USING PROGRAM VARIABLE STATEMENTS

INTRODUCTION

The program variable statements are featured in most of our flow computers. It lets users create their own program in the event that the flow computer doesn't support a particular feature. Some of these features are for instance accumulators, timers, hourly totalizers and unit conversions.

OPEARTIONS

The following are the mathematical operations allowed in the statements:

Function	Symbol	Description
ADD	**+**	Add the two variables or constant
SUBTRACT	**-**	Subtract the variable or constant
MULTIPLY	*****	Multiply the two variables or constant
DIVIDE	**/**	Divide the two variables or constants
CONSTANT	**#**	The number following is interpreted as a constant
POWER	**&**	1st variable to the power of 2nd variable
ABSOLUTE VALUE	**$**	unsigned value of variable
EQUAL	**=**	Move result to another variable
IF STATEMENT	**)**	Compares the variable to another Examples: • 7801)T7835 (if variable greater or equal to 1 go to 7835) • 7801)7802=#0 (if variable greater or equal to 1 then set variable 7802 to 0)
GOTO STATEMENT	**T**	Go to a different statement (forward only) Example: • 7801%#60T7836 (if variable equal to 60 then go to statement 7836)
COMPARE	**%**	Compare a value (EQUAL TO)
GREATER/EQUAL	**>**	Compare a value (GREATER OR EQUAL TO) Example: • 7801>7802T7836 (If variable 1 is greater or equal to variable 2 then go to 7836)
Natural Log	L	Natural Log of variable

PROGRAMMING TIPS

The statements provide great versatility but are not as flexible as a C coded program; some limitations have to be considered when creating your program:

- The program should be small. Currently we accept up to 69 statements (lines) per program.
- The program executes every second exactly so design accordingly. There is neither need nor support for loop instructions.
- The statements should not have **spaces.**
- Each statement can contain up to three variables or constants and separated by one of the mathematical functions. **4 digits are required** for referencing programmable variables, Boolean points or registers.
- Be careful where you store your results, registers in the 7000 area are floating while registers in the 5000 area are long integers which will result in decimal truncation if used as floats.
- Operations order of precedence: absolute, power, multiply, divide, add and subtract.
- Operations with same precedence are calculated left to right.
- The GOTO statement only allows **forward jumps**.

PREDEFINED REGISTERS

There are some predefined registers to facilitate the use of the program variable statements, these are:

- **Floating Point Scratch Pad:** this registers can be used as variables and temporary storage for floating values in your routine. Located in 7801 to 7830[¥].
- **Long Integer Scratch Pad:** Registers for variables and temporary storage of integer values. Located in registers 5031 to 5069. These are 32-bit integers so use registers with Odd addresses only[¥].
- **Hourly Program Variables:** Values in these registers are stored in the flow computer's historical database at the end of hour and then reset to zero. Located in 7071 to 7075[¥].
- **Daily Program Variables:** Values in these registers are stored in the flow computer's historical database at the end of day and then reset to zero. Located in 7076 to 7080[¥].
- **Monthly Program Variables:** Values in these registers are stored in the flow computer's historical database at the end of Month and then reset to zero. Located in 7081 to 7085[¥].
- **Display Program Variables:** Registers used to show calculated data into the flow computer's display. Simply go to the [Display Assignment] section and configure the display to show the program variables that you want. Registers located in 7791 to 7798[¥].
- **Program Variable Tags:** These are character registers for each of the hourly, daily & monthly program variables that provide a way to add a meaningful description for the variables.

[¥] Register addresses given are typical for most flow computer applications but they may not be right for your application. Please refer to user manual's Modbus Register Table for up to date information.

EXAMPLES

The following are a few sample statements¥:

Example 1:

Check for the Digital Input 1 every second and add its value (1 or 0) to register 7801

Line	Statement	Comment
7831:	7801=0001+7801	Add Digital Input 1 Value to 7801

Example 2:

Adding two registers into a third register.

Line	Statement	Comment
7832:	7803=7801+7802	7803 is equal to total of variable#1(Modbus addr.7801) and variable#2 (Modbus addr.7802)

Example 3:

EDS (Emergency shut down) program that sets the PID Control loop set point to zero if the ESD switch connected to status Input 1 comes ON.

Line	Statement	Comment
7831:	2751)T7833	If Status Input 1 is ON then go to line 7833
7832:	T7834	Go to Line 7834
7833:	4843=#0	Set PID loop set point to Zero
7834:	4843>#8274T7836	If PID set point greater than 8274 go to line 7836
7835:	T7837	Go to line 7837, basically skip next line
7836:	4843=#8273	Set PID Set point to 8273

Example 4:

Activate the switch output 1 when the user writes a 1 to register 7801 via Modbus. The output is enabled after a delay of 50 seconds.

Line	Statement	Comment
7831:	7801)T7833	If 7801 is 1 go to line 7833
7832:	T7846	If 7801 not 1 then exit the program
7833:	7810=7810+#1	Increase counter variable 7810 by one
7843:	7810>#5T7836	If counter 7810 is more than 5 go to line 7836
7835:	T7846	Counter not more than 5 exit the program
7836:	7810=#0	Clear counter variable 7810
7837:	7802=7802+#50	Increase delay counter by 50
7838:	4205=7802	Copy delay value to another register
7839:	4205>#500T7841	If delay greater than 500 go to line 7841
7840:	T7846	If not > 500 exit the program
7841:	2751=#1	Turn Switch Output ON
7842:	2751>#0T7844	If output ON then go to line 7844
7843:	T7846	If not ON then exit the program
7844:	7803=7803+#100	Increase Third counter by 100
7845:	7802=#0	Clear delay counter to start sequence again