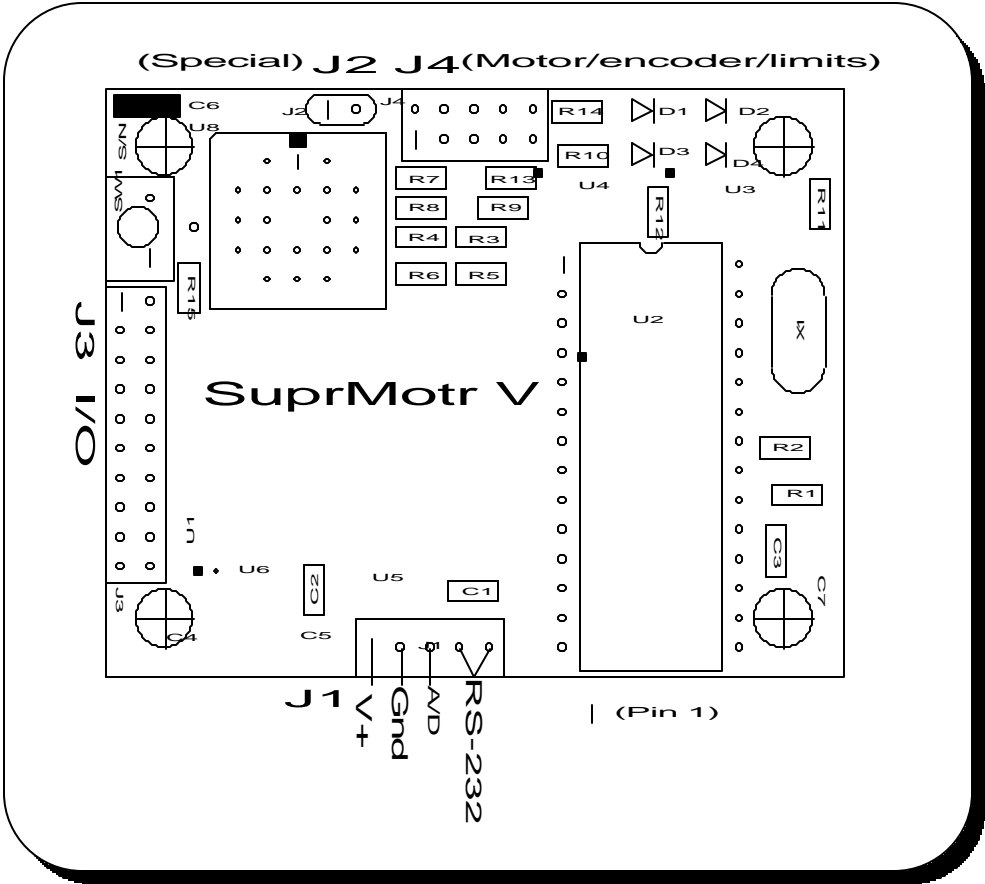


SuprMotr V ©

Stand-alone,
Miniaturized, Networkable,
Servo Control and Driver
for Small DC Motors



Rev. A, 5/1/2003
Rev. B, 5/27/03
Rev. C, 5/29/03

Copyright (c) 2003, DIVA Automation Ltd.

1.0 General

This document is provided to aid in the effective application of the DIVA Automation SuprMotr V © single axis motion controller. It is intended as an introduction and a guide for installation. A more complete discussion of user commands and applications are included in a separate manual. (In the interests of brevity and clarity, the term "SuprMotr" will be used throughout this document, without the model designation and copyright mark.)

The SuprMotr is a miniaturized servo controller intended for motion control in research as well as industrial applications. In a single package, it provides a complete stand-alone control system for the smaller motors typically used in ultra-precision positioning systems achieving precisions of less than one micron.

For larger motors or for brushless motors, an external amplifier is required.

The SuprMotr utilizes an on-board regulator to derive its own operating voltage from the motor power and for powering the incremental encoder which is required for position feedback.

To use the SuprMotr, it is only necessary to connect a source of power adequate for driving the motor under expected operating conditions, a connection to an RS-232 terminal, an incremental encoder driven by the motor, and the motor itself.

Once all connections have been made, see the section on establishing communications, then begin controlling the motor.

The SuprMotr provides simultaneous control of position, velocity, acceleration and maximum torque. Although each parameter is programmable, each one is also set to a stored value when power is applied. All parameters may be changed at any time, but the settings will be returned to those current at the last **UD** (update) command whenever power is removed and re-applied, or when an **RT** (reset) command is issued.

By use of the supplied commands and features, the user will be able very quickly to command the motor to move to any desired position. During the move, the velocity and acceleration will be controlled to the most recent settings for these parameters. It is not necessary to set them up for each move. In many applications, the initial setup values are adequate for all uses and need never be changed. Other applications may vary them continuously, as required.

To illustrate the power and simplicity of the **DIMAC** commands, we can define an application that needs only to cycle between two positions separated by 3500 encoder counts. We may wish to perform this operation 100 times, with a one second delay at each end. It may also be desirable to move at one speed in the positive direction and at a higher speed on the return. This application would require only the following commands to be issued:

Note: Default values

Each SuprMotr is shipped from the factory with values preassigned for all parameters. These are called the "default" values because they are the values that are used by default if the user does not select other values.

They are also the values stored in EPROM that will be applied each time power is applied for controllers that do not have EEPROM installed. If different values are desired for initial conditions, then please contact the factory.

For controllers with EEPROM, the UD command is used to save the present value of all parameters. These values will become the new default values. After choosing the optimum settings for your controller, use the UD command to save them.

The default values installed at the factory may change from time to time so they are not listed here. To determine the value of any default, use the reporting commands as described later in this manual. TY reports the velocity setting. If it has not been changed since the controller was last reset, then this is the default value. TL reports the acceleration. GP reports proportional gain, and so on.

Assuming that the default gain values are adequate for the motor in use, we must first enable the servo loop with the command **MN**. (**M**otor **o**n) The commands required to perform the cyclic operation would be:

SV200,MR3500,WS,SV20000,MR-3500,WS,RP99

This simple program instructs the SuprMotr to drive the motor 3500 encoder counts in the positive direction from its present position at a velocity of 200 encoder counts per second. The **WS** command tells it to wait one second after this point has been reached before continuing. The velocity is set to 20,000 cps before returning and the **RP** command tells it to repeat the entire command line 99 times for a total of 100 iterations. We could also insert a **TI** command to report the number of iterations to the operator or host program for a running status report.)

In addition to being very intuitive and easy to learn, the two-letter structure uses the initial letters of the operation to be performed as much as possible so that it becomes easy to remember them for programming without a manual at hand as well as to understand the meaning of existing command lines.

An additional feature of the SuprMotr is the use of "macro commands" to create new commands. The example above can be made into a macro by prefacing it with the **MD** (**M**acro **D**efinition) command.

MD1,SV200,MR3500,WS,SV20000,MR-3500,WS,RP99 creates a new command which may be used at any time by entering **MC1**. The maximum number of macro

commands depends on the amount of available memory and can range from just a few to several thousand. Because of the powerful nature of macro commands, even very sophisticated machine sequences can be implemented with 25 or less macro definitions.

An additional feature of the macro command capability is that if there is an MCO defined, it will be executed automatically when power is applied or whenever a reset command is given. Through use of this capability, the SuprMotr is capable of 100% autonomous operation, without the need for any external connection except power.

1.1 Multiple axis control.

The SuprMotr is unique among available controllers (and most RS-232 devices) in that it allows the connection of up to 32 controllers to a single RS-232 channel. This capability not only can save an enormous amount of cabling in larger systems, but also saves in only requiring a single communications port for each 32 axes of control.

Unlike many bussed systems, such as the IEEE-488, switching between devices is extremely simple. Only a two-character sequence must be transmitted to select a single controller on the bus and deselect all others. When the SuprMotr controller receives an ASCII 01 (control-a), it compares its own address stored in EEPROM with the next character following the ctrl-a. If there is a match with the address, it enables itself to respond to subsequent commands. If not, it disables the response to any subsequent transmissions that are not address selection sequences.

If address 0 is selected, then all SuprMotrs on the bus will be selected until a subsequent address higher than 0 is sent. This capability makes it easy to synchronize multiple-axis motion. It is also beneficial in downloading identical commands such as macro definitions to all SuprMotrs simultaneously, but will not work for reporting commands.

Any motion sequence or operation begun prior to being de-selected will continue to be executed except for those that report data. In this manner, each SuprMotr on the bus can be addressed, programmed to execute any desired operation, then de-selected before the sequence is repeated for any other motors selected.

Remember that each SuprMotr is de-selected when power is first applied, or after an **RT** command. An address selection sequence must be transmitted to begin control.

The EEPROM address for each SuprMotr is set with the **DAn** (define address) command. All addressed SuprMotrs on the bus when this command is issued will be set to address **n**.

Note: (Rev C.)

Some models of the SuprMotrV do not have EEPROM installed. As a result, the DA command is not supported. However, address selection may be performed by use of the I/O lines.

The three highest numbered I/O channels (14, 15 and 16) were used as address selection inputs. This does not interfere with their normal use as input or output channels, except during the power up phase. When power is cycled, or when the SuprMotrV is reset for any reason (including use of the RT command), the state of these three pins is sampled and used to define the address of the board.

The way it works is that J3-13 is the LSB (value 1), J3-14 is the NLSB (value 2) and J3-11 is the MSB (value 4). To assign address 5 to a board, you will connect pins 13 and 11 to +5v and pin 14 to ground. This gives value 1 plus value 4 = 5. For address 0, ground all three pins.

There are 8 possible addresses, with the following connections:

| | J3-11 | J3-14 | J3-13 |
|-----------|-------|-------|-------|
| | ---- | ---- | ----- |
| Address 0 | gnd | gnd | gnd |
| 1 | gnd | gnd | +5 |
| 2 | gnd | +5 | gnd |
| 3 | gnd | +5 | +5 |
| 4 | +5 | gnd | gnd |
| 5 | +5 | gnd | +5 |
| 6 | +5 | +5 | gnd |
| 7 | +5 | +5 | +5 |

When power is first applied, all boards are in the "addressed" state. If more than one board is connected, they will conflict if any command is given to return data. This includes the copyright notice. To select one board and deselect all others, send an ASCII ctrl-A (01) followed by the address number. Do not send an enter or carriage return code. After selecting the address, a TB command will display the address of the selected board. This makes it easy to poll through all addresses to detect which addresses have been connected.

(Continued on next page)

Notes on board addressing--with EEPROM or without

To test the controller addressing, set an address and give any reporting command, such as TT. The address of the board is reported before the data. If you select address 5, then the TT command will report "5T:+0000000000", for instance. If you select address 1, TT will report "1T:+0000000000". In each case, the board address is reported with the data.

To select a particular board, send CTRL-A and the board address. This will select that address and deselect all others. To go back to the example of a board set to address 5, hold down the CTRL key and press A, then press 4. This will select board address 4, which is not the same as the board you are testing. If you then give a TT command, nothing will happen because there is no board connected with address 4. Now, hold down the CTRL key and press A, then press 5. TT will then report "5T:+0000000000".

Please note that you should not press the Enter key after the CTRL-A 5. The Enter is used to enter commands, such as the TT. CTRL-A followed by a number is not a command, it is only an address selection.

sending: If you have five boards connected and have assigned addresses to them of 1,2,3,4 and 5, then you could communicate with each one individually by

```
CTRL-A 1 T T Enter  
CTRL-A 2 T T Enter  
CTRL-A 3 T T Enter  
CTRL-A 4 T T Enter  
CTRL-A 5 T T Enter
```

This would return:

```
"1T:+0000000000"  
"2T:+0000000000"  
"3T:+0000000000"  
"4T:+0000000000"  
"5T:+0000000000"
```

If you are sending the command from a program, the ASCII code for CTRL-A is 01, so the sequence of hex digits to send CTRL-A 5 T T Enter would be: 01 35 54 54 13. If you are controlling from a terminal emulator, such as HyperTerm, simply type the keys.

After the address selection has been sent, the boards that do not match the address will continue to monitor the commands that are sent, but will respond only to an address selection sequence. Only the board that was selected will respond. You may send any commands to it as if it were the only board connected. There is no need to send another address selection sequence until it is desired to select a different board.

1.2 Physical connections

There are four connectors on the SuprMotr to provide access to its functions.

The motor connections are included in the 10-pin IDC connector, together with the encoder power output, encoder inputs and limit/reference switch inputs.

If the encoder signals are not properly phased, the motor will run at full speed because the drive being applied causes an increase in the error. When wiring a new system, it is always wise to limit the motion of the motor until the phasing has been verified.

The **TP** (tell position) command may be used to test the encoder inputs before closing the servo loop with the **MN** (motor on) command. **TP** reports the position of the encoder when rotated manually as well as when under control of the servo loop. If the **TP** command does not report an increase in position when rotated in one direction and a decrease when rotated in the opposite direction, there is a serious problem. The motor should be disconnected until this problem is corrected.

When the encoder operation has been verified, it is good practice to set the torque limit to a low value before issuing the **MN** command. This makes it easier to restrain the motor in the event that the phasing is wrong. If the motor attempts to drive continuously, the **MF** command may be issued to turn off the servo loop. Then the **PH** command can be used to correct the phasing. After changing the phasing, the **MN** command should cause the servo loop to hold the motor at any commanded position. If the encoder has been verified and the motor is unstable with either setting of the **PH** command, then there is a serious problem.

The 10-pin IDC connector **J4** has the following connection assignments:

| <u>Pin</u> | <u>Function</u> |
|------------|-----------------------|
| 1 | Motor + |
| 2 | +5 volts to encoder |
| 3 | Encoder B |
| 4 | Encoder A |
| 5 | Ground |
| 6 | Motor - |
| 7 | Reference switch |
| 8 | Positive limit switch |
| 9 | Negative limit switch |
| 10 | Ground |

Please note that the terms "positive" and "negative" in connection with limit switches refer to the polarity of the position. The "positive" limit switch will prevent additional motion in the "positive" direction, when actuated, and vice-versa. Either switch will

allow motion in the opposite direction. Of the two switches, only one will interrupt motion in a given direction. To determine the correct wiring for the switches, set the system into a safe position, set a low velocity with the **SV** command, and begin motion with an **MR**, **MA** or **FE** command and enable limit switch operation with the **LN** command. While the motor is moving, actuate one of the limit switches. If the motor stops, that is the correct one to connect at the end of desired motion for that direction. If it does not stop, actuate the other limit switch. If the motor still does not stop, remove power and read further.

Limit switches may be connected so that the stop condition is either a low signal (ground) or a high (+5v), but both must be the same. The active state is saved in EEPROM with velocity, acceleration, PID gains and other system parameters with the **UD** command and restored on reset. The active state may be changed with the **LL** (Limits Low) and **LH** (Limits High) commands.

Power for the motor and logic functions is applied through **J1**. This connector is also used for the RS-232 signals and for one analog input.

| <u>Pin</u> | <u>Function</u> |
|------------|---------------------------|
| J1-1 | Motor + |
| J1-2 | Ground, (also DB-9 Pin 5) |
| J1-3 | Analog input |
| J1-4 | RS-232 Rx (DB-9 Pin 3) |
| J1-5 | RS-232 Tx (DB-9 Pin 2) |

J2 is a special purpose connector which is driven from the PAL. It may be used for many purposes, including a gated connection to the RS-232 data received by the SuprMotr, for controlling external assemblies. The use of this connector is controlled by the system firmware and by the PAL program. Pin 1 is the signal. Pin 2 is ground.

J3 is a 20-pin IDC header which provides access to 18 general-purpose I/O lines, plus ground and +5 volts. There are several channels of analog input which may also be used as digital I/O, and other channels which are digital-only.

| <u>J3-pin</u> | <u>Function</u> | <u>Alt. Function</u> | <u>CPU port</u> |
|---------------|-----------------|----------------------|-----------------|
| J3-1 | PAL input | | P4--3 |
| J3-2 | I/O Ch 9 | | P4--2 |
| J3-3 | I/O Ch 3 | | P8-2 |
| J3-4 | Brake out | | P3-0 |
| J3-5 | I/O Ch 8 | | P8-7 |
| J3-6 | I/O Ch 6 | | P8-5 |
| J3-7 | I/O Ch 5 | | P8-4 |
| J3-8 | I/O Ch 7 | | P8-6 |
| J3-9 | I/O Ch 2 | | P8-1 |
| J3-10 | I/O Ch 4 | | P8-3 |
| J3-11 | I/O Ch 16 | An Ch 6 | * P6-7 |
| J3-12 | I/O Ch 1 | | P8-0 |
| J3-13 | I/O Ch 14 | An Ch 4 | * P6-5 |
| J3-14 | I/O Ch 15 | An Ch 5 | * P6-6 |
| J3-15 | I/O Ch 12 | An Ch 2 | P6-3 |
| J3-16 | I/O Ch 13 | An Ch 3 | P6-4 |
| J3-17 | I/O Ch 10 | | P5-3 |
| J3-18 | I/O Ch 11 | An Ch 1 | P6-2 |
| J3-19 | Ground | | |
| J3-20 | +5V | | |

* (May also be used for board address selection. See previous notes.)

2.0 Beginning operation.

The SuprMotr command interface has been optimized for interactive control using a standard terminal emulation program, such as that provided by Hyperterm. When a command is entered, it is stored in a buffer and may be repeated with the "Enter" key (which sends a carriage return character). Command syntax is tested on a character-by-character basis, with an error flag returned to the operator immediately on detection of an error.

Set the terminal emulation to 9600 baud, no parity and one stop bit. Higher baud rates may be used, but this is the factory default setting.

To verify proper communications, use a reporting command such as **VE**. This command reports the version number of the firmware. The **HE** command reports the list of commands resident in the SuprMotr.

When you are ready to begin system operation, enter an **MN** to enable the servo loop. Enter **MR1000**(rtn). The motor should move 1000 encoder counts in the positive direction. When it stops, enter **TP**. The position should be very close to 1000. Enter **TT,TE** (rtn). This reports where the motor should be (1000) and the distance of the actual position from this target. If the reported position was 998, then the reported error should be 2. If the reported position was 1002, the reported error should be -2.

Please note that it is possible that the motor may have moved slightly, due to external forces or oscillation caused by excess gain. In this case, the difference between the target position and the actual position would not be the same as the reported error. For a better comparison, enter them all at the same time. **TT,TP,TE** (rtn) will report all three values but it is still possible that a servo loop update could have occurred during the reporting process.

In fact, at 9600 baud, it is certain that the loop will have been updated during the report. Each value reported sends a total of 14 characters, including carriage return, line feed, etc., and 9600 baud is roughly 960 characters per second. The loop is updated 1000 or more times per second, so do not expect these values to always agree with each other if there is any possibility of motion during the report.

Now enter **MR-1000** (rtn). The motor should return to zero, or home, position. Press the return (or enter) key again. The motor should move another 1000 counts in the negative direction. Press it again and again. The motor should move each time.

Now try **GH** (rtn). The motor should return to its original position when power was first applied. This command is equivalent to **MA0**.

Now for some really advanced programming. Enter **MR1000,WS,TE** (rtn). The motor should move 1000 counts positive. When it arrives at this position, it should report how close it came to the target, after a delay of one second.

If the motor is connected to a linear stage you may determine the range of motion by commanding it to move continuously with the **FE** (find edge) command. This command is normally used for locating a reference position but may also be used as a convenient means of commanding continuous motion. Our purpose here is to set the system limits to safe values and drive the stage to its physical limit. This may be accomplished by sending **SQ40,SM200,SV10000,FE1,WS,DH,TP** (rtn).

This command string sets the maximum output to a value low enough to prevent damage to most systems. It also sets the maximum allowable error to a low value which should cause the servo loop to open if any significant obstruction is encountered and sets the velocity to a low value and tells the motor to move in the negative direction. We want it to move until it can no longer maintain the velocity setting and stop there with the servo loop disabled. Once it has stopped, it should define that position as 0 position and report that position (mainly to let us know it happened).

All of these suggested values are typical and may be modified for systems that are significantly different from the default assumptions.

When the stage has run to the end of travel, the position should now have been set to 0 by the **DH** command. The position should have been reported as some very low value and the servo loop should have been disabled by the act of encountering the stop which would have caused excessive error. To move to the opposite end of travel, we must enable the loop again with the **MN** command. We can do this as a separate command or include it in the motion command. The values we set for maximum output and velocity will still be in effect.

Note: We may wish to define HOME position as some distance away from the physical stop. If so, before continuing, issue the command **MN,MR1000**. (Ordinarily, we need not include the **MN** before the **MR** command, but in this case we know that the loop has been disabled so we will not be able to move until it is re-enabled.)

We can move to the other end with this command: **MN,FE0,WS,TP** (rtn). The motor should begin moving toward the other end of travel, continuing until it is reached. When it does so, the position reported will be the effective length of travel for the stage. A **MN,GH** (rtn) command should cause the motor to return to the other end, but without disabling the servo loop on arrival because the end of travel was reached without encountering an error condition.

Now we can try some other things with the knowledge of the valid maximum distance we can travel. Assuming that the position at the positive end of travel was 1,000,000

encoder counts (a reasonable value), we can do some more experimenting without fear of damage to the mechanical system.

Note: A motor with a 500 line encoder yields 2000 encoder counts per revolution. In precision stages, a lead screw pitch of 2 turns per millimeter is not uncommon and a gear ratio of 8:1 might be found. In such a system, the motor must rotate 16 revolutions to move the stage by 1 mm. Sixteen motor revolutions will generate 32,000 encoder counts with a 500-line encoder. If the stage is 100 mm long, there will be 3,200,000 encoder counts in moving from one end to the other.

If we are at the negative travel limit of the motor, as would be the case if we are following the scenario described above, we can safely increase the values we used for finding the stops to something more consistent with full performance. In the notation above, we assumed an encoder with 500 line resolution. At 6,000 RPM, it will generate 200,000 counts per second. Therefore, to operate at full speed, we can **SV200000** (rtn). Note that there is no comma allowed when specifying values to the SuprMotr. The comma is reserved for use as a command separator. However, spaces are ignored so they may be used to visually format large numbers, such as **MA12 345 678**.

We can also increase the maximum output to full scale unless there is a safety reason requiring a lower setting. **SQ255**(rtn) will do the trick. Note that the maximum output might be required to drive the motor at the full speed we have programmed.

The maximum expected following error for a system with reasonable friction levels should be no more than 10,000 and would be expected to be no more than around 500 when moving at full speed. Unless there is some other reason for setting a lower value, set the maximum error to 10,000 using **SM10 000**(rtn).

Unless we wish to have a slower acceleration, we can set the acceleration to, say, 800,000 counts per second per second, which will attempt to accelerate to 200,000 cps in 1/4 second. If the system is not physically capable of achieving this acceleration level, the loop will be disabled when the following error exceeds the set limit. If so, enable the loop once more with the **MN** (rtn) command and set the acceleration to a lower value, such as **SA500 000** (rtn). Continue in this manner until a physically possible rate is determined.

Note that these numbers were based on certain assumptions of encoder resolution and maximum motor speed. They must be suitably adjusted for other combinations.

We are now ready to make some longer moves. Enter **MR100 000**(rtn) and verify that the motor has moved that distance. Press the (rtn) key again and see that the motor moves the same amount in the same direction. Press it again and again. The motion should continue each time the (rtn) key is pressed.

A **GH**(rtn) command will return to the previously defined HOME position. While the motor is in motion, give the command **TE,TP**(rtn). You will see the distance to the target

scrolling down your screen as it is approached. Since we are moving to HOME position, we would have had the same effect if we had commanded **TP,RP**(rtn). A note of caution: many terminal emulators cannot display data as fast as this command would send it and the operator certainly cannot read this fast, so it is usually preferable to insert a delay, such as 100 ms. **TP,WA100,RP** displays the position ten times/second.

Or, while in motion, we could have given **TV** and pressed the (rtn) key as often as desired to read the actual number of encoder counts moved during the last second. Since it measures the actual distance traveled during the previous second, **TV,WA,RP** will display the velocity each time it is updated.

Set the acceleration to a lower value, such as 50,000 (**SA50000**(rtn)) and see the velocity increase and decrease over a longer period. Set the velocity to various values and issue commands to move to various positions using either **MA** or **MR** commands. Try **MA10000**(rtn). Do it again after motion has stopped. There should be no movement. It is already at this position.

Move the stage some distance away from the end, say 100,000 counts, and try this: **MR50000,WS500,MR-50000,WS500,RP9**(rtn).

The stage should cycle back and forth 50,000 counts with a delay of a half second at each end. It should do this ten times--once by command and nine more repetitions.

By now, you have seen how easy the SuprMotr is to set up and how easily complex operations can be controlled. Use the **HE** command to display a list of all commands installed in your system and try using each one. If you don't understand hexadecimal notation, just ignore the **HM**. Hex mode is only useful to those who prefer it.

There is a separate document which covers operation of the built-in debugger. Since it provides access to the entire system without supervision, it is not recommended that non-programmers make use of the debugger. There is nothing the average user can gain by accessing the debugger.

2.1 Setting PID gain values.

The PID gain values stored in the SuprMotr have been chosen for an encoder with 100 line resolution. The encoder resolution is an effective contributor to the gain equation because it describes the physical angle through which the encoder will turn in response to a given level of drive to the motor. If an encoder with 500 line resolution is used, the gain may need to be decreased from the default setting to prevent oscillation. There are many factors in actual systems that determine the overall loop characteristics, so it is impossible to define optimum values without a complete knowledge of the system, but the default values usually are a good starting point and may be adequate for all operation.

There is a certain amount of interaction between the proportional (P), integral (I), and derivative (D) gain settings, so it is usually necessary to perform a series of trial settings before finding an optimum combination.

In general, the P gain may be thought of as the main servo component. If the system has only moderate friction and very low inertia, there is no need to assign gain values to the I-gain and D-gain if moderate values of position error are acceptable. The primary function of the integral gain is to overcome continuing errors, such as those caused by friction which prevent the target position from being achieved. If the motor consistently stops before reaching the target, then the I-gain should be increased to reduce the offset.

Before increasing the I-gain, it is usually best to adjust the P-gain and D-gain. The D-gain's main contribution is in reducing overshoot caused by load inertia and in reducing the settling time required to reach a complete stop at the target.

Start by setting the I and D gains to zero. **DIO,DDO**(rtn). Then increase the P gain until the motor oscillates at the end of a move. For best results, make a test move that allows the motor to reach something like full speed, with full acceleration, so that the target is arrived at with full force. When the P gain is so high that the motor oscillates at the end of the move, reduce the gain sufficiently so that the oscillation does not happen under the same conditions.

With the D-gain set to 0, the motor will probably overshoot the target and may oscillate about the target position momentarily before settling down. If so, increase the value for the D-gain in small steps, repeating the test move between each step until the damping is sufficiently improved.

Due to the fact that the D-gain acts in opposition to the P and I gains, after the D-gain has been set, it is probably possible to increase the P-gain again, past the original value where oscillation occurred. If so, work back and forth between the two gains until you are satisfied that the settings are optimal. Record these values. Check them by monitoring the following error during a long move by using the **TF** command. The following error is the difference between the actual position of the motor and the ideal position at that point in time. When properly set up, the value reported by the **TF** command will be very repeatable and can be decreased by increasing the P- and I-gains.

When the following error begins to jitter or oscillate during motion, the gains are too high and must be reduced.

If the static error at the end of a move remains higher than acceptable and the PD-gains have previously been optimized, then this static error may be reduced by increasing the I-gain. The I-gain also has the effect of increasing the P-gain action and of decreasing the D-gain action. If the I-gain is increased, it may be desirable to modify the PD-gains somewhat.

The good news is that this is usually a one-time procedure for a given mechanical system. Once the optimum values are determined, which shouldn't require more than a few minutes, there is no need to change them unless some significant aspect of the system is modified. A change of encoder resolution or a more powerful motor or different gear ratio or significant change in load or friction or operation in the vertical plane are instances where it would be expected that the optimum gain values would require determining once more.

In extreme cases, it may be desirable to use different sets of gain values for loaded and unloaded motion. Leadscrew-driven stages rarely encounter this type of problem because they frequently utilize gearhead reduction and the percentage of load inertia that is reflected to the motor is determined by the square root of the gear ratio.

If it were desirable to have different gain sets for varying load conditions, the SuprMotr provides several methods by which this can be easily implemented. One method would be to define a series of macro commands, each of which sets up one of the desired gain combinations. For instance, **MD1,DP80,DI30,DD70(rtn)** and **MD2,DP30,DI10,DD20(rtn)** define two sets of gain combinations. Assuming that macro #1 contains the settings that are to be used when the load is heaviest, and that occurs when the motion is in the positive direction, we could do this:

MC1,MA55 555,WS, MC2,GH(rtn)

This command sequence sets the gains according to macro #1, then commands a move to position 55,555. After arriving at this position, the gains are set according to macro #2 and the position is returned to HOME.

Of course, these gain settings could be made in the same command as the motion command, but there might be several motion command lines in the program and it is much easier to edit a single macro than to search out all cases of gain changes.

(Please refer to the software manual for additional information on programming the SuprMotrV.)