WaveSculptor CAN Bus Communications Protocol Specification

20 August 2007

©2007 Tritium Pty Ltd
Brisbane, Australia
http://www.tritium.com.au

TABLE OF CONTENTS

1 INTRODUCTION

This document describes the protocol used to communicate over a CAN bus connection to the WaveSculptor motor controller.

2 CAN CONFIGURATION

2.1 HARDWARE

The CAN hardware interface used is compatible with the CAN 2.0B standard. The supported bit rates (bits per second) are 1 Mbps (default), 500 kbps, 250 kbps, 125 kbps, 100 kbps and 50 kbps.

2.2 SOFTWARE

The CAN protocol uses data frames for most communication. Remote frames are also enabled. The identifier field uses the standard frame definition length of 11 bits, with identifiers 0x7F0 to 0x7FF reserved for use by the WaveSculptor bootloader. The bootloader will send a message with identifier 0x7F1 at 1Mbit after a system reset. All measurement data is transmitted using IEEE single-precision 32-bit format (IEEE 754) with most significant byte (MSB) sent first.

Bit Length	1	11	6	8 Bytes	16	2	7
	Start	Identifier	Control	Data Field	CRC	Ack	End

Figure 1. CAN data frame

2.2.1 Identifier

The identifier field has been split into two sections for Tritium devices. Bits 10-5 contain the device identifier and bits 4-0 contains the message identifier associated with that device, as shown in Figure 2. This means that there is a maximum of 63 Tritium device that can be on the CAN bus at any one time. The 64th location is reserved for the bootloader. Each Tritium device can have 31 different types of messages. The first message identifier is used by the device identification message. Two device slots could be used if more messages per device were required, however this has not been required yet.

The term Base Address is used instead of Device Identifier in the Windows configuration program. Base Address is simply the Device Identifier multiplied by 32. Using this term makes it more obvious what range of CAN identifiers have been used. As an example, there may be a Tritium driver controls at base address 0x400, a left wheel WaveSculptor motor controller at 0x420 and a right wheel WaveSculptor motor controller at 0x440. So, address range 0x400 – 0x45F would be used by this Tritium system.

10 … 5	4 … 0
DEVICE IDENTIFIER	MESSAGE IDENTIFIER

Figure 2. CAN device identifier address format

2.2.2 Data Field

The data field in all frames is fixed at 8 bytes (64 bits) and two IEEE 754 32-bit floating point variables as shown in Figure 3. The data field is sent and expected to be received least significant bite first. This allows a direct overlay of a float[2]

array and char[8] array on a little endian processor, such as, an Intel x86 (or clone) or the TI DSP in the WaveSculptor.

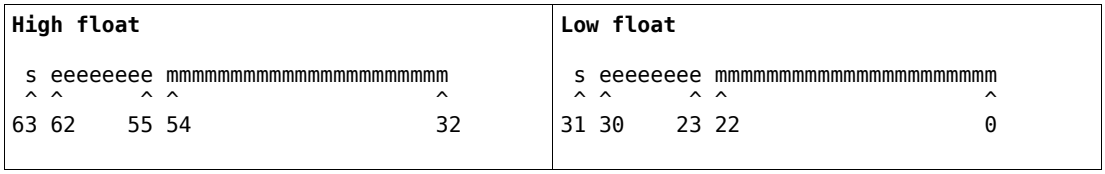

Figure 3. Format of the data field in a data frame

2.2.3 Units

Please note that variables described in the following packets with units of percent "%" should be sent with a minimum floating-point value of 0.0 and a maximum value of 1.0. Do not send 100.0 as the maximum value.

3 MESSAGE TYPES

Messages types used by the WaveSculptor are commands, identification, status, measurement & configuration.

3.1 DRIVE COMMANDS

These are data frames broadcast from the driver controls to any listening motor controller. These commands contain desired set points for the control software to operate the controller. The commands are sent as required, however there is a maximum permissible delay between consecutive Motor Drive commands.

The identifier of a received command will contain the base address of the device from which the message was sent and a message identifier offset. The base address of the sending device is a preset value stored in internal memory, which can be set during initial configuration of the motor controller using the configuration utility.

3.1.1 Motor Drive Command

ID: Driver Controls Base Address + 1

Variable	***Bits***	***Units***	***Description***
Motor Current	63 .. 32	%	Desired motor current set point as a percentage of maximum current setting.
Motor Velocity	31 .. 0	m/s	Desired motor velocity set point in metres/second.

The WaveSculptor motor controller must receive a Motor Drive Command frame at least once every 250ms. If this does not occur, the controller will assume that communications have failed and will halt all motor control functions, placing the system into neutral.

3.1.2 Motor Power Command

ID: Driver Controls Base Address + 2

Variable	*Bits*	*Units*	*Description*
Bus Current	63 .. 32	%	Desired set point of current drawn from the bus by the controller as a percentage of absolute bus current limit.
Reserved	31 .. 0	–	–

3.1.3 Reset Command

ID: Driver Controls Base Address + 3

Variable	*Bits*	*Units*	*Description*
Unused	63 .. 32	–	–
Unused	31 .. 0	–	–

Re-initialises all software modules in the DSP. Slightly different from pushing the reset button. Send a command from this address to reset the WaveSculptor.

3.2 DRIVE COMMAND EXAMPLES

Ignoring the added complexity of bus and thermal limiting, the WaveSculptor operates such that it will use the maximum available current (torque) to try and achieve the desired velocity. This is true for both accelerating and decelerating operation, i.e. the WaveSculptor will automatically regneratively brake if a setpoint velocity is provided that is slower than the current speed.

Two main drive modes will be used in a normal vehicle setup: torque control; and velocity (cruise) control. A conventional vehicle runs in torque control mode, where the position of the accelerator (gas) pedal controls the amount of torque produced by the engine. In an electric system, the motor current is proportional to torque, and can be easily regulated, as can the velocity.

To run the motor in torque control mode, set the velocity to an unobtainable value such as 100 m/s. Set the current to a value that is proportional to your accelerator pedal position. If you wish to drive in reverse, set the velocity to –100m/s. The motor will operate the same as a normal car, and will coast down to a stop if the driver removes their foot from the pedal.

To run the motor in velocity (cruise) control mode, set the current to your maximum desired acceleration force (usually 100%), and set the velocity to the desired speed. The WaveSculptor will use the setpoint current to keep the vehicle at the setpoint speed, and will use both drive and regenerative braking to do so. Use this mode to regeneretivly brake to a halt by setting current to your desired braking force, and setting velocity to zero.

3.3 MOTOR CONTROLLER BROADCAST MESSAGES

Data frames containing telemetry values are periodically broadcast onto the bus by the WaveSculptor.

3.3.1 Identification Information

ID: Motor Controller Base Address + 0

Interval: 1 second

Variable	*Bits*	*Type*	*Description*
Serial Number	63 .. 32	Uint32	Device serial number, allocated at manufacture.
Tritium ID	31 .. 0	char[4]	“TRIa” stored as a string. msg[0] = 'T', msg[1] = 'R'...

The periodic broadcast of this message cannot be disabled. It is needed to help find the motor controller on the network if the base address is lost or mis-configured by the user.

3.3.2 Status Information

ID: Motor Controller Base Address + 1

Interval: 200 ms

Variable	*Bits*	*Type*	*Description*
Reserved	63 .. 48	–	–
Active Motor	47 .. 32	Uint16	The index of the motor currently being used. Driver controls could use this to check against a switch position or to display to the driver if desired.
Error Flags	31 .. 16	Uint16	Flags indicate errors: ***Bits*** ***Parameter*** 15 .. 7 Reserved 6 A 15V rail under voltage lock out occurred 5 Config read error (some values may be reset to defaults) 4 Watchdog caused last reset 3 Bad motor position hall sequence 2 DC Bus over voltage 1 Software over current 0 Hardware over current
Limit Flags	15 .. 0	Uint16	Flags indicate which control loop is limiting the output current of the motor controller: ***Bits*** ***Parameter*** 15 .. 7 Reserved 6 Heatsink Temperature 5 Bus Voltage Lower Limit 4 Bus Voltage Upper Limit 3 Bus Current 2 Velocity 1 Motor Current 0 Bridge PWM

3.3.3 Bus Measurement

ID: Motor Controller Base Address + 2

Interval: 200 ms

Variable	*Bits*	*Units*	*Description*
Bus Current	63 .. 32	A	Current drawn from the DC bus by the controller.
Bus Voltage	31 .. 0	V	DC bus voltage at the controller.

3.3.4 Velocity Measurement

ID: Motor Controller Base Address + 3

Interval: 200 ms

Variable	*Bits*	*Units*	*Description*
Vehicle Velocity	63 .. 32	m/s	Vehicle velocity in metres / second.
Motor Velocity	31 .. 0	rpm	Motor angular frequency in revolutions per minute.

3.3.5 Phase Current Measurement

ID: Motor Controller Base Address + 4

Interval: 200 ms

Variable	*Bits*	*Units*	*Description*
Phase A Current	63 .. 32	A_{rms}	RMS current in motor Phase A.
Phase B Current	31 .. 0	A_{rms}	RMS current in motor Phase B.

While the motor is rotating at speed these two currents should be equal. At extremely low commutation speeds these two currents will only match in one third of the motor position, the other two thirds will involve current also flowing in Phase C.

3.3.6 Motor Voltage Vector Measurement

ID: Motor Controller Base Address + 5

Interval: 200 ms

Variable	*Bits*	*Units*	*Description*
Vd	63 .. 32	V	Real component of the applied non-rotating voltage vector to the motor.
Vq	31 .. 0	V	Imaginary component of the applied non-rotating voltage vector to the motor.

3.3.7 Motor Current Vector Measurement

ID: Motor Controller Base Address + 6

Interval: 200 ms

Variable	*Bits*	*Type*	*Description*
Id	63 .. 32	A	Real component of the applied non-rotating current vector to the motor. This vector represents the field current of the motor.
Iq	31 .. 0	A	Imaginary component of the applied non-rotating current vector to the motor. This current produces torque in the motor and should be in phase with the back-EMF of the motor.

3.3.8 Motor BackEMF Measurement / Prediction

ID: Motor Controller Base Address + 7

Interval: 200 ms

Variable	*Bits*	*Type*	*Description*
BEMFd	63 .. 32	V	By definition this value is always 0V.
BEMFq	31 .. 0	V	The peak of the phase to neutral motor voltage.

3.3.9 15 & 1.65 Voltage Rail Measurement

ID: Motor Controller Base Address + 8

Interval: 1 second

Variable	*Bits*	*Type*	*Description*
15V supply	63 .. 32	V	Actual voltage level of the 15V power rail.
1.65V reference	31 .. 0	V	Actual voltage level of the 1.65V analog reference.

3.3.10 2.5V & 1.2V Voltage Rail Measurement

ID: Motor Controller Base Address + 9

Interval: 1 second

Variable	*Bits*	*Type*	*Description*
2.5V supply	63 .. 32	V	Actual voltage level of the 2.5V FPGA power rail.
1.2V supply	31 .. 0	V	Actual voltage level of the 1.2V DSP power rail.

3.3.11 Fan Speed Measurement

ID: Motor Controller Base Address + 10

Interval: 1 second

Variable	*Bits*	*Type*	*Description*
Fan Speed	63 .. 32	rpm	Cooling fan speed in revolutions per minute.
Fan Drive	31 .. 0	%	Drive voltage percentage to cooling fan. If this value is above 0%, then the fan should be spinning.

3.3.12 Sink & Motor Temperature Measurement

ID: Motor Controller Base Address + 11

Interval: 1 second

Variable	*Bits*	*Type*	*Description*
Heatsink Temp	63 .. 32	°C	Surface temperature of the controller heatsink.
Motor Temp	31 .. 0	°C	Internal temperature of the motor.

3.3.13 Air In & CPU Temperature Measurement

ID: Motor Controller Base Address + 12

Interval: 5 seconds

Variable	*Bits*	*Type*	*Description*
Air Inlet Temp	63 .. 32	°C	Ambient air temperature at the ventilation inlet of the controller.
Processor Temp	31 .. 0	°C	Temperature of the internal processor.

3.3.14 Air Out & Cap Temperature Measurement

ID: Motor Controller Base Address + 13

Interval: 5 seconds

Variable	*Bits*	*Type*	*Description*
Air Outlet Temp	63 .. 32	°C	Ambient air temperature at the ventilation outlet of the controller. Unused in 20kW WaveSculptor.
Capacitor Temp	31 .. 0	°C	Ambient temperature of the internal bus capacitors. Unused in 20kW WaveSculptor.

3.3.15 Odometer & Bus AmpHours Measurement

ID: Motor Controller Base Address + 14

Interval: 1 second

Variable	*Bits*	*Type*	*Description*
DC Bus AmpHours	63 .. 32	Ah	The charge flow into the controller bus voltage from the time of reset.
Odometer	31 .. 0	m	The distance the vehicle has travelled since reset

4 CONFIGURATION COMMANDS

The Windows interface program uses most of these commands to gain direct access to the embedded data flash. Data structures used by the Windows interface program can be written and read directly to the data flash. The software modules use values within these data structures for initalisation. The only command that should be used by the driver controls is the active motor swap command.

The configuration commands use four CAN channels:

Channel	*CAN ID*
Request	Motor Controller Base Address + 18
Reply	Motor Controller Base Address + 19
Data Out (PC to WaveSculptor)	Motor Controller Base Address + 20
Data In (WaveSculptor to PC)	Motor Controller Base Address + 21

4.0.1 REQUEST: Unlock Dataflash

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 48	–	–
Command String	47 .. 0	char[6]	“UNLOCK” stored as a string. msg[0] = 'U', ...
Locked dataflash cannot be written to. After the UNLOCK command write commands will succeed. There is no reply to this packet.			

4.0.2 REQUEST: Lock Dataflash

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 32	–	–
Command String	31 .. 0	char[4]	“LOCK” stored as a string. msg[0] = 'L', ...
Locked dataflash cannot be written to. After the LOCK command write commands will not succeed. Any writes currently in progress will be stopped. There is no reply to this packet.			

4.0.3 REQUEST: Reload Configuration

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 48	–	–
Command String	47 .. 0	char[6]	“RELOAD” stored as a string. msg[0] = 'R', ...
Reinitalises all the software modules that use configuration values. There is no reply to this packet.			

4.0.4 REQUEST: Write Stream to dataflash

Variable	*Bits*	*Type*	*Description*
Address in dataflash	63 .. 32	Uint32	Flat address in the external dataflash (540672 bytes)
Length of stream	31 .. 16	Uint16	Number of bytes expected on the DATAOUT channel.
Command String	15 .. 0	char[2]	“WR” stored as a string ie msg[0] = ‘W’, ...
Opens a CAN stream into the dataflash.			

4.0.5 REPLY: Write Stream to Dataflash

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 16	–	–
Command String	15 .. 0	int16	0: Success -1: Stream already open -2: Address invalid

4.0.6 DATAOUT: Data to dataflash

Variable	*Bits*	*Type*	*Description*
Data	63 .. 0	–	Bytes of data to go to external dataflash
Packets should not be sent faster then 1kHz, otherwise packets may be lost. This will result in offset data in the dataflash. A lock command will cancel a write stream. There is no reply to this packet.			

4.0.7 REQUEST: Read dataflash through CAN stream

Variable	*Bits*	*Type*	*Description*
Address in dataflash	63 .. 32	Uint32	Flat address in the external dataflash (540672 bytes)
Length of stream	31 .. 16	Uint16	Number of bytes expected on the DATAIN stream
Command String	15 .. 0	char[2]	“RD” stored as a string. msg[0] = 'R', ...
Opens a CAN stream from the dataflash.			

4.0.8 REPLY: Read dataflash through CAN stream

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 16	–	–
Error	15 .. 0	int16	0: Success -1: Stream already open -2: Address invalid
Only transmitted if there is an error condition with the request command.			

4.0.9 DATAIN: Data from Dataflash

Variable	*Bits*	*Type*	*Description*
Data	63 .. 0	–	Bytes of data from external dataflash
Packets will be sent at approximately 1kHz unless limited by baud rate.			

4.0.10 REQUEST: Active Motor

Variable	*Bits*	*Type*	*Description*
Motor Index	63 .. 48	–	Array Index (0 .. 9)
Command String	47 .. 0	char[6]	“ACTMOT” stored as a string. msg[0] = 'A', ...
There is no reply to this packet. The currently selected active motor can be seen in the status packet.			

4.0.11 REQUEST: Read-only Configuration Data

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 40	–	–
Command String	39 .. 0	char[5]	“ROCFG” stored as a string. msg[0] = 'R', ...
Reads read-only data stored about the controller.			

4.0.12 DATAIN: Read-only Configuration Data

Variable	*Bits*	*Type*	*Description*
Hard Current Limit	63 .. 32	float32	Hardware Over Current trip point
Hardware version	31 .. 16	Uint16	Version of the DSP / FPGA PCB in the WaveSculptor.
Code version	15 .. 0	Uint16	Software revision number. 123 = 1.23

4.0.13 REQUEST: Configuration start address

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 56	-	-
Command String	55 .. 0	char[7]	“CFGADDR” stored as a string. msg[0] = 'C', ...

4.0.14 REPLY: Configuration start address

Variable	*Bits*	*Type*	*Description*
Unused	63 .. 32	–	–
Config Start Address	31 .. 0	Uint32	Position of the configuration file in the dataflash.

5 CONFIGURATION FILE STRUCTURE

VARIABLE	*TYPE*	*DESCRIPTION*
File Header String	char[26]	TRITIUM CONFIGURATION FILE
Config File Version	Uint16	Divide by 100 to get version number.
Public Structure Size	Uint16	Size in bytes of the public structure
Private Structure Size	Uint16	Size in bytes of the private structure
Motor Structure Size	Uint16	Size in bytes of the motor structures
Public Structure	publicConfig	Contains user changeable controller information
Private Structure	privateConfig	Contains factory set configuration scale factors, offsets
Motor Structure	motorConfig[10]	Contains information about the configured motors.

If structures are to be sent as a byte for byte copy to the dataflash, they must not have any packing bytes between variables. This can usually be achieved in C with a #pragma pack style command.

5.0.1 Public Structure Definition

VARIABLE	*TYPE*	*UNIT*	*DESCRIPTION*
checksum	Uint16	–	16-bit checksum of all bytes in the structure except the checksum
activeMotor	Uint16	–	The motor that the controller currently thinks in connected
currLimit	float32	Arms	The upper limit on the current set point that can be set by the command packet
speedLimit	float32	m/s	The upper limit on the speed set point that can be set by the command packet
IdcLimit	float32	A	The upper limit on the bus current set point that can be set by the command packet
sinkOverTemp	float32	°C	The set point for the maximum heatsink temperature

VARIABLE	TYPE	UNIT	DESCRIPTION
busVoltMax	float32	V	The set point for the maximum bus voltage
busVoltMin	float32	V	The set point for the minimum bus voltage
wheelDiameter	float32	m	Diameter of the vehicle's wheels
CANbaud	Uint32	–	0:50kbps, 1:100kbps, 2:125kbps, 3:250kbps, 4:500kbps, 5:1000kbps.
CANbaseAddr	Uint32	–	0x000 – 0x7D0
dcCANAddr	Uint32	–	0x000 – 0x7D0
vehicleMass	float32	kg	Used to derive the speed loop constants
rotatingMass	float32	kg	Used to derive the speed loop constants
telemSendFlags	Uint32	–	Bit position corresponds with the broadcast message ID offsets.

5.0.2 Calibrate Structure Definition

The private and motor configuration structures make use of the calibrate and logFitConst structures, which are defined as follows.

VARIABLE	TYPE	UNIT	DESCRIPTION
scale	float32	–	Application of scale should make measurement equal to 1 at base value.
offset	float32	–	Applied prior to scale to take out offset error in sensor.

5.0.3 LogFitConst Structure Definition

VARIABLE	TYPE	UNIT	DESCRIPTION
A	float32	–	Constants can be found in thermistor datasheets or applications notes. $T(R)=\left(A+B\ln\frac{R}{R_{ref}}+C\ln^2\frac{R}{R_{ref}}+D\ln^3\frac{R}{R_{ref}}\right)^{-1}$
B	float32	–	
C	float32	–	
D	float32	–	

5.0.4 Private Structure Definition

The private structure is used only by Tritium and any attempt to write to its dataflash location will return an invalid address error.

5.0.5 Motor Structure Definition

VARIABLE	TYPE	UNIT	DESCRIPTION
checksum	Uint16	–	16-bit checksum of all bytes in the structure except the checksum
nrPolePair	Uint16	–	The number of pole-pairs in the motor.
lineR	float32	Ohms	The phase-neutral resistance.
lineL	float32	H	The phase-neutral inductance.
speedConst	float32	$\frac{V_{rms}}{(rads/s)}$	The RMS phase-neutral voltage induced at 1 rad / s.
phaseSeqRev	Uint16	T/F	The sequence of the phases while the motor rolls forward. That is does A come before B or vice-versa.

VARIABLE	*TYPE*	*UNIT*	*DESCRIPTION*
motorOverTemp	float32	°C	The set point for the maximum motor temperature. Not implemented due to concerns about reliability of signal.
motorTemp	calibrate	–	Offsets and scales the voltage reading from the VtoF converter prior to applying the logFitConsts
motorTempLFC	logFitConst	–	Log constants of the thermistor used in the motor
motorDescription	char[20]		User description of the motor
hallAngle	float32[8]	–	The motor angle at the transition into the hall state given by the index.
priorHall	Uint16[8]	–	The prior hall state prior to the hall state given by the index.

The shaded constants can and should be set using the paramExtract and phasorSense systems included with the Windows interface program.

6 REVISION RECORD

REV	*DATE*	*CHANGE*
1	06 July 2006	Document creation. (PCS)
2	27 February 2007	Added programming specifications. (DAF) Added odometer & Ah totaliser message type. (DAF) Modified controller ID string in ID Info packet. (DAF) Clarified driver controls commands. (JMK)
3	2 March 2007	Formatting & typos (JMK)
4	14 March 2007	Added 15V error bit to Status Information (DAF)
5	20 August 2007	Fixed the units for Odometer and AmpHour counter (DAF)