sign in ASCII code) into the text display position 5, for channel 1.

Channel 2 data is being sent when X15 is off and X14 is on. The Out instruction moves the data from the accumulator to V2004, numeric display position 4.

Move the ASCII data for the algebraic sign from the stack into the accumulator.

Place the accumulator contents (algebraic sign in ASCII code) into the text display position 15, for channel 2.

Channel 3 data is being sent when X15 is on and X14 is off. The Out instruction moves the data from the accumulator to V2010, numeric display position 10.

Move the ASCII data for the algebraic sign from the stack into the accumulator.

Place the accumulator contents (algebraic sign in ASCII code) into the text display position 5, for channel 3.

Channel 4 data is being sent when X15 and X14 are on. The Out instruction moves the data from the accumulator to V2014, numeric display position 14.

Move the ASCII data for the algebraic sign from the stack into the accumulator.

Place the accumulator contents (algebraic sign in ASCII code) into the text display position 35, for channel 4.

Place an END coil, marking the end of the main program.

The Data Label box referenced by K1, preceeds the following ACON box.

This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

The Data Label box referenced by K1, preceeds the following ACON box.

This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

Embedded Time and Date (DL240, DL250, DL350, DL440 and DL450 CPUs Only)

In some applications you may want to embed time and date information in a message display output. The DL240, DL250, DL350, DL440 and DL450 CPUs have built-in real-time clocks. In a typical application, the time and date are displayed when a system fault occurs, along with the fault type. The example program in this section shows you how to create the display output below. On the left, the display shows a basic text form without the fault condition. When the particular fault "Bin Empty" occurs, the display fills in the fault, time, and date fields in the form, shown on the right.

Normal Display, X0 = Off

M	A	C	H	I	N	E		S	T	A	T	U	S		
F	a	u	l	t	=										
T	i	m	e		=										
D	a	t	e		=										
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Fault Display, X0 = On

M	A	C	H	I	N	E		S	T	A	T	U	S		
F	a	u	l	t	=		B	i	n		E	m	p	t	y
T	i	m	e		=	1	1	:	3	2	:	5	7	A	M
D	a	t	e		=	0	7	/	0	5	/	9	5		
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

The program uses a combination of ACON boxes, LD/OUT and LDD/OUTD instructions to create the text portion of the display. The six numeric display positions contain both numbers and text formatting characters such as ":" or "/". In some cases, the numbers have to be shifted to align them with the formatting characters.

Numeric Display Positions

3	2	1	0
7	6	5	4
13	12	11	10
17	16	15	14

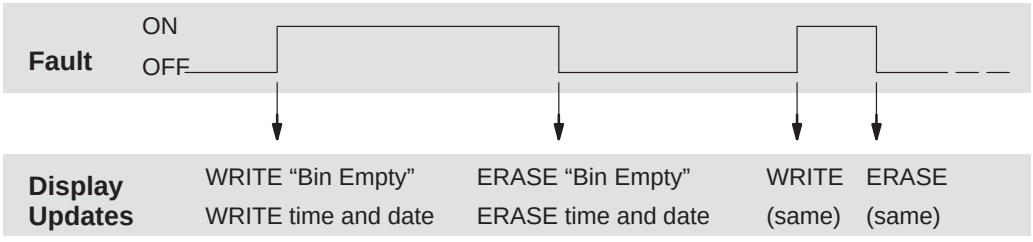
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

The following diagram shows the state of the fault input signal and the resulting display actions. Our example ladder program monitors the state of the fault signal through discrete input X0. In the Off-to-On transition, the ladder program writes "Bin Empty", and fills in the time and date data. When input X0 makes the On-to-Off transition, the program erases the "Bin Empty" message and the time and date information. Future transitions of the X0 signal cause the same display updates.

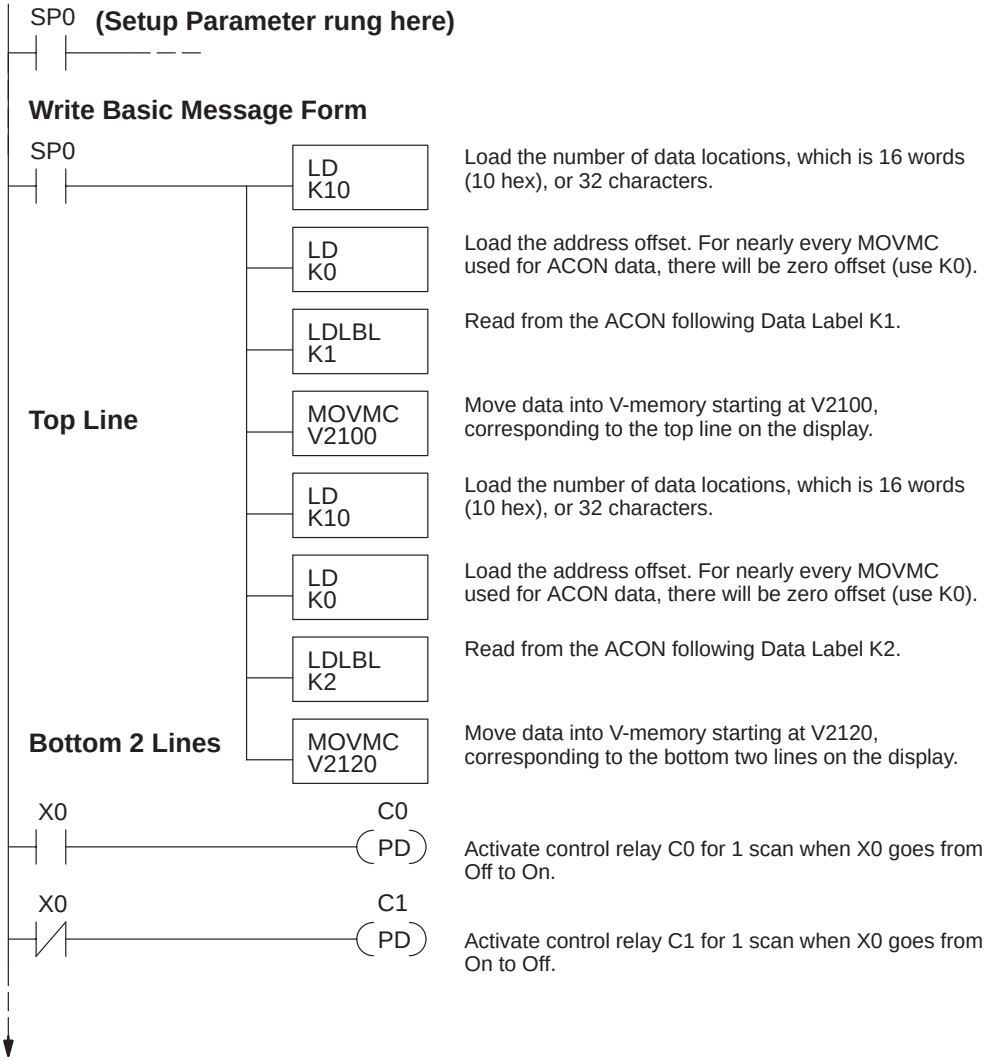


At each display update, the ladder program needs only to write the new text output data to the text data block one time. So, the program uses a PD (positive differential) coil to sense both Off-to-On and On-to-Off transitions of X0. The control relays C0 and C1 are on for just one scan, respectively. In this way the ladder program only has to display updates when the fault signal X0 makes a transition, greatly minimizing any impact to the PLC scan time.

Real-time clock data is available in system V-memory (DL240, DL250, DL350, DL440 and DL450 CPUs), at the addresses in the table below. If the current time and date in your PLC requires setting, use ***Direct***SOFT's menu "PLC", then "Settings", then "Calendar".

Time Data	Location		Date Data	Location
Hours	V7770		Day	V7772
Minutes	V7767		Month	V7773
Seconds	V7766		Year	V7774

The following program reports the time and date when the error input X0 turns on. The display clears the information when X0 turns off.



Load the number of data locations, which is 16 words (10 hex), or 32 characters.

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K1.

Move data into V-memory starting at V2100, corresponding to the top line on the display.

Load the number of data locations, which is 16 words (10 hex), or 32 characters.

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K2.

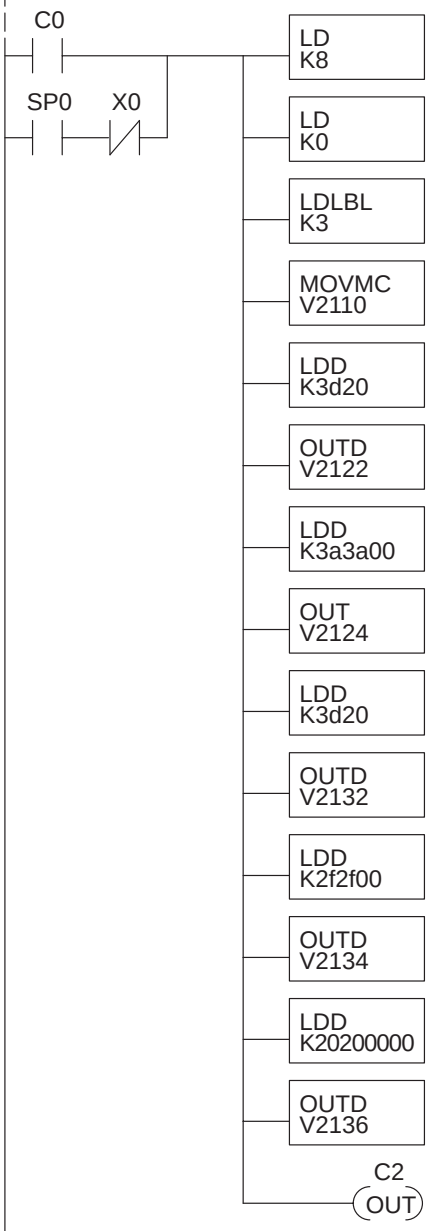
Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.

Activate control relay C0 for 1 scan when X0 goes from Off to On.

Activate control relay C1 for 1 scan when X0 goes from On to Off.

(continued)

Write Fault Message



Load the number of data locations, which is 8 words (16 characters).

Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).

Read from the ACON following Data Label K3.

Move data into V-memory starting at V2110, corresponding to the second line of the display. This writes the fault type to the display.

Load the ASCII codes two null characters (0000), followed by the ASCII codes for " =".

Place the characters " =" to the right of "Time", on the third row of the display, and unmask the two character positions for the hours digits.

Load the ASCII codes for a colon, two null characters, and another colon.

Place the colons on the third line of the display for the time output, and place the null characters so they unmask the minutes digits.

Load the ASCII codes for two null characters (0000), followed by the ASCII codes for " =".

Place the characters " =" to the right of "Date", on the fourth row of the display, and unmask the two character positions for the years digits.

Load the ASCII codes for a "/", two null characters, and another "/".

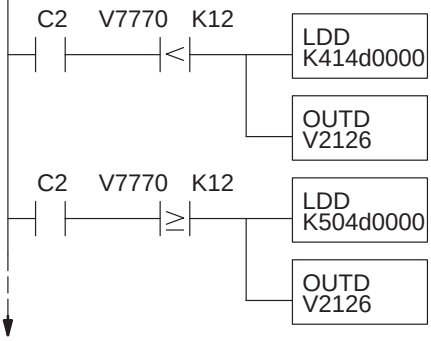
Place the "/" characters on the fourth line of the display for the date output, and place the null characters so they unmask the month digits.

Load the ASCII codes for two spaces, followed by two null characters.

Place these characters on the fourth line. The null characters unmask the year digits, and the spaces mask the last two character positions.

Use control relay contact C2 to extend the logic of this rung to more rungs, to AND with relational contacts.

Hours, AM and PM



If the hours value is less than 12, then load the ASCII codes for "AM". The null characters (0000) unmasks the minutes digits.

Place the "AM" characters to the right of the time display on the third row.

If the hours value is 12 or greater, then load the ASCII codes for "PM". The null characters (0000) unmasks the minutes digits.

Place the "PM" characters to the right of the time display on the third row.

(continued)

a:\msg14.prj

24-hr to 12-hr Clock

C2 V7770 K13 <	LD V7770	Load the hours value from the real-time clock source in V-memory into the accumulator.
	OUTD V2012	Place the hours digits in the proper numeric position on the third row.
C2 V7770 K13 ≥	LD V7770	Load the hours value from the real-time clock source in V-memory into the accumulator.
	SUBD K12	Subtract 12 from the hours value. This converts 24-hour time to 12-hour time convention.
	OUTD V2012	Place the hours digits in the proper numeric position on the third row.
C2	LD V7767	Load the minutes value from the real-time clock source in V-memory into the accumulator.
Minutes	SHFL K4	Shift the minutes digits one digit (4 bits) to the left in the accumulator.
	OUT V2011	Place the minutes digits in the proper numeric position on the third row.
Seconds	LD V7766	Load the seconds value from the real-time clock source in V-memory into the accumulator.
	SHFL K8	Shift the seconds digits two digits (8 bits) to the left in the accumulator.
	OUT V2010	Place the seconds digits in the proper numeric position on the third row.
Month	LD V7773	Load the month value from the real-time clock source in V-memory into the accumulator.
	OUT V2016	Place the month digits in the proper numeric position on the fourth row.
Day	LD V7772	Load the day value from the real-time clock source in V-memory into the accumulator.
	SHFL K4	Shift the day digits one digit (4 bits) to the left in the accumulator.
	OUT V2015	Place the day digits in the proper numeric position on the fourth row.
Year	LD V7774	Load the year value from the real-time clock source in V-memory into the accumulator.
	SHFL K8	Shift the year digits two digits (8 bits) to the left in the accumulator.
	OUT V2014	Place the year digits in the proper numeric position on the fourth row.

(continued)

a:\msg14.prj

Clear Fault Message

C1

SP0 X0

LD K20202020 — Load the ASCII codes for 4 spaces into the accumulator.

OUT V2113 — Clear text position 13.

OUTD V2114 — Clear text positions 14 and 15.

OUTD V2116 — Clear text positions 16 and 17.

OUT V2123 — Clear text position 23.

OUTD V2124 — Clear text positions 24 and 25.

OUTD V2126 — Clear text positions 26 and 27.

OUT V2133 — Clear text position 33.

OUTD V2134 — Clear text positions 34 and 35.

OUTD V2136 — Clear text positions 36 and 37.

END — Place an END coil, marking the end of the main program.

DLBL K1 — **Data for Top 2 Display Lines**

ACON MACHINE STATUS Fault= — Use the ASCII Constant (ACON) box to enter text for the top two lines of the display. It contains 32 characters, counting the spaces.

spaces: 1 2 10

DLBL K2 — **Data for Bottom 2 Display Lines**

ACON Time = Date = — Use the ASCII Constant (ACON) box to enter text for the top two lines of the display. It contains 32 characters, counting the spaces.

spaces: 1 10 1 10

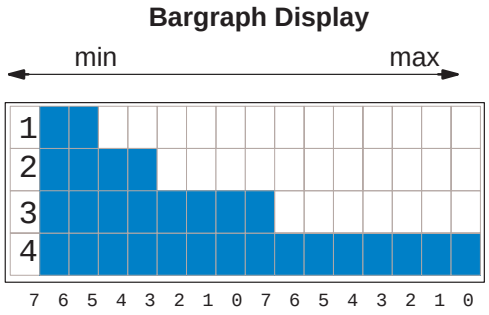
DLBL K3 — **Data for Second Display Line**

ACON Fault= Bin Empty — Use the ASCII Constant (ACON) box to enter text for the second line of the display. It contains 16 characters, counting the spaces.

spaces: 1 1

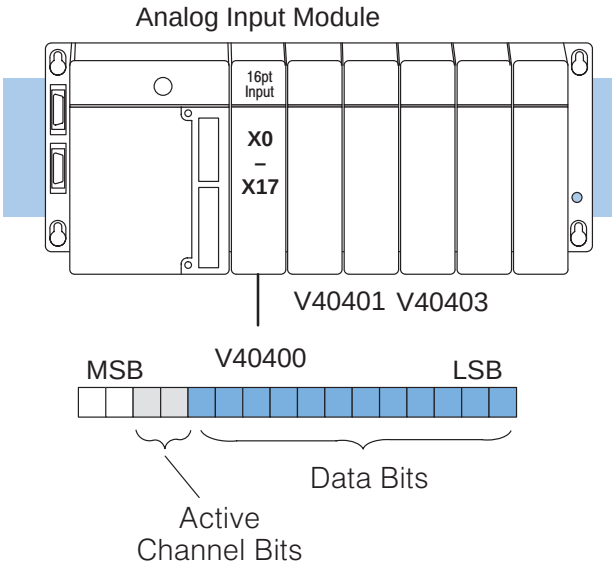
Bar Graph Example (DL240, DL250, DL350, DL440 and DL450 CPUs Only)

This example create a four-channel linear bar graph using the DV-1000's extended ASCII character set. The ASCII code FF (hex) produces a solid block character. The ladder program generates text output to display a row of these characters whose length is proportional to a numerical value. The remainder of each row consists of spaces (20 hex). This creates the bar graph effect.



The figure to the right shows a analog input module in the first module slot in the base. Its points map to X0 to X17, corresponding to data word V40400 in V-memory. Its input word is shared among the four channels of the analog module. Only one channel is active on each scan. Ladder logic decodes the active channel bits, and store each channel's data separately.

The bar graph display update is independent from the channel value update process.



The ladder program to create this display updates only one text position (two characters) per CPU scan. The following table lists location of analog values and pointers the program uses to keep track of where it is in the overall display update. Since there are 64 characters (32 text positions) in the display, the program is able to update the entire display every 32 CPU scans. The program adds approximately five milliseconds to the scan time of a DL240 CPU.

Variable	Location	Range of values
Raw Analog values	V2200 – V2203	0000 – FFFF
Processed Analog Values	V2204 – V2207	0001 – 0010
Character Pointer	V2210	0 – 15
Row Pointer	V2211	0 – 3
Raw Analog Pointer	V2212	O2200 – O2203
Processed Analog Pointer	V2213	O2204 – O2207
Text Data Pointer	V2214	O2100 – O2137
Scratchpad number	V2215	–

(Setup Parameter rung here)

SP0

Load Channel Data Values

SP1 — LD V40400

Load the contents of V40400 (X0 to X17) into the accumulator.

Ch 1

X15 X14 — OUT V2200

Channel 1 data is being sent when X15 and X14 are off. The Out instruction places the data in V2200, storing the raw data for channel 1.

Ch 2

X15 X14 — OUT V2201

Channel 2 data is being sent when X15 is off and X14 is on. The Out instruction places the data in V2201, storing the raw data for channel 2.

Ch 3

X15 X14 — OUT V2202

Channel 3 data is being sent when X15 is on and X14 is off. The Out instruction places the data in V2202, storing the raw data for channel 3.

Ch 4

X15 X14 — OUT V2203

Channel 4 data is being sent when X15 and and X14 are off. The Out instruction places the data in V2203, storing the raw data for channel 4.

Initialize Variables

SP0 — LD K0

Load the constant K0 into the accumulator.

First scan only

OUT V2210

Initialize the character pointer to zero. This points to one of 16 characters on the current row.

OUT V2211

Initialize the row pointer to zero. This points to one of 4 rows of characters on the display.

LDA O2200

Load the octal address 2200 into the accumulator. V2200 is the location of the raw analog data.

OUT V2212

Initialize the raw analog value pointer at V2212 to point to the first analog value at V2200.

LDA O2204

Load the octal address 2204 into the accumulator. V2204 is the location of the processed analog data.

OUT V2213

Initialize the raw analog value pointer at V2213 to point to the first analog value at V2204.

LDA O2100

Load the octal address 2100 into the accumulator. V2100 is the beginning of display text data.

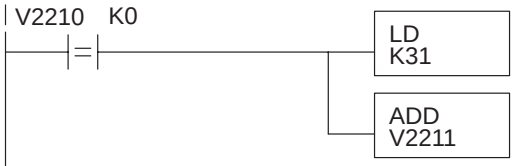
OUT V2214

Initialize the display text pointer at V2214 to point to the first text display location at V2100.

(continued)

a:\msg15.prj

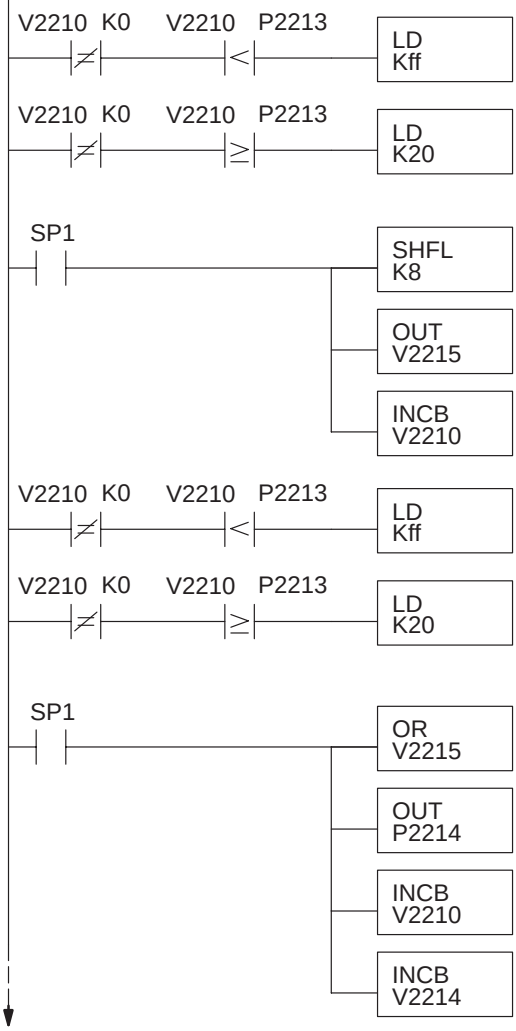
Write Channel Labels



Load the ASCII constant for “1” into the accumulator. We will modify this with an offset to create numbers 1 through 4.

Use the row data as an offset for the ASCII codes. This will number the display as rows 1 through 4.

Write Bar Graph Components



If the processed analog value is less than the character pointer, load the ASCII code for the solid block character (bar graph component) into the accumulator.

If the processed analog value is equal to or greater than the character pointer, load the ASCII code for the space character (blanks bar graph) into the accumulator.

Move the “ff” or “20” into the second nibble of the accumulator.

Save the accumulator contents at V2215 while we calculate the lower nibble.

Increment the character pointer, in order to make the next compare.

If the processed analog value is less than the character pointer, load the ASCII code for the solid block character (bar graph component) into the accumulator.

If the processed analog value is equal to or greater than the character pointer, load the ASCII code for the space character (blanks bar graph) into the accumulator.

Combine the contents of the accumulator (lower nibble) with the saved contents of V2215 (upper nibble). We have a word now ready to write.

Write the bargraph word (2 characters) to the proper text position’s address, *pointed to* by V2214.

Increment the character pointer, in order to make the next compare.

Increment the text data pointer, in order to write to the next text position on the display.

(continued)

End of Row

V2210 K10 =

LD K0	Load “0” into the accumulator.
OUT V2210	Reset the character pointer at V2210 to zero.
INCB V2211	Increment the row pointer at V2211.
LD P2212	Load the next raw analog value into the accumulator for processing.
ANDD kfff	Mask off any bits above the lower 12 bits, such as channel select numbers, etc. This asumes 12-bit analog values.
SHFR K8	Discard the lower 8 bits, and use the upper 4 bits to scale the value from 0 to 15 (0 to f hex).
OUT P2213	Write the processed analog value to the proper address, *pointed to* by V2213.
INCB P2213	Increment the processed analog value pointer.
INCB V2212	Increment the raw analog value pointer.
INCB V2213	Add one to the processed analog value, to scale it from 1 to 16 (1 to 10 hex).

End of Display

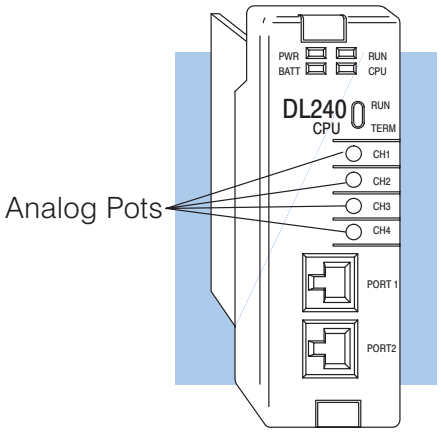
V2211 K4 =

LDA O2200	Load the beginning address of the raw analog values into the accumulator.
OUT V2212	Reset the raw analog value pointer at V2212 to point to V2200.
LDA O2204	Load the beginning address of the processed analog values into the accumulator.
OUT V2213	Reset the processed analog value pointer at V2213 to point to V2204.
LDA O2100	Load the beginning address of the text data block into the accumulator.
OUT V2214	Reset the text data pointer at V2214 to point to V2100.
LD K0	Load the constant zero into the accumulator.
OUT V2210	Reset the character pointer to zero.
OUT V2211	Reset the row pointer to zero.

END — Place an END coil, marking the end of the program.

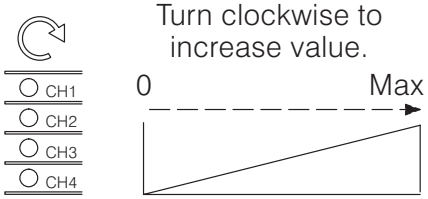
Bar Graph Example, DL240 Analog Potentiometers

The DL240 CPU features four built-in analog potentiometers. In this example we modify the previous bar graph program to read these four inputs. The potentiometers are accessible on the front bezel with a small screwdriver. These potentiometers map directly into system V-memory as 8-bit numbers, having a resolution of 1 part in 256.

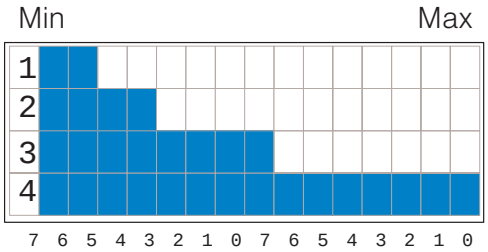


The ladder logic program takes the range of 0 to 255 and re-scales it to 0 to 15. In this way, the number of segments on each row can vary from 0 to 15, in proportion to the potentiometer adjustment.

Potentiometer Adjust



Display Output



To see this program work, just load the file from disk, and place the PLC in Run Mode.

SP0 **(Setup Parameter rung here)**

Set Up Analog Potentiometer Scalings

SP0 — **First scan only**

LD K0	Load the constant K0 into the accumulator.
OUT V7640	Set the bottom of Channel 1 range to zero.
OUT V7642	Set the bottom of Channel 2 range to zero.
LDA V7644	Set the bottom of Channel 3 range to zero.
OUT V7646	Set the bottom of Channel 4 range to zero.
LD K15	Load the constant K15 into the accumulator.
OUT V7641	Set the top of Channel 1 range to 15.
LDA V7643	Set the top of Channel 2 range to 15.
OUT V7645	Set the top of Channel 3 range to 15.
OUT V7647	Set the top of Channel 4 range to 15.

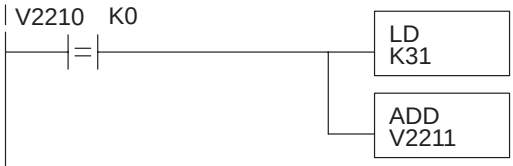
Initialize Variables

SP0 — **First scan only**

LD K0	Load the constant K0 into the accumulator.
OUT V2210	Initialize the character pointer to zero. This points to one of 16 characters on the current row.
OUT V2211	Initialize the row pointer to zero. This points to one of 4 rows of characters on the display.
LDA O3774	Load the octal address 3774 into the accumulator. V3774 is the location of the analog data.
OUT V2212	Initialize the raw analog value pointer at V2212 to point to the first analog value at V2200.
LDA O2204	Load the octal address 2204 into the accumulator. V2204 is the location of the processed analog data.
OUT V2213	Initialize the raw analog value pointer at V2213 to point to the first analog value at V2204.
LDA O2100	Load the octal address 2100 into the accumulator. V2100 is the beginning of display text data.
OUT V2214	Initialize the display text pointer at V2214 to point to the first text display location at V2100.

(continued)

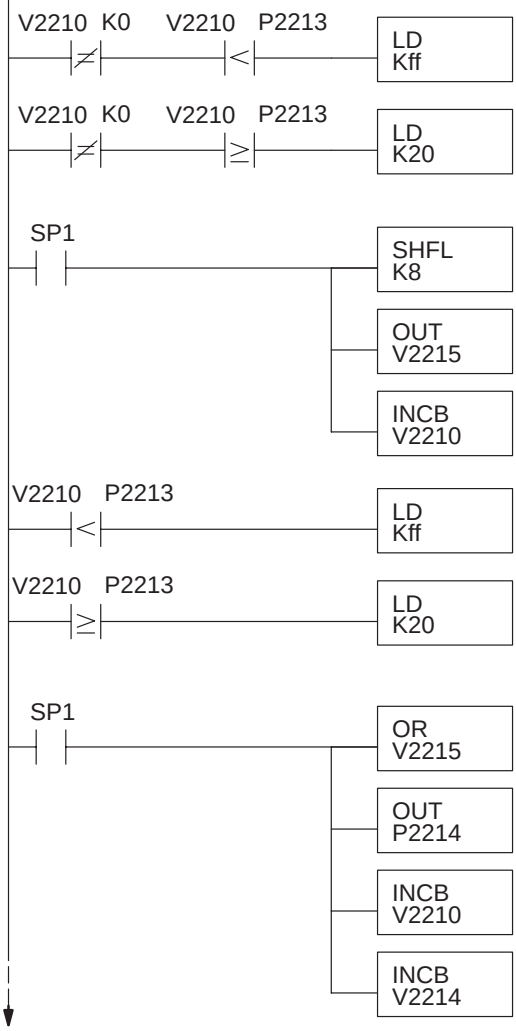
Write Channel Labels



Load the ASCII constant for “1” into the accumulator. We will modify this with an offset to create numbers 1 through 4.

Use the row data as an offset for the ASCII codes. This will number the display as rows 1 through 4.

Write Bar Graph Components



If the processed analog value is less than the character pointer, load the ASCII code for the solid block character (bar graph component) into the accumulator.

If the processed analog value is equal to or greater than the character pointer, load the ASCII code for the space character (blanks bar graph) into the accumulator.

Move the “ff” or “20” into the second nibble of the accumulator.

Save the accumulator contents at V2215 while we calculate the lower nibble.

Increment the character pointer, in order to make the next compare.

If the processed analog value is less than the character pointer, load the ASCII code for the solid block character (bar graph component) into the accumulator.

If the processed ananlog value is equal to or greater than the character pointer, load the ASCII code for the space character (blanks bar graph) into the accumulator.

Combine the contents of the accumulator (lower nibble) with the saved contents of V2215 (upper nibble). We have a word now ready to write.

Write the bargraph word (2 characters) to the proper text position’s address, *pointed to* by V2214.

Increment the character pointer, in order to make the next compare.

Increment the text data pointer, in order to write to the next text position on the display.

(continued)

End of Row

V2210 K10 =

LD K0	Load “0” into the accumulator.
OUT V2210	Reset the character pointer at V2210 to zero.
INCB V2211	Increment the row pointer at V2211.
LD P2212	Load the next raw analog value into the accumulator for processing.
BIN	Convert the accumulator contents to binary.
OUT P2213	Write the processed analog value to the proper address, *pointed to* by V2213.
INCB P2213	Increment the processed analog value pointer.
INCB V2212	Increment the raw analog value pointer.
INCB V2213	Add one to the processed analog value, to scale it from 1 to 16 (1 to 10 hex).

End of Display

V2211 K4 =

LDA O3774	Load the beginning address of the analog values into the accumulator.
OUT V2212	Reset the raw analog value pointer at V2212 to point to V2200.
LDA O2204	Load the beginning address of the processed analog values into the accumulator.
OUT V2213	Reset the processed analog value pointer at V2213 to point to V2204.
LDA O2100	Load the beginning address of the text data block into the accumulator.
OUT V2214	Reset the text data pointer at V2214 to point to V2100.
LD K0	Load the constant zero into the accumulator.
OUT V2210	Reset the character pointer to zero.
OUT V2211	Discard the lower 8 bits, and use the upper 4 bits to scale the value from 0 to 15 (0 to f hex).

(END) Place an END coil, marking the end of the program.

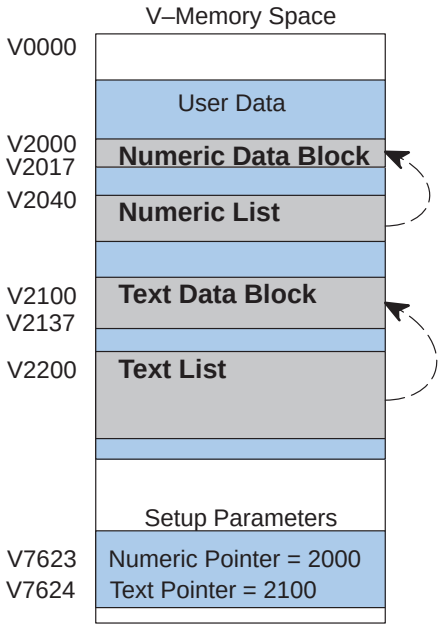
Automatic Scrolling Display (DL240, DL250, DL350, DL440 and DL450 CPUs Only)

Occasionally you may want to monitor a list of process parameters larger than the DV-1000 display can show at one time. Earlier in this chapter a “dual display” example switched between two sets of information. In this example, the automatic scrolling technique provides hands-free monitoring of multiple screens of information.

The list to the right contains 10 items, followed by a blank line item. The example program scrolls down the list one item every two seconds, and starts again with the first item.

Lot Number	3293
Setpoint	2418
Actual Temp	1644
High Alarm	5765
Low Alarm	1644
Tank 1 Lvl	5765
Tank 2 Lvl	1644
Flow Rate	5765
Temp 1	1644
Temp 2	8219

The information to be displayed will obviously not fit in the standard numeric and text data blocks. Consequently, we have to choose a method of moving the data into the standard data blocks. Refer to the memory map to the right.



Setup Parameters define the standard numeric block (at V2000) and text data block (at V2100) as in the other examples in this chapter. However, a separate numeric list (at V2040) and text list (at V2200) contains the all information to be scrolled through the display. Therefore, the main program moves a different portion of the lists into the corresponding data blocks every two seconds. Since the DV-1000 constantly re-reads the data blocks (pointed to by the setup parameters), it completes the scrolling effect.

To scroll the display message, the ladder program on the next page creates and uses the following variables to manage the data in the scrolling process.

Variable	Location	Range of values
List Item Pointer	V4000	0 to X
Item Pointer During Scroll	V4001	X to (X+3)
Text List Address Offset	V4002	0 to 40 (hex)
Text List Address Pointer	V4003	2200 to 3010 (octal)
Numeric Data Block Address Pointer	V4004	2000 to 2014 (octal)
Numeric List Address Pointer	V4005	2040 to (2040 +X) (octal)
List Length (called “X”)	V4006	1 to 20 (hex)

SP0 **(Setup Parameter rung here)**

Initialize Variables

SP0

First scan only

LD Kb	Load the constant Kb hex (or 11 decimal) into the accumulator.
OUT V4006	Set the list length at V4006 to 11 items.
LD K58	Load the constant K58 (hex) into the accumulator. This is the number of words (88 decimal) we need to move to create the text data list. Counting the words in the ACON boxes, we have 20+20+20+20+8 = 88.
LD K0	Load the constant K0 into the accumulaor. This is the offset for MOVMC, and is usually zero.
LDLBL K1	Load the data label K1 into he accumulator. The data is after the label K1.

Move text data

MOVMC V2200	Move the data to the V-memory area starting at V2200. This instruction uses the three numbers from the stack which we loaded above.

Unmask numeric positions

LDD K0	Load the constant K0 into all 32 bits of the accumulator.
OUT V4000	Initialize the item number.
OUTD V2206	Unmask the numeric position for item 1.
OUTD V2216	Unmask the numeric position for item 2.
OUTD V2226	Unmask the numeric position for item 3.
OUTD V2236	Unmask the numeric position for item 4.
OUTD V2246	Unmask the numeric position for item 5.
OUTD V2256	Unmask the numeric position for item 6.
OUTD V2266	Unmask the numeric position for item 7.
OUTD V2276	Unmask the numeric position for item 8.
OUTD V2306	Unmask the numeric position for item 9.
OUTD V2316	Unmask the numeric position for item 10.

Set Up Scroll Timer

T0

TMR T0 K20	Configure Timer 0 as a two-second self-resetting timer.

(continued)

Initialize Variables for New Display

T0 — C0 (SET)

Set C0 every 2 seconds. C0 is active for four scans and is then reset. Contact SP0 ensures we update the display starting with the first scan.

SP0

LDA O2100	Load the octal address for V2100 into the accumulator.
OUT V4003	Set the text data pointer at V4003 to point to V2100.
LDA O2000	Load the octal address for V2000 into the accumulator.
OUT V4004	Set the numeric data pointer at V4004 to point to V2000.
LD V4000	Load the current item pointer into the accumulator.
OUT V4001	Copy the list item pointer into V4001, the item pointer used during the scroll.
INCB V4000	Increment the list item pointer at V4000.

V4000 = V4006

LD K0	If the list item pointer at V4000 equals the list length at V4006, load zero into the accumulator.
OUT V4000	Set the list item pointer at V4000 to zero.

Move Data for Nth Item

C0

LD V4001	If C0 is on, a scroll is in progress. Load the item number during a scroll into the accumulator.

Move Text Data

SHFL K3	Multiply the item number by eight by shifting its binary form to the left three digits. This creates an address offset.
BCD	Convert the number in the accumulator to BCD format.
OUT V4002	Place the address offset equal to eight times the item pointer into the text data address offset pointer location at V4002.
LD K8	Load the constant K8 into the accumulator, since we need to move 8 words (16 characters per row).
LDA O2200	Load the octal address for V2200 into the accumulator. This is the base address of the source data for the text ASCII codes.
BCD	Convert the number in the accumulator to BCD format.
ADD V4002	Add the base address in the acumulator with the offset in V4002, to get the complete source address of the text data.
BIN	Convert the BCD number in the accumulator to binary format.
MOV P4003	Move eight words (16 characters) from the source address to the destination address.

(continued)

a:\msg17.prj

Move Numeric Data

LD V4001	Load the list item number at location V4001 into the accumulator.
BCD	Convert the number in the accumulator to BCD.
OUT V4001	Place the converted value back into location V4001.
LDA O2040	Load the list item pointer into the accumulator. This creates a source address for numeric list data.
BCD	Convert the number in the accumulator to BCD format.
ADD V4001	Add the list item number during the scroll at V4001 to the accumulator contents.
BIN	Convert the BCD number in the accumulator to binary format.
OUT V4005	Place the newly calculated numeric list source address pointer into its location at V4005.
LD V4001	Load the list item number at location V4001 into the accumulater.
BIN	Convert the BCD number in the accumulator to binary format.
OUT V4001	Place the converted value back into location V4001.
LD P4005	Load the numeric list data pointed to by V4005 into the accumulator.
OUT P4004	Place the data into the numeric data block destination, pointed to by the address pointer at V4004.

Increment Text Address Pointer

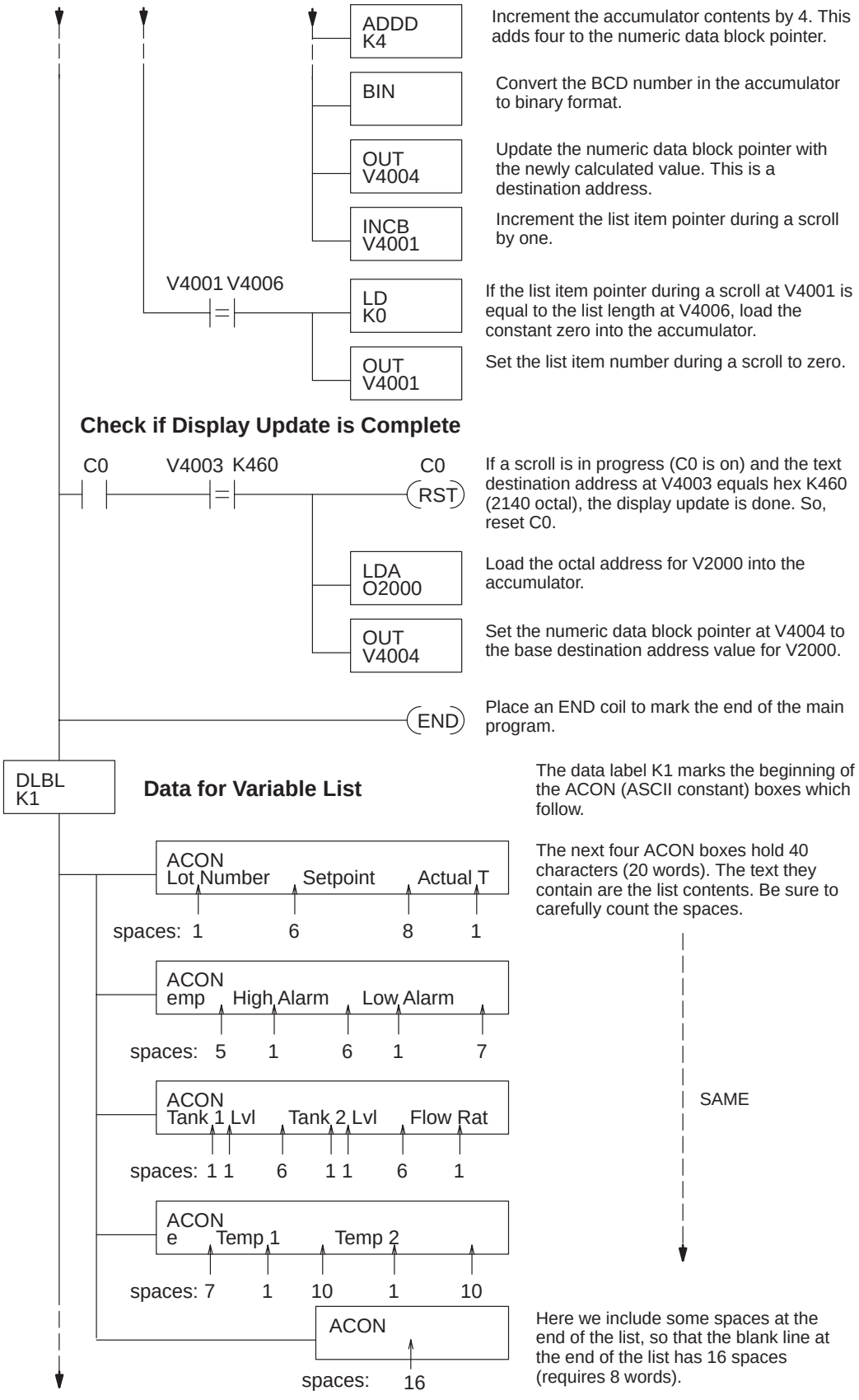
LD V4003	Load the current text data pointer value into the accumulator.
BCD	Convert the number in the accumulator to BCD format.
ADDD K8	Increment the accumulator contents by 8. This adds eight to the text data pointer.
BIN	Convert the BCD number in the accumulator to binary format.
OUT V4003	Update the text data pointer with the newly calculated value. This is a source address.

Increment Numeric Address Pointer

LD V4004	Load the current numeric data block pointer into the accumulator.
BCD	Convert the number in the accumulator to BCD format.

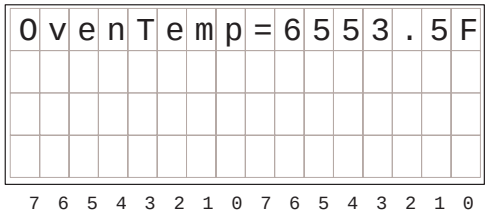
(continued)

a:\msg17.prj



Embedded Decimal Point

Some process variables will include a fractional part that you may want to display by using a decimal point. The example screen to the right displays an oven temperature with a resolution of tenths of a degree Fahrenheit.



The decimal point occupies a character position in the middle of the number, because separate decimal point segments are not built in to each character or digit field. The ladder program begins with a binary number which can vary from 0 to ffff hex, or 0 to 65535 decimal. The number is an integer, but is actually ten times larger than the actual oven temperature in degrees. This creates an implied decimal point in the location shown, so we can display tenths of a degree. The ladder program reads the oven temperature from V2200, then converts it to BCD and displays it in the appropriate numeric output display positions.

Numeric Display Positions

3	2	1	0
7	6	5	4
13	12	11	10
17	16	15	14

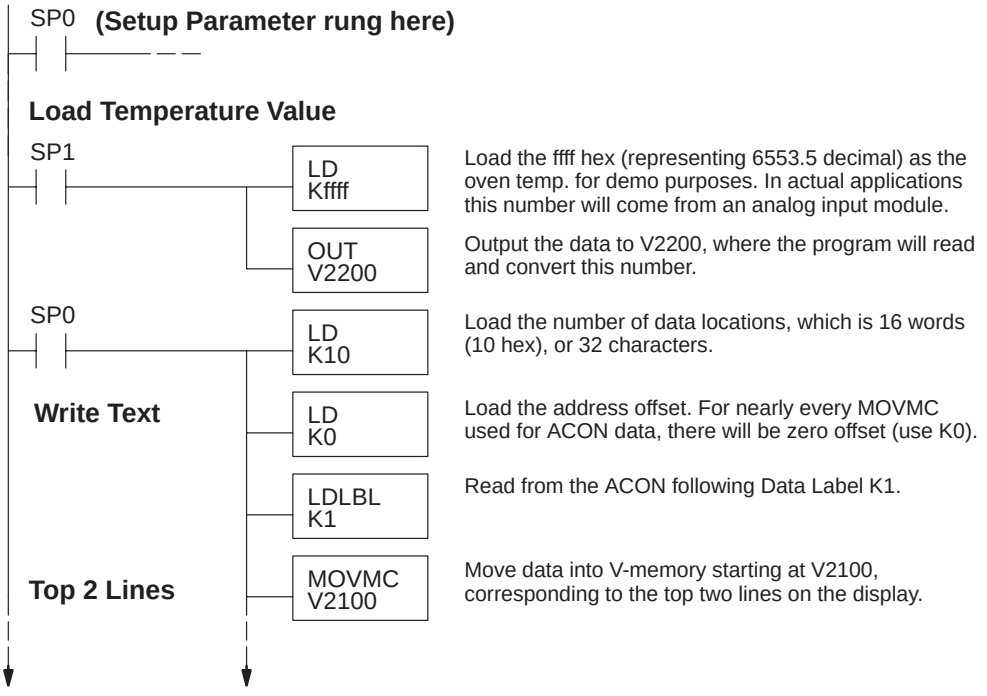
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

Text Display Positions

0	1	2	3	4	5	6	7
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

The program uses ACON boxes to generate text for “OvenTemp” and the remaining blank space on the display. The decimal point and the “F” character are written specifically to text positions embedded with null characters (ASCII code = 0) which unmasks the digits. Shift and add instructions split apart the number’s digits to place them in numeric display positions 0 and 1, making room for the decimal point.



(continued)

a:\msg18.prj

Bottom 2 Lines

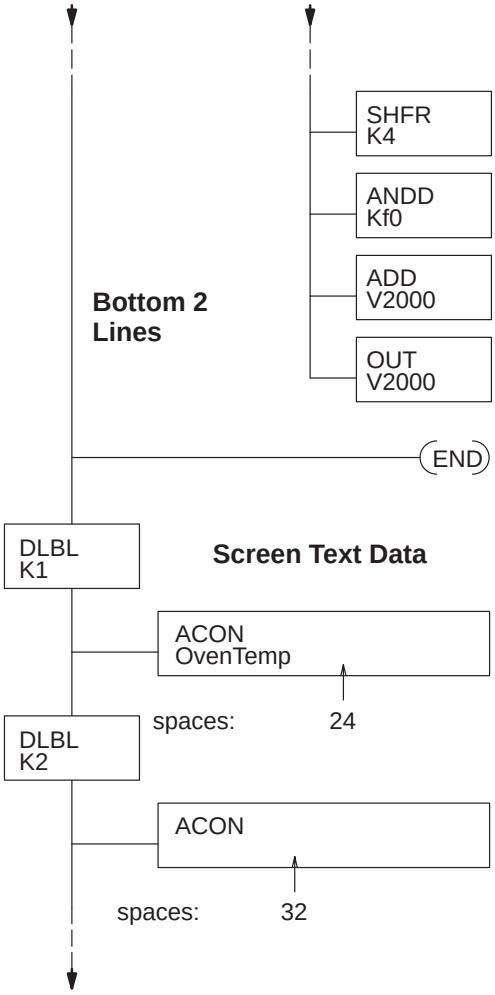
LD K10	Load the number of data locations, which is 16 words (10 hex), or 32 characters.
LD K0	Load the address offset. For nearly every MOVMC used for ACON data, there will be zero offset (use K0).
LDLBL K2	Read from the ACON following Data Label K2.
MOVMC V2120	Move data into V-memory starting at V2120, corresponding to the bottom two lines on the display.
LDD K3d00	Load ASCII codes for two null characters (0000), followed by "=" and a null character. Remember, the two pairs of character positions swap for LDD instructions.
OUTD V2104	Place the characters to the right of "OvenTemp" on the top row. Remember that null characters unmask their corresponding numeric positions.
LDD K46002e	Load the ASCII codes for a decimal point (period), two null characters, and the "F". Remember, the two pairs of character positions swap for LDD instructions.
OUTD V2106	Place these characters on the top line of the display. The null characters unmask numeric output in the corresponding character positions.

SP1

Get Temp. Value

LD V2200	Load the binary temperature value from V2200 into the accumulator.
BCD	Convert the value to BCD, leaving the result in the accumulator.
SHFR K8	Shift the bits to the right 8 places, representing two BCD digits. This leaves the three most significant digits in the accumulator.
OUT V2001	Place the three most significant digits in numeric position 1, located at V2001.
LD V2200	Load the binary temperature value from V2200 into the accumulator.
BCD	Convert the value to BCD, leaving the result in the accumulator.
SHFL K8	Shift the bits to the left 8 places, representing two BCD digits. This moves the two least significant digits to the left two places.
OUT V2000	Place the two least significant digits in numeric position 0, located at V2000.

(continued)



Shift the bits to the right 4 places, representing one BCD digit. This moves the least significant digit back one digit to the right, leaving room for “F” at the end.

This forces all digits to the left of the least significant digit to zero.

Now we combine the isolated least significant digit back with the other digit. However, it has been shifted to the right to make room for the decimal point.

Output the two LSDs to numeric output position 0. They are spaced apart one digit, to make room for the decimal point, and have been moved to the left to make room for the “F”.

Place an END coil, marking the end of the main program.

The Data Label box referenced by K1, preceeds the following ACON box.

This ACON box contains 32 characters. Be sure to carefully count the spaces to match the display.

The Data Label box referenced by K2, preceeds the following ACON box.

This ACON box contains 32 spaces. These fill the bottom two lines of the display.

NOTE: The resulting BCD number in V2000 and V2001 is useful only for display purposes. The actual numerical value is not valid for use in further computations (refer to the original value in V2200).

Chapter Summary

Now we have covered how to use Message Display Mode to communicate the status of the machine to its operator.

Summary of Key Points

We may summarize some of the key points we have learned about generating messages in this chapter:

- System Messages have priority over User Messages.
- User Messages can contain numerical values, or text.
- Numeric and Text Positions represent one 16-bit word of information.
- Numeric and Text output may be viewed simultaneously on the same display. Text output has mask and unmasking control over numeric output.
- Text is stored in V-memory in the form of ASCII codes.
- Ladder programs can load text data via the LD/OUT, LDD/OUTD, and MOVMC/ACON instructions.
- LDD (Load Double) instructions require you to swap positions of the ASCII codes of the first two characters with the second two characters, with respect to left-to-right display position orientation.
- ACON boxes is the preferred method for creating message data whenever you have more than a dozen characters to place together on the display.