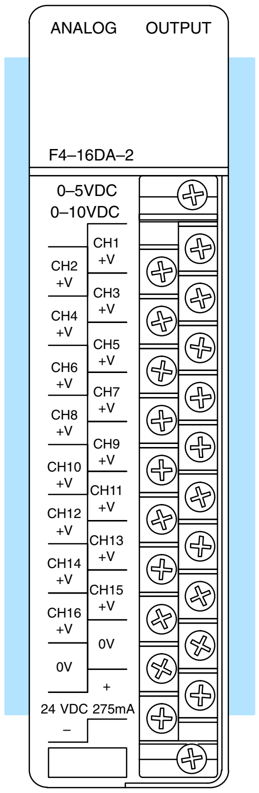
F4–16DA–2 16-Channel Analog Voltage Output

In This Chapter. . . .

Module Specifications

The F4–16DA–2 Analog Voltge Output module provides several features and benefits.

- It provides sixteen channels of 0–5VDC or 0–10VDC single ended voltage outputs, or a combination of 8 channels of 0–5VDC and 8 channels of 0–10VDC outputs.
- Analog outputs are optically isolated from PLC logic components.
- The module has a removable terminal block, so the module can be easily removed or changed without disconnecting the wiring.
- From two to sixteen analog outputs may be updated in one CPU scan (D4–440 and D4–450 CPUs only).



Analog Output Configuration Requirements

The F4–16DA–2 Analog Output module requires 32 discrete output points from the CPU. The module can be installed in any slot of a DL405 system, including remote bases. The limitations on the number of analog modules are:

- For local and expansion systems, the available power budget and discrete output points.
- For remote I/O systems, the available power budget and remote I/O points.

The following tables provide the specifications for the F4–16DA–2 Analog Output Module. Review these specifications to ensure the module meets your application requirements.

Output Specifications

Number of Channels	16, single ended (one common)
Output Range	0–5VDC, 0–10VDC, or 8 channels of 0–5VDC and 8 channels of 0–10VDC
Resolution	12 bit (1 in 4095)
Output Type	Voltage sourcing 10mA max.
External Load	1kΩ maximum / 10kΩ minimum (for example: 10 volts at 1kΩ = 10mA load; 10 volts at 10kΩ = 1mA load)
Crosstalk	–70 dB, ±1 count maximum
Linearity Error (end-to-end) and Relative Accuracy	±1 count maximum (10VDC at 25˚C)
Full Scale Calibration Error (offset error included)	±6 counts maximum (10VDC at 25˚C)
Offset Calibration Error	±3 counts maximum (0VDC at 25˚C)
Maximum Inaccuracy	±0.2% at 25˚C (77˚F) ±0.4% at 0˚ to 60˚C (32˚ to140˚F)
Conversion Time	400 μS maximum, for full scale change 4.5 to 9 mS for digital output to analog out

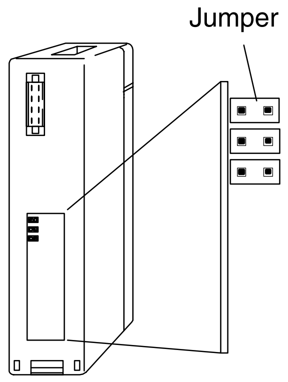
General Module Specifications

Digital Output Points Required	32 point (Y) outputs, 2 sets each of 12 bits binary data, 3 bits channel select, 1 bit output enable
Power Budget Requirement	80 mA @ 5 VDC (supplied by base)
External Power Supply	21.6 – 26.4 VDC, 275 mA. max. class 2
Accuracy vs. Temperature	±57 ppm / ˚C full scale calibration range (including maximum offset change, 2 counts)
Operating Temperature	0° to 60°C (32˚ to 140°F)
Storage Temperature	–20° to 70°C (–4˚ to 158°F)
Relative Humidity	5 to 95% (non-condensing)
Environmental Air	No corrosive gases permitted
Vibration	MIL STD 810C 514.2
Shock	MIL STD 810C 516.2
Noise Immunity	NEMA ICS3-304

Setting the Module Jumpers

Before installing and wiring the module, you may need to change the internal jumper settings. The module has three pairs of terminals, located in the open cutout at the rear of the housing. Two jumpers are supplied. These jumper settings determine whether the outputs are 16 channels of 0–5VDC (setting #1 below), 16 channels of 0–10VDC (setting #2) or eight channels of each (setting #3).

There are three possible settings. When a jumper is removed, store the jumper by placing it over one terminal as shown below to prevent losing it.



Setting #1	Setting #2	Setting #3
16 channels 0–5VDC (factory setting)	16 channels 0–10VDC	8 channels (ch.1–8) at 0–10VDC and 8 channels (ch. 9–16) at 0–5VDC
Installed	Removed	Removed
Installed	Installed	Removed
Removed	Removed	Installed

Connecting the Field Wiring

Wiring Guidelines

Your company may have guidelines for wiring and cable installation. If so, you should check those before you begin the installation. Here are some general things to consider.

- Use the shortest wiring route whenever possible.
- Use shielded wiring and ground the shield at the module or the power supply return (0V). *Do not* ground the shield at both the module and the transducer.
- Do not run the signal wiring next to large motors, high current switches, or transformers. This may cause noise problems.
- Route the wiring through an approved cable housing to minimize the risk of accidental damage. Check local and national codes to choose the correct method for your application.

User Power Supply Requirements

The F4–16DA–2 requires a field-side power supply. The module requires 21.6 to 26.4 VDC, Class 2, at 275mA max. current.

The D4–430/440/450 CPUs, D4–RS Remote I/O Controller, H4–EBC, and D4–EX Expansion Units have built-in 24 VDC power supplies that provide up to 400mA of current. You may use one of these instead of a separate supply if there is only one F4–16DA–2 module in use.

Load Requirements

Each channel in use must have a load impedance within a range of 1kΩ to 10kΩ. Unused channels must be left disconnected.

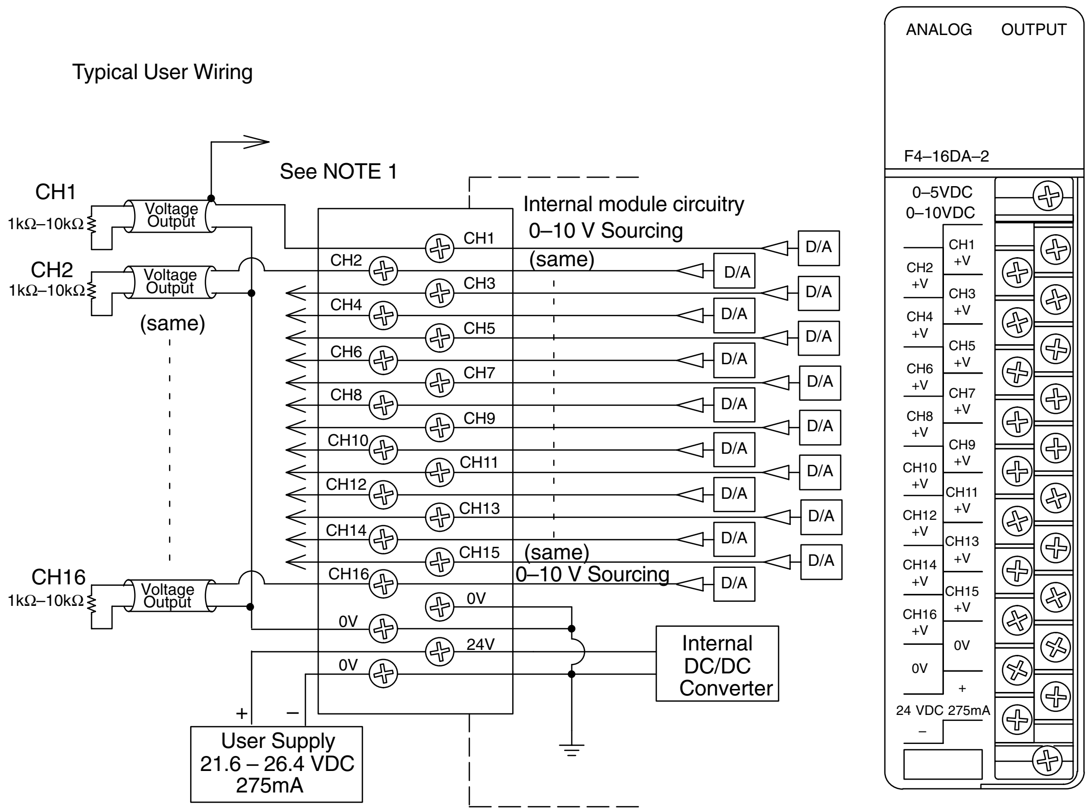
WARNING: If you are using the 24 VDC base power supply, make sure you calculate the power budget. Exceeding the power budget can cause unpredictable system operation that can lead to a risk of personal injury or damage to equipment.

Removable Connector

The F4–16DA–2 module has a removable connector to make wiring easier. Simply loosen the retaining screws and gently pull the connector from the module.

Wiring Diagram

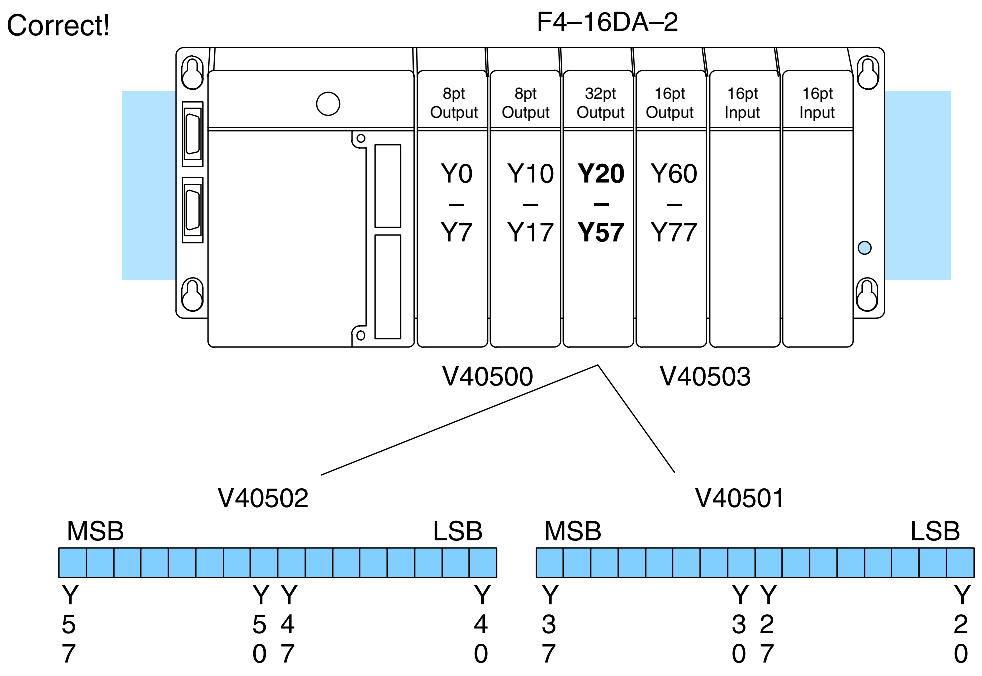
NOTE 1: Shields should be connected to the 0V terminal of the User Power Supply at the module terminal block.

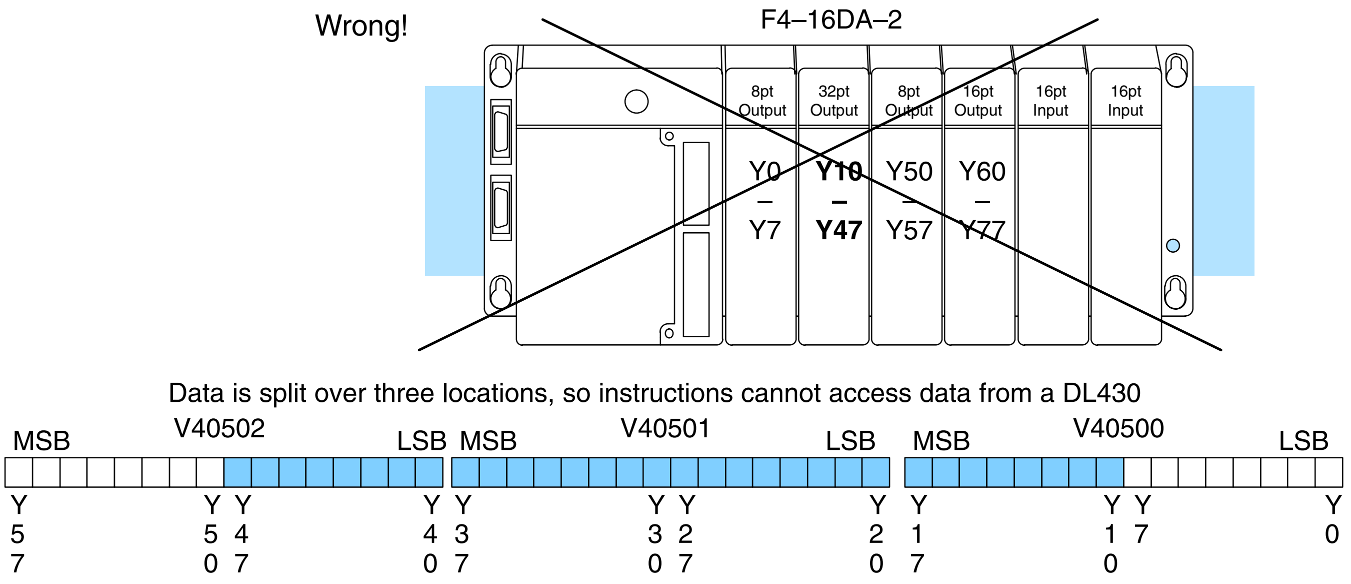


Module Operation

D4–430 Special Requirements

Even though the module can be placed in any slot, it is important to examine the configuration if you're using a D4–430 CPU. As you'll see in the section on writing the program, you use V-memory locations to send the analog data. As shown in the following diagram, if you place the module so the output points do not start on a V-memory boundary, the instructions can't access the data.

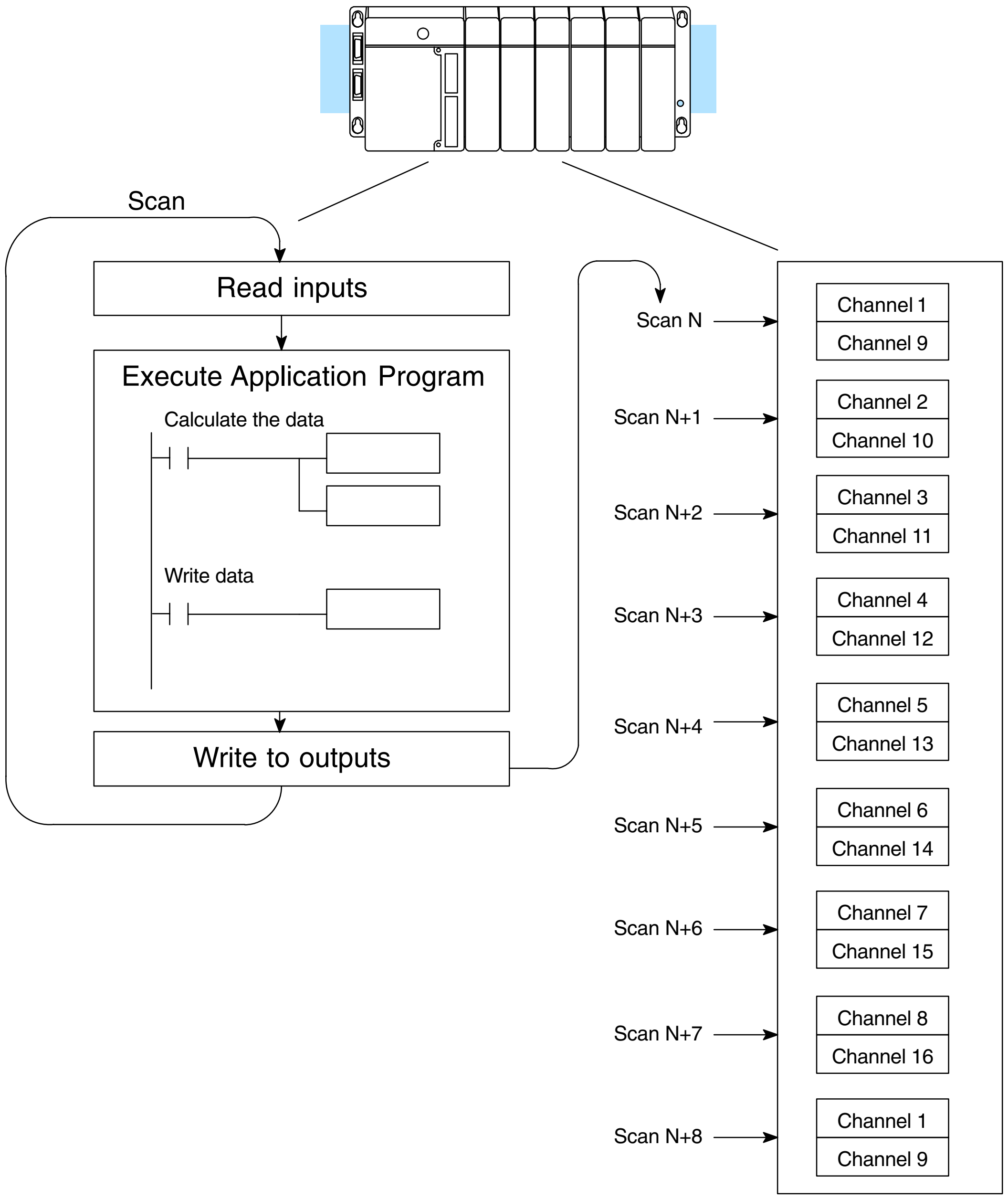




Before you begin writing the control program, it is important to take a few minutes to understand how the module processes and represents the analog signals.

Channel Scanning Sequence

The F4–16DA–2 module allows you to update the channels in any order. Your control program determines which channels get updated on any given scan. The channels are organized as two banks of outputs, eight channels each. A typical ladder program will update one channel in each bank, or two channels per CPU scan. So, all sixteen channels can be updated every eight scans. With a D4–440 or D4–450 CPU, you can use Immediate instructions to update all sixteen channels in the same scan (we'll show you how to do this later).

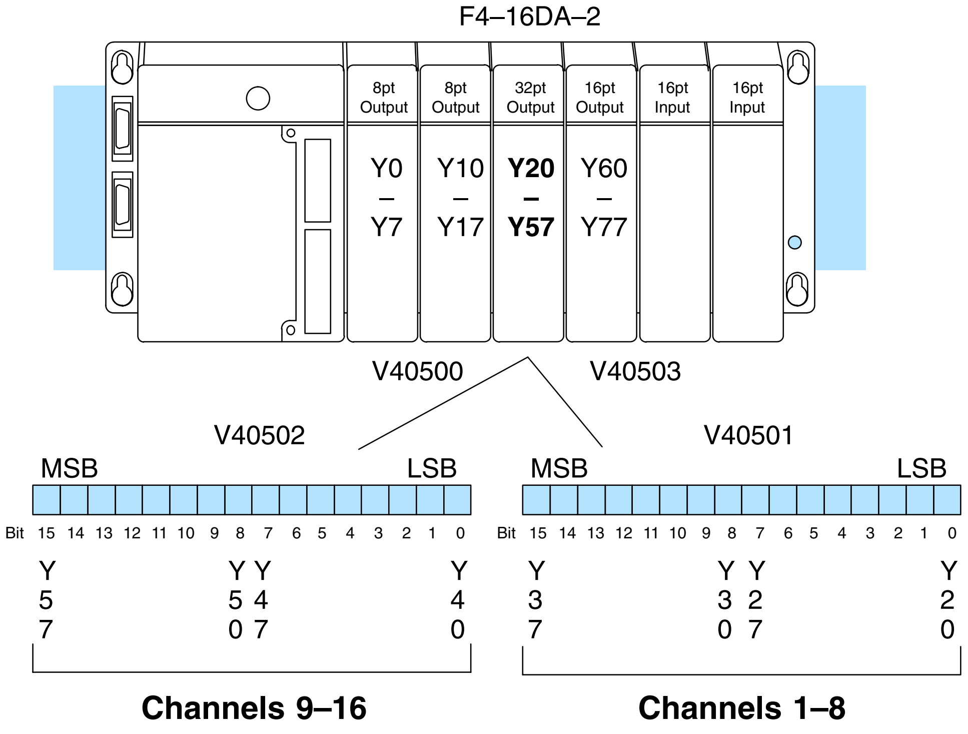


Output Bit Assignments

You may recall the F4–16DA–2 module requires 32 discrete output points from the CPU. These points provide:

- The digital representation of two analog signals per scan.
- Identification of the two channels that are to receive the data.
- Output enable control for all channels.

Since all output points are automatically mapped into V-memory, it is very easy to determine the location of the two data words that will be assigned to the module.

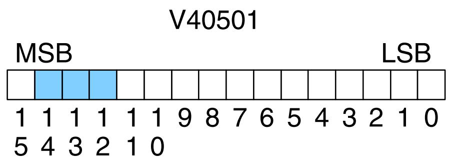


Within these V-memory locations the individual bits represent specific information about the channels selected and the analog signals.

Channel Select Bits

Output bits 12, 13, and 14 of both data words are the channel select outputs. They are binary encoded to select the channel that will be updated with the data. The bits of the lower data word select channels 1 to 8 as follows.

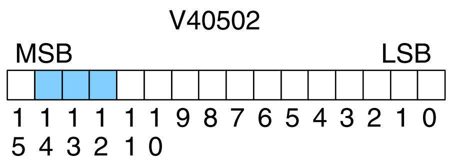
Bit 14	Bit 13	Bit 12	Channel
Off	Off	Off	1
Off	Off	On	2
Off	On	Off	3
Off	On	On	4
On	Off	Off	5
On	Off	On	6
On	On	Off	7
On	On	On	8



🟦 – Channel Select Bits, channels 1 through 8

The bits of the upper data word select channels 9 to 16 as follows.

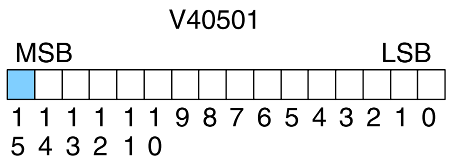
Bit 14	Bit 13	Bit 12	Channel
Off	Off	Off	9
Off	Off	On	10
Off	On	Off	11
Off	On	On	12
On	Off	Off	13
On	Off	On	14
On	On	Off	15
On	On	On	16



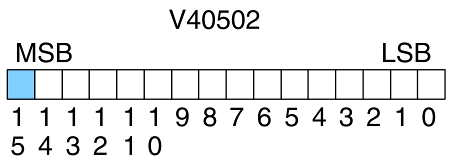
🟦 – Channel Select Bits, channels 9 through 16

Output Enable Bits

Output bit 15 is the Output Enable control bit in each data word for all eight channels of each bank. When either bit is off, the corresponding group of eight channel output voltages decrease to their lowest level, which is 0V for connected loads. Disabling the outputs also clears all eight output data registers for each bank. To resume analog output levels, first the appropriate Output Enable control bit must turn on. Then, the CPU must write new data to each channel to restore the output voltage for that channel.



🟦 – Output Enable Bit, channels 1 through 8



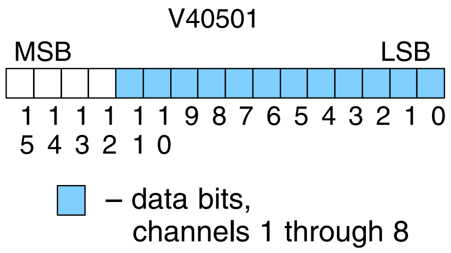
🟦 – Output Enable Bit, channels 9 through 16

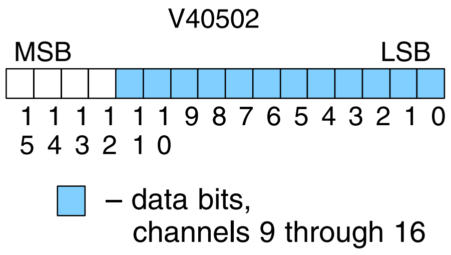
OFF = Disable (and clear)
ON = Enable

Analog Data Bits

The first twelve bits of both V-memory locations represent the analog data in binary format.

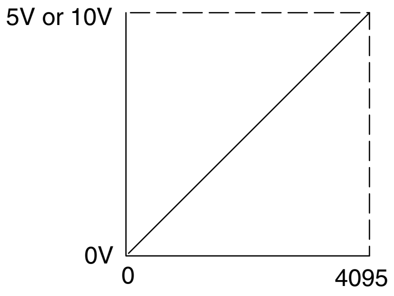
Bit	Value	Bit	Value
0	1	6	64
1	2	7	128
2	4	8	256
3	8	9	512
4	16	10	1024
5	32	11	2048





Module Resolution

Since the module has 12-bit resolution, the analog signal is made of 4096 counts ranging from 0 – 4095 (2^{12}). For the 0 to 5V scale, sending a 0 produces a 0V signal, and 4095 gives a 5V signal. This is equivalent to a binary value of 0000 0000 0000 to 1111 1111 1111, or 000 to FFF hexadecimal. The graph to the right shows the linear relationship between the data value and output signal level.



Each count can also be expressed in terms of the signal level by using the equation shown. The following table shows the smallest signal change that occurs when the digital value is increased by 1 LSB.

$$\text{Resolution} = \frac{H - L}{4095}$$

H = high limit of the signal range

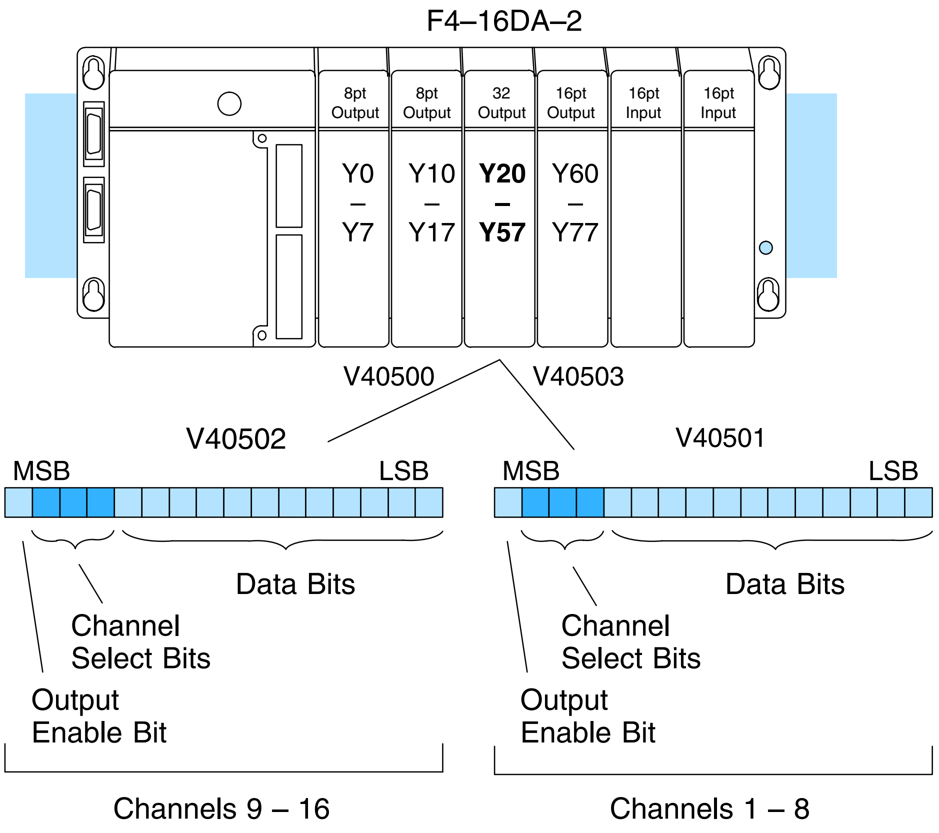
L = low limit of the signal range

Signal Range	Span (H – L)	Divide By	Smallest Change
0 to 5VDC	5VDC	4095	1.221mV
0 to 10VDC	10VDC	4095	2.442mV

Writing the Control Program

Update Any Channel

As mentioned earlier, you can update two channels per scan using regular I/O instructions, or any number of channels per scan using immediate I/O instructions. The following diagram shows the data locations for an example system. You use the channel select outputs to determine which channels get updated (more on this later).



Calculating the Digital Value

Your program has to calculate the digital value to send to the analog module. There are many ways to do this, but most applications are understood more easily if you use measurements in engineering units. This is accomplished by using the conversion formula shown.

You may have to make adjustments to the formula depending on the scale you choose for the engineering units.

$$A = U \frac{4095}{H - L}$$

A = Analog value (0 – 4095)

U = Engineering Units

H = high limit of the engineering unit range

L = low limit of the engineering unit range

Consider the following example which controls pressure from 0.0 to 99.9 PSI. By using the formula, you can easily determine the digital value that should be sent to the module. The example shows the conversion required to yield 49.4 PSI. Notice the formula uses a multiplier of 10. This is because the decimal portion of 49.4 cannot be loaded, so you adjust the formula to compensate for it.

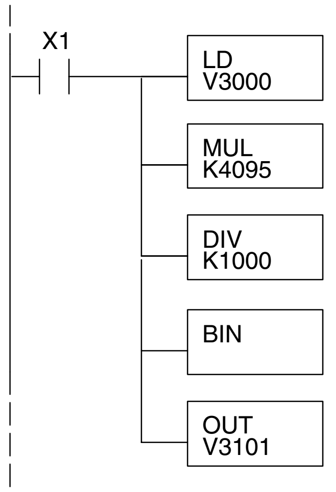
$$A = 10U \frac{4095}{10(H - L)}$$

$$A = 494 \frac{4095}{1000 - 0}$$

$$A = 2023$$

Here is how you would write the program to perform the engineering unit conversion. This example assumes you have calculated or loaded the engineering unit value and stored it in V3000. Also, you have to perform this for all sixteen channels if you're using different data for each channel.

NOTE: The DL405 offers various instructions that allow you to perform math operations using binary, BCD, etc. It's usually easier to perform any math calculations in BCD and then convert the value to binary before you send the data to the module. If you are using binary math, you do not have to include the BIN conversion.



When X1 is on, the engineering units (stored in V3000) are loaded into the accumulator. This example assumes the numbers are BCD.

Multiply the accumulator by 4095 (to start the conversion).

Divide the accumulator by 1000 (because we used a multiplier of 10, we have to use 1000 instead of 100).

Convert the BCD number to binary (you must omit this step if you have converted the data elsewhere).

Store the result in V3101. This is the digital value, in binary form, that should be sent to the module.

V-Memory Registers

The ladder program examples that follow occasionally use certain V-memory register addresses in the CPU that correspond to 16-bit Y output modules. Use the table below to find the V-memory address for the particular slot of your analog module. See Appendix A for additional addresses for D4–450 CPUs.

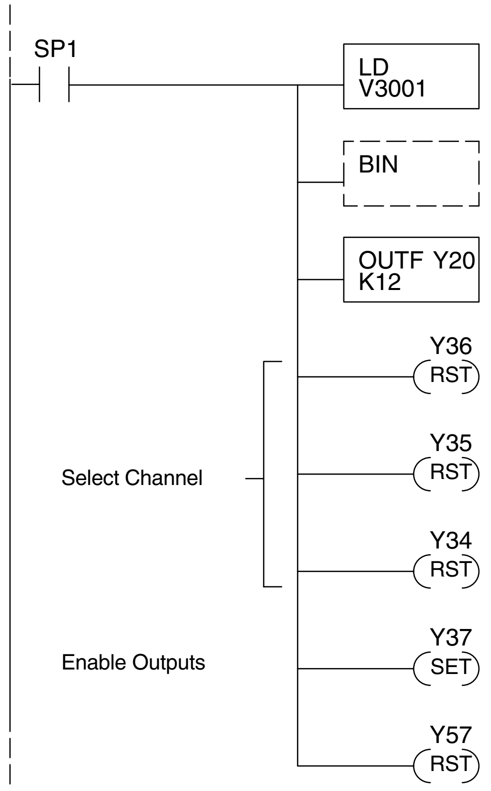
V-Memory Register Addresses for 16-Point Output (Y) Locations

Y	000	020	040	060	100	120	140	160	200	220
V	40500	40501	40502	40503	40504	40505	40506	40507	40510	40511
Y	240	260	300	320	340	360	400	420	440	460
V	40512	40513	40514	40515	40516	40517	40520	40521	40522	40523

Sending Data to One Channel, D4–440/450

✗	✓	✓
430	440	450

The following two programs show you how to update a single channel. Notice the D4–440/D4–450 and D4–430 CPUs use slightly different programs, although the D4–430 example will also work with D4–440 and D4–450 CPUs. Since the D4–430 does not support the OUTF instruction, the program must be modified to make sure the channel select bits are not accidentally changed by the data in the accumulator. This example assumes you already have the data loaded in V3001.



The LD instruction loads the data for channel 1 into the accumulator. Since SP1 is used, this rung automatically executes on every scan. You could also use an X, C, etc. permissive contact.

The BIN instruction converts the accumulator data to binary (you must omit this step if you've already converted the data elsewhere).

The OUTF sends the 12 bits to the data word. Our example starts with Y20, but the actual value depends on the location of the module in your application.

Turn Y36, Y35, and Y34 off to update Channel 1.

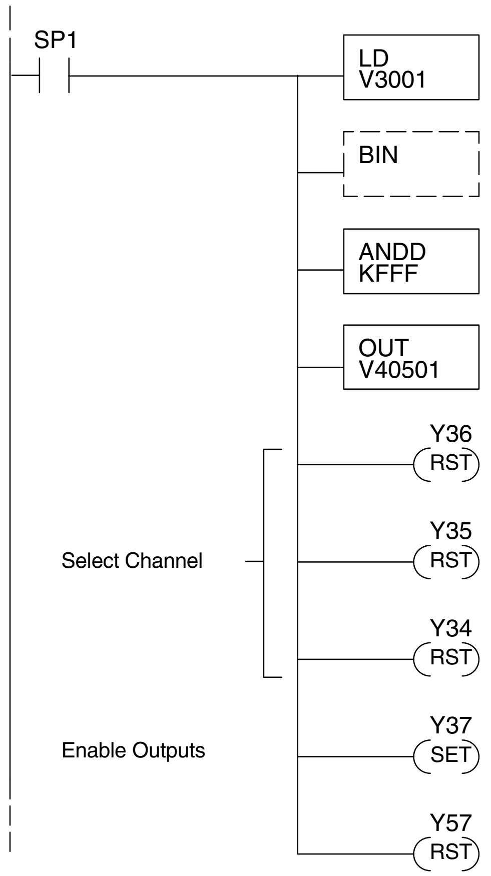
Y36	Y35	Y34	Channel
Off	Off	Off	Ch. 1
Off	Off	On	Ch. 2
Off	On	Off	Ch. 3
Off	On	On	Ch. 4
On	Off	Off	Ch. 5
On	Off	On	Ch. 6
On	On	Off	Ch. 7
On	On	On	Ch. 8

Turn on Y37 to enable all eight output channels of the first bank.

Turn off Y57 to disable the second bank of channels, 9 through 16.

Sending Data to One Channel, D4–430

✓ ✓ ✓
430 440 450



The LD instruction loads the data for channel 1 into the accumulator. Since SP1 is used, this rung automatically executes every scan. You could also use an X, C, etc. permissive contact.

The BIN instruction converts the accumulator data to binary (you must omit this step if you've already converted the data elsewhere).

The ANDD instruction masks off the channel select bits to prevent an accidental channel selection.

The OUT instruction sends the data to the module. Our example starts with V40501, but the actual value depends on the location of the module in your application.

Turn Y36, Y35, and Y34 off to update Channel 1.

Y36	Y35	Y34	Channel
Off	Off	Off	Ch. 1
Off	Off	On	Ch. 2
Off	On	Off	Ch. 3
Off	On	On	Ch. 4
On	Off	Off	Ch. 5
On	Off	On	Ch. 6
On	On	Off	Ch. 7
On	On	On	Ch. 8

Turn on Y37 to enable all eight output channels of the first bank.

Turn off Y57 to disable the second bank of channels, 9 through 16.

Sequencing the Channel Updates

The next four example programs show you how to send digital values to the module when you have more than one channel. These examples automatically update all sixteen channels over eight scans. Channel updates occur in pairs, such as channels 1 and 9, channels 2 and 10, and so on. This method uses the same binary code for the channel select bits of both groups of outputs, simplifying the programs. You may recall we stated earlier you could update all sixteen channels in the same scan. That's true, you can do that if you have a D4–440/450 CPU; however, this can increase the scan time and you may not always need to update all sixteen channels on every scan. We show you how to do this later in this chapter.

The first two sequencing examples are fairly simple and will work in most situations. We recommend these for new users. They use control relays C1 through C10 as index numbers corresponding to the pair of channels updated on any particular scan. At the end of each scan, only one control relay C1 through C10 is on. On each subsequent scan, the next control relay energizes. The channel sequencing automatically begins with channels 1 and 9 on the first scan, or after any disruption in the logic.

The last two examples, 3 and 4, are slightly more complex. However, they do not depend on the use of control relays to provide channel sequencing. Instead, they use function boxes to increment a channel pointer value in V-memory. Then, other instructions perform bit manipulations to position the channel select bits properly in the output words to the module.

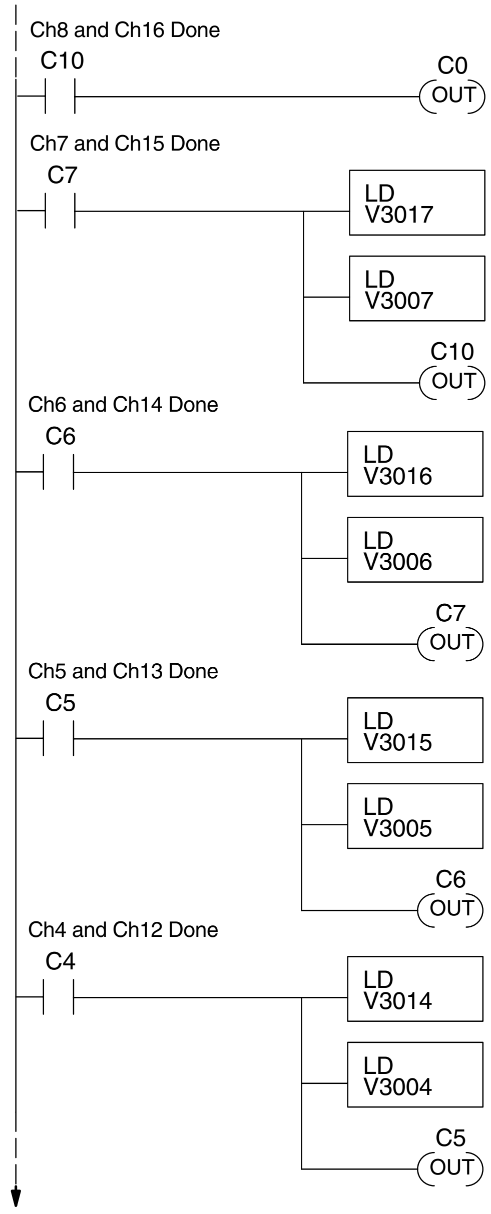
Sequencing Example 1, D4–440/450

✗ ✓ ✓
430 440 450

The following program example shows how to send digital values to the module when you have more than one channel. This example assumes you have already loaded the data according to the following table. It is important to use the rungs in the order shown for the program to work.

V-Memory Locations for Output Data in Examples 1 and 2

Channel Number	1	2	3	4	5	6	7	8
V-Memory Storage	3000	3001	3002	3003	3004	3005	3006	3007
Channel Number	9	10	11	12	13	14	15	16
V-Memory Storage	3010	3011	3012	3013	3014	3015	3016	3017



When channels 8 and 16 are updated, C0 restarts the update sequence.

When channels 7 and 15 have been updated, this rung loads the data for channel 16 into the accumulator. The second LD instruction places the data for channel 8 in the accumulator and pushes channel 16's data onto the stack.

Turning on C10 triggers the channel update (see the channel select rungs).

When channels 6 and 14 have been updated, this rung loads the data for channel 15 into the accumulator. The second LD instruction places the data for channel 7 in the accumulator and pushes channel 15's data onto the stack.

Turning on C7 triggers the channel update (see the channel select rungs).

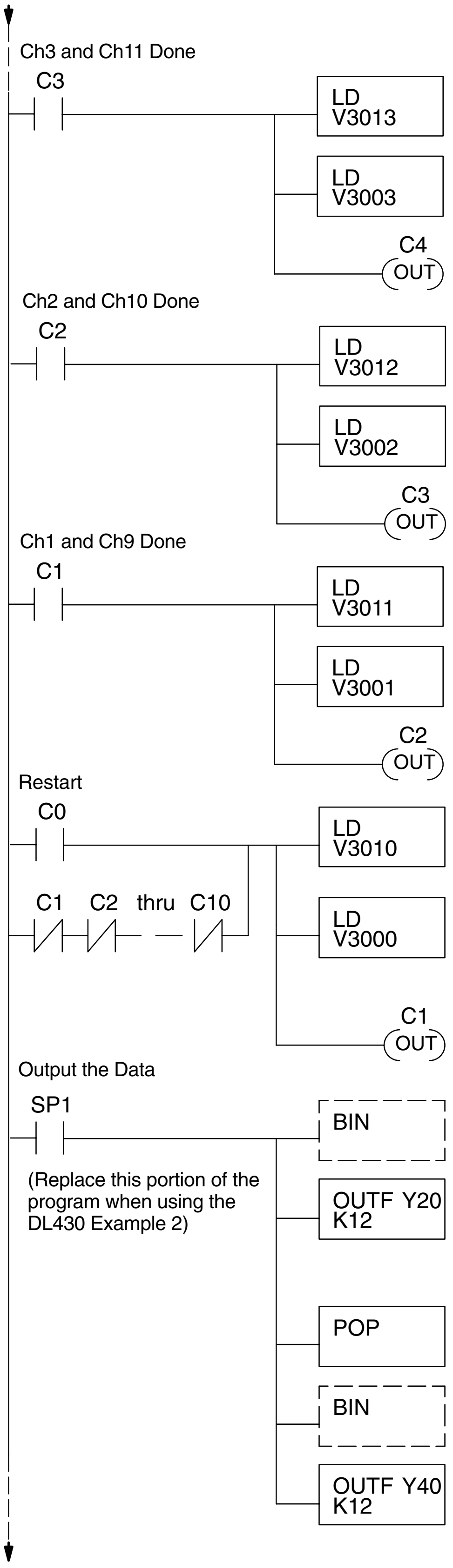
When channels 5 and 13 have been updated, this rung loads the data for channel 14 into the accumulator. The second LD instruction places the data for channel 6 in the accumulator and pushes channel 14's data onto the stack.

Turning on C6 triggers the channel update (see the channel select rungs).

When channels 4 and 12 have been updated, this rung loads the data for channel 13 into the accumulator. The second LD instruction places the data for channel 5 in the accumulator and pushes channel 13's data onto the stack.

Turning on C5 triggers the channel update (see the channel select rungs).

Example 1 Continued



When channels 3 and 11 have been updated, this rung loads the data for channel 12 into the accumulator. The second LD instruction places the data for channel 4 in the accumulator and pushes channel 12's data onto the stack.

Turning on C4 triggers the channel update (see the channel select rungs).

When channels 2 and 10 have been updated, this rung loads the data for channel 11 into the accumulator. The second LD instruction places the data for channel 3 in the accumulator and pushes channel 11's data onto the stack.

Turning on C3 triggers the channel update (see the channel select rungs).

When channels 1 and 9 have been updated, this rung loads the data for channel 10 into the accumulator. The second LD instruction places the data for channel 2 in the accumulator and pushes channel 10's data onto the stack.

Turning on C2 triggers the channel update (see the channel select rungs).

This rung loads the data for channel 9 into the accumulator. The second LD instruction places the data for channel 1 in the accumulator and pushes channel 9's data onto the stack. C0 restarts the sequence after channels 8 and 16 are done (see the top rung). The first scan or any interruption in control relay sequencing is detected when control relays C1 through C10 are off (all eight contacts are not shown here due to space constraints). In this case, we also start the sequence with channels 1 and 9.

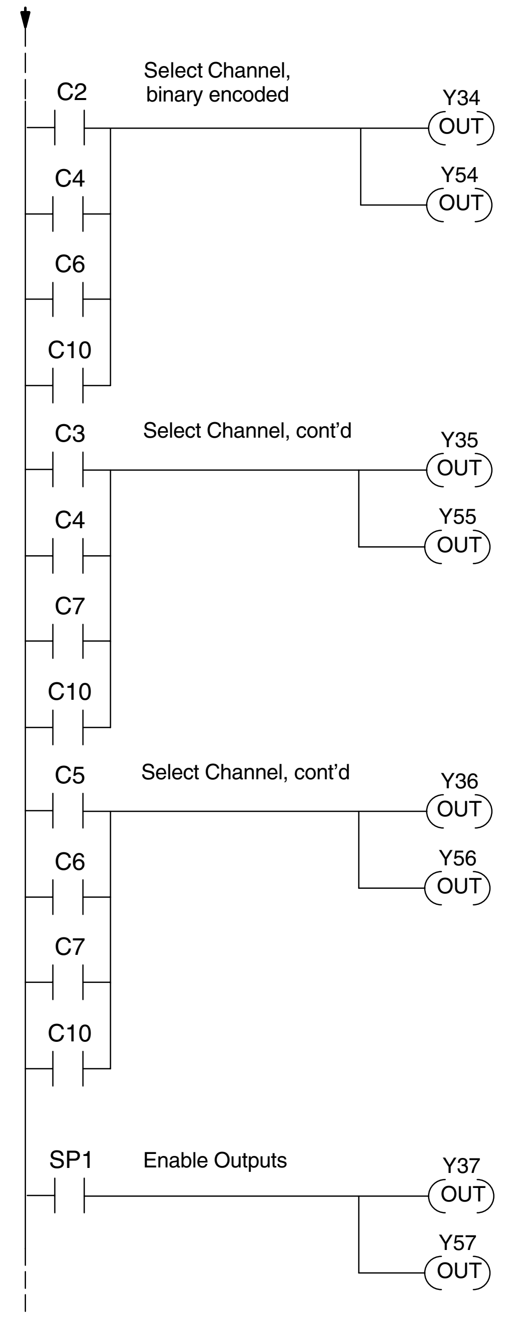
Turning on C1 triggers the channel update (see the channel select rungs).

This rung converts the accumulator data for channels 1 through 8 (one per scan) to binary (you must omit this step if you've already converted the data elsewhere). It also loads the data to the appropriate bits of the data word for the first bank of eight channels. Our example starts with Y20, but the actual value depends on the location of the module in your application.

We move the data for channels 9 through 16 (one per scan) from the stack to the accumulator.

This rung converts the accumulator data for channels 9 through 16 to binary (you must omit this step if you've already converted the data elsewhere). It also loads the data to the appropriate bits of the data word for the second bank of eight channels. In our example, it is Y40, but the actual value depends on the location of the module in your application.

Example 1 Continued



Set Y36, Y35, and Y34 to the binary code which selects the output channel 1 through 8, based on the control relay status. We use the same binary code to select channels 9 through 16.

CR(on)	Y36,Y56	Y35,Y55	Y34,Y54	Channels
C1	Off	Off	Off	Ch. 1 and 9
C2	Off	Off	On	Ch. 2 and 10
C3	Off	On	Off	Ch. 3 and 11
C4	Off	On	On	Ch. 4 and 12
C5	On	Off	Off	Ch. 5 and 13
C6	On	Off	On	Ch. 6 and 14
C7	On	On	Off	Ch. 7 and 15
C10	On	On	On	Ch. 8 and 16

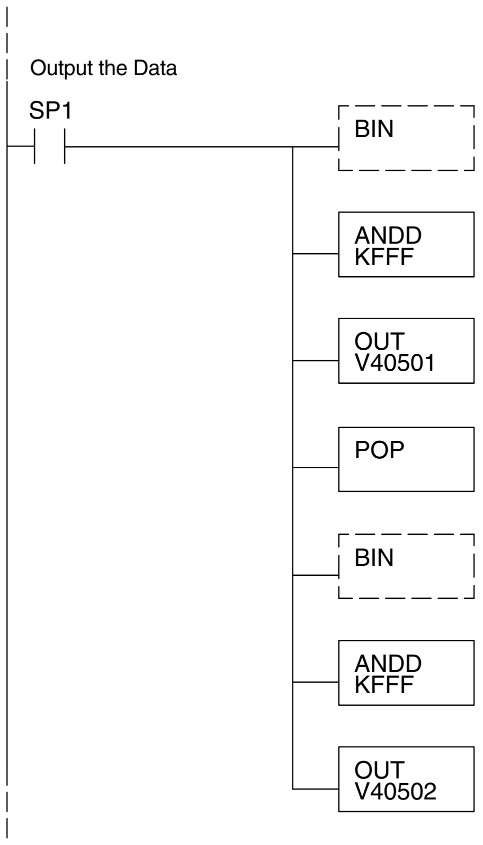
Enable channels 1 through 8. SP1 is always on.

Enable channels 9 through 16.

Sequencing Example 2, D4–430

✓ 430 ✓ 440 ✓ 450

Since the D4–430 does not support the OUTF instruction, the previous program must be modified to make sure the channel select bits or the output enable bits are not accidentally changed by the data in the accumulator. Replace the “Output the Data” rung in the middle of Example 1 with the new rung below. Be sure to retain the original order of the rungs shown in Example 1 for the program to work. This will also work with D4–440 and D4–450 CPUs.



This rung converts the accumulator data for channels 1 through 8 (one per scan) to binary (you must omit this step if you’ve already converted the data elsewhere).

The ANDD instruction masks off the channel select bits to prevent an accidental channel selection.

The OUT instruction sends the data to the module. Our example starts with V40501 for the first bank of 8 channels, but the actual value depends on the location of the module in your application.

We move the data for channels 9 through 16 (one per scan) from the stack to the accumulator.

This rung converts the accumulator data for channels 9 through 16 (one per scan) to binary (you must omit this step if you’ve already converted the data elsewhere).

The ANDD instruction masks off the channel select bits to prevent an accidental channel selection.

The OUT instruction sends the data to the module. Our example uses V40502 for the second bank of 8 channels, but the actual value depends on the location of the module in your application.

Sequencing Example 3, D4–440/450

✗	✓	✓
430	440	450

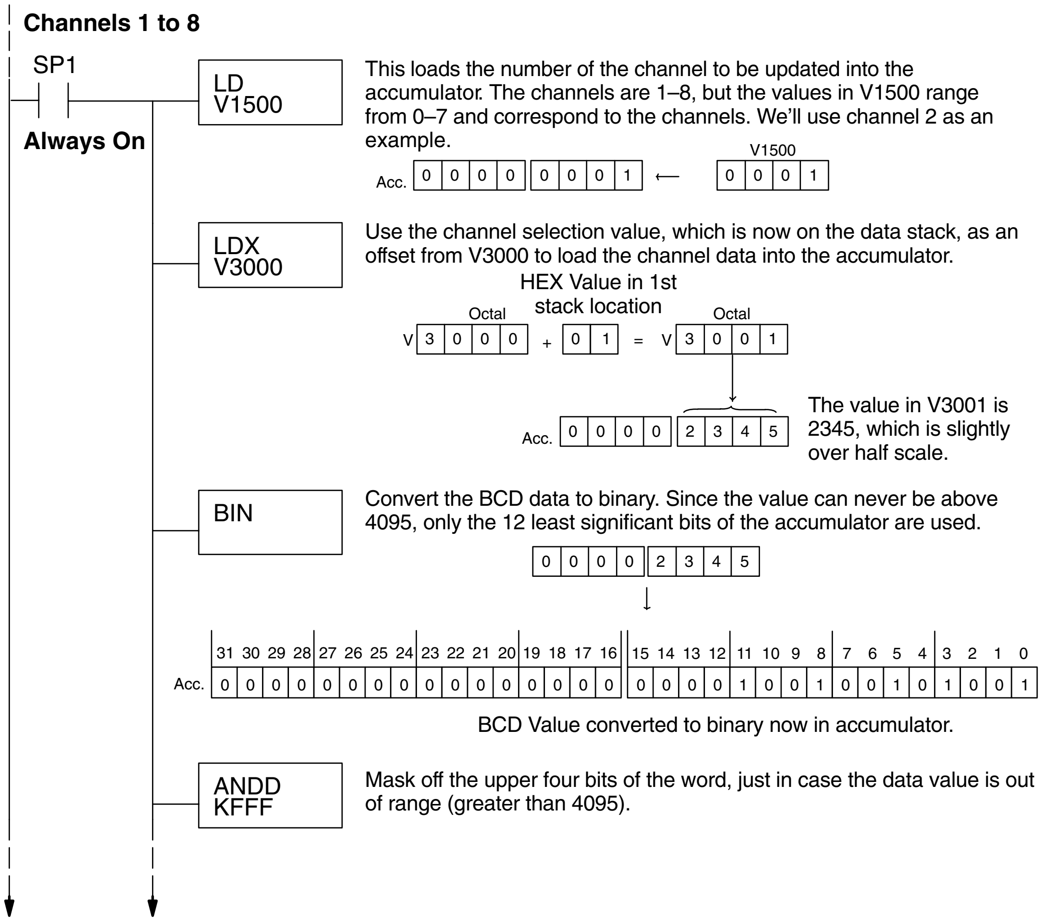
The following program example shows how to send digital values to the module when you have more than one channel. This example works only for D4–440 and D4–450 CPUs. It assumes you are using the following data locations.

V-Memory Locations for Output Data in Example 3

Channel Number	1	2	3	4	5	6	7	8
V-Memory Storage	3000	3001	3002	3003	3004	3005	3006	3007
Channel Number	9	10	11	12	13	14	15	16
V-Memory Storage	3010	3011	3012	3013	3014	3015	3016	3017

The channel index is stored in V1500. It varies from 0 to 7, pointing to pairs of channels as shown: 0 = ch. 1 and 9, 1 = ch. 2 and 10, ... and 7 = ch. 8 and16. This example assumes V1500 is initialized to "0" earlier in the program.

The first portion of the program updates one channel from the first bank of eight channels (1 to 8) during each scan. The program comments for this portion also shows the accumulator status at each step. The second part of the program updates one channel from the second bank of eight channels (9 to 16) during each scan (accumulator status not shown). So, there are two channels which are updated on each scan. The last portion of the program increments the channel index number and resets it after eight scans.



This loads the number of the channel to be updated into the accumulator. The channels are 1–8, but the values in V1500 range from 0–7 and correspond to the channels. We'll use channel 2 as an example.

Use the channel selection value, which is now on the data stack, as an offset from V3000 to load the channel data into the accumulator.

HEX Value in 1st stack location

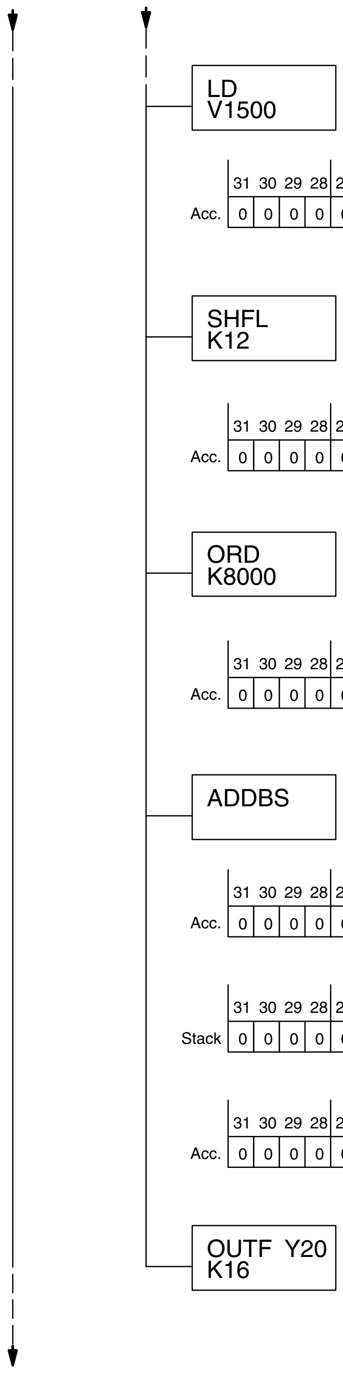
The value in V3001 is 2345, which is slightly over half scale.

Convert the BCD data to binary. Since the value can never be above 4095, only the 12 least significant bits of the accumulator are used.

BCD Value converted to binary now in accumulator.

Mask off the upper four bits of the word, just in case the data value is out of range (greater than 4095).

Example 3 Continued



Load the number of the channel to be updated back into the accumulator again (the channel data is moved to the first data stack location).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	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	1

This instruction moves the channel select bit(s) into the proper location. We'll use it later when we send the 16-bit data word to the module.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Set the Output Enable bit, by combining the value of 8000 hex with the accumulator value. This sets bit 15 to “1”, enabling channels 1 through 8.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Earlier in the program the data value was placed into the first data stack location. The ADDBS instruction adds the value currently in the accumulator with the value in the first data stack location.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

+

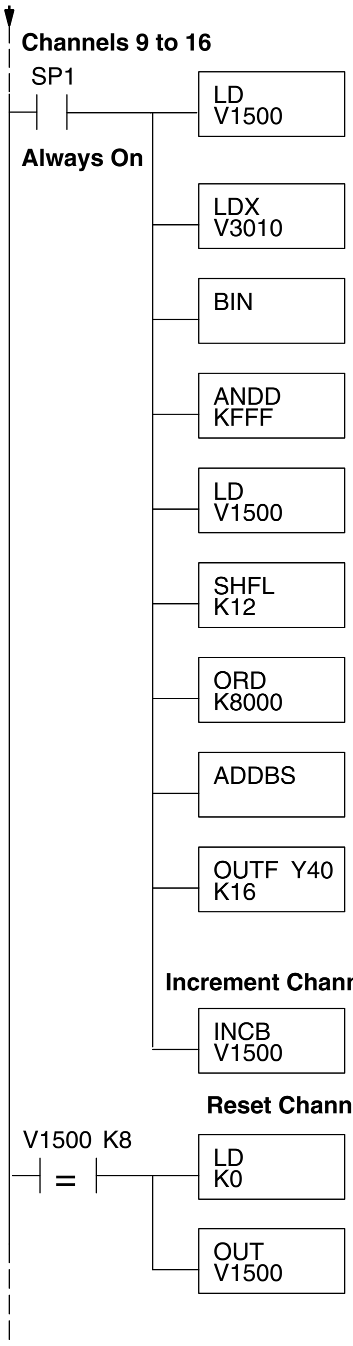
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Stack	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	1	0	0	1

Data for Analog Module

Send the lower 16 bits stored in the accumulator to the analog module.. The lowest 12 bits contain the analog data. Bits 12, 13, and 14 are the channel selection bits. Bit 15 is the Output Enable bit. This example uses Y20, but the actual value depends on the location of the modules in your application.

Example 3 Continued



This loads the number of the channel to be updated into the accumulator. The channels are 9–16, but the values in V1500 range from 0–7 and correspond to the channels. We'll use channel 10 as an example.

Use the channel selection value, which is now on the data stack, as an offset from V3010 to load the channel data into the accumulator.

Convert the BCD data to binary. Since the value can never be above 4095, only the 12 least significant bits of the accumulator are used.

Mask off the upper four bits of the word, just in case the data value is out of range (greater than 4095).

Load the number of the channel to be updated back into the accumulator again (the channel data is moved to the first data stack location).

This instruction moves the channel select bit(s) into the proper location. We'll use it later when we send the 16-bit data word to the module.

Set the Output Enable bit, by combining the value of 8000 hex with the accumulator value. This sets bit 15 to "1", enabling channels 9 through 16.

Earlier in the program the data value was placed into the first data stack location. The ADDBS instruction adds the value currently in the accumulator with the value in the first data stack location.

Send the lower 16 bits stored in the accumulator to the analog module. The lowest 12 bits contain the analog data. Bits 12, 13, and 14 are the channel selection bits. Bit 15 is the Output Enable bit. This example uses Y40, but the actual value depends on the location of the modules in your application.

Increment the channel index value. This allows the logic to cycle through all sixteen channels (8 pairs).

When channels 8 and 16 have been updated, then reset the channel selection memory location to 0 (remember, 0 represents channel 1 and channel 9).

Sequencing Example 4, D4–430

✓ 430 ✓ 440 ✓ 450

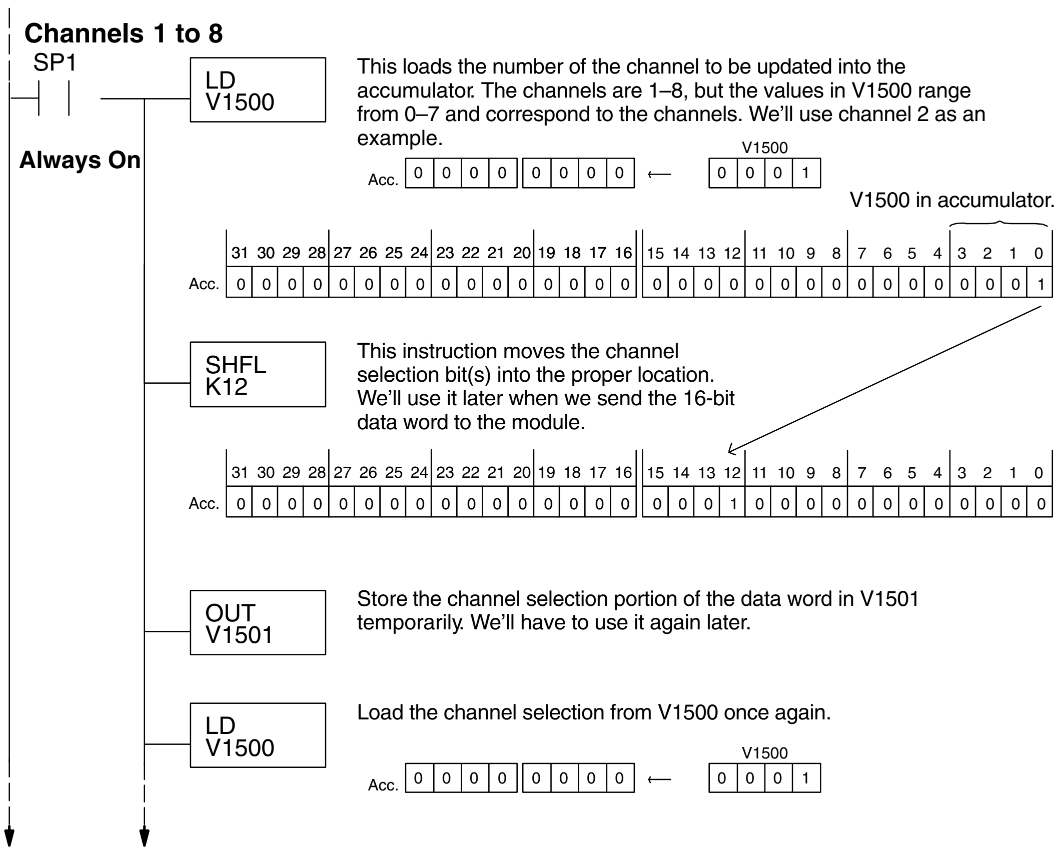
The following program example shows how to send digital values to the module when you have more than one channel. This example also works for D4–440 and D4–450 CPUs. It assumes you are using the following data locations.

V-Memory Locations for Output Data in Example 4

Channel Number	1	2	3	4	5	6	7	8
V–Memory Storage	3000	3001	3002	3003	3004	3005	3006	3007
Channel Number	9	10	11	12	13	14	15	16
V–Memory Storage	3010	3011	3012	3013	3014	3015	3016	3017

The channel index is stored in V1500. It varies from 0 to 7, pointing to pairs of channels as shown: 0 = ch. 1 and 9, 1 = ch. 2 and 10, ... and 7 = ch. 8 and16. This example assumes V1500 is initialized to “0” earlier in the program.

The first portion of the program updates one channel from the first bank of eight channels (1 to 8) during each scan. The program comments for this portion also show the accumulator status at each step. The second part of the program updates one channel from the second bank of eight channels (9 to 16) during each scan (accumulator status not shown, for brevity). So, there are two channels which are updated on each scan. The last portion of the program increments the channel index number and resets it after eight scans.



This loads the number of the channel to be updated into the accumulator. The channels are 1–8, but the values in V1500 range from 0–7 and correspond to the channels. We'll use channel 2 as an example.

This instruction moves the channel selection bit(s) into the proper location. We'll use it later when we send the 16-bit data word to the module.

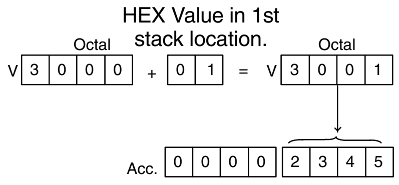
Store the channel selection portion of the data word in V1501 temporarily. We'll have to use it again later.

Load the channel selection from V1500 once again.

Example 4 Continued

LDX V3000

Use the channel selection value, which is now on the data stack, as an offset from V3000 to load the channel data into the accumulator.



The value in V3001 is 2345, which is slightly over half scale.

BIN

Convert the BCD data to binary. Since the value can never be above 4095, only the least significant 12 bits of the accumulator are used.

0	0	0	0	2	3	4	5

↓

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1

BCD Value converted to binary now in accumulator.

ANDD KFFF

Mask off the upper four bits of the word, just in case the data value is out of range (greater than 4095).

OR V1501

Earlier in the program the channel selection portion of the data word was created and stored in V1501. Now we can OR this location with the data word currently in the accumulator to get the final data word that is ready to send to the analog module.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1

OR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V1501	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

↓

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	1	0	0	1

Data for Analog Module

ORD K8000

Set the Output Enable bit, by combining the value of 8000 hex with the accumulator value. This sets bit 15 to “1”, enabling channels 1 through 8.

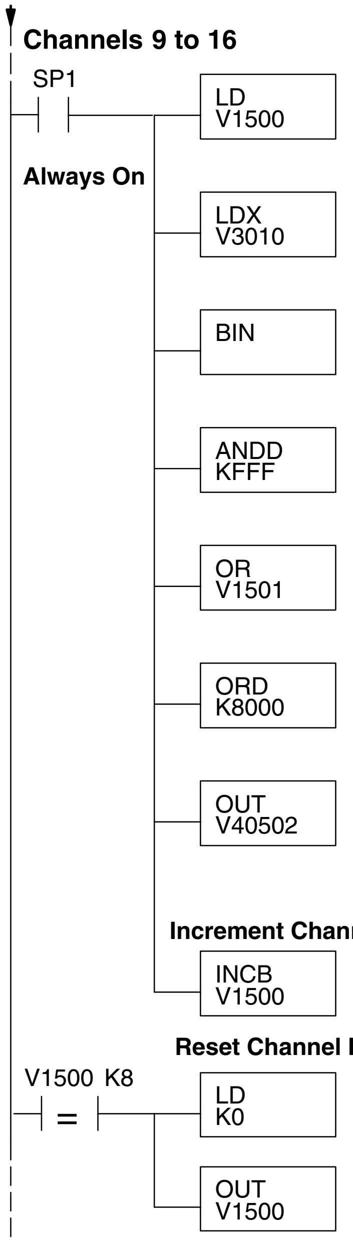
↓

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	1	0	0	1

OUT V40501

Send the data stored in the lower half of the accumulator to the analog module (the OUT instruction ignores the upper 16 bits of the accumulator). The most significant four bits of the analog word contain the channel selection bits. The remaining 12 bits contain the analog data. This example uses V40501, but the actual value depends on the location of the module in your application.

Example 4 Continued



Load the channel selection from V1500 once again.

Use the channel selection value, which is now on the data stack, as an offset from V3010 to load the channel data into the accumulator.

Convert the BCD data to binary. Since the value can never be above 4095, only the least significant 12 bits of the accumulator are used.

Mask off the upper four bits of the word, just in case the data value is out of range (greater than 4095).

Earlier in the program the channel selection portion of the data word was created and stored in V1501. Now we can OR this location with the data word currently in the accumulator to get the final data word that is ready to send to the analog module.

Set the Output Enable bit, by combining the value of 8000 hex with the accumulator value. This sets bit 15 to “1”, enabling channels 9 through 16.

Send the data stored in the lower half of the accumulator to the analog module (the OUT instruction ignores the upper 16 bits of the accumulator). The most significant four bits of the analog word contain the channel selection bits and the output enable bit. The remaining 12 bits contain the analog data. This example uses V40502, but the actual value depends on the location of the module in your application.

Increment the channel index value. This allows the logic to cycle through all sixteen channels (8 pairs).

When channels 8 and 16 have been updated, then reset the channel selection memory location to 0 (remember, 0 represents channel 1 and channel 9).

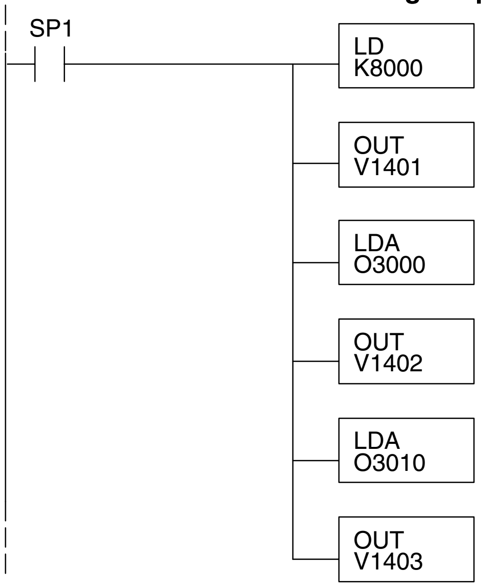
Updating All Channels in a Single Scan, D4–440/450

✗ ✓ ✓
430 440 450

By using the Immediate instructions found in the D4–440 and D4–450 CPUs, you can easily update all sixteen channels in a single scan. This method slows the CPU scan time. To minimize this impact, change the SP1 (Always On) contact to an X, C, etc. permissive contact that only updates the channels as required. This example assumes you already have the data loaded in V3000 to V3007 for channels 1 to 8 respectively, and data loaded in V3010 to V3017 for channels 9 to 16 respectively.

NOTE: This program will not work in a remote/slave arrangement. Use one of the programs shown that reads one channel per scan.

Initialize the Immediate Analog Output Pointers



The LD instruction loads the data into the accumulator. The constant K8000 represents channel 1 and 9 selected, Output Enable bit is on.

We store the channel index pointer in V1401. The program increments this in each pass, resetting it after eight scans.

The Load Address instruction takes the octal 3000 and converts it to hex, then puts it in the accumulator. V3000 is the location wich contains the data for channel 1.

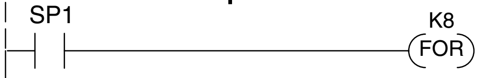
V1402 contains the pointer for channel 1 to 8 data.

The Load Address instruction takes the octal 3010 and converts it to hex, then puts it in the accumulator. V3010 is the location which contains the data for channel 9.

V1403 contains the pointer for channel 9 to 16 data.

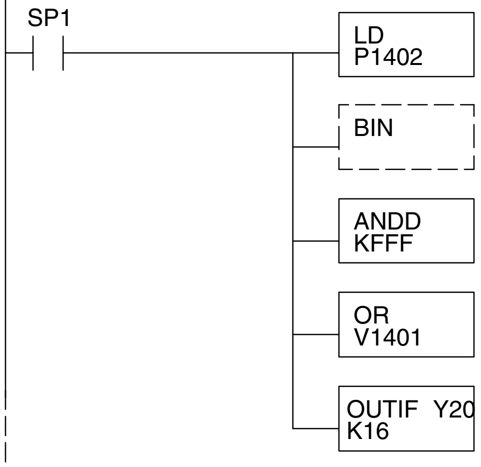
The following FOR–NEXT loop updates all 16 channels in a single scan. The first portion of the loop updates channels 1 through 8.

FOR-NEXT Loop



The following FOR–NEXT loop updates channels 1 through 16. The valid range for the constant K is 2 to 8 for the module. For example, a value of 2 will update channels 1 and 9, then 2 and 10.

Update Channels 1 - 8



Load the analog output value into the accumulator. V3000 = Ch. 1, V3001 = Ch. 2, ... V3007 = Ch. 8. The number at V1402 is a pointer to the address of the value.

Convert the data to binary. This step is optional, and your program may do this conversion elsewhere.

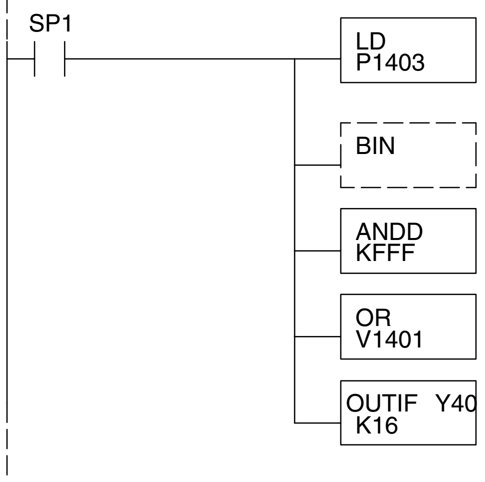
Mask off the channel select and output enable bits, so they are not corrupted by data out of range.

Combine the channel select and output enable bits. for Y36, Y35, and Y34 with the data in the accumulator.

Immediately write the lower 16 bits of the accumulator to the module's lower data word. This updates channels 1 through 8 during the FOR–NEXT loop. This example uses Y20, but the actual value depends on the location of the modules in your application.

The second portion of the FOR–NEXT loop updates channels 9 through 16.

Update Channels 9 - 16



Load the analog output value into the accumulator. V3010 = Ch. 9, V3011 = Ch. 10, ... V3017 = Ch. 16. The number at V1403 is a pointer to the address of the value.

Convert the data to binary. This step is optional, and your program may do this conversion elsewhere.

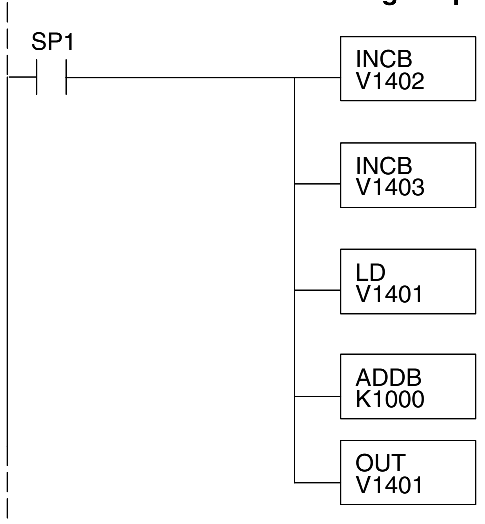
Mask off the channel select and output enable bits, so they are not corrupted by data out of range.

Combine the channel select and output enable bits. for Y56, Y55, and Y54 with the data in the accumulator.

Immediately write the lower 16 bits of the accumulator to the module's upper data word. This updates channels 9 through 16 during the FOR–NEXT loop. This example uses Y40, but the actual value depends on the location of the modules in your application.

Now we increment the Immediate Analog Output Pointers for the next pair of channels, before the next pass through the FOR-NEXT loop.

Increment Immediate Analog Output Pointers



Increment the analog output data pointer for channels 1 through 8.

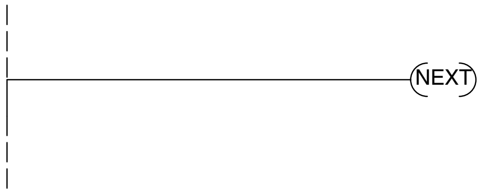
Increment the analog output data pointer for channels 9 through 16.

Load the current channel index count into the accumulator.

Add 1000 hex to the channel index value, since the channel select field is in the most significant four bits.

Save the incremented channel index value for the next pass through the FOR–NEXT loop.

Here is the end of the FOR-NEXT loop.



Execute the loop above the number of times specified in the FOR instruction.

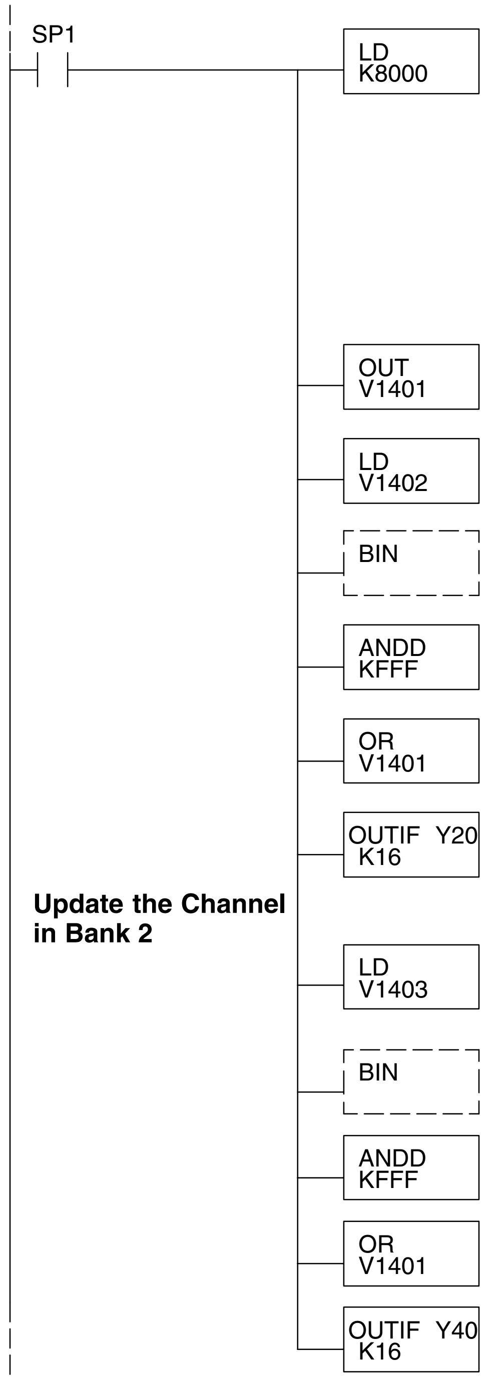
The program above can be modified to immediately output any number of channel pairs by changing the constant in the FOR instruction.

Updating a Pair of Channels in a Single Scan, D4–440/450

✗	✓	✓
430	440	450

You can also update just a pair of channels in a single scan by using the Immediate instructions found in the D4–440 and D4–450 CPUs. By removing the FOR–NEXT rungs and a couple of other rungs, we create the example below. This example assumes you will use one channel from the first eight, and a corresponding channel from the second bank of eight. Also, the data must already be loaded in V1401 and V1402 respectively.

Update the Channel in Bank 1



Load a constant representing the channel pair to be updated into the accumulator.

Index	Channels
8000	Ch. 1 and 9
9000	Ch. 2 and 10
A000	Ch. 3 and 11
B000	Ch. 4 and 12
C000	Ch. 5 and 13
D000	Ch. 6 and 14
E000	Ch. 7 and 15
F000	Ch. 8 and 16

We store the channel index in V1401.

Load the analog output value for the channel in bank 1 from V1402 into the accumulator.

Convert the data to binary. This step is optional, and your program may do this conversion elsewhere.

Mask off the channel select and output enable bits, so they are not corrupted by data out of range.

Combine the channel select and output enable bits. for Y36, Y35, and Y34 with the data in the accumulator.

Immediately write the lower 16 bits of the accumulator to the module's lower data word. This updates a channel in the first bank of eight. This example uses Y20, but the actual value depends on the location of the modules in your application.

Update the Channel in Bank 2

Load the analog output value for the channel in bank 2 from V1403 into the accumulator.

Convert the data to binary. This step is optional, and your program may do this conversion elsewhere.

Mask off the channel select and output enable bits, so they are not corrupted by data out of range.

Combine the channel select and output enable bits. for Y56, Y55, and Y54 with the data in the accumulator.

Immediately write the lower 16 bits of the accumulator to the module's upper data word. This updates a channel in the second bank of eight. This example uses Y40, but the actual value depends on the location of the modules in your application.

Analog and Digital Value Conversions

Sometimes it is helpful to be able to quickly convert between the voltage or current signal levels and the digital values. This is especially useful during machine startup or troubleshooting. The following table provides formulas to make this conversion easier.

Range	If you know the digital value ...	If you know the analog signal level ...
0 to 5VDC	$A = \frac{5D}{4095}$	$D = \frac{4095}{5}A$
0 to 10VDC	$A = \frac{10D}{4095}$	$D = \frac{4095}{10}A$

For example, if you need a 2.5V signal level with the module set for 0–5V, you would use the following formula to determine the digital value that should be stored in the V-memory location that contains the data.

$$D = \frac{4095}{5}A$$

$$D = \frac{4095}{5}(2.5V)$$

$$D = (819)\,(2.5)$$

$$D = 2048$$