Motorboard: Code documentation

Table of Contents

Disclaimer

This document might be faulty in terms of both its analysis and its included references in regards to the code. To avoid an erroneous understanding of the aforementioned it is recommended to RTFC.

Adc.h

Contains functions for initializing and reading from the Analog to Digital Converter (ADC).

void adc_init(void)

Description:	Initializes the Analog to Digital Converter, with the following attributes: - Prescaler set to 128 - Voltage reference 5V
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

uint16_t adc_read(uint8_t channel)

Description:	Reads a value via the ADC from a specified channel, which corresponds to a certain pin. If the channel read is the one designated to reading battery voltage the value is stored
Inputs:	8 bits, specifying the channel/pin
Outputs:	16 bits, value returned from the ADC
External libraries:	N/A

uint16_t getPreviousAdc4Value(void);

Description:	Returns the previously read value from the designated battery voltage which is channel 4 for the Dynamo
Inputs:	N/A
Outputs:	16 bits, previously read battery voltage
External libraries:	N/A

uint16_t getPreviousAdc8Value(void);

Description:	Returns the previously read value from the designated battery voltage which is channel 8 for the Innovator
Inputs:	N/A
Outputs:	16 bits, previously read battery voltage
External libraries:	N/A

Bluetooth.h

Contains functions for initializing, reading, and checking the status of the Bluetooth module (Adafruit Bluefruit LE UART Friend).

void blue_init(void)

Description:	Initializes the Bluetooth module @ UART0
Inputs:	N/A
Outputs:	N/A
External libraries:	Uart.h: uart0_init(BAUD, F_CPU)

void blue_putc(unsigned char data)

Description:	Transmits a character via UART0
Inputs:	8 bit, desired character to be send
Outputs:	N/A
External libraries:	Uart.h: uart0_putc(data)

void blue_puts(char * s)

Description:	Transmits a string/an array of characters via UART0
Inputs:	8 bit array, desired string/characters to be transmitted
Outputs:	N/A
External libraries:	Uart.h: uart0_puts(s)

uint16_t blue_available()

Description:	Determines the number of bytes waiting in the receive buffer related to the Bluetooth module
Inputs:	N/A
Outputs:	N/A
External libraries:	Uart.h: uart0_available()

void blue_flush_buffer()

Description:	Flushes/ignores the bytes waiting in the receive buffer related to the Bluetooth module
Inputs:	N/A
Outputs:	N/A
External libraries:	Uart.h: uart0_flush()

unsigned char blue_read_data()

Description:	Returns byte from the Bluetooth buffer if no errors are detected, i.e. stop bit error, over run error, and overflow error. If the buffer is empty or data is not available, it returns 0
Inputs:	N/A
Outputs:	8 bit, character from the buffer
External libraries:	Uart.h: uart0_getc()

void blue_read_buffer(void * buff, uint16_t len)

Description:	Loads data into an array used as a buffer via blue_read_data()
Inputs:	8 bit array, buffer; 16 bit, length of buffer
Outputs:	N/A
External libraries:	Uart.h: uart0_flush()

void blue_set_new_device()

Description:	Gives a Bluetooth module a new name, e.g. "DTU Dynamo", the string must be set in the method itself
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

Global.h

Contains global variables used throughout the code extensively, i.e. declarations for buffer sizes used for communication and the global up-time (how long the board has been running since last reset). A quick recap of the buffers contained within:

1. Motor <-> RIO (<-> counts for two buffers, one each way)
2. Motor <-> Steering
3. Motor <-> Front lights
4. Motor <-> Back lights

The header file also contains simple "Helping functions", such as:

- TESTBIT(var, bit) (var & (1<<bit))
- SETBIT(var, bit) (var |= (1<<bit))
- CLRBIT(var, bit) (var &= ~(1<<bit))
- FLIPBIT(var, bit) (var ^= (1<<bit))

void global_init(void)

Description:	Performs any initializations needed in regards to the global variables. *This method is currently empty!*
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

Inout.h

Contains methods dealing with the inputs and outputs start motor, horn, gear sensor, and wheel sensor. *N.B. The magicWheelConstant is some Henning magic.*

void io_init(void)

Description:	Initiates/declares the designated pins for starter motor and horn as output and gear sensor as an input
Inputs:	N/A
Outputs:	N/A
External libraries:	Global.h: SETBIT(var, bit), CLRBIT(var, bit)

void wheel_sensor_init(uint8_t numberOfMagnetsPerRevolution, uint32_t mMagicWheelConstant)

Description:	Initiates the wheel sensor, toggles an LED, and sets the magicWheelConstant
Inputs:	8 bit, number of magnets pr. revolution regarding the wheel sensor; 32 bit for the magic wheel constant
Outputs:	N/A
External libraries:	Global.h: CLRBIT(var, bit)

void setHornHigh(void)

Description:	Turns the horn on
Inputs:	N/A
Outputs:	N/A
External libraries:	Global.h: SETBIT(var, bit)

void setHornLow(void)

Description:	Turns the horn off
Inputs:	N/A
Outputs:	N/A
External libraries:	Global.h: CLRBIT(var, bit)

void setStarterHigh(void)

Description:	Turns the starter motor on
Inputs:	N/A
Outputs:	N/A
External libraries:	Global.h: SETBIT(var, bit)

void setStarterLow(void)

Description:	Turns the starter motor off
Inputs:	N/A
Outputs:	N/A
External libraries:	Global.h: CLRBIT(var, bit)

uint8_t digitalReadGSensor(void)

Description:	Reads the gear sensor value
Inputs:	N/A
Outputs:	8 bit, digital value of the gear sensor
External libraries:	Global.h: TESTBIT(var, bit)

uint16_t getWheelSensorPeriod(void)

Description:	Returns the period of time passed since the last wheel sensor read
Inputs:	N/A
Outputs:	16 bits, time in ms
External libraries:	Global.h: variable time

uint32_t getDistanceCompleted(void)

Description:	Returns the total distance completed
Inputs:	N/A
Outputs:	32 bits, distance in meters
External libraries:	N/A

void SetPWMDutyGear(uint16_t duty);

Description:	Sets the duty cycle of the PWM designated to the gear sensor
Inputs:	16 bit, desired duty cycle of the PWM
Outputs:	N/A
External libraries:	N/A

void SetPWMDutySpeed(uint16_t duty);

Description:	Sets the duty cycle of the PWM designated to the speed servo. *Currently not in use!*
Inputs:	16 bit, desired duty cycle of the PWM
Outputs:	N/A
External libraries:	N/A

void pwm_init(void);

Description:	Initializes the PWM Output for the gear sensor and speed servo. *N.B. The speed servo's initialization is currently out commented!*
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

LEDs.h

Contains methods for initializing, turning on/off, and toggling the various LEDs.

void LED_init(void)

Description:	Initiates/declares the designated pins for the various LEDs used
Inputs:	N/A
Outputs:	N/A
External libraries:	Global.h: SETBIT(var, bit)

void LEDnOn(void)

Description:	Turns on the n'th LED
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void LEDnOff(void)

Description:	Turns off the n'th LED
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void LED1Toggle(void)

Description:	Toggles the n'th LED
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void LEDBlink(int LED, int times)

Description:	Blinks an LED as many times as the input "times" dependent on the "LED" input. Provides an easy way to test all the LED's
Inputs:	32 bits, LED number; 32 bits, desired times that the LED/LED's should blink
Outputs:	N/A
External libraries:	N/A

Rs232.h

Contains functions for initializing, reading, and checking the status of the Reconfigurable Input/Output (RIO) module. This library contains inline methods which are used extensively throughout the library. *Be aware there is inconsistency between the c file and h file (rs232_ wait... in c file, rs485_ wait... in h file)!*

void rs232_init(void)

Description:	Initiates the communication with the RIO module
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void rs232_putc(unsigned char data)

Description:	Transmits a character via UART2 to the RIO
Inputs:	8 bits, desired data to be transmitted
Outputs:	N/A
External libraries:	N/A

void rs232_wait_transmit(void)

Description:	Used to stall the main loop until transmission from the RIO is complete or has timed out
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

uint16_t rs232_available(void)

Description:	Determines the number of bytes waiting in the receive buffer related to the Bluetooth module
Inputs:	N/A
Outputs:	N/A
External libraries:	Uart.h: uart2_available()

Rs232sync.h

The "big brother" of rs232.h, contains interrupt handler for receiving data from the RIO, as well as a method to transmit larger chunks of data to the RIO. The interrupt handler is not included in this document.

void rs232_set_car(uint8_t carId)

Description:	Sets the carID which determines the way the interrupt handler handles receiving data from the RIO. The carID only needs to be set if we are dealing with the innovator, which is out of use
Inputs:	8 bits, specified carID
Outputs:	N/A
External libraries:	N/A

void rs232_tx(void)

Description:	Transmits data to the RIO after syncing the two nodes. The data and length thereof is determined outside of the method
Inputs:	N/A
Outputs:	N/A
External libraries:	Rs232.h: rs232_putc(unsigned char data)

Rs485.h

Contains methods for initializing, transmitting, and stall the UART communication via the rs485 protocol.

void rs485_init(void)

Description:	Initializes the module to be ready for UART communication. The module starts off in receive mode
Inputs:	N/A
Outputs:	N/A
External libraries:	Uart.h: uart1_init(uint16_t baudrate)

void rs485_set_tx_mode(void)

Description:	Sets the module in transmission mode
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void rs485_set_rx_mode(void)

Description:	Sets the module in receive mode
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void rs485_putc(unsigned char data)

Description:	Transmits a character via UART1
Inputs:	8 bits, desired data to be transmitted
Outputs:	N/A
External libraries:	Uart.h: uart1_putc(uint8_t data)

void rs485_wait_transmit(void)

Description:	Used to stall the main loop until the transmission has been completed
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

Rs485sync.h

The big brother of rs485.h, contains the structure that synchronises all the communication from and to the motorboard. It also contains the interrupt handler for receiving data automatically. The interrupt handler is not included in this document.

void rs485_sync(void)

Description:	Handles all the synchronization needed for transmitting and receiving data from and to the motorboard. In broad strokes, it does the following sequentially: 1. Reads data from the steering and light (front) board 2. Reacts to said data read, e.g. sets bits needed to blink left, starts wipers, etc. 3. Synchronises data with the RIO 4. Sends data to the steering, front light, and back light board
Inputs:	N/A
Outputs:	N/A
External libraries:	Too extensive to include

Tunes.h

Contains the methods and the definitions used to make the buzzer play songs.

void tunes_init(void)

Description:	Initiates output pin used to control the buzzer
Inputs:	N/A
Outputs:	N/A
External libraries:	N/A

void sing(int s)

Description:	Makes the buzzer play a tune
Inputs:	16 bits, value of the desired tune to be played
Outputs:	N/A
External libraries:	Delay.h: delay_ms(uint16_t count)

void buzz(int targetPin, long frequency, long length)

Description:	Makes the buzzer play a tone for a certain duration. *Cation: The targetPin argument is not being used as of right now. Instead the output pin is hardcoded in the method!*
Inputs:	16 bits, specifies the output pin; 32 bits, specifies the frequency to be buzzed; 32 bits, specifies the duration of the note
Outputs:	N/A
External libraries:	Delay.h: delay_us(uint16_t count)