

# Software Manual

for

# **SuprDAQ©**

Machine Controller

## 1.0 DiMAC

It is not necessary to learn how to "program" to be able to use a SuprDAQ®. It does not require "programming" in the traditional sense. It has a built-in capability to convert very simple, direct commands into whatever complex operations are required to accomplish the desired purpose.

There are no files to load, no "booting" process to endure and nothing to learn other than a few simple guidelines on the structure of the commands. The **Diva MACro** language is deceptively simple, yet very powerful. Our goal has been to provide the user with a tool that is intuitively easy to use, yet powerful enough to implement any desired application with ease. If the application requires no more than to determine the state of the logical inputs, then he need only learn one command, **TCn** (Tell Cchannel *n*). Other commands may be learned as the need arises.

There is no formal structure. Commands are executed immediately after a carriage return character is sent. There are no line numbers, no editor and no "Go To" commands, yet we have never heard of an application that was not practical, using **DiMAC** as the control language. Moreover, **DiMAC** is extremely compact. Very complex machine control programs may be reduced to 20 or 30 steps.

Of course the user may use any preferred language on any host computer to control the SuprDAQ. All that is required is to generate ASCII text script files and transmit them through the RS-232 port. Applications have been written in C, LabView, BASIC, VisualBASIC, Pascal, Delphi and others.

### 1.1 Terms Used

Throughout this manual certain abbreviations and standards will be used. Commands are shown in **boldface type**. Expected responses are enclosed in quotes. The words "carriage return" are a holdover from the days of the typewriter and refer to the code sent by the PC keyboard key labeled "Enter".

All commands use two-letter abbreviations, indicated in bold type such as **TP** that are either the initial letters of the operation to be performed, or standardized letters for these operations, where possible. **TA** is used for **Tell Analog**, so **RA**, Report virtual Accumulator has to be used, for example.

For example: since "ON" and "OFF" both begin with the letter "O", we have chosen the letter "N" to mean "ON", and "F" means "OFF", in all commands. Example: **EN** and **EF** are commands to turn the communications echo function "ON" and "OFF".

For those commands that require a value to be entered, it will follow the command letters. In the following discussion, it will be shown as *n*. The real-time syntax checker monitors the length of the number entered and flags errors when excess digits are entered..

## 1.2 Command Types

Over 50 commands are available for programming the **SuprDAQ**. Moreover, many additional application-specific commands have been developed and can easily be added to the SuprDAQ's table-driven command structure. Use these commands for controlling and reporting.

Wherever possible, the command structure has been designed to minimize the effort required to use the **SuprDAQ**. As Everett Long used to say, "Make it as easy to use as a screwdriver."

Commands may be executed in various ways:

<b>Single command</b> --	One function executed immediately
<b>Compound command</b> --	Multiple functions executed immediately
<b>Macro command</b> --	Compound command stored for later execution
<b>Sub-commands</b> --	Single-character commands

### 1.2.1 Single Commands

A **single command** is executed immediately after the carriage return is received and will be repeated each time the carriage return is received, until a different command is entered. With this feature, it is very easy to continuously monitor the state of an input by simply holding down the "Enter" key.

Examples: <b>TA2</b> (rtn)	: Report the value of analog input 2
<b>EN</b> (rtn)	: Echo all characters sent to the SuprDAQ
<b>TC3</b> (rtn)	: Report (tell) state of channel 3
<b>io</b> (rtn)	: Increment the analog output of both channels

Both uppercase and lowercase characters are valid, and spaces are allowed.

<b>TA</b> (rtn)	: Reports value of all analog inputs
<b>ta</b> (rtn)	: Reports value of all analog inputs

### 1.2.2 Compound Commands

A **compound command** is a series of single commands separated by commas rather than by a carriage return. In this way, it is possible to string together several commands before terminating with the carriage return. These multiple commands will be executed sequentially.

The syntax for a compound command is:

CMD[*n*], CMD[*n*], ..., (rtn)

Examples: **CN5,WA100,CF5** (rtn) ;Generates a 100 ms positive pulse on channel 5 every time the enter key is pressed.

**CN5,WA100,CF5,WA500,RP19** (rtn) ;Generates 20, 100 ms wide positive pulses on channel 5, with 500 ms delay between pulses.

A compound command such as the following example may be entered as one program line. It causes analog output 2 to generate a continuous 60 Hz sawtooth waveform with 16 steps from 0 volts to 5 volts. Channel 2 is incremented by 16 steps, followed by a delay of 1 ms.

Example: "**2IO16,WA1,RP,RP**" (rtn)

Once this command is entered, it remains in the buffer until replaced by another command and can be re-executed by transferring a carriage return. Compound commands may contain up to 18 single commands.

### 1.2.3 Macro Commands

Macros can be a most powerful tool for the programmer. A **macro command** is a grouping of commands to form a short program, implemented by a macro number.

To use macros for programming the **SuprDAQ** controller, insert an **MD** (Macro Definition) command as the *first* instruction in the command string. The syntax for macro commands is:

**MD**(macro#), followed by a compound command string.

Example: **MD3,CN5,WA100,CF5,WA500,RP19,RP4** (rtn)

In this example, **MD3** defines macro #3. To call up this macro, just issue the command **MC3**. In this case, the compound command from above was used. Since it generates 20, 100 ms pulses. MC3 will generate 100 pulses. (executed once and repeated four times, for a total of five executions.)

Macro commands may be stored in any order, but you may prefer to number them sequentially as they are entered, because the system gives no warning if you define (and overwrite) an existing macro. You may wish to do this under many conditions, such as when one macro is called by another. It is sometimes desirable to define a complex process in one macro and define parameters such as initial value, in another macro which is called by the main macro.

Macro commands can call another macro, without limit. For instance, MC1 could call MC2, and MC2 could then call MC3 and still be able to return to complete the remainder of MC1.

Example: MD1,MC2,MC3,MC4,MC5,MC6

Macro commands may contain up to 17 single commands.

### 1.2.4 Sub-commands

Sub-commands may be used at any time. They are most useful for interrogating variables without disturbing an operating program. An example would be a situation where a repetitive motion is in process, such as generating a complex pulse train. The operator would like to know the status of the command without stopping it. The sub-commands can be used to read the number of iterations in a loop, current system status, position, etc. A single character is also provided for emergency stop action.

#### Sub-commands

%	(ASCII 37)	<b>TS</b>	<b>Tell Status</b>
#	(ASCII 35)	<b>TC</b>	<b>Tell I/O Channel status</b>
\	(ASCII 92)	<b>TI</b>	<b>Tell Iteration count</b>
'	(ASCII 39)	<b>TA</b>	<b>Tell Analog value</b> (single quote or apostrophe)

### 1.3 Data formats

#### 1.3.1 Reporting Commands

**Reporting commands** cause the **SuprDAQ** controller to emit a string of data, whether a voltage, channel status, help or other information. These commands are easy to remember as they usually begin with a **Tell** statement. For example, **TC** (Tell Channel), or **TA** (Tell Analog value).

The **Tell** commands query registers which may be of different lengths. Accordingly, the reports have different formats. The following examples assume that **DM** (Decimal Mode) has been selected. Note that commands that return data usually have a format that specifies first the board address to identify the source of the data, then the channel that pertains to the following data, followed by a single letter identifying the command (usually the same letter used in the command), and then the data in either decimal or hex format, depending on the mode selected.

Example:     transmit:     **DM,TI ,HM,TI(rtn)**  
               receive:     "001+0000000255"  
                               "001000000FF"

Other formats exist for special purpose commands, such as **TO**, **VE** and **CS**. The format for these commands is as shown in the detailed command description.

## 2.0 Starting Communications

Note: We strongly urge the use of a terminal emulator for initial testing and familiarity with DiMAC, even if the intended use is through a host program.

The first command to test the communications could be a **VE**rsion command.

Type in a **VE** and "enter". On your terminal monitor a message similar to the following should be displayed:

```
"(c) 2006 DIVA Automation"  
"Ver. 1.0 for SuprDAQ, 2 Oct 2006"
```

If you get a response (as above) on your terminal, the **SuprDAQ** is ready for use. Now all registers are loaded with their default values. Most tasks may be accomplished using only these default values--other tasks may require value modifications.

If you do not see this response, there are several possible reasons:

- ◆ Check the power supply voltage to the **SuprDAQ**. It must be within the specified range and must be connected with the proper polarity as shown on the connection diagram.
- ◆ Check the cable to your terminal. It should be connected to the TxD and RxD pins from the **SuprDAQ**, as well as to ground. Various terminals have differing pin assignments for TxD and RxD, so it may be necessary to reverse pins 2 and 3 on the RS-232 connector. No other connections are required. The **SuprDAQ** uses buffering, rather than handshaking, to provide error-free communications, so it may be necessary to disable handshaking on your terminal.
- ◆ Check the baud rate on your terminal. The **SuprDAQ** is designed to operate at 9600 baud as delivered. The baud rate can be changed by command after communications, but the terminal must be set to the same baud rate as the **SuprDAQ**.
- ◆ If you are using a PC-compatible computer, the HyperTerm program supplied or available for free download, is an excellent terminal emulator.

If this response does not appear, there is no point in going further until it does. If communications are not possible, please report the problem to us. We will respond as quickly as possible.

### 3.0 SuprDAQ Commands

All **SuprDAQ** commands use two-letters to identify the type of operation desired, followed by any pertinent data value. For example, **HE** by itself is adequate to display a list of available commands, but **IO** alone would be useless because the system would not know how much change to make in the analog outputs. All commands are tested for acceptability as they are entered. In the case above, **HE9** would generate an error because it does not accept data, and **IO** followed immediately by a carriage return or comma would also be rejected. In the case of **LA**, it can be followed by a minimum of 1 and a maximum of 9 digits, to accommodate the allowable range. Single byte quantities such as analog output setting command **SO** accept 1 to 3 digits, and so on.

**SuprDAQ** commands are arranged by group in the following discussion.

#### 3.1 UTILITY COMMANDS

##### **DM**      **D**ecimal **M**ode

The **DiMAC** program can handle both decimal and hexadecimal numbers. The **DM** command selects decimal mode. After entering this mode, all inputs are interpreted as decimal numbers and the system reports requested values in decimal as well, with the exception of certain commands, such as **CS** (checksum), which always reports in hex format.

The selected number mode is one of the parameters that is restored from EEPROM at reset..

##### **HM**      **H**exadecimal **M**ode

After transmitting this command, all inputs and outputs are in hexadecimal mode.

Example: **HM,2SO7F (rtn)**      : Select hex mode and set analog output channel :2 to 7F (Hex).

##### **EN**      **E**cho o**N**

Enables echoing of command characters as they are entered. Each character received is echoed unchanged. This is a very useful feature when the SuprDAQ is being controlled manually from a terminal and may be used as a high-level handshake to confirm proper command reception when controlled by a host program.

**EF**      **E**cho of**F**

Disables echoing. When control is from a computer program, it is sometimes easier to program if there is no echo, unless the program uses it for verification of successful transmission.

**BR(n)**      **B**aud **R**ate    ( $0 < n < 3$ )

Sets baud rate according to the following table:

n = 0	4,800
n = 1	9,600
n = 2	19,200
n = 3	38,400

**RT**      **R**ese**T** system

Performs a complete restart of the system.

**DE**      **D**Ebugger

Enters on-board monitor. Instructions for the use of this command are contained in a separate appendix. Please contact the factory if there is a need for it.

**DAn**      **D**efine controller **A**ddress = **n**    When multiple SuprDAQs are connected to a single RS-232 communications channel, it is necessary to assign each of them a unique address. The procedure is simply to connect each controller individually and use the DA command to assign a unique address to each. Afterwards, they may be daisy chained without communications conflict.

**IN**      **I**nterrupts enabled (**oN**)    The interrupts used by the SuprDAQ include serial data input and real time clock (RTC) operations. When it is necessary to pause these operations for any reason, the IF command can be used to disable them. They can then be re-enabled with the IN command.

**IF**      **I**nterrupts disabled (**ofF**)

**PSn**      **P**rint string **#n**. If  $n > 256$ , then it is a system string, stored in ROM. In the first release, these are the only strings supported.

### 3.2 SEQUENCING COMMANDS

#### **RP***n* RePeat *n*

This command causes the command string to repeat *n* times. If *n* is not specified, the commands are repeated 65,535 times. The repeat loop may be interrupted by transferring an escape character, which is defined as the backspace. (To repeat forever, use two **RP** commands in sequence.)

Example: **TT,WA500,RP99** (rtn) : Will display the time of day every 500 milliseconds (0.5 second) for a total of 100 times.

**TT,WA,RP,RP** (rtn) : Will display the time every second until interrupted by the escape character.

#### **WAn** WAit *n* milliseconds

This command inserts a wait period of *n* milliseconds before going to the next command. If *n* = 0, then it is set to 1000.

Example: **SO20,WA3000,TA1,SO120** (rtn)

This command line will set the output to  $(20/256 * 5)$  **391 mv**, wait for 3 seconds, then to  $(120/256*5)$  **2.344V**.

### 3.3 I/O COMMANDS

**CIn** Channel *n* is an Input

Sets the direction of channel *n* as an input. CN or CN0 sets all inputs.

**CTn** Channel *n* is an ouTput

Sets the direction of channel *n* as an output. Unless this command is used to define the channel direction, the CN and CF commands will have no effect on that channel. CT CT0 will set all channels as outputs.

It is always possible to read any channel, whether set as an input or an output. However, setting it as an output can interfere with the ability to read an external signal, depending on the relative source impedances if one is high and the other low.

**Important Note:** If a channel is defined as an output, do not connect it to a low impedance source. The I/O drivers are active pull-up/pull-down and may be damaged if there is no current limiting to an external source. If any channel is connected to an external low impedance source, the command to make all outputs simultaneously (CT[0]) should not be used.

**CNn** Channel *n* oN

Sets channel *n* to high level. Also sets direction of channel *n* to output. The state may be read with the **TC** command, regardless of the direction.

**CFn** Channel *n* ofF

Sets channel *n* to low level. Also sets direction of channel *n* to output. The state may be read with the **TC** command, regardless of the direction.

**CPn** Set Channel Pattern to *n*

Sets all channels previously defined as outputs to the level described in hex format by *n*. Does not set direction of channels to output. All sixteen channels may be set simultaneously with this command, or any subset. It is the responsibility of the user of this command to prevent accidental change of a previously set output.

For example, if channel 3 had been set oN by a previous command and it is to remain oN, then the user must make certain that the bit for channel 3 is high in all subsequent **CP** commands. If it is not practical or desirable to do so, then the use of this command should be avoided. It provides a very fast means of setting all outputs with a single command, but carries a responsibility with its use.

The format of *n* is as follows:

<u>Channel</u>	<u>Decimal value</u>	<u>Hex value</u>
1	1	1
2	2	2
3	4	4
4	8	8
5	16	10
6	32	20
7	64	40
8	128	80
9	256	100
10	512	200
11	1024	400
12	2048	800
13	4096	1000
14	8192	2000
15	16384	4000
16	32768	8000

To set channels 3,6,8 and 14 o**N**, add the values corresponding to those channels. In this case, 4+32+128+8192=8356 for DM and 4+20+80+2000=20A4 for HM. Any output channel not included will be set to of**F**. Channels defined as inputs will not be affected.

Example:    **CP8356**                    (Normal mode after reset is **DM**)  
                  **HM,CP20A4**            (Mode remains selected until reset)

**aSO $n$**         Set analog Output **a** to **n**    (**1 > a > 2**)    (**0 > n > 255**)

Sets the specified channel output to the value of **n**. If **a** = 0, then both channels are set. The output voltage is determined by the ratio of **n/256** to 5 volts.

Example:    **2SO128** sets the voltage at output #2 to 2.50 volts.

**aIO $n$**         Increment analog Output **a** by **n**        (**-255 > n > 255**)

**IO** is useful for adjusting the output in variable size steps or for creating a low frequency waveform. A simple method of testing the output stage is to use the command string: **1IO16,RP,RP**. This creates a 16-step sawtooth wave on D/A output #1 that repeats forever. If only one RP command were used, the command would run to completion very quickly because of the 65,536 limit of a single RP command. The repeat count of the second RP command is overwritten by the first command when it repeats, which results in continuous operation until reset or **interrupted by a backspace character**.

To easily adjust an output voltage, use the SO command to set an approximate voltage, then use the IO command with repeated carriage returns to adjust the output as desired. Example: **1IO-2** will decrement channel 1 by 2/256 (approx. 39 mv) each time the ENTER key is pressed.

Note: The output value is not clamped at maximum output. If the commanded value exceeds 255, the output rolls over modulo 256. In other words, **1SO250,IO10** is equivalent to **1SO4** because  $250+10-256=4$ .

The **TO** command may be used to report the level of the output, which may not be known to the operator after a series of increments..

### 3.4 REPORTING COMMANDS

**Reporting commands** cause the **SuprDAQ** controller to emit a string of data, whether a position, target, help or other information. These commands are easy to remember as they usually begin with a **Tell** statement. For example, **TT** (Tell Time), or **TO** (Tell Output).

The **Tell** commands have been structured to display only the maximum number of allowable digits. Accordingly, the reports have different formats. The following examples assume that **DM** (Decimal Mode) has been selected. In the first example, if **HM** (Hex Mode) had been selected, there would be only 8 digits after the letter 'T.'

Example for 32-bit register query:

```
transmit: RA (rtn)
receive: "00T:+0000000000"
```

Example for 8 and 16-bit register query:

```
transmit: TQ (rtn)
receive: "00Q0255" (in DM)
         "00QFF" (in HM)
```

#### **VE** Version report

Reports the copyright notice and revision level of the installed firmware. This is the default command loaded into the command buffer on startup. If the first character received is a carriage return, this command should be executed.

#### **TI** Tell Iterations

This command reports the state of the repeat counter. It is useful for determining the number of times a repetitive action has taken place.

Example: **CN8,WA250,CF8,WA250,TI,RP99**. Channel 8 will be pulsed high and low in a square wave, for a total of 100 times. It will report the number of iterations remaining to be performed after each iteration.

```
Report: "00I+0000000000" (First TI is executed before number of loops is
         "00I+0000000099" specified)
         "00I+0000000098"
         (97-03
         "00I+0000000002"
         "00I+0000000001"
```

**TA<sub>n</sub>**      Tell Analog channel *n*

If *n* = 0, reports the value of all eight analog input channels as 0-255. The normal reference value is +5 volts, so the conversion from reported value to volts is the ratio of the reported value to 256, multiplied by 5. A report of 237 relates to **4.63** volts. (237/256\*5) If *n* > 0, then only the specified channel is reported.

**TO<sub>n</sub>**      Tell Output channel *n*

Reports the value of the present analog output setting. Normally used when the output is controlled manually with the **SO** and **IO** commands. The same scale factor calculation pertains, as described for the **TA** command.

**Tc<sub>n</sub>**      Tell Channel *n*

Reports the logic state of the digital input channels. When *n* = 0, all 16 channels are reported in two bytes. See the **CP** command description for the method used to interpret the reported value by individual channels.

When *n* > 0, then only that channel is reported, and the format is **0C<sub>n</sub>:0** or **0C<sub>n</sub>:1**.

**TS**          Report status

Reports the state of the internal flags that control operation of the program.

A single byte combining eight flags is reported. The bit weights corresponding to each flag is listed below:

ECHO	=	1	;	ECHO ENABLE
TWAITF	=	2	;	WAIT IN PROGRESS
ERRFLG	=	4	;	ERROR IN COMMAND INTERPRETER
LZ	=	8	;	LEADING ZERO SUPPRESSION
PRINT	=	16	;	PRINTER ENABLE
NUMMOD	=	32	;	DECIMAL MODE
ADR	=	128	;	BOARD IS ADDRESSED

**TT**          Tell **T**ime of day

The time elapsed since the clock was set using the **SC** command is reported in HH:MM:SS format.

**HE**      HElp

This command reports the valid mnemonics for the installed firmware version. It lists all two-letter commands supported by your software version.

**TM**[*n*] Tell Macros (0 < *n* < ??)

Displays all previously stored macro commands. If *n* = 0 or, if *n* is not specified, all macros will be displayed. This is the reason for a separate command (below) for reporting macro #0. Since macros may be defined in any sequence, the **TM** command is useful for confirming the existence of, as well as the contents of, macro commands.

**TZ** Tell macro Zero

Special case of the **TM** command for Macro command zero, which is handled differently from other macros. If there is a macro zero loaded, it will be executed automatically when power is applied. This is the mechanism by which totally turn-key operation is accomplished.

Macro zero may be used only to set up initial settings parameters before manual control is begun or it may then transfer control to other macros for stand-alone operation.

**TB** Tell Board address

Reports address of enabled board. If more than one board is accidentally enabled, this command will help detect it. It is useful in confirming both the presence of a **SuprDAQ** having the most recent address, and for confirming successful reception of an address change. Please see discussion of multi-drop addressing for the **SuprDAQ** in the appendix.

Report: "A0000"

**CS** CheckSum

Calculates and reports the sum of the entire contents of the **SuprDAQ** operating system program in memory. It does not include the macro storage space. Useful for conducting a confidence test of the **SuprDAQ**. The reported value should remain constant. If not, there is a problem and the **SuprDAQ** should not be used for critical operations until the problem is resolved.

Report: "C53A"

**3.5 MACRO COMMANDS**

**MCn** execute **M**acro **C**ommand *n*

This command may be used to implement a previously defined macro command. If there is no macro defined by the number *n*, no action will be taken.

Example: **MC3** (rtn) : Will execute macro #3

Before calling MC3, that macro must have been defined with the **MD** command.

**MDn** **M**acro **D**efinition

This command is used to define a new macro command. Any duplication of numbers will simply result in the loss of any previously defined macro using that number. To define a macro, choose any number in the allowable range for the new macro and enter MD followed by this number and a comma before entering the function you wish the macro to perform, in the normal manner.

Example: **TT,TP** (rtn) :Tells target followed by position  
**MD4,TT,TP** (rtn) :Creates a macro command to Tell Target, then  
:Tell Position and assigns it to number 4.  
**MC4** (rtn) : Executes macro #4  
:(Same as entering **TT,TP** (rtn))

**Important note:** If there is a Macro #0 defined, it will be automatically executed one second after power is applied or a reset (**RT**)command is issued. Although this is a very powerful and useful capability, it may also create a problem if for some reason it is no longer desired, or has become corrupted. In order to allow the user the opportunity to modify an existing MC0 without executing it, any keyboard character may be used to interrupt operation during the one second delay. Control simply bypasses MC0 and passes to the normal command loop. From this condition, MC0 may be reset or redefined as desired.

**RMn** **R**eset **M**acro *n* space (*n* = 0 resets all macros)

Used only to initialize the memory reserved for macro commands. It clears the macro storage.

Example: **RM** (rtn) : Removes all stored macros

**RZ** **R**eset macro zero space

A separate command is required for macro zero because *n* = 0 is used as a special case for **RM**.

### 3.6 Sequencing Commands

**WN $n$**       Wait until channel  $n$  is **oN**

Command execution is paused until the specified channel is high. If already high, there is no delay.

**WF $n$**       Wait until channel  $n$  is **ofF**

Command execution is paused until the specified channel is low. If already low, there is no delay. **WF** and **WN** may be used together to synchronize execution of a loop to the actuation of a switch.

Example: **WF3,WN3,2IO1,RP** This will cause the program to wait until channel 3 is low (perhaps from a switch closure), then continue waiting until it is high (the switch is released) before incrementing the output. Each time the switch is pressed and released, the output will be incremented. Note that without the **WN** command, the loop would be executed as fast as the program could loop. The output would be incremented several thousand times each second, as long as the input is low. There may be applications where it is desired to have a simple gating action such as this, but it is important to understand the possibility of such operation.

Another sequence option would be to insert short delays to guard against multiple executions caused by "switch bounce". Many mechanical switches bounce away from the contact on actuation. An effective means of preventing response to the extra indications is to wait until the switch has had time to settle against the contact. Typical switches will settle within 10-15 milliseconds. We can insert a 20 ms delay after each of the wait commands to insure a stable condition before continuing.

**WF3,WA20,WN3,WA20,2IO1,RP** will accomplish this function. These delays are short in comparison with manual operation.

**XN $n$**       eXecute command only if channel  $n$  is **oN**

When it is desirable to perform an action only if certain conditions are met, then the **XN** command is very useful.

Example: **XN12,MC15** If we assume that macro 15 performs an operation that would be unnecessary, undesirable or dangerous if it were performed when channel 12 is low, then this sequence will prevent it from occurring. An example might be if there is an input from a limit switch of some kind or from a system logic controller that indicates when a container is available for filling. If the task is to fill bottles with a liquid, it is undesirable to dispense the liquid if the bottle is not present.

**XF $n$**       eXecute command only if channel  $n$  is **ofF**

**Counter control commands**

**aCCn**                    Configure counter **a** mode = **n**

The SuprDAQ provides two general purpose counter inputs. (X&Y). Each counter has multiple operating modes. They may be used to measure the time between input pulses, or count input pulses or measure the width of the input pulse or generate an output pulse stream.

Both counters are organized as down counters. The mode selection codes are:

1 = Timer mode                    Counts 500 Khz pulses.

2 = Pulse out                    The output is inverted every time the value set with SX or SY is counted down by a 500 Khz clock.

3 = Event counter 1            Each positive transition of the input signal is counted. The signal must remain at each state for at least 80 ns, with a minimum cycle time of 200 ns (5 MHz).

4 = Event counter 2            Each negative transition of the input signal is counted. The signal must remain at each state for at least 80 ns, with a minimum cycle time of 200 ns (5 Mhz).

5 = Pulse width 1            500 Khz pulses are counted while the clock input signal is high.

6 = Pulse width 2            500 Khz pulses are counted while the input is low.

**RC n**            Report counter **n** contents. If **n** = 0, then report both.

**SX n**            Set X counter = **n**

**SY n**            Set Y counter = **n**

**Virtual Accumulator commands**

**AM n**            Multiply virtual accumulator contents by **n**

**LA n**            Load virtual accumulator with the value **n**

**OA n**            Set output **n** to value in virtual accumulator LSB

**AA n**            Add **n** to virtual accumulator

**RA**            Report contents of virtual accumulator

**AD  $n$**       Divide virtual accumulator by  $n$

This is a binary division. The contents of the virtual accumulator are divided by two with each unit of division. AD1 divides by two. AD8 divides by 256, etc. This command is normally used in conjunction with an AM command to achieve an effective real number multiplication or divide function.

For example, AM3,AD1 results in the virtual accumulator having been multiplied by 1.5, (3/2). This technique can be used to great advantage in converting integer values to engineering units, such as converting the analog inputs to read in volts, rather than bits, or temperatures to read out in degrees Fahrenheit or Celsius.

**GA  $n$**       Get value of analog channel  $n$  into virtual accumulator

**AS  $n$**       Store contents of virtual accumulator LSB at  $n$

The least significant byte of the four-byte virtual accumulator is stored at the address specified by  $n$ . This feature allows the user to control any aspect of the controller. Using a series of LA and AS commands, a program may be downloaded to RAM, for instance, and executed using the AX command. Or, a single digital output port could be set or cleared. Care should be taken when using this command because of the power it provides the user, with no safeguards.

**Command Summary****Reporting commands**

VE Display version number  
TI Report the iteration number  
TS Report status  
HE Print list of commands  
TB Report board address  
TM Report macro contents  
TZ Tell Macro Zero  
TA Tell Analog input value  
TC Tell I/O channel  
TO Tell D/A output setting  
CS Perform self-test checksum  
TT Tell time of day

**Utility commands**

DM Input and output in decimal format  
HM Input and output in hex format  
RT Reset all parameters to default values and do power-up start  
EF Turn off echo to RS-232 port  
EN Turn on echo to RS-232 port  
BR Set baud rate  
DE Enter debugger  
DA Define controller address  
IN Interrupts enabled (on)  
IF Interrupts disabled (off)  
PS Print string  
ST Set time of day

**Motion and sequencing commands**

WA Wait specified time  
RP Repeat from beginning of line n times  
WN Wait until specified I/O channel = 1  
WF Wait until specified I/O channel = 0  
XN Execute only if specified I/O channel = 1 (If-then-else)  
XF Execute only if specified I/O channel = 0  
WG Wait until specified channel is greater than n  
WL Wait until specified channel is less than n  
XG Execute if specified channel is greater than n  
XL Execute if specified channel is less than n  
DN Do if channel n is on  
DF Do if channel n is off

**I/O Commands**

CF Channel off  
CN Channel on  
CI Channel In  
CT Channel out  
CP Channel pattern (not yet implemented)  
SO Set analog output voltage  
IO Increment analog output setting by n

**Macro commands**

MD Macro definition  
MC Macro command  
RM Reset macro  
RZ Reset macro zero  
TM Report macro contents  
TZ Tell Macro Zero

**Counter control commands**

CC Configure counter mode  
RC Report counter contents  
SX Set X counter  
SY Set Y counter

**Virtual Accumulator commands**

AM Multiply virtual accumulator contents by n  
LA Load virtual accumulator with n  
OA Set output to value in virtual accumulator LSB  
AA Add n to virtual accumulator  
RA Report contents of virtual accumulator  
AD Divide virtual accumulator by n  
GA Get value of analog channel n into virtual accumulator  
AS Store contents of virtual accumulator LSB at n

**LCD commands** (not yet implemented)

LS Put string n to LCD  
LR Reset LCD  
LC Set LCD cursor to n  
LB Backspace LCD cursor