Programming: Variables

Programming: Variables

Variables are data holders that you can set and change within the program or over the communication channel

The first 26 variables are long integers (32 bits) and are accessible with the lower case letters of the alphabet,

a=#	Set variable a to a numerical value as
a=exp	Set variable a to value of an expression

A variable can be set to an expression only one operator and two operands. The operators can be any of the f

+	Addition
-	Subtraction
*	Multiplication
/	Division
&	Bitwise AND (see appendix A)
|	Bitwise OR (see appendix A)

The following are legal:

a=b+c,	a=b+3	a=5+8
a=b-c	a=5-c	a=b-10
a=b*c	a=3*5	a=c*3
a=b/c	a=b/2	a=5/b
a=b&c	a=b&8	
a=b|c	a=b|15	

ARRAYS

xxx, yyy, zzz. The memory space that holds these 52 variables is more flexible however. This same variable s
with an array variable type. An array variable is one that has a numeric index component, so that which variable a program changes
can be selected numerically, and therefore mathematically. This memory space is further made flexible by the fact that it can hold 52
thirty two bit integers, or 104 sixteen bit integers, or 208 eight bit integers (all signed). The array variables take the following form:

ab[i]=exp	Set variable to a signed 8 bit value where index i = 0...200
aw[i]=exp	Set variable to a signed 16 bit value where index i = 0...100
al[i]=exp	Set variable to a signed 32 bit value where index i = 0...50

The index i may be a number, a variable a thru z, or the sum or difference of any two variables a thru z (variables only).

The same array space can be accessed with any combination of variable types. Just keep in mind how much space each variable
takes. We can even go so far as to say that one type of variable can be written and another read from the same space. For example,
if you assigned the first four eight bit integers as follows:

```
ab[0]=0
ab[1]=1
ab[2]=0
ab[3]=0
```

that would occupy the same space as the first single 32 bit number, and due to the way binary numbers work, would make the thirty two bit variable equal to 256 (see appendix A).

A common use of the array variable type is to set up what is called a buffer. In many applications, the SmartMotor will be tasked with inputting data about an array of objects and to do processing on that data in the same order, but not necessarily at the same time. Under those circumstances it may be necessary to “buffer” or “store” that data while the SmartMotor processes it at the proper times.

To set up a buffer the programmer would allocate a block of memory to it, assign a variable to an input pointer and another to an output pointer. Both pointers would start out as zero and every time data was put into the buffer the input pointer would increment. Every time the data was used, the output buffer would likewise increment. Every time one of the pointers is incremented, it would be checked for exceeding the allocated memory space and rolled back to zero in that event, where it would continue to increment as data came in. This is a first-in, first-out or “FIFO”, circular buffer. You need to be sure that there is enough memory allocated so that the input pointer never overruns the output pointer.

STORAGE OF VARIABLES
(Not available in SMXXX5 SmartMotors)

Each SmartMotor has 8K of non-volatile EEPROM memory to store variables when they need to survive the motor powering down.

EPTR=exp   Set EEPROM pointer, 0-7999

To read or write into this memory space you first need to properly locate the pointer. This is accomplished by setting ‘EPTR’ equal to the offset.

VST(var,index)   Store variables

To store a series of variables use the ‘VST’ command. In the “var” space of the command you put the name of the variable and in the “index” space of the command you put the number of variables following that variable that you also want to store. Put a one here if you only want to store the one variable specified. The actual sizes of the variables will be handled automatically.

VLD(var,index)   Load variables

To load variables, starting at the pointer, use the ‘VLD’ command. In the “var” space of the command you put the name of the variable and in the “index” space you put the number of subsequent variables you want loaded.

FIXED or PRE-ASSIGNED VARIABLES

In addition to the general purpose variables there are variables that are gateways into the different functions of the motor itself.

@P	Current position
@PE	Current position error
@V	Current velocity
ADDR	Motor’s self address
CHN0	RS232 comm error flags
CHN1	RS485 comm error flags
CLK	Read/write sample rate counter
CTR	External encoder count variable
I	Last recorded index position
LEN	# of characters in RS232 buffer
LEN1	# of characters in RS485 buffer
F=#	Set factory defined state, where # != 1
F=1	Decelerate on contact with limit switch
MFRATIO	Magnitude of (MFMUL<<24)/MFDIV