

<dalf1_gs.doc>

DALF – 1; Rev F Motor Control Board

Getting Started Manual

Revision 0.09

Aug 25, 2007

Table of Contents

1	OVERVIEW	2
2	THE DALF-1F BOARD FEATURES	3
3	WHAT ARE YOUR NEEDS?	4
4	SERIAL PORT	6
5	MOTOR POWER DRIVER SETUP	14
6	MOTOR SETUP	15
7	ENCODER SETUP	15
8	POT SETUP	17
9	R/C SETUP	18
10	I/O EXPANDERS	20
11	CODE DEVELOPMENT	21

Warranty

The software libraries and tools are provided "as is" without warranty. The entire risk for the results and performance of these libraries and tools is assumed by the purchaser. Embedded Electronics LLC does not warrant, guarantee or make any representation regarding the use of this product. No other warranties are made, expressly or implied, including, but not limited to, the implied warranties of merchantability and suitability of products for a particular purpose. In no event will Embedded Electronics be held liable for additional damages, including lost profits, lost savings or other incidental or consequential damages arising from the use or inability to use Embedded Electronics LLC products.

Disclaimers

Embedded Electronics LLC reserves the right to make changes without notice to this product. Changes made to improve reliability, performance, capabilities, design or ease of use, or to reduce size or cost could effect documentation, hardware, and firmware. Any Embedded Electronics LLC product may not be used as components in life support devices of any description.

Copyright

Copyright 2006 Embedded Electronics, LLC.

Software License

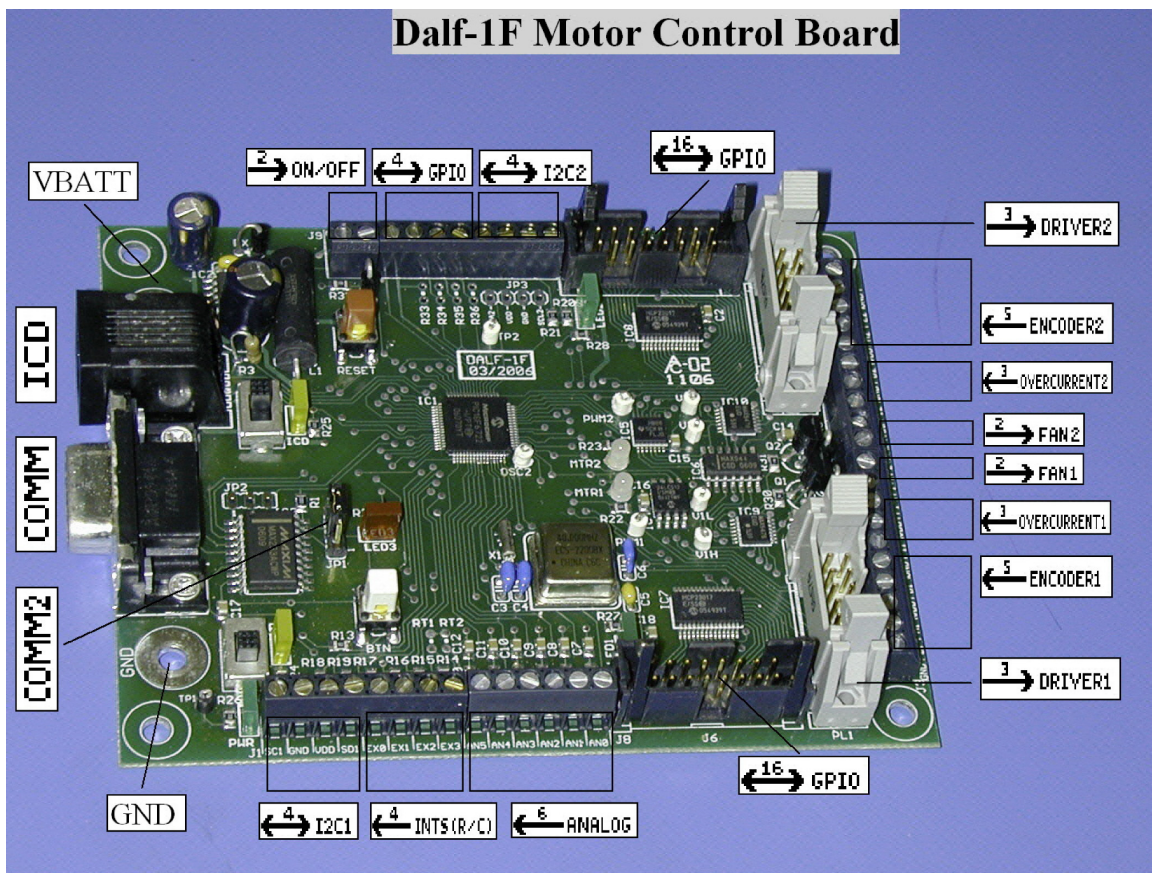
The <main.c> and <dalf.lib> files provided on the CD for the purpose of encouraging additional software development are subject to a license agreement explained in the End User License Agreement (EULA). The EULA file is included on the CD that ships with the product and may also be read on the EE Website <http://www.embeddedelectronics.net/>

1 Overview

This document is designed to get you started quickly using the Dalf-1F Motor Control Board. There are other documents: *Dalf Owner's Manual*, *Dalf API Interface*, *Dalf I2C Interface* that you should be aware of, but using this document as a guide will enable you to get started without reading the owner's manual in a cover-to-cover fashion.

Updated documentation is maintained on the web at <http://www.embeddedelectronics.net/>

Because there are many different ways to use the Dalf Board, a concise Getting Started Manual is a challenge. You may have purchased the board for use as a development platform and have no need of the motor control features. Even so, you may wish to know how to leverage existing board firmware into your design. You may have no interest in code development, but have purchased the board strictly for its built-in motor control features. In this case, you may be focused on open loop motor control operations. Alternatively, your application may need the closed loop motor control features. Finally, there are multiple user interfaces to contend with. Do you want to control your motors with R/C? Do you want to control your motors with potentiometers? Do you want to control your motors with a PC Interface - either from a terminal emulator or a "smart" PC Application? How do you make the required connections? These situations all need to be addressed in the limited space of this document.



If you have questions, contact Embedded Electronics LLC.
email: support@embeddedelectronics.net

2 The DALF-1F Board Features

The board is designed around the PIC18F6722 microcontroller running at 40MHz (10MIPS). This board features lots of available I/O routed to screw terminals and ribbon connectors for easy access. Some of the IO is dedicated to supporting off board features, but there is a lot of unused I/O for customization. This includes analog inputs, interrupt inputs, GPIO's directly from the microcontroller, and 32 GPIO's provided by I2C I/O expanders!

Partial Feature List

- Dual open loop motor controls via R/C, Pots, or Serial Interfaces.
- Dual closed loop motor positional and velocity controls via Serial Interfaces.
- PID and Trajectory Generator employed for smooth velocity ramping in closed loop controls.
- Standard quadrature encoder inputs (2 motors) allow maintenance of position and velocity.
- Analog motor position feedback supported for some applications.
- Giant Servo modes (PotSvo, RcSvo) using either Pot or R/C inputs.
- Digitally adjustable voltage windows and interrupt handlers for fast over current response when used with off board current sensors (two over current response modes).
- Support for PID "Tuning" via data capture using the "Step Response Command".
- Adjustable Slew Rate for all motor control methods.
- Serial EEPROM (64K bytes) as well as internal EEPROM (1K bytes) for non-volatile storage.
- Parameter Block in EEPROM for customization of motor characteristics, operating mode, etc.
- Built in three channel R/C interface with optional mixing.
- Three built in potentiometer motor control methods (PotC, PotF, PotMix).
- Terminal Emulator command line monitor using RS232 Channel 1.
- API command and monitor Interface using RS232 Channel 1 for a "smart" PC Application (*).
- I2C2 cmd and monitor Interface using secondary I2C Bus.
- All features interrupt driven for efficiency.
- Robust 5V power supply, with external On/Off switch, provides ample power for BEC.
- RTC for timing and scheduling.
- Both primary and secondary I2C buses routed to connectors for potential off board use.
- Second RS232 interface (Channel 2; currently unused) routed to header.
- Firmware is updatable via a boot block loader using a serial cable to a PC.
- Support is provided for customization or use as a development platform:
 - 1) Development connector.
 - 2) Provided files (main.c, dalf.lib).
 - 3) Detailed documentation.
 - 4) Lots of unused memory both in FLASH and RAM.
 - 5) Lots of unused I/O.

(*) - A Windows Graphical User Interface (GUI) is under development. Check the website for status. One nice feature of the GUI is that it provides an alternative to the use of a terminal emulator for the board setup task.

3 What Are Your Needs?

Visit the sections of this document that you need to get started with your application, but ...

Everybody should read the “Initial Power Up” and the “Serial Port” sections. The serial port connection to a PC is required for initial board setup and configuration.

Motor Control Interfaces

Operating Mode	Type of Control	Description
RCNRM (R/C normal)	Open Loop	Velocity control; One or both motors
RCMIX (R/C mixed)	Open Loop	Velocity control; Dual motors
POTC (Pot, Center Off)	Open Loop	Velocity control; One or both motors
POTF (Pot, Full range)	Open Loop	Velocity control; One or both motors
POTMIX (Pot mixed)	Open Loop	Velocity control; Dual motors
POTSV0 (Pot Servo)	Closed Loop	Position control; One or both motors
RCSV0 (R/C Servo)	Closed Loop	Position control; One or both motors
TE Cmd Interface	Both	RS232; Terminal Emulator Application
API Cmd Interface	Both	RS232; Applications Programming Interface
I2C2 Cmd Interface	Both	I2C; External MASTER, Dalf as SLAVE

Closed loop control methods generally require use of standard optical incremental encoders. Alternatively, some applications may successfully use analog position encoders.

There are three command interfaces (TE, API, I2C2) that provide essentially the same functionality:

The TE Command Interface is described in the Owner’s Manual and the API and I2C2 Interfaces are described in separate documents. The section “Terminal Emulator Cmd Interface” in the Owner’s Manual provides a complete description of all commands. With the exception of the motor movement commands (“X”, “Y”, and “S”), all commands may be issued while controlling the motors in any of the R/C or POT modes.

The API is suitable for a Windows GUI application. A Windows GUI using the API for board communication will soon be available. The GUI will replace the need for a terminal emulator application for setup operations and will be useful in testing motor commands in your application.

The I2C2 Interface uses the secondary I2C Bus, to communicate with an external control board configured as a MASTER host. The host issues commands and receives data and status from the Dalf Board configured as an I2C SLAVE.

Initial Power Up

It is a good idea to have an off board switch or connector that allows you to easily remove VBATT power from the Dalf Board without monkeying with the screw terminals on the board. For power distribution to the Dalf Board, Drivers, and Motors, I recommend a common off board GND connection (perhaps a screw terminal) and heavy short leads to various boards and devices. If your motor power and VBATT are the same, these comments also apply to your VBATT connection. Take care of these items first.

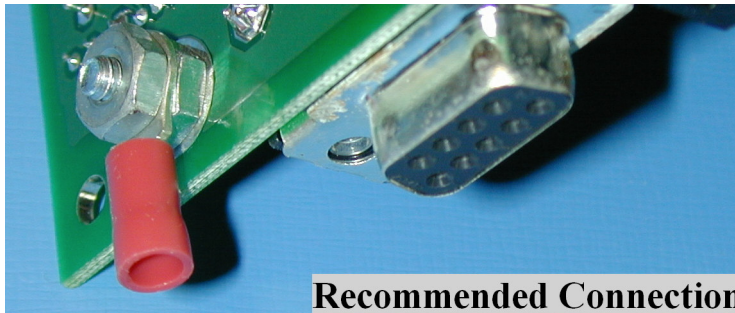
WARNING: The board does not protect against power supply {VBATT, GND} lead reversal, so be careful when attaching the power leads. Don't make this mistake!

Hook up VBATT and GND

The board has a local switching power supply that delivers a fixed VDD=+5V output for the onboard logic. The power supply delivers up to 1A current output, well more than the board needs. The input to this supply is thru the large screw terminals labeled GND and VBATT. The VBATT positive voltage can be anything in the range: [+8V...+40V] DC.

Check Slide Switches

Confirm that the two slide switches are in the "normal" state. The boards are shipped with the switches in the proper position, but it is a good idea to confirm this. The switch labeled ICD should be moved away from the ICD position (toward the power supply caps). The switch labeled PGM should be moved away from the PGM position (toward the green PWR LED).



With power disabled, connect VBATT and GND wires to the board terminals using the supplied hardware (screws, double nuts, terminal lugs).

The picture to the left shows a suitable GND connection.

Now apply VBATT power to the board.

Check Board LED's

The three programmable LED's (LED1, LED2, LED3) will briefly blink when power is first supplied. The green PWR LED should remain on. Depending on your nominal VBATT voltage, the red LED3 may turn on after a brief delay. All other LED's should be off. If either of the yellow LED's are on, it indicates that the slide switches are in the wrong position.

Turn off VBATT power to the board and proceed to the "Serial Port" section of this document.

4 Serial Port

There are no switches to select the board operating mode. Even if your application only requires R/C or POT controls for your motors, you will still need the serial port to specify, and then store, the power up operating mode in non volatile memory. You will discover that the serial port is a valuable informational tool even after setup is complete, regardless of the operating mode.

There are two different serial RS232 communication interfaces supported by the board firmware, but only the **Terminal Emulator Interface (TE)** will be used here. The second interface is the **Applications Programming Interface (API)**. Both interfaces use the same communication protocol: 8bit, no parity, 1 stop bit with the default baud rate of 19,200. They use the same [COMM] connector, use the same cable, and serve similar purposes. So, why are there two interfaces? The API interface is more suitable for a smart PC application like a Windows GUI and is explained in the document, "Dalf API Specification". A Windows GUI using this interface is under development and should soon be available.

The TE Interface is suitable for communication with any terminal emulator application. This is your "out of the box" solution since a Windows based PC will undoubtedly include Hyperterm. Other terminal emulator applications like TeraTerm (better than Hyperterm) can be downloaded free from the web. To completely understand the Dalf TE Interface you will need at least a cursory familiarity with hex number representation. If you have one, a hex calculator can be useful. A Windows OS generally supplies a hex calculator application (look in Accessories).

Recommended Dalf1 Owner's Manual reading now:

Section	Title	Comments
3.1	CMD Interface	General comments about capabilities.
4.5	RS232	RS232 Hardware and the communication protocol.
5.1 - 5.3	Parameter Block, ERAM	Board Configuration
10	Terminal Emulator CMD Interface	Syntax; Lists all commands with inputs and outputs.
App D	Parameter Block Detail	Configuration and setup: Storage in EEPROM.


The Dalf Board contains a 64K byte serial I2C EEPROM device. A small portion of this EEPROM device is used for non-volatile storage of motor parameters, operating modes, etc. This portion of the EEPROM is referred to as the Parameter Block. Section 5 explains the relationship between the Parameter Block in EEPROM and an important corresponding block of RAM that is referred to as **ERAM**. You don't need to sweat the details at this point, but it would be good to get a grasp of the general usage of these two memory blocks. Chapter 10 has a lot of content and you needn't digest it all now. Pay particular attention to the discussion and examples illustrating command syntax. You don't have to become an expert on all of the commands at this point, but it would be good to give the memory access commands ("R", "W", "L") a good scrutiny as you will shortly use them. The **Parameter Block** discussion in the appendix shows you the location in memory where changes are made to customize the board for your application.

Lets try some things!

You've done at least a cursory reading of the above material. You have previously applied power to the board and at least the LED's seem happy. You are ready for the next step where we will execute some commands from your terminal emulator application. Connect the modem cable between your PC serial port and the [COMM] connector on the board. Start your terminal emulator application. As a minimum you will probably have to change the terminal emulator default settings to use a baud rate of 19.2K, but you may need to change the communication protocol to 8N1 and identify the comm port as well (generally com1).

The only connections to the Dalf Board required at this point are VBATT, GND, and COMM.

Greeting

 <pre> Dalf-1F 18F6722 Rev:00 Software Ver: 1.70 User: 00237 </pre>	<p>Now apply power {VBATT, GND} to the board and you should see the usual “Dalf Greeting” message which should look similar to the Tera Term screen captured to the left.</p> <p>18F6722 Rev is the microcontroller chip revision. Software Ver identifies the built-in firmware. User is a board identifier.</p>
--	--

This greeting will appear every time the board powers up from a reset condition. If you are connected to a terminal emulator application and don’t see the greeting, or the message is garbled, something is wrong and you should sort this out before proceeding further.

Some possible problems:

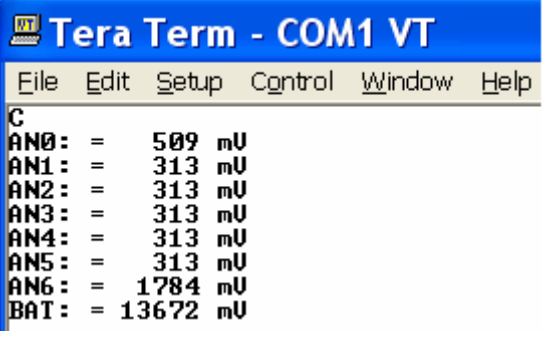
1. Slide switches in the proper positions?
2. All LED’s in proper state?
3. Serial port configured properly in the terminal emulator application?
4. Bad RS232 cable or connection?

Assuming that you got the greeting message, there are several commands that you can try right now. This will serve to illustrate some board features and at the same time enable you to become accustomed to the command syntax. Type the commands exactly as shown, including spaces, but not including the surrounding quotes. Not shown is the Carriage Return character which must also be sent by pressing the [ENTER] key at the end of each command. For example, the very first command example listed below requires two key presses; the [C] key followed by the [ENTER] key. If you make a keystroke error, before you press [ENTER] you can use the [BACKSPACE] key. Mistakes made after pressing [ENTER] will give you a curt error message. In all cases the first character of the command string will be the “CMD” character “A” - “Z”. Lower case for the CMD character is ok.

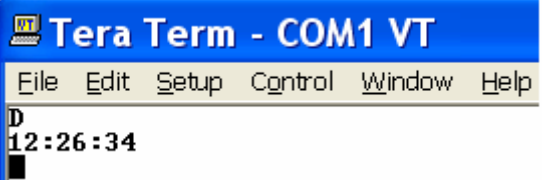
Example#	Command String	Comments
1	“C”	View all A/D Readings in mV.
2	“D”	View RTC setting (since last power up).
3	“E2”	View Mtr2 encoder position.
4	“V2”	View Mtr2 velocity.
5	“T”	Reset the board! You should see a new Dalf Greeting Msg.
6	“L1 01 00”	View RAM Block [0x0100...0x017F].
7	“L2 00 00”	View Ext EEPROM Block [0x0000...0x007F].
8	“N”	View pulse widths in microseconds for R/C channels 1,2,3.
9	“X2 00 32”	Move Mtr2 Forward, 50% power (0x32=50)
10	“X2 01 43”	Move Mtr2 Reverse, 67% power (0x43=67)
11	“U”	View Status of both motors
12	“O2”	Stop Mtr2 (that first character is “Oh”, not zero).
13	“P1”	View PID parameters for Mtr1.
14	“R1 01 7F”	Read RAM Byte at address 0x017F.
15	“R2 00 7F”	Read Ext EEPROM byte at address 0x007F.
16	“W1 01 7F 34”	Write Byte 0x34 to RAM at address 0x017F.
17	“W2 00 7F 77”	Write Byte 0x77 to Ext EEPROM at address 0x007F.
18	“Z”	Upload ERAM into the Parameter Block in Ext EEPROM.
19	“T”	This time with the BTN push-button depressed! (factory restore).

Things to observe while you are trying some of the above commands:

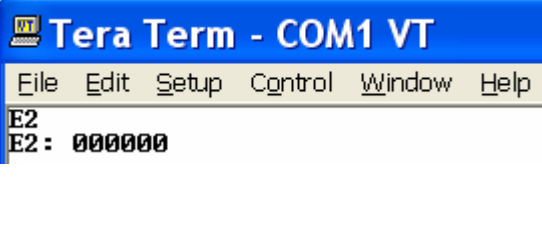
Example 1

 <pre>C AN0: = 509 mU AN1: = 313 mU AN2: = 313 mU AN3: = 313 mU AN4: = 313 mU AN5: = 313 mU AN6: = 1784 mU BAT: = 13672 mU</pre>	<p>Here you see the measured voltages on the analog inputs (screw terminals labeled AN0..AN5) and the voltage on the voltage divider (AN6) that monitors the VBATT voltage. The AN6 reading is the actual measured voltage at the divider. The VBATT voltage is a constant times AN6 and this is shown as the reading labeled "BAT". All values are shown as decimal integers with millivolt units.</p>
---	---

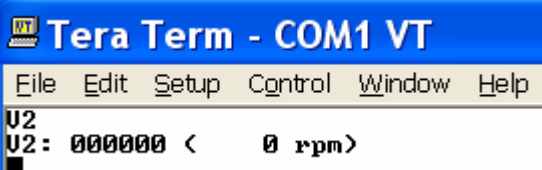
Example 2

 <pre>D 12:26:34</pre>	<p>This example illustrates that the board maintains a RTC. The display shows that 26 minutes and 34 seconds have elapsed since the unit was reset. Values are decimal integers in the usual HH:MM:SS format.</p>
---	---

Example 3

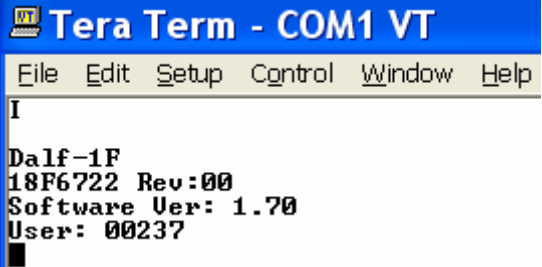
 <pre>E2 E2: 000000</pre>	<p>The "E2" (Encoder 2) Command shows the position of motor 2. Here this is the "home" (000000) position of zero. This will always be the case after the board has been reset and before any encoder movement (unless you are using an absolute analog encoder). The value is a 24 bit, 2's complement variable and is shown here as 3 hex bytes with the most significant byte leftmost.</p>
--	---

Example 4

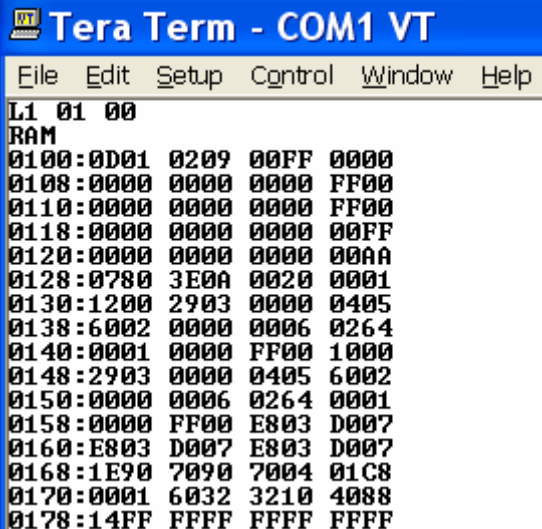
 <pre>U2 U2: 000000 < 0 rpm></pre>	<p>The "V2" command shows the velocity of motor 2. What is shown is actual velocity, not motor power. The value is a 24 bit, 2's complement variable and is shown here as 3 hex bytes (MSB leftmost). The velocity is also shown in decimal RPM (requires correct TPR in Parameter Block).</p>
---	--

Both of the preceding two examples require encoder feedback signals to be useful. Assuming that you had motor 2, its' driver, and its' encoder connected and had just executed a command to drive the motor (eg; Example 9), then the "E" and "V" commands would be showing something interesting.

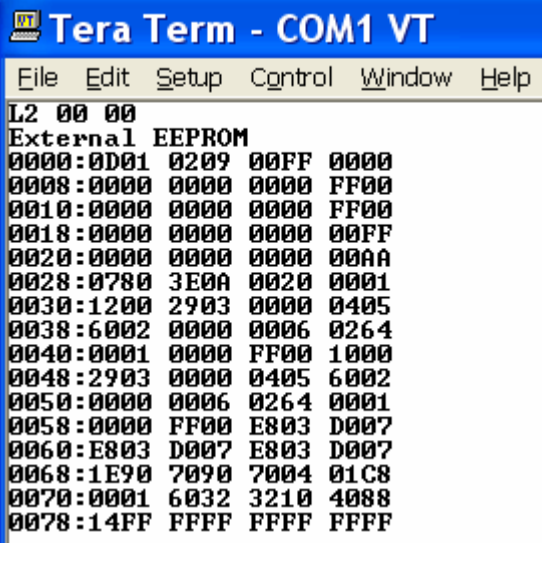
Example 5

 <pre> Tera Term - COM1 VT File Edit Setup Control Window Help I Da1f-1F 18F6722 Rev:00 Software Ver: 1.70 User: 00237 </pre>	<p>Command “I” resets the board and gives you a new greeting. You may accomplish the same thing with the RESET push button. Try both methods now.</p>
--	---

Examples 6

 <pre> Tera Term - COM1 VT File Edit Setup Control Window Help L1 01 00 RAM 0100:0D01 0209 00FF 0000 0108:0000 0000 0000 FF00 0110:0000 0000 0000 FF00 0118:0000 0000 0000 00FF 0120:0000 0000 0000 00AA 0128:0780 3E0A 0020 0001 0130:1200 2903 0000 0405 0138:6002 0000 0006 0264 0140:0001 0000 FF00 1000 0148:2903 0000 0405 6002 0150:0000 0006 0264 0001 0158:0000 FF00 E803 D007 0160:E803 D007 E803 D007 0168:1E90 7090 7004 01C8 0170:0001 6032 3210 4088 0178:14FF FFFF FFFF FFFF </pre>	<p>This example illustrates Cmd “L” which is used to show a 128 byte block of memory. The “1” that immediately follows the “L” indicates that the memory to be viewed will be RAM. The next two hex bytes are the beginning address. In this case, the example specifies a starting address of 0x0100, so 128 bytes [0x0100...0x017F] is shown.</p> <p>The hex word shown to the left of the “:” in each row is the address for the first byte in that row. The byte at address 0x128 is 0x07, the byte at 0x129 is 0x80, etc. Each row shows 8 bytes which are grouped as 4 words for readability.</p> <p>This particular block of RAM is important. It is identified in the documentation as “ERAM” and sometimes referred to as the “run-time-environment”.</p>
--	--

Examples 7

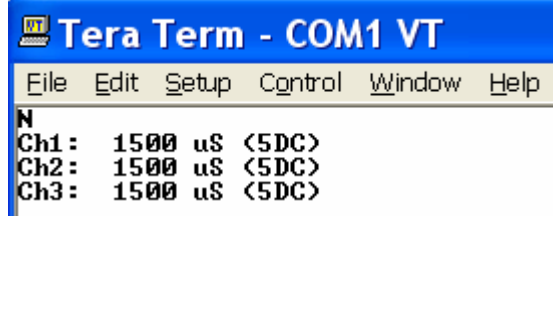
 <pre> Tera Term - COM1 VT File Edit Setup Control Window Help L2 00 00 External EEPROM 0000:0D01 0209 00FF 0000 0008:0000 0000 0000 FF00 0010:0000 0000 0000 FF00 0018:0000 0000 0000 00FF 0020:0000 0000 0000 00AA 0028:0780 3E0A 0020 0001 0030:1200 2903 0000 0405 0038:6002 0000 0006 0264 0040:0001 0000 FF00 1000 0048:2903 0000 0405 6002 0050:0000 0006 0264 0001 0058:0000 FF00 E803 D007 0060:E803 D007 E803 D007 0068:1E90 7090 7004 01C8 0070:0001 6032 3210 4088 0078:14FF FFFF FFFF FFFF </pre>	<p>Another illustration of Cmd “L”, but this time we are looking at a 128 byte block of the external EEPROM starting at address 0x0000.</p> <p>This block of the Ext EEPROM, identified in the documentation as the Parameter Block is important. This is the non-volatile storage for the power up values for operating modes, motor and encoder parameters, R/C constants, ADC timing parameters, default serial baud rate, PID constants, default PWM frequency, ... , and I could go on, but I’m sure you get the picture.</p> <p>It is no coincidence that the ERAM data matches that of the Parameter Block. One of the first tasks of the System Initialization Code on power up is to copy the Parameter Block data into ERAM. The ERAM data is then used to set operating modes and initialize the hardware.</p>
---	--

Unless you make changes to one of these areas after power up, the content of ERAM and that of the Parameter Block will always be identical. A reasonable way to think of these memory areas is that ERAM is the current, working copy, of the Parameter Block.

Your board arrives to you with “factory defaults” in the Parameter Block. Depending upon which version of firmware is installed, the values could differ from what is shown in the above screens. You will almost certainly want to make changes to some of the data in the Parameter Block to customize the board for your application. The actual “factory defaults” are stored permanently in code (FLASH) memory so if you make a mistake in changing the Parameter Block data, and it becomes necessary, you will always be able to restore the original defaults.

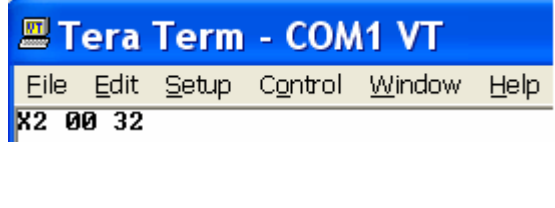
If you ever upgrade the firmware on your board to the latest version, which will likely contain new defaults, it will be necessary to initially store the new factory defaults into the Parameter Block as part of the upgrade process. The procedure is the same that you would use if you wanted to restore the factory defaults because of a customization mistake. See the Owner’s Manual for details.

Example 8

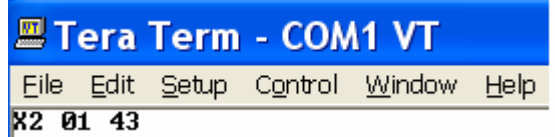
	<p>Cmd “N” is useful for verifying communication when using R/C to control the motor(s). It is also useful for transmitter switch tuning as explained in the Dalf Owner’s Manual.</p> <p>For now it is enough to observe that its function is to show the measured pulse widths from 3 channels of an R/C receiver. The values are decimal integers and the units are microseconds. Also shown is the value in hex (0x5DC = 1500).</p>
--	--

(1500 uS corresponds to servo off and is the default shown in the absence of real signals from the receiver).

Example 9

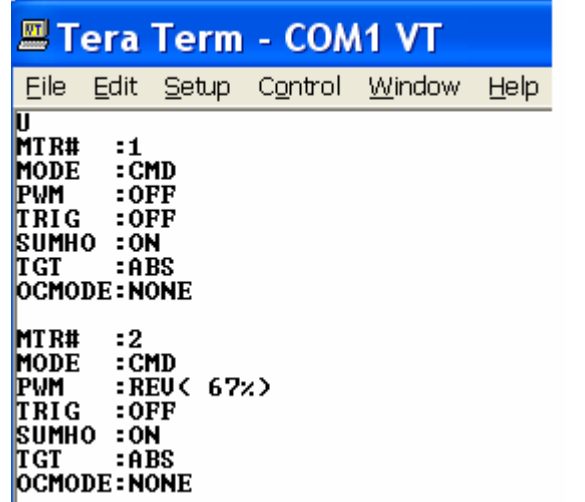
	<p>Cmd “X” is the open loop, motor move command. The “2” immediately following the “X” indicates the command applies to Mtr2. The “00” indicates forward direction, and the 32 is a speed parameter requesting application of 50% duty cycle (0x32 = 50) on the PWM signal. Arguments are in hex.</p>
---	---

Example 10

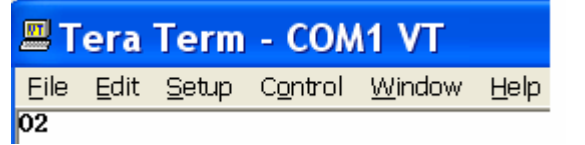
	<p>Another application of Cmd “X” to Mtr2, but this time in the reverse direction with a duty cycle of 67% (0x43 = 67).</p>
---	---

Examples 9 and 10 result in changes to the motor control signals which are routed to the connector for driver 2. If motor 2 and its’ driver were connected and powered you would be seeing movement of motor 2 as a result of these commands. Did you happen to notice the small Mtr2 LED? The color indicates motor direction and the brightness indicates the motor power level. If you watch carefully, you should be able to detect the gradual increase in brightness as the power is ramped to the requested target power. The ramp, or **Slew** rate, is the fourth parameter to the “X” command and can be specified in the command, or as in these examples, be defaulted to a given value. The default Slew value (AMINP) is one of many things that you can customize with changes to values in the Parameter Block.

Example 11

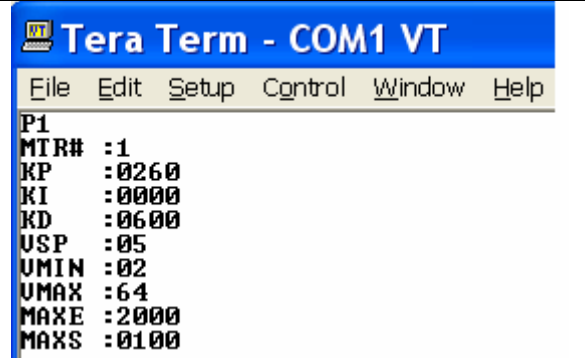
 <pre>Tera Term - COM1 VT File Edit Setup Control Window Help U MTR# :1 MODE :CMD PWM :OFF TRIG :OFF SUMHO :ON TGT :ABS OCMODE:NONE MTR# :2 MODE :CMD PWM :REU< 67%> TRIG :OFF SUMHO :ON TGT :ABS OCMODE:NONE</pre>	<p>Cmd “U” is used here to show the status of both motors. Notice that the line labeled “PWM” for Mtr2 shows that the motor is now being powered in the reverse direction with a duty cycle of 67% as directed by the command of example 10.</p>
---	--

Example 12

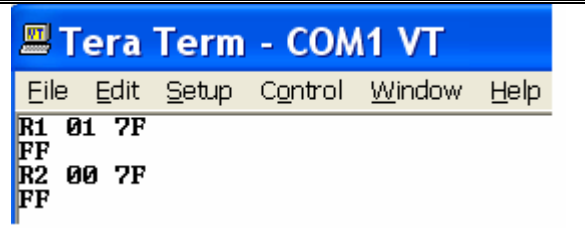
 <pre>Tera Term - COM1 VT File Edit Setup Control Window Help O2</pre>	<p>Cmd “O” is used to stop the motor(s). In this case, the “2” immediately following the “O” directs the board to stop Mtr2.</p>
---	--

Observe the effect on Mtr2 LED. (Retry example 11 after stopping the motor to see the new status).

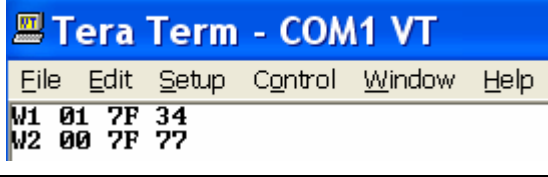
Example 13

 <pre>Tera Term - COM1 VT File Edit Setup Control Window Help P1 MTR# :1 KP :0260 KI :0000 KD :0600 USP :05 UMIN :02 UMAX :64 MAXE :2000 MAXS :0100</pre>	<p>Here the “P” command is used to show you the parameters that affect PID operation of Mtr1. This command is also used to set the KP, KI, and KD parameters during “PID Tuning” of your motors for closed loop operation.</p> <p>All of the PID stuff is discussed in “exquisite detail” in the Dalf Owners Manual. All values are shown in hex format (MSB leftmost).</p>
--	---

Examples 14 and 15


 <pre>Tera Term - COM1 VT File Edit Setup Control Window Help R1 01 7F FF R2 00 7F FF</pre>	<p>Here the “R” (Read Memory Byte) command is used to show you the last byte in the ERAM block (RAM) and the last byte in the Parameter Block (Ext EEPROM) respectively. As you saw earlier when using the “L” Command, these bytes are both equal to 0xFF.</p>
--	---

Examples 16 and 17

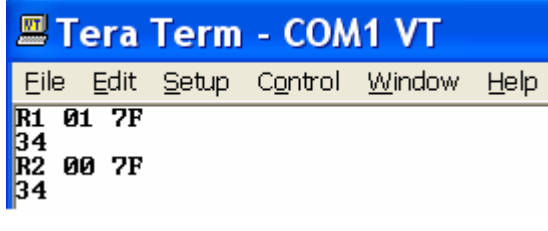
	<p>Here the “W” (Write Memory Byte) command is used to change the bytes shown previously to</p> <p>0x017F: 0x34 (RAM) 0x007F: 0x77 (EEPROM)</p>
---	---

(You might want to retry examples 14, 15, 6, and 7 at this point to confirm the change made above.)

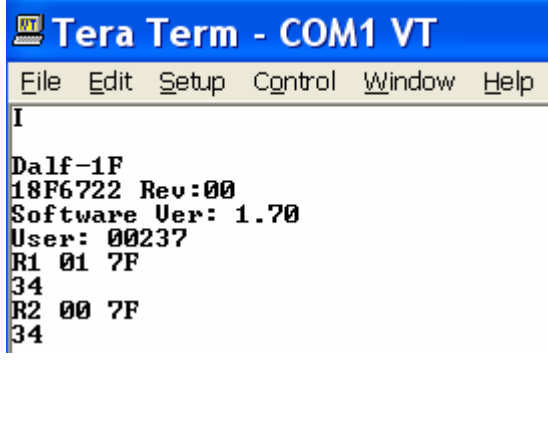
Example 18

	<p>Here the “Z” command is used to upload (copy) the current contents of ERAM (including the value 0x34 in the last byte) into the Parameter Block</p>
---	--

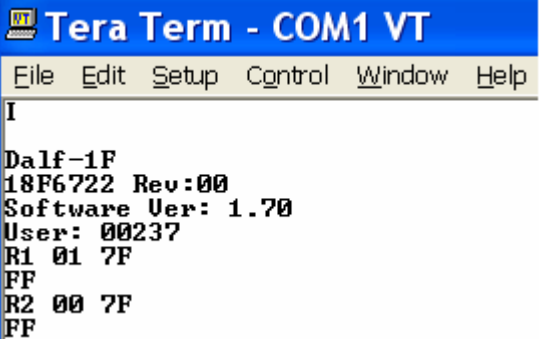
Checking out the upload

	<p>Here we again look at these last bytes after the upload has completed. Notice that the last byte of each block is now 0x34.</p> <p>If we had made other changes in the ERAM area of memory, they would also now be recorded in the Parameter Block.</p>
--	--

Reset the board and check the bytes again

	<p>Reset the board (example 4 or the RESET button) and look at those last bytes again.</p> <p>Notice that even after the board reset, the previous change made to the Parameter Block in Example 18 has been preserved (and now copied into ERAM).</p> <p>Because the Parameter Block is in EEPROM, the change is “permanent” (until you change it again).</p> <p>If this particular byte in ERAM controlled some board feature (it doesn't), you would now be using the feature.</p>
---	---

Example 19 - Restoring factory defaults

 <pre>File Edit Setup Control Window Help I Dalf-1F 18F6722 Rev:00 Software Ver: 1.70 User: 00237 R1 01 7F FF R2 00 7F FF</pre>	<p>Here we again reset the board using the “I” command, but this time first press and hold down on the push button labeled “BTN” until after you see the greeting.</p> <p>Observe that the last byte in the Parameter Block has been restored to the factory default value of 0xFF. This is an important function of the BTN switch! If pressed and held down during power up it will cause all of the factory default settings to be restored to the Parameter Block.</p>
--	---

Summary:

At this point, you have some experience with the tools that are needed to make changes to the values in memory, including the operating mode, either temporarily in ERAM or more permanently in the Parameter Block. Appendix D in the Dalf Owner’s Manual tells you where to find the MODE bytes and what bits need to be set or cleared. You don’t have to make all of your changes at once - remember, the Parameter Block is located in non-volatile storage and will be retained between sessions.

When you are ready to make changes, just follow this simple procedure:

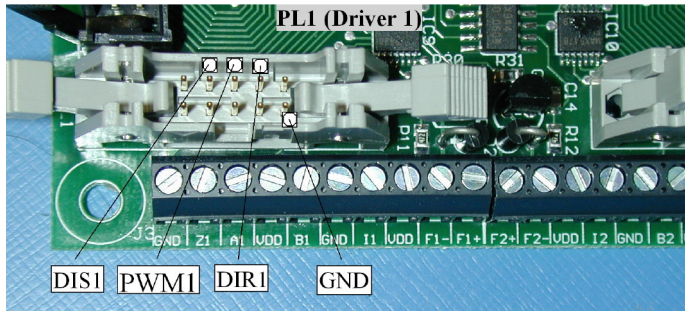
- Use the “R” command to observe the current value of the byte to be changed.
- Use the “W” command to change the value (ERAM or Parameter Block) as desired.
- Use the “R” command again to confirm that the change has been made properly.

One more thing: While the setup procedures described here are not particularly difficult, the soon to be released GUI will make all of this simpler.

Remove power. If you have motors and drivers to hook up, proceed to the next sections.

5 Motor Power Driver Setup

In addition to GND, your driver(s) should expect 3 signals: [PWM, DIR, DIS] for hook up from the Dalf Board. PWM (“Pulse Width Modulation”) provides power control using a variable duty cycle square wave at a fixed frequency. The default PWM frequency is 20 kHz, which should work fine for you, but this is easily changed (See the section “PWM Freq Control” in the owner’s manual). DIR is the motor direction control signal. DIS is the disable line. Shown below is the PL1 connector used to connect Driver1 to the Dalf Board. The PL2 connector is identical with the signals for Driver2 located in the same pin positions. You are probably better off supplying GND to your driver(s) from a common off board distribution point as suggested earlier (rather than a daisy chain approach thru the driver connectors).



If you are using an OSMC Motor Driver

Hookup will be simple. Just connect the 10 pin connector on the OSMC Board to the 10 pin connector on the Dalf Board using a standard width 10 pin ribbon cable. You are done! The high side signals of the OSMC H Bridge are pulled hi at the PL1 Connector using the +12V input from the regulated supply on the OSMC board. The motors are controlled strictly with low side H Bridge signals PWM and DIR. Motor reversal is accomplished by toggling the DIR line and complementing the PWM signal.

If you are using a different motor driver

You can still use a standard 10 pin ribbon cable and connector for the Dalf Board end of your cable. On the other end of the cable, connect the 3 control signals to the required input locations on the connector for your driver. Before connecting any of the other signals on this cable, review the connector pin out shown in the “Motor Driver Connection” section of the Dalf Owner’s Manual and the board schematic. For example, if you wish to provide a “fan” voltage for the fan drivers on the Dalf Board, you can connect a +12V supply on your driver to one of several pins on the Dalf connector (Pin1=Pin2=Pin5=Pin7=+12V). Other +DC voltages, besides +12V, are suitable for connection to the +12V pin (to power other, non-fan devices), but first review the transistor device specifications and the Dalf Schematic.

Okay, you are now connected to your non-OSMC driver. There is one, and possibly two additional complications that may require you to make changes in the Parameter Block. The OSMC Drivers use the HIP4081A chip and require a “PWM complement trick” for motor reversal. The Dalf Board employs this trick by default, so if your drivers don’t require the trick (and they likely won’t), you will need to disable it by clearing the “osmc” bit in the appropriate MODE byte in the Parameter Block. If the setting is not changed, commands to run the motor in reverse at fast speed settings will instead run slow and vice-versa. Another possible change has to do with the polarity of the DIS signal. If your driver expects this signal to be active HI (that is; motor disabled when DIS =+5V), you will be fine. However, if your driver expects the signal to be active LO (motor disabled when DIS = 0V), you will have to set the “dis_activelo” bit in the appropriate MODE parameter. See the section “Motor Driver Connection” in the owner’s manual for additional details.

6 Motor Setup

Your PM DC Motors should have just two leads which are connected to the driver outputs. Pretty simple! There is one polarity gotcha that is easily corrected. When you get around to testing the motors, issue a command over the terminal interface (See the examples in the “Serial Port” section of this document) to drive the motor in the forward direction. If your notion of direction indicates that the motor is going in reverse instead, remove power, switch the motor connections to the driver, and retest.

7 Encoder Setup

Dalf closed loop motor control operations require motor encoder feedback to determine position. The recommended position sensor is a standard optical, incremental encoder that supplies quadrature feedback signals. Alternatively, beginning with firmware version 1.50, there is built-in support for an analog motor position encoder. If you are planning on using an analog sensor you should read the “Analog Feedback” section of the Owner’s Manual to ensure that you understand the limitations.

YOU DON’T HAVE ENCODERS

If you don’t have motor encoders, the PID, Trajectory Generator, and closed loop operations will not be available to you, and there is nothing for you to do in this section of the document. However, lack of encoder(s) will not keep you from controlling the motors in open loop modes with R/C, pots, or the serial command interface(s).

YOU HAVE OPTICAL ENCODERS

Optical Encoder Tip: CPR = “Cycles-Per-Revolution” is an encoder characteristic which should be considered carefully in conjunction with your maximum motor velocity and desired position resolution. Generally CPR corresponds to the number of “slits” in an encoder wheel. The actual resolution will be TPR = “Ticks-Per-Revolution” = 4*CPR. With large values of CPR, you will gain finer position and velocity resolution, but this advantage has a cost. The larger the value of CPR that you use, the more time will be spent handling the interrupts to maintain position and velocity. In extreme cases, you may be forced to limit motor operation to some fraction of full power (see VMAX), or mount the encoder “downstream” after a gear reduction.

Important Note: See the Owners’ Manual regarding the fINT Specification to ensure that the Dalf Board can handle your particular motor/encoder setup.

The encoders will generally have 5 wires to hook up (4 if missing the index wire). The wires are frequently labeled as shown below. The “Type” field refers to things from the point of view of the encoder (ie; Signal “A” is an output from the encoder to the Dalf Board). The “Z” signal is routed to an input pin on the microcontroller, but current Dalf firmware does not monitor it.

Signal	Type	Comments
+5V	Power	Encoder power.
A	Output	With motor movement a square wave.
B	Output	With motor movement a square wave, 90 deg out of phase with Signal A.
Z (or I)	Output	Index or Home. With motor movement, a single pulse every revolution.
GND	Power	Encoder power.

An important encoder spec is **CPR**. Because the A and B signals are out of phase by 90 degrees, Dalf firmware is able to maintain a positional counter in units of “**ticks**” with 4 times the resolution of the encoder. For example, if your encoder has CPR=100, then the **TPR** (Ticks Per Revolution) is 4*100=400. This means that every revolution of the encoder will result in an increase or decrease in the encoder position counter by 400. The value of TPR is a parameter in the non-volatile memory of the Parameter Block. The value should be changed to match your particular encoder.

Encoder Hook up

This is really simple using the 5 screw terminals devoted to each encoder. For example, the set of screw terminal connectors on the Dalf Board for the Motor1 Encoder are labeled {GND, Z1, A1, VDD, B1}. Simply connect the corresponding signals and you are done. Well ... almost.

“aleadsb”

The Dalf firmware is able to distinguish forward rotation from reverse rotation, but in order to know which transition actually represents forward, it is necessary to know whether Signal A leads or lags Signal B for forward rotation. Your encoder spec will provide information, but it can be confusing because it is usually worded in terms like “for clockwise rotation, ..”. The correct setting will also depend on encoder orientation (left or right shaft mounting). For a differential drive employing two motors and encoders it would not be unusual for example to require a different setting even though the encoders are identical. Here is a simple procedure to make the correct setting: First be sure that commands to drive the motor forward actually do that. If not, reverse the motor leads first. After power up, rotate the motor manually in the forward direction while monitoring motor position using the serial port (Cmd E). If the count advances, you have it correct. Otherwise, change the “aleadsb” setting and retest. **Repeat for both motors!** For additional detail see the owners’ manual (search for “aleadsb”).

YOU HAVE ANALOG ENCODERS (Assumes the MA2 Encoder (*))

Encoder Hook up

The encoder has 3 signals to hook up: {+5V, GND, Vout} so this is pretty simple. Connect +5V and GND to any available Dalf screw terminals for VDD and GND respectively. Connect the signal output (Vout) to the AN2 screw terminal for usage on Motor 1 or to the AN3 screw terminal if the sensor is for Motor 2.

DMAX, AD_GAP

The DMAX and AD_GAP parameters are used by the algorithm employed to maintain position using an analog encoder (see the “Analog Feedback” section of the Owner’s Manual). If you use the recommended sensor and your application meets the maximum RPM guideline, the default values for these parameters will likely be fine for you.

analogfb, analogdir bits

You will need to set the analogfb (analog feedback) bit in the MODE1 parameter in the Parameter Block to inform the board that you will be using an analog sensor for position feedback. You may also need to set the analogdir (analog direction) bit in the MODE3 parameter in the Parameter Block. The usage for this bit is analogous to that for the “aleadsb” bit for optical encoders. The value of the bit will depend on the mounting orientation of the sensor. See the Owner’s Manual for additional detail.

Disclaimer

If you use a different sensor, or your application has a higher maximum shaft RPM than recommended in the Owner’s Manual, you may have difficulty obtaining reliable encoder operation. In this case, a complete understanding of the algorithm (see Owner’s Manual), and experimentation with changes to DMAX and AD_GAP control parameters may help you out.

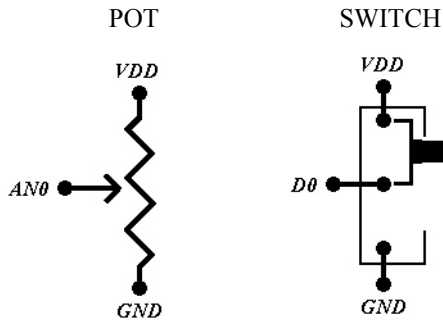
(*) The MA2 sensor is a Hall Effect based sensor available from US Digital in several configurations (I recommend the ball bearing shaft) with suitable cabling. The US Digital site <http://www.usdigital.com/> also has optical encoders.

8 Pot Setup

The setup for using pots as motor control inputs is the same for POTF, POTC, POTMIX, or POT Servo modes except that the POTF will require an extra external switch. If you will be using the POT Servo mode you will also require an optical or analog position sensor (see prior section “Encoder Setup”).

For dual motor controls using potentiometers you need two 3-terminal pots (one for each motor) and at least one switch (D0). The two pots may be mounted in a single, 2-axis, control device known as a joystick. The potentiometer range is not too important, but **I recommend that you use pots in the 2K Ohm range** to support fast ADC sampling. The D0 switch serves as an On/Off switch for purposes of safety. Without it, your motor(s) could simply start running after power up according to your pot position(s). In POTF Mode an additional switch (D1) is used to provide the forward/reverse signal for both motors. The other pot modes do not require D1. You can use SPST switches with current limiting resistors, but the simplest arrangement is to use SPDT switches as shown in the diagram below.

The signals on the Dalf Board that you are interested in are: [AN0, AN1, GND, VDD, D0] and possibly D1. These signals are all accessible as labeled screw terminal connections. For GND and VDD you may use any of the available screw terminals marked as GND and VDD. The AN0 and AN1 are the analog input signals used for Motor1 and Motor2 pot control respectively and should be connected to your POT wipers as shown. GND and VDD supply the power signals for the pots and switches. D0 is the input for the On/Off switch. Shown below are the connections for Motor1 control and the On/Off switch.



The control for the pot hookup for Motor2 is identical to that of Motor1 except it uses AN1. D1 is the motor direction input (POTF operating mode only) and has similar connections to that of the On/Off switch.

Parameter Block Changes

Before you will be able to use the potentiometers with the Dalf Board for motor control you will have to change the default operating mode to one of the four supported Pot Control Modes. This will require a change to the **MODE2** byte(s). Runtime changes are possible by changing the value in ERAM, but if you want to change the default operating mode, you will need to make the change in the Parameter Block. Other parameters affect the performance of pot mode controls, but for now you should be ok with the defaults.

See the “POT Mode Interface”, “Servo Modes”, and “Appendix D” sections of the Dalf Owner’s Manual for additional details.

9 R/C Setup

For dual motor control using a radio control system you will need an R/C Transmitter and Receiver with at least two channels. Three modes of operation are supported: RCNRM, RCMIX, and RCSVO. The first two of these are open loop methods in which the **motor speed** is controlled by the switch position. The RCSVO (R/C Servo) mode is a closed loop method to control the **motor position** as a function of the switch. RCSVO mode requires an optical or analog position sensor (see section “Encoder Setup”).

How it works

With R/C controls the transmitter switch position is converted by the receiver to periodic output pulses whose width encodes the transmitter switch position. The receiver outputs (typically routed to servos) are instead connected to screw terminal inputs on the Dalf Board. The Dalf firmware measures and records the pulse widths for use in controlling the motor(s). If the pulses are present on the signal lines the Dalf firmware measures the pulse widths regardless of operating mode. If the operating mode is one of the three R/C Control Modes, the measurements on channels 1 and 2 are translated to produce motor commands. The channel 3 measurement is currently unused.

Hookup

The signals on the Dalf Board that you are interested in are: [EX0, EX1, EX3, GND, VDD]. Connect your receiver to the screw terminals as shown below. You may use any of the screw terminals labeled VDD and GND for the receiver power connections.

Receiver	Dalf	Comments
+5V	VDD	Receiver power.
Ch1	EX0	Motor1 Control.
Ch2	EX1	Motor2 Control.
Ch3	EX3	Optional.
GND	GND	Receiver power.

Pulse Widths (Note: uS = microseconds)

Nominal pulse widths will vary in the range [1,000 uS ... 2,000 uS]. For the open loop control modes, the minimum 1,000 uS is interpreted as full reverse, the middle 1,500 uS as Off, and the maximum 2,000 uS as full forward. For optimum motor control performance, the full travel of your transmitter switches should result in a range close to the nominal values. There are two different methods, depending on where the adjustment is made, to “tune” the performance of your system. See the section titled “R/C Tuning” in the Dalf Owner’s Manual.

Communications Test

OK, you are connected, but you are not yet in one of the R/C Operating Modes. It is a good time to test communications between your transmitter, receiver, and the Dalf Board. Connect to a terminal emulator application using the COMM port. Apply power to the Dalf Board and also to your Transmitter (your receiver will likely be powered from the Dalf connections). Successively test each of the channels 1, 2 and 3 by moving the transmitter switch to the MIN, MID, and MAX positions. In each case, hold the switch in position while using command “N” to measure the pulse width. Record the results. This will have tested communication, and the recorded values will be useful in determining whether or not your system should be “R/C Tuned” as mentioned earlier.

Parameter Block Changes to use R/C in Controlling the motor(s)

Before you will be able to use R/C with the Dalf Board for motor control you will have to change the default operating mode to one of the two supported R/C Control Modes. This will require a change to the **MODE2** byte(s). Runtime (this session only) changes are possible by changing the value in ERAM, but if you want to permanently change the default operating mode, you will need to make the change in the Parameter Block. Other R/C related parameters of interest: **RC1MIN, RC1MAX, RC2MIN, RC2MAX, RC3MIN, RC3MAX, RCD, MIN, MAX, RCSP,** and **AMINP**, but you should be fine initially with the default values for these. See the “R/C Mode Interface” and “Appendix D” sections of the Dalf Owner’s Manual for additional details.

10 I/O Expanders

Each of the two I/O Expander parts provides 16 GPIO's which are currently unused. Each signal can sink or source 25 mA so is capable of directly driving LED's (you do need to observe the device limit of 200 mA). Communication between the microcontroller and the I/O Expander parts occurs over the primary I2C bus at a baud rate of 400 kHz.

Access

The GPIO's may be accessed in several ways, but none require knowledge of the I2C Bus Specification.

- With some code development of your own you can extend the Dalf Functionality by changing the firmware to call supplied library functions that provide complete access to the parts. See the section "Firmware Library Routines" in the Dalf Owner's Manual.
- TE - A terminal emulator application using the RS232 serial link can access the parts. See the commands described in the Dalf Owner's Manual.
- API - A GUI Application using the RS232 serial link can access the parts using API Interface. See the commands described in the Dalf API Specification.
- I2C2 - An application running on an external board can access the parts using the I2C2 Interface. See the commands described in the Dalf I2C2 Specification.

Applications

Having a personal interest in robotics, all of these GPIO's just scream "sensors, sensors, and more sensors". Another use would be an interface to an LCD panel to display status information. Of course there are lots of other applications including LED's, Memory Interfaces, Communication, Device Control, etc.

Speed

The GPIO's are not directly connected to the PIC Microcontroller, but are instead connected to an I/O Expander part which is accessed over the I2C Bus by the PIC. This means that access is somewhat slower than would be the case if the connector pin were connected directly to the micro. This might be a limitation for your planned usage. Refer to the section "Firmware Library Routines" in the Dalf Owner's Manual for timing information.

11 Code Development

So, you want to do code development using the Dalf-1F Board.

What is on the CD?

Included on the CD are the files <main.c>, <dalf.lib>, and data sheets for all of the on board parts. Together with a couple of other files (also included on the CD), and standard software tools described in the Owner's Manual, this provides the capability to reproduce the full functionality of the FLASH image.

The <dalf.lib> library file contains functions to access all of the on board parts as well as perform PID and Trajectory Generator computations. The <main.c> file provides the "backbone" and structure of the application. Using the indicated tools, you can add your own assembly (.asm) or C-Language (.c) files together with any modifications to the main.c file and link the resulting object files together with the dalf.lib file to customize the board for your application.

License

The PC board design and the <main.c> source file is provided open source. The <dalf.lib> library file is provided without source, and with license limitations that exclude off board usage. Your purchase of the Dalf-1F Board grants you the right to extend or modify the firmware and to use the functionality in the <dalf.lib> file in your design as long as it is used exclusively in an application running on this board. It does not allow you to reverse engineer the functionality in the <dalf.lib> file or to use this functionality in applications not running on this board.

Tool Sets

The library file <dalf.lib> provides functions callable from either PIC Assembler or C. Access to library functionality from a C Language application using the Microchip development tools {mcc18.exe, mplab ide, mpasm.exe, mplink.exe, mplib.exe} has been tested. Other tool sets have varying run time environments (eg: calling conventions), and it will likely be more difficult to access the library functionality in these cases. See the "Development Tools" section of the Dalf Owner's Manual for additional details.

Flashing the part:

There are two portals for flashing a new image into the part. With the proper tools, the 6-pin modular connector (ICD) can be used to flash the part as well as provide real time debugging capability. The serial port connector (COMM) provides another option to flash the part using the built-in boot loader feature and the supplied PC Application <p1618qp.exe>. The "Development Tools" section of the Dalf Owner's Manual includes a discussion of the boot loader feature.