KREMFORD Pty Ltd

Manual # 1009-1

HYPERDRIVE-3

An Integrated PLC Based Programmable Stepper Motor Controller

Model KF086B
User and Programming Manual

Firmware Release 1.14.0

July 28, 2014

Table of Contents

General Information 6
- Important User Information 6
- Standard Warranty 6
- Installation Precautions 6

About This Manual 7
- Audience 7

Description 8
- Features 8
- Programmed Control 8
- Control Multiple Hyperdrives 9
- Acceleration/Deceleration Profiles 9
- Input and Output Options 9

Hyperdrive Characteristics 10
  - Absolute Maximum Ratings 10
  - Typical Motor Characteristics 10
  - Terminal Program Characteristics 10
- Communication Connector 11
- Power and Stepper Motor Connector 11
- Input/Output Connector 12
- Original Hyperdrive Compatibility 12

Installation Instructions 13
- Power Supply 13
- Motor Connections 13
- RS422 Serial Connection 15
- RS485 Serial Connection 17
- Environment 17
- Input Output Pins 18
  - Input Connections 18
  - Using a Quadrature Encoder 20
  - Using Inputs for Speed Control 21
  - Output Connections 21

Terminal Control 24
- Data Rate 24
- Installing the TeraTerm Program 24
- Setup for the Serial Port 24

Commands 26
- Interactive State Commands 26
- Communication Commands 27
- Programming Instructions 27

Getting Started 32

Interactive Use 32
RPM, Steps/Second, and Radians/Second Conversions..36
Setting the Stopped Power 36
External Motor Control..37
Internal Program Control 37
Writing a Program 37
Changing Program Flow 39
Dynamic Changes in a Running Program 40
Using a Quadrature Encoder 41
Using an Encoder for Speed Control...41
Simulating a Step/Direction Interface 42
Saving the Program...42
Auto Restore and Run 43
Using an External Text Editor...44
Downloading Programs 44
Program Errors during a Download 46
A Programming Environment 46

Instruction Descriptions 48

Accel (AD) 48
Angle (SA)...48
BakTrak (BT) 49
Baud 50
Clear (CL)..50
ClrAddr (HX) 51
ContWhen (IC) 51
DcyMode (DM) 52
Debounce (DB)...52
DecelNow (DC)...53
Delay (DL)...53
Dir (DR) 54
ELoop (EL)...54
End (EN) 55
ESC 55
Exit (EX)...55
Go (GO) 56
GoTo (GT) 56
Halt (Halt) 57
JumpIf (IJ) 57
List (LS) 58
Load (LD) 59
Loop (LP)..59
Mode (MD)...60
Monitor (IM) 61
MonSkip (IA)..64

NetClose (HC) 65
NetOpen (HO) 65
Out (IO) 65
Power (PW)....................67
Program (PG) 68
PSteps (PS) 68
Renum (RN)....................69
Reset (RS) 69
RPMCtrl (SC)....................70
RS485....................70
RunUp (AR) 71
Save (Save)....................71
SetAddr (HA) 72
SetRPM (MS)....................72
State (GS) 73
Step (ST)....................74
StepMode (SM) 74
StopPwr (SP) 74
WaitFor (WT) 75

Connecting Hyperdrives in a Network 76

Setting an RS422/RS485 Address 76
Commanding Individual Hyperdrives 76
Checking the Motor State while Networked 77
Step Mode....................77
Speed Mode....................77
Loading a Program in Addressed Mode 78

Firmware Update.................... 79

Upgrader Installation....................79
A Firmware Update 79

Messages.................... 82

Appendix 1 – Installing the TeraTerm Terminal Program 83

Appendix 2 – Installing the Notepad++ Program 88

Configure Notepad++ for the Hyperdrive Programming Language....................89

Appendix 3 – Changing the Communications Mode 91

Re-Boot Procedure....................91
Notes....................91

Table of Figures

Figure 1. Connector blocks with nomenclature. 11

Figure 2. Power supply connections. 13

Figure 3. Bipolar stepper motor connections. 14

Figure 4. Correct connection for a Unipolar motor that will develop rated torque at speed. .. 14

Figure 5. An 8-lead stepper motor connected for bipolar use. 15

Figure 6. RS422 connector and cable supplied with the Hyperdrive. 15

Figure 7. The RS422 terminating link. 16

Figure 8. This shows the internal circuitry of the Hyperdrive RS422 interface. It is important to ensure the external connections match this figure. 16

Figure 9. Suggested connections to drive the 4-wire RS422 interface with a 2-wire RS485 interface. 17

Figure 10. The simplified Hyperdrive input circuit. 18

Figure 11. Using an external power source for input power. 19

Figure 12. Using the internal 5 volt supply to power an input sensor. 19

Figure 13. Typical wiring for an electronic switching sensor. 20

Figure 14. The quadrature output waveforms from a two-switch rotary mechanical encoder. 20

Figure 15. Quadrature encoder connected to the Hyperdrive to provide both event and direction input. 21

Figure 16. Typical Hyperdrive output circuit. 22

Figure 17. Connection for using an LED to indicate program events. 22

Figure 18. Typical power relay or contactor drive circuit. 23

Figure 19. Device Manager window with the Ports entry expanded. 25

Figure 20. The Hyperdrive power up banner. 32

Figure 21. The Kremford supplied TeraTerm file download macro. 44

Figure 22. Selecting the Load Macro file transfer option. 45

Figure 23. A program as displayed in Notepad++ ready for download to the Hyperdrive. 45

Figure 24. Running both a text editor (Notepad++) and a terminal program (TeraTerm) in a PC based programming session. 47

Figure 25. The H1 firmware upgrade link shown in its normal operating position. Remove and replace shorting both pins to enable a firmware upgrade. Remove and replace like this when the upgrade is complete. 80

Figure 26. Initial HyperUpdate screen. 80

Figure 27. Hyper3Update ready to execute a firmware update. 81

Figure 28. The Hyper3Update program showing an update in progress. 81

Figure 29. TeraTerm Welcome screen. 83

Figure 30. TeraTerm License Acceptance dialog. 83

Figure 31. TeraTerm installation location. 84

Figure 32. TeraTerm optional accessories and programs. 84

Figure 33. TeraTerm program links. 85
Figure 34. TeraTerm Serial port setup dialog... 86
Figure 35. Setting TeraTerm for local echo mode. 86
Figure 36. Accessing the Notepad++ Application Data section of Documents and Settings for the Current User. 90
Figure 37. Default contents of the Hyperdrive++ Application Data directory. 90
Figure 38. The Kremford userDefineLang.xml file copied from the CDROM to the Notepad++ Application data directory. 90

General Information

Important User Information

The products and application data described in this manual are useful in a wide variety of different applications. However, the user and others responsible for applying these product described herein are responsible for determining their acceptability for each application. While every effort has been made to provide accurate information within this manual, Kremford Pty Ltd assumes no responsibility for the application or the completeness of the information contained herein.

UNDER NO CIRCUMSTANCES WILL KREMFORD PTY LTD BE RESPONSIBLE OR LIABLE FOR ANY DAMAGES OR LOSSES, INCLUDING INDIRECT OR CONSEQUENTIAL DAMAGES OR LOSSES, ARISING FROM THE USE OF ANY INFORMATION CONTAINED WITHIN THIS MANUAL, OR THE USE OF ANY PRODUCTS OR SERVICES REFERENCED HEREIN.

No patent liability is assumed by Kremford Pty Ltd with respect to the use of information, circuits, equipment, or software described in this manual.

The information contained within this manual is subject to change without notice.

Standard Warranty

Kremford Pty Ltd warrants that all equipment manufactured by it will be free from defects, under normal use, in materials and workmanship for a period of [1] year. Within this warranty period, Kremford Pty Ltd shall, at its option, repair or replace, free of charge, any equipment covered by this warranty which is returned, shipping charges prepaid, within one year from date of invoice, and which upon examination proves to be defective in material or workmanship and not caused by accident, misuse, neglect, alteration, improper installation or improper testing.

The provisions of the "Standard Warranty" are the sole obligations of Kremford Pty Ltd and exclude all other warranties expressed or implied. In no event shall Kremford Pty Ltd be liable for incidental or consequential damages or for delay in performance of this warranty.

Installation Precautions

The Hyperdrive maximum input voltage is 32 VDC. Extreme care must be taken when connecting this voltage that the power is off while making the connection and that the correct polarity as described on the label is used.

If the installation also includes connecting mains voltage to the external power supply, only a qualified electrician should carry out this work.

About This Manual

Audience

This manual explains in detail the set-up, installation and operation of the KF086 Hyperdrive3™ programmable and interactive stepper motor controller as manufactured by Kremford Pty Ltd.

It is written for the engineer responsible for incorporating the Hyperdrive3 into a design, as well as the engineer or technician responsible for its actual installation.

In this manual, the name Hyperdrive should be read as Hyperdrive3.

Description

The Hyperdrive is a programmable stepper motor controller. It combines in the one device a Programmable Logic Controller (PLC) and a Stepper Motor Controller (SMC). The built-in PLC provides a set of program commands and instructions a user can use to define and program the parameters of a particular motor movement profile. Four inputs can be used to monitor events occurring during a profile and alter the profile as required. Two outputs can be used to affect external events based on the profile. A motor move profile may consist of an acceleration phase, a constant speed phase, and a deceleration phase, where the entire motor move profile is constrained to take a specified number of steps, or the profile can be a continuous set of speed settings with various accelerations between settings.

Features

- Programmable, microstepping digital stepping motor controller.
- Operates from 12 volts to 32 VDC.
- Drives bipolar stepper motors up to 3.5 amperes peak.
- Acceleration/Deceleration specified in RPM/Sec^2.
- Programmable speed expressed in RPM.
- Movement is specified as either a number of steps or a specific speed.
- Includes an RS422 serial interface for interactive and programming control.
- Four power or torque settings.
- Four microstepping settings: 1, 2, 8, 16.
- Four decay mode settings.
- Fully programmable using a simple programming language (SPL).
- The current program may be saved in non-volatile storage.
- A stored program may be automatically loaded from non-volatile storage and executed at power up.

Programmed Control

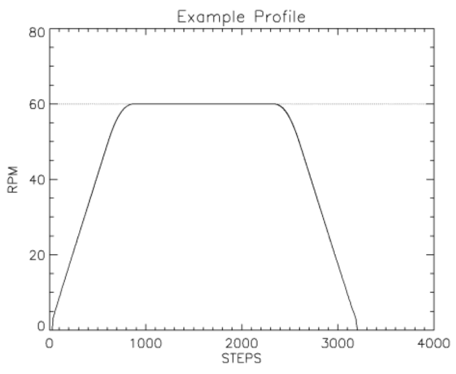
The Hyperdrive is controlled over an RS422 differential serial interface using a set of instructions and parameters that define the movement profile such as the speed, the number of steps and the acceleration. It may also be used to change the various modes the controller is capable of. Control can be from a terminal program on a PC (such as Telnet, TeraTerm, etc) connected to the Hyperdrive via a USB/RS422 or USB/RS485 interface, or directly from an external user written control program in either a PC or PLC via the RS422 link.

A particular set of motor instructions (a program) can be entered, edited and executed entirely from within the Hyperdrive. Alternately the program can be prepared in an external editor and downloaded to the Hyperdrive as required. The program can be dynamic (i.e. lost when power is removed), or stored in on-board non-volatile memory. In the latter case the Hyperdrive can also be programmed to automatically start the stored program on power up. This means that once a program has been written and qualified for a particular application, the Hyperdrive will run that program every time the machine is powered up.

Control Multiple Hyperdrives

The RS422 interface allows up to nine Hyperdrives to be connected together, with each assigned a unique address and responding to commands addressed to that particular unit only. Thus a single PLC can control several motors.

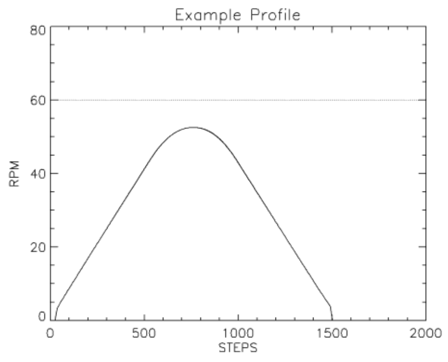
Acceleration/Deceleration Profiles



This figure shows a typical profile for a 1.8° step motor being driven at 16 microsteps per step (0.1125°), moving 3200 steps (one revolution) with a maximum speed of 60 RPM and with acceleration and deceleration times of 300 milliseconds.

Depending on the specified number of steps and acceleration, the programmed constant speed phase may not be attainable in that number of steps, in which case the acceleration phase smoothly moves from the acceleration to the deceleration phase.

The next figure shows the same motor settings but with the number of steps set to 1500. Considering the accel/decel settings along with this constraint means that the motor is unable to reach the maximum set speed of 60 RPM, but moves smoothly from acceleration to deceleration.



Input and Output Options

There are four input lines and two output lines that may be configured under program control to either control the motor profile in the case of inputs, or provide outputs that relate to points in the set motor profile. An action can be programmed to occur on a particular input being turned on or off - an input event. The action can be to pause the program waiting on that event, or branch to some other part of the program on that event. The resulting action could initiate a deceleration event, stop the motor instantly, or take some other action entirely.

An output can be programmed to turn on or off at any point within the program or after a specified number of steps.

Two of the inputs may be configured to work with a standard encoder to provide an adjustable speed control. The speed control steps may be specified as fractions of an RPM.

Hyperdrive Characteristics

Absolute Maximum Ratings.

Supply voltage	32 volts, DC
Motor peak current	3.5 amps peak
Ambient temperature	-10°C to 55°C

Typical Motor Characteristics

Operating voltage	12 – 28 volts, DC
Motor type	Bipolar two phase.
Motor power settings and corresponding maximum peak currents.	20% 0.7 amp 50% 1.75 amp 75% 2.625 amp 100% 3.5 amp
Microsteps (1:1 is the motor basic step angle)	1:1, 1:2, 1:8, 1:16
Decay mode settings	0% (none), 25% decay, 50% decay, 100% decay
Step angle	Set to match motor.
Acceleration	Motor/Load inertia dependant. Programmable. Mmaximum 30,000 RPM/Sec^2 Minimum 30 RPM/Sec^2.
Control input voltages	Minimum 5 volts, maximum 24 volts.
Control maximum output current	30 mA
Control maximum output voltage	60 volts
Maximum internal program steps	100
Maximum speeds at the four microstep settings (motor dependant).	Setting / Max Speed: 1 — 9000 2 — 6000 8 — 3000 16 — 1500

Terminal Program Characteristics

Baud rate	Selectable, 9600 or 57600
Data bits	8
Stop bits	1
Parity	None

Communication Connector

The communication connector is a 5-pin JST B5B-ZR top-entry shrouded header. The matching ZHR-5 connector with a pre-built 500mm flat cable is supplied. The connection is specified for 4-wire differential RS422 serial control plus ground. These connections are named from the point of view of the Hyperdrive.

Pin	Colour	Function
1	brown	RXD+
2	red	RXD-
3	orange	TXD-
4	yellow	TXD+
5	green	Ground

Thus when connecting an RS422 sender that was similarly marked from its point of view, the sender’s TX lines would be connected to the Hyperdrive RX lines and the sender’s RX lines to the Hyperdrive’s TX lines.

Power and Stepper Motor Connector

A six (6) position screw terminal block provides for the four motor field wires and the power supply.

Pin	Function
1	Motor field B1
2	Motor field B2
3	Motor power positive, 12 to 24V.
4	Motor power negative (GRND)
5	Motor field A2
6	Motor field A1

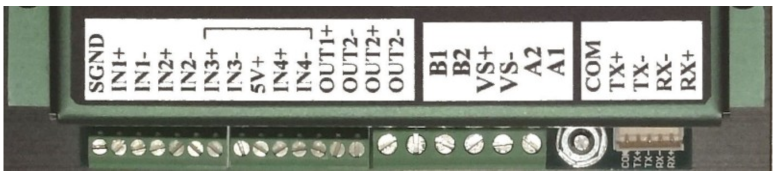


Figure 1. Connector blocks with nomenclature.

Note there is no reverse polarity protection built into the Hyperdrive. Check and double-check the power supply polarity as shown in Figure 1 before connection.

Input/Output Connector

Two seven (7) position screw terminal blocks provide for input and output control.

Pin	Function
1	Signal ground
2	IN 1 +
3	IN 1 -
4	IN 2 +
5	IN 2 -
6	IN 3 +
7	IN 3 -
8	+5 Volts
9	IN 4 +
10	IN 4 -
11	OUT 1 +
12	OUT 1 -
13	OUT 2 +
14	OUT 2 -

All input and output terminals are optically isolated for installations where the possibility of ground loops or interference are hazards.

Where isolation is not such a worry, the signal ground (SGND) and positive five volt (5V+) terminals provide a power source for some common input devices, including simple switches. The SGND terminal is only indirectly connected to the power supply ground .

Original Hyperdrive Compatibility

All previous Hyperdrive programs are compatible with the Hyperdrive3 language except for the output instruction. Originally terminal IO-4 was the single output port. In addition the original output port was ground referenced, with an **L** indicating the output would be at a low or near ground potential. The instruction syntax was:

```
IO 4, <either L or H>
```

where **L** was ON and **H** was OFF. There are now two output ports and they are named OUT1 and OUT2. However the Hyperdrive3 outputs are optically isolated (see Figure 16), and a more realistic nomenclature is **on** or **off**.

Therefore all original programs with output instructions like `IO 4,x` will need to be altered to the new syntax of:

```
Out 1, <ON or OFF>
```

or

```
Out 2, <ON or OFF>
```

Installation Instructions

Power Supply

The absolute maximum supply voltage is listed in *Absolute Maximum Ratings*. The stepper motor is not enabled until the supply voltage reaches 10 volts, and is automatically disabled if the supply voltage falls below 9 volts.

The typical range for a Hyperdrive supply voltage is from 12 to 24 volts.

The voltage source must be capable of supplying at least the peak currents as specified by the **Power** command. A typical value is about 150% of the **Power** setting. For example, a **Power** setting of 4 sets a peak current of 3.5 amps, so the recommend power supply rating would be 5 amps.

Whether the current will reach that value depends on the supply voltage, the motor winding resistance and inductance, the motor load, and the motor speed.

Note that if you are using a big motor working at high power and the operation profile includes rapid deceleration events from high to low speeds, the excess motor energy is transferred back to the power supply during the deceleration event. The power supply must be rated to both withstand and absorb this energy.

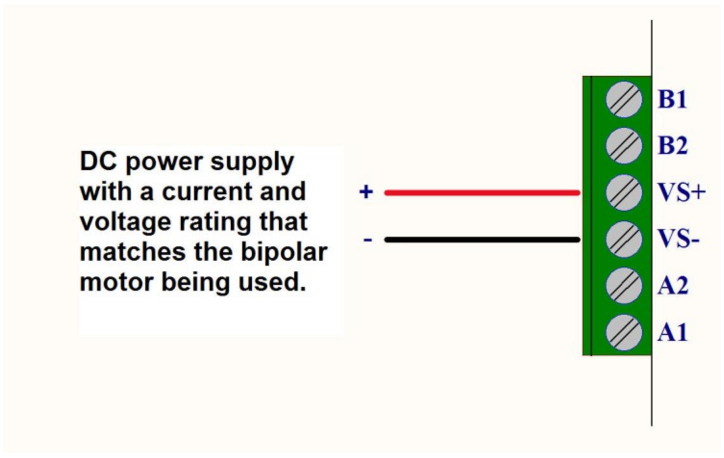

DC power supply with a current and voltage rating that matches the bipolar motor being used.
+
-
B1
B2
VS+
VS-
A2
A1

Figure 2. Power supply connections.

The DC power supply is connected to the screw terminals marked VS+ and VS- (see Figure 2). The VS- can connect to a grounded wire, but it is not connected to the Hyperdrive frame. The PCB mounting between the RS422 and power/motor connectors can be used to ground the Hyperdrive frame if required.

The voltage range is typically from 12 to 24 volts. Ensure the positive wire goes to the VS+ terminal.

The controller has no internal reverse polarity protection. Connecting a reverse polarity supply will almost certainly damage the Hyperdrive.

Motor Connections

The Hyperdrive can control bipolar two phase motors. The motor will therefore have four leads. Connect the motor using the following table and Figure 3.

Terminal	Motor
B1	Field 1 wire 1
B2	Field 1 wire 2
A1	Field 2 wire 1
A2	Field 2 wire 2

Use a multimeter or the manufacturer's documentation to select the two lead pairs associated with each field.

Keep these leads as short as possible. Ensure the cable used is rated for at least 10 Amps to minimise voltage drop. If possible use 4-wire shielded cable with the shield connected to a frame ground at the Hyperdrive end (NOT the SGND terminal).

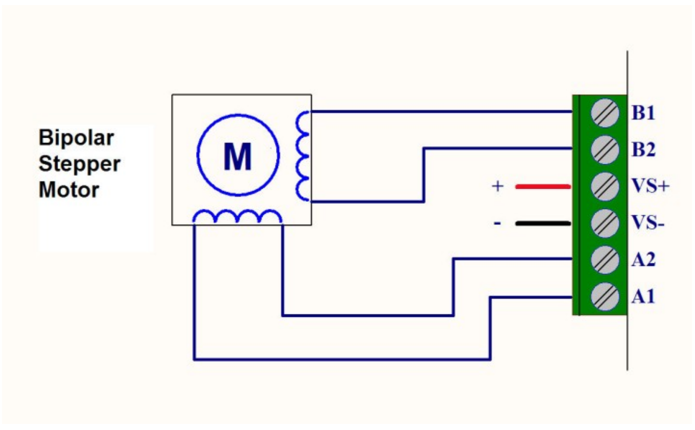


Figure 3. Bipolar stepper motor connections.

Once connected and operating, should the direction the motor turns be opposite to that specified by a **Dir** command, simply swap either the two B wires or the two A wires.

The Hyperdrive is a bipolar motor driver, but other types of stepper motors can be configured to run as bipolar. A unipolar stepper with six leads can be simply connected as bipolar by leaving the field centre-tap wires disconnected. However this will give a field coil inductance of four times that of a single coil, a definite problem if the motor is required to run at any speed and develop its rated torque. The correct connection for a six lead unipolar motor is to leave the two opposite field coils unconnected as shown in Figure 4.

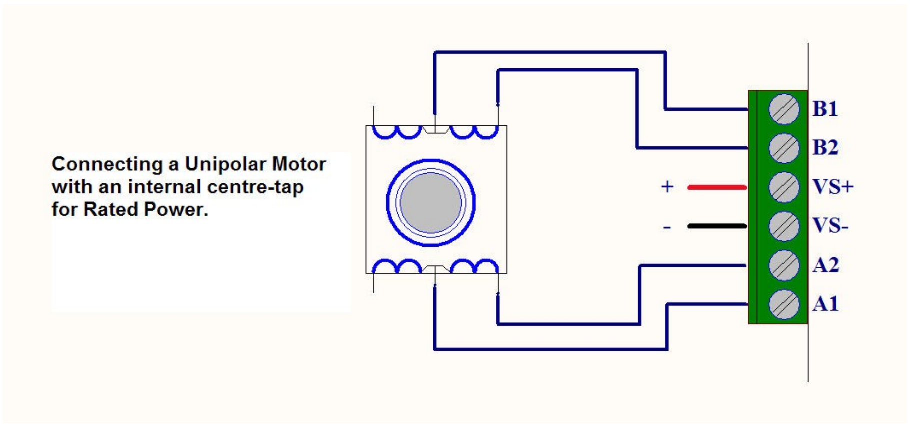

Connecting a Unipolar Motor with an internal centre-tap for Rated Power.
+
-
B1
B2
VS+
VS-
A2
A1

Figure 4. Correct connection for a Unipolar motor that will develop rated torque at speed.

An eight lead stepper motor can have the two sets of field coils connected in parallel as long as the two coil pairs polarity match. Often the latter is indicated by the eight leads just using four basic colours, but matching coils use different lead markings, such as red and red/black stripes. This connection configuration is shown in Figure 5.

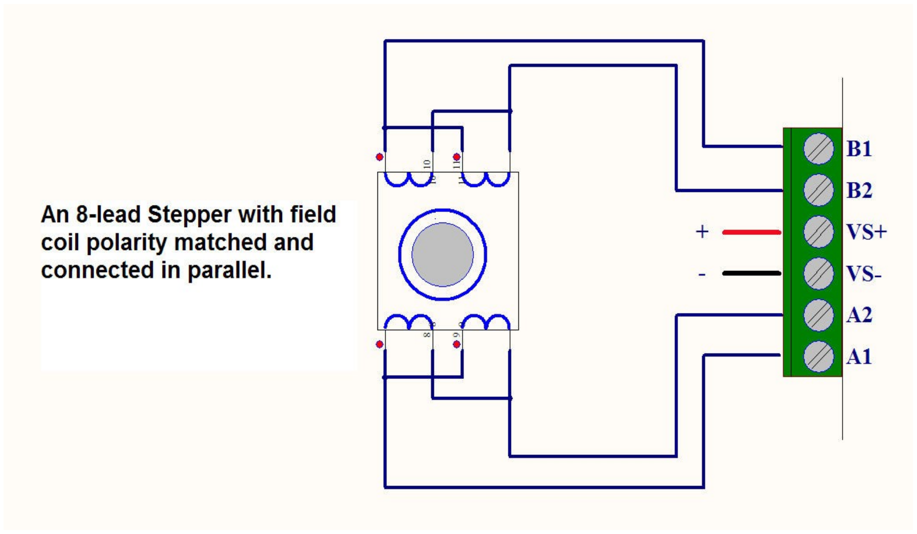
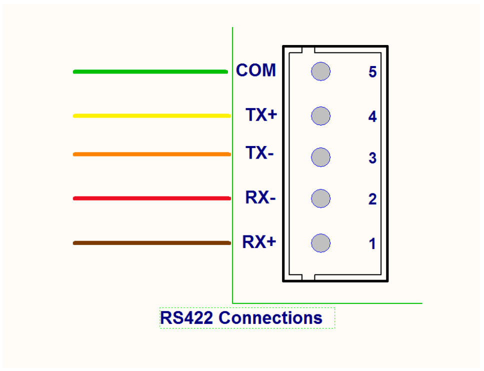

An 8-lead Stepper with field coil polarity matched and connected in parallel.
+
-
B1
B2
VS+
VS-
A2
A1

Figure 5. An 8-lead stepper motor connected for bipolar use.

Kremford Pty Ltd can offer advice with connecting stepper motors with unusual lead configurations to the Hyperdrive.

RS422 Serial Connection



An RS422 serial connection provides a programming and control connection to a PC or PLC. The communication cable can be up to 300 metres long. Standard computer Category 5 cable is satisfactory in most cases. Some installations may require a flexible or shielded cable or both.

If necessary, several Hyperdrive controllers may be connected to the one controlling computer (see *Connecting Hyperdrives in a Network*).

The Hyperdrive label names the five pins in the communications connector. Shown here are the terminal connections for the 4-wire RS422 system. The connection can be from a computer to one Hyperdrive, or to several Hyperdrives in a multi-drop network. The COM (green) wire is typically a ground shield.

Packaged with each Hyperdrive is a matching plug and 500mm lead for the RS422 socket (see Figure 6). This can be used to terminate a standard 4-wire USB/RS422 adapter.

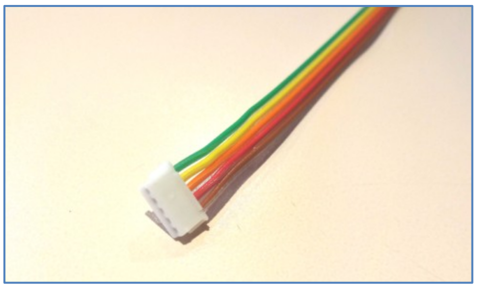


Figure 6. RS422 connector and cable supplied with the Hyperdrive.

Warning. Do not connect the RS422 COM screw terminal to the SGND screw terminal. This could give rise to the possibility of ground loops.

Each Hyperdrive includes the standard 120 ohm RS422/RS485 receiver terminating resistor. Figure 7 shows the terminating link on the Hyperdrive printed circuit board. The link is marked **RX** and consists of two pins and a shorting link. Hyperdrives are shipped with this link connected as shown in the figure. If the Hyperdrive is part of a multi-drop RS422/RS485 network the link should be removed from all units except the one at the opposite end of the network from the driving computer.

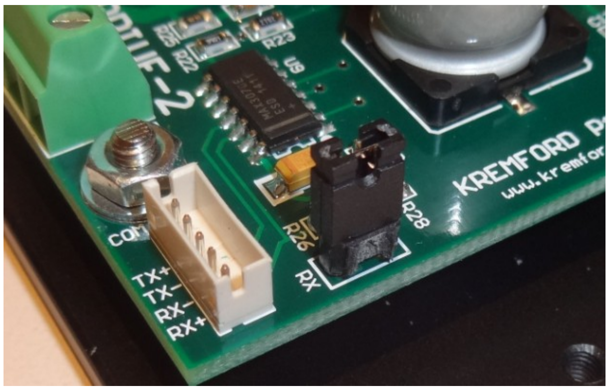


Figure 7. The RS422 terminating link.

The TX+ and TX- names refer to the Hyperdrive RS422 transmitter, while the RX+ and RX- names refer to its receiver. The diagram in Figure 8 shows the internal connections.

It is important to note that the connection names on the label are from the Hyperdrive's point of view.

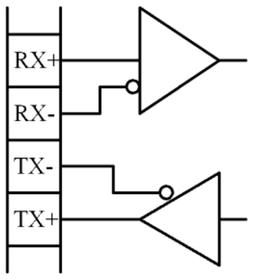


Figure 8. This shows the internal circuitry of the Hyperdrive RS422 interface. It is important to ensure the external connections match this figure.

You must confirm whether the external network cabling is labelled from either the network driver or master's point of view, or else from the point of view of the various devices (slaves) on the network. Where the labelling is from the network driver's point of view, then the network TX+/TX- connections must go to the Hyperdrive RX+/RX- connectors, and the network RX+/RX- wires to the Hyperdrive TX+/TX- connectors.

RS485 Serial Connection

The Hyperdrive is shipped with the serial interface configured as an RS422 4-wire configuration. However it can be reconfigured as an RS485 2-wire configuration if required, although Kremford makes no guarantee that the proposed connection described here will work with all RS485 drivers.

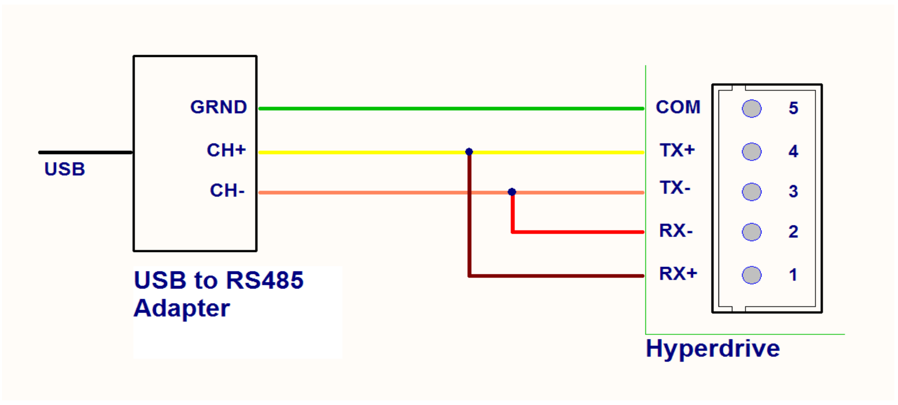


Figure 9. Suggested connections to drive the 4-wire RS422 interface with a 2-wire RS485 interface.

The **RS485** command can be used to switch between these two serial modes. The command:

```
>RS485 1
```

will switch from RS422 to an RS485 interface, while the command:

```
>RS485 0
```

will switch from RS485 to RS422. After giving this command the Hyperdrive will be in the new communications mode. Switch it off, reconnect with the new interface device, and turn it on again. It will start up in the new mode.

Note that this assumes you are currently communicating with the Hyperdrive in the *from* mode to be able to give these commands. If the Hyperdrive is in its default RS422 mode and you only have an RS485 interface (or vice versa), you will need to use the reboot method to make the switch.

The necessary procedure to make this switch from RS422 to RS485 or from RS485 to RS422 is described in *Appendix 3 – Changing the Communications Mode*.

Environment

The Hyperdrive is rated for industrial usage as stated in *Absolute Maximum Ratings*. It should be mounted in a dust and moisture free area. Temperature rise should not be a problem except where the unit runs continuously at Power setting 4, and in this case there should be adequate ventilation provided. If possible, mount the unit on a metal frame. Ensure the heatsink fins are not enclosed. The best mounting is on a vertical metal surface with the heatsink on the top side.

The Hyperdrive will shut itself down automatically should the controller internal chip temperature approach its maximum operating temperature rating, typically 150° C. Note that this will not protect the unit from a rapid temperature rise associated with abnormal operation.

If you use **Power** setting 4 with **StepMode** setting 1 and a low speed, ensure the speed chosen does not include the motor's self-resonant speed, obtainable from the manufacturer's speed/torque curve.

If temperature rise may be a problem, monitor the heatsink temperature. You should choose parameters that limit this temperature to 65°C maximum.

The unit is not insect or vermin proof. This should be kept in mind during installation.

Input Output Pins

There are six optically-isolated inputs and outputs that can be controlled by a running program. For the KF086B-6 version of the Hyperdrive these are configured as four inputs and two outputs. The four input terminals are marked as IN1, IN2, IN3, and IN4. The two output terminals are marked OUT1 and OUT2 (see Figure 1).

Each input terminal connects to the input of a constant current source. This provides a constant drive to the opto-isolator throughout the allowed input voltage range. Increasing the voltage between the IN_x+ and IN_x- terminals above 5 volts will produce a HIGH input (see the later programming definitions).

The 5V+ terminal referenced to the SGND terminal can be used to power external sensors where applicable. The maximum current draw should not exceed 50 milliamps.

Input Connections

A simplified drawing of the internal circuit for each of the four input pins is shown in Figure 10.

An input is turned **ON** by connecting an INx+ input to a positive voltage between 5 and 24 volts, and the INx- input to the voltage source negative return. The programming language sometimes calls this a **HIGH** or an **ENABLED** or a **TRUE** input. An external mechanical or electronic switch could make this connection.

The input is turned **OFF** by disconnecting power from the input. The programming language sometimes calls this a **LOW** or a **DISABLED** or a **FALSE** input.

The switch can be in either the positive (INx+) or negative (INx-) lead.

The Hyperdrive input presents to the external source a constant current load of about 7 milliamps throughout the allowable input voltage range of 0 to 24 volts.

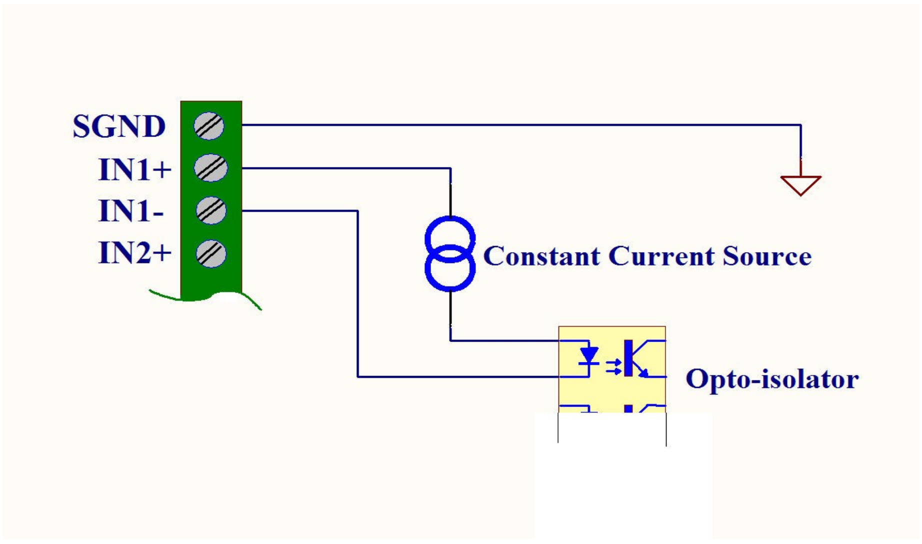


Figure 10. The simplified Hyperdrive input circuit.

The programming language includes an instruction for setting a debounce delay to provide adequate debounce times for external switches. This can be zero for electronic inputs and up to 50 mSecs for worn mechanical switches. This sets the time from when the Hyperdrive first senses a level change until it acts upon that change. Only if the input is in the same state at the end of the debounce delay as it was at the start will the input state be considered TRUE.

The circuit shown in Figure 11 is recommended for situations where the sensor is a simple mechanical switch such as a limit or positioning switch with an external power supply.

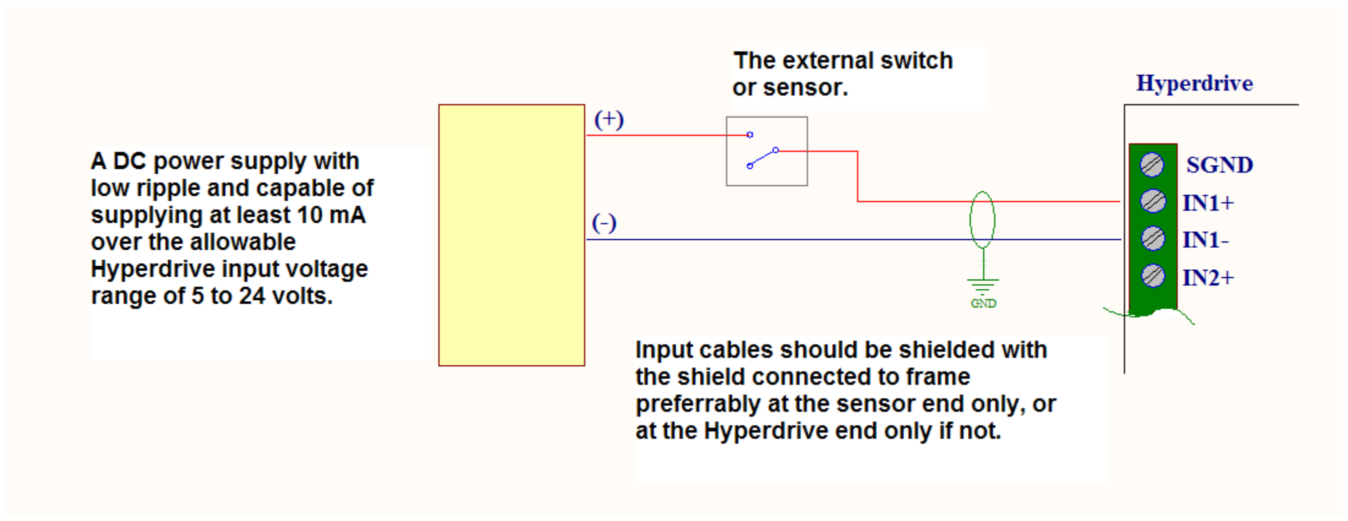

The external switch or sensor.
Hyperdrive
A DC power supply with low ripple and capable of supplying at least 10 mA over the allowable Hyperdrive input voltage range of 5 to 24 volts.
(+)
(-)
SGND
IN1+
IN1-
IN2+
GND
Input cables should be shielded with the shield connected to frame preferrably at the sensor end only, or at the Hyperdrive end only if not.

Figure 11. Using an external power source for input power.

In simple situations, the Hyperdrive internal 5 volt power supply can be used as the switch power source. A typical circuit is shown in Figure 12. Here the switch is shown in the SGND/IN1- line, with the power coming from the link between the 5V+ and IN1+ terminals. The opposite configuration would also be suitable, i.e. the switch in the positive line. Just make sure the configuration cannot connect the 5V+ terminal to the SGND terminal under any circumstances.

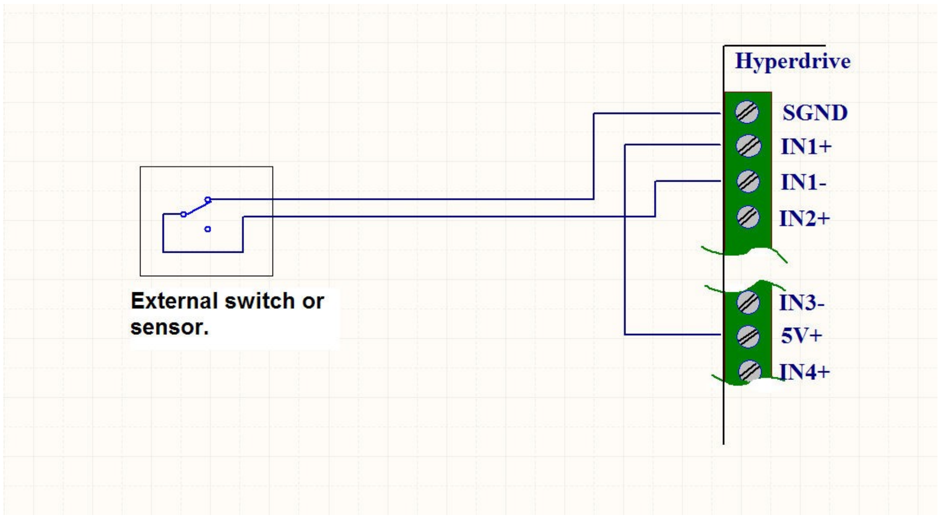


Figure 12. Using the internal 5 volt supply to power an input sensor.

In cases such as this, the recommended wiring cable is a shielded, multi-stranded twisted pair, with the shield connected to frame only at the Hyperdrive end. This gives the best protection against interference from external electro-magnetic fields from adjacent motors, solenoids, etc.

Many modern sensors have their own power supply and include an electronic switch to indicate their state. Figure 13 shows the typical wiring diagram for this type of sensor. Any shield should be connected only at the Hyperdrive SGND terminal.

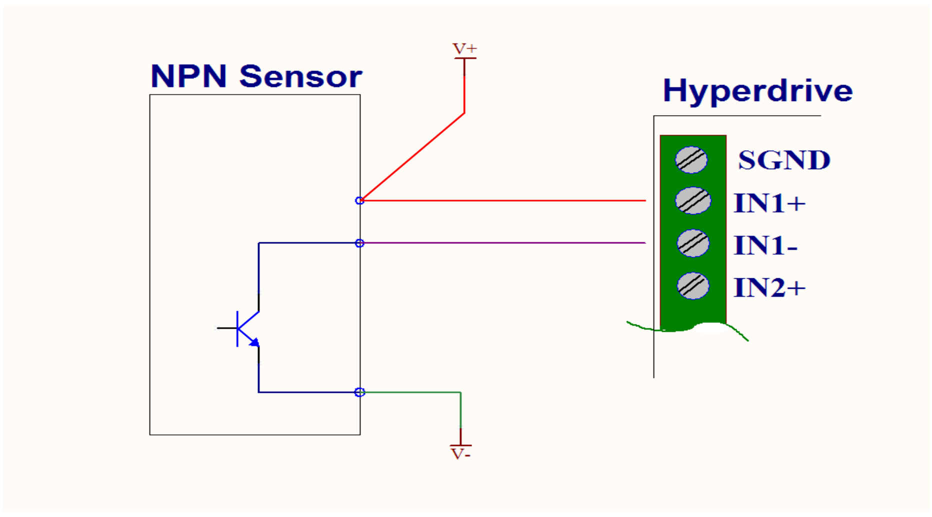


Figure 13. Typical wiring for an electronic switching sensor.

Using a Quadrature Encoder

A rotary quadrature encoder has one common pin and two output pins, typically labelled A and B. Mechanical versions have two switches inside operated by the encoder shaft that switch in quadrature to each other, with an indent mechanism that typically provides twelve or so positions per rotation. Figure 14 shows the normal output from this type of encoder.

The Hyperdrive has a number of instructions that provide for monitoring an input state and changing the program flow when a change occurs. Considering Figure 14 again, if a program is continuously waiting for Channel A to change state from low to high, and will jump when that occurs to another instruction that checks the state of Channel B, the result is that the program now knows both that an encoder click event occurred, and the direction the encoder was rotated. The program can then execute some action based on that knowledge, and return to monitoring Channel A when it is complete.

A program example is given later that uses this technique to rotate a motor a given number of degrees in the correct direction for every click of the encoder – in effect the motor will mimic the encoder rotation.

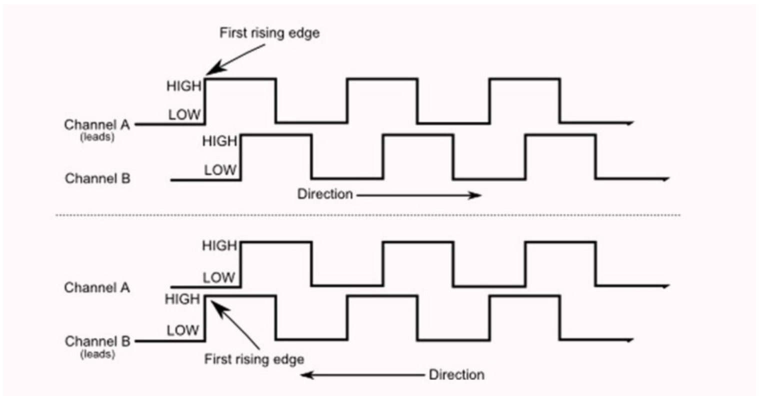


Figure 14. The quadrature output waveforms from a two-switch rotary mechanical encoder.

The typical wiring diagram for using a quadrature encoder is shown in Figure 15. While the A and B lines could be connected to any combination of the four inputs, IN3 and IN4 have a special relationship with encoders (see next section), so Figure 15 is the recommended encoder connection.

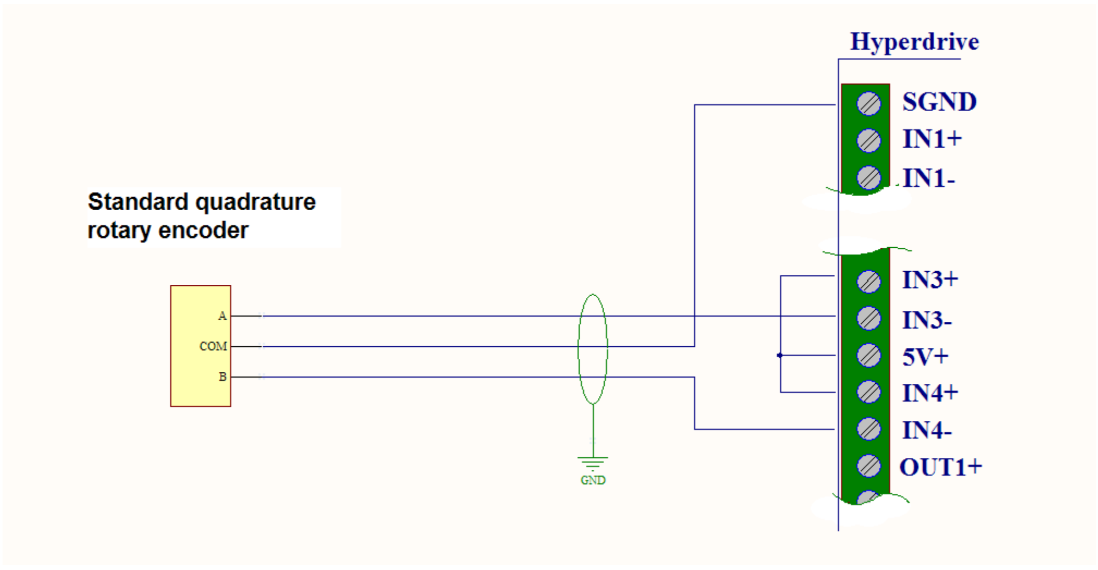


Figure 15. Quadrature encoder connected to the Hyperdrive to provide both event and direction input.

Using Inputs for Speed Control

The **RPMCtrl** (see *RPMCtrl (SC)*) command modifies the way the inputs IN3x and IN4x respond. This does not affect the operation of the other two inputs which remain as described earlier. After receiving this command, these two inputs are configured as high speed quadrature inputs. Each click of the encoder clockwise or counter clockwise will respectively increase or decrease the current motor speed.

Note carefully that if the Hyperdrive is in **SPEED** mode, a click in the increase speed direction is the same as entering a **SetRPM** command with the current **RPMCtrl** increment speed.

This means that, even if currently stopped, the motor will start running at that speed.

The **RPMCtrl** instruction requires a parameter that specifies a speed change increment in RPM. This parameter can be a floating point number and so can provide for very fine speed control.

The common terminal can be either the positive or negative voltage source. The recommended Figure 15 circuit shows using the internal Hyperdrive 5V power supply with the common terminal connected to the machine ground terminal.

If the distance to the encoder is much further than about 100 mm, the wiring should be shielded with the shield connected to the SGND as shown.

Output Connections

Figure 16 shows the Hyperdrive output circuit. The two output sources are PC357N4TJ00F opto-isolaters. At the OUTx terminals, this consists of an NPN transistor rated at $80V_{CEmax}$

and 50mA I_{Cmax}. A 1 amp, 60V schottky diode is connected between the collector and emitter of the transistor for protection. Any inductive load such as a relay connected between the OUTx+ terminal and a positive supply should have a reverse biased snubber diode across the load.

Note that the only protection provided is the normally reverse biased diode across the collector/emitter. There is no protection for over-current. This allows for the most versatile user configurations, but requires careful attention to the design of the attached circuitry.

Conservative design would limit the maximum voltage of the output circuit to 50 volts and current to 30 mA.

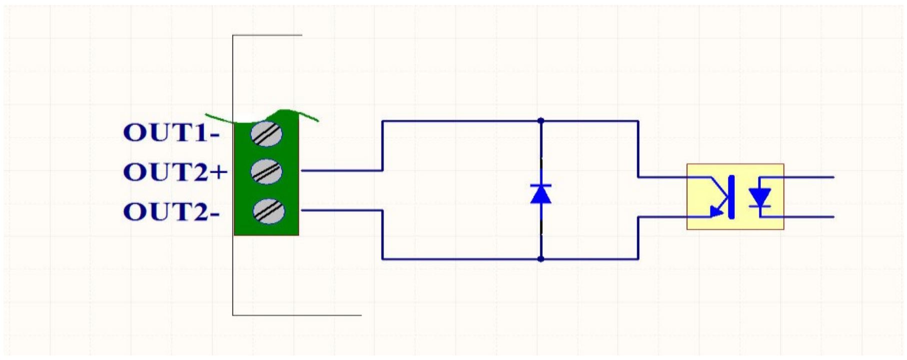


Figure 16. Typical Hyperdrive output circuit.

Often when designing a new installation it is handy to be able to monitor the state of external switches or the state of the Hyperdrive program. When only an indication is required, use an LED (Light Emitting Diode) and resistor combination as shown in Figure 17. This uses OUT2 to control the LED. The command:

```
Out 2, ON
```

For a current through the LED of about 3 mA, a resistor value of 1000 ohms will be satisfactory. The LED and/or the resistor or both could be in either the + or – leads.

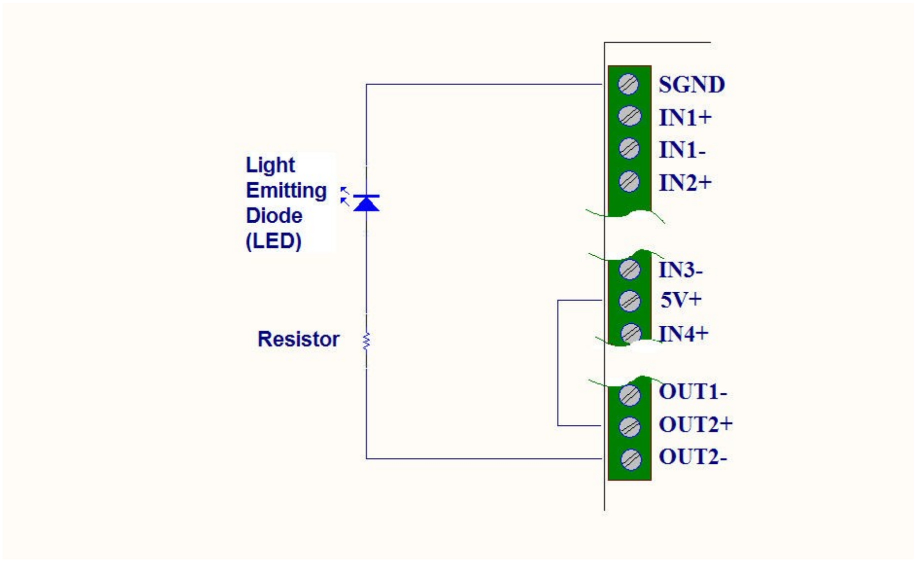


Figure 17. Connection for using an LED to indicate program events.

A solid state relay could be used with the Hyperdrive providing the switching voltage. The collector current maximum should be about 30 milliamps. This current will be sufficient to drive a small relay. However a load that requires much higher current than this could pull the output transistor out of saturation due to the fixed internal drive, and so will require amplification to drive a load such as a large relay or contactor. When driving a large relay, Figure 18 shows an example circuit driven by the Hyperdrive OUT2 output terminal.

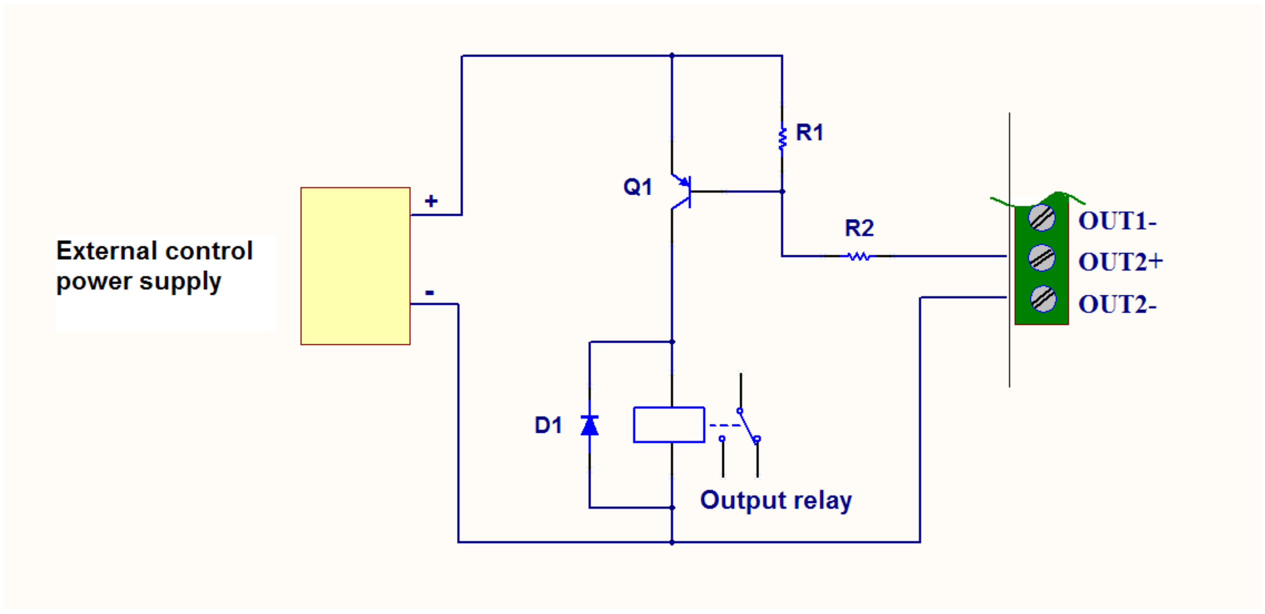


Figure 18. Typical power relay or contactor drive circuit.

The relay will close when the output is turned on by the Out 2,L instruction. The circuit uses an external PNP transistor to provide the necessary current amplification. Resistor R1 will have a typical value of 10,000 ohms. Resistor R2 will depend on the external control power supply voltage, the relay DC resistance, and the large signal gain of the transistor Q1.

As an example, we will take a standard 12 volt relay with a coil resistance of 220 ohms, a DC supply of 12 volts, and a 2N3906 as a typical PNP transistor. The coil current will be

$$I_{coil} = \frac{12}{220} = 55\ mA$$

The 2N3906 has a DC current gain h_{FE} of 60 at 50 mA. Using the typical 15% of this value to ensure the transistor saturates under all conditions, the base drive current required is

$$I_B = \frac{55}{15\%\ of\ 60} = \frac{55}{9} = 6\ mA$$

This is well within the nominal maximum Hyperdrive output current of 50 mA mentioned earlier. The value for R2 is then given by

$$R2 = \frac{V - V_{BE}Q1 - VO_{on}(IO4)}{6}$$

$$= \frac{12 - 0.7 - 0.2}{6} = 1800\ ohms$$

Terminal Control

The RS422 serial interface built into the Hyperdrive provides the means to control and program the attached motor. The interface operates in simplex mode (the Hyperdrive does not echo characters it receives) at a selectable rate of either 9600 or 57600 baud, and can be connected to a computer through an USB/RS422 or USB/RS485 adapter. It can be connected directly to a PLC that provides an RS422 4-wire, simplex serial interface or RS485 2-wire simplex interface.

Once connected to a computer, on power up the Hyperdrive terminal program will send the message:

```
Kremford Hyperdrive3
Revision x.x.x
Ready

>
```

Note the '>' character. This is called the *command prompt* and indicates the Hyperdrive is ready for a new command and in particular, has finished any previous command. When the Hyperdrive is being controlled by a computer or a PLC, the occurrence of this character can be used to indicate the operation(s) implied in a previous command are complete and that a new command can be transmitted. In computer terms it is the ACK response to the command.

Data Rate

Almost any terminal program will suffice for interactive use. However, if the option of downloading programs to the Hyperdrive is used, or if the Hyperdrive is being driven by a PLC or other computer, the chosen terminal program or controlling program will need an option for testing that the Hyperdrive command prompt character '>' has been received after each line is transmitted. This is because there is no hardware handshake control for the communications interface. After each command line is received, the Hyperdrive firmware examines the line for syntax errors, and for any errors will print a short diagnostic message. If the terminal program continues to send new lines without pausing for the '>' prompt, this message will likely be corrupted.

Where your terminal program does not support this feature, inserting a delay of about 30 mSecs after each line will ensure that syntax errors discovered during the download will have time to send a corresponding error message.

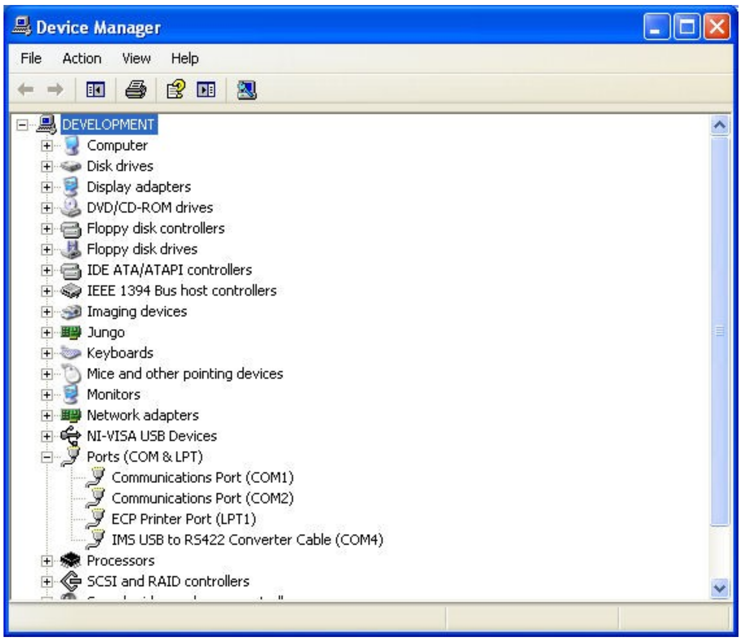
Installing the TeraTerm Program

Included on the accompanying CDROM is a copy of the public domain TeraTerm terminal emulation program. TeraTerm is a free software terminal emulation program that will run on most versions of Microsoft Windows. The complete installation procedure is detailed in *Appendix 1 – Installing the TeraTerm Terminal Program*.

Setup for the Serial Port

To setup the communication link with the Hyperdrive you will need to know the name of the communication (COM) port you have connected to the Hyperdrive. This will typically be a USB to RS422 (or USB to RS485) adapter that, when plugged into the computer, will appear as a serial port. You can see the port number Windows assigns by first clicking on *Start->Control Panel* and then double clicking on the *System* icon. In the *System Properties* dialog, click on the *Hardware* tab, and then click the *Device Manager* button. This will bring up the *Device Manager* window that will be similar to Figure 19. Click on the *Ports* entry to see the list of the serial ports on this computer.

The figure shows an IMS USB to RS422 converter connected and that it was assigned COM port number four (COM4). Now you know the port number you can dismiss this series of windows.



```
DEVELOPMENT
    Computer
    Disk drives
    Display adapters
    DVD/CD-ROM drives
    Floppy disk controllers
    Floppy disk drives
    IDE ATA/ATAPI controllers
    IEEE 1394 Bus host controllers
    Imaging devices
    Jungo
    Keyboards
    Mice and other pointing devices
    Monitors
    Network adapters
    NI-VISA USB Devices
    Ports (COM & LPT)
        Communications Port (COM1)
        Communications Port (COM2)
        ECP Printer Port (LPT1)
        IMS USB to RS422 Converter Cable (COM4)
    Processors
    SCSI and RAID controllers
```

Figure 19. Device Manager window with the Ports entry expanded.

See the *Getting Started* section for examples of using the Hyperdrive command and programming language defined in the *Instruction Descriptions* section.

Commands

The Hyperdrive provides a set of commands and instructions to supervise and drive the attached stepper motor through a movement profile. The commands fall into two groups, those associated with motor movement, and those dealing with managing the movement group.

Interactive State Commands

The interactive state commands can only be issued in interactive mode.

Name	Function	Discussion
State	Print the current state of the controller settings.	Listed are: maximum speed setting in RPM number of steps per step profile acceleration time in RPM/Sec^2 direction, CW or CCW power setting step mode setting decay mode setting the four input debounce settings speed control mode unit address stopped power setting RPM control mode number of program lines
Save	Save the current program (if any) to non-volatile storage.	Once a program is entered it can be executed immediately. However unless it is saved it will be lost when power is removed.
Program	Clear an existing program.	Typically entered before entering new instructions. The Hyperdrive switches to programming mode.
List	List the current program.	List the current program, or list the program entered so far.
Go	Start the current program.	The program will start running, executing each instruction in line number order. It will run to the end and halt unless it encounters a loop instruction, in which case it will loop based on the loop count parameter.
Load	Load the currently stored program.	Returns to active the last program saved to non-volatile storage by the Save command. Erases any current program.
RunUp	Set auto-program startup.	The program saved in non-volatile memory with the Save command will be automatically loaded and started when the Hyperdrive is powered up. NOTE: Use Ctrl-X to clear this state (ASCII “CAN” (0x18)).
Renum	Renumber a program.	Program lines are preceded by line numbers. This command will renumber the program starting with line number 100 and stepping up by 10 for each line.
Exit	Stop a running program.	Like the CtrlC keystroke. Used in a networked system to exit a running program in a particular Hyperdrive.
PSteps	Print elapsed steps.	In STEP mode, this command will print the number of steps since the last Step command.
RS485	RS485 and RS422 switch	Provides for switching between RS485 and RS422 serial communication modes.

Communication Commands

In these tables, the phrase `0<number<limit` such as `0<number<10` means a valid entry is a number greater than 0 and less then 10, so valid numbers are 1 through 9. The two-character instructions of the original Hyperdrive are included in the *Instruction Descriptions* section. You can inter-mix the new and old commands if required.

To use the Hyperdrive in a multi-drop RS422 or RS485 network that consists of several Hyperdrives, you must first set its address. Thereafter commands must be bracketed by NetOpen/NetClose commands to address that unit (see *Connecting Hyperdrives in a Network*). A move once started can be stopped immediately by sending an Escape character (ASCII 'ESC').

Name	Function	Parameter	Discussion
Baud	Sets the serial baud rate	9600 or 57600	Sets the communications baud rate. Power must be cycled before it becomes effective.
SetAddr	Set the Hyperdrive network address	0<number<10	This command sets the address for this particular Hyperdrive. The command must be given BEFORE the unit is connected into a multi-drop network.
NetOpen	Open command access to this unit.	SetAddr number	Opens this addressed unit for exclusive command access.
NetClose	Close command access to this unit.	SetAddr number	Closes this addressed unit for any further commands.
ClrAddr	Clear the ntwork address	SetAddr number	Clears the unit address. Unit subsequently responds to all commands.

Programming Instructions

The **Mode** column below includes a Y for those programming instructions that can also be given as interactive commands. Some of these instructions have parameters. The input/output instructions have fairly complex parameters and these are explained in detail in *Instruction Descriptions*.

The figure 2^{31} is actually $2^{31} - 1$ which is 2,147,483,647.

In interactive mode, all the motor setup instructions, when given as interactive commands, just store their parameters in the controller. However the set steps instruction (Step) in step mode, and the set speed instruction (SetRPM) in speed mode, will cause the motor to immediately execute a move profile using all preceding motor related instructions.

Name	Mode	Function	Parameter	Discussion
Angle	Y	Set the motor basic step angle in thousandths of a degree.	number	The default value is 1800, or 1.8°. Note that the controller sends motor commands as microsteps, and the angle the motor subsequently rotates depends on the motor's inherent step angle.

Name	Mode	Function	Parameter	Discussion
Step	Y	Set the number of steps in the current move.	$0<\text{number}<2^{31}$ or blank	In both programming and interactive modes when in step mode this instruction will cause the motor to immediately move that many steps using the currently defined set of acceleration, stepping, power, and decay modes. If no parameter is given the motor will make 2,147,483,647 steps. The stepping mode (StepMode) will affect the number of degrees a step will take.
Baktrak	N	Reverse direction for the current number of steps.	speed	Usually used in emergency situations where the motor must be reversed back to a known position in response to an input level change, typically from an emergency stop button. The parameter sets the speed.
Accel	Y	Set the acceleration rate in RPM/Sec^2.	30 thru 30000	The acceleration and deceleration profiles are both the same. The acceleration profile is linear up to about 99% of the set speed and then smoothly transitions to the set speed in both step and speed modes.
SetRPM	Y	Set the maximum speed in RPM.	See *Typical Motor Characteristics*	The speed setting for a particular step mode cannot exceed the maximums listed in the Motor Characteristics table. In step mode, whether the motor will reach this speed depends on the number of steps and the acceleration rate. Where the given speed cannot be reached in time the acceleration rate will smoothly transition to a deceleration so the move always contains exactly the given steps.
Power	Y	Set the power mode.	1 thru 4	The Hyperdrive has four power or torque settings. In effect they are current limit settings referenced to the maximum current of 3.5 Amps: 1 20% 0.7 amps 2 50% 1.75 amps 3 75% 2.625 amps 4 100% 3.5 amps These are peak current limit values and the average current will depend on the motor coil resistance, supply voltage, and speed.

Name	Mode	Function	Parameter	Discussion
StepMode	Y	Set the step mode.	1 thru 4	The four settings will divide the motor's basic step angle by these values: 1 = 1 2 = 2 3 = 8 4 = 16 For example, a 1.8° motor on setting 3 will have a step angle of 0.225° or 1600 steps per revolution. Note that setting a particular step mode may limit the maximum speed (Speed) setting to those defined in the *Motor Characteristics* table for that step mode.
DcyMode	Y	Set the decay mode.	1 thru 4	The decay mode sets how the controller manages the recirculating currents in the internal driver circuit during each "off" cycle period. The settings are 1, 2, 3 and 4, and these settings correspond to decay ratios of 0%, 25%, 50% and 100% respectively. The default setting is mode 2, and in many applications this will be sufficient.
Dir	Y	Set the motor direction.	cw or ccw	This sets the motor rotation direction (case is ignored - cw is the same as CW). cw clockwise ccw counter-clockwise
MonSkip	N	Arm a jump.	Various	Arms an eventual jump based on input state changes. State must change from current state before monitoring begins.
Monitor	N	Arm a jump.	Various	Arms an eventual jump based on input levels. The jump is not taken until the given state occurs.
ContWhen	N	Continue based on input pin state.	Various	Selects whether a program will pause or not at this instruction based on the input pin's current level.
JumpIf	N	Jump based on input pin state.	Various	Selects whether the program will jump or not based on the pin's current level.
Out	N	Setup the parameters of the output pin.	Various	Selects the output function the pin will perform.

Name	Mode	Function	Parameter	Discussion
WaitFor	N	Sets a wait event.	none	In a running program, once the motor has started on a move the program will immediately step to the next instruction. In most step mode cases the next program event must occur after the motor has completed the move. The WaitFor instruction will pause the program flow until the motor completes the move. In interactive and step mode the Hyperdrive will not print the '>' character until the move is complete, and so a controlling computer or PLC need not use this command but can wait for the '>' character before sending the next instruction.
Delay	N	Set a programmed delay.	$0<delay<2^{31}$	This instruction will halt the program flow for the given number of milliseconds.
Loop	N	Start a program loop.	$0=<count<2^{31}$ or blank	When a Loop instruction is encountered the action taken depends on the parameter: If the number is zero or blank, the program will loop forever through the instructions between the Loop and ELoop. If the number is non-zero, the program will decrement the count each time through the loop until the count reaches zero. The program will then move on to the next instruction after the ELoop. In both cases the loop can be interrupted by a Ctrl-C or ESC keystroke.
ELoop	N	End a program loop	none	Defines the end of a program loop, where all instructions between the Loop and ELoop instructions will be included in the loop.
DecelNow	N	Begin a deceleration phase immediately.	none	This instruction can be used to stop the motor via its previously programmed deceleration setting. Useful to stop the motor in a programmed way in speed mode, or when the required motor movement does not have a specific number of steps and some external event is used to stop the motor. Program flow will not continue until the motor has stopped.
Halt	N	Halt the motor instantly.	none	This instruction can be used to stop the motor drive instantly. It should only be used to manage an emergency situation, as while the motor electrical stepping stops immediately, any load inertia will require external dissipation.
StopPwr	Y	Set the stopped motor power.	1 thru 4	Rarely is the power setting required by the motor when stationary the same as when operating. Typically the stopped power is just a fraction of what is required during movement. Use this instruction to set the holding power. By default the setting is 1 or 20% power.

Name	Mode	Function	Parameter	Discussion
End	N	Mark the end of a program.	none	Enter as the last instruction in a program. It switches the terminal program from *Program* (enter a program) mode back to *Interactive* mode.
GoTo	N	Jumps to the given program line.	A valid line number	The program flow will immediately jump to the instruction at the jump address.
Debounce	N	Set an input pin debounce period.	0 thru 50	Allows setting a time period in milliseconds, starting from when that input first changes state, during which any further state changes are ignored. Only if the input still has the same state when the period ends will the input be flagged as in a new state. The default debounce setting is zero.
Clear	N	Clear the program memory.	none	Completely deletes the program in current working memory. Does not change any program in non-volatile storage.
Mode	Y	Change the speed mode.	speed or step	In STEP mode the motor will execute the number of steps given in the Step command, starting when the Step command is given. It will attempt to reach the given SetRPM speed if the accel/decel profile allows. In SPEED mode the SetRPM command will immediately accelerate the motor at the given rate to the given speed.
RPMCtrl	Y	Use an encoder to control the motor speed.	0<number<100	An encoder connected to input ports IN3 and IN4 can be used to modify the current motor speed. The parameter is the increment that is added or subtracted from the speed per encoder click.

Getting Started

This section will introduce the Hyperdrive commands and programming language that are detailed in *Instruction Descriptions*. It starts with setting up some simple move profiles in the terminal program and concludes with developing programs in an external editor or directly in the Hyperdrive.

The text assumes you have correctly installed and are using the TeraTerm communications program supplied on the Hyperdrive CDROM (*Appendix 1 – Installing the TeraTerm Terminal Program*). Hyperdrives are shipped with the data rate set to 9600 baud, 8 data bits and no stop bits. The default protocol is 4-wire RS422. See *Appendix 3 – Changing the Communications Mode* if you only have an 2-wire RS485 adapter.

Interactive Use

This section is an introduction to using the Hyperdrive interactively.

For safety reasons Kremford suggests you familiarise yourself with the Hyperdrive commands and instructions using a stepper motor unmounted in any machine.

With a suitable stepper motor connected as per the *Motor Connections* section, the Hyperdrive connected to but NOT powered up as per the *Power Supply* section, and the RS422 interface connected to a PC, start the TeraTerm program on the PC.

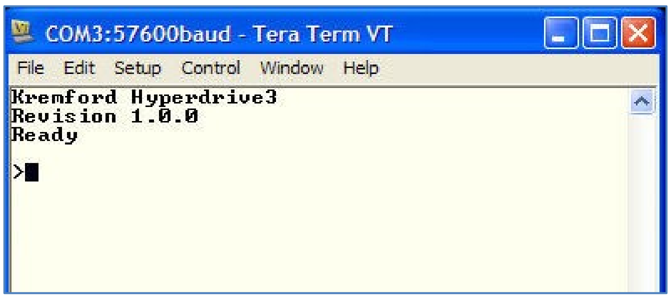
In the following we will assume a 1.8° per step motor is being used.

Before doing anything else, try to turn the stepper motor shaft by hand. This is the unpowered magnetic holding torque. In most cases the shaft should be fairly easily turned by hand although you should feel the rotor toggle over the basic magnetic steps. Remember the feel of this torque.

Start the terminal program (here assumed to be TeraTerm) and turn on the power.

Ensure the current drawn by the Hyperdrive is less than about 150 milliamps.

After a delay of a few seconds the terminal program should show a Hyperdrive power-up banner similar to Figure 20. The '>' character means the Hyperdrive is ready for a command.


COM3:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
Kremford Hyperdrive3
Revision 1.0.0
Ready

>
```

Figure 20. The Hyperdrive power up banner.

If this does not appear check the *Setup->Serial* port settings are as per the *Setup for the Serial Port* section.

Now try and turn the shaft by hand. This should be considerably more difficult and shows the Hyperdrive has energised the motor with its stopped power setting.

The Hyperdrive starts up with the following default settings.

Maximum RPM	5
Move steps	0
Accel/Decel rate	200 RPM/Sec^2
Direction	clockwise
Power mode	1
Step mode	4
Decay mode	2
Debounce	All 4 inputs = 0
Speed mode	Step
Address	(blank)
Stopped power	1
RPM control	off

The default torque setting is the lowest setting, but this should be more than adequate for an unmounted motor at lower speeds. You can increase the power (**Power**) and modify the decay mode (**DcyMode**) settings if the motor stalls as the speed is increased.

In the TeraTerm window, type the following command:

```
SetRPM 60¬
```

The ¬ symbol represents the *Enter* key. Every command must be terminated by this key and its use will be assumed in the following examples. Remember that to the Hyperdrive, “**SetRPM**” and “**setrpm**” are identical - case does not matter. You can use the computer’s BackSpace key if you make a mistake.

The **SetRPM** command above sets the maximum speed the motor can reach as 60 RPM. As mentioned elsewhere, whether it will reach this speed depends on the relationship between the accel/decel times and the number of steps, where below a certain number of steps per move the RPM setting will be unreachable in that many steps with that accel/decel rate.

Now, ensure the motor is clear and, in the same way as above, enter the command:

```
Step 3200
```

Once again terminated by the *Enter* key. The motor will rotate one full revolution (if it is a 1.8° motor) starting as soon as the *Enter* key is pressed. Note that the terminal does not print the > character until the motor has stopped, indicating the command has completed.

The command specifies the motor must execute a move profile of 3200 steps using the default acceleration rate of 200 RPM/Sec^2. The latter means the motor will accelerate at a rate of 200 RPM every second, so in one second it will be rotating at 200 RPM, after two seconds it will be rotating at 400 RPM and so on. For a given rate and speed, the time is easily calculated from:

$$\boldsymbol{Time = \frac{Rpm}{Rate} seconds}$$

For our example here, the time for the motor to reach our setting of 60 RPM will be:

$$\frac{60}{200} = 0.3\ seconds$$

With a 300 mSec accel time and a 300 mSec decel time (they are always the same), the time the total move takes must be at least 600 mSecs for the motor to reach the set speed of 60 RPM.

Note that some small stepper motors may not have the magnetic integrity required to respond to the Hyperdrive default microstep setting of one-sixteenth of a step. If the motor moves an unexpected number of steps or with varying angles, try moving to step mode 3 or 2.

Pressing the *Enter* key without entering any text will repeat a previous **Step** command if no other commands have intervened. The previously commanded move profile will be repeated. Try it now – press the *Enter* key.

Now we will speed up the acceleration time. Enter the following commands:

```
Accel 600
Step 3200
```

Note the much faster acceleration. Try the following command sequence:

```
SetRPM 90
Accel 2000
Step 3200
```

If the motor is capable, this move accelerates to 90 RPM in around 45 milliseconds. A speed of 90 RPM is 90/60 = 1.5 revolution per second.

There is a special keystroke you can use if a move profile must be immediately halted. Hit the Esc key and the move will be immediately aborted, bringing the motor to an abrupt stop. This in itself may cause mechanical damage when the motor is mounted in a machine, so keep this keystroke for emergency situations.

So far we have been working at the default microstep setting of 4, or a one-sixteenth microstep. In many applications, this will be the standard setting. However the Hyperdrive has step settings of 3, 2 and 1, so we will explore these now. Enter the commands:

```
StepMode 3
Accel 500
Step 3200
```

Remember we are assuming a 1.8° motor, so the **StepMode 3** command sets a step-angle of

$$angle = \frac{1.8}{8} = 0.225\ degrees$$

So our 3200 steps now rotates the motor through 0.225 × 3200 = 720°, or two complete revolutions. Without changing anything else, enter the commands:

```
StepMode 2
Step 3200
```

Now the step angle is 0.9° and 3200 steps is 2880° or eight revolutions. Now enter the commands:

```
StepMode 1
SetRPM 100
Step 200
```

The step angle is now the basic motor step angle, and for a 1.8° motor that is 200 steps for one revolution.

While on this setting we can explore some characteristics of stepper motors that are well worth knowing when developing step profiles. The effects you see will depend entirely on the physical aspects of the motor you are using and to a lesser extent the mechanical dynamics of the intended machine loading on the motor. But what we will try to find is the RPM range where the motor’s magnetic resonance will prevent it making accurate steps. First enter a speed of 10 RPM and an acceleration rate of 100, and then slowly increase the top speed using **SetRPM** commands and step commands of, say 400. Even better, if you have the torque/speed curve for your motor, try setting speeds around which the curve shows any torque dips. A typical indication is a sudden jerky movement of the motor rotor as it reaches or transitions this unstable speed.

Once you have found these speeds (and not all motors will show this effect), try going to higher step modes like **StepMode 2**, **StepMode 3** or **StepMode 4**. Note how the effects vanish at the smaller microstepping angles.

If you must use step mode 1 because you need the highest possible peak torque and/or RPM, try setting a high acceleration rate. This will move the motor quickly through the unstable speed range.

You should always try to use the microstepping modes rather than **StepMode 1**, but keep in mind the approximate speed limits for the various stepping modes as listed in *Typical Motor* Characteristics. These are actual limits, and mark speeds from where the actual speed attained would begin to lag behind the entered value.

While exploring the various commands and their effects, you can use the **State** command to display a summary of the current settings. A display listing where this command is given immediately after power is applied would look like:

```
Speed         5.00
Steps         0
Accel Rate    200 RPM/Sec^2
Direction     CW
Pwr mode      1
Step mode     4
Decay mode    1
Debounce      (1=0), (2=0), (3=0), (4=0)
Speed mode    Step
Address       -
Stop Pwr      1
RPM Cntrl     no
Prog Lines    0
```

Use this command at any time to review previous settings. Note that speed can be set in fractions of RPM.

When you give a maximum speed (**SetRPM**) command, the Hyperdrive will first look at the current step mode (**StepMode**) setting. If the new speed setting exceeds that listed in the *Motor Characteristics* table for the current step mode setting you will get a **`Too fast for Step Mode`** error. Also changing to a larger microstep (**StepMode**) may not be allowed until a **SetRPM** command lowers the speed.

RPM, Steps/Second, and Radians/Second Conversions

While it is usually easier to specify motor speed in RPM, letting the Hyperdrive do the conversion to steps/sec, it is sometimes handy to convert between these measures of motor speed. The two basic conversions are:

$$Steps/\sec = \frac{6 \cdot Microsteps \cdot RPM}{Motor\ step\ angle}$$

$$RPM = \frac{Motor\ step\ angle \cdot Steps/sec}{6 \cdot Microsteps}$$

For example, if you have a 1.8° motor running with a microstep setting of 16 (Hyperdrive step mode of 4) and a speed of 600 RPM, the equivalent steps/sec are:

$$Steps/\sec = \frac{6 \cdot 16 \cdot 600}{1.8} = 32{,}000$$

Another handy conversion is between radians/sec and RPM, where the former is often used to define rotational velocity.

$$RPM = \frac{30 \cdot radians/sec}{\pi}$$

$$radians/sec = \frac{RPM \cdot \pi}{30}$$

Converting acceleration from the Hyperdrive's RPM/sec^2 to Steps/sec^2 produces some rather large numbers. For example the Hyperdrive maximum acceleration rate of 30,000 RPM/sec^2 for the same case as above becomes 1,600,000 Steps/sec^2.

Setting the Stopped Power

In most cases setting the stopped power will need to wait until the motor is mounted in its machine. You should aim to use the lowest power setting to minimise power dissipation and consequent heating of both the Hyperdrive controller and the motor.

Exercise the machine though its various sequences where the static torque on the motor is at a maximum. In particular examine the first and last few steps of an accel/decel movement to ensure there is no "cogging", where the machine inertia means steps are sometimes missed. This can sometimes be prevented by lengthening the acceleration time. If acceleration times are critical and the cogging consistent, it can sometimes be ignored.

External Motor Control

If you are using a PC or PLC to interactively drive the Hyperdrive you may never need to use its ability to store and run an internal program. In this case the PLC transmits the control instructions much as we have been doing so far, with the PLC program monitoring for the '>' character that indicates a previous command has been completed.

How this is done depends on the particular PLC used and so cannot be dealt with here, but that simple transaction of sending a command and only transmitting the next when the Hyperdrive responds is very simple and reliable.

Internal Program Control

Writing a Program

The interactive mode we have been using to date is helpful while developing the requirements of a controller/machine interface, or where controlling the motor from a PC or PLC is the intended operating mode. However many applications can be simply controlled by a program residing in the Hyperdrive memory.

A program is loaded into the Hyperdrive by preceding each program instruction by a line number. This puts the Hyperdrive into a special mode where the instruction is not acted upon but stored in line number order in the controller's internal memory. As an example, type in the following program:

```
>100 SetRPM 100
>110 Accel 400
>120 StepMode 4
>130 Step 3200
>
```

You can use the computer's *BackSpace* key if you make a mistake. The Hyperdrive examines each line on entry for syntax errors if any occur and will print a corresponding error message. However it cannot detect errors such as the wrong sense in **Out** instructions or wrong destination addresses. Confirm you made no mistakes by commanding Hyperdrive to list the program contents using the **List** command. In this case the terminal interaction would look like:

```
>list
  100  SetRPM   100.00
  110  Accel    400
  120  StepMode 4
  130  Step     3200
>
```

You start a program running with the **GO** command. Enter that now. As you would expect the motor executes one full rotation.

You can use the **Clear** command to completely erase a current program.

You can replace or change an instruction by simply retyping the line with that line number.

By spacing program line numbers apart by some convenient increment like 10 as above, it is then easy to insert new instructions between the original instructions if you later decide they are necessary.

Note that as mentioned earlier, the maximum speed setting (**SetRPM**) is dependent on the step mode setting (**StepMode**). In a program, Hyperdrive will always ensure that the speed cannot exceed the maximum speed for the current step mode. A program sequence such as:

```
...
190 StepMode 4
200 SetRPM 1900
...
```

will produce an error message because the speed is too high for that step mode.

Even if you are controlling the Hyperdrive from an external computer, the ability to down load small programs that execute particular motor profiles can be very useful where the controlling PLC may not always be able to respond quickly enough to events within a complex move sequence.

For many applications the program will need to be repeated over and over while ever the Hyperdrive is powered. If you study the input/output instructions (I/O) you will see that the program can be controlled by external events using these capabilities. For now we will try some simple repetitive sequences.

When under program control, the **WaitFor** command will pause the program execution until the previous motor move has completed when in the **STEP** speed mode, and the **Delay** command will delay program execution for the given number of milliseconds. You could use the **Delay** command instead of a **WaitFor** command where the delay time is chosen to include the motor move time, but variations in motor load, etc, mean the **WaitFor** is always a better solution.

An example application of both commands is the situation where the motor must first rotate a number of degrees clockwise, and then rotate some other number of degrees counter-clockwise. Anything but the lightest load will require at least a few milliseconds after each move completes for the machine inertia to come to rest, so we will initiate a move, use the **WaitFor** command to wait until the motor stops, and then use the **Delay** command to insert a small delay before starting the next move.

Type in the following program:

```
110 Accel 2000
120 SetRPM 100
130 Dir cw
140 Step 3200
150 WaitFor
160 Delay 500
170 Dir ccw
180 Step 1600
190 WaitFor
```

Type **GO** and the motor will rotate in one direction, pause for half a second, and then rotate in the other direction.

Use the **Loop** instruction to continuously repeat a particular sequence. The instructions to be repeated are enclosed between the **Loop** and **ELoop** instructions. The loop is repeated for the number of times specified by the **Loop** parameter (the loop variable). If this value is zero or non-existent the loop will repeat forever.

The operation is the same for the case where the parameter value is non-zero, but now the loop will be repeated only the number of times defined by the parameter.

There may be several loops in a program, but no loop can contain another loop.

You can interrupt a running program by entering a *Ctrl-C* keystroke. This is where you hold down the Ctrl key and then press the C key. This will exit the program mode, decelerate the motor to a stop at the **Accel** setting, and return to the interactive mode. If the move itself must be aborted immediately for safety reasons, use the *Esc* keystroke.

Without changing anything from the previous program, enter the following lines:

```
100 Loop
200 Delay 100
210 ELoop
```

Now when you type **GO** the motor will execute these two movements over and over. Use a *Ctrl-C* to stop the program and change the loop variable to five:

```
100 Loop 5
```

Now start the program again. It will execute those two movements a total of five times.

Changing Program Flow

You can change the program flow with the **GoTo** linenumber instruction and the **JumpIf** input instruction. Go to destinations are specified as the line number of the required destination. Here is an example that changes the number of steps based on whether IN1 is either ON or OFF. Remember to clear any previous program by first typing **Clear**.

```
100 SetRPM 100         ; 100 RPM
110 Accel 500          ; 500 RPM/Sec^2
120 Loop
130   JumpIf 1,on,160  ; jump if IN1 is on
140   Step 1600        ; 180 degrees in sm4
150   goto 170
160   Step 3200        ; 360 degrees in sm4
170   WaitFor
180   Delay 1500       ; 1.5 secs delay
190 ELoop
```

Once the loop is entered, at line 130 the program examines the state of input 1. If it is off (no voltage between terminals IN1- and IN1+), program flow continues to the 1600 step instruction which immediately starts the motor on its profile. Well before this concludes the **GoTo** instruction will jump to line number 170 and remain there until the move concludes.

However if input 1 is on (a voltage of 5 volts or above between terminals IN1- and IN1+), the program will jump to line number 160 and execute the 3200 steps. When the motor comes to a halt there is a 1500 mSecs delay. The program then loops back and once again examines the IN1 input state to decide how many steps to make.

Note that for backward compatibility with earlier Hyperdrives, the letter "h" or "H" can be used for "ON", and the letter "l" or "L" can be used for "OFF". Earlier Hyperdrives used ground referenced inputs, and so an *on* input was a *high* voltage.

This program ran in step mode. What would it look like in speed mode?

In speed mode we cannot specify and exact number of steps, but we can specify speed as well as time. Here is almost the same program in speed mode:

```
>Mode speed

100 Accel 2000          ; 2000 RPM/Sec^2
110 Loop
120   JumpIf 1, ON, 150 ; jump if IN1 is on
130   SetRPM 60         ; start spinning at 60 rpm
140   goto 160
150   SetRPM 120        ; start spinning at 120 rpm
160   Delay 2000        ; spin for 2 secs
170   decelNow          ; decelerate to stop
180   Delay 1500        ; 1.5 sec delay
190 ELoop
```

Spinning at 60 RPM for 2 seconds the motor should rotate for about 2 revolutions.

Whether you use **SPEED** or **STEP** mode really depends on your application. If an exact positioning profile is necessary where the motor must rotate a specific number of degrees, then step is the way to go. It is also the recommended mode when the motor is using external switches to control the motor start and stop, because setting the move at a number of steps that just exceed the switch controlled profile provides a safety stop if a switch should fail.

Dynamic Changes in a Running Program

A common requirement when operating in speed mode is to make changes to the running motor's speed. This presents no problem when the Hyperdrive is being controlled by a sequence of commands from a PC or PLC controller - **SetRPM** commands are just sent as required to change the speed. However, the same requirement can occur when the Hyperdrive is executing an internal program – the sequence of events for which the Hyperdrive is responsible does not change, but a machine-wide change could require some small change to the motor's operating speed.

An example would be a Hyperdrive controlling a label applicator. The Hyperdrive program uses the associated label sensors to control the start/run/stop sequence, where the motor speed is related to the overall conveyor system speeds. As such the Hyperdrive runs autonomously with its parameters fixed. However, to allow for example some overall machine controller to change the entire machine operating speed, the Hyperdrive program must also be changed to maintain synchronization.

The technique is to simply send the new instruction with the modified speed setting when the change is required. The Hyperdrive will stitch the new line into its running program as it is received and the change will be made when next that line is executed.

The line must include the line number of course and there should be no comments. The line can only replace an existing line and the instruction type must match - only the parameter can change.

The instructions that can be dynamically changed this way in a running program are limited to **Step**, **Accel**, **SetRPM** and **Delay**.

Note that the ability to change the acceleration rate while the motor is running allows different accel and decel rates.

Using a Quadrature Encoder

In the section *Input and Output Options* there was a section on the connections for a quadrature encoder. Here we give two examples that use an encoder, first for position control, and second for dynamic speed control.

Many applications require a motor to be a slave to some sort of positioning control. An example might be a requirement for the incremental control of some sort of positioning motor, where a click of the encoder knob either clockwise or anti-clockwise means the motor rotates a corresponding number of degrees in the matching direction.

Figure 15 shows the typical encoder wiring diagram where IN3 and IN4 are connected to the A and B encoder outputs, and the Hyperdrive +5V and SGND provide the power. In this example we will assume that the motor must rotate 90° for each encoder click.

We will use a 1.8° motor and a **StepMode** of 4, so a single step is 0.1125°, and a 90° rotation will be:

$$Steps = \frac{90 \cdot 16}{1.8} = 800$$

The example program is in the *Programs* folder on the CDROM, but the relevant lines that query the two inputs and initiate the move in the required direction are:

```
.
.
140 Loop                    ; start an endless loop
150   ContWhen 3, h         ; wait until IN3 goes high (A input)
;
; IN3 has just changed state. Assume IN4 is high
160   Dir CW                ; set first direction option
170   JumpIf 4, h, 190      ; jump if IN4 is high
;
; IN4 is low - set opposite direction
180   dir CCW
;
; Now set steps, speed, etc as required
190   Steps  800
.
.
300   ContWhen 3, l         ; must ensure IN3 is low before cont
.
.
400 Eloop
```

Using an Encoder for Speed Control

There are applications where small changes in the machine over time, or where there are small variations in the speed of associated motors or conveyors etc, where the Hyperdrive controlled motor may need some fine speed adjustment.

Or perhaps an application simply requires a variable speed control, where turning a knob to control the speed is all that is necessary.

The **SetRPM** instruction can be edited in the program to make fractional RPM changes, but often the adjustment must be done in real-time with the machine operating. The **RPMCtrl** instruction provides for using an encoder with a knob mounted on the machine control panel that can set the RPM change per click to some suitable value.

The wiring diagram is the same as Figure 15.

The programming is simple – just include the **RPMCtrl** instruction in the setup section of the program, choosing an increment value suitable for the particular machine. To keep accelerations within bounds, the maximum increment is 100 RPM. However the minimum change can be as small as 0.01 RPM.

A program that provides for speed control using the encoder is:

```
100 Accel 1000    ; set the required accel/decel rate
110 Mode speed    ; change to speed mode
120 RPMCtrl 10    ; speed increments are 10 RPM steps
```

After running this program the motor speed is now under the control of the encoder. The first click clockwise will start the motor rotating at 10 RPM.

There is a sample program in the *Programs* folder of the CDROM that further demonstrates this instruction.

Simulating a Step/Direction Interface

You can implement a standard Step/Direction interface in the Hyperdrive with a simple program. Essentially the step input is connected to one of the inputs and the direction to another. The program then uses **ContWhen** instructions to initiate a motor move each time the step pulse changes state. When it does the motor is commanded to make a single step.

```
; Setup the required power, step mode, etc, here...
;
; Now setup for the step/direction interface.
;
200 SetRPM 1000           ; set some fairly high speed and accel
210 Accel 5000
220 Loop
220   ContWhen 1, ON      ; jump when input 1 goes high
240   Step 1              ; make a step
250   ContWhen 1, OFF     ; now must wait for input to go low
260 Eloop                 ; loop forever
```

The program must wait for the input to return to the pre-step state before once again waiting on the next step pulse rising edge. A programmed single step like this takes about 70 uSec, and a realistic step interval is about 110 uSecs. As an example, this gives a top speed of around 690 RPM in step mode 3, which could be quite useful for some applications.

Note however that the step interval is now under the control of the program system and not the hardware timers, so there will always be some time jitter between steps.

To include a *Direction* capability, include within the loop an input test that jumps around two **Dir** instructions as shown earlier. This assumes some external system is ensuring the direction input cannot change unless the motor has stopped.

It is quite a suitable solution for most applications although not capable of very high speeds.

Saving the Program

Unless you save it, any program you enter into the Hyperdrive working memory will be lost when the power is removed. However you can save a program to the Hyperdrive's non-volatile memory and restore it using the **Save** and **Load** commands. Type **Save** and the program is written to storage. When power is restored type **Load** to restore the program to working storage.

You are guaranteed 100,000 writes to non-volatile storage (i.e. you can give 100,000 Save commands). Exceeding this limit is unlikely in a manual situation, but be careful if you use this command in a computer driven application. For example, a Save instruction executed every minute during a 12 hour shift 5 days a week would wear out the non-volatile memory in about 28 weeks!

Auto Restore and Run

In many cases a Hyperdrive will be installed on a machine where the machine function is always the same and thus the program it runs will always be the same. Or perhaps the machine must be programmable, but for many days or weeks it must run the same program whenever it is powered up.

In these cases, once the program is finalised, write it to storage as described above but then type the additional **RunUp** command. This sets up a special sequence so that when power is restored the Hyperdrive automatically executes a **Load** and **Go** command sequence.

You can monitor this sequence if a terminal program is connected to the Hyperdrive at power up as it automatically lists the program prior to starting it running. An example of what you would see is:

```
Kremford Hyperdrive3
Revision x.x.x
Ready

>Run immediate:
>
  100  SetRPM   150
  120  Accel    500
  130  Loop     0
  140    Dir      CW
  150    Step     3200
  160    WaitFor
  170    Delay    500
  180    Dir      CCW
  190    Step     1600
  200    WaitFor
  210    Delay    200
  220  ELoop

>GO
```

There is a delay of five seconds after the program is listed before the **GO** command is automatically entered, giving an opportunity to abort with a *Ctrl-C* if necessary.

Note that the program need not actually do anything. For example if the Hyperdrive always works in an interactive mode, the saved program could just set some standard static settings such as the step/speed mode, power setting, acceleration, etc and then conclude.

Note also that a saved program allows a quick switch from a main program to an adjunct program to change a machine mode. For example a Hyperdrive controlling a motor via a number of sensors in a fixed program would normally have the program saved in non-volatile storage so it is automatically restored on power up. The Hyperdrive would then respond as required when the machine starts.

However, perhaps from time to time the normal machine operation is halted and a mechanical setup requires the Hyperdrive/motor combination to provide a different functionality during this operation. Preceding the downloaded replacement program with a **Program** or **Clear** command will clear any existing Hyperdrive program and replace it with the new. Once the setup is completed the original (saved) program can be instantly restored with a **Load** command.

Using an External Text Editor

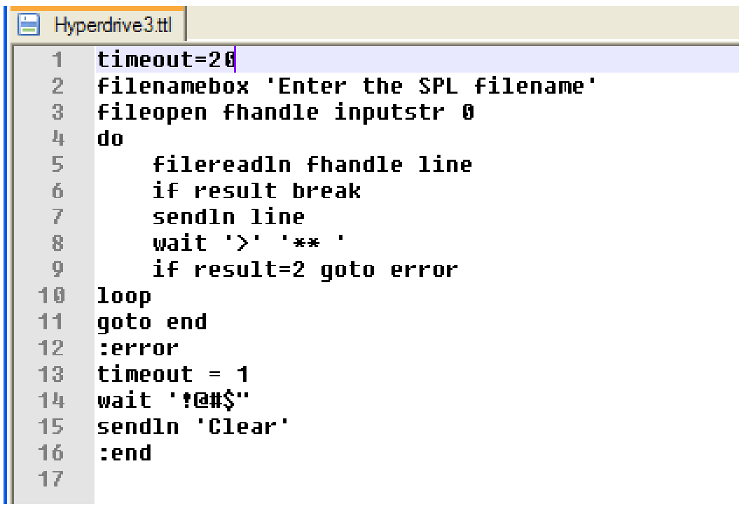
The Hyperdrive does not store comments and on entry will ignore any characters in a line after a ';' character. Thus if you develop a program using an external text editor on a PC you can include comments and cut and paste and insert and delete as required. It can be saved to a PC folder like any other text document and a library of programs built up over time. When you are satisfied you can download it to the Hyperdrive via the RS422/RS485 interface.

You can use any plain text editor; however the public domain editor Notepad++ is included on the accompanying CDROM and has a language add-on specifically for the Hyperdrive, including syntax highlighting (i.e. Hyperdrive instructions are colour-coded). *Appendix 2 – Installing the Notepad++ Program* shows how to install this editor and configure it for Hyperdrive programs.

Downloading Programs

Included on the CDROM is a selection of sample programs in the folder called Programs. Hyperdrive programs have the default filename extension SPL.

Always use the Kremford supplied TeraTerm macro for downloading programs to the Hyperdrive. The *Programs* folder of the Hyperdrive CDROM contains this macro called **`Hyperdrive3.ttl`**.



```
timeout=20
filenamebox 'Enter the SPL filename'
fileopen fhandle inputstr 0
do
    filereadln fhandle line
    if result break
    sendln line
    wait '>' '** '
    if result=2 goto error
loop
goto end
:error
timeout = 1
wait '!@#$"
sendln 'Clear'
:end
```

Figure 21. The Kremford supplied TeraTerm file download macro.

The text of the macro is shown in Figure 21. The macro line **`wait '>' '** '`** normally pauses the file transmission until the Hyperdrive prints either the command prompt character '>' indicating the line passed syntax checking, or '** ' indicating the Hyperdrive sent an error message of some sort.

Note: This means the string '** ' must not appear anywhere in your program, as it mimics the first three characters of an error message.

Subsequent code ensures that if an error does occur, the download will stop after the line in error with the error message displayed and the program downloaded so far will be cleared, ensuring a program with errors cannot be accidently executed.

You initiate a Hyperdrive file download from the TeraTerm *Control->Macro* menu option.

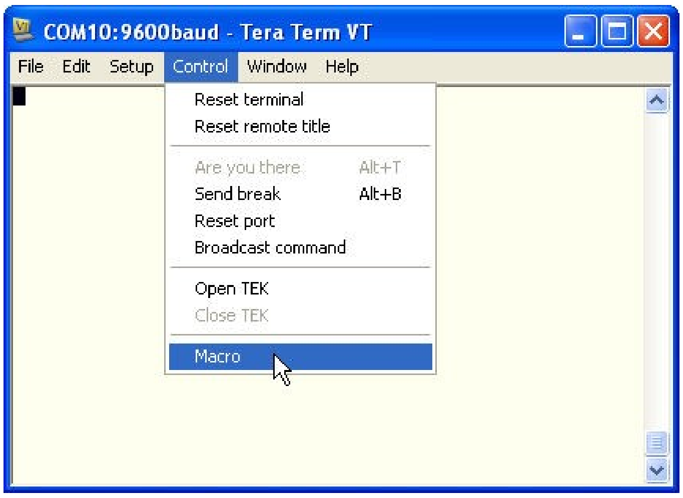
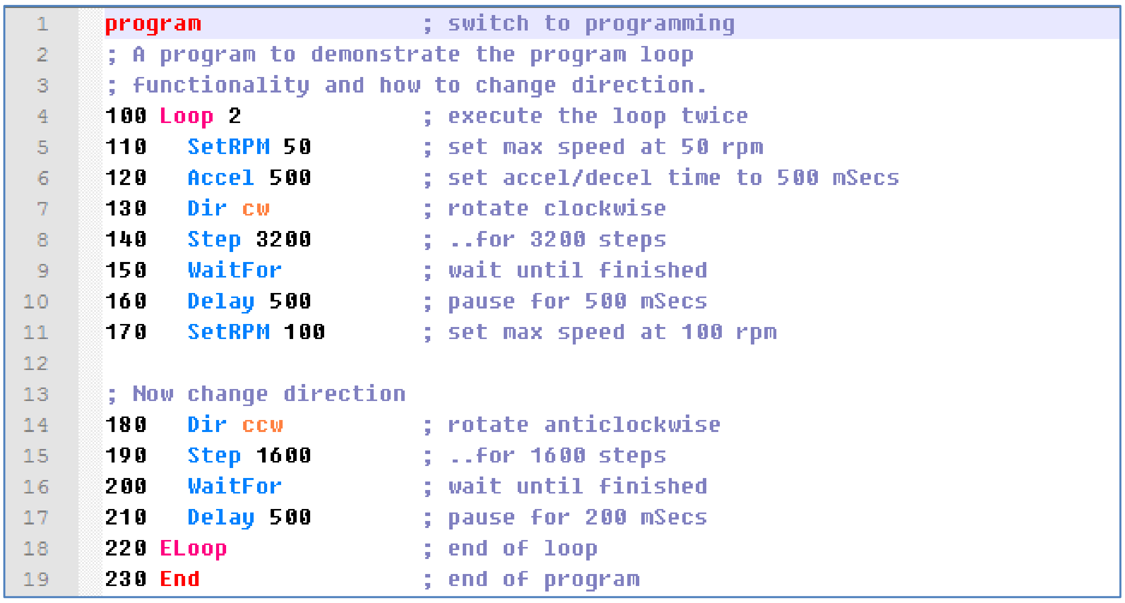


Figure 22. Selecting the Load Macro file transfer option.

Clicking on *Control->Macro*, navigating to where the Hyperdrive macro is stored and selecting it will bring up a dialog allowing selection of the appropriate SPL program. Note that this program selection dialog is sometimes displayed with background priority, meaning it can be beneath any other windows displayed on the desktop.

In your text editor of choice, select and load the program **`Reverse.spl`**. Here is what it will look like in Notepad++:



```
program                   ; switch to programming
; A program to demonstrate the program loop
; functionality and how to change direction.
100 Loop 2                ; execute the loop twice
110   SetRPM 50           ; set max speed at 50 rpm
120   Accel 500           ; set accel/decel time to 500 mSecs
130   Dir cw              ; rotate clockwise
140   Step 3200           ; ..for 3200 steps
150   WaitFor             ; wait until finished
160   Delay 500           ; pause for 500 mSecs
170   SetRPM 100          ; set max speed at 100 rpm

; Now change direction
180   Dir ccw             ; rotate anticlockwise
190   Step 1600           ; ..for 1600 steps
200   WaitFor             ; wait until finished
210   Delay 500           ; pause for 200 mSecs
220 ELoop                 ; end of loop
230 End                   ; end of program
```

Figure 23. A program as displayed in Notepad++ ready for download to the Hyperdrive.

Comments are any text preceded by a semi-colon (;). Hyperdrive will ignore any characters from a semi-colon until the end of the line. The advantage of comments is the program itself is self-documenting and the comments can be a considerable help when returning to a program at a later time.

Changes are made in the editor and saved and the **`Hyperdrive`** macro described above used to send it to the Hyperdrive.

Program Errors during a Download

Syntax errors that occur in a program line when entered either interactively or via a download will produce a descriptive error message and the line will be ignored. In an interactive mode you would observe the error report and correct the mistake.

However, in a situation where a controlling program running in a PC or a PLC is downloading programs without supervision, lines that contain an error will still be flagged as errors, but now will be missing from the program. Note that the error may just be a garbled transmission error and not in the original program.

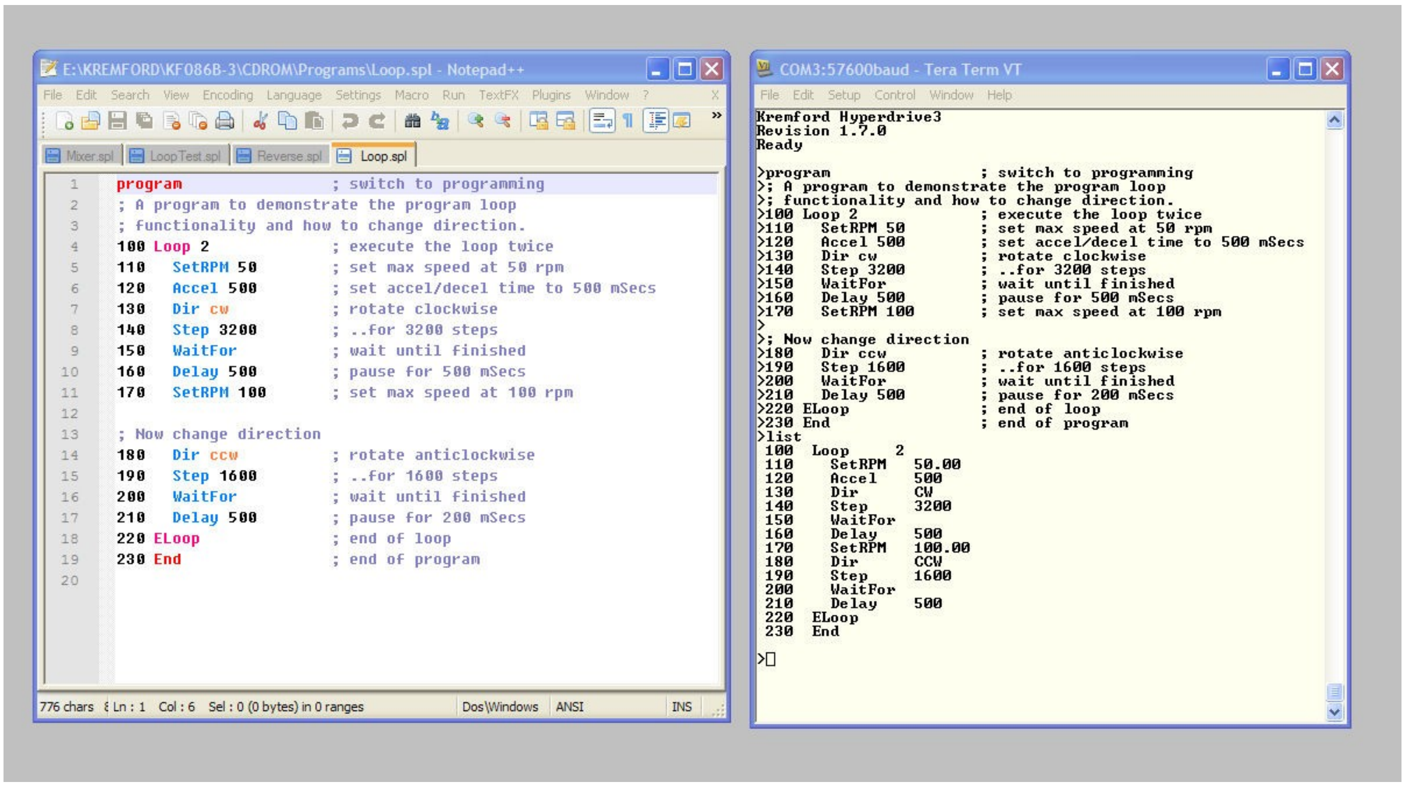
A subsequent **Go** command would attempt to run the program with these lines missing, with potentially dangerous results

It is very important in these situations that the supervising program monitors for these errors. The simplest test is to monitor for the double asterisks and space "** " that precede all error messages. They (and the error message) precede the ">" character which is the normal "ok" response to a correct program line.

A Programming Environment

The recommended method is to open both the text editor and the terminal program at the same time as shown in *Figure 24*. Here the user has been editing the program in Notepad++. It has been saved and the user then switches to the TeraTerm window. The download macro has been selected as per Figure 22 and the just saved program selected in the user's saved programs folder. TeraTerm has responded by downloading the program into the Hyperdrive over the RS422/RS485 interface. The Hyperdrive shows any comments while the download proceeds as shown here, but when the user subsequently lists the downloaded program the comments are gone. They are stripped off before being stored as they would otherwise take up memory.

This is common practise where production programs require comments for recording or qualification purposes. A program can be edited in Notepad++ and downloaded and tested in TeraTerm, repeating these steps as required until the functionality is as required.


E:\KREMFORD\KF086B-3\CDROM\Programs\Loop.spl - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
Mixer.spl LoopTest.spl Reverse.spl Loop.spl
776 chars Ln : 1 Col : 6 Sel : 0 (0 bytes) in 0 ranges Dos\Windows ANSI INS
COM3:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
program                 ; switch to programming
; A program to demonstrate the program loop
; functionality and how to change direction.
100 Loop 2              ; execute the loop twice
110   SetRPM 50         ; set max speed at 50 rpm
120   Accel 500         ; set accel/decel time to 500 mSecs
130   Dir cw            ; rotate clockwise
140   Step 3200         ; ..for 3200 steps
150   WaitFor           ; wait until finished
160   Delay 500         ; pause for 500 mSecs
170   SetRPM 100        ; set max speed at 100 rpm

; Now change direction
180   Dir ccw           ; rotate anticlockwise
190   Step 1600         ; ..for 1600 steps
200   WaitFor           ; wait until finished
210   Delay 500         ; pause for 200 mSecs
220 ELoop               ; end of loop
230 End                 ; end of program
```

```
Kremford Hyperdrive3
Revision 1.7.0
Ready

>program                ; switch to programming
>; A program to demonstrate the program loop
>; functionality and how to change direction.
>100 Loop 2             ; execute the loop twice
>110   SetRPM 50        ; set max speed at 50 rpm
>120   Accel 500        ; set accel/decel time to 500 mSecs
>130   Dir cw           ; rotate clockwise
>140   Step 3200        ; ..for 3200 steps
>150   WaitFor          ; wait until finished
>160   Delay 500        ; pause for 500 mSecs
>170   SetRPM 100       ; set max speed at 100 rpm
>
>; Now change direction
>180   Dir ccw          ; rotate anticlockwise
>190   Step 1600        ; ..for 1600 steps
>200   WaitFor          ; wait until finished
>210   Delay 500        ; pause for 200 mSecs
>220 ELoop              ; end of loop
>230 End                ; end of program
>list
 100  Loop      2
 110    SetRPM    50.00
 120    Accel     500
 130    Dir       CW
 140    Step      3200
 150    WaitFor
 160    Delay     500
 170    SetRPM    100.00
 180    Dir       CCW
 190    Step      1600
 200    WaitFor
 210    Delay     500
 220  ELoop
 230  End

>
```

Figure 24. Running both a text editor (Notepad++) and a terminal program (TeraTerm) in a PC based programming session.

Instruction Descriptions

This section describes each instruction in detail. A brief description of the command function is followed by its mnemonic, with the 2 character equivalents from the original Hyperdrive listed in brackets. Earlier programs are completely compatible, and you can intermix the instruction sets if required.

The **Program** entry specifies whether the instruction can be used in a program.

The **Interactive** entry specifies whether the instruction can be used as a direct command.

The **Parameter** entry specifies whether the instruction needs a parameter. The required format of the parameter(s) is described in the text. Unless stated otherwise the range of a numeric parameter is 1 through 2,147,483,647.

These are followed by further description and possibly one or more examples.

Accel (AD)

Set acceleration rate

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

Sets the acceleration rate in RPM/Sec^2. This sets the rate and indirectly how long the motor will take to reach the final speed defined by the **SetRPM** speed setting command. For example, an **Accel** setting of 100 means the motor will increase its speed by 100 RPM every second, and so with a speed setting of 500 RPM, this means the motor will take 5 seconds to reach that speed. An **Accel** setting of 1000 and an **SetRPM** setting of 150 means an acceleration event starting from stopped will reach the target speed in:

$$\frac{150}{1000} = 0.15\ seconds, or\ 150\ milliseconds$$

Parameter range is 30 to 30000 RPM/Sec^2. The accel and decel rates are always the same in STEP mode, but can be different in SPEED mode because this instruction can be given while the motor is running.

Example:

```
>SetRPM 80      ; set speed to 80 RPM
>Accel 200      ; 400 mSecs to reach that speed
>
```

Angle (SA)

Set motor step angle

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

This command informs the Hyperdrive of the basic step angle of the connected motor. The value is in thousandths of a degree, and the default is 1800 or 1.8°.

Note that not setting this value for motors with other basic step angles will result in acceleration times being incorrect.

Backtrack steps from here

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

Use this instruction to backtrack to or return to a known step position. The parameter is the required speed.

It is a no-operation if the motor is turning.

It reads the current step count, reverses the motor direction and, using the same profile description apart from the speed, drives the motor until the step count goes back to zero. All things being equal, this should be the position where the profile began.

Examples:

There are many mechanical situations where the stepper motor must first drive one way for a specified number of steps, and then drive in the other direction for the same number of steps to return to the start position. There would typically be a **Monitor** instruction supervising the drive out operation so that an emergency switch on an INx line will immediately halt the motor. Often the motor will be required to return to its start position after halting.

Where the start position is not controlled by some sort of electrical switch but relies only on the number of steps, jumping from where the motor stopped after some emergency (via the **Halt** instruction) to the normal reversing code will use the step count from the **Step** instruction which will be in excess of the now foreshortened number of steps required to return to the original start position. The motor will proceed past the normal parked position, possibly stalling on a mechanical stop.

By picking up the current accumulated step count, the **BakTrak** instruction starts the motor in the reverse direction with all its existing parameters (accel rate, step mode, etc). You could use it in a program that extends and retract a ram and includes an emergency switch on the IN1 like this:

```
100 debounce 1, 20      ; set IN1 debounce delay
110 angle 2             ; 0.9 degree steps
120 loop                ; start infinite loop
130   setrpm 230        ; 230 rpm
140   contwhen 1,on     ; emergency switch must be on
150   dir cw            ; cw extends the ram
160   Monitor 1,off,220 ; jump if line 1 switches off
170   step 320000       ; extend for this many steps
180   waitfor           ; wait till done
190   delay 200         ; pause extended for 200 mSecs

200   ContWhen 1,on     ; clear the jump while ram retracts
210   goto 240
220   halt              ; arrive here if emergency during extend
230   waitfor           ; ensure stopped
240   baktrak 150       ; reverse back slower to start
250   waitfor           ; wait till done
260   delay 2000        ; pause at home
270 eloop
```

The first **ContWhen** instruction will prevent the program from even starting if the IN1 is off. If on the **Dir** sets the ram extend direction.

The **Monitor** instruction arms the IN1 line to cause a program jump if the line switches to off during the time the ram is extending.

The **Step** instruction starts the ram on its way out with the **WaitFor** instruction pausing the program during the travel.

In normal operation there is a 200 mSec pause once fully extended. The next instruction is a **ContWhen** and this clears the previous **Monitor** and as a bonus ensures the IN1 line is still on. The program then skips down to the **BakTrak** instruction which sets the new speed, picks up the completed step count from the previous **Step** command (ie the 320000), reverses the motor, and starts the normal retraction phase. Now any change on the IN1 line will be ignored during retraction.

However an on to off transition on the IN1 line while the ram is extending will immediately jump to line 220, causing an immediate halt. The following **BakTrak** instruction causes the ram to immediately start back to its proper parked position.

Baud

Set the RS422 Communications Baud Rate

Program:	NO
Interactive:	YES
Parameter:	YES

Hyperdrives are shipped with the communication baud rate set at 9600. You can use this command to select from the two inbuilt baud rates of 9600 and 57600.

```
>Baud 57600
```

Clear (CL)

Clear the Program memory

Program:	NO
Interactive:	YES
Parameter:	NO

Giving this command immediately clears the program memory. A subsequent **List** command will give the error message:

```
** List... No Program
```

It does not clear the non-volatile memory.

ClrAddr (HX)

Clear an addressed Hyperdrive

Program:	NO
Interactive:	YES
Parameter:	REQUIRED

Clears an address mode. The address must match the original **SetAddr** command. Also clears the address saved in non-volatile storage.

Example:

```
>ClrAddr 5
>
```

ContWhen (IC)

Continue on an input event

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

The **ContWhen** input instruction has a number of parameters that define which input pin pair (**INx**) the instruction applies to, and whether the terminals going from driven (ON) to undriven (OFF) or vice verca initiates the action. The action is to pause the program for a FALSE test, or continue if TRUE. The syntax is:

```
ContWhen <terminal_pair>, <state>
```

The <terminal_pair> parameter can have the values 1 through 4 and selects the corresponding IN1 thru IN4 input screw terminals.

The <state> can be either ON or H, or OFF or L, with ON or H meaning a voltage above 4 volts between the INx- and INx+ terminals. As elsewhere, the case does not matter.

On encountering a **ContWhen** instruction the Hyperdrive will select and examine the current state of the specified input pin. If the state is TRUE then the program will continue immediately. A TRUE state is where, for example, the instruction specifies on and the between terminal voltage is already above 4 volts.

If the state is FALSE the program will halt at that instruction, entering an internal loop where it continuously checks for the required state becoming true, effectively pausing the program at this point.

Example:

This example demonstrates the case where the transition comes from an external sensor that signals an event to start the motor. If the sensor produces a transition which drives current through the respective input on the event, and it was connected to pin IN1, the section of the program concerned with this event would look like:

```
.
270 ContWhen 1,on    ; wait here until input 1 is on
280 Step 3200        ; start the motor on a 3200 step profile
.
```

The program will examine the **ContWhen** instruction and check if **IN1** is ON. If not it will pause waiting for this event. When **IN1** does turn ON, or if it was ON already, the program will step to the next instruction. The **Step** instruction will immediately start the motor running a 3200 step profile.

DcyMode (DM)

Set decay mode

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

Sets the output bridge decay recirculation mode. The decay mode sets how the controller manages the recirculating currents in the internal driver circuit during each “off” cycle period. The settings are 1, 2, 3 and 4, and these settings correspond to decay ratios of 0%, 25%, 50% and 100% respectively. The default setting is mode 2, and in many applications this will be sufficient. For high power and high speed situations try setting 3 or 4.

As the decay settings can be adjusted, the current discharge pattern can be modified according to motor load and speed. By choosing the best mix of decay modes, the PWM ripple that appears on the sinusoidal microstepping output can be minimised, reducing both audible noise and vibration.

Example:

```
>dcymode 3
>
```

Debounce (DB)

Set an input debounce delay

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

The **Debounce** input instruction has a number of parameters that define which input (**INx**) pin the instruction applies to, and a number between 0 and 50 that specifies the debounce delay time for that input. The delay time is expressed in milliseconds. The syntax is:

```
Debounce <terminal_pair>, <delay_time>
```

For the KF086 the pin number can be 1 through 4. The default value is 0 milliseconds, the typical setting for an electronic input device.

When the device connected to one of these inputs is a mechanical switch, the possibility exists that when it transitions from on to off or vice versa, the switch contacts will “bounce”, causing not just one clean transition but several. It is possible the Hyperdrive would detect these multiple transitions and, depending on the program, misinterpret what has occurred. So whenever the input device is a mechanical switch you should ensure this delay is set and valid for that particular switch. In most cases a value around 8 mSec should be sufficient but the only way to be sure is an analysis of the switch line with an oscilloscope.

Conversely, if the input device is an electronic device then its output during a transition is likely to be a single clean transition. Most optical and magnetic sensors are of this type and the default setting of 0 mSecs is correct for these sensors.

The setting will be the minimum delay time. Due to synchronization with the internal clock, the delay can be up to 1 millisecond longer. For example, a set debounce delay of 5 milliseconds could be up to 6 milliseconds long.

DecelNow (DC)

Force Immediate deceleration

Program:	YES
Interactive:	NO
Parameter:	NO

When encountered in a running program, this instruction will force an immediate deceleration until stopped at the current **Accel** deceleration rate.

The program pauses until the motor has stopped.

Delay (DL)

Set delay in milliseconds

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

When encountered in a running program, this instruction will pause the program for the given number of milliseconds.

Example:

There are many situations where a delay is required within a program. An obvious situation is where a motor must travel in one direction and subsequently reverse, where a small delay allows the machine to stabilise at its stopped position.

```
.
150 SetRPM 100        ; set speed at 100 RPM

160 Dir cw            ; clockwise
170 Step 182400       ; for 57 revolutions
180 WaitFor           ; wait until done
190 Delay 500         ; wait 500 mSecs before reversing
200 Dir ccw           ; anti-clockwise
210 Step 32000        ; for 10 revolutions
.
```

Another possibility is where the motor must stop for a fixed period to allow some other process to complete.

```
.
72 Dir cw
73 Step 450100        ; move to extended
74 WaitFor
75 Delay 5000         ; allow 5 second pressing time
76 Dir ccw
77 Step 450100        ; return
78 WaitFor
.
```

Dir (DR)

Set direction

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

Sets the motor direction using the words “cw” or “ccw”. The direction the motor will actually go will depend on the relative connections for the two motor windings. If giving this command produces motor rotation in the opposite direction, simply transpose the wires from one of the field windings.

Example:

```
>Dir CCW
>
```

You can insert a **Dir** command to change the motor direction while it is running when the Hyperdrive is in speed mode, but only when the speed is less than 10 rpm. This prevents accidental reversals at high speeds which could be very dangerous.

ELoop (EL)

End of program loop

Program:	YES
Interactive:	NO
Parameter:	NO

Marks the end of a program loop.

Example:

```
>loop
  .
  .
>eloop
```

End (EN)

End of program

Program:	YES
Interactive:	NO
Parameter:	NO

Marks the end of the program. Not strictly necessary except when programs will be downloaded under computer control or without supervision (see *Program Errors during a Download*).

When a running program gets to this instruction the Hyperdrive will automatically return to the Interactive mode.

ESC

The ESC (escape) command

Program:	NO
Interactive:	YES
Parameter:	NO

The Hyperdrive will immediately stop sending step pulses if it receives an escape character (*ESC*) during the execution of a **Step** profile in STEP mode, and at any time the motor is rotating in SPEED mode.

The power setting (**Power**) will remain at its run setting for several milliseconds after the last step pulse before reverting to the stop power setting (**StopPwr**).

Note: When the step pulses stop, whether the motor immediately decelerates to a stop will depend entirely on the inertia of the motor and its load and to some extent on the power setting. The Hyperdrive has no way of recording any possible motor overrun steps.

Note: As a safety measure, the motor speed setting (SetRPM) is reset to zero when an ESC character is received.

Exit (EX)

Exit a running program

Program:	NO
Interactive:	YES
Parameter:	NO

In a networked system there may be several units running individual programs. You cannot use the *Cntrl-C* character to stop the program in a particular Hyperdrive in this case as it will be received and actioned by all units, stopping the running programs in all units.

Once a networked Hyperdrive has been addressed with the **NetOpen** command, giving this command will stop the program in that Hyperdrive only.

Go (GO)

Start a program running

Program:	NO
Interactive:	YES
Parameter:	NO

Begins executing the current program from the first instruction. The program will run until it encounters an **End** instruction or the end of the program.

You can stop the program at any time by sending a *Ctrl-C* keystroke. If the motor is running it will decelerate to stopped at the currently set deceleration rate (**Accel**).

You can also stop the program at any time by sending an *Esc* keystroke. If the motor is running it will stop instantly (depending on the load inertia).

GoTo (GT)

Go to a line number

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

Unconditionally changes the current program flow to continue at the instruction at the specified line number. Use the **GoTo** instruction to alter the normal program flow. All **GoTo** destination addresses must exist before the program will run.

Example:

```
180 Step 1265
190 WaitFor
200 Delay 100
210 goto 270

..
270 SetRPM 1000
.
```

Halt (HL)

Halt immediately

Program:	YES
Interactive:	NO
Parameter:	NO

Can be used in a program to have the same effect as the *Esc* keystroke: the motor will stop instantly.

Example:

```
.
.
170 Halt          ; arrive here for emergency stop
180 Out 4,L       ; turn on warning light
190 goto 190      ; loop forever
.
.
```

JumpIf (IJ)

Jump on input event

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

The **JumpIf** input instruction has a number of parameters that define which input (INx) terminals the instruction applies to, whether the input turning on or turning off initiates the action, and the address to jump to when this particular event occurs. The syntax is:

```
JumpIf <terminal_pair>, <state>, <destination_line_number>
```

The <terminal_pair> parameter can have the values 1 through 4 and selects the corresponding IN1 thru IN4 input screw connectors.

The <state> can be either ON or H, or OFF or L, with ON or H meaning a voltage above 4 volts between the INx- and INx+ terminals. As elsewhere, the case does not matter.

The <destination_line_number> must be a valid program line number. The line number is the program address to jump to when the particular test is true.

On encountering a **JumpIf** instruction the Hyperdrive will select and examine the current state of the specified input terminals. If the state is TRUE then program flow will immediately jump to the given program address. A TRUE state is where, for example, the instruction specifies ON and the between terminal voltage is already above 4 volts.

If the state is FALSE at the instant the instruction is executed, the program simply moves on to the next instructions in sequence. Subsequent changes of the specified terminals is ignored.

Examples:

This example demonstrates how the state of an input can change the speed at which the motor executes the next step profile.

```
.
300 JumpIf 2,on,330    ; if IN2 is driven we can go fast
310 SetRPM 20          ; go slowly
320 goto 340
330 SetRPM 97          ; go quickly
340 Step 1287          ; do the steps
350 WaitFor
.
```

List (LS)

List the current program

Program:	NO
Interactive:	YES
Parameter:	NO

Giving this command will list the current program, if any.

Example:

```
>list
100 Dir         CW
110 Accel       100
120 StepMode    3
130 SetRPM      250
140 Loop        5
150   Step      1600
160   Wait
170   Delay     500
180 ELoop
190 Dir         CCW
200 StepMode    4
210 Loop        20
220   Step      800
230   Wait
240   Delay     1000
250 ELoop
```

Note that any comments have been discarded and that loops are automatically indented.

Load (LD)

Load a stored program

Program:	NO
Interactive:	YES
Parameter:	NO

Giving this command will load a program (previously saved to non-volatile storage with the **Save** command) back into working memory. It will overwrite any existing program. It produces an error message if there is no saved program.

Loop (LP)

Start a program loop

Program:	YES
Interactive:	NO
Parameter:	OPTIONAL

This instruction flags the start of a program loop. There must be a corresponding end loop instruction (**ELoop**). Loops cannot be nested.

If a parameter is included it will specify the number of times the loop will be repeated.

If there is no parameter or it is zero, the loop will execute forever.

Almost every program where the motor operations are repetitive will include a loop.

Example:

```
.
100 SetRPM 300      ; 300 RPM
110 Loop            ; loop forever
120   Dir cw        ; clockwise
130   Step 3200     ; one revolution
140   WaitFor
150   Delay 200     ; 200 mSec pause
160   Dir ccw       ; anti-clockwise
170   Step 3200     ; one revolution
180   WaitFor
190   Delay 200     ; 200 mSec pause
200 ELoop
210 End
```

Here the motor will spin backwards and forwards one revolution forever.

Mode (MD)

Set motor speed control mode

Program:	YES
Interactive:	YES
Parameter:	YES

The Hyperdrive has two speed control modes:

Step Control Mode

This is the default mode. In this mode the motor will not move until it receives a **Step** command that specifies the number of steps to move. Previous commands such as **Accel**, **SetRPM**, **StepMode**, etc will govern how the given number of steps is executed, and its acceleration rate, final speed, etc.

```
>Mode step
```

Use this mode when the motor has a well defined profile or mechanical relationship with the item it is driving. An example here would be where the motor directly drives a screw-jack, and there are a known number of steps required for its travel.

When the Hyperdrive receives a **Step** command the motor starts rotating to fulfil the given profile. In interactive mode the Hyperdrive will accept no further commands until the given number of steps have been completed. In program mode use the **WaitFor** command to pause the program until the step profile concludes, or the **Delay** command to pause the program.

Speed Control Mode

Use this mode where the motor typically runs continuously.

```
>Mode speed
```

The motor will begin to move on receipt of a **SetRPM** command. It will accelerate based on the **Accel** specified rate using the current **Power**, **StepMode**, etc settings until it reaches the given speed. It will then continue to run at this new speed until some other speed control is given. The transition between the old and new speeds is linear.

Commands allowed between **SetRPM** commands are **Accel**, **Delay**, **DecelNow** and the **Out** commands. It is thus easy to create specific speed profiles where this is required. Here is a program fragment:

```
100 Accel   110      ; specify slow accelerate to pre-speed
110 SetRPM  55       ; go to pre-speed (takes 500 mSecs)
120 WaitFor          ; wait till it gets there
130 Accel   1200     ; now use 1200 RPM/Sec^2
140 SetRPM  489      ; go to destination speed (takes 362 mSecs)
150 Delay   5000     ; run for 5 secs
160 SetRPM  0        ; decel to stop – takes 408 mSecs
```

Arm a subsequent jump

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

The **Monitor** input instruction has a number of parameters that define which input terminal pair (INx) the instruction applies to, whether the terminal going from OFF to ON or vice versa initiates the action, and the address to jump to when this particular event occurs. The syntax is:

```
Monitor <terminal_pair>, <transition>, <destination_line_number>
```

The <terminal_pair> parameter can have the values 1 through 4 and selects the corresponding IN1 thru IN4 input screw connector.

The <transition> can be either ON or H, or OFF or L, with ON or H standing for a transition from OFF to ON, and OFF or L for a transition from ON to OFF. In this case ON is a voltage greater the 4 volts between the input terminals. As elsewhere, the case does not matter.

The <destination_line_number> must be a valid program line number. The line number is the program address to jump to when the particular test is true.

Unlike the **JumpIf** instruction, the **Monitor** instruction arms the Hyperdrive to continuously watch the specified input terminals for the specified transition.

If the input state is TRUE when the instruction is executed then the instruction is ignored and program flow simply continues.

If the input state is FALSE however, program flow will still continue, but should that pin state become TRUE at some later time, the jump will occur.

The instruction is typically used to arm a subsequent jump when an event such as a limit switch being reached or perhaps an emergency switch being closed occurs.

Examples:

This example shows a program segment where the motor runs until it is stopped by one or the other of the switches connected to inputs 1 and 2.

```
.
100 Mode speed
110 Power 3                 ; power setting 3
120 StepMode 2              ; step mode 2
130 Accel 600               ; 500 mSecs accel time
140 Monitor 2,on,170        ; go to 170 when Out-2 becomes driven
150 SetRPM 300              ; start motor
160 ContWhen 1,on           ; run until Out-1 becomes driven
170 decelNow                ; decelerate to a stop
.
```

In this case the motor will start up via the **SetRPM** instruction and continue to run forever until either **IN1** or **IN2** becomes ON. The **ContWhen** instruction pauses the program waiting for **IN1** to become ON. However the **Monitor** instruction is continuously monitoring the **IN2** input while the motor runs. The program jump will occur the instant that line changes to the ON state.

In this next example the sensor is sensing a mechanical stop, so if the event occurs there is no time for the normal deceleration phase and the motor must halt immediately. A sample program segment using an OFF to ON transition on pin **IN2** could look like:

```
.
160 SetRPM 80         ; start the motor running
170 Monitor 2,ON,240  ; jump if pin 2 is, or changes to, on
.                     ;
.
240 Halt              ; pin 2 came on: halt immediately
```

An example combination of the two events above would be the situation where there is a normal transition to an ON stop event, perhaps a mass sensor or similar, and an override limit switch with a transition to OFF should the mass switch fail. The set of instructions would look like:

```
.
200 SetRPM 150          ; start the motor profile
210 Monitor 1,ON,280    ; pin 1 monitors for normal stop

220 Monitor 2,OFF,310   ; pin 2 monitors for emergency stop

.
.
280 DecelNow            ; normal stop
290 ContWhen 2,ON       ; clear other monitor
.
.
310 Halt                ; emergency stop
320 ContWhen 1,OFF      ; clear other monitor
.
```

In this case the mass sensor is on pin **IN1** and its output comes ON to switch. The limit switch is on pin **IN2** and it goes from ON to OFF to switch. The **SetRPM** instruction starts the motor running in the previously set speed mode with the previously programmed profile. Either the mass sensor or the limit switch can stop the motor, where a mass sensor event will jump the program to the **DecelNow** instruction which will bring the motor to its normal deceleration stop, but a limit switch event will jump the program to the **Halt** instruction which halts the motor immediately. What happens after the two stop instructions is up to the program.

Note that it is not the **Monitor** instruction itself that changes what happens – it just begins an internal monitor that continuously checks for the TRUE event. It is this internal monitor that then jumps out of that program segment on the programmed event into another where further action can be taken, not necessarily a motor action.

Note the **ContWhen** instructions that clear the monitoring for the other input when a monitored jump occurs. The reason for this is that otherwise the other monitor will still be active and could cause inappropriate behaviour later in the program.

Always be sure all monitors have been cleared when exiting a program segment where they were in use.

Why a Background Monitor must be Cleared

Here is an example of a possible subtle effect when using a background monitor. In this case the instruction is used in a loop that includes a specific delay, but the effect can occur in other situations.

```
.
100 Loop 0              ; start a continuous loop
110   Monitor 1,ON,190  ; monitor pin 1 to stop
120   Step 12000        ; start motor profile
130   WaitFor

   .
   .
190   DecelNow          ; stop motor
200   Delay 500         ; half second delay
210 ELoop               ; repeat the loop
.
```

The first time through if the IN1 input is OFF the monitor is armed and the motor started on its 12,000 steps. The program will then pause at the **WaitFor** instruction. If the IN1 input comes ON during this wait the program will immediately jump to line number 190 and the **DecelNow** instruction will decelerate the motor to a stop. The program then starts the 500 mSec delay.

The effect occurs if the IN1 input remains ON longer than the motor stop time plus the delay time. Now the next run through the loop the **Monitor** instruction will find the input already ON, and as mentioned earlier, monitoring is only enabled if the current state is FALSE. The **Step** instruction will start the motor running and it will run for its programmed 12,000 steps with nothing that happens on the **IN1** line able to affect the program.

You can guard against this sort of program lock by pausing the program with a **ContWhen** instruction until the previous state has re-established itself:

```
.
100 Loop
110   Monitor 1,ON,190   ; monitor for pin 1 turning on
120   Step 12000
130   Wait
   .
   .
190   DecelNow           ; stop the motor
200   Delay 500          ; half second delay
202   ContWhen 1,OFF     ; ensure pin 1 has turned off again
210 ELoop
.
```

In this example if the input is still ON at the end of the delay the program will pause until it switches OFF again, preventing the loop from being re-entered until the input line has the correct sense. You could observe this using the output pin **OUT1**.

Arm a Subsequent Jump (skip current)

Program:	YES
Interactive:	NO
Parameter:	REQUIRED

The **MonSkip** input instruction (monitor but skip current) has a number of parameters that define which input pin pair (**INx**) the instruction applies to, whether the terminal state going from OFF to ON or vice versa initiates the action, and the address to jump to when this particular event occurs. The syntax is:

```
MonSkip <terminal_pair>,<transition>, <destination_line_number>
```

The <terminal_pair> parameter can have the values 1 through 4 and selects the corresponding pair of IN1 thru IN4 input screw terminals.

The <transition> can be either ON or H, or OFF or L, with ON or H meaning a voltage above 4 volts between the INx- and INx+ terminals. As elsewhere, the case does not matter.

The <destination_line_number> must be a valid program line number. The line number is the program address to jump to when the particular test is true.

Unlike the **Monitor** instruction, the **MonSkip** instruction ignores the current input state and arms the Hyperdrive to continuously watch the specified input pin for the specified transition only after it has changed from its current state.

Its purpose is for cases where the current input state must be the state that eventually causes the jump. The **MonSkip** instruction assumes that at some time in the future, this input will first change from its current state to the opposite state, and when (or if) it returns to the current state the jump will occur.

The instruction is typically used to prepare a jump to a stop event when there is only one switch involved.

Example:

Consider a motor feeding labels - the motor is started by an ON level for input **IN2** and must stop when the input **IN1** goes OFF. However in the stopped position **IN1** is already OFF as that was required to stop the motor. In the program below the **ContWhen** instruction halts the program until the **IN2** input turns ON. When the **MonSkip** instruction examines its input it finds the current input is already OFF, the state that should cause the jump. However it will ignore this until the input transitions to ON, and only then will it begin monitoring for the transition to OFF.

```
.
130 Loop
140   ContWhen 2, ON
150   MonSkip 1, OFF, 210
160   Step
170   WaitFor
180   goto 260

.
210   decelNow
220   Delay 10
```

```
        230 ELoop
        .
```

See some examples of the similar **Monitor** instruction.

NetClose (HC)

Close a Networked Hyperdrive

Program:	NO
Interactive:	YES
Parameter:	REQUIRED

Closes this unit to network access. All subsequent instructions on the network will be ignored (see *Connecting Hyperdrives in a Network*).

NetOpen (HO)

Open a Networked Hyperdrive

Program:	NO
Interactive:	YES
Parameter:	REQUIRED

If this unit is in address mode (ie has received an address via the **SetAddr** instruction), this instruction activates the unit to respond to subsequent instructions on the network. Every instruction will be acted upon until a **NetClose** command is received (see *Connecting Hyperdrives in a Network*).

If the unit is in address mode, without this instruction it will ignore all instructions.

If the unit has not been assigned an address this instruction is ignored.

Out (IO)

Setup output parameters

Program:	YES
Interactive:	NO
Parameter	REQUIRED

The **Out** output instruction has two parameters that define which output (OUTx) terminal pair the instruction applies to, and whether the state should switch ON or OFF. There is an optional third parameter which defines a step count that changes the transition event defined by the action from happening immediately to happening sometime in the future after *count* number of steps has occurred.

```
        Out <terminal_pair>, <action> [, <count>]
```

The <terminal_pair> for the KF086B Hyperdrive can have the values 1 and 2, and selects the OUT1 or OUT2 screw connector pairs.

The <action> can be either ON to turn the output on (turns on the output transistor), or OFF to set the output off (turns off the output transistor). On encountering an **Out** instruction during normal program flow the output transistor for the specified output is driven to the required state and program flow continues (see Figure 16).

When the optional third parameter occurs the <count> is recorded at this point in the program and will be counted down during subsequent motor steps. Only when this count reaches zero will the <action> occur.

Examples:

Typically the **Out** instruction is used to indicate a particular segment of a program is being executed. This could be used just as an indication (i.e. turn an indicator light on), or used to control some external process for which the motor is a controlling force (i.e. start another motor, open a valve, close a relay, etc).

```
100 Angle 3600        ; set for a 3.6 degree step motor
110 Accel 200         ; accel/decel rate 200 RPM/Sec^2
120 SetRPM 90         ; max RPM is 90
130 StepMode 2        ; step mode 2 (makes it a 1.8 degree motor)
.
180 Loop              ; start never-ending loop
  .
210   Out 2,ON        ; set OUT2 on
220   Step 18000      ; a 90 revolution profile
230   WaitFor         ; wait until finished
240   Out 2,OFF       ; set OUT2 off again
  .
```

In this example a 3.6° motor has been setup to accelerate at 200 RPM/Sec^2 to a maximum speed of 90 RPM using step mode 2. The latter gives a step angle of 3.6 ÷ 2 or 1.8 degrees. Within the loop the first **Out** instruction switches the **OUT2** output transistor on and this could be used to start an external event that must run while the motor also runs (perhaps it energises a solenoid for example). The following **Step** instruction rotates the motor through 32400 degrees, or 90 rotations (1.8 × 18000 ÷ 360). The **WaitFor** instruction pauses the program until the motor completes the profile. Immediately the motor stops, the second **Out** instruction switches the **OUT2** transistor off again (it would de-energise the solenoid).

The next example is not directly associated with the motor at all. It simply at some point in the program switches the **OUT1** output transistor off for 650 milliseconds.

```
.
172 Out 1,off
174 Delay 650
176 Out 1,on
.
```

When an **Out** instruction includes the optional third count parameter, the action will not occur until a subsequent **Step** instruction has stepped the motor through that many steps. This can be used to initiate some external event at some physical position related to the motor's output system.

```
...
300 Out 1, ON, 3200   ; set output off after 3200 steps
400 Step 6400         ; start a 6400 step profile
500 WaitFor
...
```

For this example the **OUT1** output will turn off half way through the motor movement of 6400 steps at the 3200 step mark. (Note that it implies a fixed mechanical relationship between the motor and whatever it is driving. For instance if the drive includes belts that could slip, etc, then there would be no guaranteed fixed relationship between a specific number of counts and a particular position).

An example might be turning on an indicator after a screw-jack has moved past a specific position. Another might be initiating a saw or guillotine at a certain feed table position.

An interesting option is to synchronise two Hyperdrives so one is waiting in its program for an input transition that is given by the other as it passes a given position. At that point the second Hyperdrive begins its sequence synchronised exactly with the first.

This is implemented by connecting wires from the **OUT1** terminal pair of the initiating or master Hyperdrive to the nominated pair of input terminals on the slave Hyperdrive (IN1 through IN4). At the given step position of the master the **OUT1** line will change state which the slave can detect and act upon.

Note that this is a case where a power source will be required so that the **OUT1** transistor turning on will cause the required 7 milliamps or so to flow through the given **INx** circuit. This could be from the +5V source of either Hyperdrive.

Power (PW)

Set motor torque

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

This instruction defines the motor maximum torque by setting a limit to the current that will be delivered. Legal values are 1 through 4. These values correspond to 20%, 50%, 75% and 100% of the Hyperdrive rated current of 3.5 amperes. The current settings are therefore:

Parameter	Percentage	3.5 amp Unit
1	20%	0.7 amps
2	50%	1.75 amps
3	75%	2.625 amps
4	100%	3.5 amps

The actual peak and average currents will depend on the supply voltage and the resistance and inductance of the particular motor concerned.

Program (PG)

Start loading a program

Program:	NO
Interactive:	YES
Parameter:	NO

This command's primary purpose is as the first line of a program downloaded from a PC or PLC. The intent is to clear any existing program and switch the Hyperdrive to program mode. However, when entering a program by hand there is no real need for this command as text preceded by a line number is automatically assumed to be a program line. In this entry mode both the **Program** and **Clear** commands can be used to clear any existing program.

There are limitations on the rate at which characters can arrive. Too quickly and some characters may be lost. The recommended technique requires that the sending device use a protocol where it sends a line and then waits for the Hyperdrive to echo the ">" character. Packages such as TeraTerm can use a macro that implements this protocol, and an example macro file called `Hyperdrive3.ttl` is included in the TeraTerm folder of the distribution CDROM.

Where this protocol cannot be implemented, inserting a delay of about 30 mSecs between each line will allow sufficient time for the Hyperdrive to process each line and echo any syntax error messages.

All comments are discarded – the Hyperdrive does not store comments.

PSteps (PS)

Print elapsed steps

Program:	NO
Interactive:	YES
Parameter:	NO

This interactive command is only available in STEP mode. It will print the number of steps since the last **Step** command. For example:

```
>SetRPM 50
>Step 3200
>PSteps
2763
>
```

The command is typically used following an emergency situation where an ESC command has been used to immediately stop the motor and the motor must subsequently be reset to the **Step** command start position. It can also be given during the step transition.

Note: an ESC command will automatically reset the motor speed setting (SetRPM) to zero.

Note: When the step pulses stop, whether the motor immediately decelerates to a stop will depend entirely on the inertia of the motor and its load and to some extent on the power setting. The Hyperdrive has no way of recording any possible motor overrun steps.

A typical command sequence for this situation could be:

```
>Dir cw         ; run clockwise
>SetRPM 120     ; at 120 RPM
>Step 7800      ; for 7800 steps
>ESC            ; an ESC occurs at step 3215
>PSteps         ; print the elapsed steps
3215
>Dir ccw        ; go back to start
>SetRPM 120     ; also at 120 RPM
>Step 3215      ; for the same number of steps
>
```

Note that depending on any motor load induced overrun, the reverse sequence may require some additional steps.

Renum (RN)

Renumber a program

Program:	NO
Interactive:	YES
Parameter:	NO

Renumbers the current program starting from line number 100 with a line number spacing of 10.

Reset (RS)

Reset all parameters to power up settings

Program:	NO
Interactive:	YES
Parameter:	NO

The command resets all internal settings to their power up values. See the *Interactive Use* section for a listing. Note that it will also turn off an **RPMCtrl** setting.

RPMCtrl (SC)

Real Time Motor Speed Control

Program:	YES
Interactive:	YES
Parameter:	YES

This instruction modifies the way the inputs IN3x and IN4x respond. They are configured as high speed quadrature inputs after receiving this command. Only power cycling or a **Reset** command will return these inputs to their normal function. Figure 15 shows the typical connection diagram. Each click of an encoder or some other quadrature device either clockwise or counter clockwise will respectively increase or decrease the current motor speed. If your encoder seems to operate in the opposite sense, simply swap the A and B wires.

The instruction requires a parameter that specifies the required speed change increment in RPM. The parameter can be a floating point number and so can provide very fine speed control. The parameter is limited to a maximum increment of 100 RPM.

In **SPEED** mode, each encoder click is the same as giving a **SetRPM** command.

Note that if the motor is stopped it will immediately start to rotate at the increment speed.

For example, after giving the commands:

```
>Accel 1000
>Mode speed
>RMPCtrl 10
```

The first increase click on the encoder will start the motor rotating at 10 RPM. Subsequent increase clicks will run the motor at 20, 30, 40 RPM, etc. You can use a **SetRPM** command to set an initial running speed. Subsequent encoder clicks will modify this speed by the given **RPMCtrl** parameter, with the encoder direction specifying whether the speed increases or decreases.

You can use **RPMCtrl** in **STEP** mode and the increments or decrements made with the encoder will take effect from the start of the next **Step** instruction. Changes made while the motor is actually executing a **Step** command are queued but ignored. They only take effect the next time the **Step** instruction is executed.

RS485

Switch Between RS422 and RS485

Program:	NO
Interactive:	YES
Parameter:	YES

Provides for selection of either RS422 (4-wire) or RS485 (2-wire) serial communication. By default the Hyperdrive is in RS422 mode when shipped. The parameter determines the switch direction. To switch from RS422 to RS485 the command is:

```
>RS485 1
```

To switch from RS485 to RS422 the command is:

```
>RS485 0
```

Note that in most cases, the current setting (RS422 or RS485) will need to match the current serial adapter. *Appendix 3 – Changing the Communications Mode*, describes how a switch is made without using an adapter.

RunUp (AR)

Set Auto-Load and Run

Program:	NO
Interactive:	YES
Parameter:	NO

Configures the Hyperdrive to auto-load and run a stored program the next time it is powered up. The stored program must have been stored in non-volatile storage by a previous **Save** command.

Example:

```
>save
** SAVE... Saved
>RunUp
>
```

Save (Save)

Save a program

Program:	YES
Interactive:	YES
Parameter:	NO

Use this command to write the current program to the Hyperdrive non-volatile storage. The program will then be available to reload via the **Load** command after power has been removed.

A program must first be saved before giving the **RunUp** auto-start command which will automatically execute a Load and then a **Go** command when power is restored.

SetAddr (HA)

Set Hyperdrive network address

Program:	NO
Interactive:	YES
Parameter:	REQUIRED

The parameter can have the value 1 through 9. The instruction assigns that number to this Hyperdrive to work in a network of Hyperdrives where all units share the same RS422/RS485 lines (see *Connecting Hyperdrives in a Network*).

The number for each Hyperdrive must be unique on the network.

This command must be given individually to each unit BEFORE it is connected into the network. The address is included in the output from a **State** command.

The address is automatically saved to non-volatile storage, and is therefore reset automatically should power be lost.

Example:

```
>SetAddr 5
>NetOpen 5
>Step 1200
>
```

SetRPM (MS)

Set motor maximum speed

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

This instruction sets the steady state speed at which the motor will run. It is specified in revolutions per minute (RPM). In step mode (**Mode STEP**) a subsequent **Step** command is required to start the motor moving. Note that the number of steps specified in the **Step** movement profile overrides all other settings, and so a limited number of steps and/or a long acceleration time may not allow this speed to be reached.

When the Hyperdrive is in speed mode (**Mode SPEED**), this command immediately starts the motor running. Once up to speed at the given acceleration rate (**Accel**) the motor will run continuously.

List motor parameters

Program:	NO
Interactive:	YES
Parameter:	NO

Giving this command will list the current motor settings and some system settings that can be very helpful keeping track of the Hyperdrive state during program development.

Example:

```
>State
  Speed        100.00
  Steps        12000
  Accel Rate   100 RPM/Sec^2
  Direction    CW
  Pwr mode     1
  Step mode    4
  Decay mode   2
  Debounce     (1=0), (2=0), (3=0), (4=0)
  Speed mode   Step
  Address      -
  Stop Pwr     1
  RPM Cntrl    No
  Prog Lines   0

>SetRPM 45
>Accel 300
>State
  Speed        45
  Steps        12000
  Accel Rate   300 RPM/Sec^2
  Direction    CW
  Pwr mode     1
  Step mode    4
  Decay mode   2
  Debounce     (1=0), (2=0), (3=0), (4=0)
  Speed mode   Step
  Address      -
  Stop Pwr     1
  RPM Cntrl    No
  Prog Lines   0

>
```

Step (ST)

Set number of steps

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

If the Hyperdrive is in step mode (**Mode STEP**), in both programming and interactive modes this instruction will cause the motor to immediately move the given number of steps using the currently defined set of acceleration, stepping, power, and decay modes. The stepping mode (**StepMode**) will affect the number of degrees a step will take.

If the Hyperdrive is in speed mode (**Mode SPEED**), this instruction is ignored.

StepMode (SM)

Set the step mode

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

This command selects from the four possible step modes available in the Hyperdrive. The available settings are shown in the table. The step angle values shown here assume a 1.8° motor.

Parameter	Microsteps per Step	Step Angle
1	1	1.8°
2	2	0.9°
3	8	0.225°
4	16	0.1125°

```
>StepMode 3
```

StopPwr (SP)

Set motor stopped power

Program:	YES
Interactive:	YES
Parameter:	REQUIRED

Rarely is the power setting required by the motor when stationary the same as when operating. Typically the stopped power is just a fraction of what is required during movement. Use this instruction to set the stopped holding power. The parameter value has the same range as the **Power** command, 1 through 4. By default the setting is 1.

Wait for motor to stop

Program:	YES
Interactive:	NO
Parameter:	NO

When the program executes a **Step** command in step mode and a **SetRPM** command in speed mode, it starts the motor and continues to the next instruction. However in most step cases and some speed cases the program should pause at that point until the profile completes before continuing. This instruction does just that – only proceeding to action the next instruction when the motor comes to a stop in the step case and attains the given speed in the speed case.

See the **Mode** command.

Connecting Hyperdrives in a Network

Up to nine Hyperdrives may be connected in a common serial network, either 4-wire RS422 or 2-wire RS485. Each Hyperdrive is individually given an address and connected into the network. From then on all commands for a particular Hyperdrive must be specifically addressed to that device.

Setting an RS422/RS485 Address

Before connecting a Hyperdrive into a networked multi-unit RS422/RS485 system, you must set each controller's individual address. This is a number between 1 and 9 and is used by the controlling PC or PLC to select a particular Hyperdrive.

Connect the unit to the PC as for individual use. Program and test the system as per normal with that unit connected to a unique driver. Use the **SetAddr** command to assign the unit's address once everything is satisfactory. It can then be connected into the network.

Once addressed, you can see the address listed in the **State** command output.

Once connected, instructions for each Hyperdrive in the network must be bracketed by the **NetOpen**/**NetClose** commands to open then close access to that unit. Unless the **NetOpen** command is received, any and all instructions will be ignored.

After receiving its respective **NetOpen** command, that Hyperdrive will respond to instructions as normal.

Commanding Individual Hyperdrives

The following example command sequence from a machine controller shows two Hyperdrives that were previously assigned the addresses 3 and 5 and are now connected into a network. It shows the two motors being setup and then commanded to run at their respective operating speeds. Note how each set of commands are bracketed by their respective **NetOpen**/**NetClose** command pairs.

```
Prepare both motors
NetOpen 3           ; select motor A
  Mode SPEED        ; speed mode
  Accel 600         ; accel rate
  StepMode 2        ; step mode 2
NetClose 3          ; deselect
NetOpen 5           ; select motor B
  Mode SPEED        ; speed mode
  Accel 1500        ; accel rate
  StepMode 4
NetClose 5          ; deselect

Now start both motors

NetOpen 3           ; select motor A
  SetRPM 320        ; start it running
NetClose 3
NetOpen 5           ; select motor B
  SetRPM 60         ; start it running

NetClose 5
```

Checking the Motor State while Networked

Step Mode

The **Step** command works somewhat differently when in network mode. The '>' prompt character normally sent when the motor concludes the requested number of steps is now only sent if that Hyperdrive is selected. So to observe the end of a **Step** move, that particular unit must have been selected with a **NetOpen** command.

To provide for the situation where several Hyperdrives are executing move profiles at the same time, and the computer or PLC controlling the network needs to know the state of its current profile, the response to a **NetOpen** command differs depending on whether the motor is running or not.

If the motor is still running the prompt will be the '}' character. If the motor has stopped it will be the usual '>' character.

Thus a controller can probe for the motor running state of a particular Hyperdrive by checking the returned prompt character in response to the **NetOpen** command. In the following STEP mode example, the motor on Hyperdrive number 5 is still busy during the first two probes, but has stopped by the time the third probe is made.

```
>NetOpen 5
}NetClose 5
.
.
>NetOpen 5
}NetClose 5
.
.
>NetOpen 5
>
```

The controlling PLC could use repeating sequences of **NetOpen**/**NetClose** commands to probe several Hyperdrives for when their associated profile has concluded.

Note that if the particular Hyperdrive remains selected, the usual '>' character is sent when the motor eventually stops. In the following example, Hyperdrive number 3 has previously begun a profile and after some other processing, the controller returns to wait for that profile to conclude. The '}' prompt is sent in response to the switch via the **NetOpen** command, and some time later the '>' prompt will signal the profile's end.

```
>NetOpen 3
}
>
```

Speed Mode

The same mechanism can be used when the Hyperdrive is in SPEED mode. For the situation where the motor must now be stopped and the controller not move on until it is sure the motor has come to rest, use **NetOpen** commands while monitoring the prompt character. In this example Hyperdrive number 7 has been running at 230 RPM and must now be stopped. The controller simply loops on the **NetOpen** command until the correct response is received before proceeding.

```
>NetOpen 7
}SetRPM 0
}NetOpen 7
```

```
}NetOpen 7
}NetOpen 7
>
```

If more than one Hyperdrive has been addressed by having received their respective NetOpen commands (i.e. several units are 'open'), then subsequent instructions will be received and actioned by all the units so addressed. While this may sometimes be very useful, be aware that all units so addressed will respond with their normal prompt character sequence and due to slight time differences, the controller will almost certainly receive corrupted prompt characters.

The single character control codes Esc, Ctrl-C and Ctrl-X will affect **ALL** Hyperdrives in a networked set. They cannot be addressed to a particular Hyperdrive.

Sending a Ctrl-C character will therefore decelerate all Hyperdrives at their programmed rate. Sending an Esc character will halt all Hyperdrives immediately.

Sending a Ctrl-X will clear any 'Run from non-volatile storage' setting in all connected Hyperdrives.

Loading a Program in Addressed Mode

After the **NetOpen** command a program can be downloaded in the normal way, using the Macro command in TeraTerm or some other programmed method that waits for the '>' prompt character after each line is sent (see *Downloading Programs*).

However, be aware that inadvertently sending a program to a network of Hyperdrives where none have received a **NetOpen** command means the first line will be sent (and ignored by all), but if the sending program then waits for the prompt it will wait forever.

Also if some sending program is waiting like this, a **NetOpen** command to any Hyperdrive will suddenly allow the sender to continue, possibly sending a truncated program to probably the wrong Hyperdrive.

Firmware Update

The Hyperdrive SPL language is always under review with new instructions being developed to suit customer needs. The Hyperdrive includes the ability to field upgrade its firmware to later versions. These are published from time to time on the Kremford web site or available for specialist applications.

The distribution CDROM that comes with the Hyperdrive includes a program called Hyper3Update to manage firmware upgrades in the field.

> Do not confuse the HyperUpdate program supplied with the original KF083 Hyperdrives with the Hyper3Update program provided with later KF086 Hyperdrives.

The procedure is simple and uses the standard RS422 serial interface currently used for Hyperdrive programming and control.

> Performing an update will erase any saved program from non-volatile storage. All other settings remain the same.

Upgrader Installation

Navigate to the *Upgrader* folder on the distribution CDROM and run the Setup program. This installs the Hyper3Update program on your PC.

A Firmware Update

From time to time you should compare the firmware revision on your Hyperdrive with the latest as published on the Kremford web site. The site includes details of what the update contains – whether it includes new features or is just a bug fix.

If you decide to install the update, the first step is to download it to a convenient folder on your PC. Kremford recommend you create a folder specifically for Hyperdrive updates. The following procedure assumes you have downloaded the update and the Hyperdrive is connected to your PC via the standard USB/RS422 interface.

Power up the Hyperdrive and note the current revision number. This has three parts: the first number is the major release number, the second a minor release number and the third a revision number.

```
Kremford Hyperdrive3
Revision 1.4.11
Ready
2
>
```

Power down the Hyperdrive and close the application you were using to communicate with the Hyperdrive (TeraTerm).

> Only the Hyper3Update program can be communicating with the Hyperdrive.

Remove the two screws and lift off the Hyperdrive cover. With the connectors facing you, locate the **H1** link on the left-hand side of the PCB. It should have a shorting link attached to just one of the two pins.

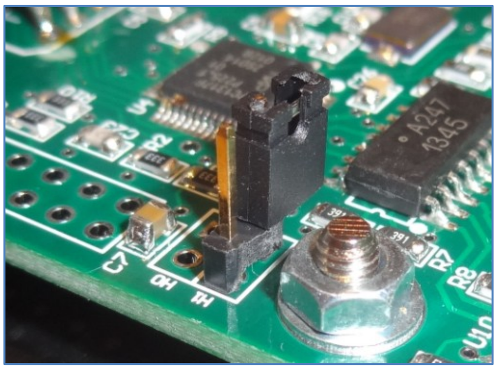
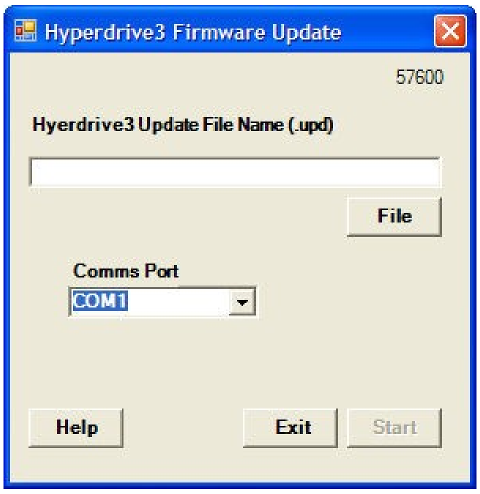


Figure 25. The H1 firmware upgrade link shown in its normal operating position. Remove and replace shorting both pins to enable a firmware upgrade. Remove and replace like this when the upgrade is complete.

Ensure your body is at the same potential as the negative power lead. Now lift the shorting link and replace it so the two **H1** pins are shorted by the link.

Start the Hyper3Update program in Windows by clicking on *Start*, *All Programs*, *Kremford*, and then *Hyper3Update*. You should see a screen similar to Figure 26.


Hyperdrive3 Firmware Update

57600

Hyerdrive3 Update File Name (.upd)

Comms Port: COM1

Figure 26. Initial HyperUpdate screen.

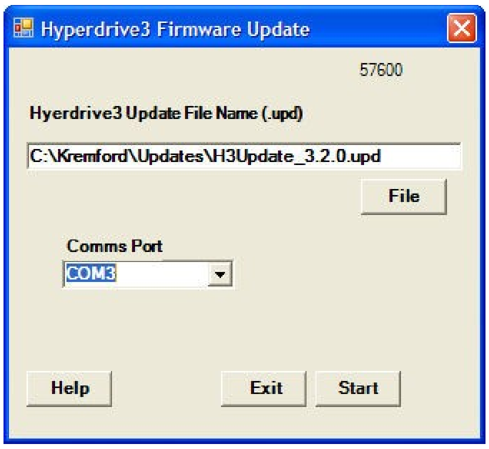
Click on the *Help* button to see a summary of this procedure.

Click the *File* button and navigate to the folder where you saved the firmware update. The filename format is **`H3Update_x.x.x.upd`**. Here we will assume a dummy update file called **`H3Update_3.2.0.upd`** as a demonstration.

Click the *Comms Port* down arrow to select the serial port your Hyperdrive serial interface is using. If you were using TeraTerm then in that program the port is listed under the *Setup* menu entry called *SerialPort*...

Make sure no other application is using that particular COM port.

The ready-to-start Hyper3Update screen should look similar to Figure 27, with the *Comms Port* box showing your serial port selection.


Hyperdrive3 Firmware Update

57600

Hyerdrive3 Update File Name (.upd)

C:\Kremford\Updates\H3Update_3.2.0.upd

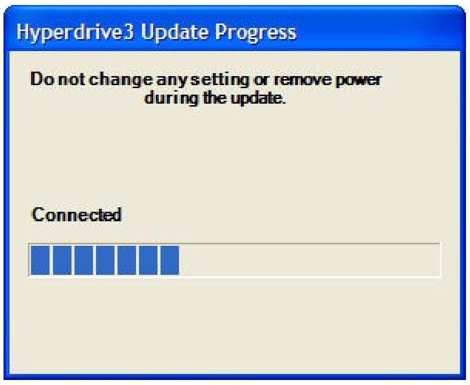
Comms Port

COM3

Figure 27. Hyper3Update ready to execute a firmware update.

Prepare for the next step. The sequence is to click the Hyper3Update *Start* button and then power up the Hyperdrive. Once you press the *Start* button Hyper3Update will try to contact the Hyperdrive for about 30 seconds before quitting, so you have plenty of time to power up the Hyperdrive.

Within a few seconds of applying power you should see Hyper3Update connect to the Hyperdrive and start the update. This will take a little while and the progress will look similar to Figure 28


Hyperdrive3 Update Progress

Do not change any setting or remove power during the update.

Connected

Figure 28. The Hyper3Update program showing an update in progress.

When the update completes dismiss the Hyper3Update application, restart your communications program (TeraTerm), and power cycle the Hyperdrive. You should see the usual startup text but now the firmware version number should match that of the update file.

The Hyperdrive is now ready to use. Note you may need to now change the baud rate setting of your communication software if you had previously set the communications baud rate to other than the as-shipped 9600 baud.

The update program performs several checks during the update to ensure the programming is proceeding successfully. However glitches can occur and if so, when the Hyperdrive powers up again your will see the message:

```
CRC Error
```

In this case simply repeat the programming procedure described above and you should see the standard hello message. Should the error persist please contact Kremford.

Messages

There are a number of messages that can be printed in response to command entry. The response will be obvious where the commands are being entered by hand. However careful testing will be required where an external computer is using the interactive mode to control the Hyperdrive to ensure all instructions/commands are valid.

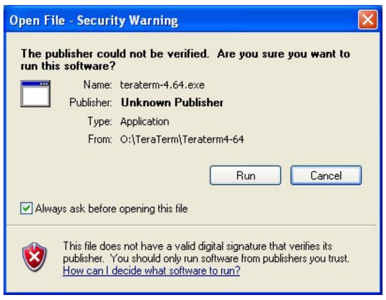
Message	Explanation
Aborted	Either a Ctrl-C or ESC keystroke/character was entered and the current program was aborted.
Bad H or L	A character other than H or L was entered where only these were expected.
Bad Out pin	For an Inx instruction a channel number outside the range 1 to 4 was entered, or a number other than 1 or 2 was entered for an Outx instruction.
Bad action	A character other than C, S or A was entered as the action for an Inx instruction.
Command is illegal	The given command is unknown.
Embedded loops	Every Loop command must be followed by an ELoop command. A loop cannot be embedded within another loop.
No loop entry	An ELoop command has occurred without a previous Loop command.
No program	The given command is only applicable within a program.
Parameter is illegal	The parameter entered for a command was either not in the correct form, included letters when numbers were expected or vice versa, or were outside the legal range for that command.
Program near full	The number of steps in a program is approaching the limit. The current program limit is 100 lines.
Program Full. Forced End	The program has reached the number of lines limit. The program has been terminated with a forced End instruction and programming mode aborted.
Syntax error	The instruction or command has an unexpected (and consequently illegal) structure or syntax.
Too fast for Step Mode	An RPM setting was entered that was outside the limits defined in the Motor Characteristics table for this step mode.
Saved	The program was successfully saved to permanent storage.
Illegal address here	Destination addresses are not allowed in Interactive Mode.
Only in program mode	This instruction type is only allowed in Program Mode.
Label missing	The End instruction was entered but one or more destination labels are still undefined.
EEPROM erased	The RunUp "run on power up" option has been turned off.
Only 0 thru 50	The entered debounce time is outside these limits.
No matching line	The line to be deleted does not exist.
No speed	A Step command was given but current SetRPM setting is 0.
Wrong Mode	A PS or similar command when not in STEP mode.
Not allowed	This instruction is not allowed in speed mode with the motor turning.

Appendix 1 – Installing the TeraTerm Terminal Program

To install this program, navigate to the CDROM and double click on the file called:

teraterm-4.64.exe

Windows will bring up its usual *Open File Security Warning*. Click on *Run*.



Open File - Security Warning

The publisher could not be verified. Are you sure you want to run this software?

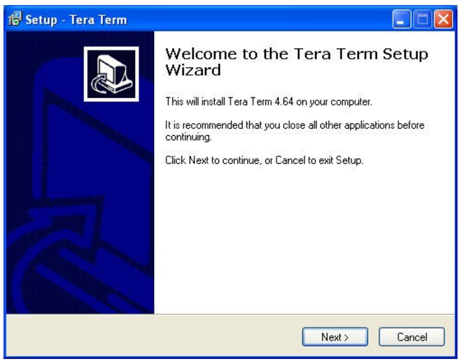
Name: teraterm-4.64.exe
Publisher: **Unknown Publisher**
Type: Application
From: O:\TeraTerm\Teraterm4-64

Run | Cancel

[✓ Always ask before opening this file]

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
How can I decide what software to run?

The TeraTerm installation program will start running and display the *Welcome* dialog shown in Figure 29. Click on the *Next* button.



Setup - Tera Term

Welcome to the Tera Term Setup Wizard

This will install Tera Term 4.64 on your computer.

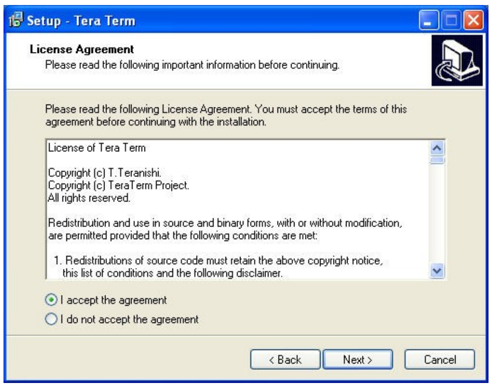
It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Next > | Cancel

Figure 29. TeraTerm Welcome screen.

The *License Agreement* dialog shown in Figure 30 is displayed. If you agree with the terms then select that button and click on the *Next* button.



Setup - Tera Term

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

License of Tera Term

Copyright (c) T.Teranishi.
Copyright (c) TeraTerm Project.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

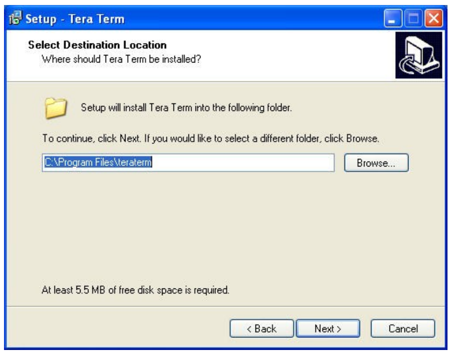
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

[✓ I accept the agreement]
I do not accept the agreement

< Back | Next > | Cancel

Figure 30. TeraTerm License Acceptance dialog.

The next dialog shown in Figure 31 indicates the default location where TeraTerm is stored. The default is reasonable but can be changed to suit your equipment. Click on the *Next* button.



Select Destination Location
Where should Tera Term be installed?

Setup will install Tera Term into the following folder.

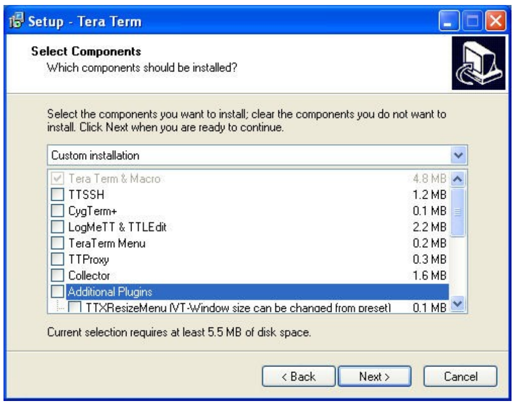
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\teraterm

At least 5.5 MB of free disk space is required.

Figure 31. TeraTerm installation location.

The next dialog lists a considerable number of optional programs and capabilities which TeraTerm can use. You can explore these options if necessary by accessing the TeraTerm web-based Help page at http://ttssh2.sourceforge.jp/manual/End/. However, only the basic terminal program is required for use with the Hyperdrive. Kremford recommends un-ticking all the options as shown in Figure 32. Click the *Next* button to proceed.



Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

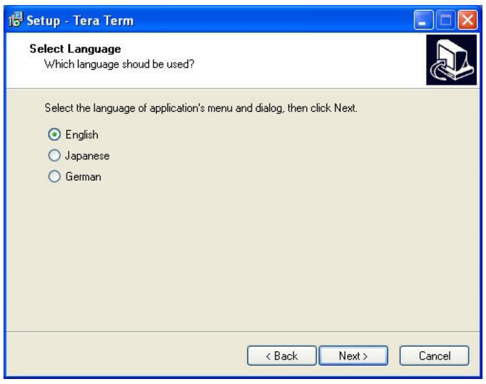
Custom installation

✓ Tera Term & Macro	4.8 MB
☐ TTSSH	1.2 MB
☐ CygTerm+	0.1 MB
☐ LogMeTT & TTLEdit	2.2 MB
☐ TeraTerm Menu	0.2 MB
☐ TTProxy	0.3 MB
☐ Collector	1.6 MB
☐ Additional Plugins	
☐ TTXResizeMenu (VT-Window size can be changed from preset)	0.1 MB

Current selection requires at least 5.5 MB of disk space.

Figure 32. TeraTerm optional accessories and programs.

The next dialog allows a selection from three possible languages. Select a suitable language and click the *Next* button.

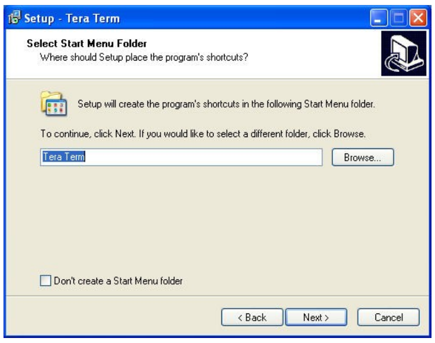


Select Language
Which language shoud be used?

Select the language of application's menu and dialog, then click Next.

[✓ English]
Japanese
German

The next dialog selects the default shortcut folder. Click the *Next* button.



Select Start Menu Folder
Where should Setup place the program's shortcuts?

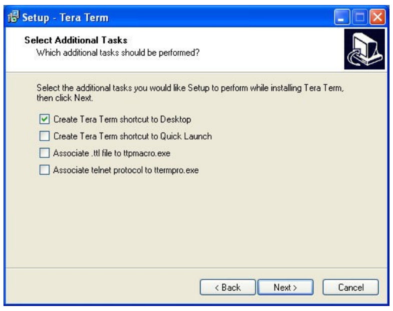
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Tera Term

☐ Don't create a Start Menu folder

The next dialog shown in Figure 33 allows you to select how you can access/start this program. Typically a desktop icon is adequate as shown, but select others as required. Click the *Next* button to continue.



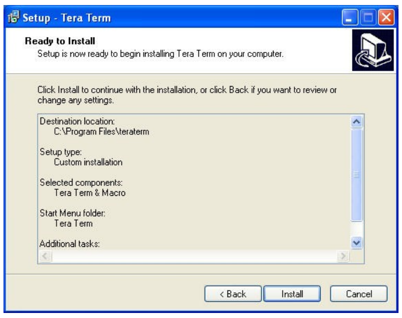
Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Tera Term, then click Next.

- ☑ Create Tera Term shortcut to Desktop
- ☐ Create Tera Term shortcut to Quick Launch
- ☐ Associate .ttl file to ttpmacro.exe
- ☐ Associate telnet protocol to ttermpro.exe

Figure 33. TeraTerm program links.

The next dialog screen lists a summary of the installation sequence. Click the *Install* button to do the software installation.



Ready to Install
Setup is now ready to begin installing Tera Term on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\teraterm

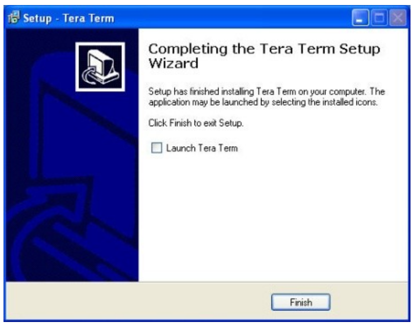
Setup type:
Custom installation

Selected components:
Tera Term & Macro

Start Menu folder:
Tera Term

Additional tasks:

When the installation is complete the final dialog is displayed. Click on the *Finish* button.



Completing the Tera Term Setup Wizard

Setup has finished installing Tera Term on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

☐ Launch Tera Term

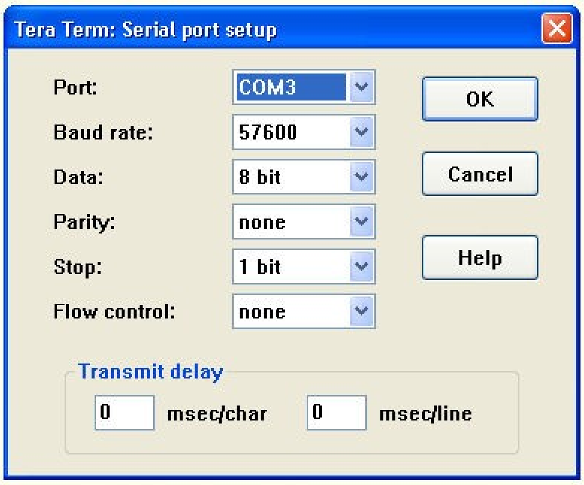
Once installed, double-click on the TeraTerm icon to start the terminal program. Click the *Setup* menu entry. Drag the cursor down to the *Serial* port entry and click on that. This will display the *Serial* port setup dialog shown in Figure 34. The settings as shown match the Hyperdrive as shipped. These are:

Baud rate	9600
Data	8
Parity	None
Stop	1 bit
Flow control	None

Note that Hyperdrives can be reconfigured for 57600 baud with the **SetBaud** command..

The *Port* entry will naturally need to match the connection specific for this computer.

Figure 34 shows a typical TeraTerm serial port setup dialog. Note this example shows the *Transmit Delay* settings. Both these entries can be zero if you use the macro method for program downloading as described in *Downloading Programs* or you use a PLC program that can wait for the '>' ready response from the Hyperdrive.

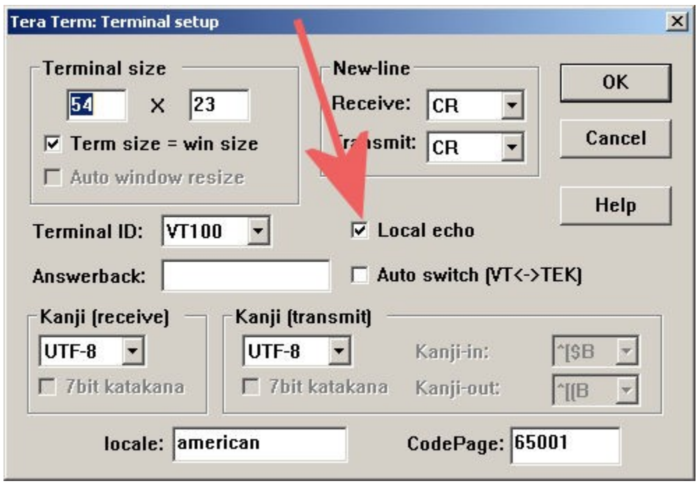

Tera Term: Serial port setup

Port: COM3
Baud rate: 57600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Figure 34. TeraTerm Serial port setup dialog.

The Hyperdrive does not echo the characters you type, so to see what you are typing you must set TeraTerm to locally echo each keystroke. Click on the *Setup* menu entry and drag the cursor to the *Terminal...* entry. This dialog is shown in Figure 35. Ensure the *Local echo* box is ticked as shown.


Tera Term: Terminal setup

Terminal size
54 x 23
☑ Term size = win size
☐ Auto window resize

New-line
Receive: CR
Transmit: CR

Terminal ID: VT100
Answerback:
☑ Local echo
☐ Auto switch (VT<->TEK)

Kanji (receive)
UTF-8
☐ 7bit katakana

Kanji (transmit)
UTF-8
☐ 7bit katakana
Kanji-in: ^[$B
Kanji-out: ^[(B

locale: american
CodePage: 65001

Figure 35. Setting TeraTerm for local echo mode.

You can leave most other TeraTerm options at their default settings although it is often worthwhile to change the window aspect ratio to better suit the Hyperdrive programming environment. You might try changing the Terminal size to 40 x 60.

The setup can be saved on the computer so that whenever TeraTerm is started this set of parameters are automatically loaded. Click on the *Setup* menu entry and drag the cursor down to the *Save Setup* entry. Click on this, select the default TERATERM.INI file, and click on the *Save* button. Now whenever TeraTerm is started it will be setup for an interface with the Hyperdrive controller.

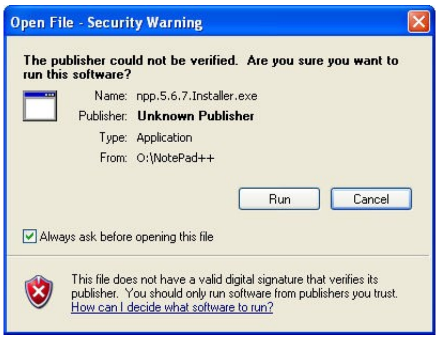
In the same directory on the CDROM is the Hyperdrive file download macro file called **`Hyperdrive3.ttl`** (see *Downloading Programs*). Copy this to the same directory you will use for your own Hyperdrive programs.

Appendix 2 – Installing the Notepad++ Program

To install this program, navigate to the CDROM and double click on the file called:

npp.5.6.7.Installer.exe

Windows will bring up its usual *Open File Security Warning*. Click on *Run*.



Open File - Security Warning

The publisher could not be verified. Are you sure you want to run this software?

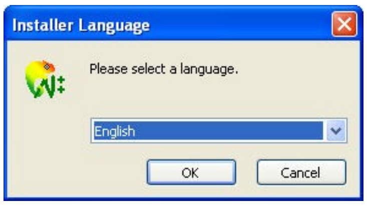
Name: npp.5.6.7.Installer.exe
Publisher: Unknown Publisher
Type: Application
From: O:\NotePad++

Run | Cancel

☑ Always ask before opening this file

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust. How can I decide what software to run?

The Notepad++ installation program will next ask for your preferred language. Select from the drop-down list and click *OK*:



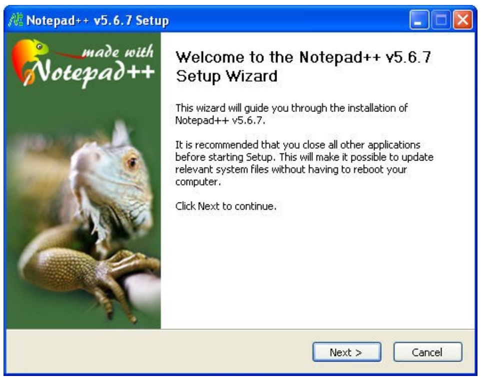
Installer Language

Please select a language.

English

OK | Cancel

The install wizard will start. Click *Next* to continue:



Notepad++ v5.6.7 Setup

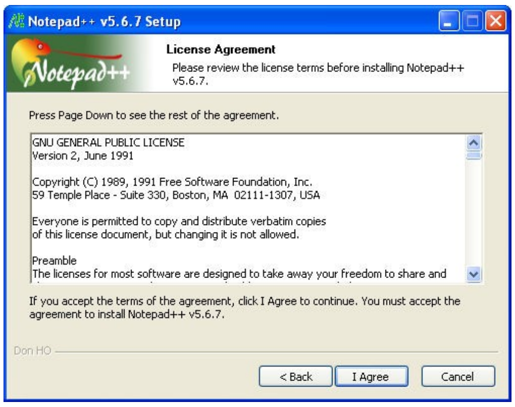
Welcome to the Notepad++ v5.6.7 Setup Wizard

This wizard will guide you through the installation of Notepad++ v5.6.7.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Next will be displayed the license agreement. Click *I Agree* if you are happy with the license:



Notepad++ v5.6.7 Setup

License Agreement

Please review the license terms before installing Notepad++ v5.6.7.

Press Page Down to see the rest of the agreement.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

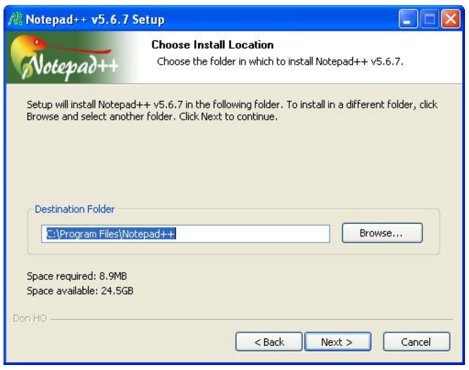
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble
The licenses for most software are designed to take away your freedom to share and

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Notepad++ v5.6.7.

Next choose an installation location. The default is usually satisfactory. Click *Next*:


Notepad++ v5.6.7 Setup

Choose Install Location

Choose the folder in which to install Notepad++ v5.6.7.

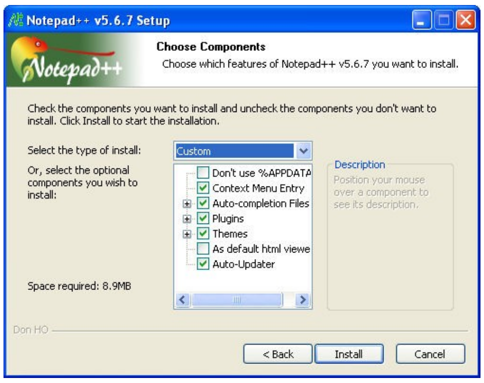
Setup will install Notepad++ v5.6.7 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Program Files\Notepad++

Space required: 8.9MB

Space available: 24.5GB

The next dialog asks you to check which options you would like installed with the Notepad++ editor. The suggestion is to select the default.


Notepad++ v5.6.7 Setup

Choose Components

Choose which features of Notepad++ v5.6.7 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Install to start the installation.

Select the type of install: Custom

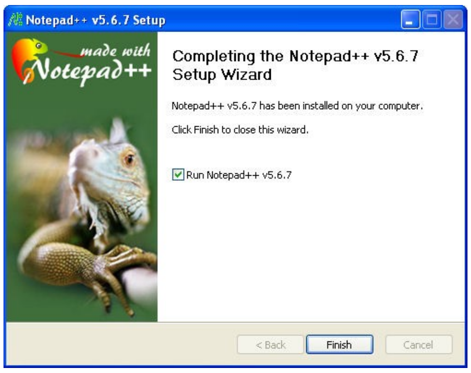
Or, select the optional components you wish to install:

- ☐ Don't use %APPDATA
- ✓ Context Menu Entry
- ✓ Auto-completion Files
- ✓ Plugins
- ✓ Themes
- ☐ As default html viewe
- ✓ Auto-Updater

Space required: 8.9MB

Description: Position your mouse over a component to see its description.

Click the *Install* button and Notepad++ will be installed.


Notepad++ v5.6.7 Setup

Completing the Notepad++ v5.6.7 Setup Wizard

Notepad++ v5.6.7 has been installed on your computer.

Click Finish to close this wizard.

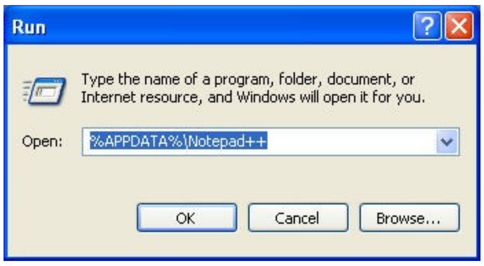
✓ Run Notepad++ v5.6.7

Configure Notepad++ for the Hyperdrive Programming Language

Notepad++ can be used quite satisfactorily as it is installed. However Kremford have included on the CDROM a Notepad++ Language File that enables syntax highlighting for Hyperdrive programs where commands and text are highlighted in colour.

Follow these steps to install this file.

Click on *Start->Run*. In the Open text box, type "**`%APPDATA%\Notepad++`**" as shown in Figure 36 and click the *OK* button.

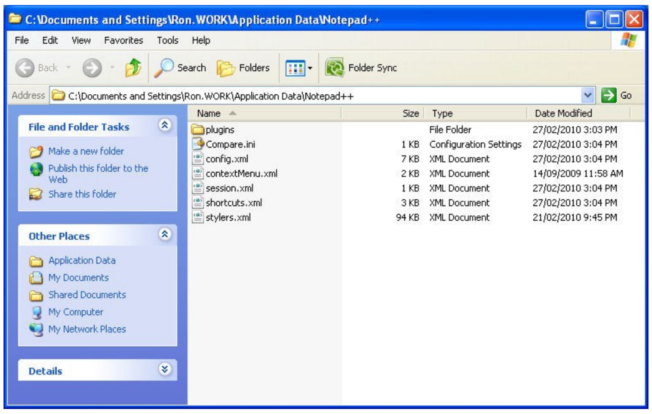

Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: %APPDATA%\Notepad++

Figure 36. Accessing the Notepad++ Application Data section of Documents and Settings for the Current User.

This will display the contents of the Notepad++ Application Data section of the Documents and Settings directory for the current user and, depending on your version of Windows, should look similar to Figure 37.

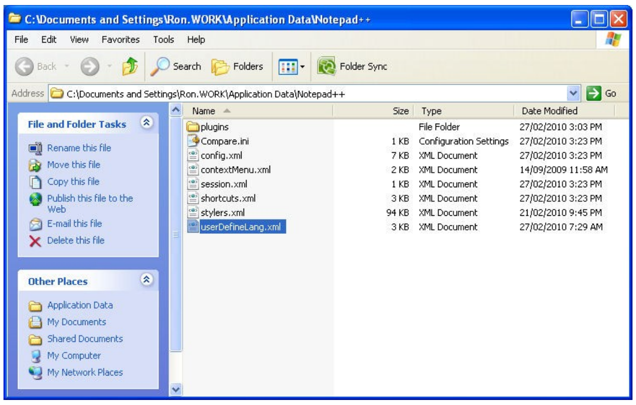

C:\Documents and Settings\Ron.WORK\Application Data\Notepad++

Address: C:\Documents and Settings\Ron.WORK\Application Data\Notepad++

Name	Size	Type	Date Modified
plugins		File Folder	27/02/2010 3:03 PM
Compare.ini	1 KB	Configuration Settings	27/02/2010 3:04 PM
config.xml	7 KB	XML Document	27/02/2010 3:04 PM
contextMenu.xml	2 KB	XML Document	14/09/2009 11:58 AM
session.xml	1 KB	XML Document	27/02/2010 3:04 PM
shortcuts.xml	3 KB	XML Document	27/02/2010 3:04 PM
stylers.xml	94 KB	XML Document	21/02/2010 9:45 PM

Figure 37. Default contents of the Hyperdrive++ Application Data directory.

Start an Explorer window and navigate to the Hyperdrive CDROM. Select and drag and drop the file called **`userDefineLang.xml`** from the CDROM to the Notepad++ Application Data directory. The contents should now look like Figure 38.


C:\Documents and Settings\Ron.WORK\Application Data\Notepad++

Address: C:\Documents and Settings\Ron.WORK\Application Data\Notepad++

Name	Size	Type	Date Modified
plugins		File Folder	27/02/2010 3:03 PM
Compare.ini	1 KB	Configuration Settings	27/02/2010 3:23 PM
config.xml	7 KB	XML Document	27/02/2010 3:23 PM
contextMenu.xml	2 KB	XML Document	14/09/2009 11:58 AM
session.xml	1 KB	XML Document	27/02/2010 3:23 PM
shortcuts.xml	3 KB	XML Document	27/02/2010 3:23 PM
stylers.xml	94 KB	XML Document	21/02/2010 9:45 PM
userDefineLang.xml	3 KB	XML Document	27/02/2010 7:29 AM

Figure 38. The Kremford userDefineLang.xml file copied from the CDROM to the Notepad++ Application data directory.

Close both windows. The next time you load a Hyperdrive SPL file into Notepad++ the syntax colouring option will be enabled. You can confirm the language file was loaded by clicking on the *Language* menu and verifying the Hyperdrive entry exists.

Appendix 3 – Changing the Communications Mode

If the Hyperdrive is in its default RS422 mode and you plug in an RS485 interface connected as per Figure 9, the terminal program will begin to continuously print an error message. This is because everything the Hyperdrive transmits will be fed back in as input, and nothing will look like valid commands. In this situation you will not be able to give the required **RS485 1** command to make the switch to the required communication mode.

To cover this situation there is a boot up procedure that uses the internal **H1** link to indicate to the program that a protocol switch is required.

Re-Boot Procedure

The procedure requires lifting off the cover and working with the **H1** shorting link.

1. Power down the Hyperdrive.
2. Remove the two screws and lift off the Hyperdrive cover. With the connectors facing you, locate the **H1** link on the left-hand side of the PCB (see Figure 25). It should have a shorting link attached to just one of the two pins.
3. The procedure requires first powering up the Hyperdrive and within 10 seconds, slipping the shorting link over both **H1** pins. You might like to practise getting the link in place a few times before actually applying power. When you are confident, move on to the next step.
4. Power up the Hyperdrive.
5. Within a few seconds, short the **H1** link.
6. Shortly after, the “heatbeat” LED should switch on and stay on.
7. When the LED turns off the protocol switch has been made.
8. Power down the Hyperdrive.
9. Remove the shorting link and replace it on just one of the **H1** pins for storage.
10. Connect the new interface and power up the Hyperdrive. It will start in the opposite mode to what it was before this procedure.
11. You can repeat this procedure any number of times. Each time the communication protocol will alternate between RS422 and RS485.

Notes

The internal program starts a background task that runs for 10 seconds every time power is applied. During that delay the **H1** link is examined every second. When it finds it shorted it will turn on the LED. After the delay concludes the **H1** link is examined again and, if still shorted, the current protocol mode is swapped. If the LED was on it will be turned off.