Graphical User Interface Manual

Revision 1.0.0
Feb 18, 2008
# Table of Contents

1 GUI APPLICATION OVERVIEW ..................................................................................................................3
   1.1 COMMUNICATION ..........................................................................................................................3
   1.2 FEATURES .......................................................................................................................................3
   1.3 COMPATIBILITY ............................................................................................................................4
   1.4 OTHER DOCUMENTATION ..........................................................................................................4
   1.5 DEVELOPMENT HISTORY ............................................................................................................4

2 GUI INSTALLATION ......................................................................................................................................5

3 OPENING SCREEN .....................................................................................................................................5

4 STANDARD WINDOWS MENUS ..............................................................................................................6
   4.1.1 [File] Menu (Saving/Restoring your Parameters) .......................................................................6
   4.1.2 [Edit] Menu (Unused) ...............................................................................................................6
   4.1.3 [View] Menu (Unused) ..............................................................................................................6

5 [SERIAL PORT] MENU (CONNECT TO THE BOARD) .............................................................................7

6 [PARMS] MENU ......................................................................................................................................9
   6.1 [PARMS/EDIT] MENU ..................................................................................................................10
   6.1.1 [Timing] Dialog .......................................................................................................................11
   6.1.2 [Motor 1] Dialog .....................................................................................................................13
   6.1.3 [Motor 2] Dialog .....................................................................................................................14
   6.1.4 [Calibration] Dialog ...............................................................................................................15
   6.1.5 [Miscellaneous] Dialog .........................................................................................................16
   6.1.6 [I/O Expander 1] Dialog .........................................................................................................17
   6.1.7 [I/O Expander 2] Dialog .........................................................................................................18
   6.1.8 [Unused] Dialog .....................................................................................................................19
   6.2 [PARMS/READ...] MENU (GUI ← BOARD) ..............................................................................20
   6.3 [PARMS/WRITE...] MENU (GUI → BOARD) .............................................................................20

7 [BOARD] MENU (BOARD STATUS) .......................................................................................................20

8 [WINDOW] MENU (MAIN ATTRACTION!) .............................................................................................22
   8.1 [HOME] .....................................................................................................................................22
   8.2 [OPENLOOP] ..............................................................................................................................22
   8.3 [CLOSEDLOOP] ..........................................................................................................................24
   8.4 [PID_TUNE] ...............................................................................................................................26
Warranty
The Dalf1 Graphical User Interface Application is provided "as is" without warranty. The entire risk for the results and performance is assumed by the user. 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.

Copyright
Copyright 2007 Embedded Electronics, LLC.

Software License
The Dalf1 Graphical User Interface Application is provided for use with the Dalf1 Motor Control Board. It is provided free of charge but without the Visual C++ source code. Its sole use is for the purpose of communication and control of the Dalf1 Motor Control Board sold by Embedded Electronics http://www.embeddedelectronics.net/. Any other use including reverse engineering to obtain a similar product or its use for an application that does not employ the Dalf1 Motor Control Board is expressly forbidden. Nothing in this license prohibits Embedded Electronics LLC from providing portions of the source code on request. Any such code would be for the exclusive use of the grantee and could not be used in a commercial application without the express written approval of Embedded Electronics, LLC.

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 a component in life support devices of any description.

NOTE: This document corresponds to the first official release (Ver 1.0.0) of the Dalf GUI Application. It is targeted for use with a Dalf Motor Control Board having firmware version >= 1.70. Use with earlier versions of board firmware will likely get you in trouble in possibly subtle ways. You should upgrade your board firmware (if necessary) before use. Periodically it is expected that improved versions of the GUI and the documentation will be made available thru the EE Website. Your feedback regarding issues or suggested improvements to the GUI is welcome. To report an issue or make a suggestion, simply send an email to:

support@embeddedelectronics.net

==========================================================================
1 GUI Application Overview

This document describes the Dalf Graphical User Interface (GUI) Application. The application is used in conjunction with a user’s Windows PC, a standard serial (RS232) cable, and the Dalf Motor Control Board. Demonstration of some of the application features requires a system that also includes one or two motors and associated motor drivers and motor position encoders. The GUI is a Windows Application which is first installed and then runs on the PC. With the PC connected to the Dalf Motor Control Board thru the serial cable, the application provides a convenient way to perform board setup and demonstrate various motor control features from within a standard Windows application environment.

1.1 Communication

The Dalf Owner’s Manual and the Dalf Getting Started Manual describe a serial (RS232), ASCII communication protocol which is referred to as the Terminal Emulator Interface (TE). The TE protocol is used with a standard terminal emulator application (eg; Hyperterm) for board setup and other operations.

The GUI application described in this document provides a convenient, easy to use alternative to the use of a terminal emulator application for many operations. The GUI communicates with the board by using a communication protocol referred to as the Application Programming Interface (API). The API is a serial (RS232), BINARY communication protocol. It is one of several serial command interfaces designed by Embedded Electronics, LLC and supported by the Dalf Board Firmware. The API is described fully in a separate document.

1.2 Features

The GUI is a Windows Split Pane Application. In general, controls appear in the left pane and graphical content appears in the right pane. A separate, tabbed dialogue type interface supports board setup operations. A PID Control Panel supports PID Tuning with the results presented graphically. The Open Loop Control Panel demonstrates open loop (no motor position feedback) control of dual motors with sliders and edit boxes as the controls. The Closed Loop Control Panel demonstrates closed loop (with motor position feedback) control of dual motors using edit and check box controls. A Status panel provides feedback on the current board status.

In general, byte and multi-byte parameters are entered in decimal or selected in scroll boxes while Radio Buttons and Check Boxes are used to deal with bit controls. Standard Windows File Save/Open operations provide the vehicle to save and restore Parameter Settings between the PC Hard Drive and the GUI Application. Other menu selections provide the ability to read or write board parameters between the GUI Application and the Dalf Board Memories.

The graphical output from the PID Tuning feature is a particularly helpful aid in determining appropriate motor control constants for your specific application. The Open Loop and Closed Loop control panels may be used to demonstrate many of the Dalf motor control features. The graphical output from these control panels demonstrates the effect of motor parameter changes.

The GUI application itself was developed using the Professional Edition of Visual Studio 2005 development environment using the Visual C++ Native Language Support.

For those (increasing numbers) of you whose PC does not support a serial port, you can still use this application with a USB to Serial converter cable such as that provided by KEYSPLAN. The GUI performs a “virtual scan” for available COM ports and will find the USB adaptor. The Keyspan 19HS part has been functionally tested with the GUI.
1.3 Compatibility

PC Operating System

The GUI has been tested on Windows XP and 2K platforms. You are welcome to try it with other Operating Systems, but be aware that it will not have been tested in those environments.

Dalf Board Firmware Compatibility

The GUI is only intended for use with Dalf Boards that contain firmware versions >= 1.70. I recommend that you upgrade the firmware in your board to the latest version (currently 1.71) before using this GUI. While the Dalf Board Command Set has remained fairly stable with earlier versions, the location and definition of important Motor Control Parameters has not. Usage of this GUI with Dalf firmware versions earlier than 1.70 will just get you into trouble in sometimes subtle ways.

1.4 Other Documentation

In addition to this GUI Document, other Dalf Motor Control Documents are available for download from the EE Website. These include

Owner's Manual: Describes all Dalf Board features using references to other documents as needed.

Getting Started Manual: Describes hook ups and configuration with a terminal emulator (TE) application.

API Interface: Describes a binary serial (RS232) protocol suitable for communication with the Dalf Board.

I2C2 Interface: Describes a serial (I2C) protocol suitable for communication with an (off board) I2C MASTER.

1.5 Development History

Dalf Motor Control Board and Firmware:

09/2006: Version 1.40 - The first release of the Dalf Motor Control Board. Since then several firmware upgrades have occurred but no hardware changes have been made to the board. The latest firmware version as of this writing is 1.71.

GUI Application:
The GUI Application was begun in January 2007 but most of the work has been done more recently.

02/18/2008: Version 1.0.0 - First official release.

See the EE Website for downloads of all of the latest documents and firmware:

http://www.embeddedelectronics.net/
2 GUI Installation
This discussion assumes that the target PC is a Windows 2K or XP box.

Copy the provided files into a convenient directory on your PC and run the setup program to install the DalF.exe Application on your PC. The installation allows you to place the executable in a folder of your choice and makes the executable available under your [Program Files] directory. You can create a shortcut and move it to your desktop if you find that convenient.

If you later decide to uninstall DalF.exe (and you will need to do so if you later want to install a more current version), just use the \Control Settings\AddRemove dialog to remove it.

3 Opening Screen
Ensure that the DalF1 Board is powered up and connected to your PC thru one of its available COMM (RS232) ports (generally COM1). Execute (Run) the GUI. You will see the initial HOME Screen which contains some information as well as a set of menu options.

Generally one of your first operations will be to connect to the board using the [SerialPort] Menu, but let’s discuss the menus in order as they appear left to right in the menu bar of the Home Screen.
4 Standard Windows Menus

4.1.1 [File] Menu (Saving/Restoring your Parameters)
This menu provides the usual {New, Open…, Save, Save As…} options. In this application you use this menu and the associated Windows Open File Dialog to save and restore Parameter Block Settings to/from your hard drive. Example; Assuming that you have established the desired motor control settings for your application in the GUI, you can use the Save As… menu to save the settings to a file called MyParms.dlf. When you again run the GUI you can use the Open menu to retrieve the MyParms.dlf file data to the GUI.

The files that you save and restore to/from your harddrive will all have the .dlf extension and will all represent the data present in the tabbed dialog boxes that you see invoke the [Parms]/[Edit] menu. This shortened version of “Dalf” should serve as a memory reminder if your “exploring” you see a file with the .dlf extension.

Notice that you can’t use the File Menu to change the actual Dalf Board Settings. You have to use another menu for that - more about that feature later.

4.1.2 [Edit] Menu (Unused)
This is the usual Windows Edit Interface. This menu is not used by the GUI and all options are “grayed out”.

4.1.3 [View] Menu (Unused)
This is the usual Windows View Interface. This menu is currently unused by the GUI application.
5  [Serial Port] Menu (Connect to the Board)

You use this dialog to select the COMM Port and Baud Rate Setting and to open the communication channel between the GUI Application and the Board Firmware. This will test board communication and serve to tell the Board that you will be using the API (binary) Command Interface.

The default values are COM1 and 19.2K which are probably ok for you, but you must do the [API Connect] Task to open the COMM Port - it does not happen automatically.
Retention of Settings: The settings are not “remembered” the next time you run the GUI.

Changing the baud rate: If you want to change the baud rate, change the board setting first. That way you can use this program [Parms]/[Write…][EEPROM] to make the board change while still communicating at a compatible baud rate. Then reset the board (so that it will now be using the new baud rate), and finally use this dialog to connect at the new baud rate.

Available COM Ports: The GUI scans for available COM ports on your PC. This includes any virtual COM ports (USB Converter) that might be present. This allows you to use a USB adapter if your PC doesn’t support a traditional COM Port. Note that just because the application finds a COM port doesn’t mean that it can use it - another application may have gotten there first and it may be currently in use.

The [Do Task] Button: When you click the [Do Task] button, the checked tasks will be executed. The default is to do both a Board Reset and an API Connect. You must do the API Connect Task before any board communication can take place. The reason for this is the board must be “told” that you will be using the API Interface for communication. There may be occasions where you will want to connect without doing a board reset (eg; preserve board RAM settings, but reestablish lost COM channel), but you will probably not have need to reset the board without the API Connect. Just as with all the other means of resetting the board (powerup reset, pressing the reset button after powerup, Cmd-I over the serial interface), you will see a brief blink of the LED’s to indicate the board has been reset.

Connection Status: Assuming the API Connect succeeds; the Component Data will be read from the board and replace the “?” marks in the edit boxes. This is your confirmation that the GUI and the board are successfully communicating. If additionally you have checked the Reset Board box, confirmation of the reset can be seen by a brief blink on the boards programmable LED’s.

Firmware Version Warning: If you connect to a board with an incompatible firmware version you will get a warning. If you proceed anyway to use the program with incompatible firmware, don’t say you weren’t warned.
6  [Parms] Menu

This menu brings up submenus for “Edit”, “Read from board”, and “Write to board” operations. You select the [Edit] submenu to bring up tabbed dialogs that will allow edit of any of the GUI versions of the motor control parameters. You select the [Write…] submenu to record all of the GUI settings to one of the two board memories. You select the [Read…] submenu to read all of the board settings into the GUI from one of the two board memories. The Read and Write operations copy the entire parameter set. There is no provision that would allow only selected parameters to be copied.

The [Edit] Submenu will open a set of tabbed dialog boxes that will allow any of the GUI version of the Dalf Parameters to be changed. More detail on the tabbed dialog, including screen shots, appears a bit later. Note that the Edit operations by themselves do not change any of the board settings.

The [Read…] Submenu will allow you to initialize all of the GUI version parameters from the board memory (RAM or EEPROM). Generally the RAM and EEPROM versions will be identical, but not always - see the [Write…] option below.

The [Write…] Submenu allows you to write the GUI version of the parameters into the selected board memory. This can be RAM or EEPROM - your choice.

Tips For Initializing the GUI version of the parameters:

1) The GUI default parameters are constants recorded within the GUI Application at the time of compilation. Depending on which version of firmware is running on your Dalf Board, the GUI defaults may not match the factory defaults located in flash memory on your board. Confused yet? A good general rule when you are beginning a parameter edit session is to replace the GUI default parameters by reading a previously saved version from your hard drive or by reading the current values stored in the board EEPROM.

2) You will probably find it useful at some point to record the original factory defaults from your board to your hard drive. In this way you can later easily reload your factory defaults into the GUI. If you have previously made changes to the Parameter Block values on your board, there is an additional step needed to ensure that you don’t lose your settings. You should first save your current settings to your hard drive using the [Parms/Read…/EEPROM] and [File/Save As…] menus - naming the file something like MySettings.dlf. Now follow the Owner’s Manual instructions to restore the factory defaults to the boards Parameter Block in EEPROM (Hold down the BTN switch while doing a board reset). Depending upon how you do the board reset, you may now have to reconnect the GUI. Now that the factory settings are in the Parameter Block of boards EEPROM, just use the [Parms/Read…/EEPROM] to read them into the GUI and the [File/Save As…] menu to save them to your hard drive - naming the file something like Factory.dlf. Don’t forget to use the [File/Open…] and [Parms/Write…/EEPROM] menus to restore your original settings from MySettings.dlf. You will need to do another board reset before they will be in use.
3) During testing and evaluation you may want to keep several versions of motor parameters on your hard drive. In this way you can quickly restore a previous setups to the board. This is particularly useful if you are testing different hardware configurations (e.g., different motors or motor encoder combinations).

### 6.1 [Parms/Edit] Menu

If you select the Edit Menu above, you will be provided with a set of tabbed dialog boxes that will allow you to set the desired GUI Parameters. After you are satisfied with your changes you select [OK] button in the dialog to exit the edit process. The changes to your settings will be remembered as the current GUI settings until the application is terminated. If you want these settings to be recorded on the board you will have to use the [Parms/Write...] submenu. If you want to record them on your hard drive you use the [File/Save As...] menu.

The tabbed dialog boxes that appear when you select the [Edit] submenu to make parameter changes are mostly self-explanatory, especially if you have experience doing things the “hard way” with the Terminal Emulator. In the next several pages I will provide a screen shot of each tab of the dialogs and provide a brief discussion. For a description of the parameters and how they are used by the Dalf Firmware you should refer to the Dalf Owner’s Manual.

The tabbed dialogs contain edit boxes for values that should be entered (decimal, not hex), scroll boxes for selections that are limited to a few options, check boxes to select various on/off options, and radio buttons for mutually exclusive options..
6.1.1 **[Timing] Dialog**

The Timing Dialog groups Dalf Parameters that have some sort of timing aspect to their function.

### Parameter Block Data

<table>
<thead>
<tr>
<th>Timing</th>
<th>Motor 1</th>
<th>Motor 2</th>
<th>Calibration</th>
<th>Miscellaneous</th>
<th>I/O Expander 1</th>
<th>I/O Expander 2</th>
<th>Unused</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFWM</td>
<td></td>
<td></td>
<td></td>
<td>IFWM Controls the pulse width modulation frequency. The choice largely depends on the inductance of your motor. Lower frequencies can produce audible noise.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC Timing Controls</td>
<td></td>
<td></td>
<td></td>
<td>AD_ACQ</td>
<td>AD_ACQ is the signal acquisition time. Largely controlled by your source impedance. Larger impedance requires longer acquisition time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AD_CNV</td>
<td>AD_CNV is the signal conversion time. Largely controlled by the number of bits to convert (10) and the ADC clock rate (20 MS should be fine).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AD_GAP</td>
<td>AD_GAP is the inter-channel gap time. This allows the processor to accomplish other stuff between ADC operations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Overall ADC Throughput:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With 7 channels (AN0, AN6), the time interval between sampling and conversion of a given channel is roughly:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T = 7*(acq + cnv + gap + latency)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Processing latency is about 50 uSec per channel.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SYSMODE

- Enable either Idle or Cmd Interface timeout
- Marginal power savings (Idle Mode)

### Rate Controls

- **PSP** (nsec): Rate at which motor commands are issued based on pot readings.
- **RCSP** (nsec): Rate at which motor commands are issued based on R/C inputs.
- **CMDSP** (nsec): Rate at which the main loop monitors the Cmd Interface for timeout.
- **AMINP** (nsec per 1% PWM duty cycle change): Slew rate [acceleration] control for open loop motor operation. Decrease for faster response.
It is interesting to compare the process of changing a parameter using the GUI versus a similar operation using the TE Interface with a Terminal Emulator Application.

**Example:** Change the PWM frequency to 13,966 Hz.

**TE:** You look up (Dalf Owner’s Manual) the location of the fPWM parameter in EEPROM. You find that the fPWM parameter is a single byte located at offset 0x00 in the EEPROM. You see from the list of table entries that the value of the byte to achieve the desired 13,966 Hz frequency is 0x0A. You use Cmd_W to write the 0x0A value into the External EEPROM Memory at address 0x0000:

```
“W 02 00 00 0A”
```

After a board reset the new PWM frequency will be active.

**GUI:** If you haven’t done so already, you would probably first use [Parms/Read…/EEPROM] to fetch all of the current board parameter settings to the GUI. Then you would use [Parms/Edit] and select 13,966 Hz in the scroll box. Click [OK] to terminate the edit dialogs and use [Parms/Write…/EEPROM] to record all of the parameters (including the fPWM change) back to the board EEPROM. After a board reset the new PWM frequency will be active. Notice that you didn’t need to know where the parameter was located in EEPROM and there was no table or Cmd syntax look up required. Of course the GUI did exactly the same operation as the TE did, namely write 0x0A to offset 0x0000 in the External EEPROM.
6.1.2  [Motor 1] Dialog

The Motor 1 dialog groups those Dalf Parameters specific to Motor 1 operation.

It is worth repeating that your changes using the [Parms/Edit] menu are to the GUI copy of the Parameters. You will need to use the [_Parms/Write...] if you wish to record the changes into the board memory.
6.1.3 **[Motor 2] Dialog**

The Motor 2 dialog groups those Dalf Parameters specific to Motor 2 operation. This is a clone of the previous dialog, but with application to Motor 2.

Note that the factory defaults for the Motor 1 and Motor 2 dialogs may not be identical.

![Parameter Block Data Dialog](image)

- **Mode1 Control**
  - Value: 0x10
  - Options: Fan, Analog, SumHD, Rel, Trig, AreadsB, DisLo

- **Mode2 Control**
  - Value: 0x00
  - Options: PotMix, PotServo, RcServ, RcMix, RcNrm, PotF, PotC, CmdOnly

- **Timing Related Parameters**
  - VSP: Velocity Sampling Period (msec)
  - VMID: PID Midcourse Velocity
  - ACC: PID Acceleration (ticks/VSP^2)
  - TPR: Ticks Per Revolution (° / CPR)

- **PWM Duty Cycle Limits (Speed Limits)**
  - VMIN: Min duty cycle (0-100%; PID)
  - VMAX: Max duty cycle (0-100%)

- **PID Control Constants**
  - KP: PID proportional constant (Gain)
  - KI: PID integral constant (Steady State)
  - KD: PID derivative constant (Damping)

- **Servo Position Limits**
  - MIN: Minimum position (ticks)
  - MAX: Maximum position (ticks)
6.1.4 [Calibration] Dialog

The Calibration Dialog groups Dalf Parameters that have calibration characteristics. These parameters affect Pot and R/C Operating modes. There is also a parameter that affects operation of the Battery Voltage Measurement and monitoring feature.

**Parameter Block Data**

<table>
<thead>
<tr>
<th>Battery Controls</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1562</td>
<td>VBCAL</td>
<td>Calibration constant</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16000</td>
<td>VBWARN</td>
<td>LowBatt warning level (mV)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VBCAL allows you to adjust the output measured by the system to accommodate resistor variances.

VBWARN controls the LowBatt indication on the LED.

<table>
<thead>
<tr>
<th>Radio Controls (mapping parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>RCD</td>
</tr>
<tr>
<td>F/C Deadband (μS)</td>
</tr>
</tbody>
</table>

RCMN and RCMAX parameters for each channel should match your receiver output at full switch travel (min and max). This will ensure that PWM power levels to the motor will correspond well to your switch positions.

RCD is an area around the neutral (nominally 1500 microseconds) area to ensure that no power is applied to the motor when your switch is in the neutral position.

<table>
<thead>
<tr>
<th>Current Limit Controls [Pot Settings]</th>
</tr>
</thead>
<tbody>
<tr>
<td>144</td>
</tr>
<tr>
<td>112</td>
</tr>
<tr>
<td>144</td>
</tr>
<tr>
<td>112</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>POT1A</td>
</tr>
<tr>
<td>WPA tap (V1H Test Point)</td>
</tr>
<tr>
<td>POT1B</td>
</tr>
<tr>
<td>WPA tap (V1L Test Point)</td>
</tr>
<tr>
<td>POT2A</td>
</tr>
<tr>
<td>WPA tap (V2H Test Point)</td>
</tr>
<tr>
<td>POT2B</td>
</tr>
<tr>
<td>WPA tap (V2L Test Point)</td>
</tr>
</tbody>
</table>

The POT settings control voltage thresholds used by the circuitry that deals with overcurrent conditions. The settings are only important if you plan to enable overcurrent detection.
6.1.5 [Miscellaneous] Dialog

The Miscellaneous Dialog groups Dalf Parameters that aren’t conveniently identified as obviously belonging to other dialogs.
6.1.6 [I/O Expander 1] Dialog

The I/O Expander 1 Dialog groups those Dalf Parameters used to initialize the I/O Expander 1 Part. The I/O Expander 1 part provides 16 GPIOs routed to the 16 pin ribbon cable connector labeled J6.

Parameter Block Data

![Parameter Block Data](image)

*Comments:

The ports are made available thru an I/O Expander that communicates with the processor over the primary I2C bus at 400 kHz. The ports can be accessed with library routines or any of the command interfaces without any required knowledge of the I2C Bus.

The 16 GPIO’s are routed to a connector for off-board use (all available for custom use). The default values configure the pins for both 8-bit ports as inputs on initial power up but that can be changed by altering the IODIR registers here.

After initial configuration typical usage would require access to only the GPIO and possibly the OLAT registers. See the Dalf Owner’s Manual for additional detail.*
6.1.7 [I/O Expander 2] Dialog

The I/O Expander 2 dialog is a clone of that for I/O Expander 1, but with application to the second I/O Expander Part. The I/O Expander 2 part provides 16 GPIOs routed to the 16 pin ribbon cable connector labeled J5.

Note that the factory defaults for the two I/O Expander parts may not be identical.

![Parameter Block Data](image)

The ports are made available thru an I/O Expander that communicates with the processor over the primary I2C bus at 400 kHz. The ports can be accessed with library routines or any of the command interfaces without any required knowledge of the I2C Bus.

The 16 GPIO’s are routed to a connector for off-board use (all available for custom use). The default values configure the pins for both 8-bit ports as inputs on initial power up but that can be changed by altering the IODIR registers here.

After initial configuration typical usage would require access to only the GPIO and possibly the OLAT registers. See the Data Owner’s Manual for additional detail.

Note that the factory defaults for the two I/O Expander parts may not be identical.
6.1.8 [Unused] Dialog

The Unused Dialog groups the currently unused bytes in the Dalf Parameter Block. Note that future firmware releases may utilize a portion or all of these resources.
6.2 [Parms/Read…] Menu (GUI ↔ Board)

If you select this menu, you will be provided with a submenu that allows you to select the source of the data that will be copied into the GUI Parameter Settings.

Select [EEPROM] to copy the Parameter Block Settings from the External EEPROM on the Dalf Board to the GUI.

Select [RAM] to copy the Parameter Block Settings currently in the Dalf Board RAM (Dalf Documentation refers to this area of memory as ERAM).

6.3 [Parms/Write…] Menu (GUI → Board)

If you select this menu, you will be provided with a submenu that allows you to write the current GUI Parameter Settings to the selected board memory.

Select [EEPROM] to copy the GUI Parameter Settings to the External EEPROM on the Dalf Board (Dalf Documentation refers to this area of the EEPROM as the Parameter Block).

Select [RAM] to copy the GUI Parameter Settings into the Dalf Board RAM (Dalf Documentation refers to this area of memory as ERAM).

It is important to note that depending on what firmware image is running on your Dalf Board, the GUI Defaults and the Factory Defaults shipped with your board (in flash memory) may not match.

7 [Board] Menu (Board Status)

The Board Menu provides a single submenu that allows you to view the current status of the board.

Select [Status] to bring up the dialog shown below.
Select the [Update View] button to update the screen with the current board status. You will obtain something similar to the screen shown below (with different values). All values are read from board RAM (not EEPROM), hence reflect the current board operating condition. The Dynamic Quantities Section shows values that could be changing between updates in this dialog. The clock indicates the RTC on the board (showing that only 6 seconds have elapsed since the board was last reset.

**Tip:** This screen is quite useful for R/C Tuning and analog testing (See Dalf Owner’s Manual).
8  [Window] Menu (Main Attraction!)
If you select this menu, you will be provided with a submenu that allows you to select among the major
GUI features. This menu also allows you to return to the Home Screen.

Select [Home] to return to the Home Screen.
Select [OpenLoop] to exercise Open Loop Motor Control
features. The graphical output shows the Dalf Board control
outputs. Of course if you want to actually drive the motors
with this panel, you will need motor and driver hookups.
Select [ClosedLoop] to exercise Closed Loop Motor Control
features. This option requires motor, driver, and motor position
(encoder) hookups. Optimal performance will depend on use
of appropriate PID parameters (see PID Tuning).
Select [PID_Tune] to utilize a graphical aid to assist you in
determining the appropriate PID parameters for your particular
application.

8.1 [Home]
Selection of this menu just closes the current screen and returns you to the Home Screen shown earlier. It
will also close motor sampling and display “threads” from the OpenLoop and ClosedLoop screens that can
produce some annoying blinking.

8.2 [OpenLoop]
Selection of this menu brings up the OpenLoop Screen shown below. The screen is organized with 2
window panes. The leftmost pane has motor controls that you manipulate, and the right pane show status
of the resulting motor control signals. The controls are organized in two groups.

Command Arguments
The first group of controls consists of the Slew Rate Edit Box, the two motor control sliders, and the Go
checkboxes. These controls provide the arguments for the actual motor commands that will be issued to
the Dalf Board (when the Go box is checked). For example, if you enter a slew rate of 50 msec, move the
motor 1 slider to 30%, and check the Go1 Checkbox, the motor1 PWM output will ramp to 30% in the
forward direction, with a slew rate of 1% change every 50 msec (1.5 seconds to 30% power). The
graphical output shows the actual resulting Dalf Board outputs obtained by periodic board sampling (not a
simulation).

Parameters
The second group of controls consists of the AMINP, VMAX1, VMAX2 Edit Boxes and the Write
Controls Checkbox. These controls can be used to affect the way that the open loop commands operate.
They correspond to parameters in the Parameter Block on the Dalf Board (in EEPROM) and it is the
version of these parameters in Dalf RAM (ERAM) that actually affect the command operation. When you
check the Write Controls Checkbox, the AMINP, VMAX1, and VMAX2 control values will be written into
the corresponding RAM variables on the board. Subsequent open loop command operations will be
affected. Note that this is an example where the board parameters in RAM and those in EEPROM might be
different.
Refer to the Dalf Owner’s Manual for detailed parameter descriptions, but here is the “executive summary”:

- AMINP - Acceleration Minimum Period (Slew rate limit).
- VMAX - Maximum PWM duty cycle (Slider limit; maximum power to motor).

This snapshot of the OpenLoop Screen was captured shortly after starting both motors using the sliders and Go Checkboxes. It is important to note that the graphical output is independent of motor loading. You are seeing the actual power level being applied to the motors, but you would see the same graph even if the motors were stalled.

**Things to try:**
Experiment with changing the slew rate. Larger values will produce slower start and stop times. Notice that AMINP acts as a governor on the slew rate. In this example, changing the slew rate to anything below 10 will have no effect. Experiment with changes to the AMINP control to allow faster start and stop response. VMAX corresponds to the dashed red lines and you will be able to see the changes graphically. Change VMAX to 50 and then try to drive the corresponding motor to 75% duty cycle.
8.3  [ClosedLoop]

Selection of this menu brings up the ClosedLoop Screen shown below. The screen controls in the left pane consist of the command arguments for the closed loop move commands (TGT, VM, ACC). There are also Enable[] and Move[] checkboxes to start the move. This pane also shows some settings (Type, Trig) that affect how the move will be performed. The Type setting is either “Relative” or “Absolute”. The Trig setting is either “Triggered” or “Un-Triggered”. The Settings reflect the current control bits in the board RAM (see the MODE1 Parameter) and cannot be changed from this screen. When you click the Enable[] checkbox, the MODE1 parameter will be read from board RAM and used to update the settings. In addition, the motor starting position shown in the display will be updated. The move(s) will not start until the Move[] checkbox is checked.

Command Arguments
The TGT argument determines the move destination. In Relative mode the destination will be START + TGT, while in Absolute Mode the destination will be TGT. The ACC argument controls the acceleration and deceleration phases of the move while the VM argument controls the midcourse velocity. Note that these values are scaled by 256. Encoder ticks are the distance component of units and VSP periods are the time component of the units.

Example: Your application setting for VSP is 5 msec and your encoder has TPR=80. You enter a midcourse velocity of 2000=VM*256. Then VM = 2000/256 = 7.8125 ticks/VSP. There are (1000/5) VSP periods per second, 60 seconds per minute, and 80 ticks per revolution so the resulting midcourse velocity is

\[
\text{Vel} = \frac{2000}{256} \times \frac{1000}{5} \times \frac{60}{80} \\
= 1,171 \text{ RPM}
\]

Similarly you can compute acceleration in your favorite units from the ACC*256 entry.

Settings
The Trig and Move Type settings in the GUI can be changed by altering the MODE1 Parameter(s) using the [Parms/Edit] menu. You can then use the [Parms/Write.../RAM] menu to record the changes in the board RAM. If you now revisit the Closed Loop Control Panel and check the Enable[] checkbox you should see the changed settings. The Type setting affects how the TGT argument is used as previously explained. The Trig setting needs a bit of explanation to clarify how it is used in the context of this GUI Control Panel.

Example: Assume that you want dual motor moves to start at the same time. If you have entered the arguments, checked both Enable[] boxes and then check the Move[] checkbox, it will appear that both motors start simultaneously. Actually that is the case only if the current Trig Setting is “Triggered”. If the setting is “Un-Triggered”, then two commands (Cmd_Y) will be sent to the Dalf Board (one command for each motor). The commands will be sent sequentially, so one motor will actually get a slight headstart on the other. If instead, the Trig Setting is “Triggered”, then three commands will be sent to the Dalf Board. The two Cmd_Y’s will be sent as in the “Un-Triggered” case, but neither motor will start until the third command (The Trigger Command - Cmd_T) is received. In this way, the motors will start the moves simultaneously. You probably wouldn’t be able to tell a difference here, but if it were critical that the motors be in sync, you would use the triggered command mode.
The screen below is shown after completion of a dual motor move with the settings shown. The settings were Absolute and Non-Triggered for both motors. The motor destinations are the same and both motors both start near the encoder origin. The midcourse velocities and acceleration differ. In this example, motor2 arrives at the destination first.

The graphs show the actual motor velocity that is being sampled during the course of the move. Notice the “noise” riding on the velocity signal. This does not indicate a non-smooth motor velocity profile. It simply tells you something about the integral sample values for velocity. Listen to your motor during the course of the move. If you have properly PID tuned your motor, its operation should be quite smooth despite the “noisy” graph. There are a couple of reasons why you will see the “noise”. First there is the fact that the encoder operation is asynchronous with respect to the velocity sampling. This means that even with absolutely constant motor velocity, and identical sampling periods, a particular VSP sampling interval will frequently contain one more encoder tick than its neighboring interval - just because of where the interval begins relative to the encoder position. Secondly there is simply the quantization error. The specified velocity is frequently fractional, yet the sampled velocity in ticks/VSP is integral. In the earlier example, VM = 7.8125 ticks/VSP, and the velocity samples would generally be 8 ticks/VSP, but occasionally you would get a sample with 7 ticks/VSP. The noisy graph is exactly what you would expect if the motor is smoothly tracking the specified position profile.
8.4 [PID_Tune]

The function of this portion of the application is to assist you in determining the proper PID controls for your particular closed loop motor application. It accomplishes this by showing you graphically the motor PID Step response which results from your choice of command arguments and PID controls. Selection of the [Window/PID_Tune] menu brings up the PID Tuning Screen shown on the next page. The screen consists of two panes. The right pane shows the graphical response to the command. Screen controls in the left pane of the window allow you to manipulate command arguments and the PID parameters that affect the results. The controls consist of two groups:

Arguments
The first group consists of the command arguments for the PID Tuning Command (Cmd_Q). These are the motor# (Motor#), the motor target (TGT) destination, and the number of samples of PID Err (NPID) that will be collected during the execution of the command. The NPID parameter simply controls how long we “look at” the graphical response. The TGT determines how long the motor will run (number of revolutions is directly related to encoder ticks) to achieve the destination. Generally 4 to 6 revolutions of the motor will provide sufficient data and NPID is chosen to show us a graph which includes target acquisition and a bit more. See the Owner’s Manual for a discussion of Cmd_Q.

For example, if your encoder spec is CPR = 64 (TPR = 4*64 = 256), you would set TGT = 5*256 = 1280 to get 5 revolutions of data for the graph. The y-extent of the graph (Ticks) is controlled by your selection for TGT. The x-extent (time) of the graph is controlled by NPID * VSP. There will be NPID samples spaced VSP msec apart. If NPID is 150, and VSP is 20, your graph will show motor behavior for 20*150 = 3000 msec = 3 seconds after the command is issued which is generally enough.

Control Parameters
The second group of controls consists of parameters that affect the outcome of Cmd_Q. These consist of the PID parameters (KP, KI, KD), the sampling interval time (VSP), bounds on the motor PWM duty cycle (VMIN, VMAX), the limit on PID Err (MAXERR), and a clamp on the PID summation term (MAXSUM). You will probably be most interested in the KP, KI, KD, and VSP parameters. The values of all 8 parameters are written into the board RAM Memory (ERAM area) just prior to execution of Cmd_Q and affect the outcome of the command. Should you wish to use them, the 3 checkboxes to the right of this group just provide a convenient way to initialize the controls as a group. See the Owner’s Manual for a description of the various control parameters.

The Graph
When the Do Cmd_Q[] check box is checked, the controls are written into ERAM, and the command (Cmd_Q) with your arguments is issued to the board. This command “tells” the board to drive the motor to the target position using PID. The board will collect PID Err data during the command which will be transmitted back to the GUI program for construction of the graph in the right pane.

The vertical axis in the graph is the PID Err. The horizontal axis is time. Since the encoder is reset to zero just before this command is started, the initial PID Err is TGT. The PID Err should rapidly diminish to near zero during the course of the command. If the system is under damped (KD too small) or over powered (KP too large) you may see overshoot (the graph will cross the x-axis). If over damped (KD too large) or under powered (KP too small) you may see a graph which never completely “arrives” at the x-axis (PID Err zero). Finally, you may need some non-zero KI term to reduce the “steady state” error (constant non-zero PID Err).
Example:
The screen below shows the window after executing Cmd_Q with the specified command arguments and PID Controls. The encoder used for the motor in this example had TPR=80, so that TGT=1000 corresponds to 12.5 revolutions. The graph starts at time 0 with a PID Err of TGT=1000 (off the bottom of the screen) and rapidly rises to the TGT position (PID Err zero) after roughly t=40 sampling periods (40*5msec = 200 msec). A series of oscillations begins at this point with the motor finally settling near the destination at about t=85 (425 msec). Notice that there is no damping (KD = 0) and there is no summation term (KI = 0) provided in this PID example. The peak oscillation is nearly 100 ticks (initial over shoot exceeds one full motor revolution!). Clearly the graph illustrates the need for some PID control improvements.
The screen below shows the same setup but with a little damping added (KD = 5000). The motor arrives at the destination slightly later (t=50), but now the oscillation is not observable. Since the motor settles at zero PID Error, there is no need for the summation term and we leave KI=0.

As a final step, we test these PID control parameters using the [Parms/ClosedLoop] portion of the application to verify satisfactory performance with various closed loop move operations. That’s it - we’re done, but don’t forget to record the parameters in the board eeprom (Parameter Block Memory).

**Tips**

1) After you have used the PID Tuning feature to establish satisfactory PID controls for your application, (the control changes are still in the board ERAM memory), try them out with some motor moves in the Closed Loop Panel ([Window/ClosedLoop] menu). A good measure of your success in establishing proper controls is to issue various closed loop motor move commands and compare the final motor position with your TGT argument. Remember, in the closed loop panel, you can simply toggle the Enable[] check box to see the actual motor position after the move is complete.

2) The PID Tuning Window will get you close to your goal of “perfect PID”, but it is not unusual to find that you have to make some minor “tweaks” to the parameters to achieve optimal performance. If you are stopping short of your target for example you might want to increase the gain (KP) a bit.