Understanding the Operation

In This Chapter. . . .

The Operating Basics

Although most readers of this manual will have had prior experience with the PLC and its various modules, there are some unique features belonging to the HSC, of which you must become familiar. These include:

- How the HSC reserves X (inputs) and Y (outputs).
- The X and Y assignment table of the HSC
- The concept of shared memory and its relation to V-memory
- Reading and writing data to shared memory
- How numbers are stored in shared memory

The pages that follow in this chapter will tell you in detail how all of the above operations are handled.

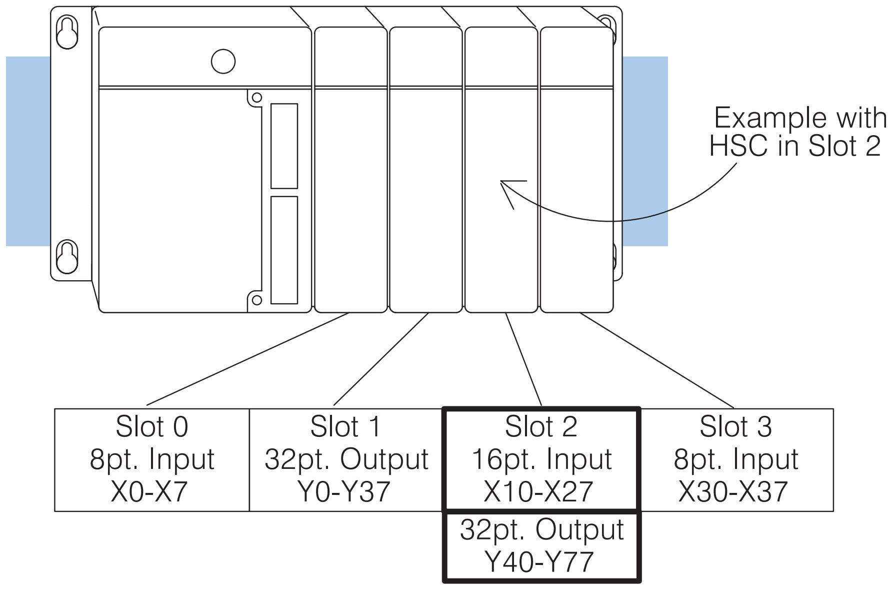
Assigning Your Data Types

The DL405 family uses the octal (base 8) numbering system to designate I/O points. The letter **X** is always used to indicate inputs and the letter **Y** indicates outputs. Each I/O point also has a number associated with it, i.e. X12, Y32, etc. Assigning letters and numbers to I/O is referred to as "configuring your I/O". I/O can be configured in one of two ways: **automatically** or **manually**. We will assume throughout this manual that you have used the automatic option. If you plan to manually configure your I/O, then you should read your DL405 User Manual in order to know how to perform the configuration.

Automatic Configuration

The DL405 CPUs automatically examine any installed I/O modules (including specialty modules) and establish the correct I/O configuration and addressing on power-up. For most applications, you never have to change or adjust the configuration.

The I/O addresses are assigned using octal numbering, starting at X0 and Y0. The addresses are assigned in groups of 8, 16, 32, or 64 depending on the number of points for the I/O module. The following diagram shows the I/O numbering scheme for an example system. Notice that the automatic addressing feature assigns numbers to the I/O of the HSC module, just as it does for other modules.


Example with HSC in Slot 2
Slot 0 8pt. Input X0-X7
Slot 1 32pt. Output Y0-Y37
Slot 2 16pt. Input X10-X27
32pt. Output Y40-Y77
Slot 3 8pt. Input X30-X37

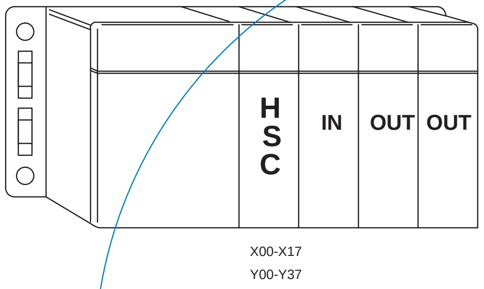
The HSC module will *automatically* consume 16 inputs and 32 outputs. The table on Page 5 of this chapter shows you how the HSC X's and Y's are assigned to specific functions.

Using the X and Y Assignment Table

If you are using automatic addressing, the DL405 CPU will look at how many I/O points are in the modules located to the left of the HSC, and automatically start numbering the X's and Y's of the HSC at the next available octal numbers. You need to know these number assignments in advance of writing your RLL, because they play an important role.

On the previous page, we placed the HSC in Slot 2. You could place it in any slot. If you place the HSC module in the first slot of your base, the process for immediately knowing the numbers assigned to your HSC X's and Y's is simple because you can merely substitute 00 for the values of *m* and *n* in the I/O assignment table. In such a case, you would not have to figure out what X's and Y's were committed to modules located to the left of the HSC, because there are none!

In this example we have placed the HSC in slot 0 of the CPU base. This simplifies X and Y identification because X's and Y's both start at 00.



For example:
Xn+6=X6=Status of CW output
Ym+3=Y3=ON for HSC RUN

X No.	Function
Xn+0	ON if current count is greater than preset
Xn+1	ON if current count is equal to preset
Xn+2	ON if current count is less than preset
Xn+3	Latched ON if overflow occurs (reset with Ym+1)
Xn+4	Status of CCW output
Xn+5	Status of OUT2 (brake) output
Xn+6	Status of CW output
Xn+7	Status of OUT1 (deceleration) output
Xn+10	Status of Limit Switch 2
Xn+11	Status of Limit Switch 1
Xn+12	ON if doing a search for home position
Xn+13	ON if a sampling is being conducted
Xn+14	NOT USED
Xn+15	ON for missing terminal block
Xn+16	ON if external power supply for outputs is missing or OFF

Y No.	Function
Ym+0	ON to reset OUT1 and OUT2 when in HSC run
Ym+1	ON to reset overflow flag (Xn+3)
Ym+2	Rising edge of this signal copies offset value into current count
Ym+3	ON for HSC run
Ym+4	Used to control CCW when not in HSC RUN or Home Search
Ym+5	Used to control OUT2 when not in HSC RUN or Home Search
Ym+6	Used to control CW when not in HSC RUN or Home Search
Ym+7	Used to control OUT1 when not in HSC RUN or Home Search

I/O Assignment Table

The following table provides all of the X and Y assignments for the HSC module. The letter “n” is the starting octal number for the inputs; and the letter “m” is the starting number for the outputs.

The Y outputs that are shaded have an external counterpart. That is, there are connecting terminals and labels on the module, so that field devices can enable these functions. Consequently, you can enable these functions either by setting the respective bits through ladder logic alone, or you can enable them by sending signals from external devices. You can use both methods within a single program. The Y outputs that are shaded in the table have external counterparts (LD=Ym+2, RUN=Ym+3, C.INH=Ym+10, LATCH=Ym+11, RST=Ym+12).

X No.	Function	Y No.	Function
Xn+0	ON if current count is greater than preset	Ym+0	ON to reset OUT1 and OUT2 when in HSC run
Xn+1	ON if current count is equal to preset	Ym+1	ON to reset overflow flag (Xn+3)
Xn+2	ON if current count is less than preset	Ym+2	Rising edge copies offset value into current count
Xn+3	Latched ON if overflow occurs (reset with Ym+1)	Ym+3	ON for HSC run
Xn+4	Status of CCW output	Ym+4	Used to control CCW when not in HSC RUN or Home Search
Xn+5	Status of OUT2 (brake) output	Ym+5	Used to control OUT2 when not in HSC RUN or Home Search
Xn+6	Status of CW output	Ym+6	Used to control CW when not in HSC RUN or Home Search
Xn+7	Status of OUT1 (deceleration) output	Ym+7	Used to control OUT1 when not in HSC RUN or Home Search
Xn+10	Status of Limit Switch 2	Ym+10	ON to temporarily suspend counting (inhibit counting)
Xn+11	Status of Limit Switch 1	Ym+11	Rising edge of this signal will latch the current count into shared memory
Xn+12	ON if doing a search for home position	Ym+12	If ON, it resets current count to zero

Xn+13	ON if a sampling is being conducted	Ym+13	OFF=Quadrature mode
			ON=UP/DOWN mode

Xn+14	NOT USED	Ym+14	Change state to change count direction
Xn+15	ON for loose or missing terminal block	Ym+15	Rising edge of this signal will invoke Home Search. You cannot use this if in HSC RUN.
Xn+16	ON if external power supply for outputs is missing or OFF	Ym+16	ON for x2 count operation (quadrature mode only/Ym+17 must be OFF)
Xn+17	ON if HSC detects an error	Ym+17	ON for x4 count operation (quadrature mode only)
		Ym+20	If OFF it will automatically reset current count to zero when current count=preset. If ON will not reset automatically unless count is at max or minimum.
		Ym+21	Rising edge of this signal will start the sampling feature
		Ym+22	Must be ON to enable external LD function
		Ym+23	ON to reset CW and CCW
		Ym +24	Not Used
		*Ym+25	ON to reset Home Search error.
		*Ym+26	ON to enable reset with INZ.
		**Ym+27	If turned ON before invoking Home Search, OUT2 (brake) will turn ON when Home is found.

*This manual was written for the latest version of the D4-HSC. If you have an HSC that was purchased from another vendor, it may not support these features.

**Available on HSC modules with production codes 9502 (Feb.’95) or later.

Reading and Writing Shared Memory

Contents and Data Flow

We provided a simple description of shared memory in Chapter 1, but it is important to understand it in detail. It resides in the HSC and it is shared by the DL405 CPU and the HSC itself. The table below shows how the memory is allocated. The data flow diagram is *from the perspective of the DL405 CPU*. Notice that it can only write four of the parameters from its V-memory to the shared memory, but it can read all seven parameters from shared memory into V-memory.

Not shown in the data flow chart is the reading and writing capability *from the HSC point of view*. The HSC reads and writes to the shared memory continuously, and it can do this for any of the seven parameters.

On the following page we will show you how to use your ladder logic program to read and write data to shared memory.

Shared Memory Map

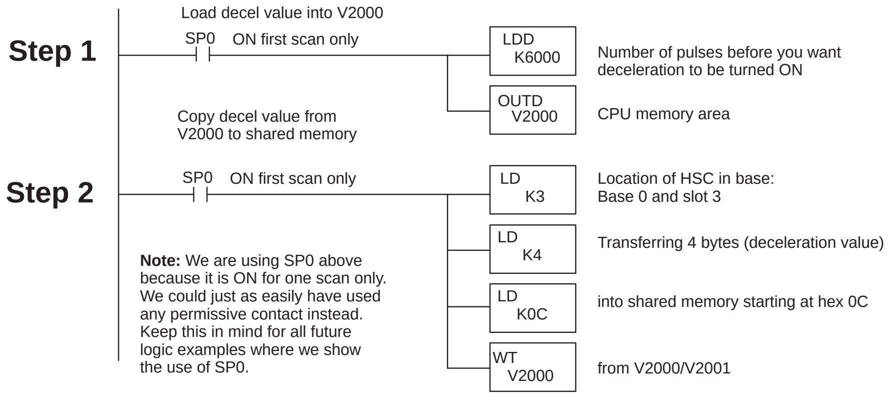
Data Flow Direction
(From DL405 CPU Perspective)

Data	Address (hex)		Address (octal)	V-Memory		Shared Memory
Current Count (4 bytes)	00 to 03	Range= –8388608 thru 8388607 Format: 8-digit BCD	00 to 03	Read	←	
Offset Value (4 bytes)	04 to 07	Range= –8388608 thru 8388607 Format: 8-digit BCD	04 to 07	Read	↔	Write
Preset Value (4 bytes)	08 to 0B	Range= –8388608 thru 8388607 Format: 8-digit BCD	10 to 13	Read	↔	Write
Deceleration (4 bytes)	0C to 0F	Range= –8388608 thru 8388607 Format: 8-digit BCD	14 to 17	Read	↔	Write
Latched Count (4 bytes)	10 to 13	Range= –8388608 thru 8388607 Format: 8-digit BCD	20 to 23	Read	←	
Timebase (2 bytes)	14 to 15	Range= 1 thru 9999 (Total time=above value x 3ms) Format: 4-digit BCD	24 to 25	Read	↔	Write
Sampled Count (4 bytes)	16 to 19	Range= 0 thru 8388608 Format: 8-digit BCD	26 to 31	Read	←	

NOTE: In the vast majority of cases, you will use your RLL program to exchange data between the CPU and the HSC. However, if you have a handheld programmer, you can use AUX 47 to read and write shared memory. This can be helpful in some troubleshooting or startup situations. There are no commands available in *Direct*SOFT at this time to directly write data to shared memory.

The Two-Step Process for Writing Data to Shared Memory

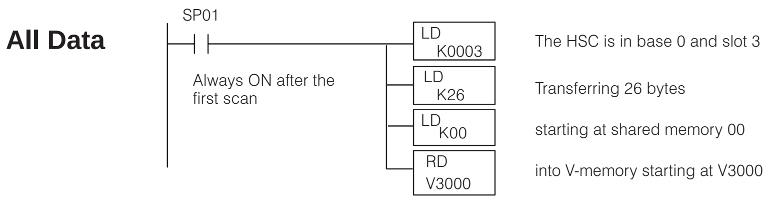
When using your RLL program to write data into shared memory, you must first load the data into V-memory. Then as a second step, you write it from V-memory into the shared memory area. The ladder logic below shows how you would use this two-step process to load a deceleration value into shared memory. Here we have loaded the deceleration value into its reserved address of shared memory. As you can tell from the table on the previous page, the starting address is hex 0C and it occupies 4 bytes.


Step 1
Load decel value into V2000
SP0 ON first scan only
LDD K6000
Number of pulses before you want deceleration to be turned ON
OUTD V2000
CPU memory area
Copy decel value from V2000 to shared memory
Step 2
SP0 ON first scan only
LD K3
Location of HSC in base: Base 0 and slot 3
LD K4
Transferring 4 bytes (deceleration value)
LD K0C
into shared memory starting at hex 0C
WT V2000
from V2000/V2001
Note: We are using SP0 above because it is ON for one scan only. We could just as easily have used any permissive contact instead. Keep this in mind for all future logic examples where we show the use of SP0.

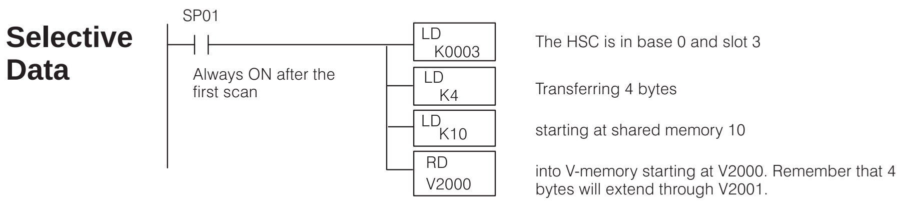
NOTE: If you are using the octal address of shared memory instead of hex, use LDA instead of LD. If you are loading a BCD number with more than 4 digits, use LDD.

Reading Data From Shared Memory

Quite often in your RLL you will want to move data the opposite direction—from shared memory to V-memory. The example below shows you how to read all 26 bytes (7 parameter values) from shared memory into V-memory.


All Data
SP01
Always ON after the first scan
LD K0003
The HSC is in base 0 and slot 3
LD K26
Transferring 26 bytes
LD K00
starting at shared memory 00
RD V3000
into V-memory starting at V3000

You don’t absolutely have to read all 26 bytes. You could only read those bytes you need. For example, you might wish to latch the count at some point. In that instance, the latched count value would be stored in shared memory (hex 10) and you could store it in V-memory for later use.


Selective Data
SP01
Always ON after the first scan
LD K0003
The HSC is in base 0 and slot 3
LD K4
Transferring 4 bytes
LD K10
starting at shared memory 10
RD V2000
into V-memory starting at V2000. Remember that 4 bytes will extend through V2001.

Understanding How Numbers are Stored In Shared Memory

Why is it important?

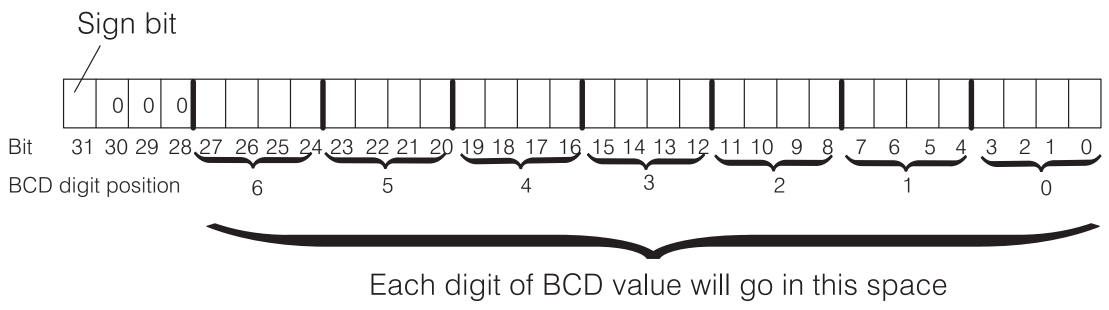
With the exception of the sampling timebase, all of the numbers stored in shared memory are 4 bytes in size. With the exception of the sampled count these numbers can either be positive or negative. If you are not familiar with a signed bit BCD format, this can be confusing.

You need to know how the numbers are stored in memory, so that you do not make mistakes when writing to shared memory or become confused when using your Watch Window in ***Direct*****SOFT** to monitor the V-memory in which you are transferring shared memory data.

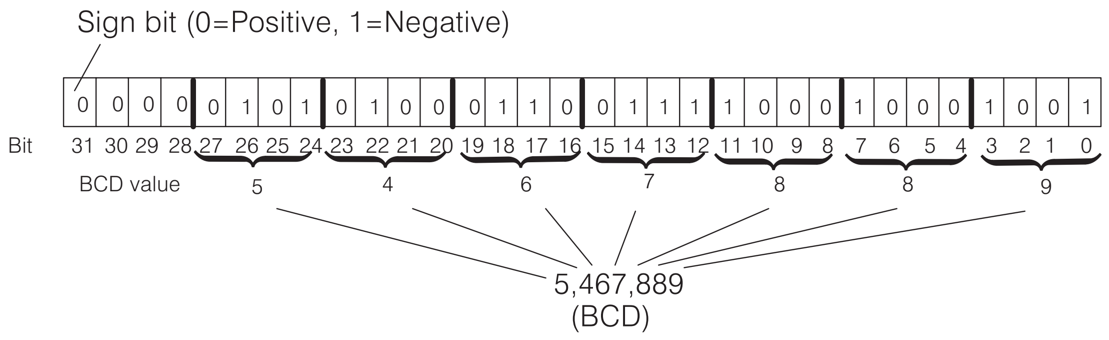
Allocating the 4 Bytes of Memory

With the exception of the timebase and the sampled count, the numbers available for each parameter of shared memory can have a maximum positive value of 8,388,607 and a negative value of –8,388,608.

28-bits of the 4 byte space are used for the absolute value of the number and the remaining 4 bits are used to indicate the sign. The diagram below explains how all the bits are used:


Sign bit
0 0 0
Bit 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
BCD digit position 6 5 4 3 2 1 0
Each digit of BCD value will go in this space

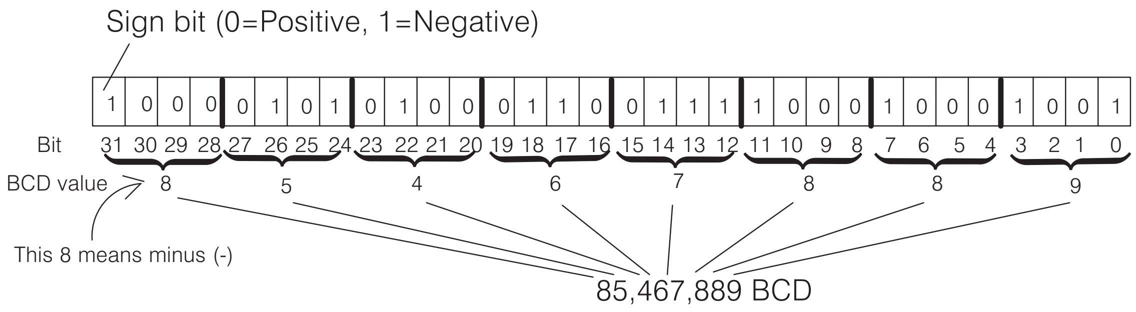
Take for example a preset value equal to 5,467,889. The diagram below shows how it is placed into the shared memory. Notice that the sign bit is zero. If you were to read this value from shared memory into V-memory; and then use the Watch Window to monitor its BCD value, you would see the BCD number 5,467,889.


Sign bit (0=Positive, 1=Negative)
0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 0 0 1
Bit 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
BCD value 5 4 6 7 8 8 9
5,467,889 (BCD)

But what would happen if this number was a negative number, say –5,467,889? The next page in this chapter will explain what you do.

Dealing With the Negative Numbers

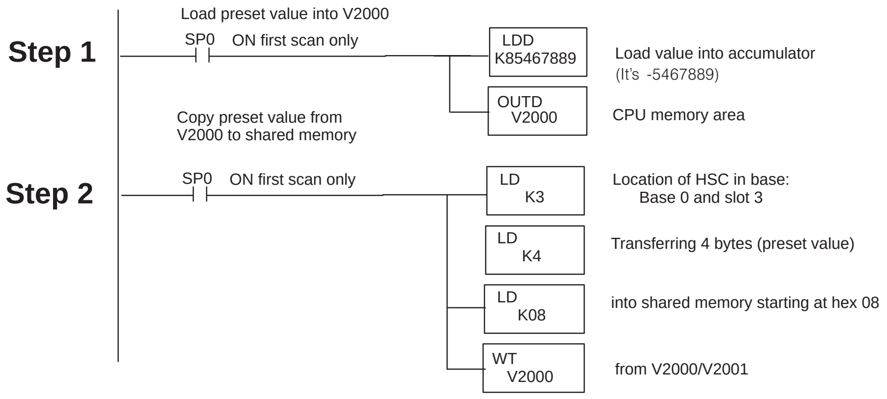
Continuing with our example from the previous page, the number –5,467,889 would have a 1 in the most significant bit (MSB) when placed in shared memory. What confuses many people is that this would show up in a Watch Window as 85,467,889 in BCD.



Placing a Negative Number into Shared Memory

Let's now take a look at some actual ladder logic to see how you would write a negative number into shared memory. In the example below, we are writing a preset into shared memory. Notice that we have used LDD to load values into the accumulator and OUTD to output them into CPU memory when they are more than 2 bytes in size. With the exception of the timebase, all of the shared memory data consumes 4 bytes of information each.

Ladder Logic Example


Load preset value into V2000
Step 1
SP0 ON first scan only
LDD K85467889
Load value into accumulator (It's -5467889)
OUTD V2000
CPU memory area
Copy preset value from V2000 to shared memory
Step 2
SP0 ON first scan only
LD K3
Location of HSC in base: Base 0 and slot 3
LD K4
Transferring 4 bytes (preset value)
LD K08
into shared memory starting at hex 08
WT V2000
from V2000/V2001

Notice that we loaded an 8-digit number with an 8 as the first digit. Remember you are loading a BCD number and not a decimal value. Remember that the MSB is always "8" when you are writing a negative number.

The Next Chapter

In the next chapter, you will learn how to put much of this knowledge to work by writing the initial relay ladder logic that you need in order to setup your application.