CAN Expansion Module V1

Document Number		PR035_CANExpansionModuleV1_12JAN2023	
Title		CAN Expansion Module V1	
Team Member / Engineer		Raheel Farouk	
Creation Date		12 JAN 2023	
Prepared By	**Date**	**Checked By**	**Change History**
Raheel Farouk	12JAN2023	-	Initial Version
Raheel Farouk	16JAN2023	-	Added Endianness and Cleaned up formatting
Raheel Farouk	06MAR2023	-	Removed slip angle sensor and added Aux2
Patrick Murphy	09MAR2023	-	Added Descriptions of data and Setup information
Patrick Murphy	21APR2023	-	Added Changes Relating to Library change, data frame update and connections.
Patrick Murphy	24MAY2023	-	Updated to Reflect Canduino Deprecation. Added custom PCB descriptions. Reworked Sections.

Purpose of Module

The purpose of the CAN Expansion Module is to convert desired analog sensor inputs into digital data on the CAN bus.

Setup Guide

Setting up the Expansion Module requires a CAN ID for the expansion module and the sensors to be plugged into their corresponding spots on the module. The MoTeC firmware will decide between what input will be used for each function, for example brake temperature will select the brake temperature input but you will be able to choose between front and rear, left and right.

Board Configuration

The initial version of the board was using the Canduino V3.1 which has the CS as Pin 10, the V3.0 is Pin 8, the custom PCB is 10. The Board Programming library for the Canduino's and the Custom PCB.

Custom PCB – V1

CS for SPI to CAN - 10. Programmed as Arduino Nano.

A custom PCB designed around the ATMEGA328P (16 MHz) with an MCP25625 (8 MHz) SPI to CAN controller at a 5v logic level. Includes 4 ADC (10 bit) pins with independent power and ground wires to reduce common wiring failure points for each sensor. The PCB has a selection bridge for enabling a 120 ohm CAN termination resistor if needed.

Featuring a serial programming header for debugging and an ICSP interface for programming.

https://github.com/Panther-Racing/Expansion-Module-V1

Canduino V3.1 (Deprecated)

CS for SPI to CAN (10). Programmed as ATMEGA328P using Board Library.

The Canduino V3.1 was an atmega328P with an attached MCP2515 and MCP2551 to allow communication onto a CAN network.

The Canduino possessed 8 ADCs, a 3v and 5v output, SPI interface and 10 Digital outputs. There is the option of having a CAN termination resistor through a solder bridge aswell as a chip select through a solder jumper.

More information on this can be found in the following locations.

This board has entered deprecated status from the manufacturer.

https://www.tindie.com/products/massivegroup/canduino-v31-atmega328p-with-can-bus/

https://github.com/MassiveButDynamic/canduino-v3.1

Can Library Configuration

The library used to communicate with the MCP2515 as recommended by the Canduino is the Arduino-mcp2515 library. Previously the MCP_CAN library by coryjfowler was used however issues were encountered relating to significant transmit errors on the bus so this is no longer used.

Analog Sensor Wiring

Sensor Name	**PIN**	**Colors**
Brake Temperature	ADC 0	White - Yellow
DPOS	ADC 1	White – Brown
AUX 1	ADC 2	White - Green
AUX 2	ADC 3	White – Black

Messaging Structure

SPI Message structure

Endianness	Big Endian	
Message Length	8	
Byte	**Contents***	
Byte0	Brake Temp (9-2)	
Byte1	Brake Temp (1-0)	DPOS (9-4)
Byte2	DPOS (3-0)	Aux 1 (9-6)

Byte3	Aux 1 (5-0)	Aux 2 (9-8)
Byte4	Aux 2 (7-0)	
Byte5	Firmware Version	
Byte6	Transmit Error Counter (TEC)	
Byte7	Counter	

*Note: The numbers in brackets are the bit numbers of the data stored in the byte. For example if Brake temp is a 10bit value, then the most significant first 8 bits of brake temp, Bits 9 to 2 will be stored in byte 1, then the remaining 2 bits will be stored in byte 1.

Can Message Structure of Data

Base CAN ID:	0x760 ; +1 ; +2 ; +3	
	FL – 0x760	FR – 0x761
	RL - 0x762	RR – 0x763

Data Name*	**Length (Bits)**	**Position in Message**	
		Start	**End**
Brake Temp	10	0	9
DPOS	10	10	19
Aux 1	10	20	29
Aux 2	10	30	39
Firmware Version	8	40	47
Transmit Error Counter (TEC)	8	48	55
Counter	4	56	59

*Note: that the Data will effectively be sent as unsigned integers with values in the range of 0 to 2^(length of bits). Any conversion of this information currently needs to be performed on the receiving end of the data.

Additional Notes

Auxiliary sensor attached to things like steering position.

Description of Firmware Version

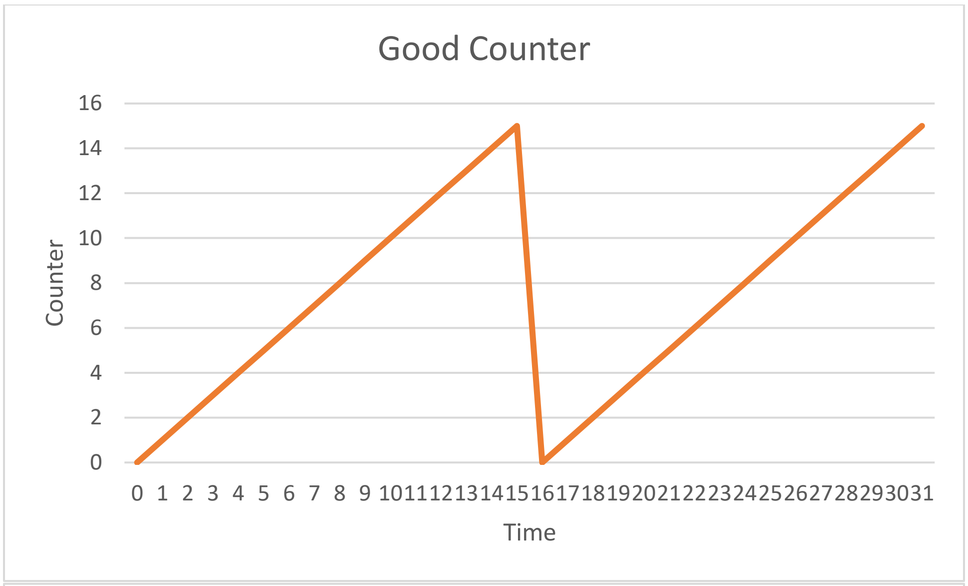
The firmware version corresponds to the current version of the Module. This is represented in format the format of vXX.X. In the software this is programmed as a uint8_t type. To achieve the decimal position the firmware version must be 100 times the current version. For example if the current version of the module is v0.1 then the firmware version shall be 100*0.1 or 100.

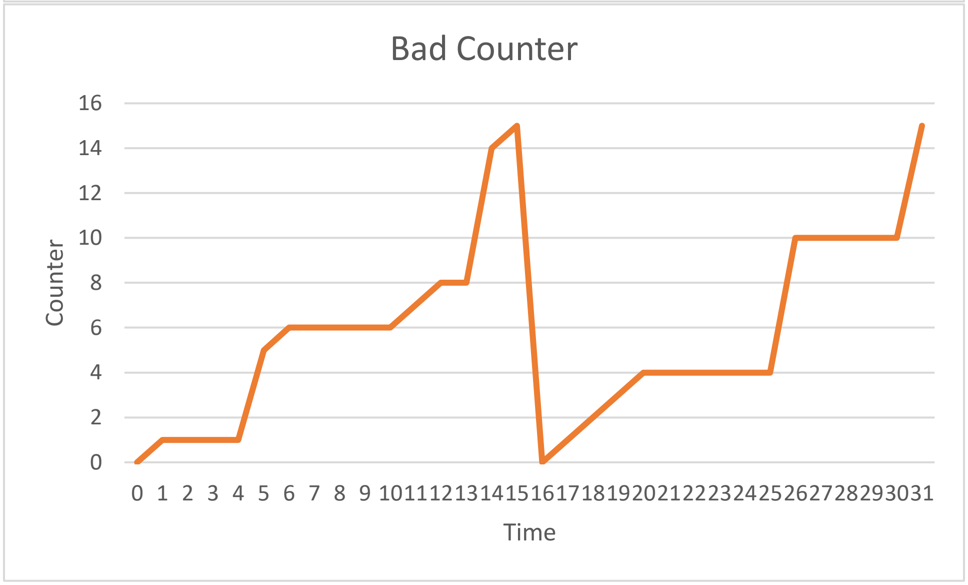
Description of Compound

Compound is unimplemented. To be used if the length of data the module transmits exceeds 8 bytes of information to allow more data extraction from the same CAN ID. On the MOTEC side of this program the compound ID is actually the TEC from the board.

Description of Counter

Counter is a 4bit value used to determine if loss of transmission occurs. If observing transmission from the board this value shall go from 0 to 15 and rollover, if any value is missed while this is occurring then a loss of transmission has occurred and should be investigated.





ADC

For the ATMEGA328P there are 6 ADC pins which are multiplexed to a single ADC at up to a 10bit accuracy.

Transmit Error Counter (TEC)

The TEC is read from the mcp2515 / 25625 registers using SPI. This describes the number of times that the controller has been broadcasting to the CAN bus but has detected that a value it sent was not properly sent. This can be seen using MOTEC via the "Compound ID".