APImate 1.0 Software User Guide

Installation User's Guide

Doc. #151-APImate-1.0 Rev B

API Controls

45 Hazelwood Drive
Amherst, New York 14228
(716) 691-9100 • Fax (716) 691-9196 • (800) 566-5274

PREFACE

Congratulations!

You've just purchased the finest programmable microstep drive in its class. The Intelligent step drive is designed to provide reliable long-term and economical operation in demanding field environments. With integrated I/O the drive provides ease of field installation and service.

The Intelligent Step Drive series are a new generation of programmable stepper drives. The drive utilizes a single micro processor to interpret programming steps and send current to the motor. Unlike the traditional drive/indexer which, uses duel processors, it does not generate a step and direction signal. All models can be configured for 200 step/rev or 100 step/rev 2 phase step motors. The drive's I/O are fully programmable to allow maximum flexibility in your application.

The drive utilizes "Midband Stabilization" to help prevent motor stall do to resonance. Utilizing the Inductance and Midband compensation feature the performance can be optimized for the selected motor. By using the drive's auto-switching to "Full Step" mode you get up to 30% more torque then traditional microstepping drives.

Each Intelligent Stepper Drive is a controller, step drive and heatsink integrated into a single stand-alone package. The compact size and integrated design simplifies the installation process and reduces downtime should a need for replacement arise. Since the drive utilizes "Flash" memory it can be preprogrammed and stored indefinitely. The drive firmware is stored in "Flash memory that can be easily upgraded through the serial port to take advantage of future API enhancements.

Fundamentally, an Intelligent Drive is a computer that is dedicated to motion control. It has it's own operating system, data storage, data ,manipulation capabilities and interface for data communication. The localized inputs/outputs allow for hard wire connection to ensure the drive is in synchronize with its environment.

- Communication interface to a PC is via RS-232, 422 or 485 for downloading programs or sending commands directly to the drive.

- **APImate 1.0** software provides easy point and click programming. The motion programs can be executed through RS-232/422/485 or by utilizing the sequence inputs for standalone application.

- Methods of control include standalone, analog control (optional), or sending commands directly from a PC through the communication port. Commands can be sent directly to the drive, from a host PC, utilizing a hexadecimal format. The I-Drive Protocol document, loaded to the PC with APImate, describes the command structure and format. (**C:/Program Files/API/APImate/IDRIVE Protocol.doc**)

Typical operation methods include standalone using an Autostart sequence and/or the sequence select inputs or sending commands directly through the communication port.

- When only a single program is required it can be uploaded to the drive and executed at power-up by defining it as the "Autostart" program.

- When multiple programs are required they can be uploaded into the drive and executed using the "Sequence Select" and "Sequence Start" inputs. The program being executed is based on the combination of the sequence select inputs when the sequence start input becomes active.

- Commands can be sent directly from the PC to up to 31 drives per serial network using RS-232/425 or 485. The RS-232 is multi-drop with each drive having a unique node identification number. The commands are sent in a hexadecimal format using the I-Drive Protocol.

- Some applications require a combination of stored programs and sending commands using the I-Drive Protocol. Examples include: changing user variables; monitoring drive parameters; begin program execution.

There are several unique features in an Intelligent Step drive compared to the traditional drive/indexer. When designing your application, keep the following APImate 1.0 features in mind.

- The values of drive and user variables are universal regardless of sequence number. The variables are not stored in non-volatile memory and are reset to zero whenever the drive is powered down or reset.
- Loops and comparative statements are handled by “While” and “If” statements. There are no “Go To” or “Jump” commands allowed within a sequence.
- A second sequence can not be called or executed within a sequence. Once the sequence finishes execution a new sequence can be executed.
- The sequence commands execute one after another. To control the flow of program execution use the “Wait” commands. (Time Delay, Wait on Input and Wait for Motor Stop)
- Since the I-Drives use a single multi-drop RS-232 to connect up to 32 drives on a single communication port. The drives can only send information when requested by the main controller. This prevents communication problems of multiple units sending information on the same RS-232 port.

Note: Refer to the glossary at the end of this document for definitions on unique terminology.

	NOTE
	Example programs are included with APImate 1.0 and can be found in the c:/Program Files/API/APImate/Examples folder. *For an explanation of the sample programs see “Read Me.txt” file.*

Changes in design, specifications and product improvements are subject to change without notice. All material is property of API Controls and can not be reproduced in full or part and distributed without prior written approval. from API Controls.

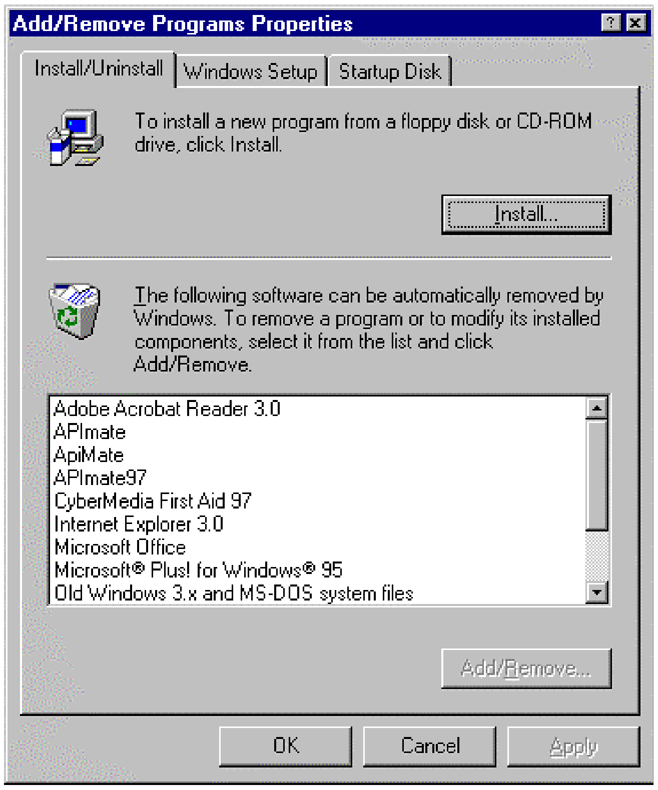
TABLE OF CONTENTS

1. Installation & Overview

This section demonstrates the correct procedures to load APImate and an overview of the APImate screen.

1.1 Installing APImate 1.0

When loading the APImate software always use the Add/Remove Programs located in Windows Settings/Control Panel. Uninstall any existing copies of APImate or APImate 97 before reloading. This will remove all existing .dll files, which are added to the Windows directory.

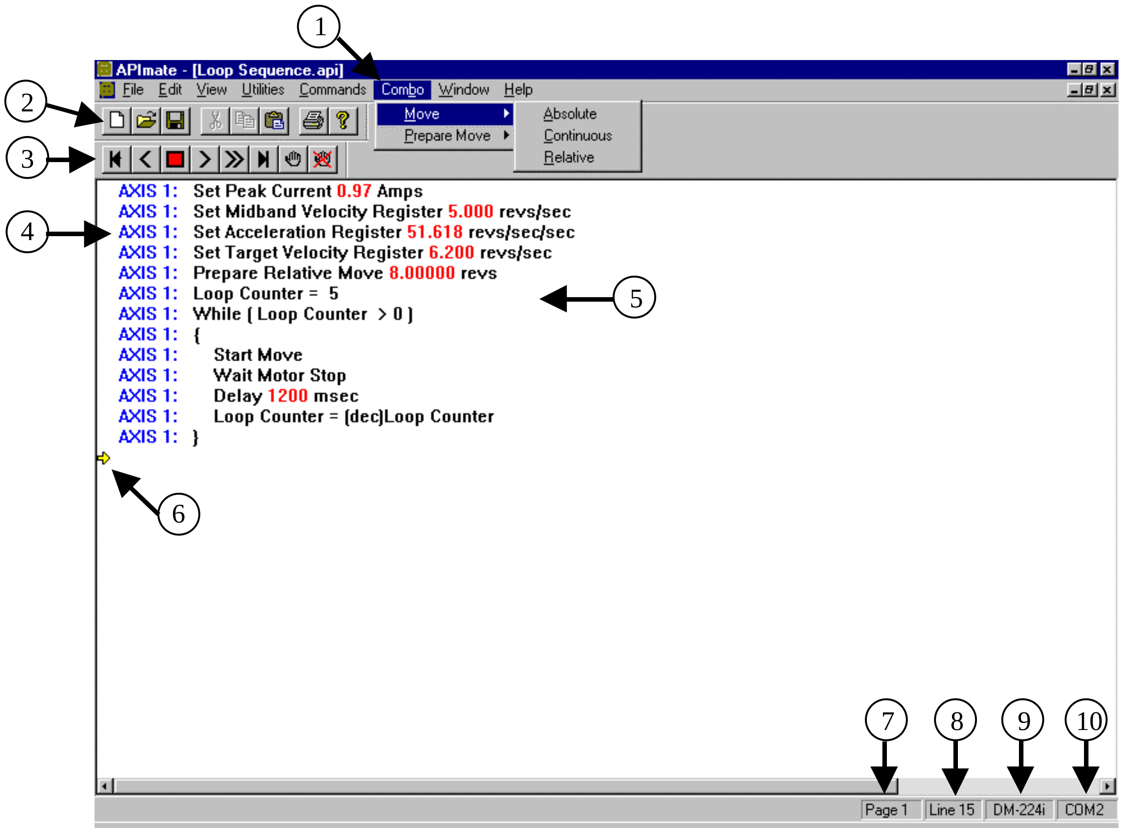


To install a new program from a floppy disk or CD-ROM drive, click Install.

The following software can be automatically removed by Windows. To remove a program or to modify its installed components, select it from the list and click Add/Remove.

Adobe Acrobat Reader 3.0
APImate
ApiMate
APImate97
CyberMedia First Aid 97
Internet Explorer 3.0
Microsoft Office
Microsoft® Plus! for Windows® 95
Old Windows 3.x and MS-DOS system files

1.2 Overview of APImate 1.0


1
2
3
4
5
6
7
8
9
10
APImate - [Loop Sequence.api]
File Edit View Utilities Commands Combo Window Help
Move
Prepare Move
Absolute
Continuous
Relative
Page 1
Line 15
DM-224i
COM2

```
AXIS 1:  Set Peak Current 0.97 Amps
AXIS 1:  Set Midband Velocity Register 5.000 revs/sec
AXIS 1:  Set Acceleration Register 51.618 revs/sec/sec
AXIS 1:  Set Target Velocity Register 6.200 revs/sec
AXIS 1:  Prepare Relative Move 8.00000 revs
AXIS 1:  Loop Counter =  5
AXIS 1:  While ( Loop Counter  > 0 )
AXIS 1:  {
AXIS 1:     Start Move
AXIS 1:     Wait Motor Stop
AXIS 1:     Delay 1200 msec
AXIS 1:     Loop Counter = (dec)Loop Counter
AXIS 1:  }
```

1. Menu

The pull down "Menus" contain the programming commands for APImate 1.0. To create a new command line simply click on the menu then click on the desired command.

2. Window Icons

These are the standard "Window's" short cut icons for opening, saving and printing an application.

3. Player Buttons

The "Player Buttons" allow the user to execute the script through the RS-232 communication port by sending each command one line at a time. The buttons can also move the pointer to the top or bottom of the script as well as reset the drive.

4. Axis Name

The “Axis Name” is displayed in the left-hand side of the script window and is set in the “Configuration Window” when a new script is open or by pressing the [F5] key.

5. Script

A “Script” is a program created in APImate 1.0 but has not been uploaded to the drive. Once stored on the drive, the program is referred to as a “Sequence”.

6. Pointer

The “Pointer” indicates which line of the script the cursor is at. When adding a command to a script, the new line will be added to the line above the pointer. When a script is being uploaded to the drive or executed using the “Player Buttons” the pointer indicates which line is currently be uploaded or executed.

7. Page Number

Displays the page numbers the pointer is on in the script. There are 50 lines per page in an APImate 1.0 script.

8. Line Number

Displays the line number of the script page the pointer is located. There are 50 lines per page in an APImate 1.0 script.

9. Drive Type

Displays the drive type selected in the “Configuration Window” when the script was created. The drive type can be changed by pressing the [F5} key to open the configuration window.

10. Communication Port

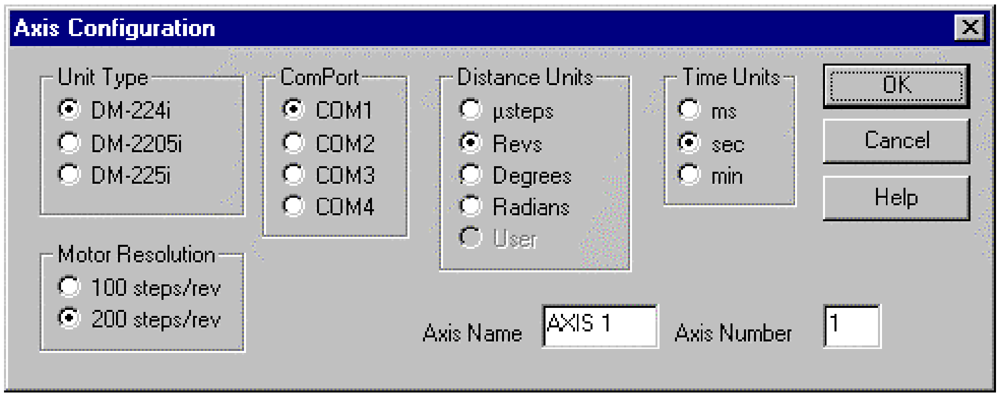
Displays the communication port selected when the script was created. To change the communication port, press [F5] to open the configuration window. The comport number is stored with the script when the application is saved to disk.

2. Drive Set-up & Configuration

This section covers drive software set-up with respect to communication as well as commands related to initializing the drive, such as current setting, power parameters, velocity functions, and Autostart sequence

2.1 Axis Configuration Window

Axis configuration is required to configure the APImate program to communicate with a specific drive. This includes setting the Unit Type, Com Port, Distance Units, Time Units, Axis Name and Axis Number. The axis configuration window will appear each time a new script is opened.



Axis Configuration

Unit Type: [✓ DM-224i] DM-2205i DM-225i

ComPort: [✓ COM1] COM2 COM3 COM4

Distance Units: µsteps [✓ Revs] Degrees Radians User

Time Units: ms [✓ sec] min

Motor Resolution: 100 steps/rev [✓ 200 steps/rev]

Axis Name: AXIS 1

Axis Number: 1

To modify the settings open the Utilities menu and clicking on Axis Configuration

2.1.1 Unit Type

The Unit Type is designated by the Model number of the Intelligent Drive and determines the software commands available for that drive. Each of the Intelligent Drives series has unique features. Setting this variable to the proper Unit Type enables the specific features for that drive.

	NOTE
	A communication error will result if the Unit Type does not match the drive type.

2.1.2 ComPort

This specifies which PC Communication Port the drive is connected to.

Range: COM1 - COM4

	NOTE
	A communication error will result if the wrong PC COM Port is selected. If you are unsure of the COM Port start with COM 1 and work your way up until you find the correct one.

2.1.3 Distance Units

The distance units selected will be used in all commands that reference distance (i.e. Velocity, Move Relative, etc.). Below are the units available.

usteps	Radians
Revs	User Units (Command not yet available)
Degrees	

The units can be changed after commands have been inserted. APImate will recalculate all the values within the sequence to reflect the new distance units.

Example;

Original units are in usteps

AXIS 1: Relative Move **12800** µsteps

After units are changed to Revs

AXIS 1: Relative Move **1.0000** revs

2.1.4 Time Units

The time units selected will be used in all commands that reference time (i.e. Set Velocity Register, Set Acceleration Register, etc.). Shown below are the units supported by APImate 1.0.

ms (milliseconds)
sec (seconds)
min (minutes)

The units can be changed after commands have been inserted. APImate will recalculate all the values within the sequence to reflect the new time units.

Example;

Original units are in sec

AXIS 1: Set Target Velocity Register **1.000** revs/sec

After units are changed to min

AXIS 1: Set Target Velocity Register **60.000** revs/min

2.1.5 Motor Resolution

The Motor Resolution sets the drive to the correct full step resolution for the motor. If the motor is

100 steps/rev (3.6° per motor step)
200 steps/rev (1.8° per motor step)

2.1.6 Axis Name

The Axis Name is the user-defined name for the axis that will appear in the left-hand margin of the script. The axis name is generally a descriptive name such as X Axis, Rotary Axis, Upper Roller, ect. The default value is Axis 1.

Example; If Axis name is changed to “Rotary”, the script will look as follows:

Rotary: Set Peak Current 1.94 Amps

2.1.7 Axis Number

The Axis Number is the number assigned to the drive based on the drive switch settings. This number must match the number specified by the switch setting on the drive. If the software setting and hardware switch setting is mismatched a communication error will result.

	NOTE
	Check the User or Hardware Manual for the correct switch settings for your drive.

2.1.8 Script

A Script is a sequence comprised of a number of motion commands that has not been uploaded to a drive. The Script can be saved on a disk or PC hard drive as back up.

	NOTE
	Once a sequence number has been assigned to a script the script will retain the sequence number every time the script is uploaded until a new number is assigned..

2.2 Autostart Sequence

Range: 1 - 256

The sequence designated as the ***Autostart Sequence*** will begin executing when power is applied or the drive is reset. The sequences is generally used to set motor current, set acceleration rate and velocity, configure inputs and outputs or any other commands to initialize the drive. To initialize the Autostart sequence:

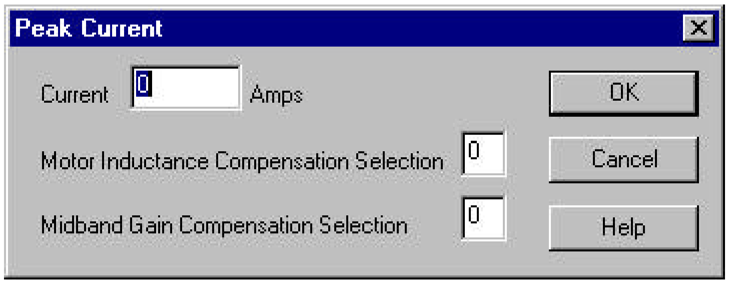
- Upload the desired Autostart sequence to the drive.
- Archive Sequences to the drive
- Create a new sequence making the Autostart command as the only line.
- Using the player buttons, execute the Autostart sequence command.

	NOTE
	The Autostart Command should not be incorporated within a program because each time executed it writes to flash memory. Flash memory can be rewritten about 10,000 times. (Default 256, Autostart disabled) *Autostart sequence remains at last set value until changed. Delete sequences does not effect Autostart value.*

	WARNING
	As a safety precaution move commands should be avoided in the Autostart Sequence. An unexpected move during power-up can result in injury or death.

2.3 Current

The ***Current*** Register sets the maximum current the drive will deliver to the motor. This command is available with drives that offer software selectable current **only**. The current needs to be initialized before any motion command is executed. The default Peak Current is 0 amps.



Current: 0 Amps
Motor Inductance Compensation Selection: 0
Midband Gain Compensation Selection: 0

***** For motors wired in series multiply the unipolar current rating by 0.707
* For motors wired in parallel multiply the unipolar current rating by 1.414

	NOTE
	The peak current register has a resolution of 0.0967742 amps. When entering a value for current, APImate will round off the current to the next lower current resolution. See Installation manual for the correct Peak Current setting for the drive/motor combination.

2.3.1 Motor Inductance

Range: 0 or 1

The **Motor Inductance Compensation** allows the user to tune the individual drive to a specific motor based on the motor winding inductance. The chopping frequency of the current wave form is lowered on motors with low inductance to reduce the amount of noise in the signal. This helps provide the optimum performance for the drive motor combination.

The following is the suggested setting for some standard API motors.

Motor	Motor Resistance (Ohms)	Motor Inductance (mH)	Induction Compensation Setting
M161-02	2.53	1.4	1
M162-03	2.53	1.9	1
M231-08	0.33	0.4	1
M232-09	0.37	0.6	1
M233-09	0.48	0.9	1
M341-06	0.95	3.2	0
M342-09	0.39	1.2	0
MH232-04	2.40	19	0
MT230-04	1.50	7.2	0
MT231-07	1.30	4.4	0
MT232-06	1.50	8.4	0

NOTE

The default Motor Inductance Compensation is 0.

2.3.2 Midband Gain Compensation

Range: 0 - 3

The **Midband Gain Compensation** allows the user to tune the individual drive to a specific motor based on the motor electrical characteristics. The function changes the gain on the midband so that smaller motors can produce a stronger EMF signal when the motor rotates. This provides the optimum performance for the drive motor combination.

The following are suggested setting for some standard API motors.

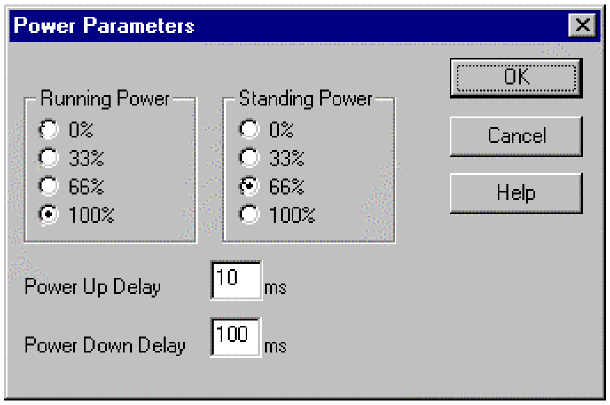
Motor	Motor Resistance (Ohms)	Motor Inductance (mH)	Midband Gain Compensation
M161-02	2.53	1.4	3
M162-03	2.53	1.9	3
M231-08	0.33	0.4	1
M232-09	0.37	0.6	1
M233-09	0.48	0.9	1
M341-06	0.95	3.2	0
M342-09	0.39	1.2	0
MH232-04	2.40	19	0
MT230-04	1.50	7.2	2
MT231-07	1.30	4.4	2
MT232-06	1.50	8.4	2

NOTE

The default Midband Gain Compensation is 0

2.4 Power Parameters

The Power Parameters are used to control the percentage of power delivered to the motor at stand still. Reducing the standing power will reduce motor heating while the motor is idle.


Power Parameters

Running Power: ( ) 0% ( ) 33% ( ) 66% (●) 100%

Standing Power: ( ) 0% ( ) 33% (●) 66% ( ) 100%

Power Up Delay: 10 ms

Power Down Delay: 100 ms

2.4.1 Running Power

The Running Power is the percentage of Peak Current the drive utilizes to execute a Move command. Select between 0%, 33%, 66% or 100% of full power. Default is 100% running power.

	NOTE
	Reducing the running power to the motor reduces motor heating but will also reduce available motor torque. This may be used in applications, which have low velocity or where full torque is not required. If Running Power is set at 0% the motor will not move.

2.4.2 Standing Power

The ***Standing Power*** is the percentage of Peak Current the drive utilizes when idle. The user can chose between 0%, 33% 66% or 100% of full power. Reducing the standing power will save energy and reduce motor heating in application full holding torque is not required. Default is 0% standing power. At 0% standing power the motor will have no holding torque.

	NOTE
	When using reduced standing power the Power Down and Power Up delay should be used to insure that the motor has full torque when it begins to accelerate.

2.4.3 Power Up Delay

Range: 0 - 255 milliseconds

The *Power Up Delay* is the time delay before a move command will begin execution when the drive is in a reduced power mode. This command is used in conjunction with power down delay to reduce motor heating. The delay begins at the time a move command is issued and allows the motor to reach full running power before the motor begins acceleration. Default is 0 ms power up delay.

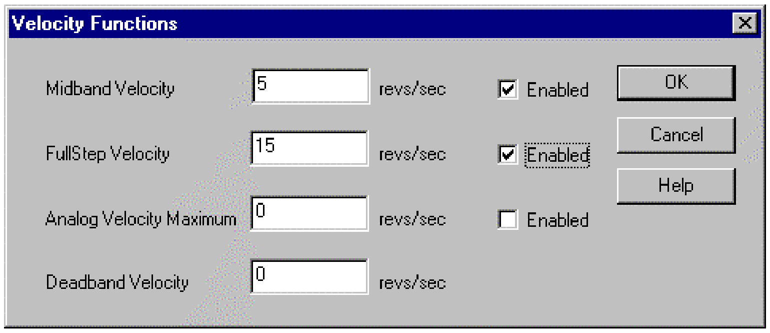
2.4.4 Power Down Delay

Range: 0 - 255 milliseconds

The *Power Down Delay* is the time delay before a drive goes into a reduced power mode after completion of a move. This command is used in conjunction with power up delay to reduce motor heating. The Power Down Delay begins after the motor comes to a complete stop. Default is 0 ms power down delay.

2.5 Velocity Functions

The ***Velocity Functions*** allow the user to enable specific drive functions at a predetermined velocity as well as set-up and enable the Analog velocity following functions. (Drive Option)



Midband Velocity	5	revs/sec	☑ Enabled
FullStep Velocity	15	revs/sec	☑ Enabled
Analog Velocity Maximum	0	revs/sec	☐ Enabled
Deadband Velocity	0	revs/sec	

2.5.1 Midband Velocity

Range: 0 to 50

Step motors systems have resonant frequencies that may cause torque loss (and stalling) at specific frequencies. Midband Stabilization hardware monitors the back EMF to detect when the motor is running at a resonant frequency and adjusts the current vector to compensate. The *Midband Velocity* register sets the motor speed at which the Midband Stabilization is enabled. Speeds below this setting will not utilize Midband Stabilization.

The following are suggested Midband Velocities for standard API motors. The default is Midband Stabilization enabled at 5 RPS.

Motor	**Motor Resistance (Ohm)**	**Motor Inductance (mH)**	**Midband Velocity (RPS)**
M161-02	2.53	1.4	5
M162-03	2.53	1.9	5
M231-08	0.33	0.4	5
M232-09	0.37	0.6	5
M233-09	0.48	0.9	5
M341-06	0.95	3.2	3
M342-09	0.39	1.2	3
MH232-04	2.40	19	5
MT230-04	1.50	7.2	3
MT231-07	1.30	4.4	3
MT232-06	1.50	8.4	3

* When setting the Midband Velocity remember to check the enable box.

2.5.2 Fullstep Velocity

Microstepping offers smoothness and accuracy at low speeds while Full Step offers greater torque at higher speeds. The ***Fullstep Velocity*** registers sets the velocity which the drive switches from a microstep mode to a Full step mode. As the drive decelerates, it will switch back to the microstep mode without loosing it’s position. This combines the benefit of microstep accuracy and the power of full step two phase on.

Microstep resolution = 12,800 steps/rev.
Full Step resolution = 200 steps/rev. (1.8° per step)

The following are ***minimum*** speeds which the Full Step Mode should be activated.

Motor	**Minimum Full Step Speed (RPS)**
M161-02	2.0
M162-03	0.5
M231-08	1.0
M232-09	1.0
M233-09	1.0
M341-06	1.0
M342-09	0.5
MH232-04	1.0
MT230-04	1.0
MT231-07	1.0
MT232-06	1.0

When setting the Full Step Velocity always remember to check the enable box.

	WARNING
⚠	*Setting the Fullstep Velocity too low can cause resonant problems and/or motor stall in some mechanical systems.*

2.5.3 Analog Velocity

Range: 0 to 50 revs/sec

The Analog Velocity registers allows the user to set the maximum motor velocity based on a -10VDC to 10 VDC analog signal. The analog signal is converted to an 8 bit digit from 0 to 256 and is stored in the Analog Register. This provides an approximate voltage resolution of ± 0.079 volts. Since the analog signal is converted to a digital reference the analog input can be used for more than just velocity following (See Installation Manual for proper wiring).

Example 1;

Analog Velocity set at 5 RPS	- +10 VDC signal motor would run 5 RPS CW
Dead Band Window 0.1 rev/sec	- 5 VDC signal motor would run 2.5 RPS CCW.
	- +1 VDC signal motor would run 0.5 RPS CW.

Example 2;

AXIS 1	V1 = Analog Input	*Set custom variable V1= analog register*
AXIS 1	V1 = V1 * **12800**	*Multiple V1 times 12,800 microsteps/rev*
AXIS 1	Move Distance: **V1**	*Move V1*

If Analog Signal is 5 volts the Analog Resister = (256 resolution /20 VDC) * 5 VDC = 64

The motor will move 819,200 microsteps or 64 revs.

	NOTE
☞	*The Deadband Velocity should always be used in conjunction with the Analog Velocity to prevent the motor from wondering at 0 volts*

2.5.4 Deadband Velocity

Range: -50 to 50 revs/sec

The **Deadband Velocity** is used in conjunction with the Analog Velocity command an allows the user to set the deadband wind around 0 RPS which the motor will not move. This function prevents the motor from wondering or chattering from noise on the analog signal.

	NOTE
☞	*The Deadband velocity is dependent on the amount of noise on the analog line. Use shielded cable with twisted pair and never run I/O cable in the same conduit or along side motor and power cables. The analog cable should be kept as short as possible to reduce ground noise*

Example:

Analog Velocity set at 10 RPS

Deadband Velocity set at 0.2 RPS.

The motor will not begin to move until the analog voltage is above 0.2 VDC or below -0.2 VDC

2.6 Reset Command

The ***Reset*** command should not be used within a sequence. This command will "soft power down" the drive. Program execution will stop, all users variables will be cleared. position register will be reset to zero and power will be cut to motor.

	WARNING
⚠	*Resetting the drive will cause the "**Autostart**" program to execute, which can result in unexpected motion. The* **Reset** *command should not be used within a sequence.*

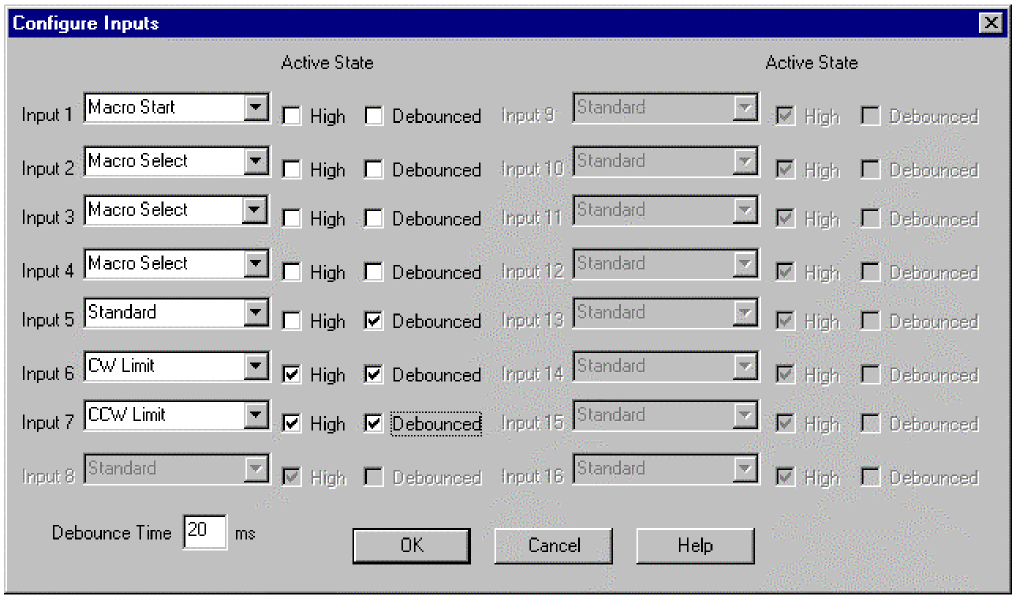
2.7 Input Configuration

The inputs can be configured as Standard, or for specific purposes such as *Clockwise Limit*, *Counter Clockwise Limit*, *Macro Select*, *Macro Start*, *Macro Start/Stop*, *Stop Input* and *Enable*. The polarity of the inputs can be configured active high or active low through software.

The Input configuration is binary and has the following default status if the inputs are not configured:

	Input 1	**Input 2**	**Input 3**	**Input 4**	**Input 5**	**Input 6**	**Input 7**
Type	Sequence Start	Sequence Select	Sequence Select	Sequence Select	Sequence Select	Standard	Standard

- Default polarity is active low. (Inputs pulled to ground)



Configure Inputs

		Active State				Active State	
Input 1	Macro Start	☐ High	☐ Debounced	Input 9	Standard	☑ High	☐ Debounced
Input 2	Macro Select	☐ High	☐ Debounced	Input 10	Standard	☑ High	☐ Debounced
Input 3	Macro Select	☐ High	☐ Debounced	Input 11	Standard	☑ High	☐ Debounced
Input 4	Macro Select	☐ High	☐ Debounced	Input 12	Standard	☑ High	☐ Debounced
Input 5	Standard	☐ High	☑ Debounced	Input 13	Standard	☑ High	☐ Debounced
Input 6	CW Limit	☑ High	☑ Debounced	Input 14	Standard	☑ High	☐ Debounced
Input 7	CCW Limit	☑ High	☑ Debounced	Input 15	Standard	☑ High	☐ Debounced
Input 8	Standard	☑ High	☐ Debounced	Input 16	Standard	☑ High	☐ Debounced

Debounce Time 20 ms

* Default Debounce Time is 0 ms.

Choose Input type

- Designate each Input as Active High (✓) or Active Low (unchecked)
- Set Debounce Time in low left hand corner of window (not shown)
- Designate Debounce time for any input by checking the "Debounced" box.

	NOTE
	Before wiring the inputs, verify if drives has TTL or Optically Isolated I/O. Refer to the ***User Guide (P/N 151-DM-224i)*** *for proper Input wiring.*

2.7.1 Macro Select Input

The ***Macro Select Inputs*** determine the sequence number to execute when the ***Macro Start*** input is triggered. When defining Macro Select inputs you ***must*** begin with Input 2 then 3, etc working your way up. Each additional Input configured as a Sequence Select Input doubles the number of sequences which can be executed through inputs. The number of Macros Select inputs determines the number of sequences, which can be executed. The default is Inputs 2, 3, 4 and 5 are macro select inputs.

Example: The sequence number to be executed when the Macro Start Input is triggered, is based on the binary equivalence plus one of the Macro Select inputs beginning with Input 2.

	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7
Input Definition.	Macro Start	Macro Select	Macro Select	Macro Select	Macro Select	Standard	Standard
Binary Value	N/A	1	2	4	8	N/A	N/A

Example: If Inputs 3 and 5 are active when Input 1 is triggered, Sequence 11 will be executed. (2+8+1)

Sequence Input Chart:

Input 2	Input 3	Input 4	Input 5	Sequence No.
Inactive	Inactive	Inactive	Inactive	1
Active	Inactive	Inactive	Inactive	2
Inactive	Active	Inactive	Inactive	3
Active	Active	Inactive	Inactive	4
Inactive	Inactive	Active	Inactive	5
Active	Inactive	Active	Inactive	6
Inactive	Active	Active	Inactive	7
Active	Active	Active	Inactive	8
Inactive	Inactive	Inactive	Active	9
Active	Inactive	Inactive	Active	10
Inactive	Active	Inactive	Active	11
Active	Active	Inactive	Active	12
Inactive	Inactive	Active	Active	13
Active	Inactive	Active	Active	14
Inactive	Active	Active	Active	15
Active	Active	Active	Active	16

NOTE

*Input 1 is always designated the **Marco Start** input. If an input is defined as a **Sequence Select Input** it can be reconfigured and used as a Standard Input within a sequence..*

2.7.2 CW Limit Input

The ***Clockwise Limit*** can be designated from the Input Configuration screen. When an input is defined as the CW Limit and becomes active during a move in the clockwise direction, the motor will decelerate to a stop and abort the sequence. No further moves in the CW direction can be executed until the input is cleared. A CCW move can be made at anytime by executing another sequence.

	NOTE
	If the CW limit switch is hit while executing a sequence the fault register will display a Hardware Fault condition.

2.7.3 CCW Limit Input

The ***Counter Clockwise Limit*** can be designated from the Input Configuration screen. When an input is defined as the CCW Limit and becomes active during a move in the counter clockwise direction, the motor will decelerate to a stop and abort the sequence. No further moves in the CCW direction can be executed until the input is cleared. A CW move can be made at anytime by executing another sequence.

	NOTE
	If the CCW limit switch is hit while executing a sequence the fault register will display a Hardware Fault condition.

2.7.4 Macro Start Input

The ***Macro Start*** input is available for Input 1 **only** and is used to execute a sequence based on the ***Macro Select*** inputs. When Input 1 is designated as the sequence start input and activated the selected sequence will be executed.

Example: If Inputs 2, 3, 4 and 5 are sequence select inputs and Inputs 2 & 5 are active when Input 1 is activated, Sequence #9 will be executed.

	NOTE
	The sequence execution begins when the input transitions from inactive to active and continues regardless if Input 1 becomes Inactive or Active again.

2.7.5 Macro Start/Stop Input

The ***Macro Start/Stop Input*** is not available at this time.

2.7.6 Stop Input

The ***Stop Input*** is used to stop the execution of a sequence. When the Stop Input becomes active the motor will decelerate to a stop and the program will be aborted. Power will still be supplied to the motor and the motor will have holding torque. The sequence can be restarted after the Stop Input becomes inactive.

	NOTE
	The Stop Input is used to stop the execution of a sequence before completion but the motor continues to have holding torque

	WARNING
	*The Stop Input is not considered an Emergency Stop and **does not** meet OSHA Standards as an Emergency Stop.*

2.7.7 Enable Input

The ***Enable Input*** is used to enable and/or disable the drive. The enable input must be active in order to run a sequence. When the input becomes inactive the sequence is aborted and power to the motor is removed. This is similar to the Stop input, except power is removed from the motor and the motor will not have holding torque.

NOTE

The Enable Input is used to stop the execution of a sequence before completion and cut power to motor

2.7.8 Input Active State

The ***Active State*** can be defined as Active High or Active Low by Checking (✓) or <u>not</u> checking the High box.

Active High - The Input is active when it is not conducting electricity. ***Normally Closed Switch*** (Sitting at 5-24 VDC)

Active Low -The Input is active when it is conducting electricity. ***Normally Open Switch*** (Pulled to ground)

2.7.9 Debounce Time

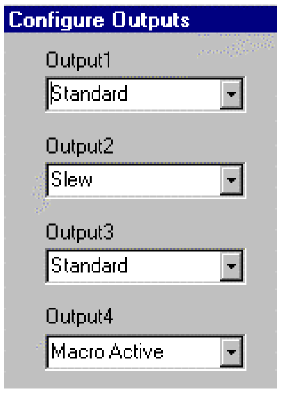
Range: 0 - 255

The Input ***Debounce Time*** when enabled (✓) will delay the response to the input until the input is held for the fixed debounce time. For a solid state relay the Debounce Time can be set low but for a mechanical switch the Debounce Time should be set high to avoid multiple input registrations due to mechanical debounce. The user can designate the Debounce active or inactive for each input.

2.8 Output Configuration

The ***Outputs*** can be configured as Standard, or for specific purposes such as ***Fault***, ***Macro Active, Moving***, or ***Slew***. The drives are available with TTL or Optically Isolated (5 – 24 VDC) Outputs. Check the drive and User Guide for proper output type and wiring.

Select the output type


Configure Outputs

Output1: Standard

Output2: Slew

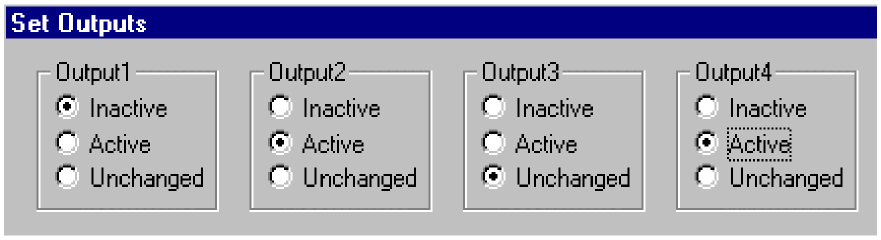
Output3: Standard

Output4: Macro Active

and click OK

2.8.1 Standard Outputs

Standard Outputs are used as general-purpose output and can be set or cleared within a sequence. To set or clear outputs click on ***Commands/Set Register/Outputs*** and choose Active, Inactive or Unchanged.


Set Outputs

Output1: [✓ Inactive] Active Unchanged

Output2: Inactive [✓ Active] Unchanged

Output3: Inactive Active [✓ Unchanged]

Output4: Inactive [✓ Active] Unchanged

2.8.2 Fault Output

The ***Fault Output*** becomes active when a software fault or certain hardware fault conditions occur. If a limit is hit or a Stop Input is active the fault output will be set. If the drive is under voltage or is not powered, the output can not turn on.

	NOTE
	For a drive OK output it is recommended to set a standard output at the beginning of a sequence and never reset the output. If the output goes away the controller will know the drive is no longer functioning

2.8.3 Macro Active Output

The ***Macro Active Output*** is active when the drive is executing a sequence. Once the sequence is complete the ouput becomes inactive.

Example: This output would be used to signal the main PLC that the drive has finished executing the sequence and is ready for the next sequence.

2.8.4 Moving Output

This ***Moving Output*** becomes active when the motor starts to move and becomes inactive once the move is complete and the motor comes to a stop.

Example: The moving output is generally used to signal a main controller that the motor is rotating.

2.8.5 Slew Output

The ***Slew Output*** becomes active when the motor velocity equals the target velocity and will become inactive once the motor begins to decelerate.

3. Programming Commands

This section covers general programming commands including, arithmetic, conditional commands, setting registers and motion commands.

3.1 Setting Registers

3.1.1 Debounce Time

Range: 0 - 255

Using the ***Commands/Set Register/Debounce Time*** command sets the Input Debounce Time. This is the same resister that is set in the Configure Input screen and controls the amount of time an input must be active/inactive before registering the change of state. When using mechanical switches the drive may register multiple contacts from debounce.

3.1.2 Feedback Register

Range: -2,147,483,648 to 2,147,483,647

The ***Feedback Register*** is used to set the counter for the encoder and can be set by using the ***Commands/Set Register/Feedback*** command. In stepper systems encoders are used to close the position loop. An encoder can be used for Position Maintenance and/or Stall Detect. (Encoder Ready is Optional)

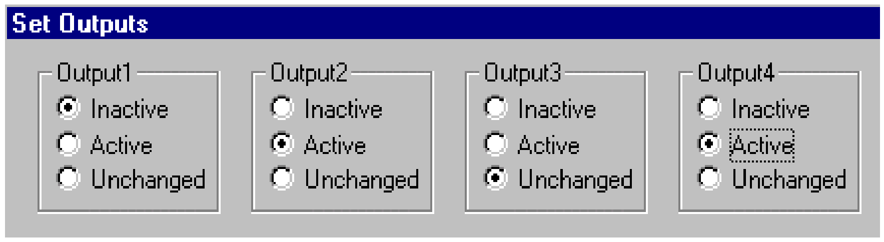
Example; After Homing a system the Position and Feedback Registers need to be reset to zero.

Axis 1 Set Position Register **0.00000** revs
Axis 1 Set Feedback Register **0**

	NOTE
	The Intelligent Step Drives do **not** support Z-Channel Homing, or have a predefined Stall Detect or Position Maintenance routine. Stall Detection and/or Position Maintenance must be written within the sequence.

3.1.3 Output Register

The ***Output Register*** is used to set and/or reset a Standard Output within a sequence. Clicking on ***Commands/Set Register/Outputs*** the output can be set "Active", "Inactive" or "Unchanged". The outputs can be used to light an LED, handshake with a PLC or second drive, close a solid state relay, etc.


Set Outputs

Output1	Output2	Output3	Output4
[✓ Inactive]	Inactive	Inactive	Inactive
Active	[✓ Active]	Active	[✓ Active]
Unchanged	Unchanged	[✓ Unchanged]	Unchanged

NOTE

To see which output is set/reset or to modify the command just double click on the command line in the sequence

3.1.4 Position Register

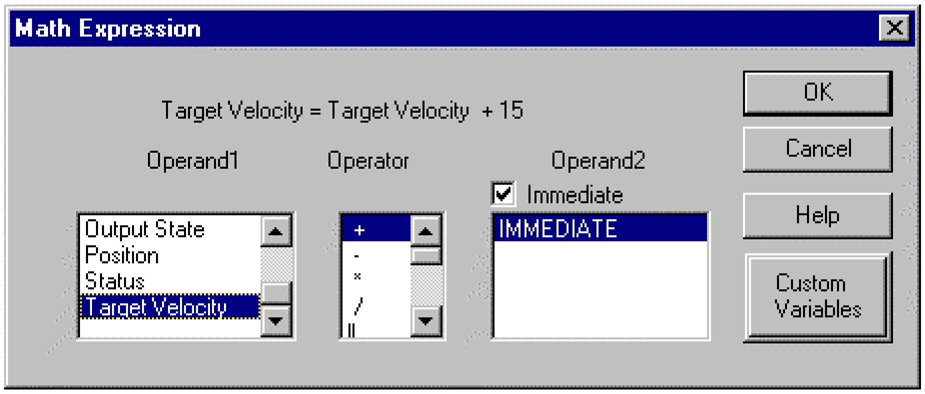
By using the ***Commands/Set Register/Position*** command the ***Position Register*** can be used to set the motor position within the sequence.

Example: After Homing a system the Position and/or Feedback Registers need to be reset to zero.

Axis 1	Set Position Register **0.00000** revs
Axis 1	Set Feedback Register **0**

3.2 Arithmetic Functions

The **Arithmetic Functions** allow the program to perform arithmetic calculations. One arithmetic function is allowed per program line making it necessary to break-up calculations into multiple lines. The calculations can use drive variables, custom variables and/or whole numbers.



The "Math Expression" window allows the programmer to create customer variables or use existing drive variables. To enter a whole number, click on the "Immediate Box" then highlight IMMEDIATE.

3.2.1 Operands

The Operands supported by APImate are divided into two categories. The first category is pre-defined operands system operands (shown below). The second category is Custom Variables which are created by the user.

Fixed Operands:

Acceleration Register	*Output State*	*Analog*
Input	*Position*	*Current*
Velocity	*Status*	*Fault*
	Target Velocity	
Feedback	*Immediate (Operand 2 only)*	*Input*
State		

	NOTE
	*Many of the "Fixed Operands" are read only registers and can **not** be altered by the Arithmetic Commands.*

Acceleration

This Operand is equal to the last acceleration rate set in the drive. The acceleration rate can be changed by the arithmetic commands or used in a comparative statement.

Example:

AXIS 1	Acceleration = **4194300**	*Changes accel rate to 50 rev/sec^2.*
	or	
AXIS 1	If (Acceleration < **70**)	*Compares accel. to a number.*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip*
AXIS 1	}	*commands if **False**.*

	NOTE
	When using a Arithmetic command to set the Acceleration Rate the acceleration must be in ministeps/sec/sec. There are 83886 ministep per revolution of the motor.

Analog

The Analog is read only operand and is equal to the analog register at the time the command is executed. This Operand can be used in a comparative statement or set equal to a User Define Variable using the arithmetic commands.

Example:

AXIS 1	If (Analog < **170**)	*Compares input state to a fixed number.*
AXIS 1	{	
AXIS 1	……	*Will execute if **True** or skip commands*
AXIS 1	}	*if **False**.*
	or	
AXIS 1	V1 = Analog	*Captures the analog input value into V1.*
AXIS 1	V1 = V1 * V1	*Squares the value of the analog input.*
AXIS 1	V1 = V1 * **13**	*Scales the* captured analog input.
AXIS 1	Target Velocity = V1	*Sets the Target Velocity V1*
AXIS 1	Continuous Move	

In the second example the analog input is used to scale the motor velocity based on the square of the analog input (0 –255). The motor velocity becomes a parabolic function to the analog velocity

Current Velocity

This Operand is equal to the velocity of the motor at the time the program line is executed. The current velocity is a **read only** register and can be used in a comparative statement or set equal to a User Defined variable using the arithmetic commands.

Example:

AXIS 1	If (Current Velocity < **10**)	*Compares current velocity to 10*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip*
AXIS 1	}	

or

AXIS 1	V1 = Current Velocity	*Captures Current Velocity.*
AXIS 1	V1 = V1 / **2**	Divides V1 Velocity” by 2.
AXIS 1	Target Velocity = V1	Sets “Target Velocity equal V1
AXIS 1	Continuous Move	Accelerates to new velocity .

Fault

This Operand is equal to the Fault Register at time command is executed. The fault register is a read only register and can be used in a comparative statement.

Example:

AXIS 1	If (Fault = **2**)	*Compares fault register to a fixed number.*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip commands if*
AXIS 1	}	***False**.*

Fault Register = 1	Bus Over Voltage	Fault
Register = 2	Short Circuit	Fault
Register = 4	CCW Limit Reached	Fault
Register = 8	CW Limit Reached	

Feedback

This Operand is equal to the feedback register at the time the command is executed. The feedback position can be changed by the arithmetic command or it can be used in a comparative statement.

Example:

```
AXIS 1    Feedback = 12000       Sets feedback position to 12000 counts.
```

or

```
AXIS 1    If (Feedback < 45000)    Compares feedback
AXIS 1    {
AXIS 1      …..                    Will execute if True or skip
AXIS 1    }                        commands if False.
```

Input State

The Input State is ***read only*** operand and is equal to the input register at the time the command is executed. This Operand can be used in a comparative statement or set equal to a User Define Variables using the arithmetic commands.

Example:

```
AXIS 1    While (Input State && 10)    Mask Inputs 2 and 4.
AXIS 1    {
AXIS 1      …..                        Will execute if True or skip
AXIS 1    }                            commands if False.
```

or

```
AXIS 1    V1 = Input State     Sets V1 equal to Input State.
AXIS 1    V1 = [NEG] V1        Takes the reverses byte value
AXIS 1    V1 = V1 && 10        Masks Inputs 2 and 4
AXIS 1    If (V1 = 2)          If V1=2 (Input 2 pulled low)
AXIS 1    {
AXIS 1      …..                Will execute if True or skip commands If
AXIS 1    }                    False.
AXIS 1    If (V1 = 8)          If V1=8 (Input 4 pulled low)
AXIS 1    {
AXIS 1      …..                Will execute if True or skip commands if
AXIS 1    }                    False.
AXIS 1    If (V1 = 10)         If V1=10 (Input 2 and 4 pulled low)
AXIS 1    {
```

AXIS 1	…..	*Will execute if **True** or skip commands if **False**.*
AXIS 1	}	

Output State

The Output State is read only and equal to the output register at the time the command is executed. This Operand can be used in a comparative statement or set equal to a User Define Variables using the arithmetic commands.

Example:

AXIS 1	If (Output State = **6**)	*If Outputs 2 and 3 are set.*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip*
AXIS 1	}	

Position

This Operand is equal to the position register at the time the command is executed. The position can be changed by the arithmetic commands or used in a comparative statement.

Example:

AXIS 1	Position = **12800**	Sets *Position 12800 microsteps (1 rev.).*

or

AXIS 1	If (Position < **45000**)	*Compares motor position to a 45000.*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip commands If **False**.*
AXIS 1	}	

NOTE

*When using the "**Position**" operand the units are always in microsteps. There are 12800 microstep per revolution of the motor.*

Status

This Operand is equal to the Status Register at time the command is executed. The fault register is a read only register and can be used in a comparative statement.

AXIS 1	If (Status = **2**)	*Compares Status register to a number.*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip commands if **False**.*
AXIS 1	}	

Status Register = 1	Slew Flag (Motor moving at target velocity)	Status
Register = 2	Moving Flag (Motor is moving)	Status
Register = 4	In Delay Flag (Executing a "wait delay" command)	Status
Register = 8	Stopped due to Stop Input.	Status
Register = 10	Disabled due to Enable Input	Status
Register = 20	Sequence Currently in Execution	

Target Velocity

This Operand is equal to the last Target Velocity set in the drive. The target velocity can be changed by the "Set Velocity" command, arithmetic command or used in a comparative statement.

AXIS 1	Target Velocity = **1258290**	Set T*arget velocity equal to 15 rev/sec.*

or

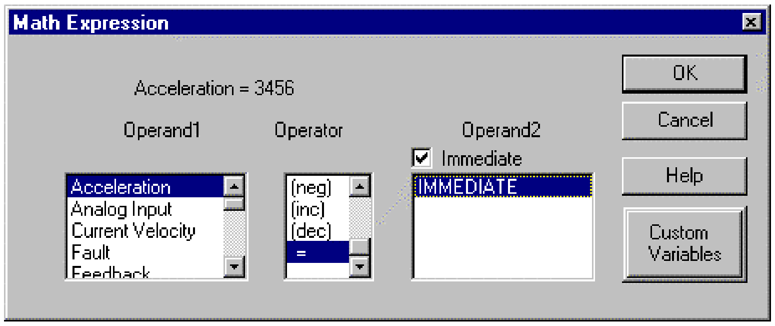
AXIS 1	If (Target Velocity < **838860**)	*If target velocity < 10 revs/sec*
AXIS 1	{	
AXIS 1	…..	*Will execute if **True** or skip commands if **False**.*
AXIS 1	}	

NOTE

*When using the "**Target Velocity**" operand the units are always in ministeps/sec. There are 83886 ministeps per revolution of the motor.*

Immediate

The Immediate function allows a whole number to be used in the expression. This function is available for Operand 2 only and is selected by checking the immediate box above the Operand 2 menu. Then click on Immediate in the menu and enter a value.



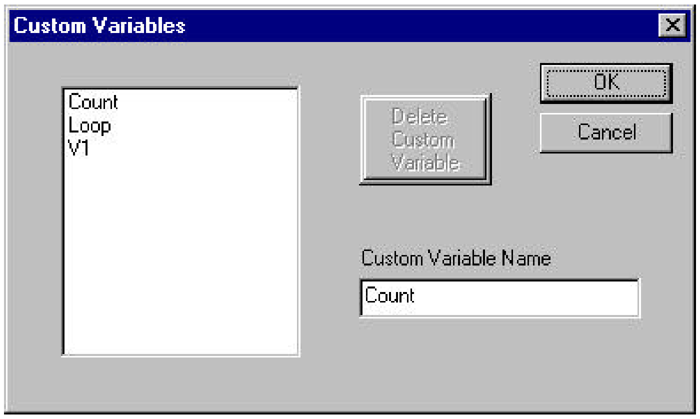
Acceleration = 3456

Immediate [✓]

	NOTE
	User Variables are reset to zero when the drive is powered down or reset.

3.2.2 Custom Variables

A Custom Variable is a user-defined variable and can be used in arithmetic function and/or to capture read only drive variables. The custom variables are created by clicking on the “Custom Variable” key in the “Math Expression Window”. The variable will then appear in the Operand list.



	NOTE
	The Custom Variables are universal throughout the drive. If a variable is defined and given a in value in Sequence 1, it will have the same value in Sequence 8.

	WARNING
	The last value of a Custom Variable is ***not*** *saved in flash memory. The values will be lost at drive fault, loss of power or when the drive is reset.*

Example 1;

AXIS 1	V1 = Analog Input	Captures Analog Input in *variable V1*
AXIS 1	V1 = V1 * 4300	*4300 usteps/inch in mechanical system.*
AXIS 1	Move Distance: V1	*Move V1*

In the above example, ***V1*** *captures the value of the Analog Input and converts it into a user friendly move distance of linear inches.*

Example 2;

AXIS 1	V1 = **5**	*Set custom variable V1 equal to 5*
AXIS 1	V1 = V1 ***** **83886**	*Multiple by 83886 ministeps/rev*
AXIS 1	Target Velocity = V1	*Target Velocity equals 5 revs/sec*.

Example 3;

AXIS 1	V1 = **60**	*Set custom variable V1 equal to 60*
AXIS 1	V1 = V1 * **83886**	*Multiple by 83886 ministeps/rev*
AXIS 1	Acceleration = V1	Acceleration rate equals 60 revs/sec^2.

Example 4;

AXIS 1	Loop = **5**	*Set custom variable Loop = 5.*
AXIS 1	While [Loop > **0**]	*Create a Loop using a "While" command* .
AXIS 1	{	*Commands within the { } will be repeated.*
AXIS 1	…..	*Arbitrary commands.*
AXIS 1	…..	*Arbitrary commands.*
AXIS 1	Loop = [dec]Loop	*The value of Loop is decreased by 1.*
AXIS 1	}	*End of the While Statement.*

In the above example the Sequence will loop through the While Statement (5) times decreasing the value of **Loop** by one each time through. When **Loop = 0** the condition is no longer true and the Sequence executes the next line below the While Statement.

3.2.3 Arithmetic Operators

APImate 1.0 supports the following Arithmetic Operators:

=	Assignment	+	Addition	-
	Subtraction	*	Multiplication	/
	Division	||	Bitwise OR	&&
	Bitwise AND	MOD	Modulo	NEG
	Negate	INC	Increment	DEC
	Decrement	>>	Right Shift	<<
	Left Shift			

The following example will show the effect of the Arithmetic Operator:

Position = 0	*;assigns position register to zero*	*Position*
= Position + 100	*;result is +100 (0 + 100)*	*Position*
= Position - 120	*;result is –20 (100 - 120)*	*Position*
*= Position * 5*	*;result is –100 (-20 * 5)*	*Position*
= Position / 10	*;result is –10 (-100 / 10)*	*Position*
= [neg]Position	*;result is 10 (~ (-10))*	*Position*
= [inc]Position	*;result is 11 (10 + 1)*	*Position*
= [dec]Position	*;result is 10 (11 - 1)*	*Position*
= Position << 3	*;result is 80 (10 * 2^{3})*	*Position*
= Position >> 2	*;result is 20 (80 / 2^{2})*	*Position*
= Position [mod] 3	*;result is 2 (20 / 3 = 6, remainder 2)*	*Position*
= Position || 17	*;result is 19 (00000010 || 00010001)*	*Position*
= Position && 254	*;result is 18 (00010011 && 11111110)*	

3.2.4 Conditional Operators

The ***Conditional Operators*** provide a comparison between two conditions and determines if the statement is **True** or **False**. The following are the conditional operators supported by APImate;

<	- Less Than	>	- Greater Than	<=
	- Less Than or Equal To	>=	- Greater Than or Equal To	!=- Not
Equal To		==	- Equal To	&&
	- Logical AND	||	- Logical OR	

Example; < Less Than

```
AXIS 1    If (Position < 0)        "IF" Conditional statement.
AXIS 1    {
AXIS 1      Set Output Register    Will set Output if True or skip
AXIS 1    }                        commands if False.
```

Example; > Greater Than

```
AXIS 1    If (Position > 0)        "IF" Conditional statement.
AXIS 1    {
AXIS 1      Set Output Register    Will set Output if True or skip
AXIS 1    }                        commands if False.
```

Example; <= Less Than or Equal To

```
AXIS 1    Position = -500000       "While" Conditional
AXIS 1    While (Position <= 0)    "While" Conditional statement.
AXIS 1    {
AXIS 1      Relative Move 12800    Will make the move and wait 1 sec if
AXIS 1      Wait for Motor Stop    True or skip commands if False.
AXIS 1      Wait 1000
AXIS 1    }
```

Example; >= Greater Than or Equal To

```
AXIS 1    Loop = 10                "While" Conditional
AXIS 1    While (Loop >= 0)        "While" Conditional statement.
AXIS 1    {
AXIS 1      Relative Move 12800    Will make the move and wait 1 sec if
AXIS 1      Wait for Motor Stop    True or skip commands if False.
AXIS 1      Wait 1000
AXIS 1    Loop = [DEC] Loop        Subtracts 1 from "Loop"
AXIS 1    }
```

Above example will loop 10 times.

Example; != Not Equal To

```
AXIS 1   Loop = 1                 "While" Conditional
AXIS 1   While (Loop != 0)        "While" Conditional statement.
AXIS 1   {
AXIS 1     Relative Move 12800    Will make the move and wait 1 sec I
AXIS 1     Wait for Motor Stop    True or skip commands if False
AXIS 1     Wait 1000
AXIS 1   }
```

Above example will loop forever.

Example; == Equal To

```
AXIS 1   V1 = Input State         Captures Input Register
AXIS 1   V1 = [NEG] V1            Takes negative bit value if Input State
AXIS 1   V1&& 96                  Masks Inputs 6 & 7
AXIS 1   If (V1 = 64)             "If" Conditional statement.
AXIS 1   {
AXIS 1     Relative Move 12800    Will make the move and wait 1 sec if
AXIS 1     Wait for Motor Stop    Input 7 active or skip commands if
AXIS 1     Wait 1000              Input 7 inactive.
AXIS 1   }
```

Example; && Logical AND

```
AXIS 1   If (Input && 32)         Conditional statement Masking Input 6.
AXIS 1   {
AXIS 1    Relative Move 12800     Will make the move and wait 1 sec if
AXIS 1     Wait for Motor Stop    Input 6 active or skip commands if
AXIS 1     Wait 1000              Input 6 inactive.
AXIS 1   }
```

Example; // Logical or

```
AXIS 1   If (Input // 32)         Conditional Masking Input 6 & 7.
AXIS 1   {
AXIS 1     Relative Move 12800    Will make the move and wait 1 sec if
AXIS 1     Wait for Motor Stop    Input 6 or 7 active or skip commands
AXIS 1     Wait 1000              if Input 6 and 7 are inactive.
AXIS 1   }
```

3.3 Conditional Commands

3.3.1 If Command

An ***"IF"*** command is used to execute a set of commands if the given condition is true. If the condition is false the program will skip over the set of commands contained within the "IF" parentheses.

NOTE

The commands contained inside the brackets will be executed when the condition is true. Conditional Commands can be nestled within each other up to 16 levels deep.

Example 1;

The example shown below will capture the input state as variable V1, reverse define V1 logic and make a move depending on Input 7 being active or inactive.

```
AXIS 1      V1= Input State        Capture input state register in V1.
AXIS 1      V1=[NEG] V1            If V1 was 0100110 it becomes 1011001
AXIS 1       If (V1 && 64)         If Input 7 not active (Input held high).
AXIS 1      {
AXIS 1          Move 1.45 revs     Move 1.45 revs CW.
AXIS 1          Wait for Motor Stop
AXIS 1      }
```

Example 2;

The example shown below will capture the input state as variable V1, reverse define V1 logic, mask Inputs 5, 6 & 7 and make a move based on Inputs 5, 6 & 7 being active or inactive.

```
AXIS 1    V1= Input State          Capture input state register in V1.
AXIS 1    V1 = [NEG] V1            If V1 was 0100110 it becomes 1011001
AXIS 1    V1= V1 && 112            Mask off Inputs 5, 6 & 7 (16 + 32 +64).
AXIS 1    If (V1 = 16)             If Input 5 active (Input held low).
AXIS 1    {
AXIS 1      Relative Move 1.45 revs   Move 1.45 revs CW.
AXIS 1      Wait for Motor Stop
AXIS 1    Wait for Motor Stop
AXIS 1    If (V1 = 32)             If Input  6 active (Input held low).
AXIS 1    {
AXIS 1      Relative Move 2.98 revs   Move 2.98 revs CW.
AXIS 1      Wait for Motor Stop
AXIS 1    }
AXIS 1    If (V1 = 48)             If Input 5 & 6 active (Input held low).
AXIS 1    {
AXIS 1      Relative Move 4.85 revs   Move 4.85 revs CW.
AXIS 1      Wait for Motor Stop
AXIS 1    }
AXIS 1    If (V1 = 64)             If Input 5 & 6 active (Input held low).
AXIS 1    {
AXIS 1       Absolute Move 0.0 revs   Move 2.98 revs CW.
AXIS 1       Wait for Motor Stop
AXIS 1    }
```

3.3.2 While Command

A ***While*** command is used to execute a set of commands over and over while the given condition is true. If the condition is false the set of commands are ignored. If the condition changes to false the set of commands finish executing then are ignored the next time through.

NOTE

The commands contained inside the brackets will be executed when the condition is true. Conditional Commands can be nestled within each other up to 16 levels deep.

Example 1;

The example shown below will Set Output 1 active during the acceleration portion of a Relative move;

```
AXIS 1    Relative Move 100.000 revs
AXIS 1    While (Target Velocity != Current Velocity)
AXIS 1    {
AXIS 1       Set Output Register          Set Output 1 active
AXIS 1    }
AXIS 1    Set Output Register             Reset Output 1 inactive
```

Example 2;

The example below the Sequence will loop 100 times until the Loop Counter = 0. Each time through the motor will move 1 rev. clockwise if the position is less than 10 revs. If the motor position is greater than or equal 10 revs the motor will return to absolute position 0.

```
AXIS 1    Loop Counter = 100          Set  Loop Counter to 100
AXIS 1    While (Loop Counter != 0)    While counter not zero
AXIS 1    {
AXIS 1      If (Position < 128000)    If position less than 10 revs.
AXIS 1      {
AXIS 1        Move 1.00 revs          Move 1 rev CW.
AXIS 1        Wait for Motor Stop
AXIS 1      }
AXIS 1      If (Position => 128000)   If position > or = to 10 revs.
AXIS 1      {
AXIS 1          Absolute  Move  0.00  revs      Move  to  absolute  position  zero.
AXIS 1          Wait for Motor Stop
AXIS 1       }
AXIS 1      Loop Counter = (dec) Loop Counter    Loop Counter (-1).
AXIS 1    }
```

3.4 Motion Commands

3.4.1 Acceleration

Range: 0 to 7812.4 rev/sec/sec (In 0.119 rev/sec/sec increments) *

* Large acceleration rates displayed in microstep/sec/sec may default to a smaller value due to math rollover. If a larger acceleration rate is required change the units to revs/sec/sec.

The ***Acceleration register*** sets the linear acceleration and deceleration ramps. During a relative or absolute move the acceleration and deceleration rates are the same. During a continuous move the accel/decel can be changed while the motor is moving. An acceleration rate ***must*** be defined in the drive before any motion can occur. The acceleration only needs to be set once in the drive but can be changed multiple times within the drive or within a single sequence if required.

Example;

When setting the Acceleration Rate by using the ***Commands/Set Register/Acceleration*** it is set in the time and distance units selected in the Configuration Screen.

Axis 1	Set Velocity Register **2.500** revs/sec
Axis 1	Set Acceleration Register 1 revs/sec/sec
Axis 1	Relative Move **10.25** revs

	NOTE
	To change the accel/decel rate during a continuous move, execute a continuous move command, set the new acceleration rate. The new acceleration rate will be used with the next continuous move command.

3.4.2 Velocity Register

Range: -50.0 to +50.0 revs/sec

The ***Velocity*** register sets the target velocity of the motor within the sequence. A velocity rate ***must*** be defined in the drive before any motion can occur. The Target Velocity only needs to be set once in the drive but can be changed multiple times within the drive or within a single sequence if required.

Example 1: Setting the Target Velocity before a relative move.

Axis 1	Set Acceleration Register **234.123** revs/sec/sec
Axis 1	Set Target Velocity Register **5.1400** revs/sec
Axis 1	Relative Move **10.25** revs
Axis 1	Wait Motor Stop

Example 2: Changing the Velocity and Acceleration on the fly for a continuous move.

Axis 1	Set Acceleration Register **234.123** revs/sec/sec
Axis 1	Set Target Velocity Register **5.1400** revs/sec
Axis 1	Continuous Move
Axis 1	Set Acceleration Register **99.953** revs/sec/sec
Axis 1	Set Target Velocity Register **2.3000** revs/sec
Axis 1	Delay **1200** msec
Axis 1	Continuous Move

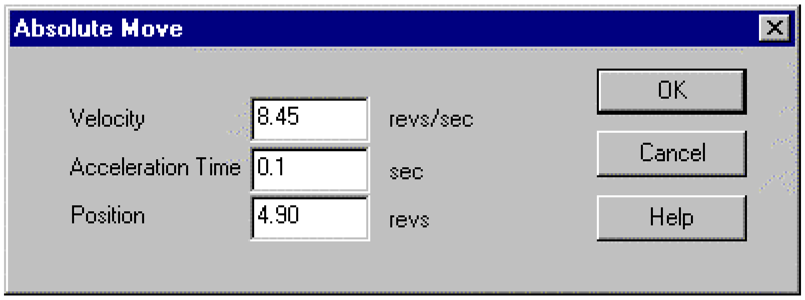
	NOTE
	For a Relative or Absolute Move the target velocity is fixed for the entire move. For a Continuous Move the velocity and/or acceleration rate can be changed on the fly by setting a new velocity/acceleration and issuing another Continuous move command.

3.4.3 Prepare Move Command

The ***Prepare Move*** command is used to calculate the move profile before the move command is executed. To execute the move the "Start Move" command is given. This can reduce command lines and speed up program execution.

Prepare Combo Move Commands

The Prepare Combo Move commands allow the programmer to prepare the complete move profile in one window. The acceleration rate, velocity and/or move distance/position will be entered together. The "Start" move command is used to execute the prepared move.


Absolute Move

Velocity: 8.45 revs/sec
Acceleration Time: 0.1 sec
Position: 4.90 revs

Absolute Move

The prepare Absolute move command sets the absolute position the motor should move to after the "Start Move" command is executed. The direction of rotation is determined by the present motor position with respect to the commanded absolute position.

Axis 1	Set Acceleration Register **50.123** revs/sec/sec
Axis 1	Set Target Velocity Register **5.1400** revs/sec
Axis 1	Prepare Absolute Move **1.0000** revs
Axis 1	Wait for Input(s) Active
Axis 1	Start Move
Axis 1	Wait for Motor Stop

The above example prepares an absolute move to a final position of 1.000 rev moving at 5.14 rev/sec. If the current position is –5.00 revs the motor will rotate 6 revs CW. If the current motor position is 5.00 revs the motor will rotate – 4 revs CCW.

Continuous Move

The prepare Continuous move command pre-calculates the acceleration or deceleration rate to the new continuous move velocity. The sign (+/-) of the velocity command determines the direction of rotation.

```
Axis 1      Set Acceleration Register 25.123 revs/sec/sec
Axis 1      Set Target Velocity Register -5.1400 revs/sec
Axis 1      Continuous Move
Axis 1      Set Target Velocity Register 8.2500 revs/sec
Axis 1      Prepare Continuous Move
Axis 1      Delay 1200 msec
Axis 1      Start Move
```

The example above begins a continuous move at –5.14 revs/sec CCW then prepares the next move velocity. After delaying 1.2 sec the motor will accelerate to 8.2 rev/sec in the CW direction.

Relative Move

The prepare Relative move command pre-calculates the Relative Move profile which will free up processor time when the Start move command is issued. Sign (+/-) of the move distance determines the direction of rotation.

```
Axis 1      Set Acceleration Register 50.123 revs/sec/sec.
Axis 1      Set Target Velocity Register 5.1400 revs/sec
Axis 1      Prepare Relative Move 12.5000 revs.
Axis 1      Loop = 10
Axis 1      While (Loop != 0).
Axis 1      {
Axis 1         Start Move.
Axis 1         Wait for Motor Stop
Axis 1         Delay 800 msec
Axis 1         Loop = [dec] Loop
Axis 1      }
```

The example above the motor will move 12.5 revs at 5.14 revs/sec then delay 0.8 sec. This process will repeat 10 times.

3.4.4 Start Move Command

The Start move command is used in conjunction with the "Prepare Move" command to begin motion. The last prepare move command will be executed.

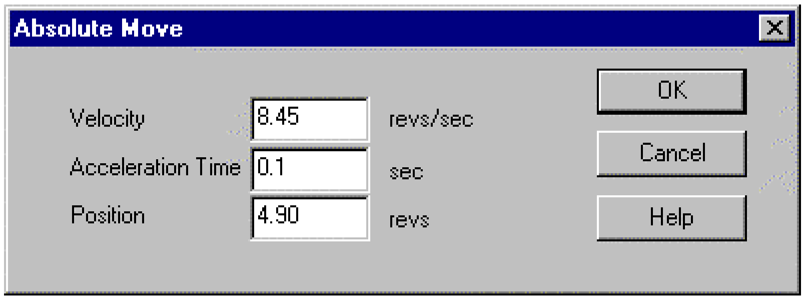
3.4.5 Move Commands

The Move command calculates the move profile and executes the motion in one command step. To add a move command click on ***Commands/ Move*** then select the move type.

	NOTE
	*A Velocity and Acceleration rate must be defined prior to executing a **Move** command.*

Combo Move Commands

The Combo Move commands allow the programmer to prepare the complete move profile in one window. The acceleration rate, velocity and/or move distance/position will be entered together


Absolute Move

Velocity 8.45 revs/sec
Acceleration Time 0.1 sec
Position 4.90 revs

Absolute Move Commands

The *Absolute Move* Command rotates the motor to an absolute position. The direction of the move is determined by the current position relative to the Absolute Move position.

Range: -167772 revs to 167772 revs

	NOTE
	A Wait for Motor Stop command should be issued after the Absolute Move command to prevent another move command from executing while the motor is moving.

Example;

An Absolute Move to a position of 3.0000 Revs will be CW if the current motor position is 1.20000 Revs but will be CCW if the current motor position is 8.20000 Revs.

Axis 1	Set Acceleration Register **30.123** revs/sec/sec
Axis 1	Set Target Velocity Register **7.5000** revs/sec
Axis 1	Absolute Move **3.0000** revs
Axis 1	Wait for Motor Stop

Relative Move Command

The *Relative Move* Command rotates the motor based on a fixed move distance. The direction of rotation is determined by the sign (+/-) of the move distance.

Range: -167772 revs to 167772 revs

	NOTE
	A Wait for Motor Stop command should be issued after the Absolute Move command to prevent another move command from executing while the motor is moving.

Example;

An Relative Move of 13.0000 revs will be CW while a move of –13.0000 revs will be in the CCW direction.

Axis 1	Set Acceleration Register **30.123** revs/sec/sec.
Axis 1	Set Target Velocity Register **7.5000** revs/sec.
Axis 1	Relative Move **13.0000** revs.
Axis 1	Wait for Motor Stop
Axis 1	Delay **2400** msec
Axis 1	Relative Move **-13.0000** revs
Axis 1	Wait for Motor Stop

Continuous Move Command

The *Continuous Move* Command rotates the motor at a constant velocity until a Stop Motor command is executed or a new target velocity is given followed by another Continuous move command. The direction of the rotation is determined by the sign (+/-) of the velocity.

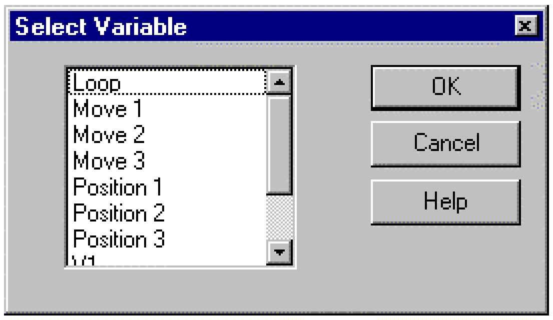
	NOTE
	A Wait for Motor Stop command should be issued after the Absolute Move command to prevent another move command from executing while the motor is moving.

Example;

Axis 1	Set Acceleration Register **25.123** revs/sec/sec
Axis 1	Set Target Velocity Register **-5.1400** revs/sec
Axis 1	Continuous Move

Distance Variable Move Command

The *Distant Variable Move* command allows the user to make a Relative move based on a User Variable. The move distance in microsteps is equal to the value of the user variable at time of execution. The user variable is selected from a variable list.



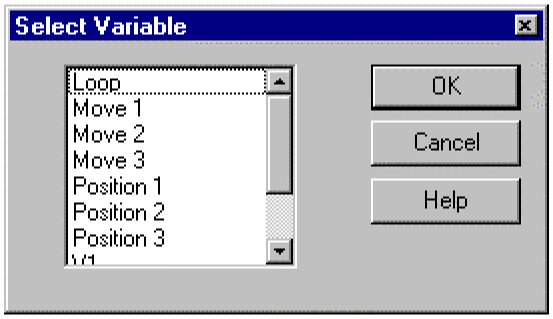
	NOTE
	A Wait for Motor Stop command should be issued after the Absolute Move command to prevent another move command from executing while the motor is moving.

Example;

Axis 1	Set Acceleration Register **30.123** revs/sec/sec.
Axis 1	Set Target Velocity Register **7.5000** revs/sec.
Axis 1	Move Distance **Move 1**.
Axis 1	Wait for Motor Stop
Axis 1	Delay **2400** msec
Axis 1	Move Distance **Move 3**.
Axis 1	Wait for Motor Stop

Position Variable Move Command

The *Position Variable Move* command allows the user to make an Absolute move to a position move on a User Variable. The position in microsteps is equal to the value of the user variable at time of execution. The user variable is selected from a variable list.



	NOTE
☞	*A Wait for Motor Stop command should be issued after the Absolute Move command to prevent another move command from executing while the motor is moving.*

Example;

Axis 1	Set Acceleration Register **30.123** revs/sec/sec.
Axis 1	Set Target Velocity Register **7.5000** revs/sec.
Axis 1	Move Absolute Position **Position 1**.
Axis 1	Wait for Motor Stop
Axis 1	Delay **2400** msec
Axis 1	Move Absolute Position **Position 2**.
Axis 1	Wait for Motor Stop

3.4.6 Stop Move

The ***Stop Move*** Command can be used to stop a Continuous Move or terminate a Relative Move and Absolute Move. The user will be required to choose between a Decelerated stop or No Deceleration.

Decel	- The motor follows a controlled decelerate using the same linear ramping profile as the acceleration ramp.
No Decel	- The power to the motor is cut causing it to coast to a stop. This may result in a longer deceleration time and will cause loss of motor position.

	NOTE
☞	*A Wait for Motor Stop command should be issued after the Absolute Move command to prevent another move command from executing while the motor is moving.*

	WARNING
⚠	*A controlled deceleration from a high speed is usually faster and safer then a hard deceleration where power is cut from the motor.*

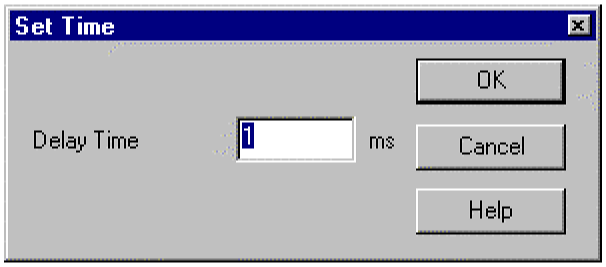
3.5 Wait Commands

The "Wait" commands delay the execution of the sequence for a specified time delay or the completion of an event. Without wait commands the sequence would continue execute one command after another. This could cause a move command to execute before another move command is complete.

3.5.1 Delay Command

The ***Delay*** command delays the execution of a sequence for a fixed amount of time.

Range: 1 to 65535 ms (0.001 to 65.535 sec)



	NOTE
	If a longer delay is required, string multiple delays together, or "loop" a single delay command..

Example 1;

AXIS 1	Delay **60000** milliseconds	*Time delay of 60 sec*
AXIS 1	Delay **60000** milliseconds	*Time delay of 60 sec*
AXIS 1	Delay **60000** milliseconds	*Time delay of 60 sec*

The combination of the above delay would be 3 minutes.

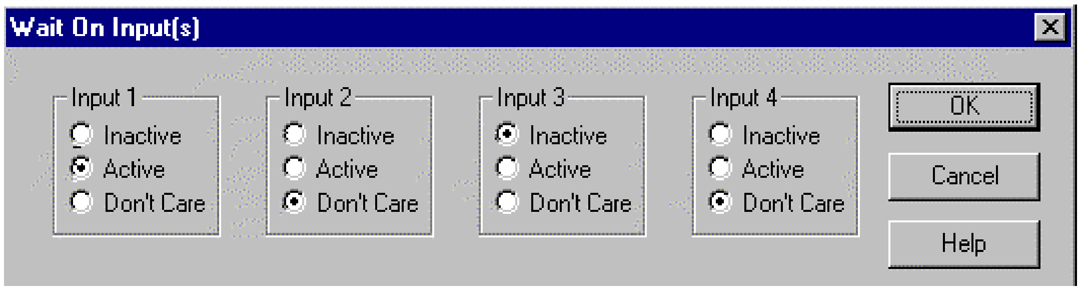
3.5.2 Wait Motor Stop Command

The ***Wait on Motor Stop*** Command delays the execution of the next command until the motor comes to a complete stop. This allows the motor to finish executing a move command before the next command begins execution.

	NOTE
	A Wait for motor stop is recommended following each Absolute Move, Relative Move or after a Stop command to prevent the drive from executing another move command while the motor is moving.

3.5.3 Wait on Input(s)

The ***Wait on Inputs*** Command allows a sequence to wait for a single input or a combination of inputs before executing the next command. The user must define which input(s) to wait for, and the status of the input(s).


Wait On Input(s)

Input 1: Inactive [✓ Active] Don't Care
Input 2: Inactive Active [✓ Don't Care]
Input 3: [✓ Inactive] Active Don't Care
Input 4: Inactive Active [✓ Don't Care]

Inactive - When the input is at it's "Inactive" state, the Wait on Input condition has been met and the sequence will continue execution at the next command.

Active - When the input is at it's "Active" state, the Wait on Input condition has been met and the sequence will continue execution at the next command.

Don't Care - The input is not being used as part Wait on Input command and it's status is ignored.

	NOTE
	To determine when an input is active "High" (non-conducting) or active "Low" (conducting), refer back to the Input Configuration window.

3.6 Creating a Simple Program

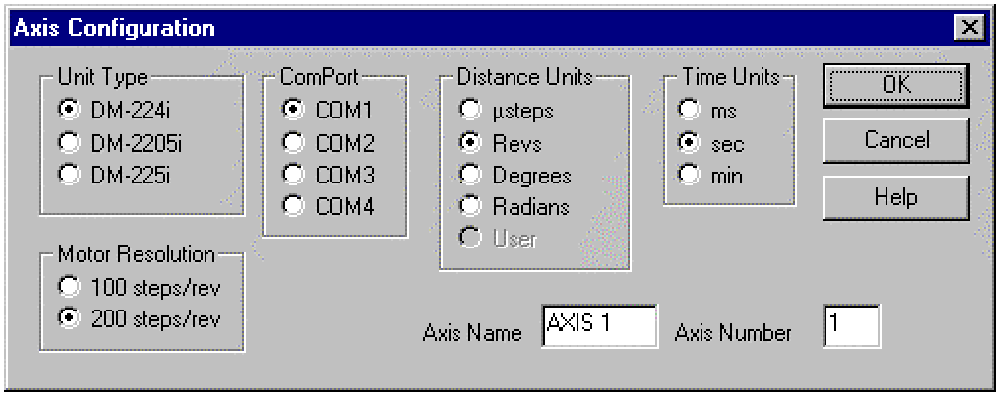
Follow the step by step procedures below to create a simple "Loop" sequence.

STEP 1

Read the Installation Manual making certain drive is wired correctly and dip switches are properly set. Verify that the power light is illuminated when power is turned on.

STEP 2

Refer to the installation manual to insure the axis designation coincides with the switch settings. With the File/New command, a window will open which allows you to define the hardware configuration and some of the parameter such as time and distance units.



Unit Type: [✓ DM-224i] DM-2205i DM-225i
ComPort: [✓ COM1] COM2 COM3 COM4
Distance Units: µsteps [✓ Revs] Degrees Radians User
Time Units: ms [✓ sec] min
Motor Resolution: 100 steps/rev [✓ 200 steps/rev]
Axis Name: AXIS 1
Axis Number: 1

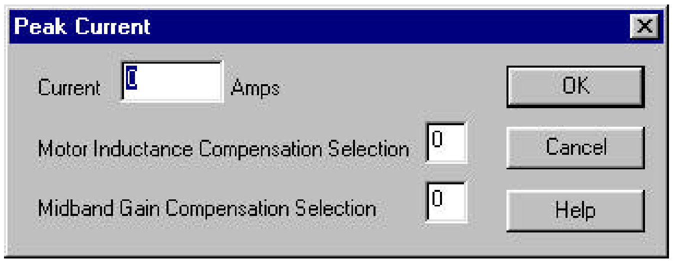
The "Axis Name" entered in this dialog box will appear in the margin for each program line of the script. APImate is designed for programming a single axis. When several axes are connected to the same PC, you can only write motion software for one driver at a time, although several can be connected on the same RS-232 link.

STEP 3

Make sure that the axis number entered corresponds to the DIP switch setting inside the driver (For more information about the DIP setting refer to the Hardware Manual). Verify that the PC is communicating to the drive buy clicking on Utilities, Get Register, Version Resister. The Version Register window should appear. If a communication error occurs see Trouble Shooting

STEP 4

If the drive type requires the motor current to be set in software click on Utilities, Initialize then Current. Enter the correct current for the motor. If the drive uses dip switches to set the current, this step can be skipped.



Current 0 Amps

Motor Inductance Compensation Selection 0

Midband Gain Compensation Selection 0

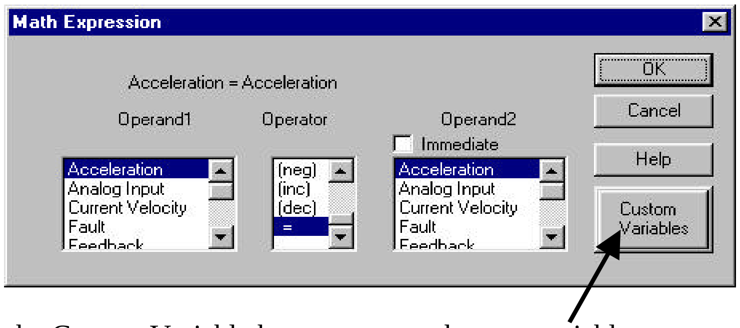
To set the motor current when using the DM-224I drive, click on Utilities/ Initialization /Current.

The Peak Current is dependent upon motor winding, motor wiring (series or parallel) and application. To insure proper setting for API Control standard motors, refer to the Hardware Manual. Setting the current is required before any motion command can be executed.

A table listing Motor inductance and Midband gain compensation for standard API Control motors can be found in the on-line help under the topics Power - Midband Gain and Motor Inductance or the Hardware Manual.

STEP 5

Create a custom variable called loop by click on Commands/Arithmetic, which will open the Arithmetic window.



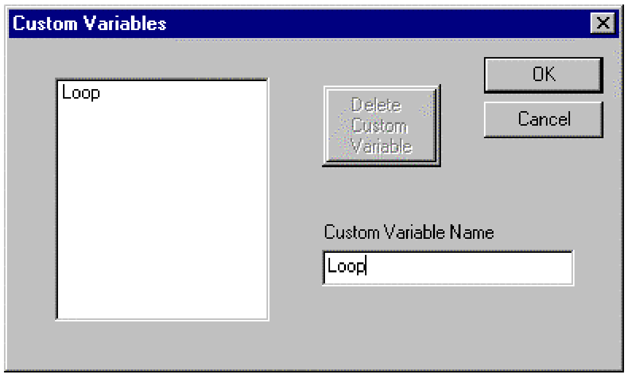
Acceleration = Acceleration

Operand1: Acceleration, Analog Input, Current Velocity, Fault, Feedback

Operand2: Acceleration, Analog Input, Current Velocity, Fault, Feedback

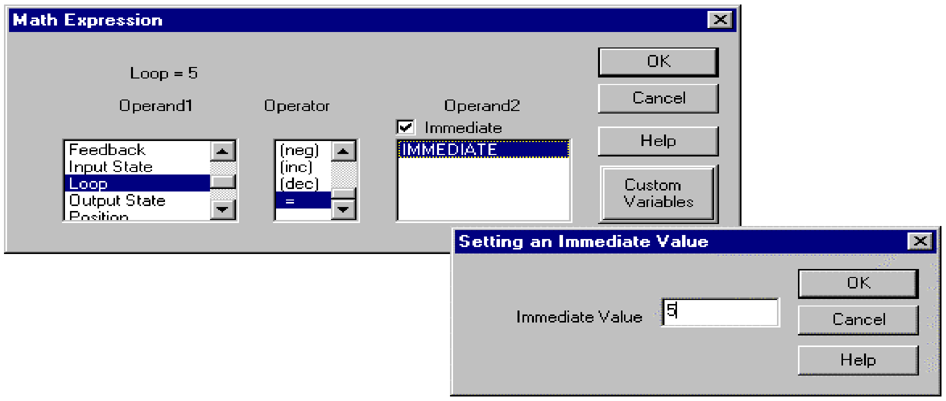
Click on the Custom Variable button to enter the new variable name.

Enter the new variable name and click on OK. The new variable will appear in the Operand window of the Arithmetic Screen



STEP 6

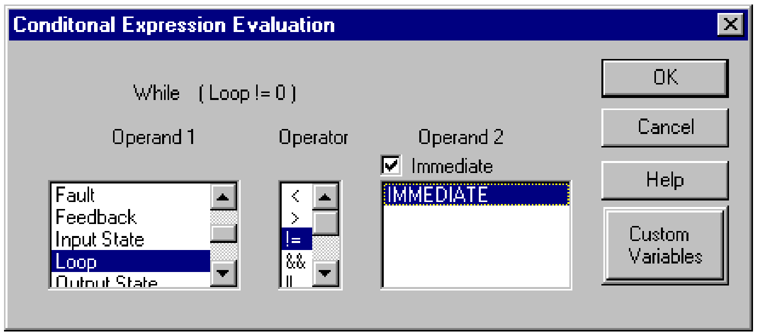
Create the math expression Loop=5 by clicking on Commands /Arithmetic.



Note: To enter a number click on the "Immediate box" under Operand 2 then highlight the "Immediate" which appears under Operand 2.

STEP 7

Click on "Commands/While" to create a While {Loop (!=) not equal to 0} command.



Conditonal Expression Evaluation

While (Loop != 0)

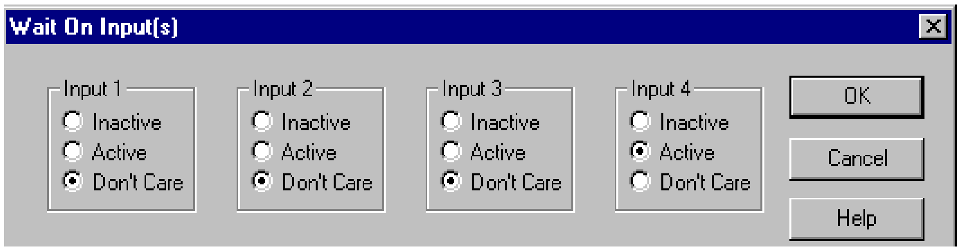
Operand 1: Fault, Feedback, Input State, [✓ Loop], Output State

Operator: <, >, [✓ !=], &&, ||

Operand 2: [✓ Immediate] IMMEDIATE

STEP 8

Line the pointer up with the lower parenthesis and click on Commands/Wait/Input(s) and click on Input 4 active.



Wait On Input(s)

Input 1: Inactive, Active, [✓ Don't Care]

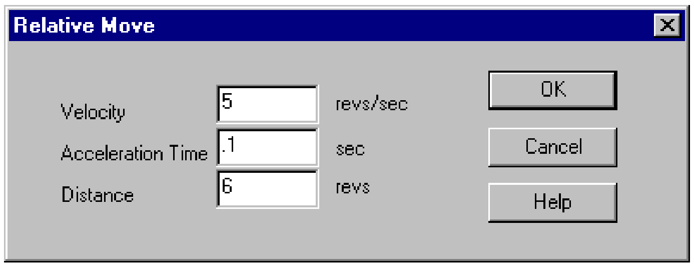
Input 2: Inactive, Active, [✓ Don't Care]

Input 3: Inactive, Active, [✓ Don't Care]

Input 4: Inactive, [✓ Active], Don't Care

STEP 9

Create a Relative Move with an Acceleration and Velocity by clicking on Combo/Move/Relative.



Relative Move

Velocity	5	revs/sec
Acceleration Time	.1	sec
Distance	6	revs

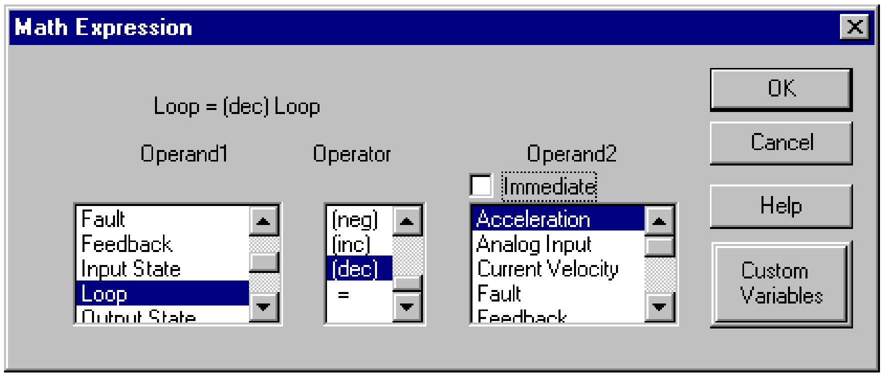
Note: To make a move the acceleration and velocity must be defined. This can be done in one window using the Combo Move command. Once an Acceleration and Velocity are defined, they do not need to be redefined unless to change them.

STEP 10

Click on Commands/Wait/Motor Stop to pause the execution of the program until the move is completed. This prevents the drive from trying to execute a second move while the motor is moving.

STEP 11

Now create an Arithmetic command to decrement the loop counter.



Loop = (dec) Loop

STEP 12

To run a saved sequence click on Utilities, Sequence and Execute Sequence or activate the sequence inputs.

The Sequence will wait for input 4 before making a fixed relative move. This process will repeat a total of 5 times before ending the program.

Note: The sequence will be executed in real time through the drive.

Additional Programming Tips

To delete commands simply point the pointer on the desired line and hit the delete button on the keyboard.

To add lines simply place the pointer on the desired line, choose the command you wish to add and it will appear on the line above the pointer.

To run the script at any time use the player buttons on the tool bar. The player buttons allow the user to run through an entire script or through one command at a time.

4. Executing Sequences

This section covers the several techniques available to execute a sequence.

4.1 Sequence Commands

4.1.1 Abort Sequences

The ***Abort Sequence*** commands allow stops the execution of a sequence. Unlike the "Reset" command the abort sequence does ***not*** soft power-up the drive erasing all unachieved sequences and/or start the "Autostart Program".

4.1.2 Archive Sequences

The ***Archive Sequence*** command saves all the sequences loaded in RAM memory (volatile) to Flash Memory (Non-volatile). When a drive is powered up or reset the sequences stored in Flash Memory are loaded to RAM memory. The programs are then executed through RAM memory. If the drive is not archived all programs loaded in RAM are erased whenever the drive is reset or powered down.

4.1.3 Delete Sequences

Range: 1 to 255

The ***Delete Sequence*** command allows a user to delete a single sequence from RAM memory. To delete the sequence from Flash Memory the "Archive Sequence" command must be used.

	NOTE
☞	*After deleting a sequence, the Archive Sequence command must be executed in order to remove the sequence from Flash Memory. If the sequence is not deleted from Flash Memory it will be reloaded from Flash to RAM when the drive is reset or powered down..*

4.1.4 Delete All Sequences

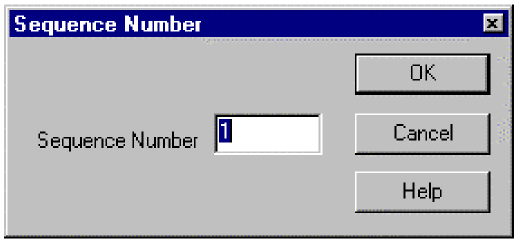
The ***Delete All Sequence*** command allows a user to delete all sequences from the RAM memory. To delete all the sequence from Flash Memory the "Archive Sequence" command must be used.

	NOTE
	After deleting all sequences, the Archive Sequence command must be executed in order to remove the sequence from Flash Memory. If the sequences are not deleted from Flash Memory they will be reloaded from Flash to RAM when the drive is reset or powered down.

4.1.5 Execute Sequences

Range: 1 to 255

The ***Execute Sequence*** command executes the specified sequence through the drive. The sequence must be uploaded into the drive and will be executed in real time.



Sequence Number: 1

To execute a sequence open the Utilities and Sequence menus then click on Execute Sequence. Then enter the sequence number to be executed.

4.1.6 Upload to Drive

Range: 1 to 255

The ***Upload Sequence*** command is used to transfer a single script from the PC to the RAM memory of the drive. Once loaded the sequence can be executed in real time by using the "*Sequence Select Inputs*"or "*Execute Sequence*" command.

With Execution

The sequence is executed through the PC as the script is uploaded. It is similar to executing the sequence using the player keys. Because of possible communication errors caused by an infinite loop or other commands, it is recommended to upload the drive without execution.

Without Execution

The sequence is ***not*** executed while transferring the program from the PC to the drive. (Recommended method)

	NOTE
	After uploading a sequence, the Archive Sequence command must be executed in order to save the sequence to Flash Memory. If the sequence is not saved in Flash Memory it will be erased when the drive is reset or powered down.

4.1.7 Download from Drive

The ***Download from Drive*** command allows the user to transfer a single sequence from the drive RAM memory to the PC. The sequence will appear in the APImate window as a script file and can be saved on a disk or the hard drive.

Range: 1 to 255

	NOTE
	When downloading sequences from the drive using Comport 1 may result in a communication error. When ever possible use Comport 2, 3 or 4 when downloading sequences from the drive to the PC.

4.1.8 Upload All to Drive

The ***Upload All Sequences*** command is used to transfer multiply scripts from the PC to the drive. All open scripts on APImate are loaded onto the drive with the user entering or confirming each sequence number before the script uploads. A ***Sequence Number*** is requested when a script is uploaded from the PC to the drive which determines the memory location of the sequence Once loaded on the drive any of the sequences can be executed by using the "*Sequence Select Inputs*"or "*Execute Sequence*" command.

Range: 1 to 255

	NOTE
	After uploading a sequence, the Archive Sequence command must be executed in order to save the sequence to Flash Memory. If the sequence is not saved in Flash Memory it will be erased when the drive is reset or powered down

4.1.9 Download All From Drive

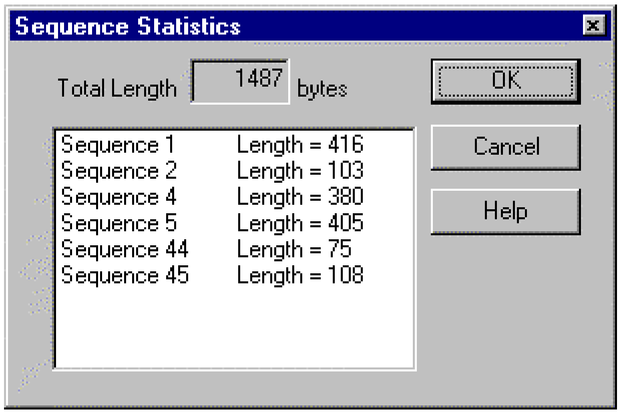
The ***Download All Sequences*** command allows the user to transfer all sequences from the drive RAM memory to the PC. The sequences will appear in the APImate window as a script files and can be saved on a disk or the hard drive.

	NOTE
	When downloading sequences from the drive using Comport 1 may result in a communication error. When ever possible use Comport 2, 3 or 4 when downloading sequences from the drive to the PC.

4.1.10 Sequence Directory

The ***Sequence Directory*** lists the Sequence Number and Sequence Length in bytes of all sequences loaded in the RAM memory of the drive.

Range: 1 - 255


Sequence Statistics

Total Length 1487 bytes

```
Sequence 1     Length = 416
Sequence 2     Length = 103
Sequence 4     Length = 380
Sequence 5     Length = 405
Sequence 44    Length = 75
Sequence 45    Length = 108
```

4.2 Player Buttons

The ***Player Buttons*** located on the toolbar allow the user to single step or run a complete sequence. The commands are sent through the PC serial communication port to the drive. The speed of execution is limited by the speed of the PC and will be slower and less consistent then executing a saved sequence in the drive. The player buttons are a good programming debug tool, which allows the user to observe the commands as they are executing.

⏮ Moves the pointer to the top of the script.

Moves the pointer to the previous line.

Will stop execution of a sequence and Reset the drive. (Is the same as cycling power to the drive.) Any Autostart sequence will be executed and any sequences download without archiving will be erased.

Allows the user to single step through a script. The command in which the pointer is on will be executed.

Executes an entire script on screen. The pointer will move down the screen executing each command until the end of the sequence or a break point is reached.

Moves the pointer to the bottom of the script.

Inserts a breakpoint in the script. When used with the player key will execute a program up to the breakpoint. A red dot will appear before a command with a breakpoint.

Removes all breakpoints from a script.

WARNING

Since the player keys execute the sequence through the serial port, communication errors can result with some sequences. The PC is continuously requesting status from the drive and when the processor is busy (like during a long move) the PC will interpret the non-response as a communication error.

* If a communication error is given while using the “Player Buttons”, Download the sequence to the drive and use the “Execute Sequence” command.

4.3 Using Inputs to Execute Sequences.

4.3.1 Macro Start Input

The ***Macro Start*** input is available for Input 1 **only** and is used to execute a sequence based on the ***Macro Select*** inputs. When Input 1 is designated as the sequence start input and activated the selected sequence will be executed.

Example: If Inputs 2, 3, 4 and 5 are sequence select inputs and Inputs 2 & 5 are active when Input 1 is activated, Sequence #9 will be executed.

	NOTE
	The sequence execution begins when the input transitions from inactive to active and continues regardless if Input 1 becomes Inactive or Active again.

4.3.2 Macro Select Input

The ***Macro Select Inputs*** determine the sequence number to execute when the ***Macro Start*** input is triggered. When defining Macro Select inputs you ***must*** begin with Input 2 then 3, etc working your way up. Each additional Input configured as a Sequence Select Input doubles the number of sequences which can be executed through inputs. The number of Macros Select inputs determines the number of sequences, which can be executed. The default is Inputs 2, 3, 4 and 5 are macro select inputs.

Example: The sequence number to be executed when the Macro Start Input is triggered, is based on the binary equivalence plus one of the Macro Select inputs beginning with Input 2.

	Input 1	**Input 2**	**Input 3**	**Input 4**	**Input 5**	**Input 6**	**Input 7**
Input Definition.	Macro Start	Macro Select	Macro Select	Macro Select	Macro Select	Standard	Standard
Binary Value	N/A	1	2	4	8	N/A	N/A

Example: If Inputs 3 and 5 are active when Input 1 is triggered, Sequence 11 will be executed. (2+8+1)

Sequence Input Chart:

Input 2	Input 3	Input 4	Input 5	Sequence No.
Inactive	Inactive	Inactive	Inactive	1
Active	Inactive	Inactive	Inactive	2
Inactive	Active	Inactive	Inactive	3
Active	Active	Inactive	Inactive	4
Inactive	Inactive	Active	Inactive	5
Active	Inactive	Active	Inactive	6
Inactive	Active	Active	Inactive	7
Active	Active	Active	Inactive	8
Inactive	Inactive	Inactive	Active	9
Active	Inactive	Inactive	Active	10
Inactive	Active	Inactive	Active	11
Active	Active	Inactive	Active	12
Inactive	Inactive	Active	Active	13
Active	Inactive	Active	Active	14
Inactive	Active	Active	Active	15
Active	Active	Active	Active	16

NOTE

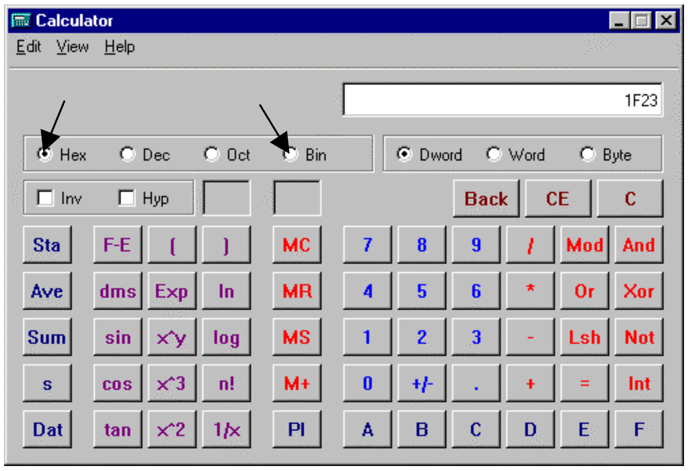
Input 1 is always designated the ***Marco Start*** *input. If an input is defined as a* ***Sequence Select Input*** *it can be reconfigured and used as a Standard Input within a sequence..*

5. Getting Registers

This section shows how to retrieve drive information such as position, I/O status, drive status and motion/diagnostic registers.

Many of the registers are represented in a hexadecimal format for efficiency. The easiest way to convert from hexadecimal to bit or decimal format is by using the scientific calculator supplied in Windows/Accessories.

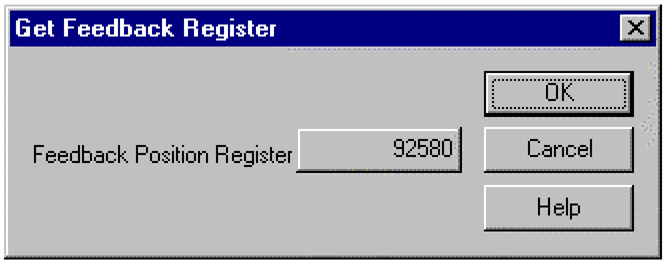
Example: Click on number format (*Hex, Dec or Bin*) then enter value. To change to another format just click on the new format type and the number will be converted.


Calculator
Edit View Help
1F23
Hex Dec Oct Bin
Dword Word Byte
Inv Hyp
Back CE C
Sta F-E ( ) MC 7 8 9 / Mod And
Ave dms Exp ln MR 4 5 6 * Or Xor
Sum sin x^y log MS 1 2 3 - Lsh Not
s cos x^3 n! M+ 0 +/- . + = Int
Dat tan x^2 1/x PI A B C D E F

- Assume the hexadecimal value of the Status Register = 1F23
- Converted to the binary equivalent = 1111100100011 (Read from right to left)
- Converted to decimal = 7971

5.1 Feedback Register

To verify the ***Feedback Position*** click on ***Utilities/Get Registers/Feedback Registers***. A window will appear showing the instantaneous value of the Feedback Register in pulses per revolution.


Get Feedback Register

Feedback Position Register 92580

Example: If the motor utilizes a 1000 line encoder there are 4000 pulses per revolution of the motor. If the Feedback Register was 92580 counts the corresponding motor position is 23.145 revs.

5.2 General Register

To check the General Register click on ***Utilities/Get Registers/General Registers***. A window showing the instantaneous value of the Fault, Drive Status, Autostart and Debounce Time.

Fault Register:

Represents the fault status of the drive in hexadecimal format. This register is generally used when talking to the drive via RS-232/485 and the Link Level Protocol.

Bit 3	Bit 2	Bit 1	Bit 0
CW Limit Active	CCW Limit Active	Short Circuit	Buss Over Voltage

Example:

No Faults	-	0x00	***Short & CCW***	-	0x06	***CCW***
& CW	-	0x0C	***Buss Fault***	-	0x01	***Buss,***
Short & CCW	-	0x07	***Buss, CCW & CW***	-	0x0D	
Short Circuit	-	0x02	***CW Limit***	-	0x08	***Short,***
CCW & CW	-	0x0E	***Buss & Short***	-	0x03	***Buss***
& CW	-	0x09	***Buss, Short, CCW & CW***	-	0x0F	***CCW***
Limit	-	0x04	***Short & CW***	-	0x0A	***Buss***
& CCW	-	0x05	***Buss, Short & CW***	-	0x0B	

Status Register:

Represents the Drive Status in hexadecimal format. This register is generally used when talking to the drive via RS-232/485 and the Link Level Protocol.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Sequence in Progress	Disable Input	Stop Input	In Delay	Moving	At Slew

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reserved	Reserved	Reserved	Axis ID0	Axis ID1	Axis ID2	Axis ID3	Axis ID4

Example: Assume the hexadecimal value of the Status Register = 1F23

- Converted to the binary equivalent = 1111100100011

Counting from Right to Left:

Bit 0 Active	- Motor at Slew
Bit 1 Active	- Motor Moving
Bit 5 Active	- Sequence in Progress
Bits 8-12 Active	- Axis ID 1

Autostart Register:

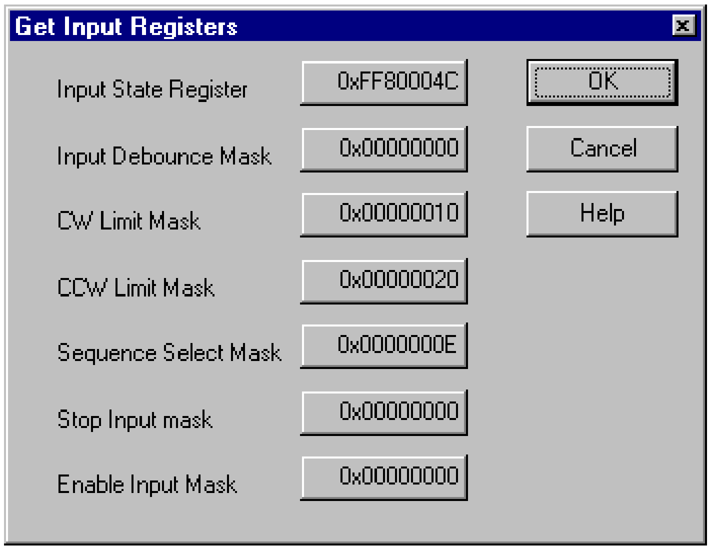
This register displays the sequence number designated as the autostart sequence which will execute when ever the drive is reset or powered. The default value is 256, which is no autostart sequence. To change the value go to ***Utilities/Initialize/Autostart***.

Debounce Time:

This register displays the Input debounce time. This is the time the input must be active or inactive before the change of state is registered within the drive. This is generally set for mechanical switches to avoid multiple contacts. Each input is individually set using the "***Configure Input*** "screen.

5.3 Input Register

The Input Register captures the current state of the Inputs as well as displaying the input configuration. This register is a read-only register. The Inputs are configured as binary but are displayed in hexadecimal format.



Input State Register	0xFF80004C
Input Debounce Mask	0x00000000
CW Limit Mask	0x00000010
CCW Limit Mask	0x00000020
Sequence Select Mask	0x0000000E
Stop Input mask	0x00000000
Enable Input Mask	0x00000000

Input State Register:

This static register captures the instantaneous value of the Input State. The value in hexadecimal format represents the active state of the inputs.

NOTE

*The Input State Register always reads an Input held high as **active** and an input pulled low as **inactive** regardless of the active state defined in the Input Configuration*

Binary Value of Inputs

	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
Active Value	64	32	16	8	4	2	1
Inactive Value	0	0	0	0	0	0	0

* Binary numbers are read from left to right.

Example:

0 Indicating Input Inactive. (Current Flowing).

1 Indicating Input active. (No Current Flow)

Hex	***Binary***	***Decimal***

Input State = *4C = 1001100 = 64 + 0 + 0 + 8 + 4 + 0 +0 = 76*

(Inputs 3, 4 and 7 are held high (active), Inputs 1, 2, 5 and 6 are pulled to ground (inactive)

Input Debounce Mask:

This register display, in hexadecimal format, which inputs have the Debounce time activated. This is the time the input must be active or inactive before the change of state is registered within the drive. This is generally set for mechanical switches to avoid multiple contacts. Each input is individually set using the "***Configure Input***" screen.

CW Limit Mask:

This register display, in hexadecimal format, which input is designated as the "*CW Limit Input*". Use the "***Configure Input***" screen to designate the inputs.

CCW Limit Mask:

This register display, in hexadecimal format, which input is designated as the "C*CW Limit Input*". Use the "***Configure Input***" screen to designate the inputs.

Sequence Select Mask:

This register display, in hexadecimal format, which input(s) are designated as the "*Sequence Select Input*". Use the "***Configure Input***" screen to designate the inputs.

Stop Input Mask:

This register display, in hexadecimal format, which input is designated as the "*Stop Input*". Use the "***Configure Input***" screen to designate the inputs.

Enable Input Mask:

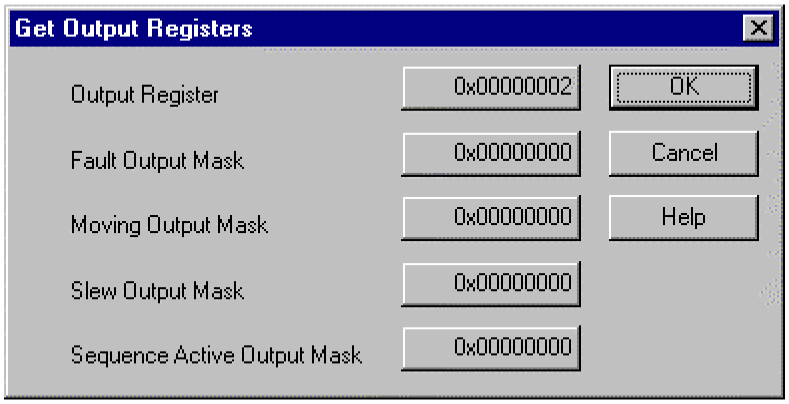
This register display, in hexadecimal format, which input is designated as the "*Enable Input*". Use the "***Configure Input***" screen to designate the inputs.

5.4 Motion Register

The ***Motion Register*** is read only register, which provides a “snap shot” of a move profile including Position, Motor Velocity, Target Velocity, and Acceleration.

5.5 Output Register

The Output Register captures the current state of the Outputs as well as displaying the output configuration. This register is a read-only register and the outputs are configured as binary but are displayed in hexadecimal format.



Output Register 0x00000002
Fault Output Mask 0x00000000
Moving Output Mask 0x00000000
Slew Output Mask 0x00000000
Sequence Active Output Mask 0x00000000

	Input 4	**Input 3**	**Input 2**	**Input 1**
Active Value	8	4	2	1
Inactive Value	0	0	0	0

Binary Value of outputs

* Binary numbers are read from left to right.

Output State Register:

This static register captures the instantaneous value of the Output State. The value in hexadecimal format represents the active state of the outputs.

Example:

0 Indicating Output Inactive.

1 Indicating Output active.

	Hex	***Binary***	***Decimal***	
Output State =	*6 =*	*0110 =*	*0 + 4 + 2 +0 = 6*	(Outputs 2 & 3 active)
Output State =	0B =	1011 =	8 + 0 + 2 + 1 = 11	(Outputs 1, 2 & 4 active)

Fault Output Mask:

This register display, in hexadecimal format, which output is designated a "*Fault Output*". The fault output becomes active when the drive faults. Use the "***Configure Output***" screen to define the outputs.

	NOTE
☞	*The fault output does not become active if power is lost or if the green/red status light is not lit.*

Moving Output Mask:

This register display, in hexadecimal format, which output is designated as a "*Moving Output*". The moving output becomes active when the motor begins a move and becomes inactive when the motor comes to a stop. Use the "***Configure Output***" screen to define the outputs.

Slew Output Mask:

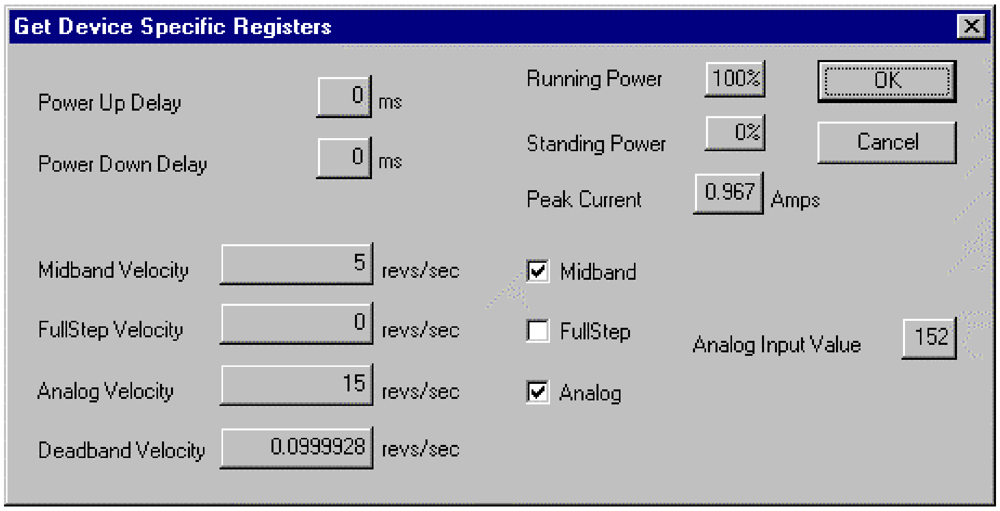
This register display, in hexadecimal format, which output is designated as a "*Slew Output*". The slew output becomes active when the motor velocity = target velocity. Use the "***Configure Output***" screen to define the outputs.

Sequence Active Output Mask:

This register display, in hexadecimal format, which output is designated a "*Sequence Active Output*". The output becomes active when ever a sequence is executing and becomes inactive when the sequence is completed. Use the "***Configure Output***" screen to define the outputs.

5.6 Specific Register

The Specific Register provides status of several functions such as Velocity Functions, Power Parameters and the value of the Analog Input.


Get Device Specific Registers

Power Up Delay	0	ms
Power Down Delay	0	ms
Midband Velocity	5	revs/sec
FullStep Velocity	0	revs/sec
Analog Velocity	15	revs/sec
Deadband Velocity	0.0999928	revs/sec
Running Power	100%	
Standing Power	0%	
Peak Current	0.967	Amps
Analog Input Value	152	

☑ Midband
☐ FullStep
☑ Analog

Power Up Delay

Displays the time in milliseconds the drive delays to go from “Standing Power” to “Running Power” when a move command is executed. The power up delay is set using the ***Utilities/Initialize/Power Parameters*** screen.

Power Down Delay

Displays the time in milliseconds the drive delays to go from “Running Power” to “Standing Power” when a move command is executed. The power down delay is set using the ***Utilities/Initialize/Power Parameters*** screen.

Running Power

Displays the percent of “Peak Current” the drive will utilize when the motor is moving. The running power is set using the ***Utilities/Initialize/Power Parameters*** screen.

Standing Power

Displays the percent of “Peak Current” the drive will utilize when the motor is ***not*** moving. The standing power is set using the ***Utilities/Initialize/Power Parameters*** screen.

Peak Current

Displays the peak current of the drive set by the ***Utilities/Initialize/Current*** command.

	NOTE
☞	*This is only available on the DM-224I drives where the current is set in software. The DM-2205i drive current is set by dipswitches.*

Midband Velocity

Displays the velocity which the midband compensation is enabled. The midband compensation monitors back EMF of the drive to detect resonant frequencies then changes the chopping frequency to prevent the motor from stalling. For midband compensation to be active the motor speed must be above the set midband velocity and the enabled box must be checked. This is set using the ***Utilities/Initialize/Velocity Functions*** command.

FullStep Velocity

Displays the velocity which the drive will switch from Microstepping to full step two phase on mode. To switch to full step mode, the motor velocity must be above the full step velocity and the enabled box must be checked. This is set using the ***Utilities/Initialize/Velocity Functions*** command.

Analog Velocity

Displays the maximum motor velocity at 10 VDC when running in analog velocity mode. The velocity will be scaled linearly from +/- 10 VDC. The enabled box must be checked, an acceleration rate set, and a continuous move command given for the analog velocity mode to work. This is set using the ***Utilities/Initialize/Velocity Functions*** command.

Deadband Velocity

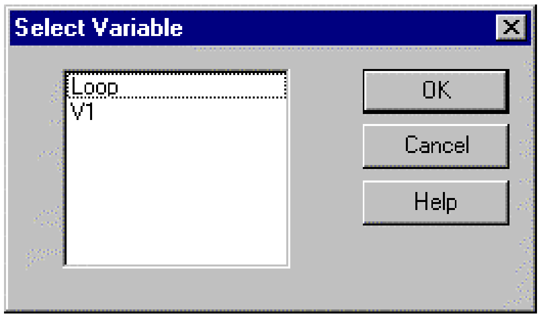
Displays the deadband window used in analog velocity mode in which the motor will not move. This prevents the motor from moving at standstill due to noise on the analog signal. This is set using the ***Utilities/Initialize/Velocity Functions*** command.

Analog Input Value

Displays the 8 bit digital value of the analog input (0-255). Since the specific register is a static screen the value is captured but not updated.

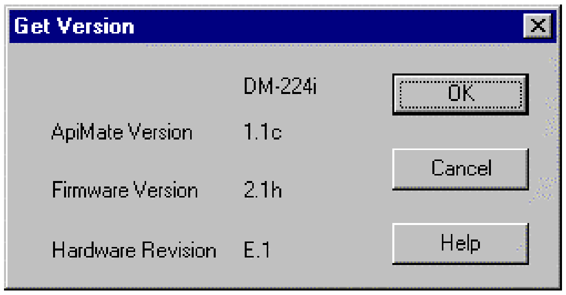
5.7 User Variables

This register displays the value of a user defined custom variable. Simply highlight the desired variable and click “OK”, the value of the variable will be given. All users variables are reset to zero when drive is powered down or reset.



5.8 Version Register

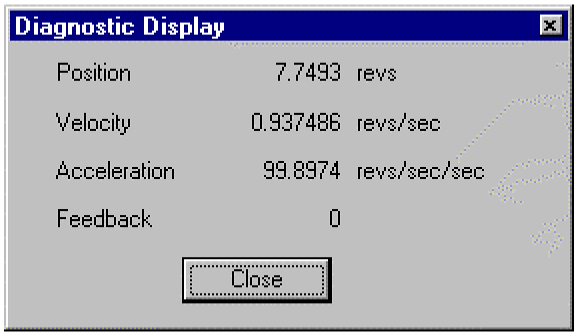
The version resister displays the drive type, APImate Version, Firmware Version and the Hardware Version of the drive. The firmware version can be updated in the field by using ***File/Flash Upgrade*** command.



	DM-224i
ApiMate Version	1.1c
Firmware Version	2.1h
Hardware Revision	E.1

Diagnostic Screen

The ***Diagnostic Screen*** window displays the current value of the Position, Velocity, Acceleration and Feedback Register. The window is updated through the communication port every ½ second and can be used to provide motion statistics while a sequence is executed.


Diagnostic Display

```
Position          7.7493  revs
Velocity        0.937486  revs/sec
Acceleration     99.8974  revs/sec/sec
Feedback               0
```

	NOTE
	*The diagnostic screen is provided as debugging tool and should only be used when running a sequence using the "***Execute***" command or the "***Sequence Inputs***". The diagnostic screen should* ***<u>not</u>*** *be used with the Player Buttons or left open when modifying, downloading sequences or during a flash upgrade. Doing so will cause APImate to lock up.* *Verify communications before opening the diagnostic window. A communication error will lock up the program*

6. Appendices

6.1 Flash Upgrade

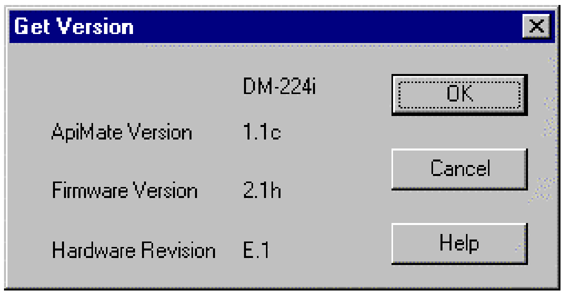
The ***Flash Upgrade*** command is used to update the firmware level within the drive. The firmware is located in the flash memory and is considered the intelligence of the drive. Flash upgrades generally correct existing firmware bugs or provides additional commands and may require an updated revision of APImate. A Flash Upgrades will retain all sequences archived in flash memory. To perform a Flash Upgrade:

- Click on File/Flash Upgrade.
- Locate the Firmware (*.bin) file and click "OK"
- Enter the Password (See Note)

	NOTE
⚠	*Password for DM-224i is **API DM-224i** and is case sensitive.* *Password for DM-2205i is **API DM-2205i** and is case sensitive.*

6.2 Firmware Version

To determine the firmware version of the drive, click on ***Utilities/Get Registers*** then click on Version Resister. The Firmware Version will be displayed.


Get Version

	DM-224i
ApiMate Version	1.1c
Firmware Version	2.1h
Hardware Revision	E.1

6.3 Trouble Shooting

Red Fault LED on during Power-up

- Verify power to drive is supplying correct voltage for the drive type. (Refer to Installation Manual)
- Check wires for a loose connection or possible short and verify drive is wired correctly. (Refer to Installation Manual)
- Power up the drive without the motor connected. If the fault LED goes away it is a good chance the motor is bad. (Ohm out motor checking for a phase to phase or phase to ground short before reconnecting to the drive)

Motor does not move

- Check to see if drive is powered-up or if the drive fault LED is on.
- Verify that an acceleration and velocity command other than zero was executed prior to the move command.
- Confirm that sequence is loaded into the drive, contains a move command and that the correct sequence is executed.
- Verify that hard limit or stop input is not active.
- Verify current has been set via software (DM-224i) or hardware (DM-2205i)

If Script Pointer does not align properly on Script

- To correct this problem click on the Windows 95 ***Start*** key, and go to ***Settings***, ***Control Panel*** then ***Display***. Click on the ***Settings*** tab and select the “Small Font” size. You may have to reboot the computer before the new setting take effect.

Communicate Errors

- Check to see if drive is powered-up or if the drive fault LED is on.
- Check communication cable for proper wiring or a loose connection. (Refer to Installation Manual)
- Confirm that switch setting for communication match the communication type. (RS-232, RS-485, etc. Refer to Installation Manual)
- Verify that the correct PC comport is selected in Axis Configuration.
- Confirm that the Axis Number on the Axis Configuration window corresponds to the Axis Number of the drive switch settings. (Refer to Installation Manual)
- Communication errors may occur while executing the sequence using the "Player Buttons" on the toolbar. Since the Player Buttons execute the program from the PC through the RS-232 port, a communication error can occur during long moves or other commands which the PC is polling the drive but the drive is too busy to respond. If this occurs download the sequence to the drive and use the Sequence Execution command.
- Diagnostic screen can **not** be active while sending commands from APImate 1.0 to the drive. This will result in communication errors caused by the diagnostic screen continuously requesting information from the drive and being scrambled by another command being sent down the multi-drop RS-232/422/485 line.

More Help

For additional assistance contact your local API Controls Motion Automation Company or call API Controls application Dept. at 1-800-566-5274

Our Web Site is www.apimotion.com

6.4 Shortcut Keys

The following shortcuts are available with APImate 1.0:

Function Keys

F1 - Help Screens

F2 - Axis Configuration Window

Initialization Commands

CTRL - ALT - S	- *Autostart Sequence* Window
CTRL - ALT - A	- *Peak Current* Window
CTRL - ALT - I	- *Configure Inputs* Window
CTRL - ALT - O	- *Configure Outputs* Window
CTRL - ALT - P	- *Power Parameters* Window
CTRL - ALT - V	- *Velocity Functions* Window

Program Control

CTRL - SHIFT - A	- Abort
CTRL - SHIFT - X	- Execute sequence (Executed through PC)

Program Control

SHIFT - A	- Abort Sequence
SHIFT - R	- Archive Sequence
SHIFT - D	- Delete Sequence
SHIFT - T	- Delete All Sequences
SHIFT - W	- Download Sequence (From Drive)
SHIFT - N	- Download All Sequences (From Drive)
SHIFT - X	- Execute Sequence
SHIFT - S	- Directory of All Sequences
SHIFT - U	- Upload Sequence with no execution
SHIFT - V	- Upload Sequence with execution

Initialization Commands

ALT - M	- ***Arithmetic*** Command
ALT - I	- Conditional "***If*** " Command
ALT - H	- Conditional "***While***" Command
ALT - E	- ***Reset*** Command
ALT - S	- ***Start Move*** Command
ALT - B	- ***Absolute Move*** Command
ALT - C	- ***Continuous Move*** Command
ALT - D	- ***Variable Distance Move*** Command
ALT - N	- ***Variable Position Move*** Command
ALT - R	- ***Relative Move*** Command
ALT - A	- Set ***Acceleration Register***

ALT - L	- Set ***Debounce Time Register***
ALT - F	- Set ***Feedback Register***
ALT - O	- Set ***Output Register***
ALT - P	- Set ***Position Register***
ALT - V	- Set ***Velocity Register***
ALT - Q	- ***Stop Move*** with Decel Command
ALT - K	- ***Stop Move*** without Decel Command
ALT - T	- ***Delay*** Command
ALT - W	- ***Wait on Inputs*** Command
ALT - X	- ***Wait Motor Stop*** Command
CTRL - ALT - B	***Combo Absolute Move*** Command
CTRL - ALT - C	***Combo Continuous Move*** Command
CTRL - ALT - R	***Combo Relative Move*** Command

Glossary

Absolute Move – A move to an absolute motor position based on the position register. The move distance and direction is relative to the current position.

Analog Velocity – The maximum/minimum motor velocity allowed based on a +/- 10 VDC analog signal when running in analog velocity mode. The motor velocity proportionally scaled to the analog input.

Archive Sequence - The process by which the programs are loaded from RAM memory and stored in nonvolatile Flash memory.

Autostart Sequence -The sequence designated to execute when the drive powers-up or when reset. This sequence will generally contain commands to configure the drive.

Continuous Move – Is a move that rotates the motor at a fixed velocity until a stop command is executed or a new velocity command is given or. The sign (+/-) of the velocity determines the direction of rotation.

Deadband Window – Sets the minimum speed with which the analog signal will be ignored while running in Analog Velocity mode.

Debounce Time – The amount of time an input must be active or inactive before the drive registers a change of state.

Download – The process of loading a sequence(s) from the drive back to the PC.

Firmware – Is the internal drive software, which contains the operating code of the drive.

Flash Memory – The nonvolatile drive memory used to store sequences when the sequence is archived. The sequences are loaded from Flash to RAM memory every time the drive is powered up or reset.

Flash Upgrade – The process of uploading new firmware to the drive. A password is required to ensure firmware is compatible with the drive.

Full Step – Each digital pulse received by the motor, the shaft will rotate 1.8° mechanically. In Full Step mode the motor requires 200 steps to move one shaft revolution.

Inductance Compensation – This parameter tunes an individual drive to the specific motor based on motor inductance. Changing the value of the inductance compensation changes the drive chopping frequency.

Inputs – Used to interface a drive with it's external environment. An input receives a digital signal from an external source.

Macro Select Input – The status of these inputs determine the sequence number to be executed when the "Macro Start" input becomes active. The sequence must be loaded in the drive RAM memory.

Macro Start Input – Starts sequence execution based on the status of the "Macro Select" inputs. The macro start input is always defined as Input 1.

Microstepping – Refers to a technique whereby the motor coil current is precisely controlled to sub-divide the 1.8° step into much finer increments. Typically it refers to 2000 steps or more per one shaft revolution.

Midband Stabilization – The technique of monitoring the back EMF of a motor to determine if the system is at a resonant frequency. The drive will compensate for resonance by changing the angle that the current is sent into the motor winding to help prevent the motor from stalling.

Midband Gain Compensation – This function changes the gain on the midband so that smaller motors can produce a stronger EMF signal when the motor rotates.

Output - Used to interface a drive with it's external environment. An Output sends a digital signal from the drive to an external relay or input.

Power Up Delay – The specified time given to allow the drive to go from reduced power mode to full power once a move command is executed.

Power Down Delay – The specified time the motor must be idle before the drive goes into a reduced power mode.

Relative Move – Is a move based on a fixed move distance with the direction of rotation determined by the sign (+/-) of the move distance.

RPS – Indicates motor velocity in revs/sec.

RAM Memory – The volatile drive memory used to execute the sequences. The sequences are loaded from Flash to RAM memory every time the drive is powered up or reset.

Script – Refers to a program created using APImate 1.0 that has not yet been uploaded to a drive.

Sequence – Refers to a program created by APImate 1.0 that has been uploaded and/or stored on a drive.

Upload – Refers to loading a script(s) from the PC to the drive.

Variables, Drive – Preset variables contained within the drive and are accessible by the user. Some of these variables maybe "Read Only"

Variables, Custom – Variables created and defined by the user. These variables are created using the "Arithmetic" function.

INDEX

North America

45 Hazelwood Drive
Amherst, New York 14228 USA
(716) 691-9100
(716) 691-9196 Fax
800-566-5274
Applications
http://www.apicontrols.com
ftp://mtg-int1.com

Europe

European Headquarters
Switzerland
57 rue Jardinière
CH-2300 La Chaux-de-Fonds
+41 (0) 32 9256111
+41 (0) 32 9256594 Fax
Applications

Asia

Japan
7-10, Nihombashi-honcho
4-chome, Chuo-ku,
Tokyo 103
Tel: +81/ (03) 3241-0201
Fax: +81/ (03) 3241-0221

China
307 Santra Building,
No. 3, West Street He Ping Li,
Chao Yang District,
Beijing 100013, P.R. China
Tel: +86/ (10) 6429-7049
Fax: +86/ (10) 6429-7049