WaveSculptor Driver Controls User's Manual

21 March 2007

Brisbane, Australia
http://www.tritium.com.au

TABLE OF CONTENTS

1 INTRODUCTION

This document details the interface, installation, and usage requirements for the Tritium WaveSculptor driver controls product. It also provides information on programming the device to implement custom functionality.

The driver controls provide an easy way to control a Tritium WaveSculptor motor controller. The driver controls come programmed from the factory configured with sensible default values that will work in a plug-and-play manner with a motor controller.

The microcontroller firmware for the device (in 'C') is available on the Tritium website under an open-source license, as are the hardware schematics and component overlays.

For further details on overall system configuration, please refer to the Wiring Engineering Reference doucment (TRI50.010), available on the Tritium website.

2 DEVICE OVERVIEW

The driver controls provide an analog and digital input interface to the CAN bus used by the WaveSculptor system. Inputs are provided for:

- 2 digital quadrature encoders, with pushbutton
- 2 analog potentiometers
- 12 digital switches

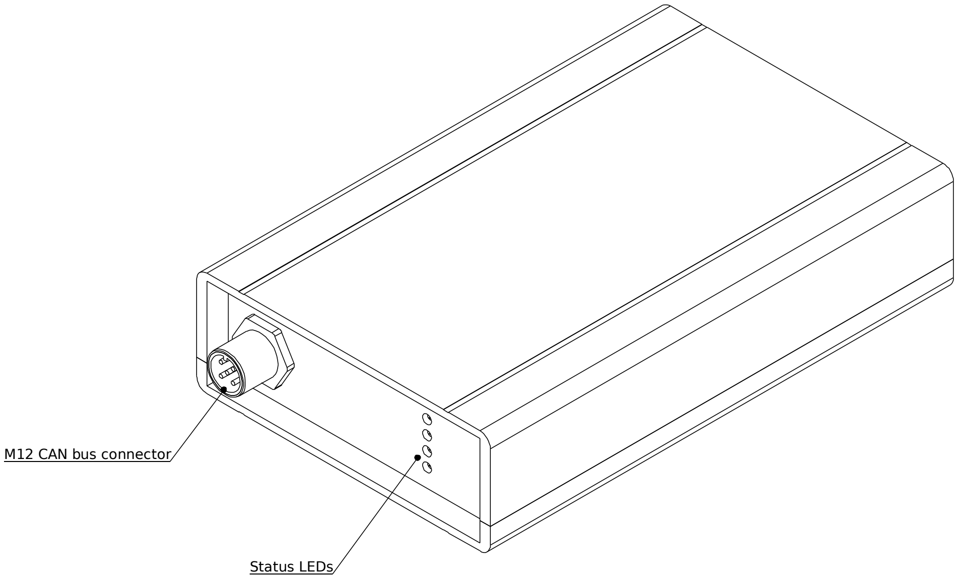
Output to the vehicle is via a CAN bus running at 1 Mbit/second, using a fixed base address of 0x500. Alteration of either the bit rate or the base address currently requires reprogramming the microcontroller, although hardware support does exist to allow remote configuration in the future.

Four LEDs are provided on the front panel of the device to indicate status.

For custom firmware development, an internal pushbutton and LED are available for debugging purposes.

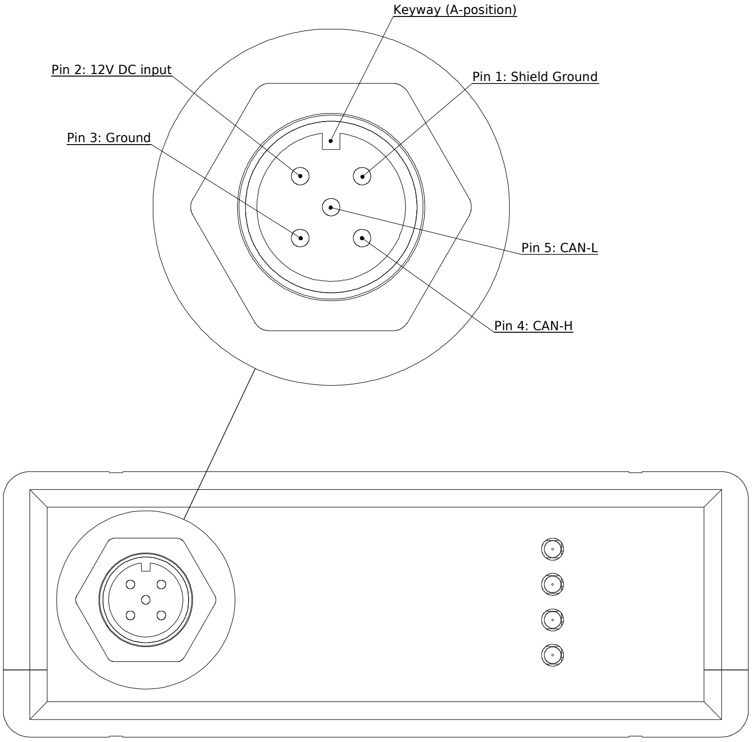
3 FRONT PANEL

The following illustration shows the connections and indicators on the front panel of the driver controls. A CAN bus connection and four status LEDs are present on this panel.



3.1 CONNECTOR

The CAN bus connector is a male 5-pin M12 screwlocking industrial standard DeviceNET connector. The pinout is shown below:



3.2 LEDS

By default, the firmware in the driver controls shows the following information on the front panel LEDs:

LED label	*Colour*	*Function*
CAN Activity	Green	Flashes on CAN activity associated with the driver controls
Error	Red	Illuminates on CAN bus errors such as loss of comms
Warning	Yellow	Unused
Status	Green	Toggles at the input sampling rate (~ 10 Hz)

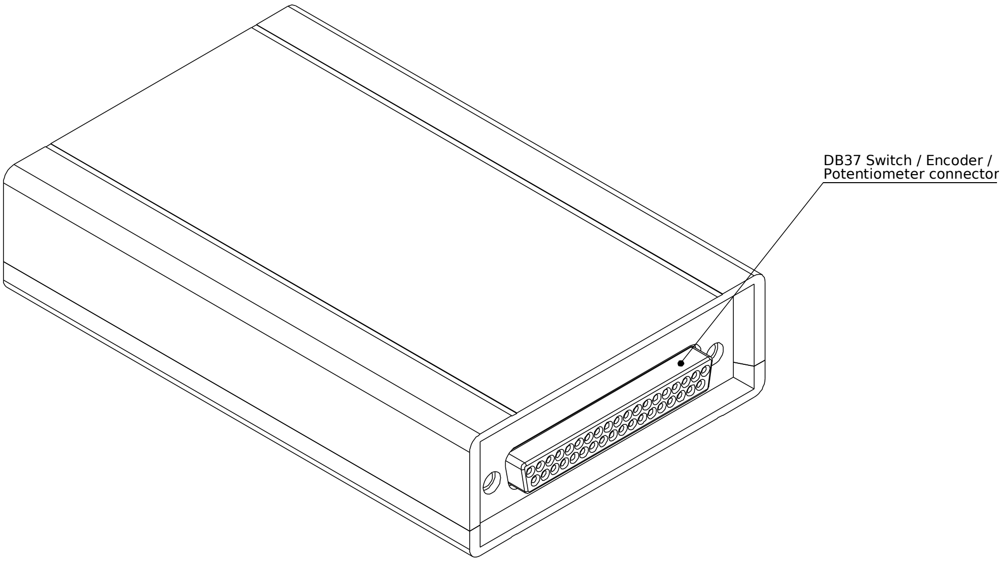
3.3 POWER INPUT

Power for the driver controls needs to be supplied along the CAN bus cable. The driver controls contain an internal switch-mode supply and will operate successfully in the range between 9V and 15V DC.

The voltage supplied to the power input is also used directly as the pull-up voltage on the switch inputs, to provide a large voltage swing on each input and reduce the effect of external interference and noise on the driver controls.

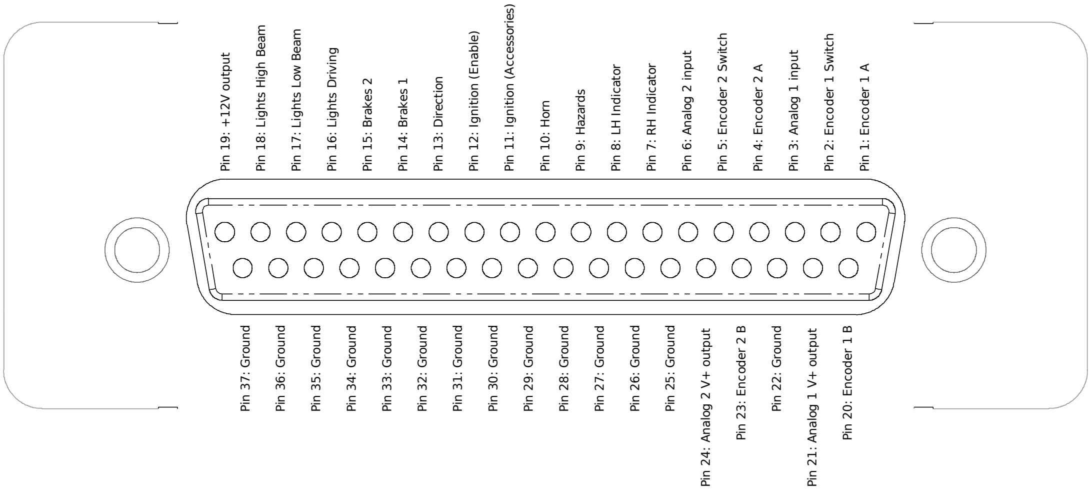
4 REAR PANEL

The following illustration shows the connections on the rear panel of the driver controls. A DB37 connector for switch, potentiometer and encoder inputs is present on this panel.



4.1 CONNECTOR

The input connection is provided as a female DB37 socket. The following illustration shows the pinout as viewed when looking into the driver controls, ie the same view you have when soldering or inserting crimps into the rear of a mating connector housing.



4.2 PIN FUNCTIONS

By default, the firmware in the driver controls uses these inputs as follows:

Input	***Function***
Analog 1	Motor current setpoint, connect to accelerator (gas) pedal position
Analog 2	Velocity setpoint
Encoder 1	Bus current setpoint
Encoder 2	Unused
Switch 1	Right-hand indicator (blinker)
Switch 2	Left-hand indicator (blinker)
Switch 3	Hazards (4-way blinker)
Switch 4	Horn
Switch 5	Ignition 'accessories' position
Switch 6	Ignition 'run' position (vehicle enable)
Switch 7	Forward / Reverse
Switch 8	Brake switch / pedal 1
Switch 9	Brake switch / pedal 2 (redundant spare)
Switch 10	Lights – side / running lights
Switch 11	Lights – headlights low beam
Switch 12	Lights – headlights high beam

All switch inputs are identical in hardware, so for custom firmware applications these pin positions are not fixed and may be used in any order or combination desired. Each encoder is also treated as three switch inputs (A, B, pushbutton) and if not used as an encoder, may be configured as three individual switch inputs.

4.3 ENCODER CONNECTIONS

The encoder inputs are designed to be driven with a low-cost mechanical (switch) type quadrature encoder, such as the Piher CI-11 family of devices. An optional switch input is also associated with the encoder inputs to allow connection of encoders with built-in pushbuttons.

All encoder inputs are active low, with an internal pull-up resistor provided by the driver controls. Connect each encoder channel between it's input pin and ground, so that when the switch is active, the pin is shorted to ground.

Although not provided by the default device firmware, the Analog V+ output pin can be driven to a variable voltage by the driver controls microcontroller. If this pin is not used as a potentiometer drive pin, then it may be used as an output for a variable brightness LED, to provide feedback to the driver for items such as encoder position.

4.4 POTENTIOMETER CONNECTIONS

Each analog input expects a voltage of between 0.0 and 3.3V. Filtering is provided inside the driver controls to attenuate incoming signals with a bandwidth above approximately 10 Hz.

The driver controls provides a variable output voltage to drive the potentiometer. By default, this voltage is fixed at 3.3V, but can be varied with custom firmware if necessary. This output is provided through a 330 Ohm resistor, to limit current in the event of a fault.

A linear potentiometer of 10kOhms or higher should be used, to avoid excessive loading the 330 Ohm resistor and affecting the input full-scale voltage measurement. Connect the potentiometer between the V+ output (Pin 21 or 24) and Ground (Pin 22 or 25) with the wiper connected to the appropriate input (Pin 3 or 6).

A 10 Mohm resistor is connected internally between each analog input and ground, to ensure that an open (or broken) input reads at 0V.

4.5 SWITCH CONNECTIONS

All switch inputs are active low, with an internal pull-up resistor provided by the driver controls. Connect each switch between it's input pin and ground, so that when the switch is active, the pin is shorted to ground.

5 PROGRAMMING

5.1 OVERVIEW

The driver controls are based around a Texas Instruments (TI) MSP430 16-bit embedded low-power microcontroller, operating from a 4 MHz clock as default. Operation at 2 or 8 MHz is also possible. CAN bus support is provided via a Microchip MCP2515 CAN controller and TI SN65HVD234 CAN transceiver. The driver controls are not isolated from the CAN network.

5.2 SCHEMATICS & SOURCE CODE

Schematics and PCB component position overlays in PDF format are available for download on the Tritium website. A zip file is also provided containing the default source code for the microcontroller, written in 'C' and available under a BSD open-source license.

Please refer to these references if developing custom firmware for the driver controls.

5.3 TOOLCHAIN

The example default code provided is configured to work with the freely-available open-source MSP430 GCC toolchain, which provides a command-line driven compiler, binutils, download, and real-time debug capability through a JTAG header present on the driver controls PCB. Please refer to the README file with the source code for download and installation instructions.

An adapter is provided with every driver controls product that converts the TI standard 14-pin JTAG debug header to the smaller 8-pin flexible printed circuit (FPC) header used on Tritium devices. Tritium recommends the use of the USB programmer part number MSP-FET430 UIF available from TI or their distributors, although lower cost and slower performance parallel-port devices are also useable.

5.4 CODE DOWNLOAD

Connect the 8-pin FPC ribbon to the header on the driver controls PCB, with the Pin 1 indication arrows matching on both the driver controls and the adapter board.

Follow the instructions in the README file to compile your 'C' source, produce an object file for loading into the microcontroller, and download the new firmware to flash memory in the microcontroller. Please feel free to email any questions or comments to James Kennedy, james@tritium.com.au.

6 REVISION RECORD

REV	***DATE***	***CHANGE***
1	21 March 2007	Document creation (JMK)